Sistemas de equações lineares

Joaquim João Júdice
João Manuel Patrício

Departamento de Matemática da Universidade de Coimbra
1996

Conteúdo

Introdução

Os sistemas de equações lineares surgem em praticamente todas as áreas da matemática aplicada. O grande desenvolvimento dos meios de cálculo automático a que se tem assistido ultimamente tem possibilitado resolver sistemas com cada vez maior número de variáveis e de equações. Este livro apresenta as técnicas mais conhecidas para a resolução destes problemas e debruça-se sobre as questões teóricas e práticas que lhes estão subjacentes.

Este trabalho é também uma compilação das aulas por nós leccionadas na disciplina de Álgebra Linear Numérica do terceiro ano da Licenciatura em Matemática do Departamento de Matemática da Universidade de Coimbra. Nesse sentido procurámos apresentar as várias técnicas para a resolução de sistemas de equações lineares de um modo detalhado para que não surjam quaisquer dúvidas sobre o seu funcionamento. Além disso tivémos especial cuidado em apresentar as implementações desses processos para o caso de sistemas com grande número de variáveis e equações. A escolha de uma técnica para a resolução de um dado sistema depende da ordem, esparsidade, estrutura e classe da sua matriz. A importância de todos esses aspectos é devidamente realçada neste livro, sendo apresentadas em muitos casos conclusões seguras sobre o algoritmo a escolher para a resolução do sistema em causa. Dado o carácter didáctico deste livro resolvemos omitir alguns resultados e demonstrações mais técnicas. Além disso muitos processos importantes para a resolução de sistemas foram simplesmente omitidos, por total indisponibilidade de tempo. No entanto esses assuntos são discutidos em alguns dos livros mencionados na Bibliografia.

Apesar de termos alguma preocupação em apresentar os assuntos de um modo um pouco diferente do habitual, aproveitámos sempre que possível ideias, resultados e demonstrações de outros autores. Isso aconteceu nomeadamente nos exemplos que constituem o primeiro capítulo do livro, que permitem verificar a importância da ordem, esparsidade e estrutura da matriz na resolução do sistema.

Nos capítulos 2 e 3 são discutidos alguns conceitos de álgebra linear que estão normalmente associados à resolução de sistemas de equações lineares. Assim, são apresentadas as definições e propriedades de normas vectoriais e matriciais, número de condição, complemento de Schur e algumas classes de matrizes.

O capítulo 4 é dedicado aos chamados métodos directos para sistemas de equações lineares. Esses processos procuram obter a solução exacta do sistema usando decomposições da sua matriz em matrizes triangulares ou diagonais. As decomposições LU e LDU são discutidas com bastante detalhe tanto do ponto de vista teórico como prático e computacional, sendo dado especial ênfase às várias formas de as obter, tanto no caso não simétrico como simétrico. A estabilidade dessas decomposições é estudada com bastante cuidado, sendo discutidas a técnica de escolha parcial de pivot e as classes de matrizes com pivots diagonais estáveis. Ainda relacionados com a precisão da solução do sistema, são descritos os processos de escalonamento, de refinamento iterativo e as estimativas do número de condição de uma matriz. Finalmente os problemas de determinação da matriz inversa e do cálculo do determinante de uma matriz são abordados na última secção deste capítulo.

Os capítulos 5 e 7 debruçam-se sobre o uso de métodos directos na resolução de sistemas de equações lineares com matrizes esparsas. Assim, as matrizes com estrutura de banda e bloco são discutidas no capítulo 5, sendo as matrizes sem estrutura especial tratadas no capítulo 7. A implementação dos métodos directos para esse tipo de matrizes está intimamente relacionada com as estruturas de dados que as armazenam. Esse assunto é discutido no capítulo 6, conjuntamente

com as operações matriciais e vectoriais com matrizes esparsas assim armazenadas.

Contrariamente aos métodos directos, os algoritmos iterativos para sistemas de equações lineares geram uma sucessão de vectores cujo limite ou ponto de acumulação é uma sua solução. No capítulo 8 são discutidos alguns dos principais métodos iterativos, nomeadamente os algoritmos básicos (Jacobi, Gauss-Seidel, SOR e partição) e o método dos gradientes conjugados. A convergência desses processos e a chamada técnica de precondicionamento são ainda analisadas com bastante detalhe neste capítulo.

Dadas as suas excelentes propriedades numéricas, as matrizes ortogonais têm merecido um estudo atento dos investigadores e utilizadores de álgebra linear numérica. A definição e algumas propriedades dessas matrizes são apresentadas no capítulo 9, conjuntamente com o processo de decomposição QR usando matrizes de Householder e de Givens. O uso dessa decomposição na resolução de sistemas lineares é ainda analisado nesse capítulo.

O capítulo 10 debruça-se sobre a resolução de sistemas de equações lineares com matrizes rectangulares de característica completa. Distinguimos sistemas horizontais e verticais, dependendo do número de equações ser inferior ou superior ao número de variáveis respectivamente. As várias técnicas para a resolução deste tipo de sistema são discutidas neste capítulo, sendo dadas indicações seguras sobre as suas vantagens ou desvantagens.

Em cada um dos capítulos houve sempre a preocupação de apresentar alguns exercícios, que certamente ajudarão o aluno a consolidar os seus conhecimentos dos vários assuntos tratados neste livro. Como conclusão final pensamos ter conseguido elaborar um livro que se torne um bom auxiliar de um Professor da disciplina de Álgebra Linear Numérica. O rigor, a clareza e o detalhe da exposição permitirão certamente a esse Professor tratar alguns assuntos de um modo sumário e assim abordar todos os tópicos subjacentes à resolução de sistemas de equações lineares. Essa foi a nossa grande intenção ao escrever este trabalho.

Os Autores

Notação

letras minúsculas - números reais ou complexos, vectores e funções.

letras maiúsculas - conjuntos e matrizes.

$|x|$ **-** módulo do número x.

$|K|$ **-** número de elementos do conjunto K.

ϵ_M **-** precisão da máquina.

$||\cdot||$ **-** norma vectorial ou matricial.

x **(**x^T**) -** vector coluna (linha).

$x^{(k)}$**,** x_k **-** valor do vector x na iteração k.

$A_{i.}$ **-** linha i de A.

$A_{.j}$ **-** coluna j de A.

x_J **-** subvector de x constituído pelas componentes de índices $i \in J$.

A_{JK} **-** submatriz de A correspondente às linhas $i \in J$ e colunas $j \in K$.

P_{ij} **-** matriz de permutação.

$(A|A_{JJ})$ **-** complemento de Schur de A_{JJ} em A.

A^T **-** transposta de A.

A^{-1} **-** inversa de A.

$c(A)$ **-** característica de A.

$det(A)$ **-** determinante de A.

$\rho(A)$ **-** raio espectral de A.

$\lambda_{\max}(A)$ **(**$\lambda_{\min}(A)$**) -** valor próprio de A de maior (menor) valor absoluto.

BAND(A**) -** banda de A.

$\beta(A)$ **(**$\alpha(A)$**) -** comprimento de banda (superior) de A.

ENV(A**) -** invólucro de A.

DIAG(A**),** $diag(A)$ **-** diagonal de A.

FILL(A**) -** conjunto dos enchimentos de A.

$cond(A)$ **-** número de condição de A.

g_A - factor de crescimento de A.

$M(A)$ - matriz companheira de A.

$|A|$ (A^p) - matriz cujos elementos são os valores absolutos (potências de expoente p) dos elementos de A.

$diag(v_1, \ldots, v_n)$ - matriz diagonal de elementos diagonais $v_1, \ldots, v_n$.

L (U) - matriz triangular inferior (superior).

D - matriz diagonal.

Q (R) - matriz ortogonal (triangular superior).

DD (RDD, CDD) - matriz diagonalmente dominante (por linhas, por colunas).

SDD (SRDD, SCDD) - matriz diagonalmente dominante estrita (por linhas, por colunas).

P, P_0, Z, K, S, H - classes de matrizes.

T_+ - matriz da classe T com elementos diagonais positivos.

PD (PSD, ND, NSD, IND) - matriz positiva definida (positiva semi-definida, negativa definida, negativa semi-definida, indefinida).

SPD (SPSD, SND, SNSD, SIND) - matriz simétrica PD (PSD, ND, NSD, IND).

Capítulo 1

Exemplos de Sistemas de Equações Lineares

Neste capítulo iremos apresentar três exemplos de sistemas de equações lineares que aparecem em três aplicações importantes da matemática, física e engenharia. Nos dois primeiros exemplos tivémos a preocupação de discutir dois casos em que as matrizes dos sistemas são quadradas de ordens elevadas e muito esparsas, isto é, contêm um elevado número de elementos nulos. Como iremos verificar em capítulos posteriores, a resolução de sistemas com matrizes esparsas será o assunto fundamental deste trabalho. Finalmente, o último caso mostra a importância da resolução de sistemas de equações lineares com matrizes rectangulares.

Resolução numérica da Equação de Laplace

Nesta secção iremos estudar a resolução de uma equação de derivadas parciais elípticas com condições de fronteira, isto é, uma equação da forma

$$\frac{\partial^2 u}{\partial x^2}(x,y) + \frac{\partial^2 u}{\partial y^2}(x,y) = 0 \tag{1.1}$$

com $u(x,y)$ uma função definida num conjunto $\Omega \subset \mathbb{R}^2$ com fronteira Γ satisfazendo

$$u(x,y) = g(x,y) \text{ para todo o } (x,y) \in \Gamma$$

Esta equação, habitualmente designada Equação de Laplace, apresenta grandes aplicações em alguns problemas de Física-Matemática, tais como a determinação da distribuição de temperatura numa superfície Ω conhecida a temperatura na sua fronteira Γ.

É possível provar que esta equação tem uma e uma só solução, desde que a função satisfaça determinadas condições [Burden *et al.*, cap. 11]. No entanto, essa solução é em geral difícil de encontrar, mesmo nos casos em que a região Ω tem uma forma relativamente simples, como um triângulo ou uma circunferência. Para isso é necessário desenvolver um método numérico que permita obter uma aproximação da solução. Seguidamente apresentaremos uma estratégia para o caso em que a região Ω é um rectângulo.

Consideremos então os intervalos $[a,b], [c,d] \subset \mathbb{R}$ e o rectângulo

$$\Omega = [a,b] \times [c,d]$$

de fronteira Γ onde pretendemos resolver o problema. Sejam $D = (a,c)$, $E = (b,c)$, $F = (b,d)$ e $G = (a,d)$ os vértices do rectângulo.

A técnica que iremos apresentar consiste na obtenção de valores aproximados da solução do problema em pontos convenientemente escolhidos, formando uma rede em Ω. Com o objectivo de

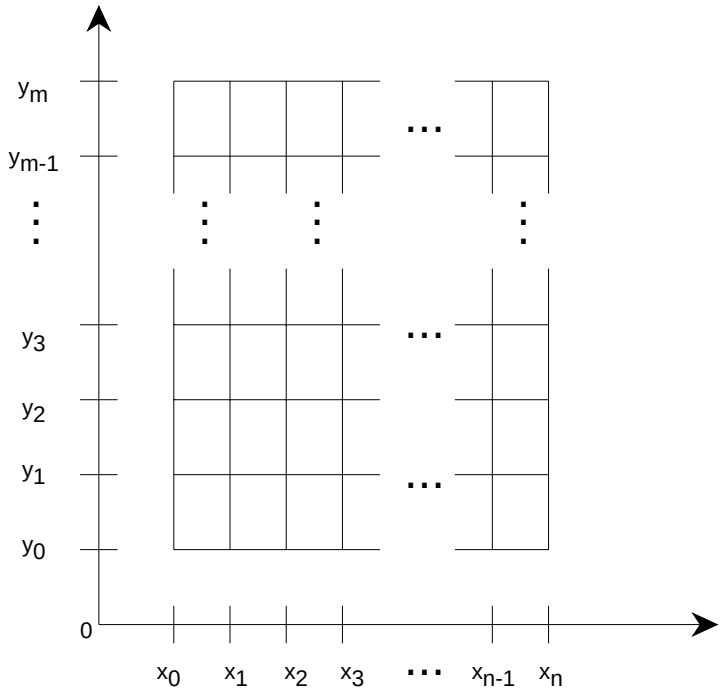


Figura 1.1: A rede onde vai ser resolvida a equação de Laplace

construir essa rede, consideremos $n+1$ pontos

$$a \equiv x_0 < x_1 < x_2 < \ldots < x_n \equiv b$$

de $[a, b]$ que dividem o intervalos em n subintervalos de igual amplitude e $m+1$ pontos

$$c \equiv y_0 < y_1 < y_2 < \ldots < y_m \equiv d$$

que dividem $[c, d]$ em m intervalos com a mesma amplitude. Consideremos os conjuntos

$$R_{n+1} = \{x_i : \ x_i = a + ih, \ i = 0, \ldots, n\}$$

com $h = (b-a)/n$ e

$$R_{m+1} = \{y_j : \ y_j = c + jk, \ j = 0, \ldots, m\}$$

com $k = (d-c)/m$. A rede em questão será então o conjunto $R_{n+1,m+1} = R_{n+1} \times R_{m+1}$, cuja representação aparece na figura 1.1.

Para resolver o problema iremos usar o método das diferenças finitas. Esse processo consiste em encontrar uma solução discreta $u = (u_{ij})_{\substack{i=1,\ldots,n \\ j=1,\ldots,m}}$ da equação de Laplace definida de modo que $u_{ij} = u(x_i, y_j)$, para $(x_i, y_j) \in R_{n+1,m+1}$.

Para cada ponto da rede tem-se

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) + \frac{\partial^2 u}{\partial y^2}(x_i, y_j) = 0 \tag{1.2}$$

É possível aproximar as derivadas parciais de segunda ordem por diferenças divididas de se-

gunda ordem [Burden *et al.*, cap. 11]. Assim, tem-se

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2}(x_i, y_j) &\approx \frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2} \\ \frac{\partial^2 u}{\partial y^2}(x_i, y_j) &\approx \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{k^2} \end{aligned} \tag{1.3}$$

Substituindo em (1.2) obtemos

$$\frac{u_{i+1,j} - 2u_{ij} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{ij} + u_{i,j-1}}{k^2} = 0 \tag{1.4}$$

com $i = 1, \ldots, n-1$ e $j = 1, \ldots, m-1$. As igualdades (1.4) correspondem a um sistema linear com $(n-1)(m-1)$ equações e $(n-1)(m-1)$ incógnitas. Note-se que u_{0j}, u_{nj}, u_{i0} e u_{im} são dados pelas condições de fronteira. Resolvido este sistema linear, obtemos os valores u_{ij} que constituem a solução da equação de Laplace associada à rede construída.

Para resolver o sistema (1.4), é conveniente considerar uma numeração eficiente para os pontos (x_i, y_j). Nesse sentido vamos considerar a seguinte correspondência

$$u_{ij} = u(x_i, y_j) \leftrightarrow p_t,$$

com

$$t = i + (m - 1 - j)(n - 1), \quad i = 1, \ldots, m-1, \quad j = 1, \ldots, n-1$$

Esta ordenação consiste em considerar os pontos (x_i, y_j) ordenados da esquerda para a direita e de cima para baixo. Usando esta ordenação, o sistema (1.4) tem a forma

$$\begin{bmatrix} B & -I & 0 & \ldots & 0 & 0 & 0 \\ -I & B & -I & \ldots & 0 & 0 & 0 \\ 0 & -I & B & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & B & -I & 0 \\ 0 & 0 & 0 & \ldots & -I & B & -I \\ 0 & 0 & 0 & \ldots & 0 & -I & B \end{bmatrix} \begin{bmatrix} p^{(1)} \\ p^{(2)} \\ p^{(3)} \\ \vdots \\ p^{(n-3)} \\ p^{(n-2)} \\ p^{(n-1)} \end{bmatrix} = \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ b^{(3)} \\ \vdots \\ b^{(n-3)} \\ b^{(n-2)} \\ b^{(n-1)} \end{bmatrix}$$

onde $p^{(i)} \in \mathbb{R}^{m-1}$, $b^{(i)} \in \mathbb{R}^{m-1}$ e $B \in \mathbb{R}^{(m-1)\times(m-1)}$ é a matriz definida por

$$B = \begin{bmatrix} \alpha & -\theta & 0 & \ldots & 0 & 0 \\ -\theta & \alpha & -\theta & \ldots & 0 & 0 \\ 0 & -\theta & \alpha & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \alpha & -\theta \\ 0 & 0 & 0 & \ldots & -\theta & \alpha \end{bmatrix}$$

com $\theta = h^2/k^2$ e $\alpha = 2(\theta + 1)$, I é a matriz identidade de ordem $(m-1)$ e o vector de termos independentes $b = [b^{(1)}, b^{(2)}, \ldots, b^{(n-1)}]^T$ é dado por

$$b^{(1)} = \begin{bmatrix} u_{01} + \theta u_{1m} \\ u_{02} \\ u_{03} \\ \vdots \\ u_{0,m-2} \\ u_{0,m-1} + \theta u_{1m} \end{bmatrix}, \quad b^{(n-1)} = \begin{bmatrix} u_{n1} + \theta u_{n-1,0} \\ u_{n2} \\ u_{n3} \\ \vdots \\ u_{n,m-2} \\ u_{n,m-1} + \theta u_{n-1,m} \end{bmatrix}$$

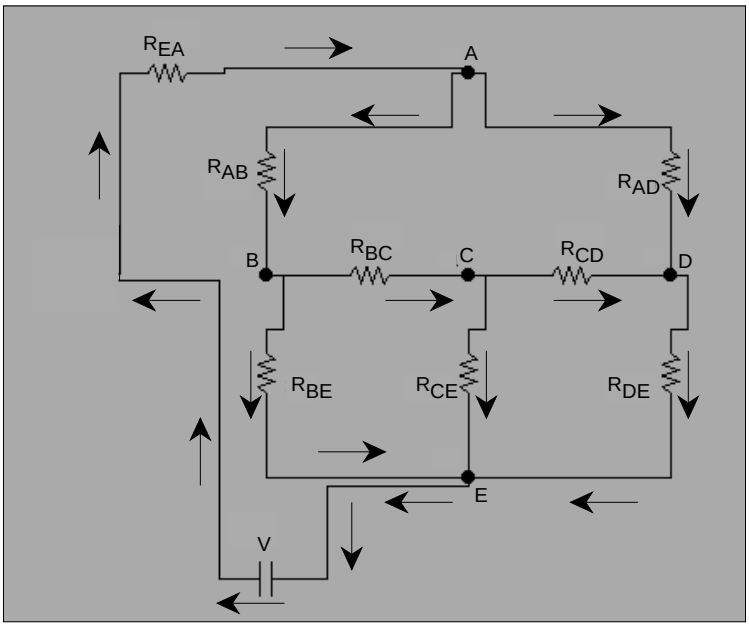


Figura 1.2: Exemplo de um circuito eléctrico

e

$$b^{(i)} = \begin{bmatrix} \theta u_{i0} \\ 0 \\ 0 \\ \vdots \\ 0 \\ \theta u_{im} \end{bmatrix}, \quad i = 2, \ldots, n-2$$

Como no método das diferenças finitas apenas se obtém uma aproximação discreta da solução da equação de Laplace, é evidente que quanto maiores forem m e n, melhor se aproximará a solução calculada da solução exacta. Assim, obteremos sistemas de grandes dimensões, cujas matrizes são simétricas e muito esparsas, isto é, terão um elevado número de elementos nulos.

Determinação de diferenças de potencial e intensidades de corrente num circuito eléctrico

Consideremos os nós A, B, C e D e suponhamos que entre eles estabelecemos o circuito representado na figura 1.2, conhecido como ponte de Julie. O nosso objectivo vai ser a determinação da diferença de potencial (voltagem) V_{XY} e a intensidade de corrente I_{XY} em cada uma das ligações do circuito. São conhecidos os valores das resistências de cada uma das ligações (representadas por R_{XY}), assim como a voltagem V da fonte de alimentação colocada entre E e A.

Vamos começar por nos debruçar sobre as voltagens. Nesse sentido, iremos usar a *lei da diferença de potencial de Kirchhoff*, que afirma que a soma algébrica das voltagens de um sistema fechado em equilíbrio é igual a zero. Se por exemplo aplicarmos a lei ao circuito ABE, temos

$$V_{AB} + V_{BE} + V_{EA} = 0 \tag{1.5}$$

Deste modo, se tivermos em conta que $V_{XY} = -V_{YX}$, obtemos

$$\left.\begin{array}{lll} \text{Circuito } ABE: & V_{AB} + V_{BE} + V_{EA} & = 0 \\ \text{Circuito } ADCB: & V_{AD} - V_{CD} - V_{BC} - V_{AB} & = 0 \\ \text{Circuito } BCE: & V_{BC} + V_{CE} - V_{BE} & = 0 \\ \text{Circuito } CDE: & V_{CD} + V_{DE} - V_{CE} & = 0 \end{array}\right\} \tag{1.6}$$

Iremos apenas considerar os quatro circuitos mencionados, porque a aplicação da lei da diferença de potencial de Kirchhoff para os restantes circuitos corresponde a adicionar algumas das equações de (1.6), obtendo-se equações redundantes que não precisam de ser consideradas.

Além disso, a *lei da intensidade de corrente de Kirchhoff* diz que a soma das intensidades de todas as correntes que passam por um determinado nó tem de ser igual a zero. Para o circuito da figura 1.2, e atendendo a que $I_{XY} = -I_{YX}$, temos então

$$\left.\begin{array}{lll} \text{Nó } A: & I_{AB} + I_{AD} - I_{EA} & = 0 \\ \text{Nó } B: & -I_{AB} + I_{BC} + I_{BE} & = 0 \\ \text{Nó } C: & -I_{BC} + I_{CD} + I_{CE} & = 0 \\ \text{Nó } D: & -I_{AD} - I_{CD} + I_{DE} & = 0 \end{array}\right\} \tag{1.7}$$

De notar que, se somarmos as quatro equações de (1.7), obtemos a equação correspondente à aplicação da lei da intensidade de corrente de Kirchhoff ao nó E. Por isso não é necessário considerar essa equação.

Finalmente, podemos obter uma relação entre as diferenças de potencial V_{XY} e as correspondentes intensidades de corrente I_{XY}, que é conhecida por *lei de Ohm* e traduzida por

$$V_{XY} = R_{XY} I_{XY} \tag{1.8}$$

A ligação EA é a única que não obedece à igualdade (1.8), em virtude de lhe ter sido aplicada uma fonte de alimentação de voltagem V. Aplicando a lei de Ohm a essa ligação, temos

$$V_{EA} + V = R_{EA} I_{EA}$$

ou seja,

$$V_{EA} = R_{EA} I_{EA} - V \tag{1.9}$$

Podemos agora combinar (1.6), (1.7), (1.8) e (1.9) no seguinte sistema de equações

$$Ax = b \tag{1.10}$$

com

$$A = \begin{bmatrix} 1 & & & & 1 & & 1 & & & & & & & & & & & & & \\ -1 & 1 & -1 & -1 & & & & & & & & & & & & & & & & \\ & & 1 & & -1 & 1 & & & & & & & & & & & & & & \\ & & & 1 & & -1 & 1 & & & & & & & & & & & & & \\ & & & & & & & 1 & 1 & & & & & & & & & & -1 \\ & & & & & & & -1 & & 1 & & & 1 & & & & & & \\ & & & & & & & & & -1 & & 1 & & & 1 & & & & \\ & & & & & & & & -1 & & -1 & & & & & 1 & & \\ 1 & & & & & & & -R_{AB} & & & & & & & & & & \\ & 1 & & & & & & & -R_{AD} & & & & & & & & & \\ & & 1 & & & & & & & -R_{BC} & & & & & & & & \\ & & & 1 & & & & & & & -R_{CD} & & & & & & & \\ & & & & 1 & & & & & & & -R_{BE} & & & & & & \\ & & & & & 1 & & & & & & & -R_{CE} & & & & & \\ & & & & & & 1 & & & & & & & -R_{DE} & & & \\ & & & & & & & 1 & & & & & & & -R_{EA} \end{bmatrix}$$

e

$$x = \begin{bmatrix} V_{AB} \\ V_{AD} \\ V_{BC} \\ V_{CD} \\ V_{BE} \\ V_{CE} \\ V_{DE} \\ V_{EA} \\ I_{AB} \\ I_{AD} \\ I_{BC} \\ I_{CD} \\ I_{BE} \\ I_{CE} \\ I_{DE} \\ I_{EA} \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -V \end{bmatrix} \tag{1.11}$$

Da descrição deste modelo nota-se que um circuito com cinco nós dá lugar a um sistema com dezasseis equações lineares. Além disso, a matriz do sistema é não simétrica e muito esparsa, isto é, contém um elevado número de elementos nulos. É fácil de concluir que à medida que aumenta o número de nós da rede se vão obtendo sistemas de equações lineares com dimensões muito elevadas e com matrizes ainda mais esparsas que a apresentada neste exemplo.

Aproximação de funções por polinómios

Suponhamos que pretendemos construir um polinómio $P(x)$ de grau n que aproxime uma função $f : \mathbb{R}^1 \mapsto \mathbb{R}^1$ em m pontos (x_i, y_i), $i = 1, \ldots, m$, isto é, que verifica

$$P(x_i) = f(x_i), i = 1, \ldots, m \tag{1.12}$$

Como

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

então as equações (1.12) podem ser escritas na forma

$$\begin{cases} a_0 + x_1 a_1 + x_1^2 a_2 + \ldots + x_1^n a_n &= f(x_1) \\ a_0 + x_2 a_1 + x_2^2 a_2 + \ldots + x_2^n a_n &= f(x_2) \\ & \vdots \\ a_0 + x_m a_1 + x_m^2 a_2 + \ldots + x_m^n a_n &= f(x_m) \end{cases} \tag{1.13}$$

A resolução deste sistema (1.13) permite obter os coeficientes a_i, $i = 0, \ldots, n$ do polinómio. É de notar que, ao contrário dos exemplos anteriores, a matriz deste sistema não é quadrada se $m \neq n + 1$. Se $m > n + 1$, então as igualdades (1.12) só se podem resolver aproximadamente, pelo que o polinómio não passa necessariamente pelos pontos $(x_i, f(x_i))$. A título de exemplo consideremos uma função f definida em quatro pontos do modo seguinte

i	x_i	y_i
1	2	2
2	4	11
3	6	28
4	8	40

e suponhamos que pretendemos determinar o polinómio $P_1(x) = a_0 + a_1 x$ de grau 1 que aproxima a função nesses pontos. Então o sistema (1.13) tem a forma

$$\begin{cases} a_0 + 2a_1 &= 2 \\ a_0 + 4a_1 &= 11 \\ a_0 + 6a_1 &= 28 \\ a_0 + 8a_1 &= 40 \end{cases} \tag{1.14}$$

e portanto tem quatro equações e duas incógnitas. É fácil de ver a característica da matriz do sistema é igual ao número de incógnitas pelo que o sistema tem solução única. Essa solução é dada por $a_0 = -\dfrac{25}{2}$ e $a_1 = \dfrac{131}{20}$ e pode ser determinada por um dos processos a discutir neste curso. Donde o polinómio é

$$P(x) = -\frac{25}{2} + \frac{131}{20}x$$

A figura 1.3 representa a recta $y = P(x)$ que aproxima a função nos pontos dados.

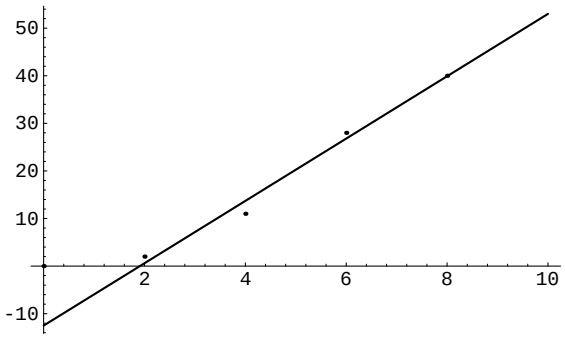


Figura 1.3: Representação dos pontos (x_i, y_i) e da recta $y = P(x)$

Capítulo 2

Alguns Conceitos de Álgebra Linear Numérica

Neste capítulo apresentamos algumas noções que são fundamentais para o desenvolvimento dos outros capítulos. Assim, começamos por rever os conceitos de precisão numérica e de erros de arredondamento, abordando os problemas da representação em vírgula flutuante e da análise do comportamento de algoritmos. Seguidamente recordamos algumas noções de álgebra linear, nomeadamente as de normas de vectores e de matrizes e de raio espectral de uma matriz. Finalmente introduzimos o conceito de número de condição de uma matriz, que tem um papel fundamental na precisão numérica da solução de um sistema de equações lineares.

2.1 Precisão numérica e erros de arredondamento

As potencialidades finitas de um computador a nível de memória e de capacidade de processamento só permitem que se trabalhe com um subconjunto dos números reais, constituído pelos Algarismos de Vírgula Flutuante. O sistema F de algarismos de vírgula flutuante é caracterizado por três números reais, que são a Base β, o Expoente e a Mantissa. Assim, todo o elemento f de F se pode escrever na forma

$$f = \pm .d_1 d_2 \dots d_t \times \beta^{\ell}$$

com $0 \leq d_i < \beta, d_1 \neq 0$. A $.d_1 d_2 \dots d_t$ dá–se o nome de mantissa, a ℓ chama–se expoente e t corresponde à Precisão do sistema. O expoente ℓ está compreendido entre dois números inteiros $\mathcal{L}$ e $\mathcal{U}$. Se $0 \neq f \in F$, então

$$m \leq |f| \leq M$$

com $m = \beta^{\mathcal{L}-1}$ e $M = \beta^{\mathcal{U}}(1 - \beta^{-t})$. A precisão t e a base β estão dependentes do computador usado. Assim, por exemplo, num CDC Cyber 830 a base é 2 e a precisão é 48. Como $2^{48} \approx 10^{14.4}$, então podemos armazenar um número real com 14 algarismos com precisão total. Por outro lado, num IBM AT baseado no processador Intel 80286 com co–processador aritmético 80287 tem–se $\beta = 16$ e $t = 6$ ($t = 14$ em precisão dupla). Portanto podemos armazenar um número real com 7 dígitos (16 em precisão dupla) sem qualquer erro. Os valores de $\mathcal{L}$ e $\mathcal{U}$ também variam de máquina para máquina. Assim, no computador IBM acima referido, tem–se $\mathcal{L} = -64$ e $\mathcal{U} = 63$, ao passo que no CDC Cyber 830 se tem $\mathcal{L} = -976$ e $\mathcal{U} = 1070$. Portanto no IBM os reais variam entre 10^{-77} e 10^{76}, enquanto que no CDC se podem considerar números entre 10^{-294} e 10^{322}.

A cada número real x podemos associar o número de vírgula flutuante $fl(x)$ definido por

$$fl(x) = \begin{cases} \text{o elemento } c \in F \text{ mais próximo de } x \text{ se aritmética} \\ \qquad \text{de arredondamento é usada} \\ \\ \text{o elemento } c \in F \text{ mais próximo de } x \text{ e tal que} \\ \quad |c| \leq |x| \text{ se aritmética de corte é usada.} \end{cases}$$

A escolha entre o uso de aritmética de arredondamento ou de corte varia de computador para computador. Da definição apresentada tem–se que

$$fl(x) = x(1+\epsilon), \quad |\epsilon| \leq \epsilon_M \tag{2.1}$$

A quantidade ϵ_M, conhecida como Precisão da máquina , define–se por

$$\epsilon_M = \begin{cases} \frac{1}{2}\beta^{1-t} & \text{se aritmética de arredondamento é usada} \\ \beta^{1-t} & \text{se aritmética de corte é usada} \end{cases}$$

isto é, é o zero para o computador. De (2.1) conclui–se que qualquer número real x satisfaz

$$\frac{|fl(x) - x|}{|x|} \leq \epsilon_M \tag{2.2}$$

ou seja, o erro absoluto entre x e $fl(x)$ é inferior ou igual a ϵ_M. Como ϵ_M representa o zero para o computador, então o número x é representado no computador por $fl(x)$. Fazendo $x = 1$ em (2.2) obtém–se

$$fl(1) \leq 1 + \epsilon_M.$$

Portanto ϵ_M pode ser determinado calculando o menor valor τ tal que $1+\tau = 1$ para o computador. Deste modo, partindo de $\tau = \frac{1}{2}$ e dividindo sucessivamente τ por 2 (isto é, fazendo sucessivamente $\tau = \tau/2$), obtém–se ϵ_M quando o computador não encontrar qualquer diferença entre $1 + \tau$ e 1.

A armazenagem de números reais usando aritmética finita de vírgula flutuante implica que só se pode esperar uma precisão em qualquer tipo de operação quanto muito tão boa como a precisão do computador. Portanto as operações aritméticas conduzem a erros na precisão do resultado, que se chamam Erros de Arredondamento.

Neste trabalho iremos abordar a resolução de problemas por meio de algoritmos, isto é, de processos sequenciais que conduzem à solução do problema em questão. Como estamos limitados à precisão finita proporcionada pelo computador, os sucessivos erros de arredondamento cometidos vão–se propagando e a solução obtida pelo algoritmo poderà em muitas circunstâncias ser muito diferente da solução real. Neste caso o algoritmo em questão não poderá ser considerado bom para a resolução do problema. Esta apreciação é contudo difícil na prática, uma vez que não temos em geral meios para avaliar a diferença entre as duas soluções. Existem no entanto dois tipos de análise que nos podem dar uma ideia do seu comportamento, conhecidas por Análises Backward e Forward em língua inglesa.

Na Análise Backward procura–se encontrar uma relação da forma

$$||s - \bar{s}|| \leq \delta$$

em que s é a solução exacta, $\bar{s}$ a solução calculada pelo algoritmo e $||\cdot||$ representa uma razoável medida para a diferença. Se δ é pequeno (grande), então o algoritmo é bom (mau). Além da dificuldade de implementar esta análise na prática, ela pode conduzir a conclusões erróneas. Com efeito, é extremamente difícil encontrar algoritmos para os quais δ seja pequeno, pois existem problemas, ditos Mal Condicionados, para os quais pequenas modificações nos dados conduzem a grandes alterações na solução. Esta propriedade é dependente do problema concreto que está a ser resolvido, e nada tem que ver com o algoritmo.

A Análise Forward considera a solução encontrada como a solução exacta de um problema perturbado $\bar{P}$ do problema P que se pretende resolver. Neste tipo de análise procura–se determinar um valor Δ tal que

$$||P - \bar{P}|| \leq \Delta \tag{2.3}$$

Um algoritmo diz–se Estável (Instável) se for possível (impossível) encontrar um valor de Δ satisfazendo (2.3). O uso de algoritmos estáveis é preferível, nomeadamente nas áreas de análise numérica e optimização.

2.2 Vectores e Matrizes

O espaço $\mathbb{R}^n$ é o conjunto dos elementos $(u_1, \ldots, u_n)$ com u_i números reais. Cada elemento u desse conjunto é denominado vector e escreve-se $u = (u_1, \ldots, u_n) \in \mathbb{R}^n$. Os números reais u_i dizem-se componentes do vector u e n é a dimensão ou ordem do vector. Uma matriz A com m linhas e n colunas é um conjunto de m vectores de dimensão n da forma

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}$$

Se $A_{i.}$ e $A_{.j}$ representarem respectivamente a linha i e a coluna j de A, então também podemos escrever

$$A = \begin{bmatrix} A_{1.} \\ \vdots \\ A_{m.} \end{bmatrix}$$

ou

$$A = \begin{bmatrix} A_{.1} & \ldots & A_{.n} \end{bmatrix}$$

Assim A é também um conjunto de n vectores de dimensão m. Uma matriz com m linhas e n colunas tem mn elementos e daí se poder afirmar que pertence ao espaço $\mathbb{R}^{m\times n}$ e escrever $A \in \mathbb{R}^{m\times n}$. Também se diz que A tem ordem (ou dimensão) $m \times n$. Cada elemento da matriz A é representado por a_{ij}, onde i e j representam respectivamente os índices da linha linha e da coluna a que esse elemento pertence.

Dois vectores (matrizes) são iguais se e só se são iguais os elementos correspondentes a índices iguais. Tal como com os números reais, também existem operações com vectores e matrizes. As definições dessas operações são apresentadas a seguir.

1. Adição

$$u = [u_i] \in \mathbb{R}^n, v = [v_i] \in \mathbb{R}^n \Rightarrow u + v = [u_i + v_i] \in \mathbb{R}^n$$
$$A = [a_{ij}] \in \mathbb{R}^{m\times n}, B = [b_{ij}] \in \mathbb{R}^{m\times n} \Rightarrow A + B = [a_{ij} + b_{ij}] \in \mathbb{R}^{m\times n}$$

2. Multiplicação por um número real

$$u = [u_i] \in \mathbb{R}^n, \lambda \in \mathbb{R}^1 \Rightarrow \lambda u = [\lambda u_i] \in \mathbb{R}^n$$

$$A = [a_{ij}] \in \mathbb{R}^{m\times n}, \lambda \in \mathbb{R}^1 \Rightarrow \lambda A = [\lambda a_{ij}] \in \mathbb{R}^{m\times n}$$

3. Produto escalar de dois vectores

$$u = [u_i] \in \mathbb{R}^n, v = [v_i] \in \mathbb{R}^n \Rightarrow u \cdot v = \sum_{i=1}^{n} u_i v_i \in \mathbb{R}^1$$

4. Produto de duas matrizes

$$A = [a_{ij}] \in \mathbb{R}^{m\times p}, B = [b_{ij}] \in \mathbb{R}^{p\times n} \Rightarrow AB = [c_{ij}] \in \mathbb{R}^{m\times n}, c_{ij} = A_{i.} \cdot B_{.j}$$

É importante notar que não faz sentido adicionar vectores e matrizes de ordens diferentes. Além disso a multiplicação de duas matrizes exige que o número de colunas de A seja igual ao número de linhas de B. Com efeito só nesse caso os produtos escalares $A_{i.} \cdot B_{.j}$ podem ser efectuados.

O vector (matriz) nulo é representado por 0 e tem todos os elementos iguais a zero. Uma matriz diz-se Rectangular (Quadrada) se o número de linhas é diferente (igual) ao número de colunas. Para matrizes quadradas também se diz que a sua ordem é o seu número de linhas e colunas. A

diagonal de uma matriz quadrada é o conjunto dos seus elementos diagonais a_{ii} em que o índice da linha é igual ao da coluna. Uma matriz diz-se Diagonal se todos os seus elementos não diagonais são iguais a zero. A matriz Identidade é a matriz diagonal com elementos diagonais iguais a um e representa-se por I ou I_n, com n a sua ordem. Uma matriz quadrada diz-se Triangular Inferior (Superior) se todos os elementos acima (abaixo) da diagonal são iguais a zero. Assim, uma matriz diagonal é triangular inferior e superior.

Diz-se que um vector $u \in \mathbb{R}^n$ é combinação linear de p vectores $u^1, \ldots, u^p$ de $\mathbb{R}^n$ se existem números reais $\lambda_1, \ldots, \lambda_p$ tais que

$$u = \lambda_1 u^1 + \ldots + \lambda_p u^p$$

É evidente que o vector nulo é sempre combinação linear de p vectores, pois tem-se

$$0 = 0u^1 + \ldots + 0u^p$$

Se essa for a única combinação linear do vector nulo em termos dos vectores u^i, diz-se que esses vectores são linearmente independentes. De outro modo, os vectores $u^1, \ldots, u^p$ são linearmente dependentes e pelo menos um desses vectores pode-se escrever como combinação linear dos restantes. O número máximo de vectores linearmente independentes em $\mathbb{R}^n$ é igual a n. Assim por exemplo os vectores

$$u^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, u^2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, u^3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, u^4 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

são linearmente dependentes, e u^4 é combinação linear dos restantes vectores. Com efeito, tem-se

$$u^4 = 1u^1 + 2u^2 + 3u^3$$

Para qualquer inteiro positivo n os vectores $e^i \in \mathbb{R}^n$ definidos por

$$e^i_j = \begin{cases} 1 \text{ se } i = j \\ 0 \text{ se } i \neq j \end{cases}$$

são linearmente independentes e qualquer vector $u \in \mathbb{R}^n$ se escreve de modo único como combinação linear desses vectores. Com efeito tem-se

$$u = [u_i] \in \mathbb{R}^n \Rightarrow u = \sum_{i=1}^{n} u_i e^i$$

O conjunto $\{e^1, \ldots, e^n\}$ é normalmente conhecido como Base Canónica de $\mathbb{R}^n$.

A característica de uma matriz $A \in \mathbb{R}^{m \times n}$ é o número máximo de linhas (e colunas) linearmente independentes e representa-se por $c(A)$. Diz-se que uma matriz A tem característica completa se

$$c(A) = \min\{m, n\}$$

Um sistema de equações lineares com m equações e n incógnitas x_i pode ser escrito na forma

$$Ax = b$$

com $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$, $x = (x_i) \in \mathbb{R}^n$. Se o sistema $Ax = b$ tem solução e A tem característica completa, então

$$\begin{cases} c(A) = n \leq m & \Rightarrow \quad \text{sistema tem solução única} \\ c(A) = m < n & \Rightarrow \quad \text{sistema é indeterminado} \end{cases}$$

Neste curso iremos essencialmente considerar sistemas de equações lineares cujas matrizes têm característica completa.

Uma matriz quadrada A diz-se Não Singular (Singular) se a sua característica é igual (menor) à sua ordem. A é uma matriz não singular se e só se o sistema $Ax = b$ tem solução única. Portanto $Ax = 0$ tem solução $x \neq 0$ se e só se A é singular.

Se A é uma matriz não singular então existe a matriz inversa de A que se representa por A^{-1} e satisfaz de modo único a equação

$$AA^{-1} = A^{-1}A = I$$

com I a matriz identidade. Desta definição conclui-se que a matriz inversa $B = A^{-1}$ de uma matriz A não singular de ordem n se pode obter resolvendo n sistemas da forma

$$AB_{.j} = I_{.j}, j = 1, \ldots, n \tag{2.4}$$

com $B_{.j}$ e $I_{.j}$ as colunas da inversa A^{-1} e da matriz identidade de ordem n respectivamente.

O Determinante de uma matriz quadrada A de ordem n representa-se por $\det(A)$ e define-se por indução do seguinte modo:

(i) Se $n = 1$, então $A = [\alpha]$ e $\det(A) = \alpha$.

(ii) Se $n > 1$, então

$$\begin{aligned} \det(A) &= \sum_{k=1}^{n} (-1)^{i+k} a_{ik} \det(A_{ik}) \\ &= \sum_{k=1}^{n} (-1)^{k+j} a_{kj} \det(A_{kj}) \end{aligned} \tag{2.5}$$

onde A_{rs} é a matriz de ordem $n-1$ que se obtém de A por supressão da linha r e da coluna s.

É possível mostrar que o valor do determinante não depende da linha i ou da coluna j escolhida na fórmula (2.5). É fácil concluir que o uso desta fórmula torna o cálculo do determinante altamente dispendioso, a não ser em casos excepcionais em que existam muitos elementos nulos. Assim por exemplo consideremos uma matriz triangular inferior

$$A = \begin{bmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ a_{31} & a_{32} & a_{33} & & \\ \vdots & \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} \end{bmatrix}$$

Então escolhendo $i = 1$ na fórmula (2.5) vem

$$\det(A) = (-1)^{1+1} a_{11} \det(A_{11}) = a_{11} \det(A_{11})$$

com

$$A_{11} = \begin{bmatrix} a_{22} & & & \\ a_{32} & a_{33} & & \\ \vdots & \vdots & \ddots & \\ a_{n2} & a_{n3} & \ldots & a_{nn} \end{bmatrix}$$

Se agora procedermos de modo semelhante mais $n-2$ vezes obtemos o seguinte resultado

$$\det(A) = a_{11} a_{22} \ldots a_{nn} \tag{2.6}$$

Assim o determinante de uma matriz triangular (inferior ou superior) é igual ao produto dos seus elementos diagonais. Em particular, o determinante da matriz identidade é igual a um. Como veremos mais adiante, a fórmula (2.6) será devidamente explorada no cálculo do valor do determinante de uma matriz quadrada. Duas outras propriedades são também muito importantes nesse sentido e são apresentadas a seguir:

Teorema 2.1

(i) *Se $\bar{A}$ é a matriz que se obtém de A por troca de duas linhas (ou colunas) então*

$$\det(\bar{A}) = -\det(A)$$

(ii) *Se A e B são matrizes quadradas da mesma ordem, então*

$$\det(AB) = \det(A)\det(B)$$

As demonstrações destas propriedades podem ser encontradas em [Mirsky, cap. 1]. Além disso, usando este teorema é possível provar que A é não singular se e só se o seu determinante é diferente de zero.

Dada uma matriz $A \in \mathbb{R}^{m\times n}$, chama-se Transposta de A, e representa-se por A^T a matriz de ordem $n \times m$ cujas linhas são as colunas de A. Assim

$$A = [a_{ij}] \in \mathbb{R}^{m\times n} \Rightarrow A^T = [a_{ji}] \in \mathbb{R}^{n\times m}$$

É fácil de provar que se A é uma matriz quadrada, então

$$\det(A^T) = \det(A)$$

pelo que A^T é não singular se e só se o mesmo acontecer com A. Uma matriz quadrada é Simétrica se $A = A^T$, isto é, se são iguais os elementos colocados simetricamente em relação à diagonal de A. Se $A \in \mathbb{R}^{m\times n}$ é uma matriz rectangular, então AA^T e A^TA são matrizes quadradas simétricas de ordem m e n respectivamente e tem-se

$$\begin{cases} c(A) = m < n & \Rightarrow \quad AA^T \text{ é não singular e } A^TA \text{ é singular} \\ c(A) = n < m & \Rightarrow \quad A^TA \text{ é não singular e } AA^T \text{ é singular} \end{cases}$$

2.3 Normas e valores próprios

É perfeitamente conhecido que a distância entre dois números reais é determinada a partir do valor absoluto da sua diferença. Para determinar a distância entre vectores ou entre matrizes, usam–se as chamadas Normas Vectoriais ou Normas Matriciais, cujas definições são apresentadas nesta secção.

Uma Norma vectorial $||\cdot||$ é uma função de $\mathbb{R}^n$ em $\mathbb{R}$ que satisfaz as seguintes propriedades:

(i) $||x|| \geq 0$ para todo o $x \in \mathbb{R}^n$;

(ii) $||x|| = 0$ se e só se $x = 0$;

(iii) $||\alpha x|| = |\alpha| \cdot ||x||$, para $\alpha \in \mathbb{R}$ e $x \in \mathbb{R}^n$;

(iv) $||x + y|| \leq ||x|| + ||y||$ para todos $x, y \in \mathbb{R}^n$

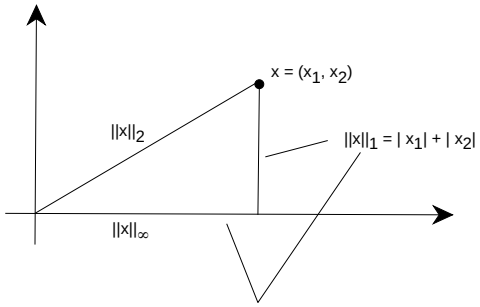
As normas mais utilizadas em Análise Numérica e Optimização são as denominadas Normas ℓ_p e são definidas a partir de

$$||x||_p = \left[\sum_{i=1}^{n} |x_i|^p\right]^{\frac{1}{p}}$$

Considerando os casos particulares em que p é igual a 1, 2 ou ∞, obtêm–se as seguintes normas:

$$\begin{aligned} \text{Norma } \ell_1: \quad ||x||_1 &= \sum_{i=1}^{n} |x_i| \\ \text{Norma } \ell_2: \quad ||x||_2 &= \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{x \cdot x} \\ \text{Norma } \ell_\infty: \quad ||x||_\infty &= \max_{i=1,\ldots,n} |x_i| \end{aligned}$$

A figura apresentada a seguir ilustra os valores dessas normas para vectores de $\mathbb{R}^2$.



Dessa figura facilmente se concluem as seguintes desigualdades

$$||x||_\infty \leq ||x||_2 \leq ||x||_1$$

$$||x||_1 \leq n||x||_2$$

$$||x||_2 \leq \sqrt{n}||x||_\infty$$

As demonstrações destas propriedades são deixadas como exercício. Portanto as normas ℓ_1, ℓ_2 e ℓ_∞ têm normalmente valores diferentes. Contudo os valores dessas normas são muito pequenos apenas quando o mesmo acontecer a todos os valores absolutos das componentes do vector, isto é, apenas quando a "grandeza" do vector é muito pequena.

Uma Norma Matricial $||\cdot||$ é uma função de $\mathbb{R}^{m\times n}$ em $\mathbb{R}$ que satisfaz as seguintes propriedades:

(i) $||A|| \geq 0$ para toda a matriz $A \in \mathbb{R}^{m\times n}$;

(ii) $||A|| = 0$ se e só se $A = 0$;

(iii) $||\lambda A|| = |\lambda| \cdot ||A||$ para qualquer $\lambda \in \mathbb{R}$ e $A \in \mathbb{R}^{m\times n}$;

(iv) $||A + B|| \leq ||A|| + ||B||$ para quaisquer $A, B \in \mathbb{R}^{m\times n}$;

(v) $||A \cdot B|| \leq ||A|| \cdot ||B||$ para quaisquer $A \in \mathbb{R}^{m\times p}$ e $B \in \mathbb{R}^{p\times n}$

Tal como as normas vectoriais, também podemos considerar as normas matriciais ℓ_1 e ℓ_∞, que são definidas a partir de

$$\begin{aligned} ||A||_1 &= \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}| \\ ||A||_\infty &= \max_{i=1,\ldots,m} \sum_{j=1}^{n} |a_{ij}| \end{aligned}$$

com $A \in \mathbb{R}^{m\times n}$. Assim, por exemplo se

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & -1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

então

$$||A||_1 = \max\{3, 3, 4\} = 4$$

$$||A||_\infty = \max\{6, 2, 2\} = 6$$

Antes de apresentarmos a norma matricial ℓ_2, necessitamos de recordar o conceito de valor próprio de uma matriz. Se A é uma matriz quadrada real de ordem n, um número real ou

complexo λ é um valor próprio de A se existe um vector não nulo v de componentes reais ou complexas, chamado vector próprio de A associado com λ, tal que

$$Av = \lambda v$$

Qualquer matriz A de ordem n tem exactamente n valores próprios que são as n raízes da equação característica

$$\det(A - \lambda I_n) = 0$$

com I_n a matriz identidade de ordem n.

A título de exemplo calculemos os valores próprios da matriz

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}$$

Então tem-se

$$\det(A - \lambda I) = \det \begin{bmatrix} 1-\lambda & 2 \\ 3 & 2-\lambda \end{bmatrix} = (1-\lambda)(2-\lambda) - 6 = \lambda^2 - 3\lambda - 4$$

Portanto os valores próprios de A são as duas raízes da equação $\det(A - \lambda I) = 0$, ou seja, $\lambda_1 = 4$ e $\lambda_2 = -1$. Para calcular um vector próprio u associado a λ_1 tem de se resolver o sistema

$$(A - \lambda_1 I)u = 0$$

cuja matriz é singular. Este sistema tem por isso uma infinidade de soluções. No nosso caso concreto tem-se

$$\begin{bmatrix} -3 & 2 \\ 3 & -2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = 0$$

e portanto qualquer vector $u \neq 0$ satisfazendo a $3u_1 = 2u_2$ é vector próprio de A associado ao valor próprio $\lambda_1 = 4$.

Este exemplo muito simples mostra a dificuldade de calcular valores próprios de uma matriz. Com efeito, tal cálculo implica a determinação das raízes de um polinómio, o que não é tarefa fácil. Felizmente existem processos mais eficientes para esse efeito, que no entanto não serão objecto de estudo neste curso. Contudo é importante referir que a determinação dos valores próprios de uma matriz A é sempre um processo mais complexo do que a resolução de um sistema com essa matriz.

O conjunto dos valores próprios da matriz A diz-se o espectro de A e é denotado por $\sigma(A)$. O raio espectral de A é o número real não negativo $\rho(A)$ definido por

$$\rho(A) = \max\{|\lambda| : \ \lambda \in \sigma(A)\} \tag{2.7}$$

Se A é uma matriz real e λ é um número real, então são reais todas as componentes do seu vector próprio associado. Além disso, matrizes simétricas têm valores próprios reais.

Como veremos no próximo capítulo, para qualquer matriz A a matriz simétrica $A^T A$ tem valores próprios não negativos. A norma matricial ℓ_2 é definida a partir de

$$||A||_2 = \sqrt{\rho(A^T A)} \tag{2.8}$$

Tal como nas normas vectoriais, também as normas matriciais ℓ_1, ℓ_2 e ℓ_∞ de uma matriz A têm em geral valores diferentes. Contudo os valores dessas normas apenas são muito pequenos quando o mesmo acontecer aos valores absolutos de todos os elementos dessa matriz. Daí a "proximidade" de uma matriz A com a matriz nula estar ligada com o valor da norma de A.

Uma norma vectorial $||\cdot||$ e uma norma matricial $||\cdot||'$ são Compatíveis se

$$||Ax|| \leq ||A||'||x||$$

para quaisquer vector x e matriz A. É fácil de ver que as normas matriciais ℓ_1, ℓ_2 e ℓ_∞ definidas anteriormente são compatíveis com as normas vectoriais correspondentes. Assim por exemplo provemos que

$$||Ax||_1 \leq ||A||_1 ||x||_1$$

Para isso tem-se

$$||Ax||_1 = \sum_{i=1}^{n} |(Ax)_i| = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} a_{ij} x_j \right| \leq \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}||x_j|$$

pois o módulo da soma é menor ou igual que a soma dos módulos. Mas

$$\begin{aligned} \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}||x_j| &= \sum_{j=1}^{n} \sum_{i=1}^{n} |a_{ij}||x_j| \\ &\leq \sum_{j=1}^{n} \left(\max_j \sum_{i=1}^{n} |a_{ij}| \right) |x_j| \\ &= \sum_{j=1}^{n} ||A||_1 |x_j| = ||A||_1 \sum_{j=1}^{n} |x_j| = ||A||_1 ||x||_1 \end{aligned}$$

o que estabelece o resultado pretendido.

Se uma norma vectorial $|| \cdot ||$ e uma norma matricial $|| \cdot ||'$ são compatíveis, então

$$||A||' \geq \max_{||x|| \neq 0} \frac{||Ax||}{||x||} = \max_{||x||=1} ||Ax||$$

Se a igualdade se verificar, então diz-se que a norma matricial é subordinada à norma vectorial. É fácil de ver que as normas matriciais ℓ_i são subordinadas às correspondentes normas vectoriais, para $i = 1, 2, \infty$, isto é, tem-se

$$||A||_i = \max_{||x||_i \neq 0} \frac{||Ax||_i}{||x||_i} = \max_{||x||_i=1} ||Ax||_i, i = 1, 2, \infty$$

É de notar que uma norma matricial pode ser compatível com uma norma vectorial mas não subordinada a ela. Assim por exemplo consideremos a chamada norma de Frobenius

$$||A||_F = \left[\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2 \right]^{\frac{1}{2}}$$

Então é possível mostrar que essa norma é compatível com a norma ℓ_2. Contudo essa norma não é subordinada à norma vectorial ℓ_2. Com efeito $||A||_F$ é em geral diferente de $||A||_2$ e portanto

$$||A||_F > \max_{||x||_2=1} ||Ax||_2$$

Consideremos agora uma norma matricial compatível com uma norma vectorial. Então para qualquer valor próprio λ de A tem-se

$$|\lambda| \ ||x|| = ||\lambda x|| = ||Ax|| \leq ||A|| \ ||x||$$

e portanto

$$\rho(A) \leq ||A|| \tag{2.9}$$

Como o raio espectral $\rho(A)$ de uma matriz A é o módulo de um valor próprio, poderá em geral existir ou não um valor próprio cujo valor seja igual a $\rho(A)$. A resposta é afirmativa em relação às matrizes não negativas ($A \geq 0$) em que todos os seus elementos são não negativos. Com efeito, verifica-se o seguinte teorema, cuja demonstração aparece em [Fiedler, cap. 4].

Teorema 2.2 *Se A é uma matriz não negativa então tem um valor próprio $\lambda \geq 0$ tal que $\lambda = \rho(A)$ e pelo menos um vector próprio associado a λ tem todas as componentes não negativas.*

Uma matriz A diz-se Convergente se $\lim_{k\to\infty} A^k = 0$, isto é, se todos os elementos de A^k tendem para zero à medida que k cresce indefinidamente. Como veremos no capítulo 8, as matrizes convergentes têm um papel fundamental na convergência dos métodos iterativos para a resolução de sistemas de equações lineares. Estas matrizes são normalmente caracterizadas em termos do seu raio espectral de acordo com o seguinte teorema:

Teorema 2.3 *Uma matriz A é convergente se e só se $\rho(A) < 1$.*

A demonstração deste resultado pode ser encontrada em [Fiedler, cap. 1]. Como ilustração deste resultado e do teorema anterior, consideremos a seguinte matriz diagonal não negativa

$$A = \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

Como A é diagonal, então os valores próprios são os seus elementos diagonais. Donde

$$\rho(A) = \max\{\frac{1}{4}, \frac{1}{2}\} = \frac{1}{2}$$

e portanto $\rho(A)$ é um valor próprio de A. Além disso $\rho(A) < 1$ e a matriz A é convergente. Com efeito $\lim_{k\to\infty} A^k = 0$, pois

$$A^k = \begin{bmatrix} \frac{1}{4^k} & 0 \\ 0 & \frac{1}{2^k} \end{bmatrix}, \lim_{k\to\infty} \frac{1}{4^k} = \lim_{k\to\infty} \frac{1}{2^k} = 0$$

Para terminar esta secção, iremos provar dois resultados que serão bastante úteis no capítulo 8.

Teorema 2.4 *Se A é uma matriz convergente, então $I - A$ é não singular e*

$$(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$$

Demonstração: Se a matriz $I - A$ é singular, então existe um vector $x \neq 0$ tal que

$$(I - A)x = 0$$

Portanto $Ax = x$ e 1 é valor próprio de A, o que contraria o teorema 2.3. Em relação à segunda parte do teorema, note-se que

$$(I - A)(I + A + A^2 + \ldots + A^k) = I - A^{k+1}$$

Aplicando limites a ambos os membros desta igualdade e tendo em conta que $\lim_{k\to\infty} A^k = 0$, obtém-se

$$(I - A)\sum_{k=0}^{\infty} A^k = I$$

e isso demonstra o resultado pretendido.

Teorema 2.5 *Se $|A| = [|a_{ij}|]$ e B é uma matriz não negativa tal que $|A| \leq B$, então $\rho(A) \leq \rho(B)$.*

Demonstração: Suponhamos que $\rho(A) > \rho(B)$. Então existe um número real s tal que $\rho(B) < s < \rho(A)$. Consideremos as matrizes $P = \frac{1}{s}A$ e $Q = \frac{1}{s}B$. Então

$$\rho(P) = \frac{\rho(A)}{s} > 1, \quad \rho(Q) = \frac{\rho(B)}{s} < 1 \tag{2.10}$$

e Q é convergente. Mas

$$|P^k| \leq |P|^k \leq Q^k$$

Portanto P é convergente, o que contradiz (2.10) e prova o teorema.

2.4 Perturbação de sistemas de equações lineares

Consideremos o sistema de equações lineares $Ax = b$, com $A \in \mathbb{R}^{n\times n}$ não singular e $x, b \in \mathbb{R}^n$. Nesta secção iremos averiguar em que medida é que pequenas alterações nos termos independentes b_i e nos elementos a_{ij} da matriz A podem afectar a solução do sistema. Por outras palavras, iremos ver quando é que um sistema de equações lineares é bem ou mal condicionado.

A solução do sistema $Ax = b$ é dada por $x = A^{-1}b$. Se b é perturbado para $b + \delta b$, necessariamente a solução do sistema passará a ser da forma $x + \delta x$. Ter–se–à então

$$x + \delta x = A^{-1}(b + \delta b) \tag{2.11}$$

Como $x = A^{-1}b$, então

$$\delta x = A^{-1}\delta b. \tag{2.12}$$

Escolhendo uma norma vectorial e uma norma matricial que sejam compatíveis, tem–se

$$||\delta x|| \leq ||A^{-1}||\ ||\delta b|| \tag{2.13}$$

Mas de $Ax = b$ tem-se $||b|| \leq ||A||\ ||x||$ e portanto

$$\frac{1}{||x||} \leq ||A|| \frac{1}{||b||} \tag{2.14}$$

Então de (2.13) e (2.14) vem

$$\frac{||\delta x||}{||x||} \leq ||A||\ ||A^{-1}|| \frac{||\delta b||}{||b||} \tag{2.15}$$

Por outro lado, se A é perturbada por δA, tem–se

$$(A + \delta A)(x + \delta x) = b \tag{2.16}$$

e portanto

$$\delta x = -A^{-1}\delta A(x + \delta x) \tag{2.17}$$

Então

$$||\delta x|| \leq ||A^{-1}|| \cdot ||\delta A|| \cdot ||x + \delta x|| \tag{2.18}$$

Dividindo ambos os membros da desigualdade (2.18) por $||x + \delta x||$ e multiplicando e dividindo o segundo membro da mesma desigualdade por $||A||$, vem

$$\frac{||\delta x||}{||x + \delta x||} \leq ||A||\ ||A^{-1}|| \frac{||\delta A||}{||A||} \tag{2.19}$$

Das fórmulas (2.15) e (2.19) conclui–se que um sistema de equações lineares é bem ou mal condicionado consoante

$$\text{cond}(A) = ||A||\ ||A^{-1}|| \tag{2.20}$$

seja pequeno ou grande. A quantidade $\text{cond}(A)$ definida em (2.20) chama–se número de condição de A. De notar que toda a matriz A tem número de condição maior ou igual do que um, pois, se I é a matriz identidade, tem–se

$$1 = ||I|| = ||AA^{-1}|| \leq ||A||\ ||A^{-1}|| = \text{cond}(A) \tag{2.21}$$

Se ϵ_M é a precisão da máquina, então matrizes com número de condição inferior a $\epsilon_M^{-2/3}$ são consideradas bem condicionadas, enquanto que matrizes com número de condição maior que $\epsilon_M^{-2/3}$ são mal condicionadas. A análise apresentada mostra que se a matriz de um sistema de equações lineares é mal condicionada então há que ter muito cuidado na formação dos termos independentes b_i e nos elementos da matriz a_{ij}. Com efeito, pequenos erros que se cometam nesses elementos

podem corresponder a erros bastante grandes na solução obtida. Como exemplo de ilustração da importância do número de condição, consideremos o sistema $Ax = b$ com

$$A = \begin{bmatrix} 0.550 & 0.423 \\ 0.484 & 0.372 \end{bmatrix} \qquad b = \begin{bmatrix} 0.127 \\ 0.112 \end{bmatrix}$$

A solução exacta do sistema é $x = \begin{bmatrix} 1.0 & -1.0 \end{bmatrix}^T$. Suponhamos que b é perturbado pelo vector $\delta b = \begin{bmatrix} 0.00007 & 0.00028 \end{bmatrix}^T$. Se resolvermos o sistema $A(x + \delta x) = b + \delta b$ com $\epsilon_M = 10^{-5}$, obtemos

$$x + \delta x = \begin{bmatrix} 1.7 \\ -1.91 \end{bmatrix} \Rightarrow \delta x = \begin{bmatrix} 0.7 \\ -0.91 \end{bmatrix}$$

Portanto, usando a norma ℓ_∞, tem–se

$$\frac{||\delta b||_\infty}{||b||_\infty} = 0.0022 \qquad \frac{||\delta x||_\infty}{||x||_\infty} = 0.91$$

ou seja, uma mudança pequena no vector b implicou uma grande mudança na solução do sistema $Ax = b$. Como

$$A^{-1} = \begin{bmatrix} -2818 & 3205 \\ 3667 & -4167 \end{bmatrix}$$

então $\text{cond}_\infty(A) = 7622$, o que confirma a análise apresentada nesta secção.

No nosso curso iremos trabalhar com algoritmos para a resolução de sistemas de equações lineares. Devido à precisão finita dos computadores, esses processos apenas são capazes de obter soluções aproximadas. Como não é conhecida a solução exacta do sistema, então temos de nos socorrer da norma do resíduo

$$r = b - A\bar{x}$$

para verificar que o vector $\bar{x}$ encontrado pelo algoritmo é uma boa aproximação da solução exacta do sistema. É evidente que em aritmética de precisão infinita x é a solução do sistema se e só se $r = 0$. Em geral, se $||r||$ é pequeno, então x é uma boa solução aproximada do sistema. No entanto a existência de uma matriz mal condicionada pode tornar esta análise menos correcta. Com efeito, seja x a solução exacta do sistema e $\bar{x}$ a solução encontrada por um algoritmo qualquer. Se fizermos $\delta x = x - \bar{x}$ em (2.15), obtemos

$$\frac{||x - \bar{x}||}{||x||} \leq \text{cond}(A)\frac{||r||}{||b||}$$

Esta desigualdade mostra que, no caso de a matriz A ser relativamente bem condicionada, $||r||$ é realmente uma boa medida para avaliar a precisão da solução $\bar{x}$. Contudo, se $\text{cond}(A)$ é elevado, então pode acontecer que $||r||$ seja pequeno e $\bar{x}$ não seja uma boa solução do sistema.

Estas considerações levam-nos à conclusão que o número de condição é uma medida extremamente importante na resolução de sistemas de equações lineares. No entanto a determinação do seu valor requer o conhecimento das normas da matriz A e da sua inversa A^{-1}. Se no primeiro caso não há grandes dificuldades (o cálculo das normas ℓ_1 e ℓ_∞ é relativamente fácil de efectuar) já o mesmo não acontece com a determinação de $||A^{-1}||$. Com efeito, o facto de trabalharmos em precisão finita implica que A^{-1} só pode ser obtida aproximadamente. Além disso, como veremos mais adiante, a determinação de A^{-1} requer bastante mais trabalho do que a resolução do sistema $Ax = b$ em causa. Por isso, na prática não se calcula o valor exacto de $\text{cond}(A)$, mas em vez disso determina–se uma aproximação β de $||A^{-1}||$ e faz–se

$$\text{cond}(A) \approx ||A||\beta$$

Existem alguns processos eficientes de obtenção de β, que serão discutidos no capítulo 4.

Se A é uma matriz simétrica, então $A^T A = A^2$ e portanto

$$||A||_2 = \rho(A) \tag{2.22}$$

com $\rho(A)$ o raio espectral de A. Além disso, se A é não singular e λ é valor próprio de A, então $\lambda \neq 0$ e $1/\lambda$ é valor próprio de A^{-1}. Portanto

$$\text{cond}_2(A) = ||A||_2||A^{-1}||_2 = \frac{\max|\lambda_i|}{\min|\lambda_i|} \tag{2.23}$$

Seja A uma matriz simétrica e consideremos a matriz

$$A + \mu I_n \tag{2.24}$$

com μ um número real positivo e I_n a matriz identidade de ordem n. Se λ é valor próprio de A, então $\lambda + \mu$ é valor próprio de $A + \mu I_n$. Se todos os valores próprios de A são positivos, então

$$\begin{aligned} \text{cond}_2(A + \mu I_n) &= \frac{\max|\lambda_i + \mu|}{\min|\lambda_i + \mu|} \\ &= \frac{\max \lambda_i + \mu}{\min \lambda_i + \mu} \end{aligned}$$

pois $\mu > 0$. Consideremos agora a função

$$\begin{aligned} f : \mathbb{R} - \{-\alpha\} &\longrightarrow \mathbb{R} \\ x &\longmapsto \frac{\beta + x}{\alpha + x} \end{aligned}$$

Se $\beta > \alpha$, então f é estritamente decrescente, pois

$$f'(x) = \frac{\alpha + x - \beta - x}{(\alpha + x)^2} = \frac{\alpha - \beta}{(\alpha + x)^2} < 0$$

Portanto o número de condição de $A + \mu I_n$ diminui à medida que o valor de μ aumenta. Esta propriedade tem importantes aplicações em álgebra linear numérica e optimização.

Exercícios

1. Considere as seguintes matrizes

$$A_1 = \begin{bmatrix} 4 & -5 \\ 2 & -3 \end{bmatrix},$$

$$A_2 = \begin{bmatrix} 3 & 2 & 4 \\ 2 & 0 & 2 \\ 4 & 2 & 3 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

$$A_5 = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

   (a) Calcule as normas $||A_i||_1$, $||A_i||_\infty$, $||A_i||_F$, com $i = 1, 2, 3, 4, 5$.

   (b) Calcule os valores próprios e os vectores próprios de A_i, $i = 1, 2, 3, 4, 5$.

(c) Calcule as normas $||A_i||_2$, $i = 1, 2, 3, 4, 5$.

2. Verifique que
$$||x||_\infty = \max_i |x_i|$$
satisfaz os axiomas da norma vectorial.

3. Mostre que
$$||x||_\infty \leq ||x||_2 \leq \sqrt{n}||x||_\infty,$$
$$||x||_1 \leq n||x||_2,$$
$$||x||_2 \leq \sqrt{n}||x||_\infty.$$

4. Verifique que
$$||A||_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}|$$
satisfaz os axiomas de norma vectorial.

5. Mostre que
$$||Ax||_\infty \leq ||A||_\infty ||x||_\infty$$
para quaisquer vector $x \in \mathbb{R}^n$ e matriz $A \in \mathbb{R}^{n\times n}$.

6. Calcule os números de condição das seguintes matrizes
$$A_1 = \begin{bmatrix} 3 & 4 \\ 5 & 7 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 2 & -1 \\ 1 & 1 & -1 \\ -2 & -5 & 4 \end{bmatrix}$$
relativamente às normas ℓ_1 e ℓ_∞.

7. Seja A uma matriz quadrada e $A^p = A \ldots A$ (p factores).
   (a) Demonstre que $||A^p|| \leq ||A||^p$ para todo o $p \in \mathbb{N}$.
   (b) Se
   $$A = \begin{bmatrix} 0.1 & 0.7 \\ 0.5 & 0.4 \end{bmatrix}$$
   mostre que todos os elementos de A^p são em valor absoluto inferiores a $(0.9)^p$.
   (c) Se
   $$A = \begin{bmatrix} 0 & 1/2 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1/2 & 0 \end{bmatrix}$$
   mostre que todos os elementos de A^p são limitados superiormente por $\sqrt{2}^{-p}$.

8. Mostre que se A é não singular, então
$$\frac{1}{||A^{-1}||} = \min\{||Ay|| : ||y|| = 1\}.$$

9.
   (a) Mostre que se λ é valor próprio de A então $\frac{1}{\lambda}$ é valor próprio de A^{-1}.
   (b) Mostre que se $\lambda_{\max}$ e $\lambda_{\min}$ são os valores próprios de maior e menor valor absoluto de uma matriz A não singular, então
   $$\text{cond}(A) \geq \frac{|\lambda_{\max}|}{|\lambda_{\min}|}.$$

10.

(a) Mostre que se A e B são matrizes não singulares, então

$$\text{cond}(AB) \leq \text{cond}(A)\text{cond}(B).$$

(b) Verifique essa fórmula com

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix}$$

e as normas ℓ_1, ℓ_2 e ℓ_∞.

11. Mostre que se B é uma matriz singular e A é uma matriz não singular, então

$$\frac{1}{\text{cond}(A)} \leq \frac{||A - B||}{||A||}.$$

Capítulo 3

Classes de Matrizes

Como será discutido em capítulos posteriores, existem várias classes de matrizes para as quais é possível desenvolver algoritmos para a resolução de sistemas de equações lineares. Neste capítulo iremos introduzir algumas das principais classes de matrizes e as suas propriedades mais relevantes para a resolução de sistemas. Os conceitos de operação pivotal, transformada principal e complemento de Schur de uma matriz são também importantes no desenvolvimento de algoritmos para sistemas de equações lineares e são por isso também discutidos neste capítulo.

3.1 Operações Pivotais, Transformada principal e Complemento de Schur

Seja A uma matriz de ordem $m \times n$ e sejam J e K dois subconjuntos de $\{1, \ldots, m\}$ e $\{1, \ldots, n\}$ respectivamente. Definimos a submatriz A_{JK} de A da seguinte forma

$$A_{JK} = [a_{jk}]_{j \in J, k \in K}$$

em que a_{jk} é um elemento genérico de A. Se A é uma matriz quadrada e $J = K$ diz-se que A_{JJ} é uma submatriz principal de A. Assim por exemplo se

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

e $J = K = \{1, 3\}$, então

$$A_{JJ} = \begin{bmatrix} 1 & 3 \\ 7 & 9 \end{bmatrix}$$

Uma matriz de permutação P_{ij} é uma matriz quadrada que se obtém da matriz identidade por troca das suas linhas e colunas i e j. Na figura 3.1 apresentamos a forma desta matriz. É fácil de ver que $P_{ij} = P_{ij}^T = P_{ij}^{-1}$. Uma matriz de permutação é o produto de um número finito (possivelmente igual a um) de matrizes da forma P_{ij}. Se P é uma matriz de permutação, então PA (AP) é uma matriz que se obtém trocando as linhas (colunas) de A correspondentes aos factores P_{ij} que constituem P. Assim, P^TAP é a matriz que difere de A na ordem de um certo número de linhas e colunas de A. Essa matriz P^TAP é denominada permutação principal de A. Assim, por exemplo, se

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 0 \\ -1 & -2 & -3 \end{bmatrix}$$

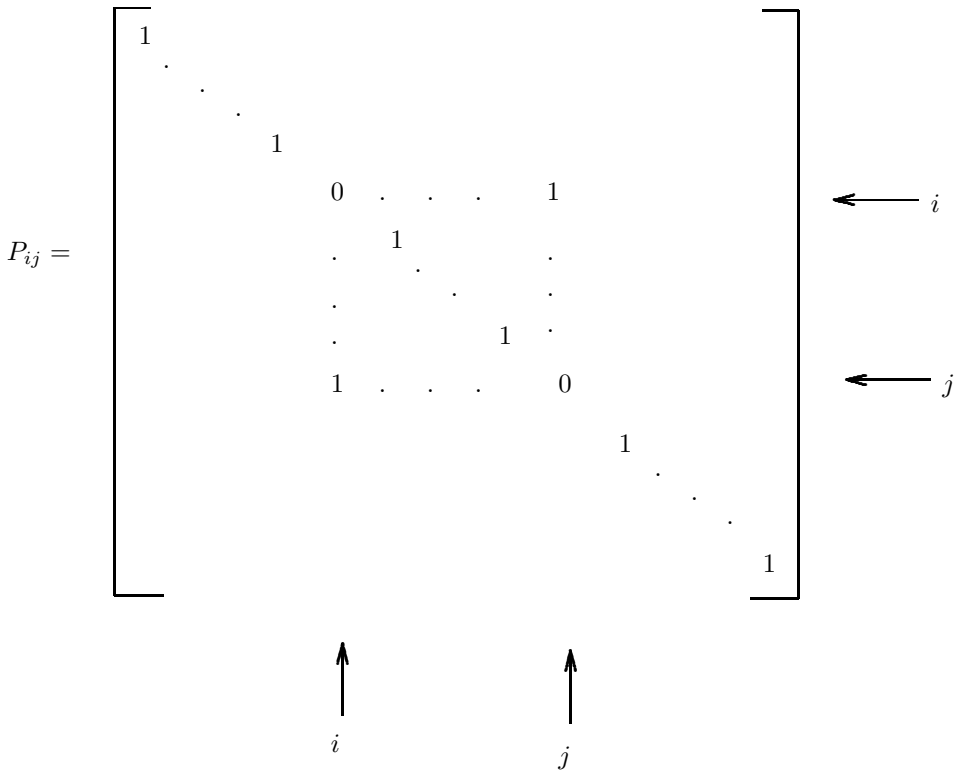


Figura 3.1: Uma matriz de permutação

e se $P = P_{12}P_{32}$ então

$$P^T AP = P_{12} \begin{bmatrix} 1 & 3 & 2 \\ -1 & -3 & -2 \\ 3 & 0 & 2 \end{bmatrix} P_{12} = \begin{bmatrix} -3 & -1 & -2 \\ 3 & 1 & 2 \\ 0 & 3 & 2 \end{bmatrix}$$

Consideremos agora um sistema de equações lineares

$$Eu = f \tag{3.1}$$

com $E \in \mathbb{R}^{m\times p}$, $f \in \mathbb{R}^m$, $u \in \mathbb{R}^p$ e $m < p$. Diz-se que u é uma solução básica desse sistema se existe um subconjunto

$$J = \{j_1, \ldots, j_m\}$$

de $\{1, \ldots, p\}$ tal que

(i) $u_j = 0$ para $j \notin J$.

(ii) as colunas $E_{.j_1}, \ldots, E_{.j_m}$ são linearmente independentes.

As variáveis u_j, $j \in J$ são chamadas básicas enquanto que as restantes variáveis se dizem não básicas.

Se u é uma solução básica do sistema $Eu = f$, então podemos escrever a matriz E na forma

$$E = \begin{bmatrix} B & N \end{bmatrix}$$

com $B = [E_{.j_1}, \ldots, E_{.j_m}]$ uma matriz não singular de ordem m, que se diz Base. Se $K = \{1, \ldots, p\} - J$, então $Eu = f$ é equivalente a

$$Bu_J + Nu_K = f$$

Como B é não singular, então podemos multiplicar ambos os membros dessa equação por B^{-1} e obter

$$u_J = B^{-1}f - B^{-1}Nu_K$$

Sejam $y = u_J$, $x = u_K$, $A = -B^{-1}N$ e n o número de elementos do conjunto K. Então podemos escrever o sistema anterior na forma

$$y = b + Ax \tag{3.2}$$

com $A \in \mathbb{R}^{m\times n}$, $y, b \in \mathbb{R}^m$ e $x \in \mathbb{R}^n$.

Dada uma solução básica do sistema (3.1), uma Operação Pivotal transforma uma solução básica u de base B numa solução básica $\bar{u}$ de base $\bar{B}$ por troca de um número finito t de variáveis básicas u_i por igual número de variáveis não básicas. A operação diz-se simples ou por blocos dependendo de t ser igual ou maior do que um respectivamente.

Consideremos primeiramente o caso da operação pivotal simples. Seja dada uma solução básica do sistema (3.1) que como vimos se pode escrever na forma

$$\left\{\begin{array}{lcl} y_1 &=& b_1 + a_{11}x_1 + \ldots + a_{1s}x_s + \ldots + a_{1n}x_n \\ \vdots & \vdots & \vdots \\ y_r &=& b_r + a_{r1}x_1 + \ldots + a_{rs}x_s + \ldots + a_{rn}x_n \\ \vdots & \vdots & \vdots \\ y_m &=& b_m + a_{m1}x_1 + \ldots + a_{ms}x_s + \ldots + a_{mn}x_n \end{array}\right.$$

Suponhamos que pretendemos efectuar uma operação pivotal simples que troca a variável não básica x_s com a variável básica y_r. Para isso, basta reescrever a equação r em ordem a x_s e substituir nas restantes equações. Então a_{rs} tem de ser diferente de zero e diz-se o pivot da operação. Se tal acontecer, tem-se

$$x_s = -\frac{b_r}{a_{rs}} - \frac{a_{r1}}{a_{rs}}x_1 - \ldots + \frac{1}{a_{rs}}y_r - \ldots - \frac{a_{rn}}{a_{rs}}x_n$$

Substituindo nas restantes equações, vem

$$y_i = \bar{b}_i + \bar{a}_{i1}x_1 + \ldots + \bar{a}_{is}y_r + \ldots + \bar{a}_{in}x_n, \quad i \neq r$$

com

$$\bar{a}_{is} = \frac{a_{is}}{a_{rs}}, \bar{b}_i = b_i - \bar{a}_{is}b_r, \bar{a}_{ij} = a_{ij} - \bar{a}_{is}a_{rj}, j \neq s \tag{3.3}$$

Chegamos assim à conclusão que uma operação pivotal com pivot a_{rs} transforma um sistema da forma (3.2) num sistema

$$\bar{y} = \bar{b} + \bar{A}\bar{x} \tag{3.4}$$

onde

$$\bar{y}_i = \left\{\begin{array}{lll} y_i & \text{se} & i \neq r \\ x_s & \text{se} & i = r \end{array}\right. , \quad \bar{x}_j = \left\{\begin{array}{lll} x_j & \text{se} & j \neq s \\ y_r & \text{se} & j = s \end{array}\right. \tag{3.5}$$

e $\bar{b}_i$ e $\bar{a}_{ij}$ satisfazem as fórmulas (3.3) para $i \neq r$ e

$$\bar{b}_r = -\frac{b_r}{a_{rs}}, \bar{a}_{rs} = \frac{1}{a_{rs}}, \bar{a}_{rj} = -\frac{a_{rj}}{a_{rs}}, j \neq s \tag{3.6}$$

Assim uma operação pivotal requer exactamente $m(n+1)$ operações (multiplicações e divisões).

Consideremos uma solução básica dada pelo sistema (3.2) e sejam $I \subseteq \{1,\ldots,m\}$ e $J \subseteq \{1,\ldots,n\}$ dois conjuntos tais que A_{IJ} é não singular. Se $K = \{1,\ldots,m\} - I$ e $L = \{1,\ldots,n\} - J$, então, a menos de permutações de linhas e colunas, podemos escrever (3.2) na forma

$$\left[\begin{array}{c} y_I \\ y_K \end{array}\right] = \left[\begin{array}{c} b_I \\ b_K \end{array}\right] + \left[\begin{array}{cc} A_{IJ} & A_{IL} \\ A_{KJ} & A_{KL} \end{array}\right] \left[\begin{array}{c} x_J \\ x_L \end{array}\right]$$

Tal como na operação pivotal simples, uma operação pivotal por blocos com pivot A_{IJ} consiste em resolver o sistema

$$A_{IJ}x_J = y_I - b_I - A_{IL}x_L$$

em ordem a x_J e substituir a solução obtida nas restantes equações. Se $|J|$ representar o número de elementos do conjunto J (e I), então há uma troca de $|J|$ variáveis não básicas x_J por outras tantas variáveis básicas y_I de modo a obter um sistema equivalente da forma

$$\begin{bmatrix} x_J \\ y_K \end{bmatrix} = \begin{bmatrix} \bar{b}_I \\ \bar{b}_K \end{bmatrix} + \begin{bmatrix} \bar{A}_{IJ} & \bar{A}_{IL} \\ \bar{A}_{KJ} & \bar{A}_{KL} \end{bmatrix} \begin{bmatrix} y_I \\ x_L \end{bmatrix}$$

com

$$\begin{array}{lcl} \bar{A}_{IJ} & = & A_{IJ}^{-1} \\ \bar{A}_{IL} & = & -A_{IJ}^{-1}A_{IL} \\ \bar{A}_{KJ} & = & A_{KJ}A_{IJ}^{-1} \\ \bar{A}_{KL} & = & A_{KL} - A_{KJ}A_{IJ}^{-1}A_{IL} \end{array} \tag{3.7}$$

Das definições apresentadas facilmente se conclui que uma operação pivotal por blocos com pivot A_{IJ} é equivalente a $|J|$ operações pivotais simples com pivots cujos índices (i, j) pertencem ao produto cartesiano $I \times J$.

Consideremos agora o caso em que A é quadrada, isto é, $m = n$. Então existe uma permutação principal de A tal que

$$P^T AP = \begin{bmatrix} A_{JJ} & A_{JK} \\ A_{KJ} & A_{KK} \end{bmatrix}$$

Uma operação pivotal com pivot A_{JJ} diz-se Principal e a matriz $\bar{A}$ obtida por essa operação é chamada Transformada Principal de A com pivot A_{JJ}. Das fórmulas (3.7) tem-se

$$\bar{A} = \begin{bmatrix} A_{JJ}^{-1} & -A_{JJ}^{-1}A_{JK} \\ A_{KJ}A_{JJ}^{-1} & (A|A_{JJ}) \end{bmatrix}$$

onde $(A|A_{JJ})$ é conhecido por Complemento de Schur de A_{JJ} em A e é dado por

$$(A|A_{JJ}) = A_{KK} - A_{KJ}A_{JJ}^{-1}A_{JK} \tag{3.8}$$

Como exemplo de ilustração destes conceitos, consideremos a solução básica definida por

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 1 \\ 2 & 0 & -1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

e que pretendemos efectuar uma operação pivotal com pivot a_{31}. Então usando as fórmulas (3.3) e (3.6) por essa ordem vem

$$\bar{a}_{11} = \frac{a_{11}}{a_{31}} = 1, \bar{a}_{21} = \frac{a_{21}}{a_{31}} = 2$$

$$\bar{a}_{12} = a_{12} - \bar{a}_{11}a_{32} = -1, \bar{a}_{13} = a_{13} - \bar{a}_{11}a_{33} = 0$$

$$\bar{a}_{22} = a_{22} - \bar{a}_{21}a_{32} = 0, \bar{a}_{23} = a_{23} - \bar{a}_{21}a_{33} = -3$$

$$\bar{b}_1 = b_1 - \bar{a}_{11}b_3 = 1, \bar{b}_2 = b_2 - \bar{a}_{21}b_3 = -1$$

$$\bar{b}_3 = -\frac{b_3}{a_{31}} = 0, \bar{a}_{31} = \frac{1}{a_{31}} = 1, \bar{a}_{32} = -\frac{a_{32}}{a_{31}} = 0, \bar{a}_{33} = -\frac{a_{33}}{a_{31}} = -1$$

Portanto a solução básica obtida por essa operação pivotal satisfaz o seguinte sistema

$$\begin{bmatrix} y_1 \\ y_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 0 \\ 2 & 0 & -3 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} y_3 \\ x_2 \\ x_3 \end{bmatrix}$$

Os valores das variáveis não básicas são iguais a zero e os das variáveis básicas são $y_1 = 1$, $y_2 = -1$, $x_1 = 0$.

Consideremos agora um sistema de equações lineares

$$Ax = b$$

em que A é uma matriz quadrada de ordem n. Então podemos escrever esse sistema na forma

$$0 = b - Ax$$

ou ainda

$$y = b - Ax, y = 0$$

Portanto a solução do sistema $Ax = b$ pode ser obtida tornando não básicas as n variáveis y_i por intermédio de n operações pivotais. A solução básica correspondente a esse sistema satisfaz

$$x = \bar{b} + \bar{A}y$$

e as variáveis básicas dessa solução constituem a solução do sistema. Portanto a solução de um sistema pode ser obtida por intermédio de n operações pivotais. Como cada operação pivotal requer $n(n+1)$ operações, então o número total de operações é da ordem de n^3. Como veremos no próximo capítulo, é possível resolver um sistema com um terço desse número de operações. Por outro lado, as operações pivotais constituem em geral um processo instável e portanto são bastante sujeitas a erros de arredondamento. Por essas razões o seu uso não tem sido recomendado na resolução de sistemas de equações lineares.

As operações pivotais também podem ser usadas para determinar a inversa de uma matriz não singular. Com efeito tem-se $y = Ax$ se e só se $x = A^{-1}y$, pelo que a matriz inversa de uma matriz não singular se pode obter a partir de n operações pivotais. Portanto o número total de operações para calcular a inversa de uma matriz é exactamente igual a n^3. No próximo capítulo iremos estudar um outro processo de calcular a inversa de uma matriz não singular e mostraremos que o número de operações é aproximadamente igual a n^3. Contudo esse processo é estável, contrariamente ao uso das operações pivotais, e daí ser em geral mais recomendado na prática. A título de exemplo calculemos a inversa da matriz

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix}$$

Então escrevamos

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Efectuando uma operação pivotal com pivot a_{12}, vem

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

Para calcular a inversa há que trocar a variável y_2 com a variável x_1, o que pode ser feito através da operação pivotal com pivot a_{21}. Então tem-se

$$\begin{bmatrix} x_2 \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} y_2 \\ y_1 \end{bmatrix}$$

ou ainda

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

Portanto

$$A^{-1} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix}$$

Como veremos mais adiante o Complemento de Schur de uma matriz desempenha um papel fundamental na resolução de sistemas de equações lineares. Para terminar esta secção iremos demonstrar algumas propriedades das transformadas principais e Complemento de Schur que nos serão muito úteis nesse sentido. Assim, o próximo teorema relaciona os determinantes de submatrizes principais de A e de uma sua transformada principal.

Teorema 3.1 *Seja $A \in \mathbb{R}^{n\times n}$ e J, L dois conjuntos de índices tais que $J, L \subseteq \{1, \ldots, n\}$. Se $\bar{A}$ é uma transformada principal de A obtida por uma operação pivotal principal com pivot A_{JJ}, então*

$$\det(\bar{A}_{LL}) = \frac{\det(A_{J\Delta L, J\Delta L})}{\det(A_{JJ})}$$

em que $J\Delta L = J \cup L - J \cap L$.

Demonstração: Se J e L são dois conjuntos de índices quaisquer, podemos escrever

$$\begin{array}{rcll} J & = & M \cup N, & M \cap N = \emptyset \\ L & = & N \cup P, & N \cap P = \emptyset \end{array} \tag{3.9}$$

onde algum destes conjuntos M, N ou P pode ser vazio. Provemos apenas o teorema no caso mais geral em que M, N e P são não vazios, pois os outros casos são mais simples.

Consideremos a submatriz de A formada pelas linhas e colunas de A de índices pertencentes a J e a L, isto é

$$A_1 = \begin{bmatrix} A_{MM} & A_{MN} & A_{MP} \\ A_{NM} & A_{NN} & A_{NP} \\ A_{PM} & A_{PN} & A_{PP} \end{bmatrix}$$

e seja $\bar{A}_1$ a matriz obtida de A_1 por uma operação pivotal com pivot A_{JJ}. Então podemos escrever

$$\bar{A}_1 = \begin{bmatrix} \bar{A}_{MM} & \bar{A}_{MN} & \bar{A}_{MP} \\ \bar{A}_{NM} & \bar{A}_{NN} & \bar{A}_{NP} \\ \bar{A}_{PM} & \bar{A}_{PN} & \bar{A}_{PP} \end{bmatrix}$$

Como de (3.9) se tem $J\Delta L = J \cup L - J \cap L = M \cup P$, então tem de se provar que

$$\det\begin{bmatrix} \bar{A}_{NN} & \bar{A}_{NP} \\ \bar{A}_{PN} & \bar{A}_{PP} \end{bmatrix} = \frac{\det\begin{bmatrix} A_{MM} & A_{MP} \\ A_{PM} & A_{PP} \end{bmatrix}}{\det\begin{bmatrix} A_{MM} & A_{MN} \\ A_{NM} & A_{NN} \end{bmatrix}} \tag{3.10}$$

Para isso, considere–se o sistema

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = A_1 \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Como $\bar{A}_1$ é a matriz obtida de A_1 por uma operação pivotal principal com pivot A_{JJ}, então tem–se

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \bar{A}_1 \begin{bmatrix} u \\ v \\ z \end{bmatrix}$$

Consideremos agora as transformações lineares f e g definidas respectivamente por

$$\begin{bmatrix} u \\ v \\ z \end{bmatrix} = \begin{bmatrix} A_{MM} & A_{MN} & A_{MP} \\ A_{NM} & A_{NN} & A_{NP} \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\begin{bmatrix} u \\ y \\ w \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ \bar{A}_{NM} & \bar{A}_{NN} & \bar{A}_{NP} \\ \bar{A}_{PM} & \bar{A}_{PN} & \bar{A}_{PP} \end{bmatrix} \begin{bmatrix} u \\ v \\ z \end{bmatrix}$$

onde I é a matriz identidade. Então a composição $g \circ f$ é a transformação linear definida por

$$\begin{bmatrix} u \\ y \\ w \end{bmatrix} = \begin{bmatrix} A_{MM} & A_{MN} & A_{MP} \\ 0 & I & 0 \\ A_{PM} & A_{PN} & A_{PP} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Mas, se B, C e D são as matrizes das transformações lineares f, g e $g \circ f$ respectivamente, então $\det(D) = \det(B) \cdot \det(C)$. Portanto

$$\det \begin{bmatrix} A_{MM} & A_{MP} \\ A_{PM} & A_{PP} \end{bmatrix} = \det \begin{bmatrix} A_{MM} & A_{MN} \\ A_{NM} & A_{NN} \end{bmatrix} \cdot \det \begin{bmatrix} \bar{A}_{NN} & \bar{A}_{NP} \\ \bar{A}_{PN} & \bar{A}_{PP} \end{bmatrix}$$

e dividindo ambos os membros desta última igualdade por $\det(A_{JJ})$ obtém–se a igualdade (3.10) pretendida.

Como corolário tem–se

Teorema 3.2 (Fórmula de Schur) *Se $A \in \mathbb{R}^{n \times n}$ e A_{JJ} é uma submatriz principal de A não singular, então*

$$\det(A) = \det(A_{JJ}) \cdot \det(A|A_{JJ})$$

Finalmente, podemos demonstrar a chamada Fórmula do Quociente:

Teorema 3.3 (Fórmula do Quociente) *Sejam*

$$A = \begin{bmatrix} B & P \\ M & N \end{bmatrix} \quad \text{e} \quad B = \begin{bmatrix} C & D \\ E & F \end{bmatrix}$$

Se C e B são não singulares, então

$$(A|B) = ((A|C)|(B|C))$$

Demonstração: Consideremos o sistema $y = Ax$, isto é

$$\begin{bmatrix} y^1 \\ y^2 \\ y^3 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} C & D \\ E & F \end{bmatrix} & P \\ M & N \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \\ x^3 \end{bmatrix}$$

Como B é não singular, então podemos efectuar uma operação pivotal principal com pivot B e obter o sistema equivalente

$$\begin{bmatrix} x^1 \\ x^2 \\ y^3 \end{bmatrix} = \begin{bmatrix} B^{-1} & \bar{P} \\ \bar{M} & (A|B) \end{bmatrix} \begin{bmatrix} y^1 \\ y^2 \\ x^3 \end{bmatrix} \tag{3.11}$$

Por outro lado, B e C são matrizes não singulares e portanto, pela Fórmula de Schur, $\det(B|C) \neq 0$, ou seja, $(B|C)$ é não singular. Então a operação pivotal com pivot B é equivalente a duas operações pivotais com pivots C e $(B|C)$. Se essas duas operações pivotais forem efectuadas sucessivamente, então obtém–se o sistema:

$$\begin{bmatrix} x^1 \\ x^2 \\ y^3 \end{bmatrix} = \begin{bmatrix} B^{-1} & \bar{P} \\ \bar{M} & ((A|C)|(B|C)) \end{bmatrix} \begin{bmatrix} y^1 \\ y^2 \\ x^3 \end{bmatrix} \tag{3.12}$$

Como (3.11) e (3.12) representam o mesmo sistema, então

$$(A|B) = ((A|C)|(B|C))$$

o que estabelece a igualdade pretendida.

Os conceitos de Complemento de Schur e de transformada principal têm um papel importante no estabelecimento de algumas propriedades de classes de matrizes que por sua vez serão determinantes para a análise dos métodos a discutir nos próximos capítulos. Até ao fim deste capítulo iremos apresentar uma revisão dessas classes.

3.2 Matrizes diagonalmente dominantes

Existem matrizes diagonalmente dominantes por linhas (RDD), por colunas (CDD) e por linhas e colunas (DD), assim como matrizes estritamente diagonalmente dominantes (SRDD, SCDD e SDD). Essas classes de matrizes são definidas do seguinte modo:

Definição 3.1 *Seja $A \in \mathbb{R}^{n\times n}$. Então*

$$A \in \text{RDD} \Leftrightarrow |a_{ii}| \geq \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|,\ \ i = 1, \ldots, n$$

$$A \in \text{CDD} \Leftrightarrow |a_{jj}| \geq \sum_{\substack{i=1\\ i\neq j}}^{n} |a_{ij}|,\ \ j = 1, \ldots, n$$

$$A \in \text{SRDD} \Leftrightarrow |a_{ii}| > \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|,\ \ i = 1, \ldots, n$$

$$A \in \text{SCDD} \Leftrightarrow |a_{jj}| > \sum_{\substack{i=1\\ i\neq j}}^{n} |a_{ij}|,\ \ j = 1, \ldots, n$$

$$A \in \text{DD} \Leftrightarrow A \in \text{RDD}\ \ e\ \ A \in \text{CDD}$$

$$A \in \text{SDD} \Leftrightarrow A \in \text{SRDD}\ \ e\ \ A \in \text{SCDD}$$

Além disso

$$A \in \text{RDD}_+(\text{CDD}_+, \text{DD}_+) \Leftrightarrow A \in \text{RDD}(\text{CDD}, \text{DD})\ \ e\ a_{ii} \geq 0, i = 1, \ldots, n$$

$$A \in \text{SRDD}_+(\text{SCDD}_+, \text{SDD}_+) \Leftrightarrow A \in \text{SRDD}(\text{SCDD}, \text{SDD})\ \ e\ a_{ii} > 0, i = 1, \ldots, n$$

Das definições apresentadas facilmente se conclui que

$$A \in \text{SRDD}(\text{RDD}) \Leftrightarrow A^T \in \text{SCDD}(\text{CDD}) \tag{3.13}$$

Por essa razão iremos apenas apresentar as propriedades para as matrizes diagonalmente dominantes por linhas. Todos esses resultados são verdadeiros para matrizes diagonalmente dominantes por colunas devido à equivalência (3.13).

Como consequência imediata da definição, é possível obter as seguintes inclusões:

$$\begin{array}{ccccc} \text{SCDD} & \supset & \text{SDD} & \subset & \text{SRDD} \\ \cap & & \cap & & \cap \\ \text{CDD} & \supset & \text{DD} & \subset & \text{RDD} \end{array} \tag{3.14}$$

Estas inclusões são estritas, conforme mostram os seguintes exemplos:

1. $A = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix} \in$ SRDD mas $A \notin$ SDD.

2. $A = \begin{bmatrix} 2 & 2 \\ 3 & 4 \end{bmatrix} \in$ RDD mas $A \notin$ SRDD.

3. $A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \in$ DD mas $A \notin$ SDD.

4. $A = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \in$ RDD mas $A \notin$ DD.

É fácil provar que as matrizes diagonalmente dominantes são invariantes em relação às submatrizes principais, isto é,

Teorema 3.4

1.
- $A \in \text{RDD}(\text{CDD}, \text{DD}) \Leftrightarrow A_{JJ} \in \text{RDD}(\text{CDD}, \text{DD})$*, para todo o subconjunto* $J \subseteq \{1, \ldots, n\}$.
- $A \in \text{RDD}_+(\text{CDD}_+, \text{DD}_+) \Leftrightarrow A_{JJ} \in \text{RDD}_+(\text{CDD}_+, \text{DD}_+)$*, para todo o subconjunto* $J \subseteq \{1, \ldots, n\}$.
- $A \in \text{SRDD}(\text{SCDD}, \text{SDD}) \Leftrightarrow A_{JJ} \in \text{SRDD}(\text{SCDD}, \text{SDD})$*, para todo o subconjunto* $J \subseteq \{1, \ldots, n\}$.
- $A \in \text{SRDD}_+(\text{SCDD}_+, \text{SDD}_+) \Leftrightarrow A_{JJ} \in \text{SRDD}_+(\text{SCDD}_+, \text{SDD}_+)$*, para todo o subconjunto* $J \subseteq \{1, \ldots, n\}$.

2.
- $A \in \text{SRDD}(\text{SCDD}) \Rightarrow a_{ii} \neq 0$*, para todo o* $i = 1, \ldots, n$.
- $A \in \text{RDD}(\text{CDD})$ *e* $a_{ii} = 0 \Rightarrow a_{ij} = 0 (a_{ji} = 0)$*, para* $j \neq i$.

Demonstração: 1. Iremos apenas demonstrar o caso em que $A \in$ RDD. Aplicando a definição, tem–se

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \ i = 1, \ldots, n$$

Sejam $J \subseteq \{1, \ldots, n\}$ e $K = \{1, \ldots, n\} - J$. Então

$$\begin{aligned} |a_{ii}| &\geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \\ &= \sum_{\substack{j \in J \\ j \neq i}} |a_{ij}| + \sum_{\substack{j \in K \\ j \neq i}} |a_{ij}| \\ &\geq \sum_{\substack{j \in J \\ j \neq i}} |a_{ij}|. \end{aligned}$$

Portanto $A_{JJ} \in$ RDD.

2. Suponhamos que $A \in$ SRDD. Então

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \geq 0$$

Portanto $a_{ii} \neq 0$. Por outro lado, se $A \in$ RDD e $a_{ii} = 0$, então

$$0 \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \geq 0$$

Donde

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| = 0$$

e portanto $a_{ij} = 0$ para $i \neq j$.

Como consequência imediata das definições de matrizes diagonalmente dominantes, facilmente se conclui a invariância dessas matrizes em relação a permutações principais de linhas e colunas, isto é, tem-se

Teorema 3.5

1. $A \in \mathrm{SRDD}(\mathrm{SDD}, \mathrm{SRDD}_+, \mathrm{SDD}_+) \Rightarrow P^T AP \in \mathrm{SRDD}(\mathrm{SDD}, \mathrm{SRDD}_+, \mathrm{SDD}_+)$ *para qualquer matriz de permutação* P.
2. $A \in \mathrm{RDD}(\mathrm{DD}, \mathrm{RDD}_+, \mathrm{DD}_+) \Rightarrow P^T AP \in \mathrm{RDD}(\mathrm{DD}, \mathrm{RDD}_+, \mathrm{DD}_+)$ *para qualquer matriz de permutação* P.

Para demonstrar a invariância de Complementos de Schur para estas classes de matrizes, vamos começar por abordar um caso particular.

Teorema 3.6

1. $A \in \mathrm{SRDD}(\mathrm{SDD}, \mathrm{SRDD}_+, \mathrm{SDD}_+) \Rightarrow (A|a_{11}) \in \mathrm{SRDD}(\mathrm{SDD}, \mathrm{SRDD}_+, \mathrm{SDD}_+)$.
2. $A \in \mathrm{RDD}(\mathrm{DD}, \mathrm{RDD}_+, \mathrm{DD}_+) \Rightarrow a_{11} = 0$ *ou* $(A|a_{11}) \in \mathrm{RDD}(\mathrm{DD}, \mathrm{RDD}_+, \mathrm{DD}_+)$.

Demonstração: Como as demonstrações das duas propriedades são semelhantes, concentrar-nos-emos apenas na primeira. Se $A \in \mathrm{SRDD}$, então $a_{11} \neq 0$ e podemos considerar a matriz $\bar{A}_1 = (A|a_{11})$. Os elementos dessa matriz são dados por

$$\bar{a}_{ij} = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}, \quad i,j = 2,\ldots,n$$

Para provar que $\bar{A}_1 \in \mathrm{SRDD}$, temos que mostrar que

$$|\bar{a}_{ii}| > \sum_{\substack{j=2\\ j\neq i}}^{n} |\bar{a}_{ij}|, \quad i,j = 2,\ldots,n$$

Mas

$$\begin{aligned}\sum_{\substack{j=2\\ j\neq i}}^{n} |\bar{a}_{ij}| &= \sum_{\substack{j=2\\ j\neq i}}^{n} \left| a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}} \right| \\ &\leq \sum_{\substack{j=2\\ j\neq i}}^{n} |a_{ij}| + \frac{|a_{i1}|}{|a_{11}|} \sum_{\substack{j=2\\ j\neq i}}^{n} |a_{1j}|\end{aligned}$$

Como $A \in \mathrm{SRDD}$, então

$$\sum_{\substack{j=2\\ j\neq i}}^{n} |a_{ij}| < |a_{ii}| - |a_{i1}|$$

$$\sum_{\substack{j=2\\ j\neq i}}^{n} |a_{1j}| < |a_{11}| - |a_{1i}|$$

Portanto

$$\begin{aligned}\sum_{\substack{j=2\\ j\neq i}}^{n} |\bar{a}_{ij}| &< |a_{ii}| - |a_{i1}| + \frac{|a_{i1}|}{|a_{11}|}\left(|a_{11}| - |a_{1i}|\right)\\ &= |a_{ii}| - \frac{|a_{i1}a_{1i}|}{|a_{11}|}\\ &\leq \left|a_{ii} - \frac{a_{i1}a_{1i}}{a_{11}}\right| = |\bar{a}_{ii}|\end{aligned}$$

Se $A \in$ SDD então $A \in$ SRDD e $A^T \in$ SRDD. Portanto $(A|a_{11}) \in$ SRDD e $(A^T|a_{11}) \in$ SRDD. Além disso $(A^T|a_{11}) = (A|a_{11})^T$ e portanto $(A|a_{11}) \in$ SRDD e $(A|a_{11})^T \in$ SRDD pelo que $(A|a_{11}) \in$ SDD.

Para completar a demonstração da parte 1. basta provar que se $A \in \text{SRDD}_+$, então os elementos diagonais de $(A|a_{11})$ são todos positivos. Suponhamos que existe um $i \geq 2$ tal que

$$\bar{a}_{ii} = a_{ii} - \frac{a_{i1}a_{1j}}{a_{11}} \leq 0$$

Como $a_{11} > 0$, então tem-se

$$a_{11}a_{ii} \leq a_{i1}a_{1j}$$

Mas $a_{11}a_{ii} > 0$ e portanto $a_{i1}a_{1j}$ tem de ser positivo e

$$a_{i1}a_{1j} = |a_{i1}a_{1j}| = |a_{i1}||a_{1j}|$$

Donde

$$a_{11}a_{ii} \leq |a_{i1}||a_{1j}|$$

e isso é impossível por $A \in \text{SRDD}_+$.

A partir deste teorema e da fórmula de Schur é possível provar o seguinte resultado:

Teorema 3.7

1. $A \in \text{SRDD}(\text{SRDD}_+) \Rightarrow \det(A) \neq 0 (> 0)$.
2. $A \in \text{RDD}_+ \Rightarrow \det(A) \geq 0$.

Demonstração: Iremos provar o caso de $A \in$ SRDD e usaremos o método de indução sobre a ordem n de A.

Para $n = 1$ tem-se $|a_{11}| > 0$ e portanto $\det(A) \neq 0$. Suponhamos agora que o resultado é válido para matrizes de ordem $p \leq n-1$. Se A é uma matriz SRDD de ordem n então $a_{11} \neq 0$ e, pelo teorema anterior, $(A|a_{11}) \in$ SRDD. Pela Fórmula de Schur vem

$$\det(A) = a_{11}\det(A|a_{11})$$

Como $(A|a_{11})$ é uma matriz de ordem $n-1$ então pela hipótese de indução tem-se

$$\det(A|a_{11}) \neq 0.$$

Donde $\det(A) \neq 0$.

Estamos agora em condições de estabelecer a invariância do Complemento de Schur relativamente a estas classes de matrizes.

Teorema 3.8

1. $A \in \text{SRDD}(\text{SDD}, \text{SRDD}_+, \text{SDD}_+) \Rightarrow (A|A_{II}) \in \text{SRDD}(\text{SDD}, \text{SRDD}_+, \text{SDD}_+)$, *para qualquer* $I \subseteq \{1, \ldots, n\}$.
2. $A \in \text{RDD}(\text{DD}, \text{RDD}_+, \text{DD}_+)$ *e* A_{II} *não singular* $\Rightarrow (A|A_{II}) \in \text{RDD}(\text{DD}, \text{RDD}_+, \text{DD}_+)$ *para qualquer* $I \subseteq \{1, \ldots, n\}$.

Demonstração: Provaremos apenas a primeira implicação. A segunda é semelhante e é deixada como exercício. Se $A \in \text{SRDD}$ então $A_{II} \in \text{SRDD}$. Portanto $\det(A_{II}) \neq 0$ e A_{II} é não singular. Logo existe $(A|A_{II})$. Para demonstrar que $(A|A_{II}) \in \text{SRDD}$ usaremos a indução sobre o número de elementos k de I. O teorema 3.5 permite-nos considerar I como sendo o conjunto $\{1, \ldots, k\}$. Para $k = 1$ estamos perante um resultado já demonstrado (teorema 3.6). Suponhamos, como hipótese de indução, que a implicação é verdadeira para matrizes de ordem até $k - 1$. Pela Fórmula do Quociente tem-se

$$(A|A_{II}) = ((A|a_{11})|(A_{II}|a_{11}))$$

Mas $(A|a_{11}) \in \text{SRDD}$ e $(A_{II}|a_{11}) \in \text{SRDD}$. Como $(A_{II}|a_{11})$ é de ordem $k-1$ então pela hipótese de indução tem-se $((A|a_{11})|(A_{II}|a_{11})) \in \text{SRDD}$, ou seja, $(A|A_{II}) \in \text{SRDD}$.

De notar que, ao contrário do que se passa com o Complemento de Schur, as matrizes destas classes não são invariantes em relação a operações pivotais principais. A título de exemplo, consideremos a matriz

$$A = \begin{bmatrix} 3 & 2 \\ 1 & 3 \end{bmatrix}$$

Verifica-se facilmente que $A \in \text{SDD}_+$ e que portanto pertence a todas as outras classes estudadas nesta secção. Se efectuarmos uma operação pivotal principal com pivot a_{11}, obtemos

$$\bar{A} = \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} \\ \\ \frac{1}{3} & \frac{7}{3} \end{bmatrix}$$

que não pertence a nenhuma das classes referidas.

3.3 Matrizes Positivas Definidas e Positivas Semi–Definidas

Definição 3.2 *Seja* A *uma matriz quadrada.*

- A *matriz* A *é positiva definida* ($A \in \text{PD}$) *se e só se* $x^T Ax > 0$ *para todo o* $x \neq 0$.
- A *matriz* A *é positiva semi–definida* ($A \in \text{PSD}$) *se e só se* $x^T Ax \geq 0$ *para todo o* x.

Além disso, notaremos por SPD e SPSD as classes das matrizes simétricas PD e PSD respectivamente. Como consequência imediata das definições tem-se

$$\begin{array}{ccc} \text{PD} & \subset & \text{PSD} \\ \cup & & \cup \\ \text{SPD} & \subset & \text{SPSD} \end{array}$$

Para qualquer matriz A tem-se

$$A = \frac{1}{2}(A + A^T) + \frac{1}{2}(A - A^T)$$

Além disso, para qualquer vector x vem

$$x^T(A - A^T)x = x^TAx - x^TA^Tx = x^TAx - \left(x^TAx\right)^T = x^TAx - x^TAx = 0$$

Portanto

$$A \in \text{PD(SPD)} \Leftrightarrow A + A^T \in \text{SPD(SPSD)} \tag{3.15}$$

As definições apresentadas mostram que para verificar se uma matriz A é PD ou PSD é necessário estudar o sinal da forma quadrática x^TAx para todos os vectores x, o que é bastante difícil de fazer na prática. Como veremos mais adiante, é no entanto possível desenvolver processos bastante eficientes para esse efeito.

Tal como as matrizes diagonalmente dominantes, também as submatrizes principais destas classes pertencem a estas classes, isto é, tem-se

Teorema 3.9

1. $A \in \text{PD(PSD, SPD, SPSD)} \Leftrightarrow A_{II} \in \text{PD(PSD, SPD, SPSD)}$ *para todo o* $I \subseteq \{1, \ldots, n\}$.
2. *(a)* $A \in \text{PD(SPD)} \Rightarrow a_{ii} > 0$, *para todo o* $i = 1, \ldots, n$.
   (b) $A \in \text{PSD(SPSD)} \Rightarrow a_{ii} \geq 0$, *para todo o* $i = 1, \ldots, n$.
3. *(a)* $A \in \text{PSD}$ *e* $a_{ii} = 0 \Rightarrow a_{ij} + a_{ji} = 0$ *para qualquer* j.
   (b) $A \in \text{SPSD}$ *e* $a_{ii} = 0 \Rightarrow a_{ij} = 0$ *para qualquer* j.
4. $A \in \text{PD(SPD)} \Rightarrow A$ *é não singular.*

Demonstração: As partes 1. e 2. são bastante fáceis de provar, pelo que a sua demonstração é deixada como exercício. Em relação à parte 3. seja $a_{ii} = 0$ e $j \neq i$ um índice qualquer. De 1. vem

$$M = \begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} = \begin{bmatrix} 0 & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \in \text{PSD}$$

Seja $\alpha = a_{ij} + a_{ji}$ e provemos que $\alpha = 0$. Para isso consideremos o vector $x = (1, \beta)^T$, com β um número real qualquer. Então

$$\begin{aligned} x^TMx &= \begin{bmatrix} 1 \\ \beta \end{bmatrix}^T \begin{bmatrix} 0 & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix} \begin{bmatrix} 1 \\ \beta \end{bmatrix} \\ &= \beta^2 a_{jj} + \beta(a_{ji} + a_{ij}) \\ &= \beta^2 a_{jj} + \beta\alpha \end{aligned}$$

Mas $x^TMx \geq 0$ e portanto $\beta^2 a_{jj} + \beta\alpha \geq 0$. Tal acontece apenas se $\alpha^2 \leq 0$, ou seja, se $\alpha = 0$.

O caso de $A \in \text{SPSD}$ é consequência imediata deste último resultado e do facto de A ser simétrica.

A demonstração de 4. é feita por absurdo. Se A é uma matriz singular, então existe pelo menos um vector $x \neq 0$ tal que $Ax = 0$. Portanto $x^TAx = 0$ e isso contradiz o facto de A ser PD.

É de notar que uma matriz PSD ou SPSD pode ser singular. Com efeito, seja

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Então para qualquer vector $x = [x_1 x_2]^T$ tem-se

$$x^TAx = [x_1 x_2] \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = (x_1 + x_2)^2 \geq 0$$

Donde $A \in$ PSD. Além disso A é singular pois o seu determinante é igual a zero.

Se P é uma matriz de permutação então

$$x^T P^T A P x = (Px)^T A (Px)$$

e portanto obtemos o seguinte teorema

Teorema 3.10 *Se P é uma matriz de permutação, então:*

1. $A \in$ PD(SPD) $\Rightarrow P^T A P \in$ PD(SPD).
2. $A \in$ PSD(SPSD) $\Rightarrow P^T A P \in$ PSD(SPSD).

A próxima propriedade mostra que as matrizes PD e PSD são invariantes em relação a operações pivotais principais.

Teorema 3.11 *Se $A \in$ PD(PSD) e $\bar{A}$ é uma transformada principal de A, então $\bar{A} \in$ PD(PSD).*

Demonstração : Suponhamos que $\bar{A}$ é obtida de A por uma operação pivotal principal com pivot A_{II}. Devido ao resultado anterior podemos supor que o conjunto I é constituído pelas primeiras k linhas de A. Portanto podemos escrever o sistema $y = Ax$ na forma

$$\begin{bmatrix} y_I \\ y_J \end{bmatrix} = \begin{bmatrix} A_{II} & A_{IJ} \\ A_{JI} & A_{JJ} \end{bmatrix} \begin{bmatrix} x_I \\ x_J \end{bmatrix}$$

Se efectuarmos uma operação pivotal principal com pivot A_{II}, obtemos

$$\begin{bmatrix} x_I \\ y_J \end{bmatrix} = \begin{bmatrix} \bar{A}_{II} & \bar{A}_{IJ} \\ \bar{A}_{JI} & \bar{A}_{JJ} \end{bmatrix} \begin{bmatrix} y_I \\ x_J \end{bmatrix}$$

Então

$$\begin{aligned} x^T A x &= \begin{bmatrix} x_I \\ x_J \end{bmatrix}^T \begin{bmatrix} A_{II} & A_{IJ} \\ A_{JI} & A_{JJ} \end{bmatrix} \begin{bmatrix} x_I \\ x_J \end{bmatrix} = \begin{bmatrix} x_I \\ x_J \end{bmatrix}^T \begin{bmatrix} y_I \\ y_J \end{bmatrix} \\ &= \begin{bmatrix} y_I \\ x_J \end{bmatrix}^T \begin{bmatrix} x_I \\ y_J \end{bmatrix} = \begin{bmatrix} y_I \\ x_J \end{bmatrix}^T \begin{bmatrix} \bar{A}_{II} & \bar{A}_{IJ} \\ \bar{A}_{JI} & \bar{A}_{JJ} \end{bmatrix} \begin{bmatrix} y_I \\ x_J \end{bmatrix} = w^T \bar{A} w \end{aligned}$$

Donde $\bar{A} \in$ PD(PSD) se $A \in$ PD(PSD).

É de notar que esta propriedade não é válida para matrizes SPD e SPSD, pois as operações pivotais não preservam a simetria.

O Complemento de Schur de uma matriz A_{II} em A é uma submatriz principal da transformada principal de A que se obtém por uma operação pivotal com pivot A_{II}. Além disso o Complemento de Schur de uma matriz simétrica é também uma matriz simétrica. Então verifica-se o seguinte resultado:

Teorema 3.12

1. $A \in$ PD(SPD) $\Rightarrow (A|A_{II}) \in$ PD(SPD) *para qualquer* $I \subseteq \{1, \ldots, n\}$.
2. $A \in$ PSD(SPSD) *e* A_{II} *não singular* $\Rightarrow (A|A_{II}) \in$ PSD(SPSD) *para qualquer* $I \subseteq \{1, \ldots, n\}$.

É de notar que em 1. A_{II} é não singular para qualquer conjunto I, devido ao teorema 3.9.

Como consequência dos três últimos teoremas, podemos ainda estabelecer as seguintes propriedades:

Teorema 3.13 *Se $A \in$ PD(PSD), então:*

1. $\det(A) > 0 (\geq 0)$.
2. $A^{-1} \in$ PD(PSD *se A é não singular*).
3. *Todos os menores principais de A são positivos (não negativos).*

A demonstração de 1. é semelhante à do teorema 3.7. A segunda propriedade é imediata, pois A^{-1} é a transformada principal de A que se obtém de A por uma operação pivotal com pivot A. Finalmente o resultado 3. é consequência imediata de 1. e do teorema 3.9.

Um resultado clássico das matrizes simétricas positivas definidas e semi–definidas diz respeito ao sinal dos valores próprios dessas matrizes. Essa propriedade é a seguir apresentada e demonstrada como consequência das propriedades anteriores.

Teorema 3.14 *$A \in$ SPD(SPSD) se e só se os seus valores próprios são positivos (não negativos).*

Demonstração: Provemos apenas o caso de $A \in$ SPD, pois a demonstração para $A \in$ SPSD é semelhante. Seja então A uma matriz SPD. Como A é simétrica, então todos os seus valores próprios são números reais. Se $\lambda \leq 0$ é valor próprio de A, existe um vector $x \neq 0$ tal que $Ax = \lambda x$. Portanto

$$x^T Ax = \lambda x^T x \leq 0$$

o que é impossível por A ser SPD. Isso demonstra a implicação $\Rightarrow$.

Suponhamos agora que todos os valores próprios $\lambda_i, i = 1, \ldots, n$ de A são positivos. Se x^i são os vectores próprios associados a λ_i e se Q é a matriz constituída pelos vectores coluna x^i, então tem-se

$$A = Q\Lambda Q^T$$

com Λ a matriz diagonal com elementos diagonais $\lambda_1, \ldots, \lambda_n$. Portanto

$$x^T Ax = x^T Q\Lambda Q^T x = (Q^T x)^T \Lambda (Q^T x) > 0$$

e $A \in$ SPD.

Como o determinante da matriz simétrica $A + A^T$ é o produto dos seus valores próprios, e A e $A + A^T$ pertencem 'a mesma classe de matrizes, então verifica-se o seguinte resultado:

Teorema 3.15 *Se $A \in$ PSD e $A + A^T$ é não singular, então $A \in$ PD.*

Portanto matrizes SPSD não singulares são SPD.

3.4 Matrizes P e P_0

Definição 3.3 *Seja A uma matriz quadrada.*

- *A matriz A é de classe* P *($A \in$ P) se e só se os menores principais de A são positivos.*
- *A matriz A é de classe* P_0 *($A \in P_0$) se e só se os menores principais de A são não negativos.*

Portanto, se A é uma matriz de ordem n, tem-se

$$A \in \mathrm{P} \Leftrightarrow \det(A_{II}) > 0 \text{ para todo o } I \subseteq \{1, \ldots, n\}$$

$$A \in \mathrm{P}_0 \Leftrightarrow \det(A_{II}) \geq 0 \text{ para todo o } I \subseteq \{1, \ldots, n\}$$

Assim por exemplo

$$\begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 5 \end{bmatrix} \in \mathrm{P}$$

pois:

$$a_{11} = 1 > 0, a_{22} = 2 > 0, a_{33} = 5 > 0$$

$$\det \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} = 1 > 0, \det \begin{bmatrix} 1 & 2 \\ 1 & 5 \end{bmatrix} = 3 > 0, \det \begin{bmatrix} 2 & 1 \\ 1 & 5 \end{bmatrix} = 9 > 0$$

$$\det(A) = 3 > 0$$

Das definições apresentadas conclui-se que toda a matriz P é necessariamente não singular, enquanto que matrizes P_0 podem ser singulares ou não singulares. Além disso, para verificar se uma matriz A é P ou P_0 é necessário determinar o sinal de $2^n - 1$ determinantes de submatrizes principais de A, o que é extremamente trabalhoso mesmo para valores relativamente pequenos de n. Seguidamente iremos estudar algumas das principais propriedades das matrizes destas classes. Assim, como consequência imediata das definições, tem-se

Teorema 3.16

1. - $A \in \mathrm{P} \Rightarrow A_{II} \in \mathrm{P}$ *para todo o* $I \subseteq \{1, \ldots, n\}$.
   - $A \in \mathrm{P}_0 \Rightarrow A_{II} \in \mathrm{P}_0$ *para todo o* $I \subseteq \{1, \ldots, n\}$.
2. - $A \in \mathrm{P} \Rightarrow a_{ii} > 0$ *para todo o* $i = 1, \ldots, n$.
   - $A \in \mathrm{P}_0 \Rightarrow a_{ii} \geq 0$ *para todo o* $i = 1, \ldots, n$.

Teorema 3.17

1. - $A \in \mathrm{P} \Leftrightarrow A^T \in \mathrm{P}$.
   - $A \in \mathrm{P}_0 \Leftrightarrow A^T \in \mathrm{P}_0$.
2. *Se* $A \in \mathrm{P}(\mathrm{P}_0)$*, então* $Q^T A Q \in \mathrm{P}(\mathrm{P}_0)$*, com* Q *uma matriz de permutação.*

O seguinte resultado mostra a invariância das matrizes destas classes em relação a operações pivotais principais.

Teorema 3.18 *Se* $A \in \mathrm{P}(\mathrm{P}_0)$ *e* $\bar{A}$ *é uma transformada principal de* A*, então* $\bar{A} \in \mathrm{P}(\mathrm{P}_0)$.

Demonstração: Sejam $J, L \subseteq \{1, \ldots, n\}$ e consideremos a submatriz principal $\bar{A}_{LL}$ de $\bar{A}$. Pelo teorema 3.1, tem-se

$$\det(\bar{A}_{LL}) = \frac{\det(A_{J\Delta L, J\Delta L})}{\det(A_{JJ})}$$

em que $J\Delta L = J \cup L - J \cap L$. Como $\det(A_{JJ}) > 0$ e $\det(A_{J\Delta L, J\Delta L}) > 0$, então $\det(\bar{A}_{LL}) > 0$ para todo o $L \subseteq \{1, \ldots, n\}$. Logo $\bar{A} \in \mathrm{P}$.

Como consequência desta propriedade e do teorema 3.16 podemos estabelecer o seguinte resultado:

Teorema 3.19

1. $A \in \mathrm{P} \Rightarrow (A|A_{JJ}) \in \mathrm{P}$ *para qualquer* $J \subseteq \{1,\ldots,n\}$.
2. $A \in \mathrm{P}_0$ *e* A_{JJ} *não singular* $\Rightarrow (A|A_{JJ}) \in \mathrm{P}_0$ *para qualquer* $J \subseteq \{1,\ldots,n\}$.

Para terminar esta secção provemos a seguinte propriedade:

Teorema 3.20 $A \in \mathrm{P}$ *se e só se para todo o vector* $x \neq 0$ *existe um índice* i *tal que* $x_i y_i > 0$ *para* $y = Ax$.

Demonstração: Seja x um vector não nulo qualquer, $y = Ax$ e $I \neq \emptyset$ um conjunto de índices tal que $x_i \neq 0$ para $i \in I$. Suponhamos por absurdo que $x_i y_i \leq 0$ para $i \in I$. Então existe uma matriz diagonal D_{II} de elementos não negativos tal que $y_I = -D_{II}x_I$. Como $A_{II}x_I = y_I$, então

$$-D_{II}x_I = A_{II}x_I$$

ou seja

$$(A_{II} + D_{II})x_I = 0$$

com $x_I \neq 0$. Portanto a matriz $A_{II} + D_{II}$ é singular o que é impossível por $\det(A_{II}) > 0$.

Como $\det(A_{II})$ tem o sinal do produto dos seus valores próprios reais, então a implicação $\Rightarrow$ fica demonstrada se provarmos que qualquer valor próprio real λ de A_{II} é positivo. Mas se λ é um valor próprio real de A_{II}, então $A_{II}x = \lambda x$ para certo vector $x \neq 0$. Além disso por hipótese existe i tal que $x_i \neq 0$ e $x_i\,(A_{II}x)_i > 0$. Donde $(\lambda x)_i\, x_i > 0$ e $\lambda x_i^2 > 0$. Portanto $\lambda > 0$ e isso demonstra a implicação.

3.5 Matrizes S

Definição 3.4 *A matriz* A *diz-se da Classe* S ($A \in \mathrm{S}$) *se existir um vector* $x \geq 0$ *tal que* $Ax > 0$.

Da definição apresentada conclui-se que mostrar que uma matriz A é S consiste em encontrar um vector $0 \neq x \geq 0$ que satisfaça $Ax > 0$. Em certos casos esse problema é muito simples. Assim por exemplo, $A \in \mathrm{S}$ se tem pelo menos uma coluna positiva. Com efeito, se i é o índice dessa coluna, então qualquer vector x com todas as componentes nulas à excepção de x_i satisfaz

$$Ax = x_i A_{\cdot i} > 0$$

Daqui também se conclui que toda a matriz positiva é S. Contudo, em geral, a determinação de um tal vector não se afigura assim tão simples. Com efeito, é fácil de ver, usando a teoria da dualidade da programação linear, que esse problema se reduz a mostrar que o conjunto convexo

$$K = \{x : A^T x \leq 0, e^T x = 1\}$$

é vazio, onde e é um vector com todas as componentes iguais a um. Assim tem-se

$$A \in \mathrm{S} \Leftrightarrow K = \emptyset$$

Além disso, a verificação de que $K = \emptyset$ pode ser feita usando técnicas de programação linear, que estão fora do âmbito deste curso.

O seguinte resultado é consequência imediata da definição e a sua demonstração é deixada como exercício.

Teorema 3.21 *Se* $A \in \mathrm{S}$, *então existe um vector* $y > 0$ *tal que* $Ay > 0$.

Tal como para as classes PD e P, podemos garantir a invariância das matrizes S em relação a operações pivotais principais:

Teorema 3.22 *Se $\bar{A}$ é uma transformada principal de $A \in$ S, então $\bar{A} \in$ S.*

Demonstração: Seja

$$A = \begin{bmatrix} A_{II} & A_{IJ} \\ A_{JI} & A_{JJ} \end{bmatrix}$$

e suponhamos que $\bar{A}$ é a transformada principal de A com pivot não singular A_{II}. Então

$$\bar{A} = \begin{bmatrix} \bar{A}_{II} & \bar{A}_{IJ} \\ \bar{A}_{JI} & \bar{A}_{JJ} \end{bmatrix} = \begin{bmatrix} A_{II}^{-1} & -A_{II}^{-1}A_{IJ} \\ A_{JI}A_{II}^{-1} & (A|A_{II}) \end{bmatrix}$$

Por outro lado, se $A \in$ S, então o sistema

$$\begin{cases} y_I &= A_{II}x_I + A_{IJ}x_J \\ y_J &= A_{JI}x_I + A_{JJ}x_J \end{cases}$$

tem uma solução $(\bar{x}_I, \bar{x}_J, \bar{y}_I, \bar{y}_J) > 0$. Para mostrar que $\bar{A} \in$ S basta-nos mostrar que

$$\bar{A} \begin{bmatrix} \bar{y}_I \\ \bar{x}_J \end{bmatrix} = \begin{bmatrix} \bar{A}_{II} & \bar{A}_{IJ} \\ \bar{A}_{JI} & \bar{A}_{JJ} \end{bmatrix} \begin{bmatrix} \bar{y}_I \\ \bar{x}_J \end{bmatrix} > 0$$

Mas,

$$\begin{aligned} \bar{A}_{II}\bar{y}_I + \bar{A}_{IJ}\bar{x}_J &= A_{II}^{-1}\bar{y}_I - A_{II}^{-1}A_{IJ}\bar{x}_J \\ &= A_{II}^{-1}\left(A_{II}\bar{x}_I + A_{IJ}\bar{x}_J\right) - A_{II}^{-1}A_{IJ}\bar{x}_J \\ &= \bar{x}_I + A_{II}^{-1}A_{IJ}\bar{x}_J - A_{II}^{-1}A_{IJ}\bar{x}_J \\ &= \bar{x}_I > 0 \end{aligned}$$

e

$$\begin{aligned} \bar{A}_{JI}\bar{y}_I + \bar{A}_{JJ}\bar{x}_J &= A_{JI}A_{II}^{-1}\bar{y}_I + (A|A_{II})\bar{x}_J \\ &= A_{JI}A_{II}^{-1}\left(A_{II}\bar{x}_I + A_{IJ}\bar{x}_J\right) + \left(A_{JJ} - A_{JI}A_{II}^{-1}A_{IJ}\right)\bar{x}_J \\ &= A_{JI}\bar{x}_I + A_{JI}A_{II}^{-1}A_{IJ}\bar{x}_J + A_{JJ}\bar{x}_J - A_{JI}A_{II}^{-1}A_{IJ}\bar{x}_J \\ &= A_{JI}\bar{x}_I + A_{JJ}\bar{x}_J = \bar{y}_J > 0 \end{aligned}$$

Donde $\bar{A} \in$ S.

Ao contrário das classes anteriormente estudadas, não é possível garantir que uma submatriz principal e o Complemento de Schur de uma matriz S pertençam a essa classe. Com efeito, consideremos o seguinte exemplo:

$$A = \begin{bmatrix} 1 & -1 \\ 1 & -2 \end{bmatrix}$$

Se $x = [1, 0]^T$ então $Ax = [1, 1]^T > 0$ e $A \in$ S. Contudo $[a_{22}] \notin$ S e $(A|a_{11}) = [-1] \notin$ S.

O próximo resultado mostra que a classe P está contida na classe S.

Teorema 3.23 *Se $A \in$ P, então $A \in$ S.*

Demonstração: Suponhamos, por absurdo, que $A \notin$ S. Então, pela teoria da dualidade da programação linear, existe um vector $0 \neq x \geq 0$ tal que $A^T x \leq 0$ [Murty, cap. 4]. Portanto, pelo teorema 3.20, $A^T \notin$ P e também $A \notin$ P.

3.6 Matrizes Z e K

Definição 3.5

- *A matriz A diz-se da classe* Z ($A \in$ Z) *se* $a_{ij} \leq 0$ *para* $i \neq j$.
- *A matriz A diz-se da classe* K ($A \in$ K) *se A pertence às classes* Z *e* P.

As matrizes K também são conhecidas por matrizes M não singulares e tem-se

$$\text{K} = \text{Z} \cap \text{P}.$$

Como consequência da definição e do teorema 3.16, podemos enunciar o seguinte resultado:

Teorema 3.24

1. $A \in \text{Z} \Rightarrow A_{II} \in \text{Z}$ *para todo o* $I \subseteq \{1, \ldots, n\}$.
2. $A \in \text{K} \Rightarrow A_{II} \in \text{K}$ *para todo o* $I \subseteq \{1, \ldots, n\}$.

Transformadas principais de matrizes Z e K não pertencem necessariamente a essas classes. Com efeito consideremos a matriz

$$\begin{bmatrix} 9 & -1 \\ -1 & 9 \end{bmatrix} \in \text{Z(K)}$$

A sua transformada principal obtida pela operação pivotal com pivot a_{11}, é dada por

$$\bar{A} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} \\ -\frac{1}{9} & \frac{80}{9} \end{bmatrix}$$

e $\bar{A} \notin$ Z(K).

Como vimos atrás, K é a intersecção das classes de matrizes Z e P. Seguidamente iremos estabelecer que K também pode ser caracterizada pela intersecção das classes Z e S. Para isso precisamos dos seguintes resultados:

Teorema 3.25 *Se* $A \in \text{Z} \cap \text{S}$, *então*

1. $A_{II} \in \text{Z} \cap \text{S}$ *para todo o* $I \subseteq \{1, \ldots, n\}$.
2. $(A|a_{11}) \in \text{Z} \cap \text{S}$.
3. $\det(A) > 0$.

Demonstração: 1. Suponhamos sem perda de generalidade que I é dado por $I = \{1, \ldots, k\}$. Então podemos escrever

$$A = \begin{bmatrix} A_{II} & A_{IJ} \\ A_{JI} & A_{JJ} \end{bmatrix}$$

com $J = \{1, \ldots, n\} - I$. Se $A \in$ S então existe $x > 0$ tal que $Ax > 0$. Mas $A_{IJ} \leq 0$ pois $A \in$ Z. Portanto $A_{II}x_I > 0$, o que implica que $A_{II} \in$ S.

2. Se $A \in \mathrm{Z} \cap \mathrm{S}$, então $a_{11} > 0$. Seja

$$\bar{A} = \begin{bmatrix} a_{11} & -\frac{1}{a_{11}} c^T \\ \frac{1}{a_{11}} b & (A|a_{11}) \end{bmatrix}$$

a transformada principal de A obtida por uma operação pivotal principal com pivot a_{11}. Então $\bar{A} \in \mathrm{S}$, pelo teorema 3.22. Além disso, $b \leq 0$ e portanto tem de existir um vector $y > 0$ tal que $(A|a_{11})y > 0$. Portanto $(A|a_{11}) \in \mathrm{S}$. Como $(A|a_{11}) \in \mathrm{Z}$, então a propriedade está demonstrada.

3. A propriedade é demonstrada por indução sobre a ordem n da matriz A. Se $n = 1$, então $A = [a_{11}]$ e $\det(A) = a_{11} > 0$. Suponhamos que a propriedade é verdadeira para matrizes de ordem $n - 1$. Então, por 2., $\det(A|a_{11}) > 0$ e, pela fórmula de Schur,

$$\det(A) = a_{11} \det(A|a_{11}) > 0$$

Como consequência destes resultados podemos estabelecer que as classes $\mathrm{Z} \cap \mathrm{P}$ e $\mathrm{Z} \cap \mathrm{S}$ são iguais.

Teorema 3.26 $\mathrm{K} = \mathrm{Z} \cap \mathrm{P} = \mathrm{Z} \cap \mathrm{S}$

Demonstração: Como por definição se tem $K = \mathrm{Z} \cap \mathrm{P}$ e, pelo teorema 3.23, $\mathrm{P} \subset \mathrm{S}$, então basta provar que $\mathrm{Z} \cap \mathrm{S} \subset \mathrm{Z} \cap \mathrm{P}$. Seja $A \in \mathrm{Z} \cap \mathrm{S}$. Então pelo teorema anterior $\det(A_{II}) > 0$, para todo o $I \subseteq \{1, \ldots, n\}$. Portanto $A \in \mathrm{P}$ e o resultado está provado.

Seguidamente apresentamos uma caracterização da classe K que é frequentemente usada como definição das matrizes K.

Teorema 3.27 $A \in \mathrm{K} \Leftrightarrow A^{-1} \geq 0$ *e* $A \in \mathrm{Z}$

Demonstração: A implicação [$\Rightarrow$] demonstra-se por indução, usando a invariância do Complemento de Schur para matrizes P e é deixada como exercício. Para estabelecer a implicação [$\Leftarrow$], seja A uma matriz Z com inversa não negativa e x um vector positivo. Então existe $y = A^{-1}x \geq 0$ tal que

$$Ay = A(A^{-1}x) = x > 0$$

Portanto $A \in \mathrm{S}$ e também $A \in \mathrm{K}$.

Este teorema mostra que para verificar se uma matriz $A \in \mathrm{Z}$ é K basta calcular a sua inversa e estudar o sinal dos seus elementos. Note-se que esse processo é bastante mais eficiente do que verificar se $A \in \mathrm{P}$. Com efeito no primeiro caso há que efectuar n^3 operações e estudar o sinal de n^2 elementos, enquanto que a verificação de $A \in \mathrm{P}$ requer o cálculo de $2^n - 1$ determinantes. A título de exemplo, consideremos a matriz

$$A = \begin{bmatrix} 1 & -1 & -2 \\ 0 & 2 & -1 \\ -1 & -1 & 4 \end{bmatrix}$$

Como vimos anteriormente, o cálculo da matriz inversa pode ser feito a partir de três operações pivotais sucessivas de modo a transformar $y = Ax$ em $x = A^{-1}y$. Se escrevermos

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & -2 \\ 0 & 2 & -1 \\ -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

e efectuarmos a operação pivotal com pivot a_{11}, obtemos

$$\begin{bmatrix} x_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 2 & -1 \\ -1 & -2 & 2 \end{bmatrix} \begin{bmatrix} y_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Efectuando agora uma operação pivotal com pivot $\bar{a}_{22}$, tem-se

$$\begin{bmatrix} x_1 \\ x_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{5}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ x_3 \end{bmatrix}$$

Portanto a inversa A^{-1} de A obtém-se efectuando uma operação pivotal com pivot $\bar{a}_{33}$. Se tal acontecer, obtém-se

$$A^{-1} = \begin{bmatrix} \frac{7}{2} & 3 & \frac{5}{2} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ 1 & 1 & 1 \end{bmatrix}$$

Donde $A^{-1} \geq 0$ e $A \in \mathrm{K}$.

Com base nos resultados até agora provados é possível estabelecer a invariância do Complemento de Schur para matrizes K.

Teorema 3.28 *Se $A \in \mathrm{K}$, então $(A|A_{II}) \in \mathrm{K}$ para qualquer $I \subseteq \{1, \ldots, n\}$.*

Demonstração: Podemos escrever

$$A = \begin{bmatrix} A_{II} & A_{IJ} \\ A_{JI} & A_{JJ} \end{bmatrix}$$

com $J = \{1, \ldots, n\} - I$. Como $A_{II} \in \mathrm{K}$ então $A_{II}^{-1} \geq 0$ e

$$(A|A_{II}) = A_{JJ} - A_{JI}A_{II}^{-1}A_{IJ} \in \mathrm{Z}$$

Então $(A|A_{II}) \in \mathrm{K}$ pelo teorema 3.19.

3.7 Matrizes H

Definição 3.6 *Dada uma matriz $A \in \mathbb{R}^{n \times n}$, chama-se matriz companheira de A à matriz $M(A) = [b_{ij}]$ dada por*

$$b_{ij} = \begin{cases} |a_{ij}| & se \quad i = j \\ -|a_{ij}| & se \quad i \neq j \end{cases}$$

A matriz A é da classe H *($A \in \mathrm{H}$) se $M(A) \in \mathrm{K}$.*

Desta definição imediatamente se conclui que todos os elementos diagonais de uma matriz H são não nulos. Além disso, definimos a classe H_+ a partir de

$$A \in \mathrm{H}+ \Leftrightarrow A \in \mathrm{H} \text{ e } a_{ii} > 0, \ i = 1, \ldots, n$$

Ainda como consequência imediata das definições apresentadas, verifica-se o seguinte teorema

Teorema 3.29

(i) $A \in \mathrm{H}(\mathrm{H}_+) \Leftrightarrow A^T \in \mathrm{H}(\mathrm{H}_+)$

(ii) $A \in \mathrm{H}(\mathrm{H}_+) \Leftrightarrow A_{II} \in \mathrm{H}(\mathrm{H}_+)$ *para qualquer conjunto* $I \subseteq \{1, \ldots, n\}$

(iii) $A \in \mathrm{K} \Rightarrow A \in \mathrm{H}_+$

Seguidamente apresentamos caracterizações das matrizes H em termos das matrizes estritamente diagonalmente dominantes.

Teorema 3.30

(i) $A \in \mathrm{H}(\mathrm{H}_+) \Leftrightarrow$ *Existe uma matriz diagonal* D *de elementos diagonais positivos tal que* $AD \in \mathrm{SRDD}(\mathrm{SRDD}_+)$.

(ii) $A \in \mathrm{H}(\mathrm{H}_+) \Leftrightarrow$ *Existe uma matriz diagonal* D *de elementos diagonais positivos tal que* $DA \in \mathrm{SCDD}(\mathrm{SCDD}_+)$.

Demonstração: (i) Das definições apresentadas tem-se

$$A \in \mathrm{H} \Leftrightarrow M(A) \in \mathrm{K} \Leftrightarrow M(A) \in \mathrm{Z} \cap \mathrm{S}$$

Seja $M(A) = [b_{ij}]$. Como $M(A) \in \mathrm{S}$, existe um vector $d > 0$ tal que $M(A)d > 0$. Mas

$$\begin{aligned} M(A)d > 0 \quad &\Leftrightarrow \quad \sum_{j=1}^{n} b_{ij} d_j > 0, \ i = 1, \ldots, n \\ &\Leftrightarrow \quad b_{ii} d_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} b_{ij} d_j > 0, \ i = 1, \ldots, n \\ &\Leftrightarrow \quad |a_{ii}| d_i + \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(-|a_{ij}|\right) d_j > 0, \ i = 1, \ldots, n \end{aligned}$$

Como $d_j > 0$, para todo $j = 1, \ldots, n$, então $M(A)d > 0$ se e só se

$$|a_{ii} d_i| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij} d_j|, \ i = 1, 2, \ldots, n$$

o que significa que $AD \in \mathrm{SRDD}$.

Seja agora $A \in \mathrm{H}_+$. Então existe uma matriz diagonal D nas condições anteriores tal que $AD \in \mathrm{SRDD}$. Além disso, os elementos diagonais de AD são da forma $a_{ii} d_i$ e portanto são todos positivos. Donde $AD \in \mathrm{SRDD}_+$ e a equivalência fica demonstrada para o caso das matrizes H_+.

(ii) Iremos apenas demonstrar o caso de $A \in \mathrm{H}$, pois a equivalência para matrizes H_+ é estabelecida segundo a mesma linha de raciocínio apresentada em (i). Como $A \in \mathrm{H}$ se e só se $A^T \in \mathrm{H}$, então por (i) $A \in \mathrm{H}$ se e só se existe uma matriz diagonal D de elementos diagonais positivos tal que $A^T D \in \mathrm{SRDD}$, ou seja,

$$DA = D^T A \in \mathrm{SCDD}$$

Isso demonstra a propriedade. .

As equivalências do teorema anterior podem ser escritas na forma

$$\begin{aligned} |a_{ii}| d_i \quad &> \quad \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| d_j, \ i = 1, 2, \ldots, n \\ |a_{jj}| d_j \quad &> \quad \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}| d_i, \ j = 1, 2, \ldots, n \end{aligned} \tag{3.16}$$

Além disso são evidentes as seguintes inclusões

$$\mathrm{SRDD} \subset \mathrm{H} \supset \mathrm{SCDD}\ ,\ \mathrm{SRDD}_+ \subset \mathrm{H}_+ \supset \mathrm{SCDD}_+$$

Seja agora

$$A = \begin{bmatrix} A_{II} & A_{IJ} \\ A_{JI} & A_{JJ} \end{bmatrix}$$

uma matriz com A_{II} não singular e D uma matriz diagonal por blocos da forma

$$D = \begin{bmatrix} D_{II} & \\ & D_{JJ} \end{bmatrix}$$

com D_{II} e D_{JJ} não necessariamente diagonais e D_{II} não singular. Então

$$AD = \begin{bmatrix} A_{II}D_{II} & A_{IJ}D_{JJ} \\ A_{JI}D_{II} & A_{JJ}D_{JJ} \end{bmatrix}$$

Como $A_{II}D_{II}$ é não singular, então podemos calcular o Complemento de Schur $(AD|A_{II}D_{II})$ e tem-se

$$\begin{aligned} (AD|A_{II}D_{II}) &= A_{JJ}D_{JJ} - A_{JI}D_{II}\left(A_{II}D_{II}\right)^{-1}A_{IJ}D_{JJ} \\ &= \left(A_{JJ} - A_{JI}A_{II}^{-1}A_{IJ}\right)D_{JJ} \end{aligned}$$

Mas

$$\begin{aligned} (A|A_{II}) &= \left(A_{JJ} - A_{JI}A_{II}^{-1}A_{IJ}\right) \\ (D|D_{II}) &= D_{JJ} \end{aligned}$$

e portanto provámos a seguinte igualdade

$$(AD|A_{II}D_{II}) = (A|A_{II})(D|D_{II}) \tag{3.17}$$

Como consequência desta fórmula podemos estabelecer o seguinte teorema

Teorema 3.31

(i) $A \in \mathrm{H}(\mathrm{H}_+) \Rightarrow (A|a_{11}) \in \mathrm{H}(\mathrm{H}_+)$

(ii) $A \in \mathrm{H}(\mathrm{H}_+) \Rightarrow \det(A) \neq 0 (> 0)$

(iii) $A \in \mathrm{H}_+ \Rightarrow A \in \mathrm{P}$

(iv) $A \in \mathrm{H}(\mathrm{H}_+) \Rightarrow (A|A_{II}) \in \mathrm{H}(\mathrm{H}_+)$ *para qualquer conjunto* $I \subseteq \{1, \ldots, n\}$

Demonstração: (i) Se $A \in \mathrm{H}$, existe uma matriz diagonal D de elementos diagonais positivos tal que $AD \in \mathrm{SRDD}$. Então pelo teorema 3.6

$$(AD|a_{11}d_{11}) \in \mathrm{SRDD}$$

Mas pela igualdade (3.17) tem-se

$$(A|a_{11})(D|d_{11}) \in \mathrm{SRDD}$$

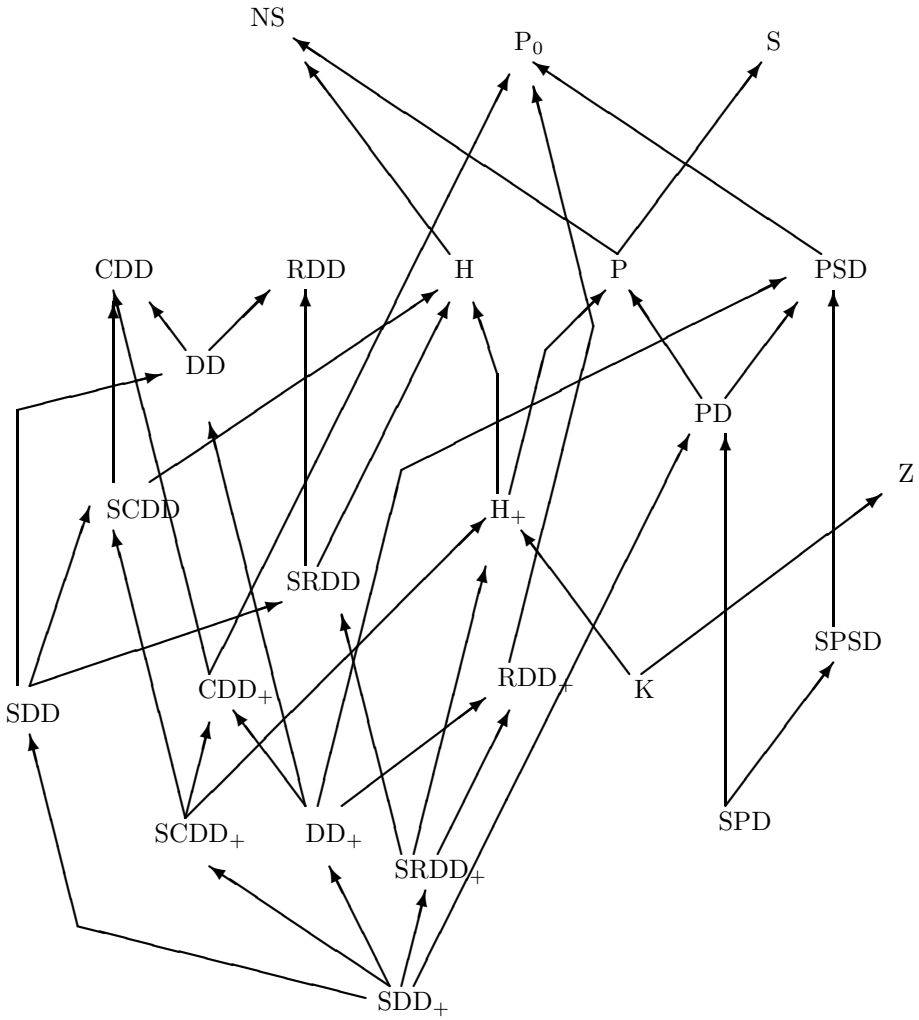
Como $(D|d_{11})$ é uma matriz diagonal de elementos diagonais positivos, então $(A|a_{11}) \in \mathrm{H}$, pelo teorema 3.30. A demonstração para $A \in \mathrm{H}_+$ é semelhante e não é por isso apresentada.

(ii) A demonstração é feita por indução sobre a ordem n da matriz A e baseia-se em (i) e na Fórmula de Schur.

(iii) É consequência de (ii) e do teorema 3.29 (ii).

(iv) A demonstração é feita por indução sobre o número de elementos $|I|$ do conjunto I e baseia-se em (i), (ii) e na Fórmula do Quociente.

Para finalizar este capítulo apresentamos um grafo dirigido que resume as inclusões entre classes de matrizes que foram discutidas. Os nós do grafo constituem classes de matrizes, enquanto as arestas representam relações de inclusão. Notemos ainda que nesse diagrama NS corresponde à classe das matrizes não singulares.



Exercícios

1.

(a) Seja A uma matriz SPD. Mostre que

$$|x^T Ay| \leq \sqrt{x^T Ax}\sqrt{y^T Ay},$$

para quaisquer x e y e use este resultado para mostrar que

$$||x||_A = \sqrt{x^T Ax}$$

satisfaz os axiomas da norma.

(b) Demonstre a igualdade de Cauchy-Schwarz:

$$|x^T y| \leq ||x||_2 ||y||_2$$

para quaisquer x e y.

(c) Mostre que

$$A = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 2 & 0 \\ 1 & 0 & 2 \end{bmatrix}$$

é SPD e calcule $||(1,1,0)||_A$.

2. Considere as seguintes matrizes

$$A = \begin{bmatrix} 2 & 0 & 1 \\ -1 & 2 & 1 \\ 3 & -2 & 1 \end{bmatrix}, B = \begin{bmatrix} -1 & 0 & 1 \\ 4 & 2 & 0 \\ -4 & 1 & 4 \end{bmatrix}, C = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 2 \\ -1 & 2 & 1 \end{bmatrix}$$

Calcule:

(a) $P_{12} A P_{12}$.

(b) $P^T A P$, com $P = P_{13} P_{23}$.

(c) As transformadas principais de A, B e C com pivots de um elemento.

(d) $(A|a_{11})$, $\left(B| \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}\right)$, $\left(C| \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}\right)$, $(B|b_{33})$, $(C|c_{33})$.

(e) $det(A)$ e $det(B)$ usando a fórmula de Schur.

(f) A^{-1} e C^{-1}.

3. Verifique a Fórmula do Quociente para

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & 2 \\ 1 & 0 & 1 & 1 \\ -1 & 2 & 2 & -1 \end{bmatrix}, B = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \\ 1 & 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

.

4.

(a) Seja $A \in \mathbb{R}^{m \times n}$ uma matriz de característica $m < n$ e D uma matriz diagonal de elementos diagonais positivos. Mostre que $ADA^T \in$ SPD.

(b) Mostre que

$$\begin{bmatrix} D & A^T \\ A & 0 \end{bmatrix}$$

é não singular.

5. Seja

$$A = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 2 & 1 \\ 3 & 1 & 1 \end{bmatrix}$$

Determine $\lambda > 0$ de modo que $A + \lambda I$ seja diagonalmente dominante por linhas, colunas, e linhas e colunas.

6. Mostre que se $A \in \text{CDD}_+$, então $A_{II} \in \text{CDD}_+$ para todo o conjunto $I \subseteq \{1, \ldots, n\}$.

7. Mostre que se P é uma matriz de permutação, então

$$A \in \text{SRDD} \Rightarrow P^T AP \in \text{SRDD}.$$

8. Mostre que se $A \in \text{PD}$ então $a_{ii} > 0$ para todo i.

9. Mostre que se $A \in \text{PD}$ então $\det(A) > 0$.

10. Verifique se as seguintes matrizes são P:

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 1 \\ 1 & -3 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & -1 & 2 \\ 1 & 3 & -1 \\ -1 & 2 & 4 \end{bmatrix}.$$

11. Mostre que se Q é uma matriz de permutação, então

$$A \in \text{P} \Rightarrow Q^T AQ \in \text{P}.$$

12. Verifique se as seguintes matrizes são PD ou PSD:

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 3 & 1 & 2 \\ 1 & -1 & 2 \end{bmatrix}, B = \begin{bmatrix} 1 & 1 & -1 \\ 2 & 4 & 1 \\ 1 & -1 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & -2 & -1 \\ -2 & 2 & 2 \\ -1 & 2 & 3 \end{bmatrix}.$$

13.

(a) Mostre que se $A_1 \in \text{PD}$ e $A_2 \in \text{PD}$, então

$$\begin{bmatrix} A_1 & -B^T \\ B & A_2 \end{bmatrix} \in \text{PD}$$

com B uma matriz qualquer de ordem apropriada.

(b) Mostre que se D é uma matriz diagonal de elementos positivos e $C \in \text{SDD}_+$, então

$$\begin{bmatrix} D & -B^T \\ B & C \end{bmatrix} \in \text{PD}.$$

(c) Mostre que

$$\begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 1 & -2 \\ 0 & 2 & 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & 3 & -2 & 1 & 0 & 1 \\ 1 & -1 & 2 & 2 & 1 & -1 & 0 \\ 0 & 1 & -1 & 1 & 3 & 1 & -1 \\ -1 & 0 & 0 & 1 & -1 & 1 & 2 \\ 2 & 1 & -1 & 0 & 1 & -2 & 1 \end{bmatrix} \in \text{PD}.$$

14. Mostre que $A \in \text{PSD} \Leftrightarrow A + \epsilon I \in \text{PD}$ para todo o $\epsilon > 0$.

15.

(a) Mostre que toda a matriz triangular com elementos diagonais positivos é P.

(b) Mostre que se $A_1, A_2 \in \text{P}$, então

$$\begin{bmatrix} A_1 & B \\ 0 & A_2 \end{bmatrix} \in \text{P}$$

com B uma matriz de ordem apropriada.

(c) Mostre que

$$A = \begin{bmatrix} 1 & 2 & 3 & -1 & 1 & -10 & 0 & 0 \\ 0 & 1 & -1 & 0 & 2 & -9 & 0 & 0 \\ 0 & 0 & 2 & -1 & 4 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & -1 & 0 & 0 \\ 0 & 0 & 0 & -2 & 1 & 5 & 0 & 0 \\ 1 & 2 & -1 & 0 & 3 & 6 & 1 & 2 \\ -1 & 0 & 3 & -4 & 5 & 7 & -1 & 2 \end{bmatrix} \in \mathrm{P}.$$

16. Mostre que se $A \in \mathrm{S}$ então existe um vector $x > 0$ tal que $Ax > 0$.

17.

(a) Mostre que toda a matriz não negativa com elementos diagonais positivos pertence à classe S.

(b) Mostre que toda a matriz com uma coluna de elementos positivos é S.

18. Verifique se as seguintes matrizes são S:

$$A = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 1 & -1 \\ 0 & 3 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 2 & -1 \\ -1 & 1 & 0 \end{bmatrix}, C = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 4 & -1 \\ -1 & -2 & 4 \end{bmatrix}.$$

19. Considere a classe S_0 das matrizes que satisfazem a seguinte equivalência

$$A \in \mathrm{S}_0 \Leftrightarrow \text{Existe } 0 \neq x \geq 0 \text{ tal que } Ax \geq 0$$

. Mostre que

$$(A + \epsilon I) \in \mathrm{S} \text{ para todo } \epsilon > 0 \Rightarrow A \in \mathrm{S}_0$$

mas que a implicação recíproca não é verdadeira.

20. Verifique se as seguintes matrizes são K:

$$A = \begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & -1 & -2 \\ -1 & 2 & -1 \\ -1 & -3 & -2 \end{bmatrix}, C = \begin{bmatrix} -1 & -2 & 0 \\ -3 & 1 & -2 \\ 0 & -1 & 3 \end{bmatrix}, D = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ -1 & -2 & -4 \end{bmatrix}.$$

21. Mostre que se $A \in \mathrm{K}$ então $A^{-1} \geq 0$.

22. Verifique se são H ou H_+ as seguintes matrizes

$$A_1 = \begin{bmatrix} 1 & -2 & 1 \\ 1 & -1 & 0 \\ -1 & 2 & 3 \end{bmatrix}, A_2 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 2 & 3 \\ -2 & 2 & 3 \end{bmatrix}, A_3 = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 2 & 1 \\ 1 & -1 & 3 \end{bmatrix}$$

23. Mostre que se A é uma matriz H de ordem n, então

(i) $\det(A) \neq 0$.

(ii) $(A|A_{II}) \in \mathrm{H}$ para todo o conjunto $I \subseteq \{1, \ldots, n\}$.

24. Determine os valores de x tais que as matrizes

$$A_1 = \begin{bmatrix} x & -1 & 2 \\ -1 & 1 & 1 \\ 1 & -1 & x \end{bmatrix}, A_2 = \begin{bmatrix} 1 & 2 & 1 \\ 1 & x & 1 \\ -1 & 0 & x \end{bmatrix}$$

são

(a) H_+.

(b) P.

25. Considere a classe de matrizes copositivas estritas SC definidas por

$$A \in \text{SC} \Leftrightarrow x^T Ax > 0 \text{ para todo } 0 \neq x \geq 0$$

(a) Mostre que

$$\text{SDD}_+ \subset \text{SC} \subset \text{S}$$

(b) Estude a invariância das matrizes da classe SC em relação a submatrizes principais, permutações principais, transformadas principais e Complementos de Schur.

Capítulo 4

Métodos Directos para Sistemas de Equações Lineares com Matrizes Quadradas Densas

Neste capítulo debruçar-nos-emos sobre a resolução de sistemas de equações lineares de dimensões pequenas usando métodos directos. Esses processos procuram determinar a solução exacta do sistema e só não a obtém devido a erros de arredondamento. Devido às dimensões dos sistemas iremos supor que as matrizes são densas, isto é, que os elementos da matriz são todos considerados diferentes de zero. Iremos começar por discutir a resolução de sistemas com matrizes triangulares. A decomposição LU é então introduzida e usada na resolução de sistemas de equações com matrizes quaisquer. Seguidamente são discutidos alguns assuntos relacionados com o uso de métodos directos para a resolução de sistemas, nomeadamente a escolha parcial de pivot, o escalonamento, o refinamento iterativo e a estimação do número de condição de uma matriz. Os métodos directos e dos bordos são processos alternativos para a determinação da decomposição LU de uma matriz e são a seguir descritos. A decomposição LDU de uma matriz é também introduzida e torna-se mais importante na resolução de sistemas com matrizes simétricas. Esse assunto será discutido na secção seguinte. Finalmente a determinação da inversa e o cálculo do determinante de uma matriz serão tratados na última secção deste capítulo.

4.1 Resolução de sistemas triangulares

Seja $L \in \mathbb{R}^{n\times n}$ uma matriz triangular inferior não singular, $x, b \in \mathbb{R}^n$ e consideremos o sistema triangular inferior

$$Lx = b \tag{4.1}$$

ou seja

$$\begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Como L é não singular, então $l_{ii} \neq 0$ para todo o $i = 1, \dots, n$, e portanto a solução do sistema (4.1) pode ser obtida resolvendo sucessivamente em ordem a x_1, x_2, …, x_n. Da primeira equação tem-se $\bar{x}_1 = \frac{b_1}{l_{11}}$. Substituindo este valor de x_1 na segunda equação e resolvendo em ordem a x_2, obtém–se

$$\bar{x}_2 = \frac{b_2 - l_{21}\bar{x}_1}{l_{22}}$$

Continuando o processo tem–se na última equação

$$\bar{x}_n = \frac{b_n - (l_{n1}\bar{x}_1 + \ldots + l_{n,n-1}\bar{x}_{n-1})}{l_{nn}}$$

Na prática o vector $\bar{x} = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n]^T$ pode ocupar o espaço reservado por b, pelo que podemos escrever o algoritmo para a resolução do sistema $Lx = b$ na seguinte forma

Algoritmo 1 (TRIANINF)

Para $k = 1, \ldots, n$ *faça*

$$b_k = \frac{b_k - \sum_{j=1}^{k-1} l_{kj} b_j}{l_{kk}}$$

Se considerarmos agora o sistema

$$Ux = b \tag{4.2}$$

com U uma matriz triangular superior não singular, então é fácil de obter o seguinte algoritmo para a sua resolução:

Algoritmo 2 (TRIANSUP)

Para $k = n, n-1, \ldots, 1$ *faça*

$$b_k = \frac{b_k - \sum_{j=k+1}^{n} u_{kj} b_j}{u_{kk}}$$

Para calcular o número de operações notemos que no passo k se têm $k-1$ multiplicações, $k-1$ adições e 1 divisão. Portanto o número total de multiplicações, adições e divisões é dado por

$$\begin{cases} \text{Número de multiplicações} & = & \frac{n(n-1)}{2} \\ \text{Número de adições} & = & \frac{n(n-1)}{2} \\ \text{Número de divisões} & = & n \end{cases} \tag{4.3}$$

Na prática as multiplicações e as divisões são as operações com um peso mais forte na determinação do esforço computacional de um algoritmo. Por isso podemos dizer que o número de operações é $\frac{n^2+n}{2}$, ou ainda, que o número de operações é $\frac{n^2}{2} + \mathcal{O}(n)$, onde $\mathcal{O}(n)$ representa termos de ordem n.

Os algoritmos TRIANINF e TRIANSUP são estáveis. Com efeito a solução $\bar{x}$ de (4.1) obtida pelo algoritmo TRIANINF é a solução exacta do sistema $(L + E)x = b$, onde a matriz erro E satisfaz

$$||E||_\infty \leq 1.01 n \epsilon_M ||L||_\infty \tag{4.4}$$

com ϵ_M a precisão da máquina. A demonstração deste resultado aparece em [Forsythe e Moler, cap. 21] e não será apresentada neste trabalho por ser demasiado técnica. Uma desigualdade semelhante é válida para o algoritmo TRIANSUP.

4.2 Decomposição LU

Consideremos o sistema

$$Ax = b \tag{4.5}$$

em que A é uma matriz quadrada de ordem n não singular e $b \in \mathbb{R}^n$. O processo mais comum para o resolver assenta na obtenção da decomposição $A = LU$, em que L é uma matriz triangular inferior e U uma matriz triangular superior. Após a determinação dessa decomposição a solução do sistema (4.5) obtém-se resolvendo os dois sistemas triangulares

$$Ly = b \quad e \quad Ux = y \tag{4.6}$$

usando os algoritmos TRIANINF e TRIANSUP.

A obtenção da decomposição LU da matriz A é normalmente feita tentando transformar A numa matriz triangular superior e aproveitando os multiplicadores necessários para essa obtenção para construir a matriz L. Assim, se $a_{11} \neq 0$, podemos escrever

$$A = \begin{bmatrix} 1 & & & \\ m_{21}^{(1)} & 1 & & \\ \vdots & & \ddots & \\ m_{n1}^{(1)} & & & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix} = L^{(1)}A^{(2)} \tag{4.7}$$

com

$$\begin{cases} m_{i1}^{(1)} & = & \frac{a_{i1}}{a_{11}}, \quad i = 2, \dots, n \\ a_{ij}^{(2)} & = & a_{ij} - m_{i1}^{(1)} a_{1j}, \quad i > 1, j > 1 \end{cases} \tag{4.8}$$

Considerando agora a matriz $A^{(2)}$ em (4.7), e supondo que $a_{22}^{(2)} \neq 0$, podemos escrever

$$A^{(2)} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & m_{32}^{(2)} & 1 & & \\ & \vdots & & \ddots & \\ & m_{n2}^{(2)} & & & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \dots & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \dots & a_{nn}^{(3)} \end{bmatrix} \tag{4.9}$$

onde os elementos $m_{i2}^{(2)}$ e $a_{ij}^{(3)}$ se obtêm de acordo com fórmulas semelhantes às apresentadas em (4.8). Se repetirmos o processo mais $n-3$ vezes, obtemos uma decomposição da matriz A da forma

$$A = L^{(1)}L^{(2)} \dots L^{(n-1)}U \tag{4.10}$$

com U uma matriz triangular superior. Como

$$L^{(1)}L^{(2)} \dots L^{(n-1)} = \begin{bmatrix} 1 & & & & \\ m_{21}^{(1)} & 1 & & & \\ & m_{32}^{(2)} & 1 & & \\ \vdots & \vdots & \vdots & \ddots & \\ m_{n1}^{(1)} & m_{n2}^{(2)} & m_{n3}^{(3)} & \dots & 1 \end{bmatrix} \tag{4.11}$$

é uma matriz triangular inferior, então obtemos a decomposição pretendida escrevendo $L = L^{(1)}L^{(2)} \dots L^{(n-1)}$.

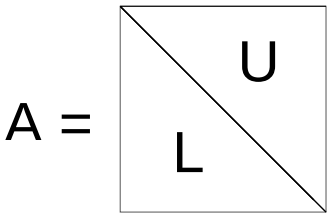


Figura 4.1: Representação esquemática da matriz A após a obtenção da decomposição LU

Como afirmámos anteriormente, a decomposição LU é determinada para a resolução do sistema (4.5). Com efeito, uma vez obtida a decomposição LU de A, a resolução de (4.5) resume–se à resolução dos sistemas triangulares (4.6). Como os elementos diagonais da matriz L são todos iguais a um e só são necessários para efectuar divisões (ver algoritmo TRIANINF), então não necessitam de ser armazenados. Por isso o espaço de armazenagem para as matrizes L e U é de n^2. Por outro lado, a matriz A não é mais necessária após a obtenção da sua decomposição LU. Deste modo é possível implementar o processo de decomposição LU de uma matriz A usando apenas o espaço de armazenagem da matriz A e modificando os seus elementos de acordo com fórmulas do tipo (4.8). O algoritmo para a obtenção da decomposição LU terá então a seguinte forma

Algoritmo 3 (DECOMP)

$$
\begin{array}{l}
\textit{Para } k = 1, 2, \ldots, n-1 \ \ \textit{faça} \\
\quad \textit{Para } i = k+1, \ldots, n \ \ \textit{faça} \\
\quad\quad a_{ik} = \dfrac{a_{ik}}{a_{kk}} \\
\quad\quad \textit{Para } j = k+1, \ldots, n \ \ \textit{faça} \\
\quad\quad\quad a_{ij} = a_{ij} - a_{ik} a_{kj}
\end{array}
$$

No final do processo a matriz A terá a forma apresentada na figura 4.1, isto é, os elementos da diagonal e acima da diagonal pertencem à matriz U e os elementos não diagonais da matriz L são os elementos abaixo da diagonal de A.

Para calcular o número de operações necessárias para a obtenção da decomposição LU de uma matriz A, notemos que em cada passo k são necessárias $n-k$ divisões, $(n-k)^2$ multiplicações e $(n-k)^2$ adições. Portanto o número total de multiplicações e divisões será

$$\sum_{k=1}^{n-1} (n-k)(n-k+1) = \frac{n(n^2-1)}{3} \tag{4.12}$$

O número total de adições é

$$\sum_{k=1}^{n-1} (n-k)^2 = \frac{n(n-1)(2n-1)}{6} \tag{4.13}$$

Podemos por isso dizer que o número de operações necessário para a obtenção da decomposição LU de A é $\frac{n^3}{3} + \mathcal{O}(n^2)$.

Como a resolução do sistema (4.5) se resume à obtenção da decomposição LU e à resolução dos sistemas triangulares (4.6), então a partir das expressões (4.3), (4.12) e (4.13) obtemos o número de operações para a resolução do sistema $Ax = b$. Assim, tem–se

$$
\begin{cases}
\text{Multiplicações e divisões para a resolução de } Ax = b: & \dfrac{n^3}{3} + n^2 - \dfrac{n}{3} \\
\text{Adições para a resolução de } Ax = b: & \dfrac{n^3}{3} + \dfrac{n^2}{2} - \dfrac{5n}{6}
\end{cases}
$$

e portanto a complexidade do algoritmo é $\frac{n^3}{3}+\mathcal{O}(n^2)$. De notar ainda que o espaço de armazenagem necessário para resolver o sistema é $n(n+1)$, ou seja, é igual ao espaço de armazenagem necessário para a matriz A e o vector b iniciais.

Como exemplo de ilustração da resolução de um sistema de equações lineares usando a decomposição LU de uma matriz, consideremos o sistema $Ax = b$ com

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 0 & 1 \\ 0 & -1 & 4 \end{bmatrix}, b = \begin{bmatrix} 4 \\ 2 \\ 3 \end{bmatrix}$$

Tal como foi referido nesta secção, há que primeiramente determinar a decomposição LU de A usando o algoritmo DECOMP. Para isso efectua-se a primeira iteração, em que a_{11} é o pivot. Segundo esse algoritmo, os elementos da linha do pivot (linha 1) não são alterados, enquanto que os elementos da coluna do pivot (coluna 1) colocados abaixo do pivot são divididos por esse elemento. Todos os elementos restantes pertencem ao Complemento de Schur e são por isso calculados a partir de

$$\bar{a}_{ij} = a_{ij} - \bar{a}_{ik}a_{kj}$$

com $\bar{a}_{ik}$ o elemento da coluna do pivot já transformado. Assim no nosso caso tem-se

$$A \to A^{(2)} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & -1 & -1 \\ 0 & -1 & 4 \end{bmatrix}$$

Repetindo agora o mesmo processo para o pivot $a_{22}^{(2)}$ vem

$$A^{(3)} = \begin{bmatrix} 1 & 1 & 2 \\ 1 & -1 & -1 \\ 0 & 1 & 5 \end{bmatrix}$$

Como a ordem da matriz A é igual a três, então a matriz $A^{(3)}$ fornece a decomposição LU de A e tem-se

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 1 & 2 \\ 0 & -1 & -1 \\ 0 & 0 & 5 \end{bmatrix}$$

Para resolver o sistema $Ax = b$, há que primeiramente calcular o vector y que é solução do sistema triangular inferior $Ly = b$. Então tem-se

$$Ly = b \Rightarrow \begin{cases} y_1 = 4 \\ y_2 = 2 - y_1 = -2 \\ y_3 = 3 - y_2 = 5 \end{cases}$$

A solução do sistema é obtida resolvendo o sistema $Ux = y$, ou seja, tem-se

$$Ux = y \Rightarrow \begin{cases} 5x_3 = 5 \Rightarrow x_3 = 1 \\ -x_2 = -2 + x_3 \Rightarrow x_2 = 1 \\ x_1 = 4 - x_2 - 2x_3 = 1 \end{cases}$$

Assim $x = (1, 1, 1)$ é a solução do sistema dado.

O processo de obtenção da decomposição LU que acabámos de apresentar é usualmente conhecido como Eliminação de Gauss. Este processo é o mais recomendado para matrizes densas e consiste em modificar os elementos da matriz A segundo as regras explicitadas no algoritmo DECOMP. A iteração k desse algoritmo é representada esquematicamente na figura 4.2.

Existem dois outros processos de obter a decomposição LU de uma matriz A, conhecidos por método dos bordos e método directo. Estes processos requerem o mesmo número de operações para obter a decomposição e, apesar de não serem muito usados com matrizes densas, assumem particular importância na obtenção da decomposição de matrizes esparsas. Esses algoritmos serão discutidos mais adiante.

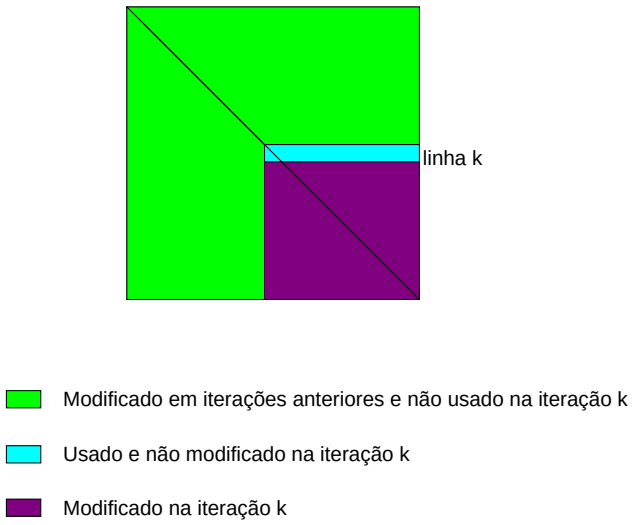


Figura 4.2: Ilustração esquemática da progressão da Eliminação de Gauss

4.3 Complemento de Schur e decomposição LU

O processo de decomposição LU pode ser apresentado de um modo bastante elegante usando o conceito de Complemento de Schur. Com efeito, de acordo com o processo de decomposição LU podemos escrever a matriz $A^{(2)}$ na forma

$$A^{(2)} = \begin{array}{|c|c|} \hline a_{11} & b^1 \\ \hline \dfrac{1}{a_{11}} c^1 & (A|a_{11}) \\ \hline \end{array}$$

com $b^1 = (a_{12}, \ldots, a_{1n})$ e $c^1 = (a_{21}, \ldots, a_{n1})^T$. Se considerarmos as matrizes

$$A_{kk} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1k} \\ a_{21} & a_{22} & \ldots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \ldots & a_{kk} \end{bmatrix}, \quad k = 1, 2, \ldots, n \tag{4.14}$$

então escrevendo $a_{11} = (A_{11}|A_{00})$ e tendo em conta a Fórmula do Quociente tem-se no fim do segundo passo:

$$A^{(3)} = \begin{array}{|c|c|c|} \hline (A_{11}|A_{00}) & b^1 & \\ \hline \dfrac{1}{(A_{11}|A_{00})} c^1 & (A_{22}|A_{11}) & b^2 \\ \hline & \dfrac{1}{(A_{22}|A_{11})} c^2 & (A|A_{22}) \\ \hline \end{array}$$

onde b^2 e c^2 são constituídos pelos elementos não diagonais respectivamente da primeira linha e primeira coluna de $(A|A_{11})$. Portanto, por indução tem–se

$$A^{(k)} = \begin{array}{|c|c|c|c|} \hline (A_{11}|A_{00}) & b^1 & & \\ \hline \dfrac{1}{(A_{11}|A_{00})} c^1 & (A_{22}|A_{11}) & b^2 & \\ \hline & \dfrac{1}{(A_{22}|A_{11})} c^2 & \ddots & \\ \hline & & \ldots & (A_{kk}|A_{k-1,k-1}) \quad b^k \\ \hline & & \ldots & \dfrac{1}{(A_{kk}|A_{k-1,k-1})} c^k \quad (A|A_{kk}) \\ \hline \end{array}$$

onde b^k e c^k são constituídos pelos elementos não diagonais respectivamente da primeira linha e primeira coluna da matriz $(A|A_{k-1,k-1})$.

Da descrição apresentada podemos concluir que a decomposição LU existe se e só se existir $(A_{kk}|A_{k-1,k-1})$, para todo o $k = 1, \ldots, n-1$. Mas pela Fórmula de Schur tem-se

$$(A_{kk}|A_{k-1,k-1}) = \det(A_{kk}|A_{k-1,k-1}) = \frac{\det(A_{kk})}{\det(A_{k-1,k-1})}$$

Podemos portanto enunciar o seguinte resultado:

Teorema 4.1 *$A = LU$ se e só se $\det(A_{kk}) \neq 0$, para todo o $k = 1, \ldots, n$, com A_{kk} as matrizes dadas por (4.14).*

Consideremos agora uma classe de matrizes $\mathbb{T}$. Da relação entre a decomposição LU e os Complementos de Schur podemos enunciar o seguinte teorema:

Teorema 4.2 *Seja $\mathbb{T}$ uma classe de matrizes tal que*

$$A \in \mathbf{T} \Leftrightarrow a_{11} \neq 0 \quad e \quad (A|a_{11}) \in \mathbf{T}$$

Então

$$A \in \mathbb{T} \Rightarrow A = LU$$

Demonstração: Se o elemento diagonal a_{11} de A é não nulo, então a primeira iteração do processo de obtenção da decomposição LU pode ser efectuada. Como $(A|a_{11}) \in \mathbb{T}$ então o elemento da primeira linha e primeira coluna é diferente de zero. Portanto pode ser levada a cabo a segunda iteração da decomposição LU. O resultado é então verdadeiro por indução.

No capítulo 3 estudámos várias classes de matrizes nas condições do teorema 4.2. Em particular, podemos enunciar o seguinte teorema:

Teorema 4.3 *Se A é uma matriz P, H ou diagonalmente dominante (por linhas ou por colunas) não singular, então $A = LU$.*

O resultado é igualmente válido para todas as subclasses dessas classes apresentadas na figura 3.1, nomeadamente para as matrizes K, PD e estritamente diagonalmente dominantes por linhas ou por colunas.

4.4 Escolha parcial de pivot

Como afirmámos anteriormente, para a determinação da decomposição LU de uma matriz A é necessário e suficiente que os elementos $a_{kk}^{(k)}$ sejam diferentes de zero. Contudo, a verificação de tal propriedade não conduz necessariamente a um algoritmo estável, e portanto a resolução de um sistema $Ax = b$ usando a decomposição LU poderá dar resultados erróneos. Como exemplo de ilustração do que acabamos de afirmar, consideremos o seguinte sistema

$$\left\{ \begin{array}{rcl} 0.003x_1 + 59.14x_2 & = & 59.17 \\ 5.291x_1 - 6.130x_2 & = & 46.78 \end{array} \right. \tag{4.15}$$

cuja solução exacta é $x_1 = 10.0$ e $x_2 = 1.0$. Se efectuarmos a decomposição LU da matriz do sistema e resolvermos o sistema usando essa decomposição com aritmética de arredondamento de base $\beta = 10$ e precisão $t = 4$, então obtemos a solução dada por $x_1 = 1.001$ e $x_2 = -10.00$, que obviamente é muito pouco correcta. No entanto se trocarmos a primeira e segunda linhas e resolvermos o sistema usando a decomposição LU da matriz permutada e a mesma precisão, obtemos a solução exacta. A razão da obtenção de uma solução sem significado prende-se com o facto de o pivot 0.003 ser um número muito próximo da precisão da máquina 10^{-4} que considerámos neste exemplo. A troca das linhas ultrapassa este problema e permite a obtenção de uma solução bastante precisa.

O exemplo apresentado mostra a utilidade da técnica da Escolha Parcial de Pivot, que consiste em escolher para pivot da iteração k (isto é, o elemento $a_{kk}^{(k)}$) o elemento de maior valor absoluto das linhas $i \geq k$ e na coluna k. Se esse elemento está na linha $r > k$, então trocam-se as linhas r e k e procede-se como anteriormente. Se usarmos esta técnica então tem-se

$$P_{n-1} \dots P_1 A = LU$$

com P_i matrizes de permutação correspondentes às trocas de linhas efectuadas. De notar que, se não houver necessidade de troca de linhas na iteração k, então $P_k = I$, onde I é a matriz identidade de ordem n.

A análise do algoritmo para a obtenção da decomposição LU com escolha parcial de pivot mostra que se $LU = P(A + E)$, onde E é a matriz erro, então

$$||E||_\infty \leq n^2 \epsilon_M g ||A||_\infty \tag{4.16}$$

onde ϵ_M é a precisão da máquina e g é o Factor de Crescimento definido por

$$g = \frac{\max\limits_{i,j \geq k} |a_{ij}^{(k)}|}{\max\limits_{i,j} |a_{ij}|} \tag{4.17}$$

com $a_{ij}^{(k)}$ os elementos das matrizes $A^{(k)}$ obtidas no processo de obtenção da decomposição LU com escolha parcial de pivot. A demonstração deste resultado é muito técnica e aparece em [Forsythe e Moler, cap. 21]. Portanto esse processo é estável se g for pequeno. Mas das fórmulas (4.8) tem-se

$$|m_{i1}^{(1)}| = \frac{|a_{i1}^{(1)}|}{|a_{11}^{(1)}|} \leq 1$$

e

$$|a_{ij}^{(2)}| \leq |a_{ij}^{(1)}| + |m_{ij}^{(1)}||a_{1j}^{(1)}| \leq 2 \max_{i,j} |a_{ij}^{(1)}| = 2 \max_{i,j} |a_{ij}|$$

Usando o método de indução, facilmente se conclui que

$$|a_{ij}^{(k)}| \leq 2^{k-1} \max_{i,j} |a_{ij}|$$

para todo o $k = 1, \dots, n-1$ e portanto $g \leq 2^{n-1}$. Além disso, é possível construir uma matriz A para a qual $g = 2^{n-1}$. Portanto o processo de escolha parcial de pivot pode conduzir a resultados numéricos pouco precisos. Contudo na prática o valor de g é bastante pequeno, e a escolha parcial de pivot é por isso uma técnica bastante recomendável e universalmente aceite para a resolução de sistemas de equações lineares com matrizes densas.

Consideremos o sistema (4.5) e suponhamos que usámos escolha parcial de pivot para obter a decomposição LU de A. Então a solução $\bar{x}$ obtida na resolução do sistema (4.5) é a solução exacta do problema

$$(A + \delta A)x = b \tag{4.18}$$

Para estudar a estabilidade do processo de resolução do sistema iremos obter um limite superior da norma ℓ_∞ da matriz erro δA. Notemos que a imprecisão da solução obtida é devida aos erros

de arredondamento na determinação da decomposição LU de A e na resolução dos dois sistemas triangulares (4.6). Podemos então escrever $A + E = LU$ com E a matriz erro da decomposição LU e além disso

$$(L + \delta L)(U + \delta U)\bar{x} = b \tag{4.19}$$

onde δL e δU são as matrizes erro relativas à resolução dos dois sistemas triangulares. Mas de (4.18) e (4.19) vem

$$\delta A = E + U\delta L + L\delta U + \delta L\delta U$$

Além disso, das fórmulas (4.4), (4.16) e (4.17), e tendo em conta que todos os elementos da matriz L são em valor absoluto menores ou iguais a um, tem-se

$$||L||_\infty \leq n$$

$$||U||_\infty \leq n \max_{i,j} |u_{ij}| = ng \max_{i,j} |a_{ij}| \leq ng||A||_\infty$$

$$||\delta L||_\infty \leq 1.01n\epsilon_M||L||_\infty \leq 1.01n^2\epsilon_M$$

$$||\delta U||_\infty \leq 1.01n\epsilon_M||U||_\infty \leq 1.01n^2\epsilon_M g||A||_\infty$$

$$||E||_\infty \leq n^2\epsilon_M g||A||_\infty$$

Como $n^2\epsilon_M \leq 1$, então

$$||\delta L||_\infty||\delta U||_\infty \leq 1.01^2 n^2\epsilon_M g||A||_\infty \leq 2n^2\epsilon_M g||A||_\infty$$

Portanto

$$\begin{aligned} ||\delta A||_\infty &\leq ||E||_\infty + ||U||_\infty||\delta L||_\infty + ||L||_\infty||\delta U||_\infty + ||\delta L||_\infty||\delta U||_\infty \\ &\leq 1.01(3n^2 + 2n^3)g||A||_\infty\epsilon_M \end{aligned}$$

e a resolução de sistemas de equações lineares com escolha parcial de pivot é um processo estável se g for pequeno. É de notar que o cálculo do valor de g envolve a determinação de $\max_{i,j\geq k} |a_{ij}^{(k)}|$ para o que é necessário efectuar

$$\sum_{k=1}^{n-1}(n-k)^2 = \frac{n^3}{3} + \mathcal{O}(n^2)$$

comparações. Portanto a determinação de g exige um grande esforço computacional. Contudo, como afirmámos anteriormente, esse valor de g não necessita de ser calculado, pois é normalmente bastante pequeno.

Para a implementação da escolha parcial de pivot consideramos um vector *perm* de n números inteiros, onde para $k = 1, \ldots, n-1$, $perm(k) = j$ significa que no passo k a linha j é a linha do pivot. Além disso $perm(n)$ contém o número de trocas efectuadas, o que, como veremos mais adiante, terá importância no cálculo do determinante de uma matriz. Esse vector é inicializado com $perm(i) = 0$, para todo o $i = 1, \ldots, n$. Se na iteração k o pivot estiver na linha $j > k$, então os elementos das linhas j e k são trocados dois a dois e o mesmo acontece às componentes j e k do vector *perm*. Além disso $perm(n)$ é incrementado em uma unidade nesse caso. Se introduzirmos essas alterações no algoritmo 3, então obtemos a decomposição LU da matriz PA, onde a matriz de permutação P é dada pelo vector *perm*. Como $PA = LU$, então a resolução do sistema (4.5) reduz-se à resolução dos sistemas triangulares $Ly = Pb$ e $Ux = y$, pelo que o algoritmo 2 tem de ser modificado, ficando com a forma seguinte

Algoritmo 4

$b_{perm(1)} \leftrightarrow b_1$

Para $k = 2, \ldots, n$

$b_k \leftrightarrow b_{perm(k)}$

$b_k = b_k - \sum_{j=1}^{k-1} a_{kj} b_j$

Neste algoritmo, $a \leftrightarrow b$ significa troca de valores entre as variáveis a e b.

Como exemplo de ilustração deste algoritmo, consideremos o sistema de equações lineares $Ax = b$ com

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & -1 \\ 1 & \frac{3}{2} & 1 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 5 \\ 3 \end{bmatrix}$$

Tal como foi referido anteriormente, consideremos inicialmente o vector

$$perm = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}$$

Na primeira iteração do processo de obtenção da decomposição LU com escolha parcial de pivot, o pivot é o elemento

$$|a_{21}| = \max\{|a_{i1}| : i = 1, 2, 3\}$$

Então trocam-se as linhas 1 e 2, ou seja, obtém-se a matriz

$$P_{12}A = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 1 & 1 \\ 1 & \frac{3}{2} & 1 \end{bmatrix}$$

Além disso o vector *perm* passa a ser igual a $\begin{bmatrix} 2 & 0 & 1 \end{bmatrix}$. Usando o primeiro elemento diagonal da matriz $P_{12}A$ como pivot e efectuando uma iteração do algoritmo DECOMP obtém-se a seguinte matriz

$$(P_{12}A)^{(2)} = \begin{bmatrix} 2 & 1 & -1 \\ \frac{1}{2} & \frac{1}{2} & \frac{3}{2} \\ \frac{1}{2} & 1 & \frac{3}{2} \end{bmatrix}$$

Na segunda iteração a escolha parcial de pivot indica $a_{32}^{(2)}$ como pivot e portanto há que trocar as linhas 2 e 3 da matriz. Então $perm = \begin{bmatrix} 2 & 3 & 2 \end{bmatrix}$ e

$$(P_{32}P_{12}A)^{(2)} = \begin{bmatrix} 2 & 1 & -1 \\ \frac{1}{2} & 1 & \frac{3}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{3}{2} \end{bmatrix}$$

Efectuando agora uma iteração com pivot $a_{22}^{(2)}$ obtém-se

$$(P_{32}P_{12}A)^{(3)} = \begin{bmatrix} 2 & 1 & -1 \\ \frac{1}{2} & 1 & \frac{3}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{3}{4} \end{bmatrix}$$

Portanto

$$P_{32}P_{12}A = LU$$

com

$$L = \begin{bmatrix} 1 & & \\ \frac{1}{2} & 1 & \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 2 & 1 & -1 \\ & 1 & \frac{3}{2} \\ & & \frac{3}{4} \end{bmatrix}$$

Além disso as matrizes de permutação estão "armazenadas" no vector $perm = \begin{bmatrix} 2 & 3 & 2 \end{bmatrix}$. Para resolver o sistema há que resolver os sistemas triangulares

$$Ly = P_{32}P_{12}b, \quad Ux = y$$

Para isso há que calcular primeiramente $P_{32}P_{12}b$ o que se faz do seguinte modo

$$P_{32}P_{12}b = P_{32}\left(P_{12}b\right) = P_{32}\begin{bmatrix} 5 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \\ 2 \end{bmatrix}$$

Depois resolvem-se os sistemas

$$Ly = \begin{bmatrix} 5 \\ 3 \\ 2 \end{bmatrix}, Ux = y$$

e obtém-se a solução $x = [1\ 2\ 1]^T$.

Na prática as matrizes de permutação não são armazenadas, mas em vez disso é utilizado o vector $perm$, sendo o sistema $Ly = Pb$ resolvido de acordo com o algoritmo 4. Note-se que

$$perm(1) = 2 \Rightarrow b_1 \leftrightarrow b_2 \Rightarrow b = [5\ 2\ 3]$$

$$perm(2) = 3 \Rightarrow b_2 \leftrightarrow b_3 \Rightarrow b = [5\ 3\ 2]$$

Isto mostra que o vector b obtido após as duas trocas de linhas é exactamente o mesmo que foi determinado usando as matrizes de permutação.

É importante notar que uma matriz é não singular se e só se é possível obter a sua decomposição LU com escolha parcial de pivot. Com efeito, suponhamos que não é possível encontrar um pivot diferente de zero numa determinada iteração k. Então a matriz $A^{(k)}$ tem a forma

$$A^{(k)} = \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A_{22}^{(k)} \end{bmatrix}$$

com

$$A_{22}^{(k)} = \begin{bmatrix} 0 & a_{k,k+1}^{(k)} & \dots & a_{kn}^{(k)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n,k+1}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix} = (A|A_{11})$$

Portanto pela Fórmula de Schur

$$\det(A) = \det(A_{11})\det(A|A_{11}) = 0$$

o que mostra que a matriz A é singular.

Devido ao uso de aritmética de precisão finita, pode acontecer que uma matriz seja singular e não apareça uma coluna nula no processo de decomposição. Na prática, é considerada uma tolerância da ordem de $\sqrt{\epsilon_M}$ e o processo termina com uma mensagem de singularidade da matriz sempre que o pivot a escolher tenha valor absoluto inferior a essa tolerância.

Na prática é muitas vezes da máxima conveniência a implementação de algoritmos orientados por colunas, em virtude de ser esse o modelo de armazenagem de algumas linguagens de programação, como o FORTRAN 77 ou, mais recentemente, o FORTRAN 90. Nesse sentido, apresentamos na figura 4.3 formas orientadas por colunas para a decomposição LU e para a resolução dos sistemas triangulares.

Algoritmo 5 (Decomposição LU)

Para $k = 1, 2, \ldots, n-1$

  Para $i = k+1, \ldots, n$

    $a_{ik} = \frac{a_{ik}}{a_{kk}}$

  Para $j = k+1, \ldots, n$

    Para $i = k+1, \ldots, n$

      $a_{ij} = a_{ij} - a_{ik}a_{kj}$

Algoritmo 6 (Resolução de $Ly = b$)

Para $k = 1, \ldots, n-1$

$$\begin{bmatrix} b_{k+1} \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} b_{k+1} \\ \vdots \\ b_n \end{bmatrix} - \begin{bmatrix} a_{k+1,k} \\ \vdots \\ a_{nk} \end{bmatrix} b_k$$

Algoritmo 7 (Resolução de $Ux = b$)

Para $k = n, n-1, \ldots, 2$

$$b_k = \frac{b_k}{a_{kk}}$$

$$\begin{bmatrix} b_1 \\ \vdots \\ b_{k-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{k-1} \end{bmatrix} - \begin{bmatrix} a_{1k} \\ \vdots \\ a_{k-1,k} \end{bmatrix} b_k$$

$$b_1 = \frac{b_1}{a_{11}}$$

Figura 4.3: Forma orientada por colunas para a decomposição LU e resolução de sistemas triangulares

4.5 Escalonamento

Diz-se que duas matrizes A e B são Diagonalmente Equivalentes se existem duas matrizes diagonais D_1 e D_2 não singulares tais que

$$B = D_1^{-1} A D_2 \tag{4.20}$$

Consideremos agora o sistema (4.5) e seja B uma matriz diagonalmente equivalente a A. Então os sistemas $Ax = b$ e

$$D_1^{-1} A D_2 y = D_1^{-1} b \tag{4.21}$$

são equivalentes, pois $\bar{x}$ é solução de $Ax = b$ se e só se $\bar{y} = D_2^{-1}\bar{x}$ é solução do sistema (4.21). Apesar de equivalentes, o uso da decomposição LU (com ou sem escolha parcial de pivot) para a resolução dos dois sistemas pode conduzir a soluções completamente diferentes. Assim, por exemplo, se considerarmos o sistema (4.15) e multiplicarmos todos os elementos da primeira equação por 10^4 obtemos o seguinte sistema

$$\begin{cases} 30x_1 + 591400x_2 &= 591700 \\ 5.291x_1 - 6130x_2 &= 46.73 \end{cases} \tag{4.22}$$

Se utilizarmos a técnica de escolha parcial de pivot com aritmética de arredondamento de base $\beta = 10$ e precisão $t = 4$, então obtemos a solução $x_1 = 1.001$ e $x_2 = -10.0$, que como vimos é bastante errónea. Notemos que as matrizes A e B dos sistemas (4.5) e (4.22) respectivamente são diagonalmente equivalentes, pois satisfazem a equação (4.20) com

$$D_1^{-1} = \begin{bmatrix} 10^4 & \\ & 1 \end{bmatrix}, \quad D_2^{-1} = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$$

Esta técnica de multiplicar as linhas e colunas da matriz $[A|b]$ de um sistema de modo a obter um sistema cuja matriz seja diagonalmente equivalente a A é denominada Escalonamento.

O exemplo anterior mostra que a técnica de escalonamento pode conduzir a resultados bastante perigosos, mas também pode fazer com que a solução do sistema obtida pelo processo de decomposição LU seja bastante melhor, conforme facilmente se conclui considerando o sistema (4.22) como o original.

O resultado fundamental em que se baseia a técnica de escalonamento afirma que, se A e B são diagonalmente equivalentes tais que $B = D_1^{-1} A D_2$, então as soluções dos sistemas (4.5) e (4.21) são exactamente iguais se os elementos diagonais das matrizes D_1 e D_2 forem potências de β (onde β é a base do sistema de vírgula flutuante) e se a ordem de pivotagem para a obtenção das decomposições LU de A e de B for a mesma. Esse resultado é aliás confirmado no exemplo anterior, pois $\beta = 10$ e a ordem de pivotagem usada no sistema (4.22) é a mesma que a ordem original usada na determinação da solução do sistema (4.15) sem escolha parcial de pivot.

Pode-se agora perguntar qual a razão que faz com que o mesmo processo de escolha parcial de pivot possa dar resultados tão diferentes em sistemas diagonalmente equivalentes. Como afirmámos anteriormente, o número de condição da matriz A é o factor que mais condiciona a resolução numérica de um sistema de equações lineares. Mas se A e $B = D_1^{-1} A D_2$ são matrizes diagonalmente equivalentes, então

$$\begin{aligned} \text{cond}(B) &= ||D_1^{-1} A D_2|| \cdot || \left(D_1^{-1} A D_2\right)^{-1} || \\ &= ||D_1^{-1} A D_2|| \cdot ||D_2^{-1} A^{-1} D_1|| \\ &\leq ||D_1^{-1}|| \cdot ||A|| \cdot ||D_2|| \cdot ||D_2^{-1}|| \cdot ||A^{-1}|| \cdot ||D_1|| \end{aligned}$$

ou seja

$$\text{cond}(B) \leq \text{cond}(D_1) \cdot \text{cond}(D_2) \cdot \text{cond}(A) \tag{4.23}$$

Esta desigualdade significa que o número de condição de B pode ser bastante superior ao número de condição de A (é justamente o que acontece no exemplo anterior) e isso vai tornar a precisão numérica da solução obtida pelo processo de decomposição LU bastante má mesmo usando escolha parcial de pivot.

A desigualdade (4.23) mostra que as matrizes D_1 e D_2 devem ser escolhidas de modo a que $\text{cond}(D_1^{-1}AD_2)$ seja o menor possível. Apesar deste problema de optimização estar teoricamente resolvido, não existem algoritmos eficientes para encontrar essas matrizes D_1 e D_2. Uma técnica que tem fornecido bons resultados experimentais consiste em escolher as matrizes D_1 e D_2 de modo a que a matriz $B = D_1^{-1}AD_2$ de elemento genérico b_{ij} seja Equilibrada Por Linhas ou Por Colunas, isto é, satisfaça

$$\beta^{-1} \leq \max_{1\leq j\leq n} |b_{ij}| \leq 1, \quad \text{para todo o } \; i = 1, \ldots, n$$

$$\beta^{-1} \leq \max_{1\leq i\leq n} |b_{ij}| \leq 1, \quad \text{para todo o } \; j = 1, \ldots, n$$

respectivamente, com β a base do sistema de vírgula flutuante usado. Contudo mesmo neste caso pode haver problemas por não haver um processo único de equilibrar uma matriz, podendo-se obter uma matriz com bom ou mau número de condição. Assim, por exemplo, considere-se a matriz

$$A = \begin{bmatrix} 1 & 1 & 2\times 10^9 \\ 2 & -1 & 10^9 \\ 1 & 2 & 0 \end{bmatrix}$$

Se escolhermos

$$D_2 = \begin{bmatrix} 10^{-1} & & \\ & 10^{-1} & \\ & & 10^{-10} \end{bmatrix}, \quad D_1^{-1} = I$$

obtém-se a matriz

$$B_1 = D_1^{-1}AD_2 = \begin{bmatrix} 0.1 & 0.1 & 0.2 \\ 0.2 & -0.1 & 0.1 \\ 0.1 & 0.2 & 0.0 \end{bmatrix}$$

que é equilibrada por linhas e por colunas. Por outro lado, para

$$D_1^{-1} = \begin{bmatrix} 10^{-10} & & \\ & 10^{-10} & \\ & & 10^{-1} \end{bmatrix}, \quad D_2 = I$$

tem-se

$$B_2 = D_1^{-1}AD_2 = \begin{bmatrix} 10^{-10} & 10^{-10} & 0.2 \\ 2\times 10^{-10} & -10^{-10} & 0.1 \\ 0.1 & 0.2 & 0 \end{bmatrix}$$

que também é equilibrada por linhas e por colunas. É contudo possível verificar que o número de condição de B_1 é bastante inferior ao de B_2, pelo que a solução do sistema com a matriz B_1 deve ser bastante melhor. De notar que as linhas da matriz B_1 têm normas ℓ_∞ iguais, o que não acontece com B_2. Escalonar as linhas de modo a que tenham normas ℓ_∞ iguais é em geral um bom procedimento prático. De notar que a matriz B_1 satisfaz essa pretensão, o que não acontece com B_2 e muito menos com a matriz A.

4.6 Refinamento iterativo

O Refinamento Iterativo é uma técnica *a posteriori* que visa melhorar a precisão numérica de uma solução $\bar{x}$ do sistema (4.5) obtida pelo processo descrito anteriormente. Se $\bar{x}$ é a solução do sistema (4.5), então o resíduo

$$r = Ax - b \tag{4.24}$$

satisfaz $||r|| = 0$. Contudo, devido à ocorrência de erros de arredondamento no processo de resolução do sistema (4.5), o resíduo é normalmente não nulo. Se $\bar{z}$ é a solução do sistema

$$Az = r \tag{4.25}$$

então $\bar{x} + \bar{z}$ é uma solução do sistema (4.5) e tem uma precisão melhor que $\bar{x}$. O Refinamento Iterativo consiste exactamente em repetir o processo acabado de descrever até se atingir uma solução com uma precisão aceitável. É evidente que o refinamento iterativo está sujeito a erros de arredondamento, nomeadamente no cálculo do resíduo e na resolução do sistema (4.25). Para obstar ao primeiro problema o vector r é calculado em precisão dupla. O segundo tipo de erros de arredondamento é que vai determinar o número de iterações executadas até se obter uma dada precisão. Na figura 4.4 apresentamos os passos deste processo.

Algoritmo 8

Seja ϵ uma tolerância para zero que depende
da precisão que pretendemos para a solução do
sistema e $\bar{x}$ a solução obtida usando decomposição LU
com escolha parcial de pivot (se necessário).

(1) Calcule $r = b - A\bar{x}$ em precisão dupla.

(2) Se $||r|| < \epsilon$, $\bar{x}$ é a solução pretendida e páre.

De outro modo resolva, em precisão simples,

$$Az = r$$

e obtenha a solução $\bar{z}$.

Faça $\bar{x} = \bar{x} + \bar{z}$ e volte a (1).

Figura 4.4: Refinamento iterativo

Para estudarmos a convergência do processo, seja x a solução exacta do sistema (4.5) e seja $\bar{x}$ a solução calculada pelo processo de decomposição LU. Então o erro e dessa solução satisfaz a

$$e = x - \bar{x} = A^{-1}b - \bar{x} = A^{-1}(b - A\bar{x}) = A^{-1}r$$

Portanto

$$||e|| \leq ||A^{-1}|| \cdot ||r|| \tag{4.26}$$

Como a solução $\bar{x}$ foi obtida por um processo estável e

$$r_i = b_i - \sum_{j=1}^{n} a_{ij}\bar{x}_j, \quad i = 1, \ldots, n$$

então muitas parcelas do somatório anterior cancelarão umas com as outras e com os dígitos mais significativos das componentes b_i. Deste modo a i-ésima componente r_i de r tem uma grandeza sensivelmente igual à grandeza dos dígitos menos significativos do maior (em valor absoluto) dos termos $a_{ij}\bar{x}_j$, ou seja, r_i é da grandeza $\beta^{-t} \max_j |a_{ij}||\bar{x}_j|$, com β a base do sistema de vírgula flutuante. Então podemos escrever

$$||r|| \approx \beta^{-t}||A|| \cdot ||\bar{x}|| \tag{4.27}$$

Considerando que a desigualdade (4.26) se verifica quase como uma igualdade, então, de (4.26) e (4.27), tem-se

$$||e|| \approx \beta^{-t}||A|| \cdot ||A^{-1}|| \cdot ||\bar{x}||$$

ou seja

$$||e|| \approx \text{cond}(A)||\bar{x}||\beta^{-t} \tag{4.28}$$

Seja $\text{cond}(A) = \beta^p$. Então tem-se

$$||e|| \approx ||\bar{x}||\beta^{p-t}$$

Se $p > t$, então o erro da solução $\bar{x}$ é superior ao próprio $||\bar{x}||$ e portanto a solução $\bar{x}$ é bastante errónea. Isso confirma que soluções de sistemas com matrizes mal condicionadas são normalmente bastante pouco precisas. Assumamos portanto que $p \leq t-1$ e seja $q = t - p$. Então

$$\frac{||e||}{||\bar{x}||} \approx \beta^{-q}$$

ou seja, $\bar{x}$ tem sensivelmente q dígitos correctos.

Como r é calculado em precisão dupla, então podemos assumir que r tem t dígitos correctos. Quando resolvemos o sistema $Az = r$, então, de acordo com o discutido anteriormente, obtemos uma solução $\bar{z}$ com q dígitos correctos. Esses algarismos irão corrigir os primeiros q dígitos incorrectos de $\bar{x}$ e assim $\bar{x} + \bar{z}$ tem $2q$ dígitos correctos. Portanto, para se obter uma solução com t dígitos correctos são necessárias k iterações, onde k é o menor inteiro positivo que satisfaz $kq \geq t$.

Da discussão apresentada imediatamente se conclui que o número de condição da matriz A desempenha um papel determinante na convergência do processo de refinamento iterativo. Assim, se $\text{cond}(A) \approx \beta^t$, então o processo não converge. Por outro lado, a convergência será tanto mais rápida quanto menor for o número de condição de A, sendo a eficiência do processo bastante grande para matrizes bem condicionadas. A título de exemplo consideremos o sistema $Ax = b$ com

$$A = \begin{bmatrix} 3.3330 & 15920 & -10.333 \\ 2.2220 & 16.710 & 9.6120 \\ 1.5611 & 5.1791 & 1.6852 \end{bmatrix}, b = \begin{bmatrix} 15913 \\ 28.544 \\ 8.4254 \end{bmatrix}$$

que tem solução $x = (1, 1, 1)^T$. Se resolvermos esse sistema usando escolha parcial de pivot e $\epsilon_M = 10^{-5}$ obtemos a solução

$$\bar{x} = (1.20010, 0.99991, 0.92538)$$

Calculando o resíduo r em precisão dupla ($\epsilon_M = 10^{-10}$) e considerando apenas as cinco primeiras casas decimais de cada componente, vem

$$r = (-0.00518, 0.27413, -0.18616)$$

Agora há que resolver o sistema

$$Az = r$$

obtendo-se a solução $\bar{y} = (-0.20008, 0.00008, 0.07460)$. Donde

$$\bar{x} + \bar{z} = (1.00002, 0.99999, 0.99998)$$

Portanto a solução corrigida já tem quatro casas decimais correctas. É de notar que se calcularmos a inversa com o mesmo tipo de precisão ($\epsilon_M = 10^{-5}$) e seguidamente o número de condição da matriz A obtemos $\text{cond}(A) = 15999$. Portanto número de condição da matriz tem uma grandeza da ordem de 10^4 e daí sermos capazes de obter 4 casas decimais correctas em apenas uma iteração. Isso confirma a análise apresentada anteriormente. Segundo essa mesma análise não deve haver hipótese de melhorar a precisão da solução se prosseguirmos o processo de refinamento iterativo.

Apesar de se tratar de um processo bastante rápido para matrizes bem condicionadas, o refinamento iterativo não é muito usado, particularmente na resolução de sistemas de equações lineares

com matrizes densas. Com efeito, a experiência tem mostrado que a técnica de decomposição LU com escolha parcial de pivot conduz normalmente a soluções bastante precisas se precisão dupla for usada mesmo em casos de matrizes relativamente mal condicionadas. Por outro lado, se a matriz for bastante mal condicionada, o refinamento iterativo revelar-se-à incapaz de fornecer qualquer ajuda, pois normalmente não converge. Uma outra desvantagem prende-se com a determinação do resíduo que obriga a guardar uma cópia da matriz A e do vector b originais do sistema (4.5), o que conduz à duplicação do espaço de armazenagem. Além disso, não obstante a decomposição LU de A poder ser usada na resolução dos sucessivos sistemas $Az = r$, cada iteração do refinamento iterativo necessita de $2n^2 + \mathcal{O}(n)$ operações. Finalmente, o processo é dependente da arquitectura computacional em utilização.

Devido às razões aduzidas, o refinamento iterativo é pouco recomendado na resolução de sistemas de equações lineares, embora recentemente o seu uso tenha vindo a ser sugerido na resolução de sistemas de equações lineares com matrizes esparsas.

Da primeira iteração do refinamento iterativo podemos obter um estimador para o número de condição de uma matriz A. Com efeito, da discussão anterior, se $\bar{x}$ é a solução obtida pelo processo de decomposição LU com escolha parcial de pivot e $\bar{z}$ é a solução do sistema (4.25), então são introduzidos q dígitos correctos na solução $\bar{x} + \bar{z}$. Portanto

$$\frac{||\bar{z}||}{||\bar{x}||} \approx \beta^{-q} = \beta^{p-t}$$

Como $\text{cond}(A) = \beta^p$ então tem-se

$$\text{cond}(A) \approx \frac{||\bar{z}||}{||\bar{x}||}\beta^t \tag{4.29}$$

A experiência tem mostrado que esta estimativa não é má, mas existem processos mais eficientes para estimar o número de condição. Uma dessas técnicas é apresentada na secção seguinte.

4.7 Estimação do número de condição de uma matriz

Devido à importância que o número de condição assume na resolução de sistemas de equações lineares, há todo o interesse em se conhecer o seu valor antes de resolver um sistema. Contudo o seu cálculo exacto necessita da determinação da norma da inversa de A e isso é impraticável. Nesse sentido, têm vindo a ser desenvolvidos processos no sentido de encontrar estimativas para o número de condição. A seguir discutimos o estimador LINPACK, que é actualmente o mais usado na prática.

Se $Ay = d$, então $y = A^{-1}d$ e portanto

$$||A^{-1}||_\infty \geq \frac{||y||_\infty}{||d||_\infty} \tag{4.30}$$

Se $||y||_\infty$ é um número grande, então é muito possível que $\frac{||y||_\infty}{||d||_\infty}$ seja uma boa estimativa para $||A^{-1}||_\infty$. Nesse caso podemos estimar o número de condição da matriz A a partir de

$$\text{cond}_\infty(A) = ||A||_\infty \frac{||y||_\infty}{||d||_\infty} \tag{4.31}$$

O sucesso da estimativa LINPACK depende da escolha do vector d. Consideremos primeiramente o caso em que A é uma matriz triangular superior ($A = U$). A versão por colunas para a resolução do sistema $Uy = d$ pode ser escrita na seguinte forma:

Para $i = 1, \ldots, n$

  $p_i = 0$

Para $k = n, \ldots, 1$

  $y_k = (d_k - p_k)/u_{kk}$

  Para $i = 1, \ldots, k-1$

    $p_i = p_i + u_{ik} y_k$

onde se assume que o último ciclo não é efectuado quando $k = 1$.

Se considerarmos que $d_k \in \{-1, 1\}, k = 1, \ldots, n$, então $||d||_\infty = 1$. Como pretendemos que a norma do vector y seja elevada, então devemos escolher o valor de d_k igual a 1 ou -1 de modo a que y_k seja o maior possível. Assim d_k deve ser escolhido igual a 1 ou -1 consoante $(1 - p_k)/u_{kk}$ é maior ou menor que $(-1 - p_k)/u_{kk}$. Uma escolha simples para d_k consiste em fazer $d_k = -sign(p_k)$, com

$$sign(p_k) = \begin{cases} 1 & \text{se } p_k \geq 0 \\ -1 & \text{se } p_k < 0 \end{cases} \tag{4.32}$$

Como exemplo de aplicação deste algoritmo, consideremos a seguinte matriz triangular superior

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

É fácil de ver que a sua inversa é a seguinte matriz

$$A^{-1} = \begin{bmatrix} 1 & -1 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Portanto

$$\text{cond}_\infty(A) = ||A||_\infty \cdot ||A^{-1}||_\infty = 3 \cdot 4 = 12$$

Consideremos agora a aplicação do algoritmo LINPACK à determinação do número de condição de A. Como $u_{33} = 1$ então $d_3 = 1$ e tem-se

$$y_3 = \frac{1}{1} = 1$$

Para calcular y_2, determina-se primeiramente $u_{23} y_3 = -1$. Como esse elemento é negativo, então faz-se $d_2 = +1$ e portanto

$$y_2 = \frac{1 + 1}{1} = 2$$

Finalmente para calcular y_1 há que obter o valor de

$$u_{12} y_2 + u_{13} y_3 = 2 + 1 = 3$$

Portanto $d_1 = -1$ e tem-se $y_1 = -4$. Donde $y = (-4, 2, 1)$ e $||y||_\infty = 4$. Portanto a estimativa do número de condição fornecida pelo algoritmo é

$$\text{cond}_\infty(A) = ||A||_\infty \cdot ||y||_\infty = 12$$

que é exactamente o valor correcto do número de condição.

É agora fácil de desenvolver um processo semelhante para a determinação do número de condição de uma matriz triangular inferior. Esse processo deve incorporar a versão por linhas para a resolução de um sistema triangular inferior e o vector d deve ser escolhido de um modo semelhante ao apresentado anteriormente.

Consideremos agora o caso em que A é uma matriz não triangular. Então

$$PA = LU$$

com P uma matriz de permutação. Como é explicado em [Golub e Van Loan, cap. 4], o vector d deve ser escolhido a partir de

$$U^T L^T d = r$$

onde o vector r é definido de um modo semelhante ao vector d usado anteriormente. Esse vector pode então ser calculado a partir da resolução dos sistemas

$$U^T w = r, L^T d = w$$

onde as componentes do vector r são iguais a 1 ou -1 de modo a que o vector w tenha norma o maior possível. Então a estimativa do número de condição é dada pela forma (4.31) onde o vector y é a solução do sistema

$$LUy = Pd$$

A descrição do estimador LINPACK aparece na figura 4.5. É de acrescentar que a estimativa encontrada por este processo é um limite inferior para o número de condição de uma matriz. Experiência computacional com matrizes de números de condição conhecidos mostrou que as estimativas obtidas por este processo são normalmente bastante boas. Aliás no exemplo anterior, a estimativa obtida era o valor correcto do número de condição.

4.8 Método dos bordos para a decomposição LU

Suponhamos que conhecemos a decomposição LU da matriz $A \in \mathbb{R}^{n\times n}$, isto é, as matrizes L e U tais que $A = LU$ e pretendemos determinar a decomposição $\bar{L}\bar{U}$ da matriz

$$\bar{A} = \begin{bmatrix} A & b \\ a^T & \alpha_0 \end{bmatrix}$$

com a e b vectores de ordem n e α_0 um número real. Como

$$\bar{A} = \begin{bmatrix} L & \\ a^T U^{-1} & 1 \end{bmatrix} \begin{bmatrix} U & L^{-1}b \\ & \bar{\alpha}_0 \end{bmatrix}, \quad \bar{\alpha}_0 = (\bar{A}|A)$$

então tem–se

$$\bar{A} = \bar{L}\bar{U}$$

com

$$\bar{L} = \begin{bmatrix} L & \\ \bar{a}^T & 1 \end{bmatrix}, \quad \bar{U} = \begin{bmatrix} U & \bar{b} \\ & \bar{\alpha}_0 \end{bmatrix}$$

Portanto para obter a decomposição $\bar{L}\bar{U}$ de $\bar{A}$ basta determinar os vectores $\bar{a}$ e $\bar{b}$ e o número real $\bar{\alpha}_0$, o que pode ser feito a partir das seguintes expressões:

$$U^T\bar{a} = a, \quad L\bar{b} = b, \quad \bar{\alpha}_0 = \alpha_0 - \bar{a}^T\bar{b}$$

Suponhamos agora que a decomposição LU da matriz A se pode obter sem troca de linhas ou colunas. Então as submatrizes principais $A_{kk},\ k = 1, \ldots, n$ de A definidas por (4.14) são

Algoritmo 9

1. Seja $PA = LU$ com P uma matriz de permutação

2. Construção do vector d

$$w_1 = \frac{r_1}{u_{11}}$$

Para $k = 2, \ldots, n$

$$\left| \quad w_k = (r_k - \sum_{j=1}^{k-1} u_{jk} w_j)/u_{kk} \right.$$

onde

$$r_k = -sign\left(\sum_{j=1}^{k-1} u_{kj} w_j\right), \text{ com } sign \text{ o operador definido por (4.32)}$$

Resolva $L^T d = w$

3. Resolva

$$Lz = Pd$$
$$Uy = z$$

4. $\text{cond}(A) = ||A||_\infty \dfrac{||y||_\infty}{||d||_\infty}$

Figura 4.5: Estimador LINPACK do número de condição

não singulares. O processo referido pode ser usado $n-1$ vezes, terminando com a decomposição LU de A. A este processo dá–se o nome de método dos bordos, porque em cada iteração são transformados os elementos dos bordos inferiores de $A_{k+1,k+1}$. A representação esquemática e o algoritmo deste processo são apresentados na figura 4.6.

Como exemplo de ilustração deste processo, calculemos a decomposição LU da matriz

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 3 & -1 \\ 1 & 1 & 4 \end{bmatrix}$$

Então $L_1 = [1]$, $U_1 = [1]$. Na primeira iteração ($k = 1$) tem-se

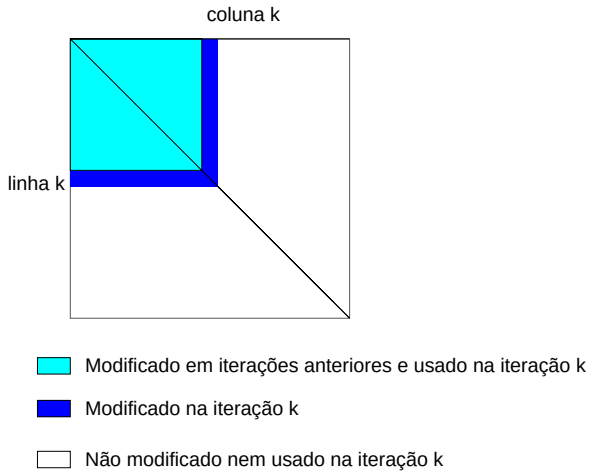
$$U_1 x = a \Leftrightarrow 1x = 1 \Rightarrow x = 1$$

$$L_1 y = b \Leftrightarrow 1y = 2 \Rightarrow y = 2$$

$$\alpha = a_{k+1,k+1} - x^T y = a_{22} - xy = 3 - 2 = 1$$

Donde

$$L_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, U_2 = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}$$



Passo 1: Faça $L_1 = [1]$, $U_1 = [a_{11}]$.

Passo 2: Para $k = 1, 2, \ldots, n-1$ *faça*

Seja $A_{kk} = L_k U_k$ *e* $A_{k+1,k+1} = \begin{bmatrix} A_{kk} & b \\ a^T & a_{k+1,k+1} \end{bmatrix}$

Então

Resolva $U_k^T x = a$

Resolva $L_k y = b$

Calcule $\alpha = a_{k+1,k+1} - x^T y$

Faça $A_{k+1,k+1} = \begin{bmatrix} L_k & 0 \\ x^T & 1 \end{bmatrix} \begin{bmatrix} U_k & y \\ 0 & \alpha \end{bmatrix} = L_{k+1} U_{k+1}$

Algoritmo 10

Figura 4.6: Método dos bordos

Na última iteração ($k = 2$) vem

$$U_2^T x = a \Leftrightarrow \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Leftrightarrow x = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$L_2 y = b \Leftrightarrow \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \Leftrightarrow y = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$\alpha = a_{33} - x^T y = 4 - \begin{bmatrix} 1 \\ -1 \end{bmatrix}^T \begin{bmatrix} 1 \\ -2 \end{bmatrix} = 1$$

Portanto

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & -1 & 1 \end{bmatrix}, U = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

Do que acabamos de apresentar conclui–se que em cada iteração k se determina uma linha da matriz L e uma coluna da matriz U da decomposição LU de A e que não há necessidade de qualquer espaço de armazenagem adicional durante o processo. Portanto o processo pode ser facilmente implementado para matrizes densas. No entanto o algoritmo 3 é de mais fácil compreensão e tem exactamente o mesmo número de operações. O facto de processar uma linha e uma coluna em cada iteração torna-o mais recomendável na resolução de sistemas de equações lineares com matrizes esparsas. Este assunto será discutido no capítulo 7.

4.9 Método directo para a decomposição LU

Este método consiste em obter as matrizes L e U da decomposição da matriz A directamente a partir da igualdade $A = LU$. Para isso escreve–se

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \dots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ & u_{22} & \dots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix}$$

Multiplicando a primeira linha de L pela matriz U e igualando à primeira linha de A, obtêm–se os valores u_{1j}, $j = 1, \dots, n$, dados por

$$u_{1j} = a_{1j}, \quad j = 1, 2, \dots, n$$

Os elementos l_{j1} da primeira coluna de L obtêm–se multiplicando a matriz L pela primeira coluna de U e igualando aos elementos da primeira coluna de A. Assim, tem–se

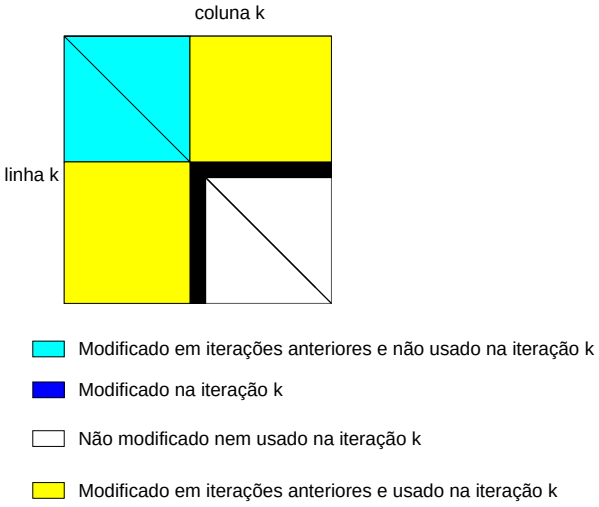
$$l_{j1} u_{11} = a_{j1}, j = 2, \dots, n \Rightarrow l_{j1} = \frac{a_{j1}}{u_{11}}, j = 2, \dots, n$$

O processo continua, obtendo–se os elementos da segunda linha da matriz U e da segunda coluna de L. Assim, para determinar a segunda linha da matriz U multiplica–se a segunda linha da matriz L pela matriz U e iguala–se à segunda linha da matriz A. Então tem–se

$$a_{2j} = l_{21} u_{1j} + u_{2j} \Rightarrow u_{2j} = a_{2j} - l_{21} u_{1j}, \quad j = 2, \dots, n$$

Do mesmo modo multiplicando a matriz L pela segunda coluna da matriz U e igualando à segunda coluna da matriz A, obtém–se a segunda coluna de L. Assim, tem–se

$$l_{j1} u_{12} + l_{j2} u_{22} = a_{j2} \Rightarrow l_{j2} = \frac{a_{j2} - l_{j1} u_{12}}{u_{22}}, j = 3, \dots, n$$



Algoritmo 11

$Para\ k = 1, 2, \ldots, n$

$Para\ j = k, k+1, \ldots, n$

$$u_{kj} = a_{kj} - \sum_{i=1}^{k-1} l_{ki} u_{ij}$$

$Para\ j = k+1, \ldots, n$

$$l_{jk} = \frac{a_{jk} - \sum_{i=1}^{k-1} l_{ji} u_{ik}}{u_{kk}}$$

Figura 4.7: Método directo

O processo seria agora repetido para as outras $n-2$ linhas de U e $n-2$ colunas de L. No algoritmo 11 da figura 4.7 apresentamos o método directo para obtenção da decomposição LU de A.

Tal como os outros dois métodos não é necessário qualquer espaço de armazenagem adicional para a determinação da decomposição LU, ocupando essas matrizes o espaço reservado à matriz A. Esquematicamente, a situação toma o aspecto apresentado na figura 4.7. Do algoritmo apresentado conclui–se que em cada iteração k se determina uma linha da matriz U e uma coluna da matriz L.

Consideremos novamente a matriz

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 3 & -1 \\ 1 & 1 & 4 \end{bmatrix}$$

e determinemos a sua decomposição LU usando o método directo que acabamos de descrever. Então tem-se $A = LU$, ou seja,

$$\begin{bmatrix} 1 & 2 & 1 \\ 1 & 3 & -1 \\ 1 & 1 & 4 \end{bmatrix} = \begin{bmatrix} 1 & & \\ l_{21} & 1 & \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ & u_{22} & u_{23} \\ & & u_{33} \end{bmatrix}$$

Multiplicando a primeira linha da matriz L pela matriz U e igualando à primeira linha de A, vem

$$\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & u_{13} \end{bmatrix}$$

e assim se obteve a primeira linha de U. Para calcular a primeira coluna de L multiplica-se a matriz L pela primeira coluna de U e iguala-se à primeira coluna de A. Assim tem-se

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} u_{11} \\ l_{21}u_{11} \\ l_{31}u_{11} \end{bmatrix} \Rightarrow \begin{cases} l_{21} = 1 \\ l_{31} = 1 \end{cases}$$

Seguidamente calcula-se a segunda linha de U, o que se pode fazer multiplicando a segunda linha de L por U e igualando à segunda linha de A. É de notar que o elemento da primeira coluna não necessita de ser calculado, pois é igual a zero. Donde

$$\begin{bmatrix} 3 & -1 \end{bmatrix} = \begin{bmatrix} 1 \times 2 + u_{22} & 1 \times 1 + u_{23} \end{bmatrix} \Rightarrow \begin{cases} u_{22} = 1 \\ u_{23} = -2 \end{cases}$$

O cálculo dos elementos da segunda coluna de L é feito multiplicando L pela segunda coluna de U e igualando à segunda coluna de A. Neste caso há apenas que calcular l_{32} o que se faz a partir de

$$1 = 1 \times 2 + l_{32} \times 1 \Rightarrow l_{32} = -1$$

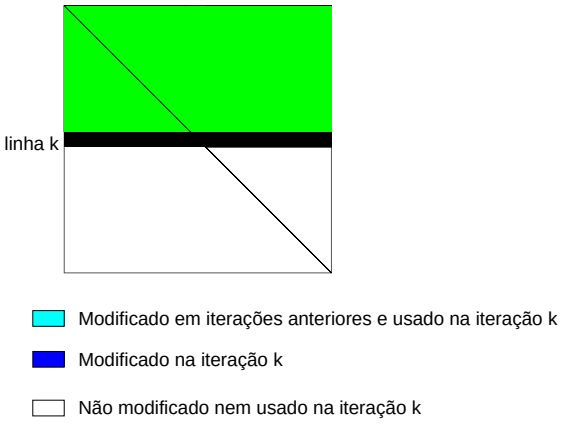
Para obter a decomposição LU de A basta calcular os elementos da terceira linha de U, o que se faz multiplicando L pela terceira coluna de U e igualando à terceira linha de A. Como os elementos não diagonais são iguais a zero, basta calcular u_{33}, o que é feito a partir de

$$4 = 1 \times 1 + (-1) \times (-2) + u_{33} \Rightarrow u_{33} = 1$$

Donde

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & -1 & 1 \end{bmatrix}, U = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

como anteriormente.



Algoritmo 12

Para $k = 1, 2, \ldots, n$

  Para $j = 1, \ldots, k-1$

$$l_{kj} = \left(a_{kj} - \sum_{i=1}^{j-1} l_{ki} u_{ij} \right) / u_{jj}$$

  Para $j = k, k+1, \ldots, n$

$$u_{kj} = a_{kj} - \sum_{i=1}^{k-1} l_{ki} u_{ij}$$

Figura 4.8: Método de Crout

O método directo apresentado baseia-se nas igualdades

$$\begin{aligned} u_{kj} &= a_{kj} - \sum_{i=1}^{k-1} l_{ki} u_{ij}, \quad j \geq k \\ l_{jk} &= \frac{a_{jk} - \sum_{i=1}^{k-1} l_{ji} u_{ik}}{u_{kk}}, \quad j > k \end{aligned} \tag{4.33}$$

onde os índices j e k variam segundo uma dada ordenação e prioridade. Fazendo j e k variar de maneiras diferentes, é possível obter outros dois métodos directos, em que em cada iteração k apenas a linha k ou a coluna k da decomposição LU é obtida. Assim, o método que satisfaz a primeira pretensão, normalmente denominado Método de Crout, é descrito no algoritmo 12 da figura 4.8. É de notar que o primeiro ciclo do algoritmo 12 não é executado quando $k = 1$. Esquematicamente a situação é a representada na figura 4.8.

Os métodos directos e o método dos bordos só podem ser usados se a decomposição LU se puder obter usando pivots na diagonal, isto é, se $\det(A_{kk}) \neq 0$, para todo o $k = 1, 2, \ldots, n$. Contudo tal condição não é suficiente para que o processo seja estável. Posteriormente estudaremos classes de matrizes que admitem processo de decomposição estável usando sempre como pivots elementos

diagonais das sucessivas matrizes reduzidas $A^{(k)}$.

Os métodos directos podem ser facilmente implementados para matrizes densas, ocupando os elementos l_{ij} e u_{ij} o mesmo espaço de armazenagem dos correspondentes elementos da matriz A do sistema. Tal como o método dos bordos, estes processos têm a grande vantagem de em cada iteração apenas transformarem uma linha e uma coluna da matriz A, sendo por isso muito usados na resolução de sistemas lineares com matrizes esparsas.

4.10 Decomposição LDU

No algoritmo 3 para a obtenção da decomposição LU de uma matriz A, assumimos que em cada iteração k a linha do pivot não é alterada e a coluna k é dividida pelo pivot, sendo a parte restante referente ao Complemento de Schur de uma determinada submatriz. É também possível desenvolver um processo de decomposição LU de modo a que a coluna k não seja alterada e seja a linha k a ser dividida. De acordo com esse processo, se

$$A = A^{(1)} = \begin{bmatrix} a_{11} & b^T \\ c & B \end{bmatrix} \tag{4.34}$$

e $a_{11} \neq 0$, então

$$A^{(2)} = \begin{bmatrix} a_{11} & \frac{1}{a_{11}} b^T \\ c & (A|a_{11}) \end{bmatrix} \tag{4.35}$$

e é fácil desenvolver um processo semelhante ao algoritmo 3 para a obtenção da decomposição LU da matriz A. Nessa decomposição os elementos diagonais da matriz U são todos iguais a um, enquanto que a diagonal de L contém os pivots das matrizes reduzidas. Assim por exemplo, seja

$$A = \begin{bmatrix} -1 & 1 & -1 \\ 1 & -2 & 1 \\ -1 & 0 & 3 \end{bmatrix}$$

Então tem-se

$$A^{(2)} = \begin{bmatrix} -1 & -1 & 1 \\ 1 & -1 & 0 \\ -1 & -1 & 4 \end{bmatrix}$$

Usando agora $a_{22}^{(2)}$ como pivot, obtém-se finalmente a matriz

$$A^{(3)} = \begin{bmatrix} -1 & -1 & 1 \\ 1 & -1 & 0 \\ -1 & -1 & 4 \end{bmatrix}$$

Donde

$$L = \begin{bmatrix} -1 & & \\ 1 & -1 & \\ -1 & -1 & 4 \end{bmatrix}, U = \begin{bmatrix} 1 & -1 & 1 \\ & 1 & 0 \\ & & 1 \end{bmatrix}$$

Do mesmo modo se podem desenvolver algoritmos correspondentes à forma orientada por colunas desse processo e aos métodos directo e dos bordos para esse tipo de decomposição LU. É evidente que os números de operações para a obtenção da decomposição LU e para a resolução do sistema $Ax = b$ são exactamente os mesmos que anteriormente. Por essa razão esse processo não é normalmente usado a não ser num caso que abordaremos posteriormente.

Suponhamos agora que $A = LU$ e sem perda de generalidade assumamos que $l_{ii} = 1$ para todo o $i = 1, \ldots, n$. Então podemos escrever $U = D\bar{U}$ com

$$D = diag(U) = \begin{bmatrix} u_{11} & & & \\ & u_{22} & & \\ & & \ddots & \\ & & & u_{nn} \end{bmatrix}, \quad \bar{U} = D^{-1}U = \begin{bmatrix} 1 & \frac{u_{12}}{u_{11}} & \cdots & \frac{u_{1n}}{u_{11}} \\ & 1 & \cdots & \frac{u_{2n}}{u_{22}} \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix}$$

e portanto toda a matriz A se pode escrever na forma $A = LD\bar{U}$, com L e $\bar{U}$ matrizes triangulares com elementos diagonais iguais a um e D uma matriz diagonal. O processo de obtenção da decomposição LDU de uma matriz A pode ser feita usando eliminação de Gauss de modo semelhante ao usado para obter a decomposição LU de uma matriz. Assim, se A é dada por (4.34) e $a_{11} \neq 0$, então

$$A^{(1)} = \begin{bmatrix} a_{11} & \frac{1}{a_{11}} b^T \\ \frac{1}{a_{11}} c & (A|a_{11}) \end{bmatrix} \tag{4.36}$$

e assim sucessivamente. Assim, em cada iteração do processo de obtenção da decomposição LDU, os elementos pertencentes às linhas abaixo da linha pivotal são transformadas usando as regras da decomposição LU explicadas na secção 4.2. A iteração termina com a divisão de todos os elementos da linha pivotal situados à direita do pivot por esse elemento. Assim por exemplo, consideremos novamente a matriz

$$A = \begin{bmatrix} -1 & 1 & -1 \\ 1 & -2 & 1 \\ -1 & 0 & 3 \end{bmatrix}$$

Então a_{11} é o pivot da primeira iteração. Os elementos abaixo da linha 1 são primeiramente transformados e tem-se

$$\begin{bmatrix} -1 & 1 & -1 \\ -1 & -1 & 0 \\ 1 & -1 & 4 \end{bmatrix}$$

Agora os elementos da linha pivotal a_{12} e a_{13} são divididos pelo pivot para se obter finalmente a matriz

$$A^{(2)} = \begin{bmatrix} -1 & -1 & 1 \\ -1 & -1 & 0 \\ 1 & 1 & 4 \end{bmatrix}$$

Procedendo agora de modo semelhante com o pivot $a_{22}^{(2)}$, vem

$$\begin{bmatrix} -1 & -1 & 1 \\ -1 & -1 & 0 \\ 1 & -1 & 4 \end{bmatrix}$$

e finalmente

$$A^{(3)} = \begin{bmatrix} -1 & -1 & 1 \\ -1 & -1 & 0 \\ 1 & -1 & 4 \end{bmatrix}$$

Como $n = 3$, então o processo de decomposição termina, e tem-se

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & -1 & 1 \end{bmatrix}, D = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 4 \end{bmatrix}, U = \begin{bmatrix} 1 & -1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Facilmente se conclui que o número de adições necessárias para a obtenção da decomposição LDU de uma matriz A é exactamente o mesmo que para a decomposição LU, mas o número de multiplicações é acrescido de

$$\sum_{k=1}^{n-1} (n-k) = \frac{n(n-1)}{2}$$

Uma vez obtida a decomposição LDU de uma matriz A, então a resolução do sistema $Ax = b$ resume-se a

$$\begin{cases} 1. \text{ Resolver } Ly = b \\ 2. \text{ Calcular } v = D^{-1}y, \text{ ou seja, fazer } v_i = \frac{y_i}{d_{ii}}, i = 1, \ldots, n \\ 3. \text{ Resolver } Ux = v \end{cases} \tag{4.37}$$

É fácil de ver que o número de operações para executar os passos referidos em (4.37) é o mesmo que para resolver os dois sistemas $Ly = b$ e $Ux = y$ referentes à decomposição LU de A. Concluimos assim que não há qualquer vantagem em usar a decomposição LDU quando a matriz A é não simétrica. Contudo, esta decomposição é usada no caso simétrico, como veremos mais adiante.

4.11 Classes de matrizes não simétricas com decomposição LU estável

Como referimos anteriormente, existem algumas classes de matrizes para as quais a decomposição LU sem escolha parcial de pivot é um processo estável.

Da discussão apresentada na secção 4.4, concluimos que o processo de decomposição LU sem trocas de linhas ou colunas é estável se é possível encontrar um limite superior para o factor de crescimento, definido por:

$$g_A = \frac{\max_{i,j\geq k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|}$$

Para matrizes densas o processo de escolha parcial de pivot é muito fácil de implementar, pelo que não é muito grande o interesse da investigação de classes de matrizes para as quais g_A é limitado superiormente. Contudo, como veremos no capítulo 7, esse problema assume uma notável importância na resolução de sistemas de equações lineares com matrizes esparsas.

É possível obter alguns resultados interessantes em relação às classes de matrizes introduzidas no capítulo 3. Nesta secção iremos apresentar esses resultados.

(i) Matrizes diagonalmente dominantes

Como vimos no capítulo 3, se A é uma matriz diagonalmente dominante não singular, então é possível obter a decomposição LU sem troca de linhas ou colunas. Além disso, o processo é estável, como mostra o seguinte teorema:

Teorema 4.4 *Se A é uma matriz diagonalmente dominante e não singular, então $g_A \leq 2$.*

Demonstração: Seja $A \in$ CDD não singular e consideremos a matriz

$$A^{(2)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(2)} & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix} \tag{4.38}$$

o que se obtém após a primeira iteração do processo de decomposição. Como $A \in$ CDD, então para $i = 2, \dots, n$ tem-se

$$|a_{i1}| \leq \sum_{j=2}^{n} |a_{j1}| \leq |a_{11}| \tag{4.39}$$

e portanto

$$|a_{i1}^{(2)}| = \frac{|a_{i1}|}{|a_{11}|} \leq 1, \ i = 2, \dots, n$$

Por outro lado

$$|a_{1j}^{(2)}| = |a_{1j}| \text{ para } j = 1, 2, \dots, n$$

Além disso, para $i, j \geq 2$

$$|a_{ij}^{(2)}| = \left| a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}} \right| \leq |a_{ij}| + \frac{|a_{i1}|}{|a_{11}|}|a_{1j}|$$

Portanto, usando (4.39), vem

$$\begin{aligned}\sum_{i=2}^{n}|a_{ij}^{(2)}| &\leq \sum_{i=2}^{n}|a_{ij}| + \frac{1}{|a_{11}|}\left[\sum_{i=2}^{n}|a_{i1}|\right]|a_{1j}| \\ &\leq \sum_{i=2}^{n}|a_{ij}| + |a_{1j}| = \sum_{i=1}^{n}|a_{ij}|\end{aligned}$$

Donde, para $i, j \geq 2$, tem-se

$$\begin{aligned}\max_{i,j\geq 2}|a_{ij}^{(2)}| &\leq \max_{j\geq 2}\sum_{i=2}^{n}|a_{ij}^{(2)}| \leq \max_{j\geq 2}\sum_{i=1}^{n}|a_{ij}| = \max_{j\geq 2}\left[\ \sum_{\substack{i=1\\ i\neq j}}^{n}|a_{ij}| + |a_{jj}|\ \right] \\ &\leq 2\max_{j\geq 2}|a_{jj}| \leq 2\max_{i,j}|a_{ij}|\end{aligned}$$

Para um passo k qualquer vem

$$|a_{ij}^{(k)}| = |a_{ij}^{(k-1)}| \text{ para } i \leq k \text{ ou } j < k$$

Além disso a submatriz principal correspondente às linhas $i > k$ e colunas $j > k$ é CDD e portanto

$$|a_{ij}^{(k)}| \leq 1,\ i = k+1, \ldots, n$$

e também

$$\sum_{i=k}^{n}|a_{ij}^{(k)}| \leq \sum_{i=k-1}^{n}|a_{ij}^{(k-1)}| \leq \sum_{i=1}^{n}|a_{ij}|$$

Donde

$$\max_{i,j\geq k}|a_{ij}^{(k)}| \leq \max_{j\geq k}\sum_{i=k}^{n}|a_{ij}^{(k)}| \leq \max_{j\geq k}\sum_{i=1}^{n}|a_{ij}| \leq 2\max_{i,j}|a_{ij}|$$

e $g_A \leq 2$, o que estabelece o resultado pretendido.

Se $A \in$ RDD, então $A^T \in$ CDD e portanto $A^T = LU$ e $g_{A^T} \leq 2$. Como $g_A = g_{A^T}$ então o teorema fica imediatamente demonstrado, obtendo-se a decomposição $A = U^T L^T$.

Para matrizes K diagonalmente dominantes o factor de crescimento é ainda menor. Com efeito verifica-se o seguinte resultado.

Teorema 4.5 *Se A é uma matriz* K *diagonalmente dominante, então $g_A \leq 1$.*

Demonstração: Iremos apenas considerar o caso de A ser diagonalmente dominante por colunas, pois o resultado para $A \in \text{RDD}_+$ obtém-se por argumentos semelhantes aos usados no teorema anterior.

Seja $A^{(2)}$ a matriz (4.38) que se obtém após a primeira iteração do processo de decomposição. Então, como vimos no teorema anterior, $|a_{i1}^{(2)}| \leq 1$, para $i = 2, \ldots, n$. Além disso

$$a_{1j}^{(2)} = a_{1j}, \quad j = 1, 2, \ldots, n$$

e, para $i \geq 2$ e $j \geq 2$, $a_{ij}^{(2)}$ é elemento do Complemento de Schur $(A|a_{11})$. Como essa matriz é CDD, então o seu elemento de maior valor absoluto pertence à sua diagonal. Portanto existe um $r \geq 2$ tal que

$$\max_{i,j\geq 2}|a_{ij}^{(2)}| = |a_{rr}^{(2)}|$$

Mas

$$|a_{rr}^{(2)}| = \left|a_{rr} - \frac{a_{1r}a_{r1}}{a_{11}}\right|$$

Como $a_{1r} \leq 0$ e $a_{r1} \leq 0$, então $a_{1r}a_{r1} \geq 0$ e vem

$$|a_{rr}^{(2)}| \leq |a_{rr}| \leq \max_{i,j} |a_{ij}|$$

Donde

$$\max_{i,j\geq 2} |a_{ij}^{(2)}| \leq \max_{i,j} |a_{ij}|$$

Como $(A|a_{11}) \in \text{CDD} \cap \text{K}$, então o mesmo raciocínio conduz a

$$\max_{i,j\geq 3} |a_{ij}^{(3)}| \leq \max_{i,j\geq 2} |a_{ij}^{(2)}| \leq \max |a_{ij}|$$

Continuando este tipo de processo, tem-se, para qualquer $k \geq 2$,

$$\max_{i,j\geq k} |a_{ij}^{(k)}| \leq \max_{i,j\geq k-1} |a_{ij}^{(k-1)}| \leq \ldots \leq \max_{i,j\geq 2} |a_{ij}^{(2)}| \leq \max |a_{ij}|$$

o que demonstra que $g_A \leq 1$.

Devido ao teorema 4.4, podemos obter a decomposição LU de uma matriz CDD não singular por eliminação de Gauss, pelo método dos bordos ou pelos métodos directos, sem necessidade de escolha parcial de pivot. Se $A \in \text{RDD}$ e é não singular, então a escolha parcial de pivot também não é necessária se procurarmos obter a decomposição LU de A^T e fizermos $A^T = U^T L^T$. Esse processo consiste simplesmente em obter a decomposição LU de A pela variante da eliminação de Gauss referida na secção anterior.

(ii) Matrizes positivas definidas

Para as matrizes destas classes o teorema 4.2 garante a existência de decomposição LU sem necessidade de permutação de linhas. Em relação à estabilidade do processo, consideremos o seguinte exemplo:

$$A = \begin{bmatrix} \epsilon & -m \\ m & \epsilon \end{bmatrix}$$

onde ϵ e m são dois números reais positivos. Então

$$A + A^T = \begin{bmatrix} 2\epsilon & 0 \\ 0 & 2\epsilon \end{bmatrix} \in \text{SPD}$$

e $A \in \text{PD}$ por (3.15). A decomposição LU pode ser calculada sem permutação de linhas e colunas e tem-se

$$A = \begin{bmatrix} 1 & 0 \\ m/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & -m \\ & \epsilon + m^2/\epsilon \end{bmatrix}$$

Se ϵ é muito pequeno e m é muito grande então o factor de crescimento é

$$g_A = \frac{\epsilon + \frac{m^2}{\epsilon}}{m} = \frac{m}{\epsilon} + \frac{\epsilon}{m}.$$

Portanto g_A é muitíssimo grande e isso indica a instabilidade da decomposição. Este exemplo mostra que se $A \in \text{PD}$ e é não simétrica, então a decomposição LU deve ser obtida com escolha parcial de pivot.

Podemos assim concluir que a decomposição LU sem escolha parcial de pivot não é estável para matrizes PD, assim como para todas as classes de matrizes que a contêm e que estão referidas na figura do capitulo 3.

Se $A \in \text{PSD}$ e é não simétrica, então A pode ser singular. Se $A \in \text{PSD}$ e é não singular, a_{11} pode ser nulo e são necessárias trocas de linhas ou colunas para a obtenção da decomposição LU. Portanto também neste caso a escolha parcial de pivot é recomendável.

Como veremos mais adiante, a situação é completamente diferente quando A é simétrica PD (ou PSD). Com efeito, iremos mostrar que o factor de crescimento é muito pequeno nesse caso. O próximo teorema estabelece que o mesmo é verdadeiro para matrizes não simétricas PD com elementos não diagonais não positivos.

Teorema 4.6 *Se A é uma matriz não simétrica* $\mathrm{PD} \cap \mathrm{Z}$*, então $g_A \leq 2$.*

Demonstração: Se $A \in \mathrm{PD} \cap \mathrm{Z}$, então $B = A + A^T \in \mathrm{PD} \cap \mathrm{Z}$. Além disso, o elemento máximo de B pertence à sua diagonal. Com efeito, se $|b_{rs}| = \max |b_{ij}|$ com $r \neq s$, então $b_{rs}^2 \geq b_{rr} b_{ss}$ e

$$\det \begin{bmatrix} b_{rr} & b_{rs} \\ b_{sr} & b_{ss} \end{bmatrix} \leq 0$$

o que contraria o teorema 3.13. Portanto, da definição da matriz B tem-se

$$|a_{ij} + a_{ji}| \leq 2 \max_{1 \leq r \leq n} a_{rr}$$

Seguidamente iremos provar que

$$|a_{ij}| \leq 2 \max_{1 \leq r \leq n} a_{rr} \tag{4.40}$$

para qualquer i e j. Como $A \in \mathrm{PD}$, então $a_{ii} > 0$ para todo $i = 1, \ldots, n$ e a desigualdade (4.40) é verdadeira para $i = j$. Se $i \neq j$, então $a_{ij} \leq 0$ e $a_{ji} \leq 0$ e portanto

$$|a_{ij}| \leq |a_{ij} + a_{ji}| \leq 2 \max_{1 \leq r \leq n} a_{rr}$$

o que demonstra a desigualdade pretendida.

Consideremos agora a matriz (4.38) obtida após a primeira iteração do processo de decomposição LU de A. Para $i \geq 2$ e $j \geq 2$ os elementos $a_{ij}^{(2)}$ pertencem ao Complemento de Schur $(A|a_{11})$ de a_{11} em A. Como $(A|a_{11}) \in \mathrm{PD} \cap \mathrm{Z}$ então da desigualdade (4.40) tem-se

$$\max_{i,j \geq 2} |a_{ij}^{(2)}| \leq 2 \max_{2 \leq r \leq n} a_{rr}^{(2)}$$

Mas

$$a_{rr}^{(2)} = a_{rr} - \frac{a_{r1} a_{1r}}{a_{11}} \leq a_{rr}$$

pois $a_{r1} \leq 0$, $a_{1r} \leq 0$ e $a_{11} > 0$. Donde

$$\max_{i,j \geq 2} |a_{ij}^{(2)}| \leq 2 \max_{1 \leq r \leq n} a_{rr} \leq 2 \max_{i,j} |a_{ij}|$$

De igual modo num passo k do processo de decomposição LU tem-se

$$\max_{i,j \geq k} |a_{ij}^{(k)}| \leq 2 \max_{r \leq k \leq n} a_{rr}^{(k)} \leq 2 \max_{1 \leq r \leq n} a_{rr} \leq 2 \max_{i,j} |a_{ij}|$$

Portanto

$$g_A = \frac{\max\limits_{i,j \geq k} |a_{ij}^{(k)}|}{\max\limits_{i,j} |a_{ij}|} \leq 2$$

como desejávamos demonstrar.

Seja agora $A \in \mathrm{PSD} \cap \mathrm{Z}$. Se $a_{11} = 0$, então pelo teorema 3.8 tem-se $a_{1j} = a_{j1} = 0$, para todo $j = 1, 2, \ldots, n$. Portanto A é singular. Por outro lado, se $a_{11} > 0$, então $(A|a_{11}) \in \mathrm{PSD} \cap \mathrm{Z}$. De

igual modo, se o primeiro elemento de $(A|a_{11})$ é nulo, então a linha e coluna correspondente dessa matriz também é nula e $(A|a_{11})$ é singular. Donde

$$\det(A) = a_{11}\det(A|a_{11}) = 0$$

e A é singular. De outro modo o primeiro elemento de $(A|a_{11})$ é positivo e é efectuado um novo passo do processo de decomposição. Este tipo de raciocínio pode ser repetido e assim concluir que se $A \in \text{PSD} \cap \text{Z}$ e é não singular, então $A = LU$. Além disso, uma demonstração semelhante à do teorema anterior permite concluir que também $g_A \leq 2$ nesse caso.

(iii) Matrizes H e K

Se A é uma matriz K ou H, então o teorema 4.3 garante a existência da sua decomposição LU. Em relação à sua estabilidade, é possível obter um limite superior para o factor de crescimento, pois verifica-se o seguinte resultado

Teorema 4.7

(i) *Se $A \in$ H, então existe um vector $d > 0$ tal que*

$$g_A \leq 2\frac{\max_i d_i}{\min_i d_i}$$

(ii) *Se $A \in$ K, existe um vector $d > 0$ tal que*

$$g_A \leq \frac{\max_i d_i}{\min_i d_i}$$

Demonstração: (i) Se $A \in$ H, então pelo teorema 3.30 existe uma matriz diagonal D de elementos diagonais d_i positivos tal que $DA \in$ SCDD. Portanto pelo teorema 4.3

$$B = DA = \bar{L}\bar{U}$$

com $\bar{L}$ e $\bar{U}$ matrizes triangulares inferiores e superiores respectivamente e $|\bar{l}_{ij}| \leq 1$. Por outro lado, devido à igualdade (3.17) cada elemento $b_{ij}^{(k)}$ da iteração k satisfaz

$$b_{ij}^{(k)} = d_i a_{ij}^{(k)}, \quad i, j \geq 2$$

com $a_{ij}^{(k)}$ o crrespondente elemento da matriz $A^{(k)}$ referente à decomposição LU de A. Como $B = DA \in$ SCDD, então pelo teorema 4.4, tem-se

$$|\bar{u}_{ij}| \leq \max_{i,j\geq k} |b_{ij}^{(2)}| \leq 2 \max_{i,j} |b_{ij}|$$

e portanto

$$\max_{i,j\geq k} |d_i a_{ij}^{(k)}| \leq 2 \left(\max_i d_i\right) \max_{i,j} |a_{ij}| \tag{4.41}$$

Consideremos agora as matrizes

$$L = D^{-1}\bar{L}D, \quad U = D^{-1}\bar{U} \tag{4.42}$$

Então

$$LU = D^{-1}\bar{L}DD^{-1}\bar{U} = D^{-1}\bar{L}\bar{U}$$

ou seja, as matrizes L e U definidas por (4.42) são os factores da decomposição LU de A. Mas dessas definições tem-se

$$\begin{aligned} |l_{ij}| &\leq \frac{d_j}{d_i}|\bar{l}_{ij}| \\ &\leq \frac{\max_i d_i}{\min_i d_i}|\bar{l}_{ij}| \end{aligned}$$

Como $|\bar{l}_{ij}| \leq 1$, vem

$$|l_{ij}| \leq \frac{\max\limits_i d_i}{\min\limits_i d_i} \tag{4.43}$$

Além disso

$$|u_{ij}| \leq \frac{1}{d_i}|\bar{u}_{ij}| \leq \frac{1}{d_i} \max_{i,j\geq k} |d_i a_{ij}^{(k)}|$$

Como $d_i > 0$, $i = 1, \ldots, n$, então de (4.41) vem

$$|u_{ij}| \leq 2\frac{\max\limits_i d_i}{\min\limits_i d_i} \max_{i,j} |a_{ij}| \tag{4.44}$$

O resultado pretendido é consequência de (4.43) e (4.44).

(ii) A demonstração é semelhante à anterior. Como $AD \in$ K, então pode-se usar o teorema 4.5 e concluir que

$$\max_{i,j\geq k} |d_i a_{ij}^{(k)}| \leq (\max d_i) \max_{i,j} |a_{ij}|$$

em vez de (4.41). Então o número 2 também desaparece da desigualdade (4.44) e a propriedade fica demonstrada.

Estes resultados permitem-nos concluir que em princípio não é necessário recorrer à escolha parcial de pivot para preservar a estabilidade no processo de decomposição LU de uma matriz K ou H. Existem contudo situações em que a escolha parcial de pivot é aconselhável. A título de exemplo consideremos a seguinte matriz

$$A = \begin{bmatrix} 2 & 0 & -x \\ -x & x & -1 \\ 0 & -1 & x \end{bmatrix} \tag{4.45}$$

Então $A \in$ Z para $x > 0$. Além disso $A \in$ K para $x > \sqrt{2}$, pois

$$A^{-1} = \frac{1}{x^2-2} \begin{bmatrix} x^2-1 & x & x^2 \\ x^2 & 2x & x^2+2 \\ x & 2 & 2x \end{bmatrix} \geq 0$$

se $x > \sqrt{2}$. A sua decomposição LU é dada por

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{x}{2} & 1 & 0 \\ 0 & -\frac{1}{x} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 2 & 0 & -x \\ 0 & x & -1-\dfrac{x^2}{2} \\ 0 & 0 & \dfrac{x}{2}-\dfrac{1}{x} \end{bmatrix}$$

Portanto para x grande

$$g_A = \frac{1+\frac{x^2}{2}}{x} = \frac{x}{2} + \frac{1}{x}$$

Donde g_A pode ser muito grande se o mesmo acontecer a x, o que indicia a instabilidade da decomposição. É de notar que o número de condição de A relativamente à norma ℓ_∞ é

$$\text{cond}_\infty(A) = ||A||_\infty ||A^{-1}||_\infty = (2x-1)\frac{2x^2+2x+2}{x^2-2}$$

que também tende para $+\infty$ quando o mesmo acontece com x. Poder-se-à perguntar se o valor de g_A está relacionado com o número de condição da matriz A. O próximo teorema indica que isso é verdadeiro pelo menos para as matrizes K.

Teorema 4.8 *Se $A \in \mathrm{K}$ e $a_{ii} = 1$, para $i = 1, \ldots, n$, então $g_A \leq \mathrm{cond}_\infty(A)$.*

Demonstração: Consideremos o sistema $Ax = e$, com e um vector de componentes $e_j = 1$, para $j = 1, \ldots, n$. Como $A^{-1} \geq 0$, então a solução d desse sistema é um vector positivo. Além disso

$$d_i = 1 - \sum_{j \neq i} a_{ij} d_j \geq 1, \quad i = 1, \ldots, n$$

por $a_{ij} \leq 0$, para $i \neq j$. Portanto $\min_i d_i \geq 1$ e pelo teorema anterior

$$\begin{aligned} g_A &\leq \frac{\max_i d_i}{\min_i d_i} \leq \max_i d_i = ||d||_\infty \\ &= ||A^{-1}e||_\infty \leq ||A^{-1}||_\infty ||e||_\infty = ||A^{-1}||_\infty \end{aligned}$$

Mas

$$||A^{-1}||_\infty = \frac{\mathrm{cond}_\infty(A)}{||A||_\infty}$$

Como $a_{ii} = 1, i = 1, \ldots, n$, então $||A||_\infty \geq 1$ e obtém-se a desigualdade pretendida.

Consideremos um sistema de equações lineares

$$Ax = b$$

com $A \in \mathrm{K}$. Se E é uma matriz diagonal de elementos diagonais $e_{ii} = \frac{1}{a_{ii}}$, então a matriz EA está nas condições do teorema anterior. Se EA é bem condicionada, o uso de pivots diagonais é um processo estável e conduz a uma solução com boa precisão numérica. Se pelo contrário EA é mal condicionada então poderá haver ou não problemas de estabilidade no processo de decomposição. Com efeito, g_A pode ser pequeno mesmo em casos em que $\mathrm{cond}_\infty(A)$ seja elevado. A título de exemplo consideremos novamente a matriz (4.45) e seja

$$E = \begin{bmatrix} \frac{1}{2} & & \\ & \frac{1}{x} & \\ & & \frac{1}{x} \end{bmatrix}$$

Então

$$B = EA = \begin{bmatrix} 1 & 0 & -\frac{x}{2} \\ -1 & 1 & -\frac{1}{x} \\ 0 & -\frac{1}{x} & 1 \end{bmatrix}$$

está nas condições do teorema anterior. Após duas iterações do processo de decomposição LU obtém-se

$$B^{(3)} = \begin{bmatrix} 1 & 0 & -\dfrac{x}{2} \\ -1 & 1 & \dfrac{x^2+2}{2x} \\ 0 & -\dfrac{1}{x} & \dfrac{x^2-2}{2x^2} \end{bmatrix}$$

Portanto para x grande

$$g_A = \frac{\frac{x^2+2}{2x}}{\frac{x}{2}} = \frac{x^2+2}{x^2} = 1 + \frac{2}{x^2} \approx 1$$

Por outro lado

$$B^{-1} = A^{-1}E^{-1} = \frac{1}{x^2-2} \begin{bmatrix} 2x^2-2 & x^2 & x^3 \\ 2x^2 & 2x^2 & x(x^2+2) \\ 2x & 2x & 2x^2 \end{bmatrix}$$

e

$$\mathrm{cond}_\infty(B) = ||B||_\infty ||B^{-1}||_\infty = \left(1 + \frac{x}{2}\right) \frac{x^3 + 4x^2 + 2x}{x^2 - 2}$$

tende para $+\infty$ com x.

Na secção 3.6 mostrámos que a melhor maneira de verificar se uma matriz Z é K consiste em estudar o sinal dos elementos da sua inversa. O próximo teorema mostra que a decomposição LU serve para o mesmo efeito.

Teorema 4.9 *Seja $A \in$ Z uma matriz de ordem n. Então $A \in K$ se e só se*

1. $A = LU$
2. $l_{ij} \leq 0$ *e* $u_{ij} \leq 0$, *para* $i \neq j$
3. $l_{ii} = 1$ *e* $u_{ii} > 0$, *para* $i = 1, \cdots, n$

Demonstração: Se $A \in$ K, então, pelo teorema 4.2, $A = LU$. Além disso se escrevermos

$$A = \begin{bmatrix} a_{11} & b^T \\ a & A_{11} \end{bmatrix}$$

então, após a primeira iteração do processo de decomposição, vem

$$A^{(2)} = \begin{bmatrix} a_{11} & b^T \\ \frac{1}{a_{11}}a & (A|a_{11}) \end{bmatrix}$$

Como $b \leq 0$, $a \leq 0$, $a_{11} > 0$, $(A|a_{11}) \in$ K, então o resultado fica provado por indução.

Para demonstrar a implicação inversa, bastar notar que $L \in$ K, $U \in$ K e portanto $L^{-1} \geq 0$, $U^{-1} \geq 0$, pelo teorema 3.27. Então tem-se $A^{-1} = U^{-1}L^{-1} \geq 0$ e $A \in$ K pelo mesmo teorema.

Devido às regras da decomposição LU é fácil de ver que se $A \in Z$ então os elementos não diagonais de L e U são sempre não positivos desde que os pivots sejam todos positivos. Portanto para estudar se $A \in$ K basta determinar a sua decomposição LU e verificar se todos os elementos diagonais de U são positivos. Assim por exemplo, mostremos que a matriz definida por (4.45) é K para $x > \sqrt{2}$. Os elementos diagonais de U são todos positivos se e só se

$$\frac{1}{x} > 0, \frac{x}{2} - \frac{1}{x} > 0$$

Então $x > 0$ e $x^2 - 2 > 0$. Donde $x > \sqrt{2}$, como queríamos mostrar.

Este processo de verificar se uma matriz Z é K é bastante mais eficiente do que o anteriormente mencionado na secção 3.6. Com efeito, o número de operações para obtenção da decomposição LU é sensivelmente $\frac{1}{3}$ do necessário para obter a inversa de A. Além disso basta estudar o sinal de n elementos diagonais, em vez de verificar n^2 elementos da inversa de A.

4.12 Resolução de sistemas com matrizes simétricas

Nesta secção iremos estudar a resolução de sistemas de equações lineares com matrizes simétricas e mostraremos que o número de operações necessárias para a obtenção da solução de um sistema com uma matriz simétrica é sensivelmente metade das necessárias para o caso da matriz ser não simétrica. Iremos começar por considerar sistemas com matrizes positivas definidas para finalmente estudarmos o caso de a matriz ser indefinida.

(i) Matrizes positivas definidas e semi-definidas

Se A é uma matriz SPD, então pelo que foi dito anteriormente a decomposição LDU existe sem necessidade de troca de linhas ou de colunas. Além disso, como $(A|a_{11})$ é simétrica, então o processo de obtenção da decomposição LDU de A termina com $U = L^T$. Portanto

$$A = LDL^T \tag{4.46}$$

com L uma matriz triangular inferior com elementos diagonais iguais a um e D uma matriz diagonal com elementos diagonais positivos. Essa decomposição é normalmente denominada decomposição de Cholesky. É evidente que a matriz L^T não precisa de ser determinada, o que vai reduzir substancialmente o número de operações necessárias à obtenção da decomposição. É ainda de acrescentar que o mesmo acontece para matrizes SPSD não singulares, pois pelo teorema 3.15, essas matrizes são SPD.

Seguidamente iremos debruçar-nos sobre a decomposição LDL^T de uma matriz $A \in$ SPD. Iremos começar por provar que o processo de obtenção da decomposição é estável e por isso não necessita de troca de linhas e colunas.

Teorema 4.10 *Se $A \in$ SPD (A é não singular* SPSD*), então $g_A \leq 1$, onde g_A é o factor de crescimento definido por (4.17).*

Demonstração: Começaremos por provar que $\max_{i,j} |a_{ij}|$ é um elemento da diagonal de A. Com efeito, se $|a_{rs}| = \max_{i,j} |a_{ij}|$ com $r \neq s$, então $a_{rs}^2 \geq a_{rr}a_{ss}$ e portanto

$$\det \begin{bmatrix} a_{rr} & a_{rs} \\ a_{sr} & a_{ss} \end{bmatrix} \leq 0$$

o que é absurdo, porque, pelo teorema 3.13, os menores principais de uma matriz SPD são todos positivos. Consideremos agora a matriz

$$A^{(2)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(2)} & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{bmatrix}$$

obtida após o primeiro passo do processo de decomposição LDL^T de A e provemos que

$$\max_{i,j \geq 2} |a_{ij}^{(2)}| \leq \max_{i,j} |a_{ij}| \tag{4.47}$$

Para $i, j \geq 2$, os elementos $a_{ij}^{(2)}$ pertencem ao Complemento de Schur $(A|a_{11})$. Como $(A|a_{11}) \in$ SPD, então, como vimos anteriormente, existe um $r \geq 2$ tal que

$$\max_{i,j \geq 2} |a_{ij}^{(2)}| = a_{rr}^{(2)} \tag{4.48}$$

Mas como A é simétrica, então

$$a_{rr}^{(2)} = |a_{rr}^{(2)}| = \left| a_{rr} - \frac{a_{r1}^2}{a_{11}} \right| \leq a_{rr} \leq \max_{i,j} |a_{ij}|$$

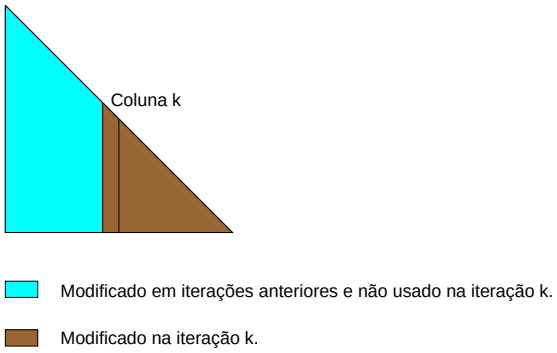
o que conjuntamente com (4.48) estabelece (4.47).

Considerando agora um passo k qualquer, então tem-se

$$\max_{i,j \geq k} |a_{ij}^{(k)}| \leq \max_{i,j \geq k-1} |a_{ij}^{(k-1)}| \leq \max_{i,j} |a_{ij}|$$

que estabelece o pretendido.

Por este teorema, toda a matriz $A \in$ SPD admite uma decomposição LDL^T estável, ou seja, não são precisas permutações de linhas ou colunas para a obter. Na prática, apenas as matrizes L e D são determinadas, modificando a diagonal e o triângulo inferior de A segundo as regras da eliminação de Gauss. O algoritmo para a obtenção das matrizes L e D é apresentado na figura 4.9. É de notar que a quantidade auxiliar aux é necessária porque o algoritmo não utiliza a parte



Algoritmo 13

$Para\ k = 1, 2, \ldots, n-1$

$Para\ i = k+1, \ldots, n$

$aux = a_{ik}$

$a_{ik} = \dfrac{a_{ik}}{a_{kk}}$

$Para\ j = k+1, \ldots, i$

$a_{ij} = a_{ij} - a_{jk} aux$

Figura 4.9: Decomposição LDL^T para uma matriz SPD

triangular superior das sucessivas matrizes reduzidas $A^{(k)}$. A iteração k do processo é representada esquematicamente na figura 4.9.

Facilmente se conclui que o número total de adições é dado por

$$\frac{1}{2}\sum_{k=1}^{n-1}(n-k)(n-k+1) = \frac{n(n^2-1)}{6} \tag{4.49}$$

enquanto que o número total de multiplicações e divisões é

$$\frac{n(n^2-1)}{6} + \sum_{k=1}^{n-1}(n-k) = \frac{n^3+3n^2-4n}{6} \tag{4.50}$$

Além disso são necessárias

$$\sum_{k=1}^{n-1}(n-k) = \frac{n(n-1)}{2} \tag{4.51}$$

cópias. Então o número total de operações é $\frac{n^3}{6} + \mathcal{O}(n^2)$ e portanto é aproximadamente metade do número de operações necessário para obter a decomposição LU de uma matriz não simétrica.

Notemos ainda que uma vez transformada a matriz A de acordo com o algoritmo 13, então as matrizes L e D são dadas por

$$L = \begin{bmatrix} 1 & & & \\ a_{21} & 1 & & \\ \vdots & & \ddots & \\ a_{n1} & a_{n2} & \dots & 1 \end{bmatrix} \quad D = \begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix} \tag{4.52}$$

Consideremos agora o sistema $Ax = b$, com $A \in \text{SPD}$. Se L e D são as matrizes obtidas pelo algoritmo 13, então a resolução do sistema consiste dos seguintes passos

$$\begin{cases} 1.\ \text{Resolver } Ly = b \\ 2.\ \text{Calcular } v = D^{-1}y, \text{ ou seja, fazer } v_i = \dfrac{y_i}{a_{ii}}, i = 1, \dots, n \\ 3.\ \text{Resolver } L^T x = v. \end{cases} \tag{4.53}$$

Como a matriz L tem elementos diagonais iguais a um, então o número total de operações para executar o processo (4.53) é exactamente o mesmo que para resolver os sistemas $Ly = b$ e $Ux = y$ necessários na resolução de um sistema com uma matriz não simétrica. O número total de operações para resolver o sistema $Ax = b$ pode então ser facilmente determinado, sendo dado por

$$\begin{cases} \text{Número de adições} & = & \dfrac{n^3+6n^2-7n}{6} \\ \\ \text{Número de multiplicações / divisões} & = & \dfrac{n^3+9n^2+2n}{6} \end{cases} \tag{4.54}$$

Portanto o número de operações é $\frac{n^3}{6} + \mathcal{O}(n^2)$.

Como exemplo de ilustração, consideremos o sistema de equações lineares $Ax = b$, onde

$$A = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 3 & 0 \\ 1 & 0 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ -1 \\ -2 \end{bmatrix}$$

Segundo as regras do algoritmo 13, apenas os elementos do triângulo inferior de A são transformados, sendo necessário em cada iteração efectuar uma cópia dos elementos da coluna pivotal abaixo do pivot. Assim na primeira iteração considera-se a matriz

$$A^{(1)} = \left[\begin{array}{ccc} 1 & -1 & 1 \\ -1 & 3 & \\ 1 & 0 & 3 \end{array}\right]$$

cópias

a transformar

e tem-se

$$A^{(2)} = \left[\begin{array}{ccc} 1 & & \\ -1 & 2 & 1 \\ 1 & 1 & 2 \end{array}\right]$$

cópia

Efectuando agora a segunda iteração, vem

$$A^{(3)} = \left[\begin{array}{ccc} 1 & & \\ -1 & 2 & \\ 1 & \frac{1}{2} & \frac{3}{2} \end{array}\right]$$

Como $n = 3$, então a decomposição LDL^T de A foi obtida e tem-se

$$L = \left[\begin{array}{ccc} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & \frac{1}{2} & 1 \end{array}\right], \quad D = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & \frac{3}{2} \end{array}\right]$$

Para resolver o sistema $Ax = b$, há que primeiramente obter a solução do sistema $Ly = b$, ou seja,

$$\left\{\begin{array}{rrrcr} y_1 & & & = & 0 \\ -y_1 & +y_2 & & = & -1 \\ y_1 & +\frac{1}{2}y_2 & +y_3 & = & -2 \end{array}\right.$$

Então $y_1 = 0$, $y_2 = -1$ e $y_3 = -\dfrac{3}{2}$. Agora

$$u = D^{-1}y = \begin{bmatrix} 0 \\ -\dfrac{1}{2} \\ -1 \end{bmatrix}$$

Finalmente tem de se resolver o sistema $L^T x = u$, ou seja,

$$\begin{cases} x_1 & -x_2 & +x_3 & = & 0 \\ & x_2 & +\dfrac{1}{2}x_3 & = & -\dfrac{1}{2} \\ & & x_3 & = & -1 \end{cases}$$

Então $x_3 = -1$, $x_2 = 0$, $x_1 = 1$ é a solução do sistema.

Como foi referido anteriormente, se $A \in$ SPD então a resolução do sistema $Ax = b$ apenas necessita da consideração da diagonal e da parte abaixo da diagonal da matriz A, pelo que apenas esses elementos têm que ser armazenados. Tal pode ser levado a cabo usando dois vectores d e s de ordens n e $\frac{n(n-1)}{2}$ respectivamente, que são definidos de acordo com as seguintes correspondências:

$$\begin{array}{cccc} a_{11} & a_{22} & \dots & a_{nn} \\ \updownarrow & \updownarrow & \dots & \updownarrow \\ d_1 & d_2 & \dots & d_n \end{array} \qquad \Leftrightarrow \qquad d_i = a_{ii}, i = 1, 2, \dots, n$$

$$\begin{array}{cccccccc} a_{21} & a_{31} & a_{32} & a_{41} & a_{42} & a_{43} & \dots & a_{n,n-1} \\ \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \cdots & \updownarrow \\ s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & \dots & s_{\frac{n(n-1)}{2}} \end{array} \quad \Leftrightarrow \quad a_{ij} = s_l, \ \text{ com } l = \frac{(i-1)(i-2)}{2} + j \tag{4.55}$$

Tendo em conta o algoritmo 13 para a obtenção das matrizes L e D da decomposição LDL^T da matriz A e os algoritmos 1 e 2 para a resolução dos sistemas triangulares superior e inferior, então o algoritmo para a resolução do sistema $Ax = b$, com $A \in$ SPD armazenada de acordo com o esquema (4.55), tem a forma apresentada na figura 4.10.

Portanto a resolução de um sistema $Ax = b$, com $A \in$ SPD necessita de um espaço de armazenagem igual a

$$n + \frac{n(n-1)}{2} + n = \frac{n(n+3)}{2},$$

de um número de operações da ordem $\frac{n^3}{6} + \mathcal{O}(n^2)$, e de operações secundárias para manobrar com o vector s. Assim, se por um lado este processo tem a vantagem de reduzir o espaço de armazenagem, por outro tem a desvantagem de efectuar um número elevado de operações com números inteiros. Essas operações são todavia mais rápidas de efectuar, em virtude de serem efectuadas sobre números inteiros.

Na secção 3.3 realçámos a dificuldade em provar que uma matriz (simétrica ou não) A é PD ou PSD a partir das suas definições. Seguidamente mostramos que a decomposição de Cholesky é particularmente importante para esse efeito.

Teorema 4.11 *Se A é uma matriz simétrica de ordem n, então*

$$A \in \text{SPD} \Leftrightarrow A = LDL^T \ e \ d_{ii} > 0, \ i = 1, \dots, n$$

Demonstração: Devido ao teorema anterior basta provar que se $A = LDL^T$ com $d_{ii} > 0$, $i = 1, \dots, n$ então $A \in$ SPD. Para isso, seja $x \neq 0$ um vector de ordem n qualquer e provemos que $x^T Ax > 0$. Tem-se

$$x^T Ax = x^T(LDL^T)x = (L^T x)^T D(L^T x)$$

Algoritmo 14

Para $k = 1, 2, \ldots, n-1$

Determinação de L e D

- *Para* $i = k+1, \ldots, n$
  - $l = \frac{(i-1)(i-2)}{2} + k$
  - $aux = s_l$
  - $s_l = \frac{s_l}{d_k}$
  - $d_i = d_i - s_l aux$
  - *Para* $j = k+1, \ldots, i-1$
    - $\alpha = l + j - k$
    - $\gamma = \frac{(j-1)(j-2)}{2} + k$
    - $s_\alpha = s_\alpha - s_\gamma aux$

Para $k = 2, \ldots, n$

Resolução de $Ly = b$

- *Para* $j = 1, \ldots, k-1$
  - $l = \frac{(k-1)(k-2)}{2} + j$
  - $b_k = b_k - s_l b_j$

Para $k = 1, \ldots, n$

Cálculo de $y = D^{-1}y$

- $b_k = \frac{b_k}{d_k}$

Para $k = n-1, \ldots, 1$

Resolução de $Lx = y$

- *Para* $j = k+1, \ldots, n$
  - $l = \frac{(j-1)(j-2)}{2} + k$
  - $b_k = b_k - s_l b_j$

Figura 4.10: Algoritmo para a resolução do sistema $Ax = b$ com $A \in$ SPD

Se $y = L^T x$, podemos escrever

$$x^T Ax = y^T Dy = \sum_{i=1}^{n} y_i^2 d_i$$

Mas L é não singular e portanto $y \neq 0$. Como $d_{ii} > 0$, $i = 1, \ldots, n$, então $x^T Ax > 0$.

Este teorema indica que para mostrar que uma matriz simétrica A é SPD basta calcular a sua decomposição de Cholesky e investigar o sinal dos elementos diagonais de D. Se existir um $d_{ii} \leq 0$, então $A \notin$ SPD e o processo de decomposição deve parar. De outro modo a decomposição LDL^T é obtida e a matriz A é SPD. Como para uma matriz A não simétrica se tem

$$A \in \text{PD} \Leftrightarrow A + A^T \in \text{SPD}$$

então também é possível verificar se uma matriz não simétrica é PD, investigando se o mesmo acontece à sua parte simétrica $A + A^T$. A título de exemplo consideremos a seguinte matriz não simétrica

$$A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 1 & 1 \\ -1 & 1 & 2 \end{bmatrix}$$

Então

$$A + A^T = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 4 \end{bmatrix}$$

Na primeira iteração do processo de cálculo da decomposição LDL^T de $B = A + A^T$ tem-se

$$B^{(2)} = \begin{bmatrix} 2 & & \\ \frac{1}{2} & \frac{3}{2} & \\ 1 & 1 & 2 \end{bmatrix}$$

Então $b_{22}^{(2)} > 0$ e pode ser usado como pivot na segunda iteração. Efectuando esse passo vem

$$B^{(3)} = \begin{bmatrix} 2 & & \\ \frac{1}{2} & \frac{3}{2} & \\ 1 & \frac{2}{3} & \frac{4}{3} \end{bmatrix}$$

Portanto

$$D = \begin{bmatrix} 2 & & \\ & \frac{3}{2} & \\ & & \frac{4}{3} \end{bmatrix}$$

Como todos os elementos diagonais de D são positivos, então $A + A^T \in$ SPD e $A \in$ PD.

Consideremos agora o caso em que A é uma matriz SPSD de ordem n. Se A é não singular, então $A \in$ SPD e $A = LDL^T$ com $d_{ii} > 0$, $i = 1, \ldots, n$. Suponhamos que A é singular. Então

tem de ocorrer uma iteração k em que $a_{kk}^{(k)} = 0$. Tendo em conta os teoremas 3.9 e 3.12 podemos escrever $A^{(k)}$ na forma

$$A^{(k)} = \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A_{22}^{(k)} \end{bmatrix}$$

com

$$A_{22}^{(k)} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & A_{33}^{(k)} & \\ 0 & & \end{bmatrix} = (A|A_{11})$$

Como todos os elementos da primeira coluna são iguais a zero, não se efectua qualquer operação e passa-se imediatamente à iteração seguinte. Se procedermos deste modo sempre que ocorre um pivot nulo, então obtemos no fim do processo a decomposição LDL^T de A em que alguns d_{ii} são iguais a zero e

$$d_{ii} = 0 \Rightarrow l_{ji} = 0,\ j = i+1, \ldots, n$$

A título de exemplo, procuremos obter a decomposição LDL^T da matriz

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & 3 \end{bmatrix}$$

Na primeira iteração $a_{11} > 0$ e portanto pode-se usar como pivot, obtendo-se

$$A^{(2)} = \begin{bmatrix} 1 & & \\ 1 & 0 & \\ -1 & 0 & 2 \end{bmatrix}$$

Como $a_{22}^{(2)} = 0$ e $a_{32}^{(2)} = 0$, não se efectua qualquer operação na segunda iteração. Então a decomposição LDL^T de A foi obtida e tem-se

$$L = \begin{bmatrix} 1 & & \\ 1 & 1 & \\ -1 & 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & & \\ & 0 & \\ & & 2 \end{bmatrix}$$

Mostrámos assim que a decomposição LDL^T de uma matriz $A \in$ SPSD pode ser sempre obtida de uma forma semelhante à do caso em que $A \in$ SPD. Tal como anteriormente è possível provar o seguinte resultado.

Teorema 4.12 *Seja A uma matriz simétrica de ordem n. Então $A \in$ SPSD se e só se*

1. $A = LDL^T$.
2. $d_{ii} \geq 0$, $i = 1, \ldots, n$.
3. $d_{ii} = 0 \Rightarrow l_{ji} = 0$, $j = i+1, \ldots, n$.

Demonstração: O processo de decomposição LDL^T de $A \in$ SPSD apresentado anteriormente mostra que a implicação $\Rightarrow$ é verdadeira. A demonstração de $\Leftarrow$ é semelhante à do teorema 4.11.

Este teorema permite estabelecer se uma matriz A é SPSD ou não usando a sua decomposição LDL^T. Com efeito $A \notin$ SPSD se para uma determinada iteração k um dos seguintes casos ocorre:

(i) $a_{kk}^{(k)} < 0$

(ii) $a_{kk}^{(k)} = 0$ e $a_{jk}^{(k)} \neq 0$ para certo $j > k$.

De outro modo é possível obter a sua decomposição LDL^T com $d_{ii} \geq 0$ e $A \in$ SPSD. Como uma matriz não simétrica A é PSD se e só se $A + A^T \in$ SPSD, então é possível verificar se A é PSD, utilizando este tipo de processo para a sua parte simétrica $A + A^T$. Assim, por exemplo, considere-se a matriz simétrica

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 1 & 2 \\ -1 & 2 & 3 \end{bmatrix}$$

Como $a_{11} > 0$, então pode ser usado como pivot na primeira iteração e tem-se

$$A^{(2)} = \begin{bmatrix} 1 & & \\ 1 & 0 & \\ -1 & 3 & 2 \end{bmatrix}$$

Então $a_{22}^{(2)} = 0$ e $a_{32}^{(2)} \neq 0$, e isso implica que $A \notin$ SPSD. Note-se que o processo de decomposição pára nesta iteração.

No capítulo 3 mostrámos que as matrizes PD (PSD) constituem uma subclasse das matrizes P (P_0). Os dois teoremas anteriores permitem mostrar que PD = P e PSD = P_0 para matrizes simétricas. Com efeito podemos estabelecer os seguintes resultados.

Teorema 4.13 *Se A é uma matriz simétrica, então*

$$A \in \text{PD} \Leftrightarrow A \in \text{P}$$

$$A \in \text{PSD} \Leftrightarrow A \in \text{P}_0$$

Demonstração: Como PD $\subset$ P, então basta provar a implicação $\Leftarrow$. Se $A \in$ P é uma matriz de ordem n, então $A = LDU$ com $d_{ii} > 0$, $i = 1, \ldots, n$. Se além disso A é simétrica, então $U = L^T$. Portanto $A = LDL^T$ com $d_{ii} > 0$ e $A \in$ PD pelo teorema 4.11. A demonstração da segunda equivalência é semelhante e baseia-se no teorema 4.12.

Tal como para as matrizes não simétricas, a decomposição LDL^T de uma matriz $A \in$ SPD pode ser obtida usando os métodos directos e dos bordos. Estes métodos não são usados em matrizes densas, mas assumem particular importância na resolução de sistemas de equações lineares com matrizes esparsas, como veremos posteriormente. Seguidamente, iremos apresentar esses dois processos.

(ii) Método directo para a decomposição LDL^T de $A \in$ SPD

Este processo consiste em obter as matrizes L e D da decomposição LDL^T da matriz A directamente a partir da igualdade $A = LDL^T$. Para isso escreve-se

$$\begin{bmatrix} a_{11} & a_{21} & \ldots & a_{n1} \\ a_{21} & a_{22} & \ldots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \ldots & 1 \end{bmatrix} \begin{bmatrix} d_{11} & & & \\ & d_{22} & & \\ & & \ddots & \\ & & & d_{nn} \end{bmatrix} \begin{bmatrix} 1 & l_{21} & \ldots & l_{n1} \\ & 1 & \ldots & l_{n2} \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \ldots & 1 \end{bmatrix} \begin{bmatrix} d_{11} & d_{11}l_{21} & \ldots & d_{11}l_{n1} \\ & d_{22} & \ldots & d_{22}l_{n2} \\ & & \ddots & \vdots \\ & & & d_{nn} \end{bmatrix}$$

Multiplicando a matriz L pela primeira coluna de DL^T e igualando à primeira coluna da matriz A, obtém-se

$$\begin{aligned} d_{11} &= a_{11} \\ a_{i1} = l_{i1}d_{11} \quad &\Rightarrow \quad l_{i1} = \frac{a_{i1}}{d_{11}},\ i = 2, \ldots, n \end{aligned}$$

Para obter os elementos da segunda coluna de L e o elemento d_{22}, tem-se

$$a_{22} = l_{21}d_{11}l_{21} + d_{22} \Rightarrow d_{22} = a_{22} - l_{21}d_{11}l_{21}$$

$$a_{i2} = l_{i1}d_{11}l_{21} + l_{i2}d_{22} \Rightarrow l_{i2} = \frac{a_{i2} - l_{i1}d_{11}l_{21}}{d_{22}},\ i = 3, \ldots, n$$

O processo poderia agora ser continuado obtendo-se sucessivamente as outras colunas da matriz L e os elementos diagonais da matriz D. Na prática os elementos de L e D são obtidos transformando a matriz A de acordo com o esquema apresentado anteriormente. A figura 4.11 apresenta os passos do algoritmo e a sua ilustração esquemática.

A figura 4.11 representa esquematicamente a iteração k deste processo. Se compararmos os algoritmos 13 e 15 facilmente concluímos que o método directo necessita de igual número de adições, não necessita de quaisquer cópias, mas precisa de mais

$$\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2} \tag{4.56}$$

multiplicações respeitantes aos produtos $aux = a_{kk}a_{kt}$. Notemos ainda que em cada iteração do método directo se determina uma coluna da matriz L e um elemento da diagonal da matriz D. Esse facto torna o método particularmente recomendado para a resolução de sistemas com matrizes esparsas SPD.

(iii) Método dos bordos para a decomposição LDL^T de $A \in$ SPD

Se A_{kk} são as submatrizes principais de A definidas por (4.14), então, como vimos anteriormente, $\det(A_{kk}) \neq 0$ para todo o $k = 1, 2, \ldots, n$ e

$$A_{k+1,k+1} = \begin{bmatrix} A_{kk} & a \\ a^T & a_{k+1,k+1} \end{bmatrix} \tag{4.57}$$

Se $A = L_k D_k L_k^T$, então

$$A_{k+1,k+1} = \begin{bmatrix} L_k & \\ \bar{a}^T & 1 \end{bmatrix} \begin{bmatrix} D_k & \\ & d_{k+1,k+1} \end{bmatrix} \begin{bmatrix} L_k^T & \bar{a} \\ & 1 \end{bmatrix}$$

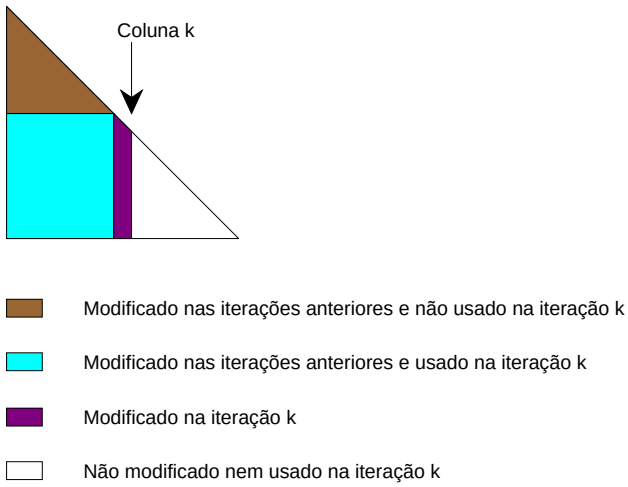
e portanto a obtenção da decomposição LDL^T de $A_{k+1,k+1}$ resume-se ao cálculo do vector $\bar{a}$ e do elemento $d_{k+1,k+1}$. Multiplicando as três matrizes do membro direito da igualdade anterior, tem-se

$$\begin{bmatrix} A_{kk} & a \\ a^T & 1 \end{bmatrix} = \begin{bmatrix} L_k D_k L_k^T & L_k D_k \bar{a} \\ \bar{a} D_k L_k^T & d_{k+1,k+1} + \bar{a}^T D_k \bar{a} \end{bmatrix}$$

Igualando as duas matrizes elemento a elemento, obtém-se

$$\begin{cases} L_k D_k \bar{a} = a \\ d_{k+1,k+1} = a_{k+1,k+1} - \bar{a}^T D_k \bar{a} \end{cases} \tag{4.58}$$

Deste modo, considerando inicialmente $L_1 = [1]$ e $D_1 = [a_{11}]$ e repetindo o processo $n-1$ vezes, obtém-se a decomposição LDL^T da matriz A. A figura 4.12 apresenta o algoritmo para o método dos bordos assim descrito.



Algoritmo 15

$Para\ k = 1, 2, \ldots, n$

$Para\ t = 1, \ldots, k-1$

$aux = a_{tt} a_{kt}$

$a_{kk} = a_{kk} - a_{kt} aux$

$Para\ i = k+1, \ldots, n$

$a_{ik} = a_{ik} - a_{it} aux$

$Para\ i = k+1, \ldots, n$

$a_{ik} = \frac{a_{ik}}{a_{kk}}$

Figura 4.11: Método directo para matrizes SPD

O número de operações para a obtenção da decomposição LDL^T da matriz A pelo método dos bordos é exactamente o mesmo que para o método directo.

Como é evidenciado na figura 4.12, o método dos bordos determina em cada iteração uma nova linha da matriz L e um elemento diagonal da matriz D. Por essa razão o método dos bordos é bastante usado na resolução de sistemas com matrizes esparsas.

(iv) Decomposição LL^T

Seja $A \in$ SPD. Então podemos escrever $A = \bar{L}\bar{D}\bar{L}^T$, com $\bar{L}$ uma matriz triangular inferior com elementos diagonais iguais a um e $\bar{D}$ uma matriz diagonal com elementos diagonais positivos. Se $\bar{D}^{1/2}$ é a matriz diagonal cujos elementos diagonais são as raízes quadradas dos elementos diagonais de $\bar{D}$, então tem-se

$$A = \bar{L}\bar{D}\bar{L}^T = \bar{L}\bar{D}^{1/2}\bar{D}^{1/2}\bar{L}^T = LL^T \tag{4.59}$$

com $L = \bar{L}\bar{D}^{1/2}$. Portanto toda a matriz $A \in$ SPD se pode escrever na forma $A = LL^T$, com L uma matriz triangular inferior. Essa decomposição de A é também denominada Decomposição de Cholesky. Tal como anteriormente, a decomposição pode ser obtida por eliminação de Gauss por linhas ou por colunas, ou ainda usando os métodos directo ou dos bordos. Esses algoritmos são pequenas modificações dos algoritmos correspondentes para a obtenção da decomposição LDL^T de uma matriz $A \in$ SPD. A título de exemplo apresentamos o algoritmo baseado na eliminação de Gauss por linhas. Se

$$A = \begin{bmatrix} a_{11} & a^T \\ a & B \end{bmatrix} \tag{4.60}$$

então

$$A^{(2)} = \begin{bmatrix} \sqrt{a_{11}} & \frac{1}{\sqrt{a_{11}}}a^T \\ \frac{1}{\sqrt{a_{11}}}a & (A|a_{11}) \end{bmatrix} \tag{4.61}$$

Portanto o algoritmo para a obtenção da matriz L da decomposição LL^T usando a eliminação de Gauss por linhas tem a forma apresentada na figura 4.13. Além disso a iteração k pode ser apresentada esquematicamente através da figura 4.9 para a obtenção da decomposição LDL^T de A. Como exemplo de ilustração, calculemos a decomposição LL^T da matriz

$$A = \begin{bmatrix} 4 & -2 & 2 \\ -2 & 5 & 0 \\ 2 & 0 & 2 \end{bmatrix}$$

Na primeira iteração o pivot a_{11} é substituído pela sua raíz quadrada. Os elementos da coluna pivotal abaixo do pivot são então divididos pelo valor actual do pivot. Finalmente os elementos do Complemento de Schur $(A|a_{11})$ são calculados usando apenas a coluna pivotal. Tal como anteriormente, apenas interessa determinar os valores dos elementos pertencentes ao triângulo inferior dessa matriz. Assim tem-se

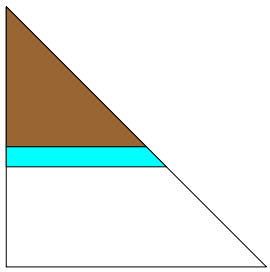
$$A^{(2)} = \begin{bmatrix} 2 & & \\ -1 & 4 & \\ 1 & 1 & 1 \end{bmatrix}$$

Agora $a_{22}^{(2)}$ é o pivot e obtém-se

$$A^{(3)} = \begin{bmatrix} 2 & & \\ -1 & 2 & \\ 1 & \frac{1}{2} & \frac{3}{4} \end{bmatrix}$$

Finalmente a raíz quadrada de $a_{33}^{(3)}$ é calculada e tem-se

$$L = \begin{bmatrix} 2 & & \\ -1 & 2 & \\ 1 & \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}$$



Modificado em iterações anteriores e usado na iteração k

Modificado na iteração k

Não modificado nem usado na iteração k

Algoritmo 16

Faça $L_1 = [1]$ *e* $D_1 = [a_{11}]$

Para $k = 1, 2, \ldots, n-1$

Seja $A_{kk} = L_k D_k L_k^T$ *e* $A_{k+1,k+1}$ *definida por (4.57)*

Seja $D_k = diag(d_{11}, \ldots, d_{kk})$. *Então*

Resolva $L_k \bar{a} = a$

Calcule $\bar{a}_i = \frac{\bar{a}_i}{d_{ii}},\ i = 1, 2, \ldots, k$

$$d_{k+1,k+1} = a_{k+1,k+1} - \sum_{t=1}^{k} d_{tt} \bar{a}_t^2$$

Figura 4.12: Método dos bordos para matrizes SPD

Algoritmo 17

Para $k = 1, 2, \ldots, n-1$

  $a_{kk} = \sqrt{a_{kk}}$

  Para $i = k+1, \ldots, n$

    $a_{ik} = a_{ik}/a_{kk}$

    Para $j = k+1, \ldots, i$

      $a_{ij} = a_{ij} - a_{jk}a_{ik}$

$a_{nn} = \sqrt{a_{nn}}$

Figura 4.13: Decomposição LL^T de uma matriz SPD

É ainda de referir que o número de multiplicações e adições do algoritmo é exactamente o mesmo que as do algoritmo 13. Além disso, neste algoritmo não há necessidade de quaisquer cópias, havendo contudo a necessidade de calcular n raízes quadradas. Realçamos também que os outros algoritmos para a obtenção da decomposição LL^T de A requerem exactamente o mesmo número de multiplicações e adições, o que é uma vantagem em relação à decomposição LDL^T. Contudo a necessidade de calcular n raízes quadradas torna a decomposição LL^T menos recomendável em geral. Além disso a decomposição LDL^T pode ser generalizada para matrizes indefinidas, o que não acontece com a decomposição LL^T. É ainda de notar que a fase de solução tem mais n divisões se a decomposição LL^T é usada em vez da decomposição LDL^T. Essa é mais uma razão para a preferência pela decomposição LDL^T na resolução de sistemas de equações lineares com matrizes SPD.

(v) Matrizes indefinidas

Para introduzir a definição de matriz indefinida iremos primeiramente considerar as matrizes negativas definidas (ND) e negativas semi-definidas (NSD).

Definição 4.1 *Seja A uma matriz quadrada de ordem n. Então*

- $A \in \text{ND} \Leftrightarrow x^T Ax < 0$ *para todo o* $x \in \mathbb{R}^n - \{0\}$.
- $A \in \text{NSD} \Leftrightarrow x^T Ax \leq 0$ *para todo o* $x \in \mathbb{R}^n$.

Com base na definição, é trivial a seguinte propriedade.

Teorema 4.14

- $A \in \text{ND} \Leftrightarrow -A \in \text{PD}$.
- $A \in \text{NSD} \Leftrightarrow -A \in \text{PSD}$.

Por isso estas classes não merecem um interesse especial do ponto de vista das suas propriedades. Contudo, estas matrizes são bastante usadas em algumas áreas da Matemática Aplicada, como por exemplo em Optimização. Como $Ax = b$ se e só se $-Ax = -b$, então a resolução de um sistema de equações lineares com uma matriz ND ou NSD resume-se à obtenção da solução de um sistema com uma matriz PD ou PSD.

Definição 4.2 *Uma matriz A quadrada de ordem n é indefinida* ($A \in$ ID) *se existirem vectores $u \neq 0$ e $v \neq 0$ tais que $v^T Av < 0$ e $u^T Au > 0$.*

Se $A \in$ ID e é simétrica, então escreve-se $A \in$ SID. Tal como para as outras classes, tem-se

$$A \in \text{ID} \Leftrightarrow A + A^T \in \text{SID} \tag{4.62}$$

A título de exemplo, considere-se a matriz

$$A = \begin{bmatrix} 2 & -1 \\ 1 & -1 \end{bmatrix} \tag{4.63}$$

Então

$$A + A^T = \begin{bmatrix} 4 & \\ & -2 \end{bmatrix} \in \text{SID}$$

e portanto a matriz A é indefinida.

Seguidamente, vamos apresentar um processo de resolução do sistema para matrizes SID, que explora a simetria e requer um número de operações sensivelmente igual ao do caso de $A \in$ SPD. Para ilustração do processo, consideremos a seguinte matriz SID não singular

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{4.64}$$

e suponhamos que pretendemos determinar a sua decomposição LDL^T. É evidente que a_{11} não pode ser escolhido como pivot e que a_{21} é o único elemento da primeiro coluna em condições de assumir tal função. Contudo, se o escolhermos como pivot, teremos de trocar a primeira e segunda linhas, o que vai destruir a simetria da matriz A.

Uma estratégia que visa a preservação da simetria é a consideração de pivots que são matrizes 2×2 não singulares. Trata-se de uma generalização do conceito usual de pivot 1×1 segundo a qual se

$$A = A^{(1)} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

então

$$A^{(2)} = \begin{bmatrix} A_{11} & A_{11}^{-1} A_{12} \\ A_{21} A_{11}^{-1} & (A|A_{11}) \end{bmatrix}$$

Como A_{11} é de ordem 2 e a matriz $(A|A_{11})$ é simétrica, então apenas a parte triangular inferior e diagonal de $A^{(2)}$ precisam de ser calculadas. Isso é feito linha a linha resolvendo sistemas de ordem 2 com a matriz A_{11} para a determinação dos elementos de $A_{11}^{-1} A_{12}$ e calculando produtos escalares para a obtenção dos elementos necessários de $(A|A_{11})$. Assim, para a matriz (4.64) tem-se

$$A_{11} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad A_{12} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

e portanto para calcular $\alpha = A_{11}^{-1} A_{12}$ tem de se resolver o sistema

$$A_{11}\alpha = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow \alpha = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Então

$$(A|A_{11}) = [1] - \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = [1]$$

Portanto

$$A^{(2)} = \left[\begin{array}{cc|c} 0 & 1 & 1 \\ 1 & 0 & 0 \\ \hline 1 & 0 & 1 \end{array}\right]$$

e $A = LDL^T$ com

$$L = \left[\begin{array}{cc|c|c} 1 & & & \\ 0 & 1 & & \\ \hline & & & \\ 1 & 0 & & 1 \end{array}\right], \quad D = \left[\begin{array}{cc|c|c} 0 & 1 & & \\ 1 & 0 & & \\ \hline & & & \\ & & & 1 \end{array}\right]$$

O processo de obtenção da decomposição LDL^T de uma matriz simétrica indefinida consiste em usar pivots de ordem 1×1 pertencentes às diagonais das sucessivas matrizes reduzidas $A^{(k)}$ e pivots de ordens 2×2, definidos de acordo com o explicado atrás. Em cada iteração k a escolha de um pivot 1×1 ou 2×2 é feita de acordo com um determinado critério, desenvolvido para manter a estabilidade da decomposição e que foi primeiramente apresentado em [Baunch e Kaufman]. No fim do processo obtém-se a decomposição LDL^T da matriz A, onde L é uma matriz triangular inferior com elementos diagonais iguais a um e D é uma matriz diagonal por blocos, em que cada bloco de ordem 1×1 é constituído por um elemento diferente de zero e cada bloco 2×2 é uma matriz indefinida não singular.

Na figura 4.14 apresentamos o algoritmo 18 da decomposição LDL^T de uma matriz A simétrica indefinida não singular. Nesse algoritmo usamos a notação

$$i \leftrightarrow j$$

para representar a troca da linha (e coluna) i com a linha (e coluna) j. Tal como anteriormente, as matrizes L e D vão ocupar o espaço reservado à matriz A, pelo que o algoritmo consiste em modificar os elementos da matriz A de acordo com as regras definidas pelos pivots de ordens 1×1 ou 2×2.

É de notar que as fórmulas para a obtenção das quantidades a_{ik} e $a_{i,k+1}$ no Passo 3 são consequência do uso da Regra de Cramer para a resolução do sistema

$$\left[\begin{array}{cc} a_{kk} & a_{k+1,k} \\ a_{k+1,k} & a_{k+1,k+1} \end{array}\right] \left[\begin{array}{c} x_0 \\ y_0 \end{array}\right] = \left[\begin{array}{c} a_{ik} \\ a_{i,k+1} \end{array}\right]$$

É fácil de ver que o número de operações (multiplicações e divisões) é da ordem $\frac{n^3}{6} + \mathcal{O}(n^2)$. O algoritmo requer em cada iteração mais comparações que o correspondente algoritmo para a decomposição LDL^T de uma matriz $A \in$ SPD. Além disso, é possível mostrar que o algoritmo é estável, pelo facto de o factor de crescimento g_A ser limitado. A demonstração dessa propriedade é bastante técnica e aparece em [Bunch e Kaufman].

Consideremos agora o sistema $Ax = b$ e suponhamos que A é simétrica indefinida. Então

$$P^T A P = LDL^T$$

com P o produto de t matrizes de permutação P_{ij} $(t \geq 0)$,

$$D = \left[\begin{array}{cccc} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_p \end{array}\right]$$

e $D_i,\ i = 1, \cdots, p$ matrizes de ordem um ou dois $(p \leq n)$. Mas

$$Ax = b \Leftrightarrow P^T A P (P^T x) = P^T b \Leftrightarrow LDL^T (P^T x) = P^T b$$

e portanto a resolução do sistema $Ax = b$ consiste na resolução dos seguintes sistemas

1. $Ly = P^T b$
2. $Dv = y$
3. $L^T (P^T x) = v$

Algoritmo 18

Passo 0: *Faça* $k = 1$ *e seja* α *tal que* $0 < \alpha < 1$.

Passo 1: *Critério de escolha de pivot*. *Seja* $\lambda = \max_{i>k} |a_{ik}| = |a_{rk}|$

Se $|a_{kk}| > \alpha\lambda$ *vá para* **Passo 2**. *Doutro modo, seja*

$$\sigma = \max_{s \geq k, s \neq r} |a_{sr}|$$

1. *Se* $\alpha\lambda^2 \leq |a_{kk}|\sigma$, *vá para* **Passo 2**.
2. *Se* $\alpha\lambda^2 > |a_{kk}|\sigma$ *e* $|a_{rr}| \geq \alpha\sigma$, *faça* $r \leftrightarrow k$ *e vá para* **Passo 2**.
3. *Se 1. e 2. não se verificarem, faça* $r \leftrightarrow k+1$.

Vá para **Passo 3**.

Passo 2: *Pivot de ordem* 1×1.

Para $i = k+1, \ldots, n$

$aux = a_{ik}$

$a_{ik} = \frac{a_{ik}}{a_{kk}}$

Para $j = k+1, \ldots, i$

$a_{ij} = a_{ij} - aux a_{jk}$

Faça $k = k+1$. *Se* $k = n$, *termine. Doutro modo, vá para* **Passo 1**.

Passo 3: *Pivot de ordem* 2×2.

Para $i = k+2, \ldots, n$

$aux1 = a_{ik}$

$aux2 = a_{i,k+1}$

$\alpha_0 = a_{kk}a_{k+1,k+1} - a_{k+1,k}^2$

$a_{ik} = (aux1 a_{k+1,k+1} - aux2 a_{k+1,k})/\alpha_0$

$a_{i,k+1} = (aux2 a_{kk} - aux1 a_{k+1,k})/\alpha_0$

Para $j = k+2, \ldots, i$

$a_{ij} = a_{ij} - aux1 a_{jk} - aux2 a_{j,k+1}$

Faça $k = k+2$. *Se* $k = n$, *termine. Doutro modo, vá para* **Passo 1**.

Figura 4.14: Algoritmo para a obtenção da decomposição LDL^T de uma matriz $A \in$ SID

É de notar que a resolução de $Dv = y$ se resume a p sistemas $D_i v^i = y^i$, cujas matrizes têm ordens iguais a um ou dois.

A título de exemplo, consideremos o sistema $Ax = b$ com

$$A = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} -1 \\ 2 \\ 0 \\ 0 \end{bmatrix}$$

Na fase de factorização há que determinar a decomposição LDL^T de A. Para isso utilizamos o algoritmo 18 com $\alpha = \frac{1}{2}$. Na primeira iteração tem-se

$$a_{11} = 0, \quad \lambda = 1, \quad r = 2$$

Além disso

$$\sigma = \max_{s \neq 2} |a_{sr}| = 1$$

Então verifica-se a condição 2. do algoritmo 18. Portanto as linhas e colunas 1 e 2 são trocadas, ou seja, obtém-se a matriz

$$P_{12} A P_{12} = \begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & 0 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

Além disso efectua-se o passo 2, isto é, usa-se um pivot de ordem 1×1 e vem

$$B^{(2)} = (P_{12} A P_{12})^{(2)} = \begin{bmatrix} -1 & & & \\ -1 & 1 & & \\ -1 & 0 & 0 & \\ -1 & 0 & 2 & 0 \end{bmatrix}$$

Na segunda iteração a condição 1 do Passo 1 é satisfeita. Então $b_{22}^{(2)}$ deve ser escolhido como pivot e obtém-se a mesma matriz que $B^{(2)}$, isto é, $B^{(3)} = B^{(2)}$. Na terceira iteração tem-se

$$b_{33}^{(3)} = 0, \lambda = 2, \sigma = 2, b_{44}^{(3)} = 0$$

Então as condições 1. e 2. do Passo 1 não se verificam e como $k = n-2$ o processo de decomposição termina com

$$L = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ -1 & 0 & 1 & \\ -1 & 0 & 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} -1 & & & \\ & 1 & & \\ & & 0 & 2 \\ & & 2 & 0 \end{bmatrix}$$

Na fase de solução, começa-se por resolver o sistema $Ly = P^T b$. Como $P = P^T = P_{12}$, então $P^T b = [2, -1, 0, 0]^T$ e a solução do sistema $Ly = P^T b$ é $y = [2, 1, 2, 2]^T$. Seguidamente há que resolver $Dv = y$, ou seja

$$\begin{aligned} -v_1 &= 2 \\ v_2 &= 1 \\ \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} v_3 \\ v_4 \end{bmatrix} &= \begin{bmatrix} 2 \\ 2 \end{bmatrix} \end{aligned}$$

Usando a regra de Cramer para o último sistema, obtém-se $v = [-2, 1, 1, 1]$. Finalmente tem de se resolver o sistema $L^T(P_{12}x) = v$, ou seja,

$$\begin{bmatrix} 1 & -1 & -1 & -1 \\ & 1 & 0 & 0 \\ & & 1 & 0 \\ & & & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_1 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Então $x = [1, 1, 1, 1]^T$ é a solução do sistema dado.

4.13 Cálculo do determinante e da inversa de uma matriz

Seja A uma matriz quadrada não singular de ordem n. Nesta secção iremos mostrar como a decomposição LU ou LDL^T de uma matriz pode ser usada para calcular o seu determinante e a sua inversa.

Consideremos primeiramente o caso em que A é uma matriz não simétrica. Então

$$PA = LU$$

com P o produto de k matrizes de permutação P_{ij} $(k \geq 0)$. Devido ao teorema 4.1 tem-se

$$\det(A) = (-1)^k \det(L)\det(U)$$

Mas L e U são matrizes triangulares e todos os elementos diagonais de L são iguais a um. Então $\det(L) = 1$ e

$$\det(A) = (-1)^k \det(U) = (-1)^k u_{11} \dots u_{nn} \tag{4.65}$$

Portanto o determinante de uma matriz pode ser calculado num número de operações de ordem $\frac{n^3}{3}$ em vez de $n!$, como seria necessário se fosse usada a definição. É ainda de notar que o valor k é armazenado na componente n do vector `perm` que é usado na implementação do processo de decomposição LU com escolha parcial de pivot.

Se A é uma matriz simétrica PD, então $A = LDL^T$ e

$$\det(A) = \det(D) = d_{11} \dots d_{nn} \tag{4.66}$$

Finalmente, se A é simétrica indefinida, então

$$P^T AP = LDL^T$$

com P um produto de k matrizes de permutação P_{ij}. Portanto

$$\det(A) = \det(D) = \det(D_1) \dots \det(D_p)$$

onde D_i, $i = 1, \dots, p$ são os blocos diagonais de ordens um ou dois da decomposição. Embora não se tratem de matrizes diagonais, o cálculo dos seus determinantes é muito fácil de efectuar, o mesmo acontecendo portanto ao determinante da matriz A.

Como vimos anteriormente, a matriz inversa A^{-1} de A é por definição a única matriz B cujas colunas $B_{.j}$ são soluções dos sistemas

$$AB_{.j} = e^j, j = 1, \dots, n \tag{4.67}$$

com e^j a coluna j da matriz identidade. O cálculo da matriz inversa explora essas igualdades. Se A é uma matriz não simétrica não singular, então, tal como anteriormente, podemos escrever

$$PA = LU$$

com P o produto de k matrizes de permutação P_{ij} $(k \geq 0)$. Portanto tem-se

$$LUB_{.j} = Pe^j$$

ou ainda

$$Ly^j = Pe^j \tag{4.68}$$

$$UB_{.j} = y^j \tag{4.69}$$

com $j = 1, \dots, n$. O facto de o vector dos termos independentes do sistema (4.68) ser uma coluna da matriz identidade pode ser explorado devidamente de modo a reduzir o número total de operações necessárias para calcular a matriz inversa. Com efeito, tem-se

$$Pe^j = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \hline 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (t+1) \qquad = \qquad \begin{bmatrix} 0 \\ \hline \\ e^{j_2} \end{bmatrix}$$

onde $e^{j_2} \in \mathbb{R}^{n-t}$ e $0 \leq t \leq n-1$. Então podemos escrever a matriz L e o vector y^j na forma

$$L = \begin{bmatrix} L_1 & \\ L_2 & L_3 \end{bmatrix}, \quad y^j = \begin{bmatrix} y^{j_1} \\ y^{j_2} \end{bmatrix}$$

com $L_3 \in \mathbb{R}^{(n-t)\times(n-t)}$ e $y^{j_2} \in \mathbb{R}^{(n-t)}$. Portanto o sistema (4.69) é equivalente a

$$L_1 y^{j_1} = 0 \Rightarrow y^{j_1} = 0$$

$$L_3 y^{j_2} = e^{j_2} - L_2 y^{j_1} = e^{j_2}$$

Assim as dimensões dos sistemas (4.68) variam de 1 a n, consoante a coluna da matriz identidade escolhida. Portanto o número total de operações para a resolução dos n sistemas (4.68) é dado por

$$\sum_{k=1}^{n} \frac{k(k-1)}{2} = \frac{n(n^2-1)}{6}$$

Então o número total de operações para calcular a inversa será a soma deste número, com o que é necessário para calcular a decomposição e as $\dfrac{n(n+1)}{2}n$ operações necessárias para a resolução dos n sistemas (4.69). Assim tem-se

$$\text{op} = \frac{n(n^2-1)}{6} + \frac{n^2(n+1)}{2} + \frac{n^3}{3} - \frac{n}{3} = n^3 + \frac{n(n-1)}{2}$$

Mostrámos assim que o número de operações para calcular a inversa de uma matriz não simétrica de ordem n é aproximadamente n^3. Portanto o uso da decomposição LU para o cálculo da inversa de uma matriz não simétrica requer sensivelmente o mesmo número de operações que é necessário se as operações pivotais forem usadas. Por outro lado, a decomposição LU com escolha parcial de pivot é um processo estável, o que não acontece com as operações pivotais. Daí a decomposição LU com escolha parcial de pivot ser normalmente usada no cálculo da matriz inversa.

Consideremos agora o caso em que a matriz A é simétrica. Então A^{-1} é também simétrica e isso pode ser explorado para reduzir ainda mais o número de operações necessárias para calcular a inversa de A. Com efeito como $B = A^{-1}$ é simétrica, então apenas a diagonal e a parte abaixo da diagonal necessitam de ser calculadas. Tal como na resolução de sistemas de equações lineares, também a determinação da inversa de uma matriz simétrica requer sensivelmente metade das operações necessárias para o caso não simétrico. Para verificar isso, iremos considerar apenas o caso de A ser SPD. Então $A = LDL^T$ e a determinação da matriz inversa é obtida a partir de

$$LDL^T B_{\cdot j} = e^j, \quad j = 1, \ldots, n$$

ou seja

$$\begin{array}{rcll} Ly^j & = & e^j & \\ Dz^j & = & y^j & j = 1, 2, \ldots, n \\ L^T B_{\cdot j} & = & z^j & \end{array}$$

Tal como referimos anteriormente, a resolução de n sistemas $Ly^j = I_{\cdot j}$, $j = 1, \ldots, n$ requer um número de operações igual a

$$\sum_{k=1}^{n} \frac{k(k-1)}{2} = \frac{n(n^2-1)}{6}$$

Consideremos agora o sistema $L^T B_{\cdot j} = z^j$. Então podemos escrever

$$\begin{bmatrix} L_1^T & L_2^T \\ & L_3^T \end{bmatrix} \begin{bmatrix} B_{\cdot j}^1 \\ B_{\cdot j}^2 \end{bmatrix} = \begin{bmatrix} z^{j_1} \\ z^{j_2} \end{bmatrix}$$

com

$$B_{\cdot j}^1 = \begin{bmatrix} b_{1j} \\ \vdots \\ b_{1,j-1} \end{bmatrix} \in \mathbb{R}^{(j-1)}, \quad B_{\cdot j}^2 = \begin{bmatrix} b_{jj} \\ \vdots \\ b_{nj} \end{bmatrix} \in \mathbb{R}^{(n-j+1)}$$

Como estamos apenas interessados em determinar os elementos diagonais e abaixo da diagonal, então basta resolver o sistema

$$L_3^T B_{\cdot j}^2 = z^{j_2}$$

Portanto número total de operações para resolver os n sistemas triangulares superiores é igual a

$$\sum_{k=1}^{n} \frac{k(k-1)}{2} = \frac{n(n^2-1)}{6}$$

Consideremos finalmente o sistema $Dz^j = y^j$. Como D é diagonal e apenas as últimas $(n-j+1)$ componentes de z^j são necessárias, então aqui também vai haver uma redução no número de operações. Com efeito, se escrevermos

$$D = \begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix}, \quad z^j = \begin{bmatrix} z^{1j} \\ z^{2j} \end{bmatrix}, \quad y^j = \begin{bmatrix} y^{1j} \\ y^{2j} \end{bmatrix}$$

com $D_1 \in \mathbb{R}^{(j-1)\times(j-1)}$, $D_2 \in \mathbb{R}^{(n-j+1)\times(n-j+1)}$, $z^{1j}, y^{1j} \in \mathbb{R}^{(j-1)}$ e $z^{2j}, y^{2j} \in \mathbb{R}^{(n-j+1)}$, então basta resolver o sistema

$$D_2 z^{2j} = y^{2j}$$

ou seja, fazer

$$z^{2j} = D_2^{-1} y^{2j}$$

Portanto o número de operações necessárias para resolver os n sistemas diagonais é igual a

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

Tendo em conta estes números de operações obtidos e o número de operações que a decomposição LDL^T requer, concluimos que o número de operações necessárias para calcular a inversa de uma matriz simétrica é igual a

$$\frac{n^3 + 3n^2 - 4n}{6} + 2\frac{n(n^2-1)}{6} + \frac{n(n+1)}{2} = \frac{n^3}{2} + n^2 - \frac{n}{2}$$

Mostrámos assim que a determinação da matriz inversa de uma matriz simétrica requer sensivelmente $\frac{n^3}{2}$ operações.

A título de exemplo, calculemos o determinante e a matriz inversa da matriz

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & 0 \\ -1 & 0 & 3 \end{bmatrix}$$

Como A é simétrica, iremos calcular a sua decomposição LDL^T. Após o primeiro passo da decomposição, obtém-se a matriz

$$A^{(2)} = \begin{bmatrix} 1 & & \\ 1 & 1 & \\ -1 & 1 & 2 \end{bmatrix}$$

Usando agora $a_{22}^{(2)}$ como pivot, obtemos finalmente a decomposição LDL^T de A

$$L = \begin{bmatrix} 1 & & \\ 1 & 1 & \\ -1 & 1 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

Então

$$\det(A) = \det(D) = 1$$

A primeira coluna de A^{-1} é calculada a partir de

$$Ly^1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad z^1 = D^{-1}y^1, \quad L^T B_{.1} = z^1$$

Donde

$$y^1 = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} \Rightarrow z^1 = \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} \Rightarrow B_{.1} = \begin{bmatrix} 6 \\ -3 \\ 2 \end{bmatrix}$$

Em relação à segunda coluna de $B = A^{-1}$, basta calcular os elementos da diagonal e abaixo da diagonal. Portanto tem-se

$$\begin{bmatrix} 1 & \\ 1 & 1 \end{bmatrix} y^2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow y^2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$z^2 = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ & 1 \end{bmatrix} \begin{bmatrix} b_{22} \\ b_{32} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \Rightarrow \begin{cases} b_{32} = -1 \\ b_{22} = 2 \end{cases}$$

Finalmente é necessário calcular b_{33} e vem

$$y^3 = 1, \quad z^3 = 1, \quad b_{33} = z^3 = 1$$

Donde

$$A^{-1} = B = \begin{bmatrix} 6 & -3 & 2 \\ -3 & 2 & -1 \\ 2 & -1 & 1 \end{bmatrix}$$

Exercícios

1. Calcule a decomposição LU das matrizes

$$A_1 = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 0 & 1 \\ 1 & -1 & 2 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 1 & 2 & -1 & 2 \\ 1 & -1 & 1 & 1 \\ -2 & 1 & 3 & -1 \\ 1 & 1 & -1 & 4 \end{bmatrix}.$$

2. Calcule a decomposição LU das matrizes

$$A_1 = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 3 & 1 \\ 0 & 2 & 2 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 2 & 1 \\ -1 & 1 & 2 \\ 1 & -3 & 1 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 1 & 1 & 2 & 2 \\ 1 & -1 & 1 & 1 \\ -2 & 1 & 3 & -1 \\ 1 & 1 & -1 & 4 \end{bmatrix}$$

   usando escolha parcial de pivot.

3. Resolva os seguintes sistemas:

   (a)
$$\begin{cases} -x_1 + 2x_2 + x_3 = 2 \\ x_1 - 3x_2 + x_3 = -1 \\ x_2 + 2x_3 = 3 \end{cases}$$

   (b)
$$\begin{cases} x_1 - x_2 + x_3 = 1 \\ -x_2 + x_3 = 2 \\ x_3 = -1 \end{cases}$$

   (c)
$$\begin{cases} 2x_1 - 2x_2 + x_3 = 5 \\ -x_1 + 3x_2 + x_3 = -3 \\ -x_1 + x_2 + 2x_3 = 0 \end{cases}$$

   (d)
$$\begin{cases} 3x_1 - x_2 - x_3 = -1 \\ 2x_1 + 4x_2 + x_3 = -7 \\ -x_1 + x_2 + 3x_3 = -3 \end{cases}$$

4. Calcule o número de operações necessárias para a determinação da decomposição LU de uma matriz usando os métodos directos e dos bordos.

5. Determine a decomposição LU das matrizes A_1, A_2 e A_3 do exercício 1 usando os métodos directo e dos bordos.

6. Uma matriz diz-se de Hessenberg Superior se

$$a_{ij} = 0 \text{ para } i > j + 1$$

   Indique o número de operações necessárias para determinar a decomposição LU de uma matriz de Hessenberg Superior de ordem n.

7. Mostre que a inversa de uma matriz triangular é triangular e indique o número de operações necessárias para a calcular.

8. Considere as seguintes matrizes

$$A_1 = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 2 & 3 & 2 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 1 & 1 & -1 \\ 2 & 1 & 1 \\ 1 & 3 & 4 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Determine o número de condição na norma $||\cdot||_\infty$ das matrizes A_i, $i = 1, 2, 3, 4$ e compare-o com a estimativa que se obtém usando o estimador LINPACK.

9. Determine a decomposição LDL^T das seguintes matrizes:

$$A_1 = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & 1 \\ -1 & 1 & 3 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 3 & -1 \\ -1 & -1 & 4 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & -1 \\ 1 & -1 & 5 \end{bmatrix}.$$

10. Calcule o número de operações necessárias para a determinação da decomposição LDL^T de uma matriz SPD usando os métodos directo e dos bordos.

11. Calcule as decomposições LDL^T das matrizes A_1, A_2 e A_3 do exercício 9 usando os métodos directo e dos bordos.

12. Determine a decomposição LL^T das matrizes:

$$A_1 = \begin{bmatrix} 4 & 1 & 2 \\ 1 & 5 & 1 \\ 2 & 1 & 8 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 8 & 0 \\ -1 & 0 & 8 \end{bmatrix}.$$

13. Desenvolva os algoritmos directo e dos bordos para a decomposição LL^T de uma matriz SPD.

14. Mostre que $A \in$ SPD se e só se $A = LL^T$ com L uma matriz triangular inferior.

15. Verifique se as seguintes matrizes são SPD, SPSD, PD ou PSD:

$$A_1 = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 2 & 1 \\ 1 & 5 \end{bmatrix}, \quad A_5 = \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix}, \quad A_6 = \begin{bmatrix} 2 & 4 \\ 1 & 1 \end{bmatrix}$$

$$A_7 = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 4 & 8 \\ 2 & 8 & 4 \end{bmatrix}, \quad A_8 = \begin{bmatrix} 3 & 1 & -1 \\ 1 & 2 & -1 \\ -1 & -1 & 3 \end{bmatrix}, \quad A_9 = \begin{bmatrix} 2 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 4 \end{bmatrix}, \quad A_{10} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 3 & 1 \\ 2 & 1 & 4 \end{bmatrix}$$

$$A_{11} = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 6 & 9 \\ 2 & 4 & 8 \end{bmatrix}, \quad A_{12} = \begin{bmatrix} 2 & 5 & 8 \\ 4 & 3 & 2 \\ 7 & 6 & 1 \end{bmatrix}, \quad A_{13} = \begin{bmatrix} 4 & 1 & -2 \\ -1 & 5 & 3 \\ 0 & 2 & 4 \end{bmatrix}$$

16. Determine os conjuntos dos números reais x tais que

(a)
$$\begin{bmatrix} 1 & -1 & -2 \\ -1 & 2 & -1 \\ -1 & -1 & x+4 \end{bmatrix} \in \mathrm{K}$$

(b)
$$\begin{bmatrix} 1 & -1 & x \\ 1 & 2 & 1 \\ 1 & -x & x^2 \end{bmatrix} \in \mathrm{PD}$$

(c)

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 2 & 1 \\ -1 & x^2 & x+6 \end{bmatrix} \in \mathrm{H}_+$$

(d)

$$\begin{bmatrix} 1 & 0 & -1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & x & 1 & x \\ -1 & 1 & 3 & 3 \end{bmatrix} \in \mathrm{H}(x > 0)$$

(e)

$$\begin{bmatrix} \frac{1}{2} & 1-x & 1+x & 1 \\ x & \frac{1}{2} & 0 & 2 \\ -x & 1 & \frac{3}{2} & x-1 \\ 1 & 0 & 1-x & x \end{bmatrix} \in \mathrm{PSD}$$

.

17. Demonstre o teorema 4.12.

18. Mostre que se A é uma matriz simétrica então

$$A \in \mathrm{PSD} \Leftrightarrow A \in \mathrm{P}_0.$$

19. Mostre que $A \in \mathrm{ND}$ se e só se $A = LDL^T$ e $d_{ii} < 0$ para todo o i.

20. Resolva os seguintes sistemas de equações lineares usando o algoritmo para matrizes simétricas indefinidas:

$$\begin{cases} x_1 - x_2 + x_3 = 1 \\ -x_1 - 2x_2 - 3x_3 = -6 \\ x_1 - 3x_2 = -2 \end{cases} \qquad \begin{cases} x_1 + x_2 - x_3 + 2x_4 = 5 \\ x_1 - 2x_2 - x_4 = -2 \\ -x_1 - x_4 = -2 \\ 2x_1 - x_2 - x_3 = 2 \end{cases}$$

21. Calcule os determinantes e inversas das seguintes matrizes

$$A_1 = \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & 2 \\ 1 & 0 & -1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -1 & 2 & 4 \\ -1 & 1 & -1 \\ 2 & 0 & 2 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 3 \end{bmatrix}, \quad A_4 = \begin{bmatrix} -1 & 1 & -1 \\ 1 & -3 & 1 \\ -1 & 1 & -5 \end{bmatrix}$$

22. Dada a função $f : x \mapsto x + \cos \pi x$, determine o polinómio $P(x)$ de grau 2 tal que $P(i) = f(i)$, $i = 1, 2, 3$.

Capítulo 5

Métodos Directos para Sistemas de Equações Lineares com Estrutura Especial

Neste capítulo iremos estudar sistemas de equações lineares cujas matrizes têm estruturas especiais que permitem explorar o número de elementos nulos de modo a reduzir o número de operações e o espaço de armazenagem. Trata-se do primeiro exemplo de resolução de sistemas de equações com matrizes esparsas. No entanto, não abordaremos este caso no capítulo referente à resolução de sistemas com matrizes esparsas, em virtude de se tratar de casos especiais, que se aproximam mais da resolução de sistemas de equações lineares com matrizes densas. Iremos ver que em muitos casos a redução do número de operações é considerável, sendo inclusivamente possível resolver alguns sistemas de grandes dimensões com um número de operações relativamente pequeno e um espaço de armazenagem reduzido. Consideraremos primeiramente o caso da estrutura de banda, para depois estudarmos os sistemas com estrutura de blocos.

5.1 Matrizes com estrutura de banda

(i) Matrizes simétricas

A definição de Matriz Banda está relacionada com o facto de os elementos não nulos não diagonais da matriz estarem dispostos ao longo de linhas paralelas à diagonal principal. Como para matrizes simétricas o número de linhas paralelas acima da diagonal é igual ao número de linhas paralelas abaixo da diagonal, apenas consideraremos este último número. Antes de apresentarmos a definição rigorosa de Banda de uma matriz, consideremos a matriz de ordem 4

$$A = \begin{bmatrix} a_{11} & a_{21} & & \\ a_{21} & a_{22} & & a_{42} \\ & & a_{33} & a_{43} \\ & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{5.1}$$

Os elementos não diagonais não nulos dispõem-se em duas linhas paralelas à diagonal principal e dizemos por isso que o Comprimento de Banda de A é igual a 2. A Banda da matriz será constituída pelos elementos diagonais e pelos elementos dessas duas linhas paralelas.

Para a introdução das noções de Comprimento de Banda e de Banda de uma matriz simétrica $A \in \mathbb{R}^{n\times n}$, consideremos as quantidades $f_i(A)$ e $\beta_i(A)$ definidas por

$$\begin{array}{rcl} f_i(A) & = & \min\{j : j \leq i,\ a_{ij} \neq 0\} \\ \beta_i(A) & = & i - f_i(A) \end{array} ,\ i = 1, \ldots, n \tag{5.2}$$

O Comprimento de Banda $\beta(A)$ da matriz A é definido por

$$\beta(A) = \max\{\beta_i(A) :\ 1 \leq i \leq n\} = \max\{|i-j| :\ a_{ij} \neq 0\} \tag{5.3}$$

O número $\beta_i(A)$ chama-se i-ésimo comprimento de banda de A. A Banda da matriz A é o conjunto dos elementos que vão da diagonal até à $\beta(A)$-ésima linha paralela à diagonal inclusivé, ou seja,

$$\text{BAND}(A) = \{\{i,j\} :\ 0 \leq i-j \leq \beta(A)\} \tag{5.4}$$

Assim, para a matriz (5.1), tem-se

$$\left.\begin{array}{l} f_1(A)=1 \\ f_2(A)=1 \\ f_3(A)=3 \\ f_4(A)=2 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} \beta_1(A)=0 \\ \beta_2(A)=1 \\ \beta_3(A)=0 \\ \beta_4(A)=2 \end{array}\right. \Rightarrow \beta(A)=2$$

Das definições apresentadas é fácil derivar o número de elementos $|\text{BAND}(A)|$ da banda de uma matriz $A \in \mathbb{R}^{n\times n}$ de comprimento de banda $\beta(A)$. Com efeito tem-se

$$\begin{aligned} |\text{BAND}(A)| &= n + (n-1) + \ldots + [n-\beta(A)] \\ &= [\beta(A)+1]\, n - [1 + \ldots + \beta(A)] \\ &= [\beta(A)+1]\, n - \frac{\beta(A)\,[\beta(A)+1]}{2} \end{aligned}$$

ou seja

$$|\text{BAND}(A)| = [\beta(A)+1]\left[n - \frac{\beta(A)}{2}\right] \tag{5.5}$$

Uma matriz diz-se Tridiagonal se $\beta(A) = 1$. O número de elementos da banda de uma matriz tridiagonal é assim igual a $2n-1$.

A grande vantagem da consideração da banda de uma matriz está no facto de a banda não ser alterada no decurso da obtenção da sua decomposição LDL^T, se os pivots forem sempre escolhidos na diagonal. Como tal acontece com as matrizes SPD, então podemos enunciar o seguinte resultado.

Teorema 5.1 *Se* $A \in$ SPD*, então* $\text{BAND}(A) = \text{BAND}(L+D+L^T)$.

Demonstração: Seja $A^{(2)}$ a matriz obtida após a primeira iteração do processo de decomposição LDL^T por eliminação de Gauss. Então

$$A^{(2)} = \begin{bmatrix} a_{11} & \frac{a_{21}}{a_{11}} & \ldots & \frac{a_{n1}}{a_{11}} \\ \frac{a_{21}}{a_{11}} & \bar{a}_{22} & \ldots & \bar{a}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}}{a_{11}} & \bar{a}_{n2} & \ldots & \bar{a}_{nn} \end{bmatrix} \tag{5.6}$$

com

$$\bar{a}_{ij} = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}, \quad i,j \geq 2$$

Para mostrar que $\text{BAND}(A) = \text{BAND}(A^{(2)})$ seja $p = \beta(A)$. Então por definição tem-se

$$p = \beta(A) \Leftrightarrow a_{ij} = 0 \text{ para } i > j+p \tag{5.7}$$

e portanto teremos que estabelecer que

$$a_{ij}^{(2)} = 0 \text{ para } i > j+p \tag{5.8}$$

Dois casos podem acontecer e são apresentados de seguida.

Algoritmo 19

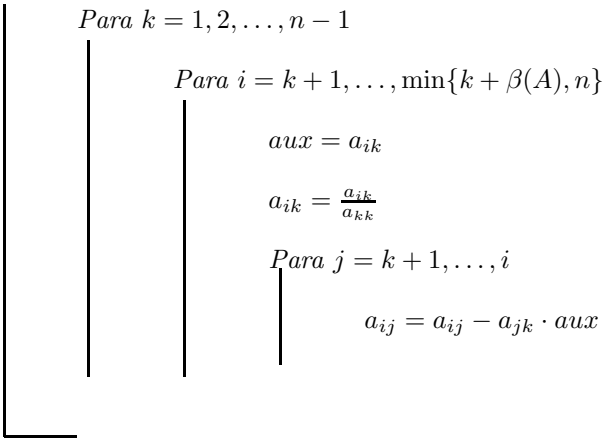


Figura 5.1: Decomposição LDL^T para uma matriz SPD com estrutura de banda

1. Se $j = 1$, então por (5.6) tem-se $a_{ij}^{(2)} = \frac{a_{ij}}{a_{jj}}$ e de (5.7) vem $a_{ij}^{(2)} = 0$ para $i > j + p$.

2. Se $j > 1$, então de (5.6) tem-se $a_{ij}^{(2)} = a_{ij} - a_{i1}^{(2)} a_{1j}$. Mas para $i > j + p$ tem-se $i > j + 1$ e portanto $a_{i1}^{(2)} = 0$. Donde (5.8) é verdadeira.

Provámos assim que $\text{BAND}(A) = \text{BAND}(A^{(2)})$. O teorema fica demonstrado por indução.

Por este teorema, a determinação das matrizes L e D da decomposição LDL^T de uma matriz $A \in$ SPD necessita apenas de usar os elementos da banda da matriz. Essa propriedade pode ser extremamente vantajosa se o comprimento de banda da matriz for bastante pequeno. Com efeito, não só o espaço de armazenagem é reduzido, como o número de operações desce substancialmente. O algoritmo 13 para a obtenção da decomposição LDL^T de uma matriz $A \in$ SPD é modificado minimamente de modo a explorar a banda da matriz e tem a forma expressa na figura 5.1.

Para calcular o número de operações, dois casos podem acontecer e são apresentados seguidamente.

1. Se $k + \beta(A) \leq n$, cada iteração k tem

$$\beta(A) \text{ divisões}$$

$$1 + 2 + \ldots + \beta(A) = \frac{[\beta(A)+1]\,\beta(A)}{2} \text{ multiplicações e adições}$$

2. Se $k + \beta(A) > n$, então em cada iteração k tem-se

$$n - k \text{ divisões}$$

$$(n-k) + \ldots + 1 = \frac{(n-k)(n-k+1)}{2} \text{ multiplicações e adições}$$

Então o número total de multiplicações e adições é dado por

$$\begin{aligned} & \sum_{k=1}^{n-\beta(A)} \frac{[\beta(A)+1]\,\beta(A)}{2} + \sum_{k=n-\beta(A)+1}^{n-1} \frac{(n-k)(n-k+1)}{2} \\ = \;& [n-\beta(A)]\,\frac{[\beta(A)+1]\,\beta(A)}{2} + \frac{1}{2} \sum_{k=1}^{\beta(A)-1} [\beta(A)-k]\,[\beta(A)-k+1] \\ = \;& \frac{\beta(A)\,[\beta(A)+1]}{2} n - \frac{\beta(A)\,[\beta(A)+1]\,[2\beta(A)+1]}{6} \end{aligned}$$

Algoritmo 20

Para $k = 2, \ldots, n$

Resolução de $Ly = b$

$$b_k = b_k - \sum_{j=\max\{k-\beta(A),1\}}^{k-1} l_{kj} b_j$$

Para $k = n-1, \ldots, 1$

Resolução de $L^T x = y$

$$b_k = b_k - \sum_{j=k+1}^{\min\{k+\beta(A),n\}} l_{jk} b_j$$

Figura 5.2: Resolução de sistemas triangulares com matrizes de estrutura de banda

e o número total de divisões é

$$\sum_{k=1}^{n-\beta(A)} \beta(A) + \sum_{k=n-\beta(A)+1}^{n-1} (n-k) = n\beta(A) - \frac{[\beta(A)+1]\,\beta(A)}{2}$$

Portanto o número total de operações é dado por

$$\begin{aligned} \text{Adições} &= \frac{[\beta(A)+1]\,\beta(A)}{2} n - \frac{\beta(A)\,[\beta(A)+1]\,[2\beta(A)+1]}{6} \\ \text{Multiplicações / divisões} &= \frac{[\beta(A)+3]\,\beta(A)}{2} n - \frac{\beta(A)\,[\beta(A)+1]\,[\beta(A)+2]}{3} \end{aligned} \tag{5.9}$$

Então o número de operações é da ordem $\frac{[\beta(A)+3]\beta(A)}{2} n$ e facilmente se conclui da enorme redução no número de operações quando o comprimento de banda é suficientemente pequeno. Assim, por exemplo, se $n = 100$ e $\beta(A) = 10$, então a obtenção da decomposição LDL^T sem explorar a banda requer cerca de $\frac{10^6}{6}$ operações, mas apenas 6060 operações são necessárias no caso de a banda ser devidamente aproveitada.

Como a matriz L tem comprimento de banda $\beta(A)$, a resolução dos sistemas triangulares $Ly = b$ e $L^T x = y$ pode ser feita de modo a explorar esse facto e assim reduzir também o número de operações. O processo para a resolução desses dois sistemas é apresentado na figura 5.2.

Portanto cada um desses processos requer

$$\sum_{k=1}^{n-\beta(A)} \beta(A) + \sum_{k=n-\beta(A)+1}^{n-1} (n-k) = n\beta(A) - \frac{[\beta(A)+1]\,\beta(A)}{2} \tag{5.10}$$

multiplicações e adições. Se agora considerarmos que a resolução de um sistema $Ax = b$ necessita da obtenção da decomposição LDL^T de A, da resolução dos dois sistemas triangulares referidos e de n divisões referentes ao cálculo de $D^{-1}y$, então facilmente se conclui que o número de operações necessárias para a resolução do sistema $Ax = b$ com $A \in$ SPD de ordem n e comprimento de banda

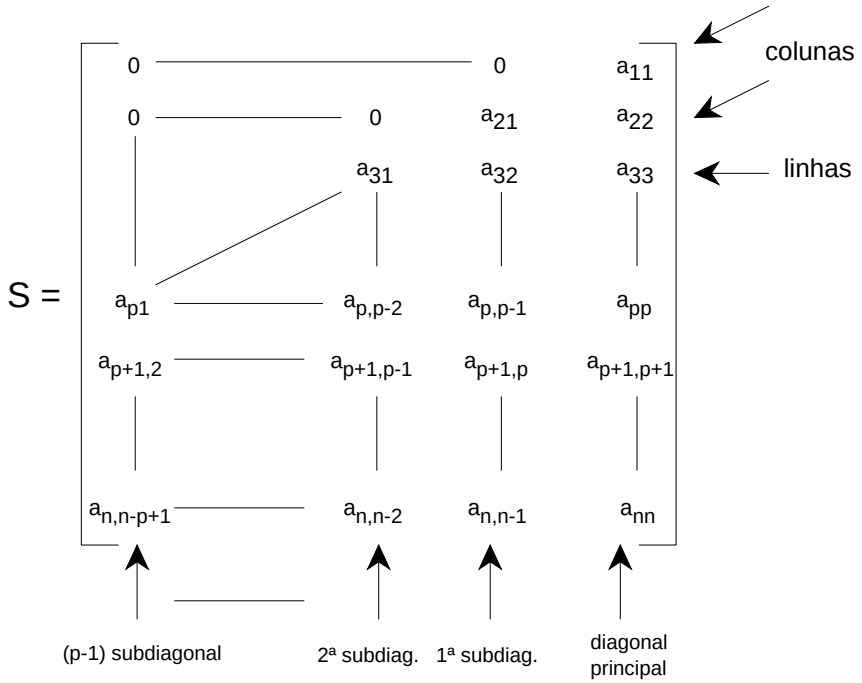


Figura 5.3: Armazenagem de uma matriz SPD com estrutura de banda

$\beta(A)$, é dado por

$$\begin{cases} \text{Adições} & = & \dfrac{[\beta(A)+1]\,\beta(A)}{2}n - \dfrac{\beta(A)\,[\beta(A)+1]\,[2\beta(A)+7]}{6} \\ \text{Multiplicações/divisões} & = & \dfrac{\beta(A)^2+7\beta(A)+4}{2}n - \dfrac{\beta(A)\,[\beta(A)+1]\,[\beta(A)+5]}{3} \end{cases} \tag{5.11}$$

Como apenas os elementos da banda da matriz A são transformados durante todo o processo de resolução do sistema $Ax = b$, então há apenas a necessidade de armazenar esses elementos, o que pode ser feito usando uma matriz rectangular S de ordem $n \times [\beta(A)+1]$ cuja estrutura é apresentada na figura 5.3, onde $p = \beta(A)+1$ e $a_{ik} = s_{i,k-i+p}$. Notemos que nesta representação são armazenados

$$(p-1) + \ldots + 1 = \frac{p(p-1)}{2} = \frac{\beta(A)\,[\beta(A)+1]}{2}$$

zeros que nunca são transformados durante todo o processo de resolução do sistema $Ax = b$. É fácil de escrever os algoritmos 19 e 20 usando este tipo de representação. Contudo não o iremos fazer deixando a escrita desses processos como exercício.

Um caso especial de matrizes banda que aparecem frequentemente em aplicações são as chamadas matrizes Tridiagonais, que têm comprimento de banda igual a 1. Essas matrizes têm a forma

$$A = \begin{bmatrix} d_1 & e_1 & & & & \\ e_1 & d_2 & e_2 & & & \\ & e_2 & d_3 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & e_{n-1} \\ & & & & e_{n-1} & d_n \end{bmatrix}$$

Algoritmo 21

Determinação de L e D

Para $k = 1, 2, \ldots, n-1$

- $aux = e_k$
- $e_k = \frac{e_k}{d_k}$
- $d_{k+1} = d_{k+1} - e_k \cdot aux$

Resolução de $Ly = b$

Para $k = 1, 2, \ldots, n-1$

- $b_{k+1} = b_{k+1} - b_k e_k$

Cálculo de $y = D^{-1}y$

Para $k = 1, 2, \ldots, n$

- $b_k = \frac{b_k}{d_k}$

Resolução de $L^T x = y$

Para $k = n-1, \ldots, 1$

- $b_k = b_k - e_k b_{k+1}$

Figura 5.4: Decomposição LDL^T para matrizes SPD tridiagonais

pelo que são armazenadas recorrendo apenas aos vectores $d \in \mathbb{R}^n$ e $e \in \mathbb{R}^{n-1}$. Neste caso, os algoritmos 19 e 20 são muitíssimo mais simples, com reflexos na eficiência da resolução do sistema $Ax = b$. A figura 5.4 apresenta a forma do processo de decomposição e solução. Como facilmente se verifica, o número de operações para a resolução do sistema $Ax = b$ quando A é tridiagonal é dado por

$$\begin{cases} \text{Adições} & = & 3n-3 \\ \text{Multiplicações/divisões} & = & 5n-4 \end{cases} \tag{5.12}$$

Assim, a resolução de sistemas com matrizes tridiagonais SPD é um processo extremamente simples de implementar, obtendo-se a solução de um modo muito rápido.

(ii) Matrizes não simétricas

Se A é uma matriz não simétrica, então é evidente que o número de linhas paralelas à diagonal que se situam acima dela pode ser diferente do número de linhas paralelas à diagonal que se situam abaixo da diagonal. Há portanto a necessidade de mudar as definições de banda e comprimento de banda de uma matriz. Seja A uma matriz de ordem n e consideremos as quantidades

$$\begin{array}{rcl} f_i^r(A) & = & \min\{j:\ j \le i, a_{ij} \neq 0\} \\ \beta_i(A) & = & i - f_i^r(A) \end{array} \quad , \quad i = 1, \ldots, n$$

$$\begin{array}{rcl} f_j^c(A) & = & \min\{i:\ i \le j, a_{ij} \neq 0\} \\ \alpha_j(A) & = & j - f_j^c(A) \end{array} \quad , \quad j = 1, \ldots, n \tag{5.13}$$

O Comprimento de Banda Inferior $\beta(A)$ da matriz A é definido por

$$\beta(A) = \max\{\beta_i(A) : 1 \leq i \leq n\} \tag{5.14}$$

O Comprimento de Banda Superior é dado por

$$\alpha(A) = \max\{\alpha_j(A) : 1 \leq j \leq n\} \tag{5.15}$$

Os números $\alpha_j(A)$ e $\beta_i(A)$ chamam-se respectivamente o j-ésimo comprimento de banda superior e i-ésimo comprimento de banda inferior. A Banda de A é o conjunto de elementos entre a $\beta(A)$-ésima e a $\alpha(A)$-ésima linhas paralelas à diagonal inclusivé, isto é

$$\text{BAND}(A) = \{(i,j) : -\alpha(A) \leq i - j \leq \beta(A)\} \tag{5.16}$$

De notar que esta definição implica que

$$p = \beta(A) \wedge q = \alpha(A) \Leftrightarrow a_{ij} = 0 \text{ para } i > j + p \text{ ou } i < j - q \tag{5.17}$$

Para ilustrar estas definições consideremos a seguinte matriz de ordem 5:

$$A = \begin{bmatrix} 1 & -1 & & & \\ & 2 & & 1 & \\ & -1 & 3 & & \\ & & 2 & 4 & -1 \\ & & & & 5 \end{bmatrix}$$

Então

$$\left.\begin{array}{l} \beta_1(A) = 0 \\ \beta_2(A) = 0 \\ \beta_3(A) = 1 \\ \beta_4(A) = 1 \\ \beta_5(A) = 0 \end{array}\right\} \Rightarrow \beta(A) = 1 \qquad \left.\begin{array}{l} \alpha_1(A) = 0 \\ \alpha_2(A) = 1 \\ \alpha_3(A) = 0 \\ \alpha_4(A) = 2 \\ \alpha_5(A) = 1 \end{array}\right\} \Rightarrow \alpha(A) = 2$$

e

$$\text{BAND}(A) = \begin{array}{l} \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(2,4),(3,2) \\ (3,3),(3,4),(3,5),(4,3),(4,4),(4,5),(5,5)\} \end{array}$$

Se $|\text{BAND}(A)|$ representa o número de elementos da Banda de A, então tem-se

$$\begin{aligned} |\text{BAND}(A)| &= n + (n-1) + \ldots + [n - \beta(A)] + (n-1) + \ldots + [n - \alpha(A)] \\ &= [\beta(A) + \alpha(A) + 1]\, n - [1 + \ldots + \beta(A)] - [1 + \ldots + \alpha(A)] \end{aligned}$$

ou seja

$$\text{BAND}(A) = [\beta(A) + \alpha(A) + 1]\, n - \frac{1}{2}\{\beta(A)\,[\beta(A)+1] - \alpha(A)\,[\alpha(A)+1]\} \tag{5.18}$$

Uma matriz diz-se Tridiagonal se $\beta(A) = \alpha(A) = 1$. O número de elementos da banda é neste caso igual a $3n - 2$.

Tal como para as matrizes SPD, a vantagem da consideração da banda da matriz A está no facto de esta não ser alterada durante o processo de obtenção da decomposição LU da matriz A, desde que os pivots sejam escolhidos na diagonal principal das sucessivas matrizes transformadas $A^{(k)}$. Como é sabido, tal é verificado se a matriz A é das classes H, K, ou diagonalmente dominantes. Portanto podemos enunciar o seguinte teorema

Teorema 5.2 *Se A admite decomposição LU sem necessidade de troca de linhas e colunas (pivots diagonais estáveis), então* $\text{BAND}(A) = \text{BAND}(L + U)$.

Algoritmo 22

Para $k = 1, 2, \ldots, n-1$

Determinação da decomposição LU

- *Para* $i = k+1, \ldots, \min\{k+\beta(A), n\}$
  - $a_{ik} = \frac{a_{ik}}{a_{kk}}$
  - *Para* $j = k+1, \ldots, \min\{k+\alpha(A), n\}$
    - $a_{ij} = a_{ij} - a_{ik}a_{kj}$

Resolução de $Ly = b$

Para $k = 2, \ldots, n$

$$b_k = b_k - \sum_{j=\max\{k-\beta(A),1\}}^{k-1} l_{kj} b_j$$

Resolução de $Ux = y$

Para $k = n, \ldots, 1$

$$b_k = \left(b_k - \sum_{j=k+1}^{\min\{k+\alpha(A),n\}} u_{kj} b_j \right) / u_{kk}$$

Figura 5.5: Resolução de um sistema de equações com uma matriz A não simétrica com pivots diagonais estáveis e estrutura de banda

A demonstração deste teorema é bastante semelhante à usada para estabelecer o teorema 5.1 e será deixada como exercício.

Por este teorema, todo o processo de resolução de um sistema $Ax = b$, com A não singular e admitindo decomposição LU, pode ser feito transformando apenas os elementos da banda. Assim os algoritmos 3, 1 e 2 são modificados minimamente e têm a forma que apresentamos na figura 5.5.

É também fácil de determinar o número de operações necessárias para a resolução do sistema $Ax = b$ com A uma matriz que admita pivots diagonais estáveis de ordem n e de comprimentos de banda $\beta(A)$ e $\alpha(A)$. A grandeza desses números depende dos tamanhos das componentes de banda inferior e superior. A determinação desses números é deixada como exercício.

Tal como para as matrizes SPD com estrutura de banda, também as matrizes não simétricas de estrutura de banda se armazenam usando uma matriz rectangular S. A ordem dessa matriz é $n \times [\beta(A) + \alpha(A) + 1]$ e a sua forma é apresentada na figura 5.6, onde $p = \beta(A)+1$ e $q = \alpha(A)+1$.

De notar que nesta representação tem-se

$$a_{ik} = s_{i,k-i+p} \tag{5.19}$$

e são armazenados

$$\begin{aligned} (p-1) + \ldots + 1 + 1 + \ldots + (q-1) &= \frac{1}{2}\left[p(p-1) + q(q-1)\right] \\ &= \frac{1}{2}\left\{\alpha(A)\left[\alpha(A)+1\right] + \beta(A)\left[\beta(A)+1\right]\right\} \end{aligned}$$

zeros que nunca são modificados durante o processo de resolução do sistema $Ax = b$.

Tal como para as matrizes tridiagonais SPD, também as matrizes tridiagonais não simétricas com pivots diagonais estáveis aparecem frequentemente em aplicações práticas. Essas matrizes

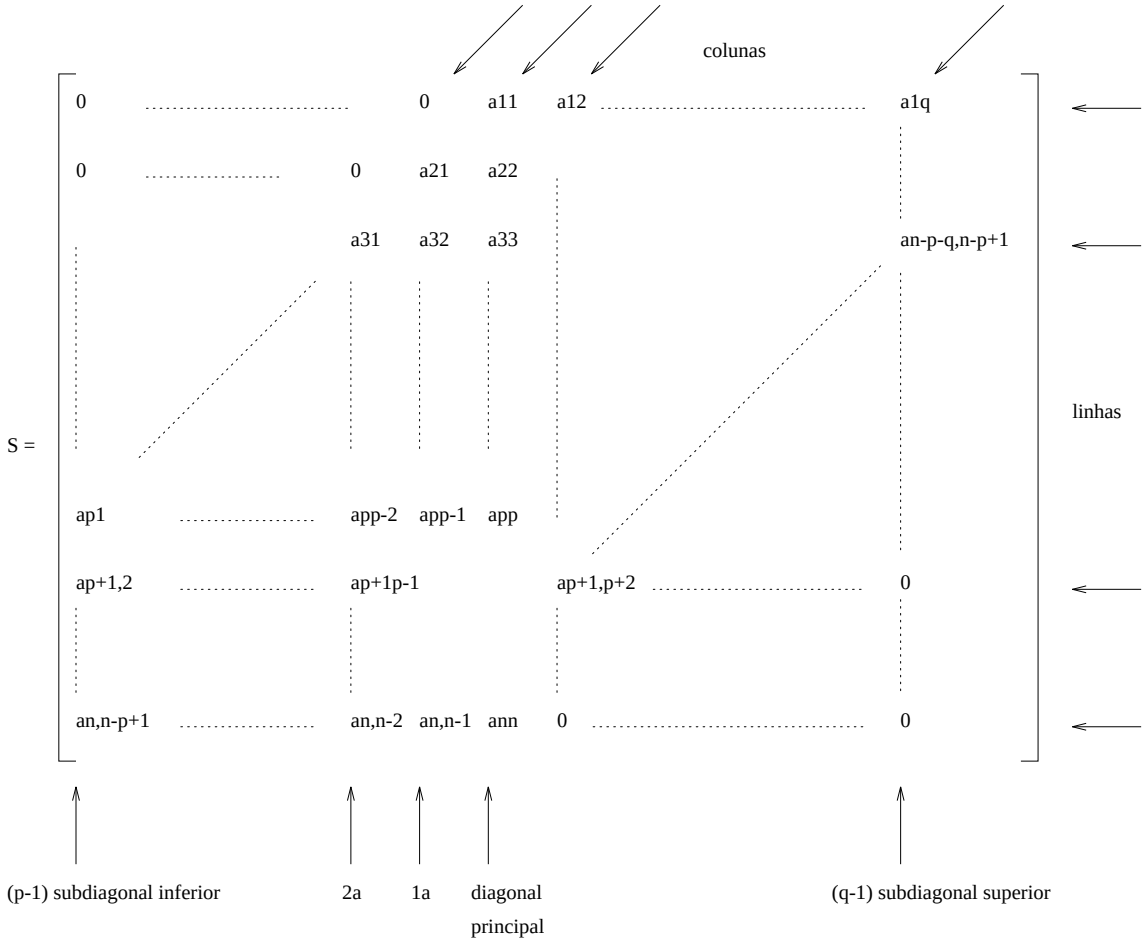


Figura 5.6: Armazenagem de uma matriz não simétrica com estrutura de banda

Algoritmo 23

Determinação de L e U

Para $k = 1, 2, \ldots, n-1$

$e_k = \frac{e_k}{d_k}$

$d_{k+1} = d_k - e_k f_k$

Resolução de $Ly = b$

Para $k = 1, 2, \ldots, n-1$

$b_{k+1} = b_{k+1} - b_k e_k$

Resolução de $Ux = y$

Para $k = n, \ldots, 1$

$b_k = \frac{b_k - f_k b_{k+1}}{d_k}$

Figura 5.7: Resolução de um sistema de equações com uma matriz não simétrica tridiagonal com pivots diagonais estáveis

têm a forma

$$A = \begin{bmatrix} d_1 & f_1 & & & & \\ e_1 & d_2 & f_2 & & & \\ & e_2 & d_3 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & f_{n-1} \\ & & & & e_{n-1} & d_n \end{bmatrix}$$

e são por isso armazenadas considerando apenas as $3n-2$ quantidades e_i, d_i e f_i. O algoritmo para a resolução de um sistema $Ax = b$ é extremamente simplificado neste caso, tendo a forma apresentada na figura 5.7. O número de operações para resolver o sistema $Ax = b$ quando A é tridiagonal é dado por

$$\begin{cases} \text{Adições} & = & 3n-3 \\ \text{Multiplicações/divisões} & = & 5n-4 \end{cases} \tag{5.20}$$

5.2 Matrizes com estrutura de blocos

Nesta secção iremos estudar a resolução de sistemas de equações lineares $Ax = b$ em que a matriz A tem uma estrutura de blocos. Iremos assumir que as matrizes em cada bloco são matrizes densas, mas nada impede que estes esquemas não possam ser usados com matrizes esparsas dentro de cada bloco. Estas técnicas são particularmente recomendadas quando se usa armazenagem auxiliar, obtendo-se processos em que poucos blocos são trazidos para a memória principal de cada vez, permitindo assim a resolução de sistemas de grandes dimensões. Iremos considerar as

quatro formas por blocos mais consideradas nas aplicações, nomeadamente as formas Diagonal, Triangular, Diagonal com Bordo e Tridiagonal por Blocos.

(i) Forma Diagonal por Blocos

Neste caso a matriz A tem a forma

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_m \end{bmatrix} \tag{5.21}$$

com A_k matrizes não singulares de ordem n_k e $\sum_{k=1}^{m} n_k = n$. Se A tem essa forma, então a resolução do sistema $Ax = b$ resume-se à resolução de m sistemas

$$A_k x^k = b^k,\ k = 1, \ldots, m \tag{5.22}$$

Como afirmámos anteriormente, se as matrizes A_k forem armazenadas em memória auxiliar, então a resolução do sistema $Ax = b$ pode ser feita com m "chamadas" das matrizes A_k e correspondente resolução dos sistemas (5.22). Deste modo é apenas necessário um espaço de armazenagem em memória principal igual a

$$n + [\max\{n_k :\ 1 \leq k \leq m\}]^2$$

para resolver o sistema $Ax = b$.

(ii) Forma Triangular por Blocos

Tal como para as matrizes triangulares, existem as chamadas Formas Triangulares Superior e Inferior. A matriz A tem a Forma Triangular Inferior se se puder escrever na forma

$$A = \begin{bmatrix} A_{11} & & & \\ A_{21} & A_{22} & & \\ \vdots & \vdots & \ddots & \\ A_{m1} & A_{m2} & \ldots & A_{mm} \end{bmatrix},\ A_{ii} \in \mathbb{R}^{n_i \times n_i},\ \sum_{i=1}^{m} n_i = n \tag{5.23}$$

Neste caso a resolução de $Ax = b$ é equivalente a resolver os m sistemas

$$A_{kk} x^k = b^k - \sum_{j=1}^{k-1} A_{kj} x^j,\ k = 1, 2, \ldots, m \tag{5.24}$$

Por outro lado, A tem a Forma Triangular Superior se A se puder escrever na forma

$$A = \begin{bmatrix} A_{11} & A_{12} & \ldots & A_{1m} \\ & A_{22} & \ldots & A_{2m} \\ & & \ddots & \vdots \\ & & & A_{mm} \end{bmatrix},\ A_{ii} \in \mathbb{R}^{n_i \times n_i},\ \sum_{i=1}^{m} n_i = n \tag{5.25}$$

A resolução de $Ax = b$ é então equivalente a resolver

$$A_{kk} x^k = b^k - \sum_{j=k+1}^{m} A_{kj} x^j,\ k = m, m-1, \ldots, 1 \tag{5.26}$$

Se usarmos armazenagem auxiliar e como as matrizes A_{ij}, com $i \neq j$, são apenas necessárias para calcular produtos matriz-vector, então a solução dos sistemas (5.24) e (5.26) requer um espaço em memória principal de $n + t_0 + t_0^2$, com

$$t_0 = \max\{n_i :\ 1 \leq i \leq m\} \tag{5.27}$$

(iii) Forma Diagonal com Bordo por Blocos

Neste caso a matriz A tem a forma

$$A = \begin{bmatrix} A_1 & & & & C_1 \\ & A_2 & & & C_2 \\ & & \ddots & & \vdots \\ & & & A_{m-1} & C_{m-1} \\ B_1 & B_2 & \dots & B_{m-1} & A_m \end{bmatrix} \tag{5.28}$$

onde $A_i \in \mathbb{R}^{n_i \times n_i}$, $B_i \in \mathbb{R}^{n_m \times n_i}$, $C_i \in \mathbb{R}^{n_i \times n_m}$ e $\sum_{i=1}^{m} n_i = n$. Podemos então escrever a matriz (5.28) na forma

$$\begin{bmatrix} A_1 & & & & \\ & A_2 & & & \\ & & \ddots & & \\ & & & A_{m-1} & \\ B_1 & B_2 & \dots & B_{m-1} & \bar{A}_m \end{bmatrix} \begin{bmatrix} I_1 & & & & \bar{C}_1 \\ & I_2 & & & \bar{C}_2 \\ & & \ddots & & \vdots \\ & & & I_{m-1} & \bar{C}_{m-1} \\ & & & & I_m \end{bmatrix} = LU$$

com

$$A_k \bar{C}_k = C_k$$

$$\bar{A}_m = A_m - \sum_{k=1}^{m-1} B_k \bar{C}_k$$

Tendo em conta que $Ax = b$ é equivalente a $Ly = b$ e $Ux = y$ e dadas as características das matrizes envolvidas na resolução desses dois sistemas, facilmente chegamos à conclusão que a resolução do sistema $Ax = b$ se pode fazer de acordo com os algoritmos para matrizes com formas triangulares por blocos. É evidente que as matrizes identidades dos blocos diagonais de U vão ser exploradas na resolução do sistema com essa matriz. Além disso, o processo de formação das matrizes A_k pode ser feito ao mesmo tempo que se resolve o sistema com a matriz L. Tendo em conta estas considerações obtemos o algoritmo descrito na figura 5.8.

Se usarmos memória auxiliar e armazenarmos as matrizes $\bar{C}_k$ e A_m em memória principal, então podemos resolver o sistema $Ax = b$ com $n - 1$ "chamadas" e um espaço de armazenagem em memória principal igual a

$$\underbrace{n}_{\text{vector } b} + \underbrace{n \times n_m}_{\text{matrizes } \bar{C}_k \text{ e } A_m} + \underbrace{t_0^2 + t_0}_{\text{matrizes } A_k \text{ e } B_k}$$

com $t_0 = \max\{n_i : \ 1 \le i \le m-1\}$. Este processo é portanto eficiente se n_m for suficientemente pequeno, que é o que acontece geralmente na prática.

(iv) Forma Tridiagonal por Blocos

Neste caso a matriz tem a forma

$$A = \begin{bmatrix} A_1 & C_1 & & & & \\ B_1 & A_2 & C_2 & & & \\ & B_2 & A_3 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & C_{m-1} \\ & & & & B_{m-1} & A_m \end{bmatrix} \tag{5.29}$$

Algoritmo 24

Para $k = 1, 2, \ldots, m-1$

Resolva $A_k x^k = b^k \rightarrow b^k$

$A_k \bar{C}_k = C_k$

Calcule $A_m = A_m - B_k \bar{C}_k$

$b^m = b^m - B_k b^k$

Resolva $A_m x^m = b^m \rightarrow b^m$

Para $k = 1, \ldots, m-1$

Calcule $b^k = b^k - \bar{C}_k b^m$

Figura 5.8: Resolução de sistemas de equações com matrizes na Forma Diagonal com Bordo por Blocos

onde $A_i \in \mathbb{R}^{n_i \times n_i}$, $B_i \in \mathbb{R}^{n_{i+1} \times n_i}$, $C_i \in \mathbb{R}^{n_i \times n_{i+1}}$ e $\sum_{i=1}^{n} n_i = n$. Podemos escrever a matriz (5.29) na forma

$$\begin{bmatrix} \bar{A}_1 & & & & \\ B_1 & \bar{A}_2 & & & \\ & B_2 & \ddots & & \\ & & \ddots & \ddots & \\ & & & B_{m-1} & \bar{A}_m \end{bmatrix} \begin{bmatrix} I_1 & \bar{C}_1 & & & \\ & I_2 & \bar{C}_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \bar{C}_{m-1} \\ & & & & I_m \end{bmatrix} = LU \tag{5.30}$$

com

$$\bar{A}_1 = A_1$$

$$\bar{A}_k \bar{C}_k = C_k, k = 1, 2, \ldots, m-1$$

$$\bar{A}_{k+1} = A_{k+1} - B_k \bar{C}_k, k = 1, 2, \ldots, m-1$$

Tendo em conta que $Ax = b$ é equivalente a $Ly = b$ e $Ux = y$ e seguindo um raciocínio semelhante ao usado na forma anterior, facilmente obtemos o algoritmo 25, que apresentamos na figura 5.9.

Se usarmos memória auxiliar, então em cada passo k é necessário considerar espaço de armazenagem em memória principal para as matrizes $\bar{C}_{k-1}$ e A_k (ou A_{k-1}). Portanto o sistema $Ax = b$ pode ser resolvido em $3n - 2$ "chamadas" e um espaço de armazenagem em memória principal igual a

$$\underbrace{n}_{\text{vector } b} + \underbrace{2t_0^2}_{\text{matrizes } A_k(A_{k-1}) \text{ e } \bar{C}_k} + \underbrace{t_0}_{\text{matriz } B_{k-1}}$$

Algoritmo 25

Resolva o sistema $A_1 x^1 = b^1 \rightarrow b^1$

Para $k = 2, \ldots, m$

Resolva o sistema $A_{k-1}\bar{C}_{k-1} = C_{k-1}$

Calcule $A_k = A_k - B_{k-1}\bar{C}_{k-1}$

$$b^k = b^k - B_{k-1}b^{k-1}$$

Resolva o sistema $A_k x^k = b^k \rightarrow b^k$

Calcule $b^{k-1} = b^{k-1} - \bar{C}_{k-1}b^k$

Figura 5.9: Algoritmo para resolução de sistemas de equações com matrizes na Forma Tridiagonal por Blocos

com t_0 dado por (5.27).

É evidente que os processos referidos nesta secção podem também ser usados trabalhando apenas em memória principal, desde que haja suficiente espaço em RAM para a armazenagem de todas as matrizes blocos que constituem as diferentes formas. Com efeito a consideração das formas por blocos referidas faz diminuir consideravelmente o espaço de armazenagem, mesmo neste último caso. Se além disso as matrizes bloco forem esparsas, então é mesmo possível resolver sistemas de grandes dimensões integralmente em memória principal, conforme veremos nos dois próximos capítulos.

Exercícios

1. Considere as seguintes matrizes

$$A_1 = \begin{bmatrix} 1 & 2 & 3 & 0 & 0 \\ -1 & 1 & -1 & 0 & 0 \\ 0 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & -1 \\ 0 & 0 & 3 & 1 & 3 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 4 & -1 & 0 & 0 & 0 \\ -1 & 4 & -2 & 1 & 0 \\ 0 & -2 & 3 & 0 & -1 \\ 0 & 1 & 0 & 4 & 0 \\ 0 & 0 & -1 & 0 & 3 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 \\ 2 & 1 & -2 & 0 & 0 \\ 0 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 1 & 3 & 4 & 1 & 0 & 0 \\ 0 & 0 & -2 & 2 & 1 & 0 \\ 0 & 0 & 1 & -1 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 3 \end{bmatrix}$$

Determine os seus comprimentos de banda e armazene-as de acordo com o esquema de armazenagem de banda.

2. Desenvolva uma implementação dos algoritmos 19 e 20 para a resolução do sistema $Ax = b$ usando o esquema de armazenagem de banda.

3. Desenvolva um algoritmo para calcular a norma ℓ_∞ de uma matriz $A \in$ SPD usando o esquema de banda.

4. Demonstre o teorema 5.2.

5. Calcule o número de operações necessárias para determinar a decomposição LU de uma matriz não simétrica A com comprimentos de banda inferior $\alpha(A)$ e superior $\beta(A)$.

6. Considere uma matriz A de ordem n e de comprimentos de banda superior e inferior iguais a 3 e 2 respectivamente.

   (a) Diga como armazena a matriz A.

   (b) Desenvolva um algoritmo para o cálculo de Ab, com b um vector de ordem n usando o esquema de armazenagem mencionado na alínea anterior.

7. Considere uma matriz A de comprimentos de banda inferior e superior iguais a 2. Desenvolva um algoritmo para verificar se a matriz A é diagonalmente dominante por colunas, usando o esquema de armazenagem apropriado.

8. Sejam dadas duas matrizes tridiagonais não simétricas A e B.

   (a) Determine o comprimento de banda da matriz AB.

   (b) Desenvolva um algoritmo para calcular o produto AB.

9. Diga como resolve os sistemas de equações lineares com as matrizes

$$A_1 = \begin{bmatrix} B_1 & 0 & 0 & 0 \\ D_1 & B_2 & D_2 & 0 \\ 0 & 0 & B_3 & D_2 \\ 0 & 0 & 0 & B_4 \end{bmatrix}, \quad A_2 = \begin{bmatrix} B_1 & 0 & 0 \\ 0 & B_2 & D_1 \\ D_1 & 0 & B_3 \end{bmatrix}$$

sabendo que as matrizes B_i são quadradas e não singulares.

10. Resolva os sistemas $Ax = b$, onde

(a)

$$A = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 2 \\ -1 & 2 & 1 & -1 & 0 & 0 & 2 & 0 \\ 0 & 1 & -1 & 2 & -1 & 0 & -1 & 3 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ -1 & 2 & 0 & 0 & -1 & 1 & -1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 4 \\ 3 \\ 3 \\ 3 \\ 2 \\ 2 \\ 3 \end{bmatrix}$$

(b)

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & -1 & 2 \\ 0 & 0 & 0 & 2 & 0 & 1 & 0 & -3 \\ 0 & 0 & 0 & 0 & 3 & -2 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & -3 & 0 & 0 & 0 & 1 & 0 & 3 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 0 \\ 2 \\ 1 \\ 3 \\ 1 \end{bmatrix}$$

(c)

$$A = \begin{bmatrix} 1 & 1 & 1 & 3 & 2 & 0 & 0 \\ -1 & 2 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ -1 & 2 & 0 & 3 & 1 & 1 & 0 \\ 1 & 0 & -1 & 2 & 1 & -1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 7 \\ 1 \\ 0 \\ 4 \\ 3 \\ 6 \\ 4 \end{bmatrix}$$

11. Considere a matriz

$$A = \begin{bmatrix} B & -e & 0 & 0 \\ e^T & 1 & -2 & g^T \\ 0 & 2 & 1 & h^T \\ 0 & 0 & 0 & D \end{bmatrix}$$

com $e \in \mathbb{R}^m$, $g, h \in \mathbb{R}^n$, B uma matriz PD tridiagonal de ordem m e D uma matriz SPD de ordem n e comprimento de banda igual a 3.

(a) Mostre que $\det(A) > 0$.

(b) Diga como resolve um sistema com essa matriz, indicando o número de operações necessárias para o resolver.

12. Considere a matriz

$$A = \begin{bmatrix} B_1 & 0 & -e \\ 0 & B_2 & -f \\ e^T & f^T & 0 \end{bmatrix}$$

com $e \in \mathbb{R}^m$, $f \in \mathbb{R}^n$, B_1 e B_2 duas matrizes SPD de ordens m e n respectivamente.

(a) Mostre que A é uma matriz PSD e que $\det(A) > 0$.

(b) Diga como resolve um sistema com essa matriz tirando partido da sua estrutura.

Capítulo 6

Armazenagem e Operações com Vectores e Matrizes Esparsas

Por matriz esparsa entendemos uma matriz com um número suficientemente elevado de elementos nulos, que se torna compensador procurar esquemas de armazenagem (estruturas de dados) que permitam tirar partido da existência dos seus poucos elementos não nulos. Um conceito idêntico pode ser introduzido para vectores.

Neste capítulo iremo-nos debruçar sobre as questões levantadas pela armazenagem e operações com matrizes e vectores esparsos.

6.1 Armazenagem de vectores

O processo mais simples de se armazenar um vector esparso real x consiste em considerarmos uma estrutura de dados constituída por dois vectores:

1. um vector de reais, VALX, contendo os sucessivos elementos não nulos do vector x;
2. um vector de inteiros INDX, tal que INDX(k) indica a posição no vector VALX do seu elemento não nulo VALX(k).

A título de exemplo, consideremos o seguinte vector:

$$x = \left[\begin{array}{cccccccccc} 1. & 0. & 0. & -1. & 0. & 0. & 0. & 2. & 0. \end{array} \right]$$

Neste caso, temos

$$\text{VALX} = \left[\begin{array}{ccc} 1. & -1. & 2. \end{array} \right]$$
$$\text{INDX} = \left[\begin{array}{ccc} 1 & 4 & 8 \end{array} \right]$$

Na armazenagem do vector esparso é ainda necessário conhecer a dimensão N do vector e o seu número de elementos não nulos NZX. Assim, neste exemplo tem-se N= 9 e NZX= 3. Em termos do espaço de armazenagem, esta estrutura de dados requer um espaço de 2NZX, o que se torna vantajoso se NZX for muito menor que N, ou seja, se o vector for muito esparso.

Uma característica desta estrutura de dados e comum a todos os esquemas esparsos (sejam para vectores ou para matrizes), é o facto de requererem dois tipos de armazenagem:

1. Armazenagem primária, constituída pelos elementos não nulos propriamente ditos que se pretendem armazenar (VALX);
2. Armazenagem secundária, correspondente às variáveis e vectores auxiliares que nos permitem identificar as posições dos elementos não nulos (INDX).

Nas próximas secções apresentamos algoritmos que implementam algumas operações envolvendo vectores esparsos, tais como o produto interno e a combinação linear de dois vectores esparsos.

6.2 Operações com vectores esparsos

(i) Produto interno de dois vectores

Sejam $x, y \in \mathbb{R}^n$, com $x = (x_1, \ldots, x_n)$ e $y = (y_1, \ldots, y_n)$. O seu produto interno é dado por

$$x^T y = \sum_{i=1}^{n} x_i y_i$$

A implementação deste processo para o caso denso é muito simples, tendo a seguinte forma:

Algoritmo 26

$sum = 0.$

Para $k = 1, \ldots, n$

$sum = sum + x(i) * y(i)$

Portanto o cálculo do produto de dois vectores densos de ordem n requer n multiplicações e $n-1$ adições.

A implementação que vamos apresentar para o caso esparso baseia–se na equivalência:

$$x_i y_i \neq 0 \Leftrightarrow x_i \neq 0 \wedge y_i \neq 0$$

Portanto basta–nos considerar as componentes i tais que $x_i \neq 0$ e $y_i \neq 0$. Deste modo, um primeiro processo a considerar seria percorrer todas as posições não nulas de x (por exemplo), indicadas pelo vector INDX, e multiplicar os elementos nessas posições pelos elementos correspondentes de y, caso estes sejam não nulos. Esta implementação requer que se faça uma comparação para cada x_i não nulo, o que pode ser gravoso para vectores de ordem elevada e não muito esparsos. Nesse sentido, a implementação que vamos apresentar leva a cabo o produto interno sem fazer comparações. Em vez disso é considerado um vector auxiliar denso de ordem n, que irá corresponder à *representação densa* do vector y. Assim, por exemplo, se o vector y esparso de ordem 10 é dado por

$$\text{VALY} = \begin{bmatrix} 3. & 1. & -2. & 1. \end{bmatrix} \quad \text{INDY} = \begin{bmatrix} 2 & 4 & 7 & 9 \end{bmatrix}$$

então o vector auxiliar referido tem a forma

$$w = \begin{bmatrix} 0. & 3. & 0. & 1. & 0. & 0. & -2. & 0. & 1. & 0. \end{bmatrix}$$

Note–se que o k-ésimo elemento não nulo de y é dado por $w(\text{INDY}(k))$. Então, cada uma das parcelas não nulas $x_i y_i$ é facilmente acedida por

$$\text{VALX}(K) * W(\text{INDX}(K)),$$

e portanto o Algoritmo do Cálculo do Produto Interno de dois vectores esparsos x e y de ordem n e com nzx e nzy elementos não nulos respectivamente tem a forma:

Algoritmo 27

$prod = 0.$

(construção do vector auxiliar w *)*

Para $k = 1, \ldots, n$

$w(k) = 0.$

Para $k = 1, \ldots, nzy$

$w(indy(k)) = valy(k)$

(obtenção de* $\sum_{i=1}^{n} x_i y_i$ **)*

Para $k = 1, \ldots, nzx$

$prod = prod + valx(k) \cdot w(indx(k))$

Este algoritmo requer NZX multiplicações, o que se torna particularmente eficaz se NZX for muito menor que n, ou seja, se x for muito esparso. Tem no entanto a desvantagem de requerer a armazenagem adicional de um vector de ordem n.

(ii) Combinação linear de dois vectores

Nesta secção iremos considerar uma combinação linear do tipo $x + \alpha y$, com $x, y \in \mathbb{R}^n$ esparsos e $\alpha \in \mathbb{R}$. A soma $x + y$ é um caso particular e este procedimento poderá ser generalizado para a combinação linear $\alpha_1 x^1 + \alpha_2 x^2 + \ldots + \alpha_k x^k$, com $\alpha_i \in \mathbb{R}, x^i \in \mathbb{R}^n, i = 1, \ldots, k$.

Iremos descrever o processo de obtenção do vector $x + \alpha y$ com o auxílio de um exemplo. Para isso consideremos os vectores esparsos x e y de ordem 10 definidos por

$$\begin{array}{ll} \text{VALX} = \left[\begin{array}{cccc} 1. & 2. & 1. & 4. \end{array}\right] & \text{VALY} = \left[\begin{array}{cccc} 1. & 1. & 2. & 1. \end{array}\right] \\ \text{INDX} = \left[\begin{array}{cccc} 1 & 4 & 6 & 7 \end{array}\right] & \text{INDY} = \left[\begin{array}{cccc} 2 & 5 & 6 & 10 \end{array}\right] \end{array}$$

Comecemos por considerar as representações densas destes dois vectores:

$$x = \left[\begin{array}{cccccccccc} 1. & 0. & 0. & 2. & 0. & 1. & 4. & 0. & 0. & 0. \end{array}\right]$$

$$y = \left[\begin{array}{cccccccccc} 0. & 1. & 0. & 0. & 1. & 2. & 0. & 0. & 0. & 1. \end{array}\right]$$

A combinação $x + \alpha y$ é feita componente a componente, bastando considerar as componentes que sejam não nulas em um dos vectores. Há então três possibilidades a considerar para a i-ésima componente:

Caso 1: $x_i \neq 0$ e $y_i = 0$;

Caso 2: $x_i = 0$ e $y_i \neq 0$;

Caso 3: $x_i \neq 0$ e $y_i \neq 0$.

O algoritmo que vamos apresentar vai supor que os vectores x e y são dados pelas suas representações de esparsidade, guardando o resultado da combinação $x + \alpha y$ na estrutura de x, perdendo–se deste modo a informação relativa a este vector. Em seguida analisamos cada uma das três hipóteses apresentadas:

Caso 1: Neste caso não há nada a somar ao valor da i-ésima componente de x, pelo que nada há a fazer;

Caso 2: Temos que acrescentar o valor não nulo de αy a VALX e o índice i a INDX;

Caso 3: Se $x_i \neq 0$, este elemento está assinalado na estrutura de esparsidade de x, pelo que basta colocar $x_i + \alpha y_i$ em seu lugar.

Voltando ao nosso exemplo inicial, vejamos para que componentes ocorre cada um dos casos mencionados:

$x =$	1.	0.	0.	2.	0.	1.	4.	0.	0.	0.
`Caso:`	1	2	-	1	2	3	1	-	-	2
$y =$	0.	1.	0.	0.	1.	2.	0.	0.	0.	1.

As componentes assinaladas por um traço (`-`) são as componentes i tais que

$$x_i = 0 \text{ e } y_i = 0.$$

Essas componentes não estão assinaladas nas estruturas de dados de x e de y, e, como não há nada a fazer com elas, podemos proceder como se elas não existissem.

A última questão consiste em determinar, para cada componente, qual o caso em que ela pode ser incluída. Tal irá ser feito considerando um vector auxiliar w que inicialmente corresponderá à representação densa de y. Como só são conhecidas de início as representações esparsas de x e de y, este vector w terá que ser efectivamente construído, o que pode ser feito de forma semelhante à apresentada no caso do produto interno, correspondendo aos dois primeiros ciclos do algoritmo que apresentamos a seguir:

Algoritmo 28

Para $k = 1, \ldots, n$
  $w(k) = 0.$

Para $k = 1, \ldots, nzy$
  $w(indy(k)) = valy(k)$

Para $k = 1, \ldots, nzx$
  $valx(k) = valx(k) + \alpha \cdot w(indx(k))$
  $w(indx(k)) = 0.$

Para $k = 1, \ldots, nzy$
  Se $w(indy(k)) \neq 0$ *então*
    $nzx = nzx + 1$
    $valx(nzx) = \alpha \cdot w(indy(k))$
    $indx(nzx) = indy(k)$

O vector w pode agora ser usado para construir as componentes de $x + \alpha y$ associadas ao terceiro caso, ou seja, as componentes i para as quais $x_i \neq 0$ e $y_i \neq 0$. Para obter os valores nas restantes componentes será conveniente eliminá–las de w. Estas duas tarefas podem ser efectuadas simultaneamente de acordo com o terceiro ciclo do algoritmo 28.

É de notar que ficaram assim resolvidos os casos 1 e 3 (fazer a soma $x_i + \alpha y_i$ e colocá–la na posição de x_i na estrutura de dados de x). Além disso eliminámos de w os elementos não nulos de y que estavam em posições não nulas em x. Por isso, restam–nos as componentes que correspondem ao caso 2, isto é, $x_i = 0$ e $y_i \neq 0$.

O tratamento do caso 2 é feito por consulta de w, uma vez que este vector já só contém os elementos para os quais $y_i \neq 0$ e $x_i = 0$. Neste caso, teremos que acrescentar os valores αy_i à estrutura de esparsidade de x, acrescentando αy_i a VALX e i a INDX. Isso pode ser feito de acordo com o quarto ciclo do algoritmo 28.

Este processo requer que se percorra a estrutura de y duas vezes e a de x uma vez. Além disso, são necessárias NZX adições e NZY multiplicações, o que torna este esquema bastante eficaz se x e y são muito esparsos.

6.3 Armazenagem de matrizes esparsas

Se A é uma matriz densa não há muito a fazer em termos de poupança de espaço de armazenagem, uma vez que uma matriz de ordem n necessita sempre de n^2 elementos para a representar. Neste caso, as operações de adição e produto de matrizes são de implementação trivial e do domínio da Álgebra Linear, pelo que não voltaremos a referir este tipo de armazenagem ao longo deste capítulo.

Na armazenagem das matrizes densas todos os elementos nulos ou não nulos são considerados. A aramazenagem da matriz A requer exactamente n^2 elementos, que podem ser dispostos por linhas ou por colunas. A linguagem de programação usada é muitas vezes importante nessa escolha. Assim, em FORTRAN, uma matriz A real quadrada de ordem n pode ser declarada da seguinte forma:

```
DIMENSION A(N, N)
```

e é armazenada do seguinte modo

Coluna 1				Coluna 2				...	Coluna n			
a_{11}	a_{21}	...	a_{n1}	a_{12}	a_{22}	...	a_{n2}	...	a_{1n}	a_{2n}	...	a_{nn}

devendo os algoritmos ser construídos de modo a operarem sobre esta estrutura de dados.

As estruturas de dados que se implementam para armazenar matrizes esparsas têm geralmente em vista considerar apenas os elementos não nulos da matriz. Outras alternativas são possíveis, nomeadamente as que permitem a armazenagem de alguns elementos nulos que estejam colocados em posições estratégicas. Esses esquemas não serão para já alvo do nosso estudo, centrando–se a nossa atenção por agora no Esquema Coordenado e na Colecção de Vectores.

Estes esquemas têm como característica comum requererem dois tipos de armazenagem, nomeadamente uma Parte Primária, constituída por um (ou mais) vector de reais contendo todos os elementos não nulos da matriz, e uma Parte Secundária correspondente a pelo menos um vector de inteiros contendo a informação necessária para localizar na parte primária cada um dos elementos não nulos da matriz.

(i) Esquema Coordenado

É um esquema de armazenagem usado como *interface* com o utilizador em algumas *packages* de resolução de sistemas de equações lineares com matrizes esparsas (MA27, MA28, por exemplo). A parte primária é constituída por todos os elementos não nulos da matriz A dispostos em qualquer ordem. A parte secundária é constituída por dois vectores que contêm os índices de linha e de coluna dos elementos não nulos. Assim, por exemplo, se A for dada por

$$A = \begin{bmatrix} 0. & 2. & 0. & 3. & 0. \\ 0. & 0. & -1. & 1. & -2. \\ 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. \end{bmatrix}$$

então o esquema coordenado para representar A toma a seguinte forma:

VALA =	2.	3.	-1.	1.	-2.	1.
IRN =	1	1	2	2	2	3
ICN =	2	4	3	4	5	2

Do ponto de vista da armazenagem, este esquema requer 2 NZA de armazenagem inteira e NZA de armazenagem real, em que NZA é o número de elementos não nulos da matriz A. Por outro lado, usando este esquema, podemos aceder aos elementos não nulos de A através de

```
A(IRN(K), ICN(K)) = VALA(K), K = 1, ..., NZA
```

Apesar de a sua formulação ser extremamente simples, este esquema necessita de um espaço de armazenagem elevado. Além disso algumas operações com matrizes esparsas (multiplicação de matrizes, produto de uma matriz por um vector) não podem ser implementadas de uma forma eficiente usando este esquema, pelo menos em comparação com outros esquemas de que falaremos em seguida. É por essa razão que o esquema coordenado é habitualmente usado apenas como esquema de input/output entre o programa de manipulação de matrizes esparsas e o utilizador, sendo feita, após a leitura, a passagem para um esquema mais facilmente manuseável pelo programa, como o da colecção de vectores.

(ii) Colecção de vectores para matrizes não simétricas

Conforme o nome indica, este esquema tem como ponto de partida a *vectorização* da matriz, que consiste em considerar cada linha (ou coluna) da matriz como sendo um vector esparso, e armazenar depois esta Colecção de Vectores. Obtemos assim um Esquema Orientado por Linhas ou um Esquema Orientado por Colunas, conforme o caminho escolhido para a vectorização.

No Esquema Orientado por Linhas, a parte primária consiste num vector real VALA, de ordem NZA, onde são guardadas as componentes não nulas da matriz A (tal como no esquema coordenado), e cuja parte secundária consiste em dois vectores de inteiros: INDA, de ordem NZA, e PNTA, de ordem $n+1$. O vector INDA vai armazenar na sua componente k o índice de coluna do elemento não nulo de A que figura na k-ésima posição de VALA, com $k = 1, \ldots,$ NZA. Além disso, PNTA(k) representa a posição em INDA do primeiro elemento não nulo da linha k da matriz A, com $k = 1, \ldots, n$. Falta esclarecer o que será colocado em PNTA($n+1$). Para o fazer, comecemos por observar que o vector PNTA, tal como está definido, permite determinar o número de elementos não nulos da linha k de A, através de PNTA($k+1$)-PNTA(k), com $k = 1, \ldots, n-1$. Para que possamos ter essa informação relativamente à última linha de A armazenada em PNTA, teremos que fazer

```
PNTA(N + 1) = PNTA(N) + NZA(N)
```

em que `NZA(N)` representa o número de elementos não nulos na linha n de A.

Consideremos novamente a matriz apresentada na secção anterior:

$$A = \begin{bmatrix} 0. & 2. & 0. & 3. & 0. \\ 0. & 0. & -1. & 1. & -2. \\ 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. \end{bmatrix}$$

O esquema orientado por linhas será:

VALA =	2.	3.	-1.	1.	-2.	1.
INDA =	2	4	3	4	5	2
PNTA =	1	3	6	7	7	

Se a matriz A tiver n linhas, m colunas e NZA elementos não nulos e se usarmos o esquema orientado por linhas, então o espaço de armazenagem total requerido será de $2\text{NZA} + n + 1$.

Um esquema de armazenagem diz–se Ordenado se os elementos de cada linha estiverem dispostos pela sua ordem natural, ou seja, por ordem crescente de índices de colunas. Nesse sentido, o exemplo anterior corresponde a um esquema ordenado. De outro modo diz–se Não Ordenado.

O Esquema Orientado por Colunas tem uma formulação semelhante à do esquema orientado por linhas, com a diferença de que a armazenagem é feita coluna a coluna em vez de ser linha a linha. É portanto utilizado um vector VALA de ordem NZA para armazenar os sucessivos elementos não nulos da matriz, considerados coluna a coluna, e dois vectores de inteiros, INDA e PNTA, de ordens NZA e $m + 1$ repectivamente, e com funções idênticas às que desempenham no esquema orientado por linhas.

Assim, e retomando o exemplo que temos vindo a utilizar, a matriz A pode ser representada, usando o esquema orientado por colunas, por:

VALA =	1.	2.	-1.	3.	1.	-2.
INDA =	3	1	2	1	2	2
PNTA =	1	1	3	4	6	7

Como é evidente, para o esquema orientado por colunas podem–se considerar versões Ordenadas e Não Ordenadas.

Se todos os elementos diagonais de uma matriz são não nulos, então a sua diagonal pode ser considerada como um vector denso. Nesse caso, é possível aliviar um pouco a estrutura de dados, usando um Esquema de Diagonal Separada, que consiste na remoção dos elementos diagonais da estrutura construída, colocando-os num vector separado DIAG. Tal alteração corresponde não só à remoção de elementos reais do vector VALA, mas também dos elementos correspondentes em INDA, assim como à correspondente actualização em PNTA. Assim, por exemplo, a matriz

$$A = \begin{bmatrix} 3. & 0. & -1. & 0. \\ 0. & 2. & 0. & 0. \\ 0. & 1. & -3. & 0. \\ 1. & 0. & 0. & 1. \end{bmatrix}$$

pode ser armazenada, usando o esquema orientado ordenado por linhas com diagonal separada, do modo seguinte:

DIAG =	3.	2.	-3.	1.	
VALA =	-1.	1.	1.		
INDA =	3	2	1		
PNTA =	1	2	2	3	4

Usando este esquema, há uma redução em n do espaço de armazenagem total.

(iii) Colecção de vectores para matrizes simétricas

Se uma matriz é simétrica, então apenas a diagonal e os elementos abaixo da diagonal têm que ser armazenados. Os esquemas de armazenagem para matrizes simétricas entram com isso em linha de conta, e acabam por ser bastante semelhantes aos das matrizes não simétricas. Assim, podemos considerar esquemas orientados por linhas ou por colunas, ordenados ou não ordenados, com ou sem diagonal separada , que consistem no recurso aos vectores VALA, INDA, PNTA e DIAG com significados semelhantes aos dos definidos anteriormente. Para a ilustração destes esquemas consideremos a matriz simétrica:

$$A = \begin{bmatrix} 2. & 0. & 0. & 1. \\ 0. & 1. & 0. & -1. \\ 0. & 0. & 3. & 0. \\ 1. & -1. & 0. & 3. \end{bmatrix}$$

Então A pode ser armazenada usando o esquema orientado por linhas, do seguinte modo:

VALA =	2.	1.	3.	1.	-1.	3.
INDA =	1	2	3	1	2	4
PNTA =	1	2	3	4	7	

Por outro lado, o esquema orientado por colunas tem a seguinte forma:

VALA =	2.	1.	1.	-1.	3.	3.
INDA =	1	4	2	4	3	4
PNTA =	1	3	5	6	7	

Este esquema traduz–se numa significativa poupança em termos de armazenagem, uma vez que só são armazenados metade dos elementos não nulos não diagonais, além dos elementos não nulos na diagonal. Se todos os elementos diagonais de A são não nulos, então será útil considerar esquemas de diagonal separada.

Se considerarmos o exemplo anterior, a sua armazenagem através do esquema orientado por linhas com diagonal separada irá ser:

DIAG =	2.	1.	3.	3.	
VALA =	1.	-1.			
INDA =	1	2			
PNTA =	1	1	1	1	3

6.4 Operações com matrizes esparsas

Nesta secção iremos considerar algumas operações com matrizes esparsas, nomeadamente soma e o produto de duas matrizes, produto de uma matriz por um vector, transposição de uma matriz e resolução de sistemas triangulares. Estes problemas são bastante fáceis de resolver no caso denso, tornando-se mais complicados quando se usam matrizes esparsas. Tal dificuldade é motivada pelas estruturas de dados específicas a que se tem que recorrer. É essa a razão que nos leva a dar-lhes uma atenção especial.

(i) Adição de duas matrizes esparsas

Sejam dadas duas matrizes $A, B \in \mathbb{R}^{m\times n}$. Se suposermos que as matrizes se encontram armazenadas segundo um qualquer esquema, seja ele por linhas ou por colunas, a adição das duas matrizes resume-se à adição de m ou n vectores esparsos. Com efeito, se A e B estiverem armazenadas por colunas, então, tem-se

$$A = [A_{.1}, A_{.2}, \ldots, A_{.n}], \quad B = [B_{.1}, B_{.2}, \ldots, B_{.n}]$$

e

$$C = A + B = [A_{.1} + B_{.1}, A_{.2} + B_{.2}, \ldots, A_{.n} + B_{.n}]$$

Por outro lado, se A e B estiverem armazenadas por linhas

$$A = \begin{bmatrix} A_{1.} \\ A_{2.} \\ \vdots \\ A_{m.} \end{bmatrix}, \quad B = \begin{bmatrix} B_{1.} \\ B_{2.} \\ \vdots \\ B_{m.} \end{bmatrix}$$

então

$$C = \begin{bmatrix} A_{1.} + B_{1.} \\ A_{2.} + B_{2.} \\ \vdots \\ A_{m.} + B_{m.} \end{bmatrix}$$

De notar que cada linha ou coluna de C pode ser obtida como a soma de dois vectores esparsos, pelo que a implementação do processo de adição de duas matrizes se baseia no algoritmo 28 para a soma de dois vectores. Seguidamente apresentamos a implementação desse processo quando A e B estão armazenadas por linhas em esquemas de colecção de vectores sem diagonal separada, constituídos pelos vectores (VALA,INDA,PNTA) e (VALB,INDB,PNTB) respectivamente. Esse processo vai construindo os vectores VALC, INDC e PNTC correspondentes à estrutura de dados que armazena por linhas a matriz C e tem a forma apresentada na Figura 6.1.

(ii) Produto de uma matriz esparsa por um vector coluna denso

Seja A uma matriz esparsa $m \times n$ armazenada segundo o esquema orientado e

$$b = [b_1, b_2, \ldots, b_n]^T$$

um vector coluna. Suponhamos que se pretende determinar $c = Ab = [c_1, c_2, \ldots, c_m]^T$. Dois casos podem acontecer e são discutidos seguidamente.

1. Se A está armazenada por linhas, isto é, se

$$A = \begin{bmatrix} A_{1.} \\ A_{2.} \\ \vdots \\ A_{m.} \end{bmatrix}$$

então

$$c = Ab = \begin{bmatrix} A_{1.} \\ A_{2.} \\ \vdots \\ A_{m.} \end{bmatrix} b = \begin{bmatrix} A_{1.}b \\ A_{2.}b \\ \vdots \\ A_{m.}b \end{bmatrix}$$

Portanto o processo pode ser descrito pelo seguinte algoritmo:

Algoritmo 30

Para $k = 1, 2, \ldots, m$

$c_k = A_{k.}b$ *(* produto interno *)*

Algoritmo 29

```
PNTC(1) = 1
Para l = 1, ..., m
    Para k = 1, ..., n
        w(k) = 0.
    inicio = PNTB(l)
    fim = PNTB(l + 1) − 1
    Se fim ≥ inicio faça
        Para k = inicio, ..., fim
            w(INDB(k)) = VALB(k)
    inicio2 = PNTA(l)
    fim2 = PNTA(l + 1)
    nz = PNTC(l)
    Se fim2 ≥ inicio2 faça
        Para k = inicio2, ..., fim2
            VALC(nz) = VALA(k) + w(INDA(k))
            w(INDA(k)) = 0
            INDC(nz) = INDA(k)
            nz = nz + 1
        Para k = inicio, ..., fim
            Se w(INDB(k)) ≠ 0 faça
                VALC(nz) = w(INDB(k))
                INDC(nz) = INDB(k)
                nz = nz + 1
    PNTC(l + 1) = nz
```

Figura 6.1: Adição de duas matrizes esparsas armazenadas segundo o esquema de colecção de vectores por colunas.

Há portanto a necessidade de efectuar m produtos internos de um vector esparso por um vector cheio. O algoritmo 27 poderá ser usado para esse efeito, usando $b = w$, não sendo desta forma necessário o recurso a um vector auxiliar. Assim por exemplo se A é uma matriz quadrada de ordem n ($m = n$) armazenada por linhas numa estrutura de diagonal separada, constituída pelos vectores VALA, INDA, PNTA e DIAG, então a implementação desse algoritmo tem a seguinte forma

Algoritmo 31

Para $l = 1, \ldots, n$
- $c(l) = \text{DIAG}(l) \times b(l)$
- inicio $= \text{PNTA}(l)$
- fim $= \text{PNTA}(l+1) - 1$
- *Se* fim $\geq$ inicio *faça*
  - *Para* $k =$ inicio$, \ldots,$ fim
    - $c(l) = c(l) + \text{VALA}(k) \times b(\text{INDA}(k))$

2. Se A está armazenada por colunas, isto é, se

$$A = \left[\begin{array}{cccc} A_{.1} & A_{.2} & \ldots & A_{.n} \end{array} \right]$$

então

$$c = Ab = \left[\begin{array}{cccc} A_{.1} & A_{.2} & \ldots & A_{.n} \end{array} \right] \left[\begin{array}{c} b_1 \\ b_2 \\ \vdots \\ b_n \end{array} \right] = b_1 A_{.1} + b_2 A_{.2} + \ldots + b_n A_{.n}$$

Deste modo, para determinar o vector c há necessidade de adicionar n vectores esparsos. Contudo, como se pretende armazenar o vector c como um vector denso, podemos simplificar o algoritmo 28 e descrever o processo na seguinte forma:

Algoritmo 32

$c = 0$

Para $k = 1, 2, \ldots, n$

- $c = c + b_k A_k$ *(* adição de vectores esparsos *)*

Apesar de a implementação deste processo usando uma estrutura de dados de colecção de vectores parecer ser mais complexa do que no caso em que A é armazenada por linhas, tal não acontece na realidade. Com efeito, como o vector c é denso, então o processo consiste simplesmente em ir colocando os elementos não nulos do vector coluna de A no vector c. A implementação deste processo é deixada como exercício.

(iii) Produto de um vector linha denso por uma matriz esparsa

Seja A uma matriz $m \times n$ esparsa, b um vector linha da forma

$$b = \left[\begin{array}{cccc} b_1 & b_2 & \ldots & b_m \end{array} \right]$$

e suponhamos que se pretende determinar o vector linha

$$c = \left[\begin{array}{cccc} c_1 & c_2 & \ldots & c_n \end{array} \right] = bA$$

Tal como anteriormente, duas situações podem ter lugar:

1. Se A está armazenada por linhas, então

$$c = bA = \left[\begin{array}{cccc} b_1 & b_2 & \ldots & b_m \end{array} \right] \left[\begin{array}{c} A_{1.} \\ A_{2.} \\ \vdots \\ A_{m.} \end{array} \right] = b_1 A_{1.} + b_2 A_{2.} + \ldots + b_m A_{m.}$$

   e o algoritmo 32 pode ser usado.

2. Se A está armazenada por colunas, então

$$c = bA = b \left[\begin{array}{cccc} A_{.1} & A_{.2} & \ldots & A_{.n} \end{array} \right] = \left[\begin{array}{cccc} bA_{.1} & bA_{.2} & \ldots & bA_{.n} \end{array} \right]$$

   e o algoritmo 30 pode ser usado.

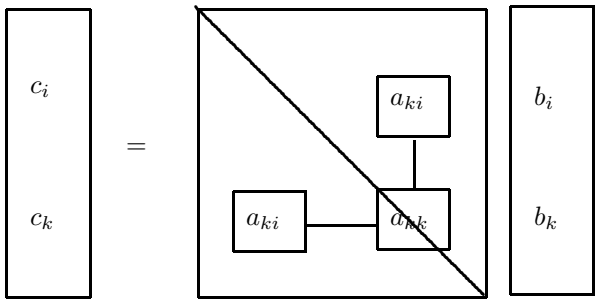
(iv) Produto de uma matriz simétrica esparsa por um vector

Se A é uma matriz simétrica esparsa de ordem n e b é um vector coluna, então b^T é um vector linha e além disso

$$b^T A = (A^T b)^T = (Ab)^T$$

pelo que podemos abordar apenas o caso da multiplicação de uma matriz por um vector coluna.

A particularidade das matrizes simétricas reside em apenas ser necessário armazenar a diagonal e o triângulo superior ou o inferior. De resto, os princípios fundamentais são os expressos nos algoritmos 30 e 32. Suponhamos que armazenamos o triângulo inferior e a diagonal. Assim se A estiver armazenada por linhas, tem-se a situação seguinte:



Assim, quando se faz o produto $a_{ki}b_i$ e esse valor é somado ao de c_k, também se terá que somar o valor $a_{ki}b_k$ ao de c_i. Tendo em conta estas considerações, então o algoritmo para calcular $c = Ab$, com A uma matriz simétrica esparsa armazenada por linhas com diagonal separada, tem a forma apresentada na Figura 6.2.

Por outro lado, se A estiver armazenada por colunas, a situação corresponde à que se representa na seguinte figura:

Algoritmo 33

Para $k = 1, \ldots, n$

$c(k) = \text{DIAG}(k) \times b(k)$

$início = \text{PNTA}(k)$

$fim = \text{PNTA}(k+1) - 1$

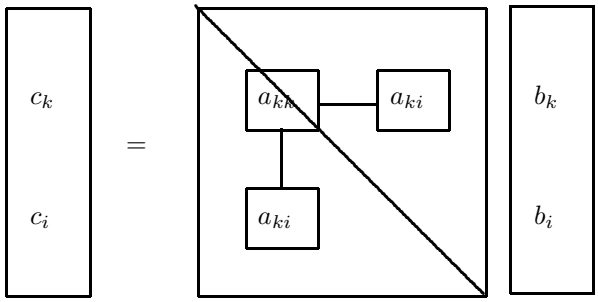
Se $fim \geq início$ *faça*

Para $j = início, \ldots, fim$

$c(k) = c(k) + \text{VALA}(j) \times b(\text{INDA}(j))$

$c(\text{INDA}(j)) = c(\text{INDA}(j)) + b(k) \times \text{VALA}(j)$

Figura 6.2: Algoritmo para o cálculo de $c = Ab$, com A uma matriz simétrica esparsa armazenada por linhas com diagonal separada.



Portanto o algoritmo 32 terá que ser modificado de uma forma idêntica à usada no caso de A ser armazenada por linhas.

(v) Produto de duas matrizes esparsas

Sejam A e B duas matrizes esparsas de ordens $m \times p$ e $p \times n$ respectivamente, e suponhamos que estamos interessados em obter a estrutura de dados de $C = AB$. Tal como anteriormente os algoritmos que vamos apresentar dependem da forma como as matrizes estão armazenadas.

1. Suponhamos que A e B estão armazenadas por linhas. Para se entender mais facilmente o processo que vamos descrever, consideremos A e B como sendo de ordem 2:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

Então a primeira linha $C_{1.}$ de C é dada por

$$\begin{aligned} C_{1.} &= \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \end{bmatrix} \\ &= \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} \end{bmatrix} + \begin{bmatrix} a_{12}b_{21} & a_{12}b_{22} \end{bmatrix} \\ &= a_{11} \begin{bmatrix} b_{11} & b_{12} \end{bmatrix} + a_{12} \begin{bmatrix} b_{21} & b_{22} \end{bmatrix} \end{aligned}$$

$$
\begin{aligned}
&= a_{11}B_{1.} + a_{12}B_{2.} \\
&= \sum_{k=1}^{2} a_{1k}B_{k.}
\end{aligned}
$$

onde $B_{k.}$ representa a k-ésima linha de B. Estendendo a fórmula para o caso geral,

$$C_{l.} = \sum_{k=1}^{p} a_{lk}B_{k.}, \quad l = 1, \ldots, m$$

podendo-se assim considerar o algoritmo

Algoritmo 34

Para $i = 1, \ldots, m$

$$C_{i.} = \sum_{k=1}^{p} a_{ik}B_{k.}$$

Portanto apenas as linhas de A e de B são usadas para determinar as sucessivas linhas de C.

2. Suponhamos agora que A e B estão armazenadas por colunas. Considerando os mesmos exemplos de ordem 2, a primeira coluna $C_{.1}$ de C pode ser obtida à custa de

$$C_{.1} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} \\ a_{21}b_{11} + a_{22}b_{21} \end{bmatrix} = b_{11}\begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} + b_{21}\begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} = b_{11}A_{.1} + b_{21}A_{.2}$$

em que $A_{.k}$ é a k-ésima coluna de A. Generalizando para A de ordem $m \times p$ e B de ordem $p \times n$ obtém-se

$$C_{.j} = \sum_{k=1}^{p} b_{kj}A_{.k}, \quad j = 1, \ldots, n$$

e o algoritmo será então o seguinte:

Algoritmo 35

Para $j = 1, \ldots, n$

$$C_{.j} = \sum_{k=1}^{p} b_{kj}A_{.k}$$

Tal como anteriormente, apenas as colunas de A e de B são usadas para calcular as sucessivas colunas de C.

(vi) Transposição de uma matriz esparsa

Consideremos uma matriz esparsa não simétrica A armazenada segundo um esquema de colecção de vectores por linhas (ou por colunas). Suponhamos que pretendemos obter a matriz transposta armazenada segundo a mesma estrutura.

Nesta secção iremos considerar apenas o caso em que a matriz A está armazenada por linhas, descrevendo os sucessivos passos do algoritmo com recurso a um exemplo. Consideremos então a matriz

$$A = \begin{bmatrix} 0. & 0. & a_{13} & 0. & a_{15} & a_{16} \\ a_{21} & 0. & 0. & a_{24} & 0. & 0. \\ 0. & 0. & a_{33} & a_{34} & 0. & 0. \\ a_{41} & 0. & a_{43} & a_{44} & 0. & 0. \\ 0. & a_{52} & 0. & 0. & a_{55} & a_{56} \end{bmatrix}$$

A representação de A por linhas segundo um esquema de colecção de vectores ordenado tem a forma

$$\begin{array}{rcl} \text{VALA} & = & \left[\begin{array}{cccccccccccccc} a_{13} & a_{15} & a_{16} & a_{21} & a_{24} & a_{33} & a_{34} & a_{41} & a_{43} & a_{44} & a_{52} & a_{55} & a_{56} \end{array} \right] \\ \text{INDA} & = & \left[\begin{array}{cccccccccccccc} 3 & 5 & 6 & 1 & 4 & 3 & 4 & 1 & 3 & 4 & 2 & 5 & 6 \end{array} \right] \\ \text{PNTA} & = & \left[\begin{array}{cccccc} 1 & 4 & 6 & 8 & 11 & 14 \end{array} \right] \end{array}$$

Se nza, m e n representarem respectivamente o número de elementos não nulos, de linhas e de colunas da matriz A, então neste caso temos $nza = 13$, $m = 5$ e $n = 6$. A estrutura de dados da matriz A^T é constituída por um vector VALAT de dimensão nza, um vector inteiro INDAT de dimensão nza e por um vector PNTAT de dimensão $n + 1$.

O algoritmo que passamos a apresentar tem início considerando todas as componentes desses vectores como nulas. Em seguida, determina-se o número de elementos não nulos em cada linha de A^T (isto é, em cada coluna de A) de modo a se poder obter o vector PNTAT. Essa determinação faz-se muito facilmente percorrendo o vector INDA. Como para já apenas pretendemos obter o número de elementos não nulos nas $n - 1$ primeiras linhas de A^T (as $n - 1$ primeiras colunas de A), basta-nos utilizar então as $n - 1$ últimas posições de PNTAT, deixando-se, por agora, as duas primeiras posições desocupadas. O esquema descrito pode então ser implementado do modo seguinte:

Para $k = 1, \ldots, nza$

$\quad w = \text{INDA}(k) + 2$

$\quad \text{PNTAT}(w) = \text{PNTAT}(w) + 1$

Fazendo esta operação para a matriz A, obtemos

$$\text{PNTAT} = \left[\begin{array}{cccccccc} 0 & 0 & 2 & 1 & 3 & 3 & 2 \end{array} \right]$$

Tal como está, PNTAT ainda não é um vector de apontadores, limitando-se a indicar quantos elementos não nulos ocorrem em cada linha de A^T, e não onde estão posicionados em VALAT e INDAT. Todavia, se $nzat(k)$ for o número de elementos não nulos na linha k de A^T, com $k = 1, \ldots, n$, então

$$\text{PNTAT}(k + 1) = \text{PNTAT}(k) + nzat(k)$$

pelo que a construção de PNTAT pode prosseguir através de

PNTAT(1) = 1

PNTAT(2) = 1

Para $k = 2, \ldots, n$

    PNTAT($k+1$) = PNTAT(k) + PNTAT($k+1$)

onde a primeira componente de PNTAT tomou o valor um por motivos óbvios. Assim, no nosso exemplo tem-se, no fim da aplicação deste passo,

$$\text{PNTAT} = \left[\begin{array}{cccccccc} 1 & 1 & 3 & 4 & 7 & 10 & 12 \end{array}\right]$$

Para terminar o processo de obtenção da estrutura de dados da matriz A^T há que determinar as componentes dos vectores INDAT e VALAT. Esse processo é feito usando o vector PNTAT de modo a que as suas componentes vão sendo modificadas até se obterem as componentes correctas desse vector. Seguidamente apresentamos o algoritmo que executa o pretendido.

Para $i = 1, \ldots, n$

    Para $k = \text{PNTA}(i), \ldots, \text{PNTA}(i+1) - 1$

        $j = \text{INDA}(k) + 1$

        $l = \text{PNTAT}(j)$

        INDAT(l) = i

        VALAT(l) = VALA(k)

        PNTAT(j) = $l + 1$

Assim, no exemplo considerado, no fim do ciclo $i = 1$ obtêm-se os seguintes vectores

$$\begin{array}{rcl} \text{VALAT} & = & \left[\begin{array}{ccccccccccccc} 0. & 0. & 0. & a_{13} & 0. & 0. & 0. & 0. & 0. & a_{15} & 0. & a_{16} & 0. \end{array}\right] \\ \text{INDAT} & = & \left[\begin{array}{ccccccccccccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right] \\ \text{PNTAT} & = & \left[\begin{array}{ccccccc} 1 & 1 & 3 & 5 & 7 & 11 & 13 \end{array}\right] \end{array}$$

e no fim do processo, isto é, no fim do ciclo $i = m$, obtém-se os vectores da estrutura de dados que representa a matriz A^T por linhas e que são dados por

$$\begin{array}{rcl} \text{VALAT} & = & \left[\begin{array}{ccccccccccccc} a_{21} & a_{41} & a_{52} & a_{13} & a_{33} & a_{43} & a_{24} & a_{34} & a_{44} & a_{15} & a_{16} & a_{56} \end{array}\right] \\ \text{INDAT} & = & \left[\begin{array}{ccccccccccccc} 2 & 4 & 5 & 1 & 3 & 4 & 2 & 3 & 4 & 1 & 5 & 1 & 5 \end{array}\right] \\ \text{PNTAT} & = & \left[\begin{array}{ccccccc} 1 & 3 & 4 & 7 & 10 & 12 & 14 \end{array}\right] \end{array}$$

(vii) Resolução de sistemas triangulares com matrizes esparsas

Consideremos o sistema triangular inferior

$$Lx = b \tag{6.1}$$

com L uma matriz de ordem n armazenada por um esquema de colecção de vectores. Então uma versão por linhas ou colunas do algoritmo para matrizes triangulares inferiores deve ser escolhida para a resolução do sistema (6.1) dependendo do facto de L estar armazenada por linhas ou por colunas. Esses algoritmos podem ser facilmente implementados para uma estrutura de dados de colecção de vectores com ou sem diagonal separada. Assim, se L está armazenada por linhas com diagonal separada numa estrutura de dados constituída pelos vectores VALL, INDL, PNTL e DIAGL, então o algoritmo para a resolução do sistema (6.1) tem a seguinte forma

Algoritmo 36

$b(1) = \frac{b(1)}{\text{DIAG}(1)}$

Para $t = 2, \ldots, n$

  prod $= 0$

  inicio $= \text{PNTL}(t)$

  fim $= \text{PNTL}(t+1) - 1$

  Se fim $\geq$ inicio faça

    Para $k =$ inicio$, \ldots,$ fim

      prod $=$ prod $+ \text{VALL}(k) \times b(\text{INDL}(k))$

  $b(t) = \frac{b(t) - \text{prod}}{\text{DIAGL}(t)}$

Por outro lado, se L está armazenada por colunas com diagonal separada numa estrutura de dados constituída pelos vectores VALL, INDL, PNTL, DIAGL, então o algoritmo tem os seguintes passos

Algoritmo 37

Para $t = 1, \ldots, n-1$

  $b(t) = \frac{b(t)}{\text{DIAGL}(t)}$

  inicio $= \text{PNTL}(t)$

  fim $= \text{PNTL}(t+1) - 1$

  Se fim $\geq$ inicio *faça*

    Para $k =$ inicio$, \ldots,$ fim

      $b(\text{INDL}(k)) = b(\text{INDL}(k)) - b(t) \times \text{VALL}(k)$

$b(n) = \dfrac{b(n)}{\text{DIAGL}(n)}$

Consideremos agora um sistema triangular superior

$$Ux = b$$

com U uma matriz de ordem n armazenada por um esquema de colecção de vectores. Se U está armazenada por linhas, então o algoritmo 2 (TRIANSUP) pode ser facilmente implementado para qualquer tipo de armazenagem, ordenada ou não, com ou sem diagonal separada. Se U está armazenada por colunas, então o algoritmo orientado por colunas para uma matriz triangular superior deve ser usado.

Antes de terminar esta secção, vamos calcular o número de operações necessárias para resolver um sistema triangular. Para isso, seja η o número de elementos não nulos não diagonais da matriz L ou U. Então facilmente se conclui que o número de operações é dado por

$$\begin{cases} \text{número de adições} & = & \eta \\ \text{número de multiplicações} & = & \eta \\ \text{número de divisões} & = & n \end{cases} \tag{6.2}$$

Portanto o número de operações para resolver um sistema triangular é igual a $n + \eta$. Esse número é bastante menor que $\frac{n^2}{2}$ quando n é elevado e a matriz é muito esparsa, isto é, η é pequeno. Isso mostra a importância da exploração da esparsidade das matrizes na resolução de sistemas de grandes dimensões.

Exercícios

1. Armazene as seguintes matrizes por linhas e por colunas no esquema de colecção de vectores que achar mais apropriado:

$$A_1 = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 \\ 0 & -1 & 2 & 0 & -1 \\ -1 & 2 & 3 & 0 & 2 \\ 0 & 0 & 1 & 0 & -1 \\ -1 & 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & -2 & -3 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 \\ -3 & 0 & -1 & 1 & -1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 2 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 0 & -1 & 2 & 0 \\ 1 & 0 & 1 & 0 \\ 2 & -1 & 0 & 1 \\ 0 & -1 & 2 & 3 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 1 & 0 & -1 & 2 & 0 \\ 0 & 2 & 0 & 0 & -1 \\ -1 & 0 & 3 & 1 & 2 \\ -1 & 0 & 2 & 4 & 0 \\ 4 & 0 & 1 & 5 & -5 \end{bmatrix}$$

2. Armazene como matrizes densas as seguintes matrizes armazenadas por linhas nos esquemas de colecção de vectores:

   (a)

VALA	1	2	-1	3	1	4	5	6

INDA	1	2	2	3	1	1	2	4

PNTA	1	3	5	6	9

   (b)

DIAG	1	2	1	3	5	6	
VALA	-1	2	3	1	-2	4	5
INDA	1	1	3	2	4	3	5
PNTA	1	1	2	3	4	6	8

(A simétrica)

(c)

DIAG	1	-2	3	-4	5		
VALA	1	2	-1	2	1	2	3
INDA	2	3	5	1	1	3	5
PNTA	1	2	4	4	5	8	

3. Desenvolva um algoritmo para a adição de duas matrizes esparsas SPD armazenadas por colunas em esquemas de colecção de vectores.

4. Desenvolva um algoritmo para a multiplicação de uma matriz esparsa simétrica armazenada por colunas num esquema de colecção de vectores por um vector denso.

5. Desenvolva um algoritmo para o cálculo de $b^T A$ com b um vector coluna denso de ordem n e $A \in \mathbb{R}^{n \times n}$ uma matriz esparsa não simétrica armazenada por colunas num esquema de colecção de vectores.

6. Seja A uma matriz quadrada de ordem n não simétrica esparsa, armazenada segundo um esquema de colecção de vectores. Para x e y vectores esparsos de ordem n, descreva o processo que permite calcular o produto $x^T Ay$.

7. Desenvolva um algoritmo para a resolução de um sistema com uma matriz triangular superior esparsa armazenada por linhas num esquema de colecção de vectores.

8. Desenvolva um algoritmo para calcular o produto de duas matrizes quadradas esparsas não simétricas armazenadas por linhas num esquema de colecção de vectores.

9. Desenvolva um algoritmo para implementar o cálculo do produto $ABA^T b$, com A uma matriz esparsa de ordem $m \times n$, B uma matriz tridiagonal de ordem n e b um vector denso de ordem m.

10. Desenvolva um algoritmo para calcular a norma ℓ_1 de uma matriz rectangular armazenada por colunas num esquema de colecção de vectores.

11. Desenvolva um algoritmo para calcular a norma ℓ_∞ de uma matriz SPD armazenada por linhas num esquema de colecção de vectores.

12. Desenvolva uma implementação do algoritmo LINPACK para determinação de uma estimativa do número de condição de uma matriz triangular inferior.

Capítulo 7

Métodos Directos para Sistemas de Equações Lineares com Matrizes Esparsas

Consideremos o sistema de equações lineares

$$Ax = b \tag{7.1}$$

onde A é uma matriz esparsa não singular de ordem n. Conforme vimos anteriormente, três casos distintos devem ser considerados, nomeadamente a matriz A ser simétrica positiva definida, não simétrica com pivots diagonais estáveis e simétrica ou não simétrica com pivots diagonais instáveis. Nos dois primeiros casos os pivots para a obtenção das decomposições LDL^T ou LU podem sempre ser escolhidos na diagonal das sucessivas matrizes reduzidas, enquanto que no último caso pode ser necessário recorrer a trocas de linhas ou colunas para preservar a estabilidade do processo.

Para matrizes esparsas, para além da estabilidade da factorização há que tentar manter no máximo a esparsidade das matrizes que vão sendo construídas durante o processo. Assim, associado ao processo de factorização está o chamado fenómeno do Enchimento (ou *Fill–in*) da matriz provocado pelos elementos nulos de A que se vão transformar em elementos não nulos durante o processo de factorização. Se FILL(A) representar o conjunto de tais elementos, tem-se

$$\text{FILL}(A) = \{(i,j) : a_{ij} = 0 \text{ e } (L+U)_{ij} \neq 0\} \tag{7.2}$$

O processo de factorização da matriz A deve ser levado a cabo de modo a reduzir o máximo possível o número de elementos de FILL(A). Com efeito, quanto maior for o número de elementos desse conjunto, maior vai ser o espaço de armazenagem necessário para armazenar as matrizes L e U e maior é o número de operações necessárias para obter essas matrizes. Para esse fim o processo de factorização incorpora permutações de linhas e de colunas que tentam minimizar o número de elementos desse conjunto. Para ilustrar a importância da necessidade de permutar linhas e colunas para reduzir o número de elementos de FILL(A), consideremos a matriz:

$$A = \begin{bmatrix} 4. & -1. & -1. & -2. \\ -1. & 1. & & \\ -1. & & 1. & \\ -1. & & & 1. \end{bmatrix}$$

Se procurarmos obter a decomposição de A pela ordem natural, iremos obter uma matriz $L+U$ completamente cheia. Contudo, se trocarmos a ordem das linhas e colunas e seguirmos a ordem $(2,3,4,1)$, então a matriz $L+U$ tem o mesmo número de elementos não nulos que a matriz A e portanto FILL(A) $= \emptyset$.

Tal como para a resolução de sistemas de equações lineares com matrizes densas, três casos distintos devem ser considerados na resolução de sistemas com matrizes esparsas. Se a matriz do sistema (7.1) é simétrica positiva definida, então não há problemas de estabilidade se a factorização for obtida escolhendo os pivots na diagonal das sucessivas matrizes reduzidas. Portanto há a possibilidade de efectuar permutações principais de linhas e colunas de modo a escolher os pivots por critérios de esparsidade. O processo de factorização é normalmente iniciado com a consideração do grafo associado à matriz. Algoritmos combinatórios são a seguir aplicados de modo a determinar uma ordem das linhas e colunas da matriz segundo a qual os pivots devem ser escolhidos. Além disso, é possível determinar as posições onde os elementos de FILL(A) ocorrem e assim construir uma estrutura de dados que armazene todos os elementos não nulos de A e os elementos nulos correspondentes ao conjunto FILL(A) segundo a ordem obtida. O processo de factorização é a seguir aplicado e consiste em apenas modificar os elementos do vector VALA da estrutura de dados respeitante aos valores numéricos. Portanto a estrutura de dados não é modificada no decurso da factorização e por isso o processo diz-se Estático. De notar ainda que esse processo permite fixar o número de total de elementos a armazenar para a obtenção da decomposição e consequente resolução dos sistemas triangulares.

Se A é não simétrica com pivots diagonais estáveis, então também não há problemas de estabilidade na escolha dos pivots na diagonal. O processo para matrizes simétricas positivas definidas pode então ser estendido a este caso, obtendo-se assim um processo estático para a resolução do sistema (7.1).

Se a matriz A não pertence às duas classes anteriores, então os pivots para a obtenção da decomposição LU têm de ser escolhidos de acordo com algum critério de preservação da estabilidade. Contudo a necessidade de redução do número de elementos de FILL(A) implica o uso de critérios menos rigorosos do que a escolha parcial de pivot. Para esses critérios serem aplicados há a necessidade de saber em cada iteração os valores numéricos da correspondente matriz reduzida. Como na determinação desses elementos há sempre enchimentos, então a estrutura de dados vai sendo sucessivamente modificada. O processo é por isso Dinâmico, e não há possibilidade de prever o número total de elementos a armazenar.

Nas próximas secções iremos apresentar métodos para resolver o sistema (7.1) em cada um dos três casos referidos, apresentando as vantagens e inconvenientes de cada um dos métodos.

7.1 Matrizes simétricas positivas definidas

A resolução do sistema (7.1) com A uma matriz simétrica positiva definida consiste de três fases a seguir apresentadas.

1 - Análise: Nesta fase pretende-se determinar uma ordem para as linhas e colunas de modo a que os enchimentos sejam os menores possíveis. Para esse fim considera-se o grafo associado à matriz A e procura-se uma reordenação dos seus nós. Essa reordenação irá dar a ordem segundo a qual a factorização deverá ter lugar. Na prática, um vector *perm* é construído, tal que

$$perm(i) = j$$

indica que o elemento diagonal da linha e coluna j deverá ser o i–ésimo pivot da decomposição.

Nesta fase é ainda construído um vector *invp* que define a permutação inversa de *perm* e portanto satisfaz

$$invp(perm(i)) = i, \quad i = 1, 2, \ldots, n \tag{7.3}$$

Após a determinação da ordem das linhas e colunas, é possível construir uma estrutura de dados (colecção de vectores com diagonal separada) que inicialmente armazena os elementos não nulos de A segundo a ordem obtida e os elementos nulos respeitantes aos enchimentos que se irão verificar no processo de factorização. Essa construção usa a matriz original e o vector *invp* definido por (7.3).

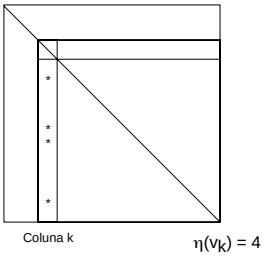


Figura 7.1: Representação esquemática de $\eta(v_k)$

2 - Factorização: Consiste na obtenção das matrizes L e D da decomposição LDL^T da matriz permutada P^TAP. Na prática, isto é conseguido modificando apenas os valores do vector VALA que armazena os elementos não nulos e alguns elementos nulos da matriz P^TAP.

3 - Solução: Consiste na resolução de dois sistemas triangulares e no cálculo do vector $D^{-1}y$ de acordo com o processo anteriormente descrito e usando as estruturas de dados das matrizes L e D. A solução obtida $\bar{b}$ tem as componentes segundo a ordem definida pelo vector $perm$, pelo que a solução $\bar{x}$ de (7.1) satisfaz

$$\bar{x}(perm(i)) = \bar{b}(i), \quad i = 1, 2, \ldots, n \tag{7.4}$$

Como a estrutura de dados que armazena a matriz permutada é orientada por linhas ou colunas, então o método dos bordos ou o método directo são usados para obter as matrizes L e D da decomposição LDL^T. Esses métodos requerem o mesmo número de operações para determinar essas matrizes. Para calcular esse número, seja

$$\eta(v_k) \quad = \quad \text{número de elementos não nulos abaixo da diagonal da coluna } k \text{ da matriz } A^{(k)} \text{ da iteração } k.$$

Uma melhor compreensão de $\eta(v_k)$ é proporcionada pela figura 7.1. Então tem-se

$$\begin{cases} \text{número de multiplicações / divisões:} & \displaystyle\sum_{k=1}^{n-1} \frac{\eta(v_k)\,(\eta(v_k)+5)}{2} \\ \text{número de adições:} & \displaystyle\sum_{k=1}^{n-1} \frac{\eta(v_k)\,(\eta(v_k)+1)}{2} \end{cases} \tag{7.5}$$

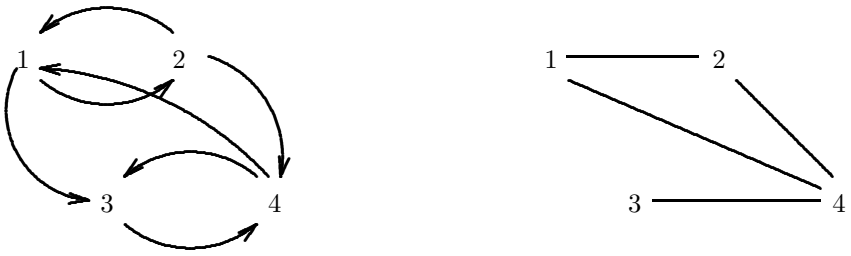
Como referimos anteriormente, a primeira fase da resolução do sistema $Ax = b$ é baseada na consideração do grafo associado à matriz A e em algoritmos combinatórios que operam com esse grafo. Por isso precisamos de algumas noções de teoria de grafos que introduzimos a seguir.

Um Grafo $G = (X, E)$ consiste de um conjunto finito X de nós ou vértices e de um conjunto finito E de arestas ou arcos, que são pares ordenados ou conjuntos de dois nós. Um grafo diz-se orientado ou não orientado consoante as arestas tenham ou não orientação. Assim, por exemplo, a figura 7.2 apresenta dois grafos, em que o primeiro é orientado e o segundo não é.

Uma Ordenação do grafo $G = (X, E)$ é uma aplicação

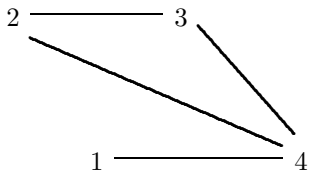
$$\begin{array}{rcl} \alpha : \{1, 2, \ldots, n\} & \rightarrow & X \\ k & \mapsto & \alpha(k) \end{array}$$

onde n é o número de nós. Facilmente se conclui que um mesmo grafo pode ter diversas ordenações. Assim, por exemplo, consideremos o grafo



Grafo orientado Grafo não orientado

Figura 7.2: Ilustração de um grafo orientado e de um grafo não orientado



Este grafo difere do segundo grafo da figura 7.2 apenas na ordenação dos seus nós. Um grafo ordenado por uma ordenação α é representado por $G^\alpha = (X^\alpha, E)$.

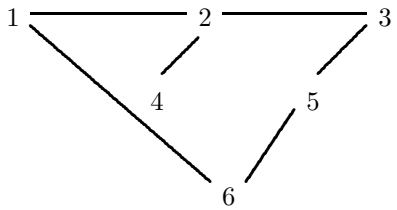
Dada uma matriz simétrica esparsa de ordem n, podemos-lhe associar um grafo de n nós numerados de 1 a n de tal modo que

$$\{x_i, x_j\} \in G^\alpha \Leftrightarrow a_{ij} = a_{ji} \neq 0$$

É evidente que o grafo associado a uma matriz simétrica é não orientado. Assim, por exemplo, se considerarmos a matriz

$$A = \begin{bmatrix} 1 & * & & & & * \\ * & 2 & * & * & & \\ & * & 3 & & * & \\ & * & & 4 & & \\ & & * & & 5 & * \\ * & & & & * & 6 \end{bmatrix}$$

onde $*$ representa um elemento diferente de zero, então o seu grafo pode ser representado da seguinte forma

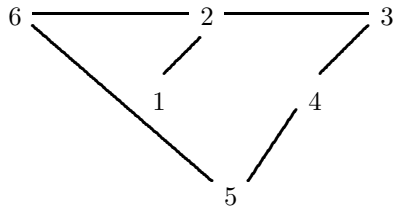


Para uma qualquer matriz de permutação P os grafos das matrizes A e $P^T AP$ diferem apenas na ordenação dos seus nós. Assim, por exemplo, se efectuarmos uma permutação principal das

linhas e colunas de A de modo a que

$$P^T AP = \begin{bmatrix} 1 & * & & & & \\ * & 2 & * & & & * \\ & * & 3 & * & & \\ & & * & 4 & * & \\ & & & * & 5 & * \\ & * & & & * & 6 \end{bmatrix}$$

então apenas a ordenação do grafo anterior é modificada, obtendo-se o seguinte grafo não orientado associado a $P^T AP$:



Para o estudo que se pretende fazer nesta secção são necessárias as noções de Nós Adjacentes e de Grau de um Nó. Dois nós x e y são Adjacentes se estiverem ligados por uma aresta, isto é, se $\{x, y\} \in E$. Dado um nó $y \in X$, o Conjunto Adjacente do Nó $y \in X$ é denotado por $Adj(y)$ e é definido por

$$Adj(y) = \{x \in X : \{x, y\} \in E\} \tag{7.6}$$

isto é, é o conjunto dos nós adjacentes a y. O Grau do nó $y \in X$ é o número de elementos de $Adj(y)$ e representa-se por $deg(y)$. Assim, por exemplo, no grafo anterior tem-se

$$deg(x_1) = 1, deg(x_2) = 3, deg(x_3) = 2, \ldots, deg(x_6) = 2$$

Se G^α é o grafo associado a uma matriz simétrica esparsa, então $deg(x_k)$ é o número de elementos não nulos não diagonais na linha (ou coluna) k, onde se assume que a ordem das linhas e das colunas é dada pela ordenação α. Esta propriedade será usada no desenvolvimento de um algoritmo para obter uma ordenação das linhas e das colunas da matriz A, como será referido mais adiante.

Para implementar a estrutura de dados representativa de um grafo ordenado basta considerar dois vectores de inteiros XADJ e ADJNCY definidos do seguinte modo:

ADJNCY: contém todos os nós adjacentes dos nós $x_1, x_2, \ldots, x_n$ segundo essa ordem e portanto tem dimensão $2|E|$, com $|E|$ o número de arestas do grafo.

XADJ: é um vector de dimensão $n + 1$ tal que XADJ(k) representa o primeiro nó adjacente de x_k na lista de adjacentes ADJNCY. Além disso,

$$\text{XADJ}(k + 1) - \text{XADJ}(k) = deg(x_k).$$

Nesta secção iremos estudar dois processos para a determinação de uma ordem segundo a qual a factorização se deve efectuar de modo a que haja poucos enchimentos no processo de factorização. Esses algoritmos são denominados Método do Grau Mínimo e Método do Invólucro e baseiam-se fundamentalmente na noção de grau de um nó de um grafo não orientado.

(i) Algoritmo do Grau Mínimo

Das fórmulas (7.5) conclui-se imediatamente que o número de operações necessárias para obter a decomposição LDL^T de uma matriz A está dependente dos números $\eta(v_k)$ de elementos não nulos abaixo da diagonal da coluna k da matriz $A^{(k)}$ da iteração k e será tanto menor quanto menores

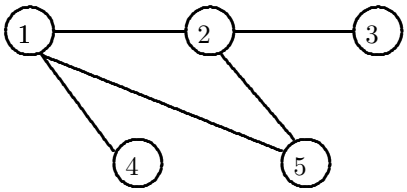
forem esses números. Assim, uma boa estratégia para obter uma ordenação para as linhas e colunas da matriz A será escolher em cada iteração k a coluna k para a qual $\eta(v_k)$ é mínimo. Para implementar esse algoritmo, seja $A^{(k)}$ a matriz transformada de A na iteração k e consideremos as matrizes Complementos de Schur $\bar{A}^{(k)}$ definidas por

$$\bar{A}^{(1)} = A^{(1)} = A, \quad \bar{A}^{(k+1)} = \left(\bar{A}^{(k)} | a_{kk}^{(k)}\right), \quad k = 1, 2, \ldots, n-1 \tag{7.7}$$

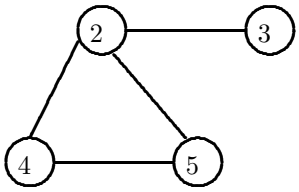
Seja ainda $G^{(k)}$ o grafo associado à matriz $\bar{A}^{(k)}$. Da definição de Complemento de Schur de $\left[a_{kk}^{(k)}\right]$ em $A^{(k)}$ facilmente se conclui que o grafo $G^{(k+1)}$ associado à matriz $\bar{A}^{(k+1)}$ se pode obter do grafo associado à matriz $\bar{A}^{(k)}$ a partir da seguinte regra:

Regra 1: Para obter $G^{(k+1)}$ a partir de $G^{(k)}$ suprimir o nó x_k de $G^{(k)}$ e todas as arestas que unem x_k com outros nós e acrescentar arestas ligando os nós que eram adjacentes a x_k sem ser adjacentes entre si.

Assim, por exemplo, considerando a matriz $A = A^{(1)} = \bar{A}^{(1)}$ e o seu grafo $G^{(1)}$ dados por

$$\bar{A}^{(1)} = \begin{bmatrix} 1 & * & & * & * \\ * & 2 & * & & * \\ & * & 3 & & \\ * & & & 4 & \\ * & * & & & 5 \end{bmatrix} \qquad G^{(1)} = \quad \text{  } \quad \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

então a matriz $\bar{A}^{(2)} = \left(\bar{A}^{(1)} | a_{11}^{(1)}\right)$ e o seu grafo associado $G^{(2)}$ são dados por

$$\bar{A}^{(2)} = \begin{bmatrix} 2 & * & \otimes & * \\ * & 3 & & \\ \otimes & & 4 & \otimes \\ * & & \otimes & 5 \end{bmatrix} \qquad G^{(2)} = \quad \boxed{1} \quad \text{  } \quad \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

De notar que houve dois enchimentos no cálculo de $A^{(2)}$ e duas novas arestas $\{2,4\}$ e $\{4,5\}$ foram criadas para obter o grafo $G^{(2)}$.

Como $G^{(k)}$ é o grafo associado à matriz $\bar{A}^{(k)}$, então $\eta(v_k)$ representa o grau do nó x_k do grafo $G^{(k)}$ e portanto a estratégia referida anteriormente consiste em escolher em cada iteração k o nó de $G^{(k)}$ que tem grau mínimo. Daí chamar-se Algoritmo do Grau Mínimo a esse procedimento. Se x_t é o nó escolhido então $perm(k) = t$ e o grafo $G^{(k+1)}$ é obtido usando a regra 1 e o processo é repetido. Em termos computacionais a determinação de $G^{(k+1)}$ é feita modificando a estrutura de dados que representa o grafo $G^{(k)}$. Essa modificação é baseada nas seguintes alterações dos conjuntos adjacentes $Adj(x_j)$:

$$\begin{aligned} &Adj(x_j) = Adj(x_j) \ \ se \ \ x_j \notin Adj(x_t) \\ &Adj(x_j) = Adj(x_j) - \{x_t\} \cup \mathrm{FILL}(t,j) \ \ se \ \ x_j \in Adj(x_t) \end{aligned} \tag{7.8}$$

onde

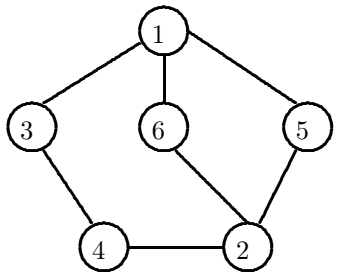
$$\mathrm{FILL}(t,j) = \{x_i : i \neq t, x_i \in Adj(x_t), x_i \notin Adj(x_j)\}$$

Para ilustrar o algoritmo consideremos a seguinte matriz:

$$A = \begin{bmatrix} 1 & & * & & * & * \\ & 2 & & * & * & * \\ * & & 3 & * & & \\ & * & * & 4 & & \\ * & * & & & 5 & \\ * & * & & & & 6 \end{bmatrix} \tag{7.9}$$

O seu grafo associado tem a forma:

$G^{(1)} =$

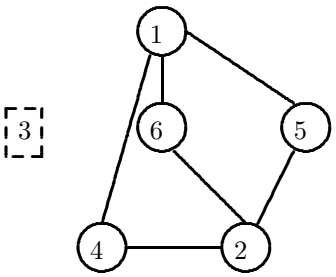


Os graus dos nós de $G^{(1)}$ vêm então dados pela seguinte tabela:

Nós	x_1	x_2	x_3	x_4	x_5	x_6
Graus	3	3	2	2	2	2

Portanto há quatro possíveis nós a escolher. Escolhendo x_3 vem $perm(1) = 3$ e o grafo $G^{(2)}$ e a tabela dos graus dos nós desse grafo têm a seguinte forma

$G^{(2)} =$



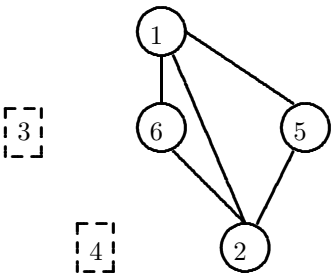
Nós	x_1	x_2	x_4	x_5	x_6
Graus	3	3	2	2	2

Além disso houve um enchimento $\{1, 4\}$. Se representarmos por FILL o conjunto de todos os enchimentos que a factorização provoca usando a ordem que for obtida pelo algoritmo do grau mínimo, então no fim da primeira iteração deste algoritmo temos

$$\text{FILL} = \{\{1, 4\}\}$$

Agora $perm(2) = 4$ e o grafo $G^{(3)}$ e a tabela dos graus dos nós desse grafo são os seguintes:

$G^{(3)} =$

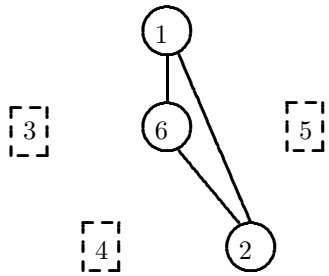


Nós	x_1	x_2	x_5	x_6
Graus	3	3	2	2

Além disso

$$\text{FILL} = \text{FILL} \cup \{1,2\} = \{\{1,4\},\{1,2\}\}$$

Na terceira iteração tem-se $perm(3) = 5$ e o grafo $G^{(4)}$ tem a seguinte forma:

$G^{(4)} =$ 

Neste caso não há lugar a qualquer enchimento. Todos os nós têm grau 2, o que torna a escolha dos últimos três nós completamente indiferente. Donde podemos escrever

$$perm(4) = 1, perm(5) = 2, perm(6) = 6$$

não havendo qualquer enchimento.

A matriz permutada é

$$P^T AP = \begin{bmatrix} 3 & * & & * & & \\ * & 4 & & 0 & * & \\ & & 5 & * & * & \\ * & 0 & * & 1 & 0 & * \\ & * & * & 0 & 2 & * \\ & & & * & * & 6 \end{bmatrix}$$

onde os elementos nulos considerados são referentes aos enchimentos que o processo de factorização provocará, sendo portanto necessário armazená-los na estrutura de dados orientada por colunas, cuja construção é o último passo da Fase de Análise. Se a_{ij} representarem os valores numéricos dos elementos não nulos da matriz original A, então a estrutura de dados que armazena a matriz permutada tem a forma seguinte:

$$\begin{array}{lcl} \text{DIAG} & : & \left[\begin{array}{cccccc} a_{33} & a_{44} & a_{55} & a_{11} & a_{22} & a_{66} \end{array} \right] \\ \text{VALA} & : & \left[\begin{array}{cccccccccc} a_{43} & a_{13} & 0 & a_{24} & a_{15} & a_{25} & 0 & a_{61} & a_{62} \end{array} \right] \\ \text{INDA} & : & \left[\begin{array}{cccccccccc} 2 & 4 & 4 & 5 & 4 & 5 & 5 & 6 & 6 \end{array} \right] \\ \text{PNTA} & : & \left[\begin{array}{cccccccc} 1 & 3 & 5 & 7 & 9 & 10 & 10 \end{array} \right] \end{array}$$

De notar que o espaço de armazenagem total é igual a 31. Como referimos anteriormente, a Fase de Factorização teria agora início e consistiria em ir modificando os elementos dos vectores DIAG e VALA segundo o método directo. Após essas transformações o vector DIAG apresenta a diagonal da matriz D da decomposição LDL^T da matriz permutada e os vectores VALA, INDA e PNTA constituem a estrutura de dados orientada por colunas da matriz L (os elementos diagonais de L são iguais a um e por isso não precisam de ser armazenados). A Fase de Solução pode então ser aplicada, usando a estrutura de dados referida e consiste dos seguintes passos:

1. Resolução do sistema $Ly = b$, usando uma implementação do algoritmo para a resolução de sistemas triangulares inferiores por colunas.

2. Cálculo do vector $y = D^{-1}y$ usando o vector DIAG.

3. Resolução do sistema $L^T x = y$ usando uma implementação do algoritmo para a resolução de sistemas triangulares superiores por linhas.

O processo acabado de descrever é talvez o algoritmo mais eficiente na determinação de uma ordem para as linhas e colunas de A que reduza o número de enchimentos o máximo possível. Contudo, a sua aplicação requer em cada iteração a lista de adjacências do grafo $G^{(k)}$ referido anteriormente, o que pode tornar o processo de ordenação bastante demorado. Existem contudo programas que contém implementações bastante eficientes do algoritmo do grau mínimo, como a MA27 [Duff *et al*] e a SPARSPAK [George e Liu].

(ii) O Método do Invólucro

Nesta secção iremos apresentar um processo que pode ser visto como uma extensão das matrizes com estrutura de banda, mas que usa estruturas de dados semelhantes às do método do grau mínimo. Considerando as quantidades $f_i(A)$ e $\beta_i(A)$ definidas no capítulo 5, chama-se Invólucro de A, e representa-se por $\mathrm{ENV}(A)$, o conjunto definido por

$$\mathrm{ENV}(A) = \{\{i,j\} : \ f_i(A) \leq j < i\} \tag{7.10}$$

Assim, por exemplo, para a matriz (7.9), temos

$$\mathrm{ENV}(A) = \left\{ \begin{array}{l} \{3,1\},\{3,2\},\{4,2\},\{4,3\},\{5,1\},\{5,2\} \\ \{5,3\},\{5,4\},\{6,1\},\{6,2\},\{6,3\},\{6,4\},\{6,5\} \end{array} \right\}$$

Facilmente se conclui que para uma matriz A simétrica de ordem n se tem

$$|\mathrm{ENV}(A)| = \sum_{i=1}^{n} \beta_i(A) \tag{7.11}$$

e

$$\mathrm{ENV}(A) \cup \mathrm{DIAG}(A) \subseteq \mathrm{BAND}(A) \tag{7.12}$$

onde $\mathrm{DIAG}(A) = \{\{i,i\} : \ i = 1,\ldots,n\}$, $\mathrm{BAND}(A)$ foi definido no capítulo 5 e $|\mathrm{ENV}(A)|$ é o número de elementos de $\mathrm{ENV}(A)$.

Tal como para as matrizes com estrutura de banda, a grande vantagem da consideração do invólucro de uma matriz simétrica reside no facto de o invólucro não ser alterado durante o processo da decomposição LDL^T duma matriz simétrica, se os pivots das sucessivas iterações forem sempre escolhidos na diagonal. Como as matrizes SPD satisfazem essa pretensão podemos enunciar o seguinte teorema.

Teorema 7.1 *Se* $A \in \mathrm{SPD}$*, então* $\mathrm{ENV}(A) = \mathrm{ENV}(L + D + L^T)$.

Demonstração: Sejam $A \in \mathrm{SPD}$ e $A^{(2)}$ a matriz obtida após o primeiro passo da decomposição LDL^T de A. Então demonstrar que $\mathrm{ENV}(A^{(2)}) = \mathrm{ENV}(A)$ é equivalente a mostrar que para todo $i = 1, 2, \ldots, n$, se tem

$$a_{ij}^{(2)} = 0 \quad para \quad i > j + \beta_i(A) \tag{7.13}$$

Dois casos podem acontecer e são apresentados a seguir:

1. Se $j = 1$, então $a_{ij}^{(2)} = \frac{a_{ij}}{a_{11}}$. Mas, por definição de $\mathrm{ENV}(A)$ tem-se

$$a_{ij} = 0 \quad para \quad i > j + \beta_i(A) \tag{7.14}$$

   e portanto (7.13) é verdadeira para $j = 1$.

2. Se $j > 1$, então $a_{ij}^{(2)} = a_{ij}$ para $i > j + \beta_i(A)$. Como (7.14) é verdadeira, e o mesmo acontece a (7.13).

Portanto $\mathrm{ENV}(A) = \mathrm{ENV}(A^{(2)})$ e o teorema fica demonstrado por indução.

Devido a este teorema, no processo de obtenção a decomposição LDL^T de A é apenas necessário considerar os elementos da matriz A correspondentes aos conjuntos $\mathrm{ENV}(A)$ e $\mathrm{DIAG}(A)$. Os elementos diagonais de A são armazenados num vector DIAG, onde

$$\mathrm{DIAG}(k) = a_{kk},\ k = 1, 2, \ldots, n$$

Os elementos correspondentes a $\mathrm{ENV}(A)$ são armazenados por linhas, numa estrutura de dados que consiste em dois vectores ENV e XENV definidos do seguinte modo:

1. ENV é um vector de números reais que armazena todos os elementos de $\mathrm{ENV}(A)$ por linhas. A dimensão desse vector é portanto $|\mathrm{ENV}(A)|$.
2. XENV é um vector de números inteiros de dimensão $n+1$ tal que $\mathrm{XENV}(k)$ representa a posição em ENV do primeiro elemento da linha k que foi armazenado e $\mathrm{XENV}(n+1) = |\mathrm{ENV}(A)| + 1$.

Este esquema de armazenagem acabado de descrever é chamado esquema de banda variável. A sua principal vantagem em relação à colecção de vectores reside no facto de utilizar menos um vector, com a desvantagem de ter que armazenar todos os elementos nulos correspondentes ao invólucro de A. Para ilustrar o esquema de banda variável, consideremos a matriz

$$A = \begin{bmatrix} a_{11} & a_{21} & & & \\ a_{21} & a_{22} & & a_{42} & a_{52} \\ & & a_{33} & & \\ & a_{42} & & a_{44} & a_{54} \\ & a_{52} & & a_{54} & a_{55} \end{bmatrix}$$

Então tem-se

$$\begin{aligned} \mathrm{DIAG} &= \begin{bmatrix} a_{11} & a_{22} & a_{33} & a_{44} & a_{55} \end{bmatrix} \\ \mathrm{ENV} &= \begin{bmatrix} a_{21} & a_{42} & 0. & a_{52} & 0. & a_{54} \end{bmatrix} \\ \mathrm{XENV} &= \begin{bmatrix} 1 & 1 & 2 & 2 & 4 & 7 \end{bmatrix} \end{aligned}$$

Como o esquema de banda variável é orientado por linhas, então o método dos bordos deve ser usado para a obtenção da decomposição LDL^T da matriz A. Além disso, na Fase de Solução, o sistema $Ly = b$ deve ser resolvido por um processo orientado por linhas, enquanto que um processo orientado por colunas deve ser usado na resolução de $L^T x = y$. Como apenas os elementos correspondentes ao invólucro são armazenados e são referenciados pela quantidade β_k, então os algoritmos para a resolução desses dois sistemas são modificados e apresentam as formas seguintes:

Algoritmo 38 (Algoritmo para a resolução do sistema $Ly = b$)

Para $k = 2, \ldots, n$

$$b_k = b_k - \sum_{j=k-\beta_k}^{k-1} l_{kj} b_j$$

Algoritmo 39 (Algoritmo para a resolução do sistema $L^T x = y$)

Para $k = n, n-1, \ldots, 2$

$$\begin{bmatrix} b_{k-\beta_k} \\ \vdots \\ b_{k-1} \end{bmatrix} = \begin{bmatrix} b_{k-\beta_k} \\ \vdots \\ b_{k-1} \end{bmatrix} - b_k \begin{bmatrix} l_{k,k-\beta_k} \\ \vdots \\ l_{k,k-1} \end{bmatrix}$$

De notar que são necessárias $|\mathrm{ENV}(L)| = \sum_{i=1}^{n} \beta_i(A)$ multiplicações e adições para resolver cada um dos sistemas referidos. A implementação desses algoritmos usando o esquema de banda variável é muito simples de fazer e é deixada como exercício.

Suponhamos agora que se pretende determinar as matrizes L e D da decomposição LDL^T da matriz A usando o método dos bordos. Este método consiste em resolver em cada iteração um sistema triangular inferior e calcular um elemento diagonal de D a partir de um produto escalar de dois vectores. Em relação ao sistema triangular, é possível reduzir altamente o esforço computacional se notarmos que apenas os elementos correspondentes ao invólucro de A têm que ser modificados. Para verificar isso, suponhamos que

$$A_{k-1,k-1} = L_{k-1} D_{k-1} L_{k-1}^T$$

e se pretende determinar o vector $\bar{a}$ tal que

$$L_k = \begin{bmatrix} L_{k-1} & \\ \bar{a}^T & 1 \end{bmatrix}$$

Se β_k é o comprimento de banda de ordem k, então

$$a^T = \begin{bmatrix} 0 & \ldots & 0 & a_{k-\beta_k} & \ldots & a_{k-1} \end{bmatrix} = \begin{bmatrix} 0 & | & a^2 \end{bmatrix}$$

Portanto

$$L_{k-1}\bar{a} = a \tag{7.15}$$

pode ser escrito na seguinte forma:

$$\begin{bmatrix} L_{k-\beta_k-1} & \\ E & \bar{L}_k \end{bmatrix} \begin{bmatrix} \bar{a}^1 \\ \bar{a}^2 \end{bmatrix} = \begin{bmatrix} 0 \\ a^2 \end{bmatrix}$$

Portanto $\bar{a}^1 = 0$ e o sistema (7.15) reduz-se à solução de

$$\bar{L}_k \bar{a}^2 = a^2 \tag{7.16}$$

cuja matriz é de ordem β_k.

Chegámos assim à conclusão que em cada iteração k do processo de factorização é necessário resolver um sistema triangular inferior cuja matriz é de ordem β_k. Além disso, para calcular o elemento diagonal d_{kk} da matriz diagonal D referente a essa iteração, temos de efectuar os seguintes passos:

Calcule $\bar{a}_i = \dfrac{\bar{a}_i}{d_{ii}}, i = k-\beta_k, \ldots, k-1$

$$d_{kk} = d_{kk} - \sum_{t=k-\beta_k}^{k-1} d_{tt} \bar{a}_t^2$$

Como o processo é repetido $n-1$ vezes para $k=2,\ldots,n$ e $\beta_1=0$, e tendo em conta que as matrizes $\bar{L}_k$ dos sistemas (7.16) são cheias, então o número de operações necessárias para obter a decomposição LDL^T da matriz A é dado por

$$\begin{cases} \text{Adições} & = & \displaystyle\sum_{k=1}^{n}\left[\frac{\beta_k(\beta_k-1)}{2}+\beta_k\right]=\sum_{k=1}^{n}\frac{\beta_k(\beta_k+1)}{2} \\ \text{Multiplicações} & = & \displaystyle\sum_{k=1}^{n}\left[\frac{\beta_k(\beta_k-1)}{2}+\beta_k+2\beta_k\right]=\sum_{k=1}^{n}\frac{\beta_k(\beta_k+5)}{2} \end{cases} \tag{7.17}$$

Das fórmulas (7.17) e do número de operações necessárias para resolver os sistemas triangulares e calcular o vector $D^{-1}y$ na Fase de Solução, imediatamente se conclui que o número de operações para resolver o sistema $Ax=b$ é dado por

$$\begin{aligned} \text{Adições} &= \sum_{k=1}^{n}\frac{\beta_k(\beta_k+5)}{2} \\ \text{Multiplicações} &= \sum_{k=1}^{n}\frac{\beta_k(\beta_k+9)}{2}+n \end{aligned} \tag{7.18}$$

O algoritmo do invólucro é tanto mais eficiente quanto menor for o número de elementos do invólucro da matriz A. Com efeito, não só o espaço de armazenagem depende desse valor, mas também o número de operações para resolver um sistema por este processo está extremamente dependente deste número. Por isso, e tal como anteriormente, há a necessidade de considerar primeiramente uma Fase de Análise na qual se encontra uma ordem definida por um vector *perm*, segundo a qual o número de elementos do invólucro da matriz permutada é bastante reduzido. O processo mais conhecido para encontrar essa ordem é denominado Método de Cuthill—McKee (Método CM) e baseia-se no facto de que o número de elementos no invólucro de uma matriz é habitualmente reduzido se nós adjacentes do grafo da matriz forem numerados com números cuja diferença seja relativamente pequena. A figura 7.3 ilustra esse facto.

Os passos do algoritmo CM são apresentados a seguir.

Algoritmo 40 (Algoritmo CM)

Passo 1: Considere um nó qualquer x_1.

Passo 2: Para $k=1,2,\ldots,n$, considere os nós adjacentes a x_k e enumere-os por ordem crescente dos seus graus.

É possível demonstrar que, se invertermos a ordem obtida pelo algoritmo CM, então o número de elementos do invólucro de A nunca aumenta. Além disso, a experiência tem mostrado que diminuições substanciais do invólucro podem ocorrer. O algoritmo assim obtido é denominado Algoritmo de Cuthill—McKee Inverso ou Algoritmo RCM e difere do Algoritmo CM na existência de um Passo 3 adicional, no qual a ordem obtida no Passo 2 é invertida, isto é

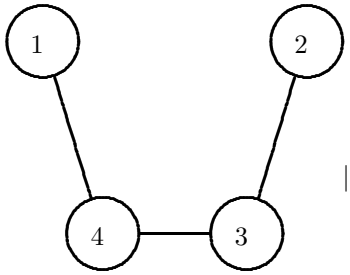
Passo 3: Enumere os nós na ordem $y_1,\ldots,y_n$ com

$$y_k = x_{n-k+1}, \quad k=1,2,\ldots,n$$

onde $x_1,\ldots,x_n$ é a ordem obtida no Passo 2.

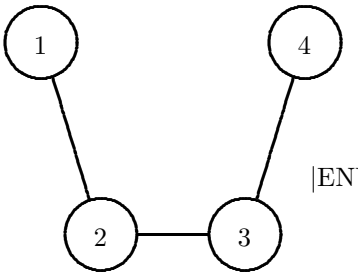
É de notar que ao minimizarem o invólucro da matriz A, estes processos também procuram reduzir a banda dessa matriz. Aliás estes algoritmos foram inicialmente propostos com esse objectivo.

$$A = \begin{bmatrix} 1 & * & & * \\ & 2 & * & \\ & * & 3 & * \\ * & & * & 4 \end{bmatrix}$$



$|\mathrm{ENV}(A)| = 4$

$$P^T AP = \begin{bmatrix} 1 & * & & \\ * & 2 & * & \\ & * & 3 & * \\ & & * & 4 \end{bmatrix}$$



$|\mathrm{ENV}(P^T AP)| = 3$

Figura 7.3: Efeito da reordenação das linhas e colunas de uma matriz sobre o número de elementos do seu invólucro

Para ilustrar o emprego dos algoritmos CM e RCM, consideremos a matriz (7.9), assim como o seu grafo e a sua lista de graus. Escolhamos o nó x_3 (nó de grau mínimo) para nó inicial. Então os nós x_4 e x_1 são adjacentes de x_3 , que são ordenados por esta forma em virtude de x_4 ter grau inferior a x_1. Agora x_4 é o próximo nó escolhido e como os seus nós adjacentes são x_2 e x_3, então x_2 é adicionado à lista já iniciada. O nó x_1 é a seguir escolhido e faz entrar na lista os nós x_5 e x_6, que podem ser colocados nesta ou noutra ordem por terem graus iguais. A lista está completa e tem a seguinte forma

$$x_3 \to x_4 \to x_1 \to x_2 \to x_5 \to x_6.$$

Portanto o vector *perm* que define a permutação obtida pelo algoritmo CM é dado por

$$perm = \begin{bmatrix} 3 & 4 & 1 & 2 & 5 & 6 \end{bmatrix}$$

e a matriz permutada tem a forma

$$P^T AP = \begin{bmatrix} 3 & * & & & & \\ * & 4 & & * & & \\ * & & 1 & & * & * \\ & * & & 2 & * & * \\ & & * & * & 5 & \\ & & * & * & & 6 \end{bmatrix}$$

De notar que $|\mathrm{ENV}(P^T AP)| = 10$ e $|\mathrm{ENV}(A)| = 13$, pelo que houve uma redução de 3

elementos na aplicação do algoritmo CM. O algoritmo RCM dá lugar à seguinte matriz:

$$\bar{P}^T A \bar{P} = \begin{bmatrix} 6 & & * & * & & \\ & 5 & * & * & & \\ * & * & 2 & & * & \\ * & * & & 1 & & * \\ & & * & & 4 & * \\ & & & * & * & 3 \end{bmatrix}$$

Portanto $|\mathrm{ENV}(\bar{P}^T A \bar{P})| = 9$, o que mostra que neste caso houve uma melhoria com o uso do algoritmo RCM.

Uma vez obtida a ordem é fácil determinar a estrutura de dados que armazena a matriz permutada segundo o esquema de banda variável. Se a_{ij} for a representação dos elementos não nulos da matriz original A, então os elementos são armazenados na ordem dada pelo algoritmo RCM na seguinte forma:

$$\begin{aligned} \mathrm{DIAG} &= \left[\begin{array}{cccccc} a_{66} & a_{55} & a_{22} & a_{11} & a_{44} & a_{33} \end{array} \right] \\ \mathrm{ENV} &= \left[\begin{array}{ccccccccc} a_{26} & a_{25} & a_{16} & a_{15} & 0. & a_{42} & 0. & a_{31} & a_{34} \end{array} \right] \\ \mathrm{XENV} &= \left[\begin{array}{cccccccc} 1 & 1 & 1 & 3 & 6 & 8 & 10 \end{array} \right] \end{aligned}$$

De notar que o espaço total de armazenagem é 22, o que é manifestamente inferior ao correspondente espaço para o algoritmo do grau mínimo. Por outro lado, o número de elementos nulos armazenados é o mesmo que para o algoritmo do grau mínimo, pelo que neste exemplo o uso do método do invólucro é preferível. No entanto, em geral, o algoritmo do grau mínimo é mais eficiente na redução dos enchimentos.

Uma vez obtida a estrutura de dados da matriz permutada, as fases de Factorização e Solução teriam agora lugar. A primeira consiste na modificação das componentes de DIAG e de ENV de acordo com o método dos bordos discutido nesta secção e a segunda consiste na resolução dos sistemas triangulares de acordo com os algoritmos 38 e 39 e no cálculo do vector $D^{-1}y$, usando o vector DIAG.

Antes de terminar esta secção gostariamos de salientar que a implementação do método do invólucro é bastante mais simples do que a do método do grau mínimo. O método do invólucro tem no entanto a desvantagem de não reduzir tanto os enchimentos como o método do grau mínimo. Um dos meios de melhorar a prestação do método do invólucro consiste numa escolha inteligente do nó inicial. Existem processos combinatórios que têm esse fim em vista, e aparecem descritos em [George e Liu]. Uma boa implementação desses processos aparece descrita nessa última referência.

7.2 Matrizes não simétricas com pivots diagonais estáveis

Se A é uma matriz deste tipo (diagonalmente dominante, H, K) então os pivots necessários para a obtenção da decomposição LU podem ser escolhidos nas diagonais das sucessivas matrizes reduzidas. Portanto é possível obter a decomposição LU da matriz A por um processo estático, que consiste das três fases, Análise, Factorização e Solução. Dois casos podem acontecer, que serão discutidos nesta secção.

(i) Matrizes simétricas de posição

A matriz $A = [a_{ij}]$ diz–se simétrica de posição se

$$a_{ij} \neq 0 \Leftrightarrow a_{ji} \neq 0$$

Portanto qualquer matriz A simétrica de posição tem associado um grafo não orientado. Usando a estrutura de adjacência deste grafo e os algoritmos do grau mínimo ou RCM podemos encontrar uma ordem para as linhas e colunas de A e construir a estrutura de dados da matriz permutada

segundo essa ordem. Como A é uma matriz não simétrica, então os elementos abaixo e acima da diagonal colocados simetricamente em relação à diagonal têm valores reais normalmente diferentes. Por isso a estrutura de dados da matriz permutada P^TAP tem de armazenar os elementos não nulos acima e abaixo da diagonal e os correspondentes aos enchimentos. Assim, por exemplo, suponhamos que após o uso do algoritmo do grau mínimo se obteve uma matriz permutada da forma

$$P^TAP = \begin{bmatrix} a_{11} & a_{12} & & a_{14} & & \\ a_{21} & a_{22} & & & & \\ & & a_{33} & & a_{35} & a_{36} \\ a_{41} & & & a_{44} & a_{45} & \\ & & a_{53} & a_{54} & a_{55} & \\ & & a_{63} & & & a_{66} \end{bmatrix}$$

e que

$$\text{FILL} = \{(4,2),(6,5),(2,4),(5,6)\}$$

Então a estrutura de dados para o processo de factorização tem a forma

$$\begin{aligned} \text{DIAG} &= \begin{bmatrix} a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \end{bmatrix} \\ \text{VALL} &= \begin{bmatrix} a_{21} & a_{41} & 0. & a_{53} & a_{63} & a_{54} & 0. \end{bmatrix} \\ \text{VALU} &= \begin{bmatrix} a_{12} & a_{14} & 0. & a_{35} & a_{36} & a_{45} & 0. \end{bmatrix} \\ \text{INDA} &= \begin{bmatrix} 2 & 4 & 4 & 5 & 6 & 5 & 6 \end{bmatrix} \\ \text{PNTA} &= \begin{bmatrix} 1 & 3 & 4 & 6 & 7 & 8 & 8 \end{bmatrix} \end{aligned}$$

Neste esquema, os elementos abaixo da diagonal são armazenados por colunas no vector VALL e os elementos acima da diagonal são armazenados por linhas usando o vector VALU. Para obter as matrizes L e U da decomposição LU de P^TAP, o método directo para matrizes não simétricas é usado. Esse processo consiste em transformar os elementos dos vectores DIAG, VALL e VALU. Estes vectores contêm no fim do processo de Factorização os elementos reais não nulos das matrizes L e U, sendo L armazenada por colunas e U por linhas. O vector DIAG contém os elementos diagonais da matriz U. O processo de Solução consiste em resolver o sistema $Ly = b$, usando um processo orientado por colunas, e o sistema $Ux = y$ usando um processo orientado por linhas.

Consideremos agora a aplicação do método do invólucro a matrizes deste tipo. Usando o algoritmo RCM aplicado ao grafo não orientado da matriz A, obtém-se uma matriz permutada, cuja estrutura de dados é a seguir construída. Assim, por exemplo, se a matriz permutada é dada por

$$P^TAP = \begin{bmatrix} a_{11} & a_{12} & & & & \\ a_{21} & a_{22} & & a_{24} & & \\ & & a_{33} & & & \\ & a_{42} & & a_{44} & a_{45} & a_{46} \\ & & & a_{54} & a_{55} & \\ & & & a_{64} & & a_{66} \end{bmatrix}$$

então a estrutura de dados para armazenar essa matriz tem a forma

$$\begin{aligned} \text{DIAG} &= \begin{bmatrix} a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \end{bmatrix} \\ \text{ENVR} &= \begin{bmatrix} a_{21} & a_{42} & 0. & a_{54} & a_{64} & 0. \end{bmatrix} \\ \text{ENVC} &= \begin{bmatrix} a_{12} & a_{24} & 0. & a_{45} & a_{46} & 0. \end{bmatrix} \\ \text{XENV} &= \begin{bmatrix} 1 & 1 & 2 & 2 & 4 & 5 & 7 \end{bmatrix} \end{aligned}$$

Segundo este esquema de banda variável os elementos do invólucro abaixo da diagonal são armazenados por linhas usando o vector ENVR e os elementos do invólucro acima da diagonal são armazenados por colunas usando o vector ENVC. O método dos bordos para matrizes não simétricas é usado no processo de Factorização para obter as matrizes L e U da decomposição LU

da matriz $P^T AP$. Esse processo consiste na modificação das componentes dos vectores DIAG, ENVR e ENVC. No fim do processo, os elementos reais correspondentes ao invólucro da matriz $P^T AP$ das matrizes L e U são armazenados nos vectores ENVR, ENVC e DIAG. De notar que o vector DIAG contém os elementos diagonais de U. A fase de Solução consiste em resolver o sistema $Ly = b$, usando um processo orientado por linhas e o sistema $Ux = y$ usando um processo orientado por colunas.

(ii) Matrizes não simétricas de posição

Se A é uma matriz não simétrica de posição, então o seu grafo associado é orientado e os algoritmos de ordenação referidos anteriormente não podem ser usados. Contudo, a matriz $A+A^T$ é simétrica e um processo de efectuar a Fase de Análise consiste em considerar o grafo associado a essa matriz e usar um dos algoritmos do grau mínimo ou RCM para obter uma ordem para a matriz A. A estrutura de dados para a matriz A é construída como no caso anterior, com a pequena diferença de haver mais elementos nulos a armazenar e que correspondem às posições (i, j) para as quais $a_{ij} = 0$ e $a_{ji} \neq 0$. Os processos de Factorização e Solução são semelhantes ao caso em que A é não simétrica com pivots diagonais estáveis mas simétrica de posição.

Para ilustrar este caso, consideremos a matriz

$$A = \begin{bmatrix} a_{11} & & a_{13} & \\ a_{21} & a_{22} & & a_{24} \\ & a_{32} & a_{33} & a_{34} \\ & & & a_{44} \end{bmatrix}$$

onde, tal como anteriormente, os espaços em branco representam elementos nulos. A estrutura da matriz $A + A^T$ tem a forma

$$A + A^T = \begin{bmatrix} 1 & * & * & \\ * & 2 & * & * \\ * & * & 3 & * \\ & * & * & 4 \end{bmatrix}$$

É evidente que não há hipótese de melhorar alguma coisa, pois a matriz $A + A^T$ é uma matriz banda com todos os elementos da banda diferentes de zero. Portanto a ordem original é a melhor e há que armazenar todos os elementos não nulos de A que correspondam aos elementos não nulos de $A + A^T$. Assim é necessário armazenar a matriz

$$A = \begin{bmatrix} a_{11} & 0 & a_{13} & \\ a_{21} & a_{22} & 0 & a_{24} \\ 0 & a_{32} & a_{33} & a_{34} \\ & 0 & 0 & a_{44} \end{bmatrix}$$

onde os elementos nulos são considerados como diferentes de zero. Usando, por exemplo, o esquema de banda variável, A é armazenada na forma

$$\begin{aligned} \text{DIAG} &= \begin{bmatrix} a_{11} & a_{22} & a_{33} & a_{44} \end{bmatrix} \\ \text{ENVR} &= \begin{bmatrix} a_{21} & 0. & a_{32} & 0. & 0. \end{bmatrix} \\ \text{ENVC} &= \begin{bmatrix} 0. & a_{13} & 0. & a_{24} & a_{34} \end{bmatrix} \\ \text{XENV} &= \begin{bmatrix} 1 & 1 & 2 & 4 & 6 \end{bmatrix} \end{aligned}$$

O método dos bordos para matrizes não simétricas seria agora usado na fase de Factorização e a fase de Solução é exactamente igual à referida no caso anterior de A ser simétrica de posição.

O processo apresentado tem a vantagem de ser estático, isto é, da estrutura de dados ser construída no fim da fase de Análise, o que torna bastante simples o processo de Factorização. Contudo só deve ser usado quando a matriz A for pouco simétrica de posição, isto é, quando

existirem poucos pares (i,j) tais que $a_{ij}=0$ e $a_{ji}\neq 0$. Com efeito, se tal não acontecer a estrutura de dados tem de armazenar demasiados elementos nulos, tornando-se o espaço de armazenagem bastante elevado. Isso é aliás possível de observar no exemplo apresentado, onde apenas 2 dos $4^2=16$ elementos da matriz não foram armazenados e metade dos elementos não diagonais armazenados são iguais a zero.

Podemos então concluir que se uma matriz não simétrica com pivots diagonais estáveis é simétrica de posição ou pouco não simétrica de posição, então é possível desenvolver um processo estático composto de três fases: Análise, Factorização e Solução. Se a matriz é muito não simétrica de posição, então é conveniente usar um processo dinâmico. Esse processo é discutido na próxima secção.

7.3 Matrizes não simétricas com pivots diagonais instáveis

Seja A uma matriz que não admite pivots diagonais estáveis. Se A for densa, a escolha parcial de pivot é recomendada e consiste em escolher na iteração k o elemento $a_{rk}^{(k)}$ que satisfaz o critério

$$|a_{rk}^{(k)}| = \max\{|a_{ik}^{(k)}| : i \text{ é linha da matriz reduzida } \bar{A}^{(k)}\} \tag{7.19}$$

onde $\bar{A}^{(k)}$ é a matriz definida por (7.7). Se A é uma matriz esparsa, então o processo de escolha parcial de pivot é demasiado restritivo. Com efeito, normalmente apenas um elemento satisfaz o critério (7.19) e se esse elemento pertence a uma linha com muitos elementos não nulos, então há muitos enchimentos se ele for escolhido para pivot. Assim, por exemplo, se considerarmos a matriz

$$A = \begin{bmatrix} 1.0 & 3.0 & & & \\ 2.0 & 1.2 & 2.0 & 2.0 & 3.0 \\ 0.15 & 0.01 & & & \\ 1.5 & & 0.1 & & \\ 1.1 & & & 0.1 & 1.0 \end{bmatrix} \tag{7.20}$$

então, usando a escolha parcial de pivot, a_{21} é o pivot da primeira iteração e é fácil de ver que a matriz reduzida $\bar{A}^{(2)}$ é completamente cheia.

A estabilidade do processo de escolha parcial de pivot decorre do facto de o factor de crescimento g_A ser limitado superiormente por 2^{n-1}, com n a ordem da matriz A. Este limite superior para g_A foi obtido considerando que o único elemento da matriz reduzida $\bar{A}^{(n-1)}$ é transformado $n-1$ vezes. Se A é uma matriz esparsa e o mesmo acontecer às sucessivas matrizes reduzidas, então a presença de elementos nulos na linha e na coluna do pivot faz com que poucos elementos sejam transformados em cada iteração. Como consequência, o factor de crescimento é limitado superiormente apenas por 2^{α}, com α o número máximo de vezes que um dado elemento da matriz é transformado. Portanto, se α for muito pequeno, não há grande problema em escolher o pivot por um critério menos forte do que a escolha parcial de pivot e que permita que a escolha seja feita tendo em linha de conta a preservação da esparsidade da matriz. Esta é a filosofia da chamada Escolha Limite de Pivot, que consiste em aceitar para pivot da iteração k qualquer elemento $a_{rs}^{(k)}$ que satisfaça o seguinte critério

$$\begin{array}{l} |a_{rs}^{(k)}| \geq u|a_{is}^{(k)}| \text{ para todas as linhas } i \text{ da matriz } \bar{A}^{(k)} \\ \text{ou} \\ |a_{rs}^{(k)}| \geq u|a_{rj}^{(k)}| \text{ para todas as colunas } j \text{ da matriz } \bar{A}^{(k)} \end{array} \tag{7.21}$$

onde $u \in]0,1[$ é o chamado parâmetro limite. Notar que se $u=1$ então obtém-se a escolha parcial de pivot. Na prática, o valor $u=0.1$, usado por exemplo na *package* MA28 [Duff *et al*], é considerado bastante satisfatório e dá bastante liberdade para se escolher o pivot em termos da esparsidade da matriz. Assim por exemplo na coluna 1 da matriz (7.20) todos os elementos à excepção de 0.15 podem ser escolhidos como pivots se considerarmos $u=0.1$.

A decomposição LU com escolha limite de pivot é um processo estável, verificando-se a seguinte desigualdade

$$g_A \leq \left(1 + \frac{1}{u}\right)^{\alpha} \tag{7.22}$$

onde u e α foram definidos anteriormente. Tal como na escolha parcial de pivot, o valor de g_A é normalmente muito inferior ao valor do segundo membro da desigualdade (7.22).

Como há uma grande liberdade na escolha dos pivots, então ela pode ser feita em termos da esparsidade da matriz, escolhendo-se para pivot aquele que, obedecendo ao critério (7.21) provoca menos enchimentos. Um bom critério para este fim é o chamado Critério de Markowitz, que consiste em considerar na iteração k o elemento $a_{rs}^{(k)}$ que satisfaça

$$nl(r) \cdot nc(s) = \min\{nl(i) \cdot nc(j) : a_{ij}^{(k)} \text{ é elemento de } \bar{A}^{(k)}\} \tag{7.23}$$

onde $nl(i)$ e $nc(j)$ são respectivamente os números de elementos não nulos da linha i e coluna j da matriz reduzida $\bar{A}^{(k)}$ diferentes de $a_{ij}^{(k)}$.

A decomposição LU de uma matriz A não simétrica e com pivots não estáveis na diagonal é normalmente levada a cabo usando em cada iteração k os critérios de escolha limite de pivot e de Markowitz. Como as matrizes esparsas são armazenadas por colunas ou por linhas, então as colunas ou as linhas das sucessivas matrizes reduzidas são investigadas para a procura de um tal pivot. Para não tornar essa pesquisa demasiado trabalhosa, normalmente apenas um subconjunto de linhas ou colunas é considerado. Existem alguns processos de implementar essa escolha de um modo mais ou menos eficiente, mas não iremos discuti-los neste trabalho.

A título de exemplo, consideremos o sistema $Ax = b$, com

$$A = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 2 & -1 & 1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & 2 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 1 \end{bmatrix}$$

Na primeira iteração todos os elementos não nulos de A satisfazem o critério de escolha de limite com $u = 0.1$. Além disso a_{34} tem número de Markowitz igual a zero, pois $nr(3) = 2$ e $nc(4) = 0$. Então a_{34} pode ser escolhido como pivot. Para isso tem de ser colocado na posição $(1, 1)$, o que se faz por permutação das colunas 1 e 4 e das linhas 1 e 3. Assim tem-se

$$P_{13}AP_{14} = \begin{bmatrix} -1 & -1 & 2 & 0 \\ 0 & -1 & 1 & 2 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 2 & -1 \end{bmatrix}$$

Esse elemento é usado como pivot e obtém-se a seguinte matriz

$$B^{(2)} = (P_{13}AP_{14})^{(2)} = \left[\begin{array}{cc|ccc} -1 & & -1 & 2 & 0 \\ - & - & - & - & - \\ 0 & & -1 & 1 & 2 \\ 0 & & 0 & -1 & 1 \\ 0 & & 0 & 2 & -1 \end{array}\right]$$

É de notar que essa matriz é exactamente igual à anterior, o que significa que não foi efectuada qualquer operação. Agora $b_{22}^{(2)}$ tem número de Markowitz igual a zero, pois $nr(2) = 2$ e $nc(2) = 0$ (em relação ao Complemento de Schur). Portanto $b_{22}^{(2)}$ é usado como pivot e obtém-se a mesma matriz

$$B^{(3)} = (P_{13}AP_{14})^{(3)} = \left[\begin{array}{cc|ccc} -1 & -1 & & 2 & 0 \\ 0 & -1 & & 1 & 2 \\ - & - & - & - & - \\ 0 & 0 & & -1 & 1 \\ 0 & 0 & & 2 & -1 \end{array}\right]$$

Nesta última iteração todos os elementos satisfazem o critério limite de escolha de pivot e têm número de Markowitz igual a um. Então $b_{33}^{(3)}$ pode ser usado como pivot e obtém-se a matriz

$$(P_{13}AP_{14})^{(4)} = \begin{bmatrix} -1 & -1 & 2 & 0 \\ 0 & -1 & 1 & 2 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & -2 & 1 \end{bmatrix}$$

Donde

$$L = \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & -2 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} -1 & -1 & 2 & 0 \\ & -1 & 1 & 2 \\ & & -1 & 1 \\ & & & 1 \end{bmatrix}$$

Portanto a decomposição LU da matriz A foi obtida com apenas duas operações, o que confirma a redução do número de operações quando a esparsidade da matriz é devidamente explorada. Para resolver o sistema $Ax = b$ tem-se

$$Ax = b \Leftrightarrow P_{13}AP_{14}(P_{14}x) = P_{13}b$$

pois $P_{14}^{-1} = P_{14}$. Donde $Ax = b$ é equivalente a

$$LU \begin{bmatrix} x_4 \\ x_2 \\ x_3 \\ x_1 \end{bmatrix} = \begin{bmatrix} b_3 \\ b_2 \\ b_1 \\ b_4 \end{bmatrix}$$

Para obter a solução do sistema, resolve-se primeiramente $Ly = P_{13}b$, ou seja

$$\begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 1 \end{bmatrix}$$

e tem-se

$$y = \begin{bmatrix} 0 & 2 & 0 & 1 \end{bmatrix}$$

Finalmente tem de se resolver $U(P_{14}x) = y$, isto é,

$$\begin{bmatrix} -1 & -1 & 2 & 0 \\ & -1 & 1 & 2 \\ & & -1 & 1 \\ & & & 1 \end{bmatrix} \begin{bmatrix} x_4 \\ x_2 \\ x_3 \\ x_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 1 \end{bmatrix}$$

Donde $x_1 = x_2 = x_3 = x_4 = 1$ é a solução do sistema dado.

Um ponto muito importante a salientar é que, contrariamente ao que sucedia nos casos anteriores, a fase de Análise necessita em cada iteração dos valores numéricos dos elementos das matrizes reduzidas. Por isso, em muitas implementações, as fases de Análise e Factorização são combinadas numa só. No cálculo das matrizes reduzidas há normalmente enchimentos. Por isso a estrutura de dados que armazena a matriz original tem de ser modificada de modo a armazenar esses elementos. Daí dizermos que o processo de decomposição da matriz é dinâmico.

Na maior parte das implementações, a matriz A é armazenada, por linhas ou por colunas, usando o esquema de colecção de vectores que consiste nos vectores VALA, INDA e PNTA que foram referidos anteriormente. Além disso é considerado um vector NZA em que NZA(k) representa o número de elementos na linha ou coluna k de A. Para aplicar o critério de Markowitz, é ainda necessário ter a informação do número de elementos não nulos em cada linha (se a matriz estiver armazenada por colunas), ou número de elementos não nulos em cada coluna (se a matriz

estiver armazenada por linhas). Além disso, é necessário considerar um espaço de armazenagem bastante superior ao necessário para armazenar a matriz A, pois de outro modo os elementos correspondentes aos enchimentos não poderiam ser armazenados. Como referimos atrás, a estrutura de dados é modificada sempre que há enchimentos. Para ilustrar como a estrutura é modificada, consideremos a matriz (7.20). Esta matriz pode ser armazenada por linhas da seguinte forma

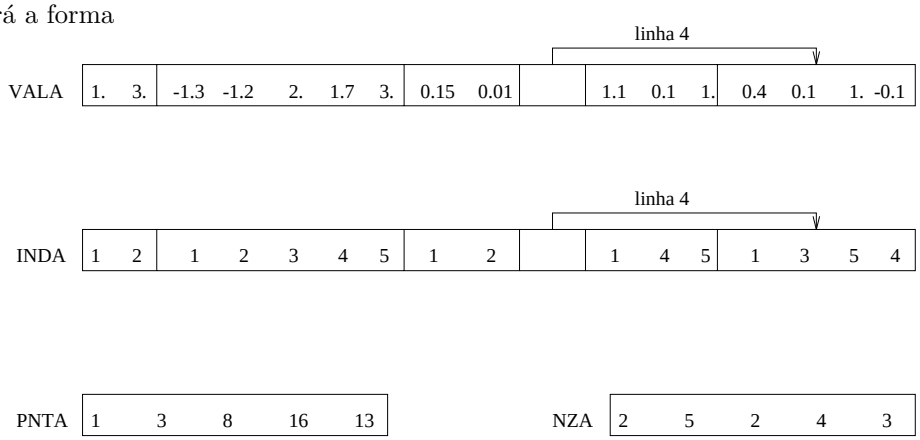
VALA = | 1. 3. | 2. 1.2 2. 2. 3. | 0.15 0.01 | 1.5 0.1 | 1.1 0.1 1. |

INDA = | 1 2 | 1 2 3 4 5 | 1 2 | 1 3 | 1 4 5 |

PNTA = | 1 3 8 10 12 |

NZA = | 2 5 2 2 3 |

É fácil de ver que a_{55} é um dos elementos que satisfaz simultaneamente os critérios de Markowitz e de escolha limite de pivot. Se escolhermos esse elemento para pivot, então há a necessidade de trocar duas linhas e duas colunas, modificando duas componentes nos vectores *perml* e *permc* de números inteiros correspondentes às permutações de linhas e colunas respectivamente. Além disso, há um enchimento na posição $(4, 4)$. Como não há espaço para armazenar esse elemento, os elementos correspondentes a essa linha 4 vão ser colocados no fim dos vectores VALA e INDA, deixando três espaços em branco nas posições onde essa linha estava armazenada. Os vectores PNTA e NZA serão também modificados, e os elementos numéricos de VALA são transformados de acordo com as regras da eliminação de Gauss. A estrutura de dados após a primeira iteração terá a forma



De notar que os elementos da linha 4 ficam desordenados, pois o processo de eliminação de Gauss para matrizes esparsas baseia-se no algoritmo 28 para determinar adições algébricas de vectores esparsos.

Em certos casos, os elementos referentes aos enchimentos podem ser armazenados imediatamente a seguir aos elementos da mesma linha já armazenados. Tal acontece quando a linha é a última ou quando já há espaços em branco na estrutura de dados que armazena a matriz. Assim, no exemplo apresentado anteriormente, há espaço para qualquer enchimento que se dê na linha 3. De notar que neste último caso os elementos reais e correspondentes índices são acrescentados nos espaços em branco e o vector PNTA não é modificado, sendo NZA modificado da mesma maneira que anteriormente.

Se durante o processo de decomposição houver muitos enchimentos pode acontecer que não haja mais espaço para colocar uma linha que tenha sido modificada. Neste caso, a estrutura de dados é *comprimida* de modo a desaparecerem os espaços em branco que os vectores VALA e INDA contêm. Quando tal acontece o vector PNTA é também modificado, e após essa operação já há espaço suficiente para colocar a nova linha.

O processo de Análise e Factorização é baseado nas ideias que apresentámos nesta secção e termina com a representação das matrizes triangulares da decomposição. O processo de Solução

consiste, como anteriormente, na resolução de dois sistemas triangulares $Ly = b$ e $Ux = y$, usando essas estruturas de dados.

Como referimos no capítulo 5, a forma triangular por blocos pode ser altamente vantajosa para a resolução de sistemas de equações com matrizes esparsas. A construção dessa forma triangular consiste de duas fases, que apresentamos a seguir.

1. Encontrar matrizes de permutação P e Q tais que PAQ tem uma diagonal sem zeros.
2. Encontrar uma matriz de permutação $\bar{P}$ tal que $\bar{P}^T(PAQ)\bar{P}$ tem a forma pretendida.

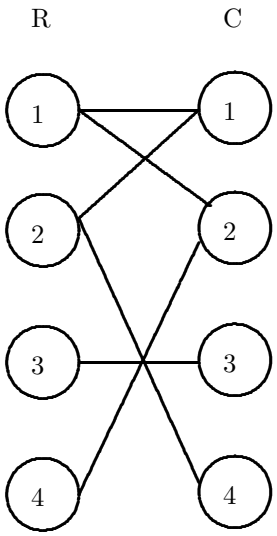
Estas duas fases são efectuadas usando algoritmos combinatórios. Em relação à Fase 1, podemos sempre associar a uma matriz A de ordem n um Digrafo $G = (R, C, E)$, tal que os conjuntos R e C de nós correspondem às linhas e colunas de A e

$$x_i \in R, x_j \in C, \{x_i, x_j\} \in E \Leftrightarrow a_{ij} \neq 0$$

Assim por exemplo a matriz

$$A = \begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & & & a_{24} \\ & & a_{33} & \\ & a_{42} & & \end{bmatrix}$$

tem associada um digrafo da forma



Se P e Q são matrizes de permutação, então os digrafos de A e PAQ tém a mesma estrutura mas numerações diferentes para os nós. De acordo com a correspondência apresentada, a Fase 1 consiste em encontrar uma numeração para os nós de R e de C de modo a que exista sempre uma aresta de $i \in R$ para $j \in C$. Esse processo costuma-se denominar de "procura de uma transversal" e existem algoritmos combinatórios para o efeito. De notar que no exemplo anterior basta considerar a ordem $(1, 4, 3, 2)$ nos nós de C para obter o pretendido. Portanto a troca da segunda e quarta colunas de A é suficiente para terminar a Fase 1.

Como a matriz PAQ tem todos os seus elementos diagonais diferentes de zero, podemos-lhe associar um grafo orientado. Em termos de teoria de grafos, encontrar a forma triangular por blocos consiste em encontrar as componentes fortemente conexas do grafo associado à matriz PAQ. Diz-se que um grafo G é Fortemente Conexo se dois quaisquer nós u e v de G são mutuamente alcançáveis, isto é, se existe sempre um caminho de u para v e outro de v para u. Uma Componente Fortemente Conexa de um grafo G é um subgrafo que é fortemente conexo e que não pode ser aumentado sem

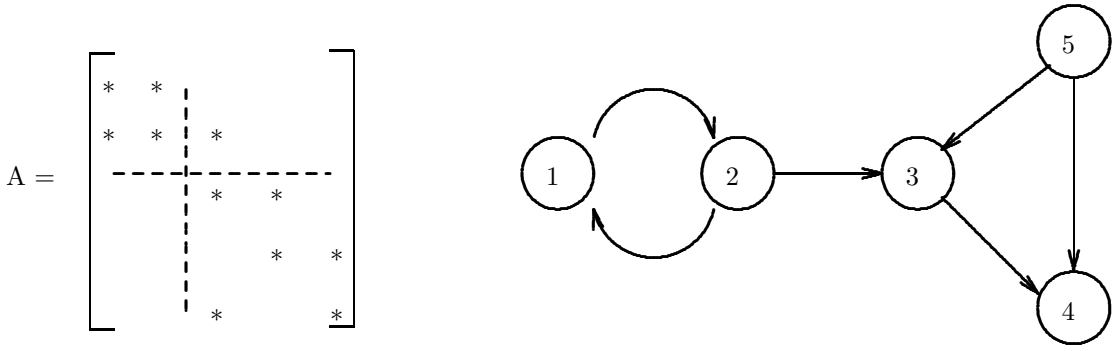


Figura 7.4: Uma matriz triangular por blocos e o grafo fortemente conexo que lhe está associado.

perder essa propriedade. Na figura 7.4 apresentamos uma matriz triangular por blocos com dois blocos diagonais, apresentando um grafo com duas componentes fortemente conexas.

Dada uma matriz A, encontrar uma matriz de permutação $\bar{P}$ tal que $\bar{P}^T A \bar{P}$ é da forma

$$\bar{P}^T A \bar{P} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1l} \\ & A_{22} & \dots & A_{2l} \\ & & \ddots & \vdots \\ & & & A_{ll} \end{bmatrix} \tag{7.24}$$

é equivalente a encontrar uma numeração para os nós do seu grafo orientado G, tal que G possa ser particionado em l componentes fortemente conexas e arestas de ligação entre essas componentes. Tal como para a fase 1, existem algoritmos combinatórios que levam a cabo o pretendido, mas que não serão aqui apresentados.

Se uma matriz A for particionada na forma (7.24), então apenas as matrizes A_{kk} têm de ser factorizadas, pois resolver o sistema $Ax = b$ é equivalente a resolver l sistemas com matrizes A_{kk}, com $k = 1, \dots, l$. Os blocos não diagonais apenas são necessários para multiplicações matriz por vector. Essas matrizes A_{kk} são factorizadas usando os critérios de Markowitz e escolha limite de pivot, se tiverem pivots diagonais instáveis.

Consideremos agora o caso de uma matriz ser não simétrica, e não simétrica de posição mas com pivots diagonais estáveis. Se A é redutível, isto é, se se puder reduzir à forma triangular por blocos, suponhamos que os blocos diagonais A_{kk} satisfazem essas pretensões. Como os pivots diagonais são estáveis, então as Fases de Análise e Factorização para A ou para cada um dos blocos diagonais, consistem em usar em cada iteração o Critério de Markowitz para escolha do pivot diagonal.

Como vimos na secção 4.12 do capítulo 4, a resolução de sistemas de equações lineares com matrizes simétricas indefinidas utiliza uma escolha de pivot do tipo parcial e pivots de ordens 1×1 e 2×2 de modo a que a simetria seja mantida durante o processo de factorização. Se a matriz é esparsa é possível modificar o critério de escolha de pivot de modo a que esse elemento também seja obtido segundo um critério que tenha em conta a preservação da esparsidade da matriz. Esse tipo de escolha e o critério de Markowitz ($nr(k) = nc(k)$) são normalmente usados no processo de resolução de sistemas de equações lineares com matrizes indefinidas, que é implementado de acordo com as ideias explicadas nesta secção. Uma implementação desse processo é uma opção do programa MA27 referido anteriormente.

7.4 Uso do Complemento de Schur

Em muitas aplicações o sistema de equações lineares $Ez = g$ a resolver pode ser decomposto na forma

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix} \tag{7.25}$$

com $A \in \mathbb{R}^{n\times n}$, $D \in \mathbb{R}^{m\times m}$, $x, b \in \mathbb{R}^n$, $y, d \in \mathbb{R}^m$, m pequeno e n muito elevado. Além disso A é muito esparsa e não singular. Como A é não singular, então podemos decompor a matriz do sistema na forma

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} A & \\ C & I_m \end{bmatrix} \begin{bmatrix} I_n & A^{-1}B \\ & (E|A) \end{bmatrix} \tag{7.26}$$

com $(E|A)$ o Complemento de Schur de A em E dado por

$$(E|A) = D - CA^{-1}B \tag{7.27}$$

e I_p é a matriz identidade de ordem p. Como A é não singular e a matriz E do sistema também o é, então pela Fórmula de Schur o mesmo acontece ao Complemento de Schur $(E|A)$. Portanto o sistema (7.25) pode ser resolvido a partir de

$$\begin{bmatrix} A & 0 \\ C & I_m \end{bmatrix} \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix} \tag{7.28}$$

e

$$\begin{bmatrix} I_n & A^{-1}B \\ 0 & (E|A) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} \tag{7.29}$$

Mas (7.28) é equivalente a

$$\begin{cases} A\bar{x} &= b \\ \bar{y} &= d - C\bar{x} \end{cases} \tag{7.30}$$

Além disso (7.29) pode ser escrito na forma

$$\begin{cases} (E|A)y &= \bar{y} \\ x &= \bar{x} - A^{-1}B\bar{y} \end{cases} \tag{7.31}$$

Tendo em conta as fórmulas (7.30) e (7.31) podemos resolver o sistema (7.25) por

$$\begin{cases} A\bar{x} &= b \\ \bar{y} &= d - C\bar{x} \\ (E|A)y &= \bar{y} \\ \bar{d} &= B\bar{y} \\ Af &= \bar{d} \\ x &= \bar{x} - f \end{cases} \tag{7.32}$$

Este processo de resolver o sistema (7.25) requere a solução de dois sistemas com a matriz A. Como A é esparsa e tem ordem elevada, então as técnicas discutidas neste capítulo podem ser usadas para determinar a sua decomposição LU (ou LDL^T). No segundo sistema $Af = \bar{d}$ essa decomposição é usada, pelo que é apenas necessário resolver dois sistemas com matrizes triangulares neste último caso. Para completar a descrição deste processo é importante descrever como é que o Complemento de Schur $F = (E|A)$ é calculado. Essa matriz é construída coluna a coluna a partir do seguinte processo

$$\begin{array}{l} \text{Para } j = 1, \ldots, m \\ \quad Av = B_{\cdot j} \\ \quad F_{\cdot j} = D_{\cdot j} - Cv \end{array} \tag{7.33}$$

Nome	Origem
BCTK06	Colecção Harwell-Boeing
BCTK08	
BCTK09	
BCTK11	
BCTK14	
RR1030	
OOP110	Equações de derivadas parciais
OOP210	

Tabela 7.1: Origens dos problemas-teste

			Involucro						MA27					
				CPU						CPU				
	N	NZA	|ENV|	AN	FACT	SOLV	RES	ERR	N2L	AN	FACT	SOLV	RES	ERR
BCTK06	420	3720	26624	0.03	0.67	0.06	1E-05	8E-11	11941	0.13	0.83	0.03	9E-06	8E-11
BCTK08	1074	5943	407624	0.06	89.07	1.00	9E-05	1E-10	30684	2.37	3.96	0.12	1E-04	1E-10
BCTK09	1083	8677	355539	0.06	63.92	0.84	3E-07	1E-12	63979	0.76	5.93	0.18	2E-07	9E-14
BCTK11	1473	16384	263102	0.13	30.83	0.63	2E-06	7E-10	52715	0.44	3.18	0.15	1E-06	5E-10
BCTK14	1806	30824	410137	0.22	46.56	0.96	7E-05	8E-11	114263	0.86	12.34	0.34	4E-05	8E-11
RR1030	1086	10492	117590	0.07	5.59	0.28	7E-08	1E-10	23271	0.25	0.98	0.07	5E-08	1E-11
OOP110	1000	1890	18251	0.02	0.22	0.04	2E-14	4E-15	8039	0.21	0.29	0.04	1E-14	1E-15
OOP210	3000	5395	15012	0.09	0.17	0.05	1E-14	7E-15	16171	0.61	0.64	0.08	2E-14	4E-16

Tabela 7.2: Experiência computacional para matrizes simétricas esparsas

Como a decomposição LU de A é conhecida, é apenas necessário resolver $2m$ sistemas triangulares e efectuar m multiplicações da matriz C pelos m vectores v que são soluções dos sistemas $Av = B_{.j}$.

Assim podemos concluir que o processo do Complemento de Schur necessita de uma Fase de Análise e Factorização da matriz A, $(m+2)$ fases de Solução com A e da resolução de um sistema com o Complemento de Schur $F = (E|A)$. É evidente que este processo só é recomendado se m é pequeno, pois de outro modo envolveria um grande número de sistemas triangulares para resolver. Como m é pequeno, então a matriz Complemento de Schur F tem de ser armazenada como densa. Com efeito, a resolução de sistemas com matrizes densas é um procedimento bastante mais rápido do que o correspondente procedimento com matrizes esparsas. Além disso a matriz F tem muitos poucos elementos nulos. Este processo do Complemento de Schur encontra uma aplicação importante na resolução de um sistema cuja matriz é esparsa de ordem elevada, mas contém algumas linhas quase densas (ou pelo menos com muitos elementos não nulos). Nesse caso efectuam-se permutações principais de modo a colocar essas linhas na parte final, isto é, a constituir o bloco $[C\ D]$ da matriz do sistema (7.25).

7.5 Experiência computacional

Para terminar este capítulo apresentamos na tabela 7.2 a aplicação dos métodos do invólucro (algoritmo RCM) e do grau mínimo (MA27) para a resolução de sistemas de equações lineares com matrizes simétricas positivas definidas de oito problemas de aplicações concretas, cujas origens são apresentadas na tabela 7.1.

Na tabela 7.2 usámos as seguintes notações:

N: Ordem da matriz.

NZA: Número de elementos não nulos abaixo da diagonal.

|ENV|: Dimensão do invólucro da matriz permutada (RCM).

NZL: Número de elementos não nulos da matriz L (MA27).

CPU: Tempo de CPU:

AN: Fase de Análise;

FACT: Fase de Factorização;

SOLV: Fase de Solução.

RES: Norma do resíduo, dada por

$$||b - A\bar{x}||_\infty$$

em que $\bar{x}$ é a solução obtida pelo método.

ERR: Norma do erro, dada por

$$||\bar{x} - x^*||_\infty$$

em que x^* é a solução exacta do sistema.

Os termos independentes b foram gerados por forma a que a solução dos sistemas fosse o vector $e = [1, 1, 1, \ldots, 1]^T$, isto é, $b = Ae$.

Comparando os valores de |ENV| e NZL facilmente se conclui que, à excepção do problema OOP210, o espaço de armazenagem para o método do invólucro é bastante superior ao exigido no algoritmo do grau mínimo. Por isso não é de estranhar que as fases de factorização e solução do método do grau mínimo sejam em geral mais rápidas do que as do invólucro. Em certos casos as diferenças são mesmo bastante significativas.

Como vimos anteriormente, a fase de Análise do método do invólucro é bastante mais simples do que a do método do grau mínimo. Essa maior simplicidade é confirmada pela leitura dos tempos de execução das Fases de Análise dos dois processos onde o método do invólucro é sempre mais rápido.

Não se observam diferenças significativas nos valores dos resíduos correspondentes às soluções obtidas pelos métodos do invólucro e do grau mínimo. O algoritmo do grau mínimo obtém soluções um pouco mais precisas (ERR é menor), sendo isso consequência de haver menos operações nesse último processo.

Exercícios

1. Considere a seguinte matriz

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 2 \\ -1 & 2 & 1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 & -2 & 0 \\ 0 & 2 & 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & 0 & 2 & 1 \\ 2 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

   (a) Mostre que $A \notin$ SPD e determine um real $\mu > 0$ tal que $B = A + \mu I \in$ SPD.

   (b) Efectue a Fase de Análise do algoritmo do grau mínimo para a matriz B.

2. Considere a seguinte matriz

$$A = \begin{bmatrix} 3.0 & 0.0 & 0.0 & -1.0 & 0.0 & -0.2 \\ 0.0 & -4.1 & 1.0 & 1.0 & 0.0 & 1.2 \\ 0.0 & -1.0 & 1.4 & 0.0 & 0.2 & 0.0 \\ 1.0 & 1.0 & 0.0 & 4.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -0.5 & 0.0 & 2.1 & 1.1 \\ -1.0 & 0.5 & 0.0 & 0.0 & 1.8 & -3.0 \end{bmatrix}$$

(a) Justifique se pode usar o algoritmo do grau mínimo para a resolução de um sistema com essa matriz.

(b) Em caso afirmativo, efectue a Fase de Análise desse algoritmo.

3. Efectue a Fase de Análise do método do invólucro para as matrizes B e A dos exercícios anteriores.

4. Considere a matriz

$$A = \begin{bmatrix} 3 & -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 5 & 1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 4 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & -1 \\ 0 & 0 & 2 & 0 & 4 & -1 & 0 \\ 1 & -1 & 0 & 0 & -1 & x & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 3 \end{bmatrix}$$

(a) Determine um valor de x tal que $A \in \text{SPD}$.

(b) Efectue a Fase de Análise do algoritmo do invólucro para esse valor de x, apresentando a estrutura de dados que armazena a matriz permutada.

5. Desenvolva uma implementação do algoritmo para a resolução de um sistema triangular $Lx = b$, com L armazenada segundo o esquema de banda variável.

6. Desenvolva uma implementação do algoritmo de decomposição LDL^T de uma matriz simétrica SPD usando o método do invólucro.

7. Desenvolva uma implementação do algoritmo RCM usando uma lista de adjacência.

8. Desenvolva uma implementação da Fase de Solução do Algoritmo do Grau Mínimo para a resolução de um sistema $Ax = b$, com A uma matriz não simétrica não singular simétrica de posição e diagonalmente dominante.

9. Considere as seguintes matrizes

$$A_1 = \begin{bmatrix} 3 & 0 & 0 & 2 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & 1 & 3 & -1 & -1 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & -1 & 0 & 3 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 2 & 0 & -1 & -1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ -1 & 0 & 2 & 1 & 0 \\ 0 & -1 & 0 & 3 & 0 \\ 0 & 0 & -1 & 1 & 2 \end{bmatrix}$$

(a) Mostre que os pivots diagonais são estáveis.

(b) Efectue a Fase de Análise como matriz simétrica de posição.

(c) Efectue a Fase de Análise e a Fase de Factorização sem assumir que as matrizes são simétricas de posição.

10. Considere a seguinte matriz

$$A = \begin{bmatrix} 5 & -1 & 1 & -2 & 0 & 0 \\ -1 & 3 & 0 & 0 & 1 & 0 \\ -2 & 0 & 4 & 0 & 0 & 1 \\ 1 & 0 & 0 & \frac{7}{2} & 1 & -1 \\ 0 & -1 & 0 & 3 & 6 & 1 \\ 0 & 0 & \frac{1}{2} & -1 & 2 & 4 \end{bmatrix}$$

(a) Mostre que o algoritmo do grau mínimo pode ser aplicado na resolução de um sistema com essa matriz.

(b) Efectue a Fase de Análise desse algoritmo, apresentando a estrutura de dados que armazena a matriz permutada.

11. Resolva os seguintes sustemas:

(a)
$$\begin{cases} x_1 & & +x_3 & -2x_4 & = & 0 \\ & x_2 & & -x_4 & = & 0 \\ -x_1 & +2x_2 & & +x_4 & = & 2 \\ & -x_2 & & +2x_4 & = & 1 \end{cases}$$

(b)
$$\begin{cases} x_1 & & -x_3 & & = & 0 \\ 2x_1 & -x_2 & +x_3 & & = & 2 \\ & -x_2 & +2x_3 & -x_4 & = & 0 \\ -x_1 & & +2x_3 & & = & 1 \end{cases}$$

considerando as suas matrizes como esparsas.

12. Considere a seguinte matriz
$$A = \begin{bmatrix} B & -e & 0 \\ e^T & 0 & 0 \\ C & f & D \end{bmatrix}$$
com B uma matriz SPD de ordem n, e um vector positivo de dimensão n, $C \in \mathbb{R}^{m\times n}$, $f \in \mathbb{R}^m$, D uma matriz H_+ de ordem m e 0 elementos, vectores ou matrizes nulas de ordens apropriadas.

(a) Mostre que $\det(A) > 0$.

(b) Diga como resolve um sistema com essa matriz quando m e n são elevados.

13. Considere a matriz
$$A = \begin{bmatrix} I_n & L & C \\ -L^T & 0 & 0 \\ 0 & -C^T & B \end{bmatrix}$$
com L uma matriz SDD_+ esparsa triangular inferior de ordem n, B uma matriz PD de ordem 10, C uma matriz esparsa de ordem $n \times 10$ e I_n a matriz identidade de ordem n.

(a) Diga como armazena as matrizes L, B e C.

(b) Mostre que $\det(A) > 0$.

(c) Indique como resolve um sistema com a matriz A sem ocorrência de enchimentos.

14. Considere a matriz
$$A = \begin{bmatrix} I_n + pp^T & -C^T & E \\ C & 0 & F \\ 0 & 0 & D \end{bmatrix}$$
com 0 matrizes nulas, I_n a matriz identidade de ordem n, $p \in \mathbb{R}^n$, $C \in \mathbb{R}^{m\times n}$, $E \in \mathbb{R}^{n\times r}$, $F \in \mathbb{R}^{m\times r}$ e D uma matriz P de ordem r.

(a) Mostre que $\det(A) > 0$.

(b) Mostre que
$$\left(I_n + pp^T\right)^{-1} = I_n - \frac{1}{1+p^Tp}pp^T$$

(c) Indique como resolve um sistema com a matriz A quando n e m são elevados e r é pequeno.

Capítulo 8

Métodos Iterativos para Sistemas de Equações Lineares com Matrizes Quadradas

Neste capítulo iremos analisar alguns dos métodos iterativos mais usados para resolver sistemas de equações lineares. Começaremos por introduzir os algoritmos básicos, baseados nos métodos de Jacobi e Gauss–Seidel. Seguidamente apresentaremos técnicas iterativas mais avançadas, como os métodos dos gradientes conjugados e da partição. A finalizar este capítulo será analisado o uso da técnica de precondicionamento na resolução de sistemas de equações lineares.

8.1 Métodos iterativos básicos

Consideremos o sistema de equações lineares

$$Ax = b$$

onde $A \in \mathbb{R}^{n\times n}$ é uma matriz não singular tal que $a_{ii} \neq 0, i = 1, \ldots, n$.

Os métodos iterativos que iremos estudar nesta secção têm a forma

$$x^{(k+1)} = Hx^{(k)} + d \tag{8.1}$$

com $x^{(0)}$ um vector de ordem n dado e $d \in \mathbb{R}^n$, $H \in \mathbb{R}^{n\times n}$ tais que

$$Ax = b \Leftrightarrow x = Hx + d \tag{8.2}$$

Contrariamente aos métodos directos, os métodos iterativos não procuram obter uma solução exacta do sistema, mas geram uma sucessão de vectores $x^{(k)}$ cujo limite é uma solução do sistema. Por isso para qualquer método iterativo há a necessidade de estabelecer condições que assegurem a sua convergência. O teorema seguinte apresenta-nos uma condição necessária e suficiente de convergência da família de algoritmos representada por (8.1).

Teorema 8.1 (Teorema Fundamental dos Métodos Iterativos) *O algoritmo definido por (8.1) converge para a solução do sistema $Ax = b$, qualquer que seja o vector inicial $x^{(0)}$, se e só se $\rho(H) < 1$, com $\rho(H)$ o raio espectral de H.*

Demonstração: Seja x^* a solução exacta de $Ax = b$. Se $x^{(k+1)}$ é a solução aproximada desse sistema obtida pelo método iterativo (8.1), então tem-se

$$x^{(k+1)} - x^* = Hx^{(k)} + d - (Hx^* + d) = H(x^{(k)} - x^*)$$

Portanto também

$$x^{(k)} - x^* = H(x^{(k-1)} - x^*)$$

Logo

$$\begin{aligned} x^{(k+1)} - x^* &= H^2(x^{(k-1)} - x^*) = \dots \\ &= H^{k+1}(x^{(0)} - x^*) \end{aligned}$$

Então a sucessão $\{x^{(k)}\}_k$ converge para a solução do sistema se e só se $\lim_k H^k = 0$, o que, pelo teorema 2.4, é equivalente a $\rho(H) < 1$.

Como referimos anteriormente, o método iterativo definido por (8.1) é um processo que vai obtendo uma sucessão de vectores cujo limite é a solução do sistema. Como o limite de uma sucessão não pode ser usada num computador é necessário desenvolver Critérios de Paragem que permitam terminar o algoritmo com uma aproximação da solução que satisfaça os nossos objectivos. O próximo teorema dá uma ajuda nesse sentido.

Teorema 8.2 *Se $||H|| < 1$ para qualquer norma matricial subordinada $||\cdot||$, então o algoritmo definido por (8.1) converge para a solução x^* do sistema qualquer que seja o vector inicial $x^{(0)}$. Além disso para qualquer iteração k*

$$||x^* - x^{(k)}|| \leq \frac{||H||}{1 - ||H||} ||x^{(k)} - x^{(k-1)}|| \tag{8.3}$$

Demonstração: A primeira parte deste teorema é consequência de $\rho(H) \leq ||H||$ para qualquer norma matricial subordinada e do teorema anterior. Em relação à igualdade (8.3) tem-se

$$\begin{aligned} x^{(k+1)} - x^{(k)} &= Hx^{(k)} + d - \left(Hx^{(k-1)} + d\right) \\ &= H\left(x^{(k)} - x^{(k-1)}\right) \end{aligned}$$

Portanto para qualquer inteiro m positivo vem

$$x^{(k+m+1)} - x^{(k+m)} = H^{m+1}\left(x^{(k)} - x^{(k-1)}\right)$$

Além disso

$$x^{(k+p)} - x^{(k)} = \sum_{m=0}^{p-1} \left(x^{(k+m+1)} - x^{(k+m)}\right)$$

para qualquer inteiro positivo p. Donde

$$\begin{aligned} ||x^{(k+p)} - x^{(k)}|| &\leq \sum_{m=0}^{p-1} ||H^{m+1}||\ ||x^{(k)} - x^{(k-1)}|| \\ &\leq \sum_{m=0}^{p-1} ||H||^{m+1} ||x^{(k)} - x^{(k-1)}|| \\ &= \frac{||H|| - ||H||^{p+1}}{1 - ||H||} ||x^{(k)} - x^{(k-1)}|| \end{aligned}$$

Mas

$$\lim_{p\to\infty} x^{(k+p)} = x^*$$

e por $||H|| < 1$, tem-se

$$\lim_{p\to\infty} ||H||^{p+1} = 0$$

Então aplicando limites na desigualdade anterior obtém-se a desigualdade (8.3).

Este teorema mostra que se pretendermos uma solução aproximada $x^{(k)}$ de x^* tal que

$$||x^{(k)} - x^*|| \leq \epsilon$$

então devemos parar o processo iterativo quando

$$||x^{(k)} - x^{(k-1)}|| \leq \delta \tag{8.4}$$

com

$$\delta = \frac{1 - ||H||}{||H||}\epsilon$$

Portanto (8.4) pode servir como Critério de Paragem do método iterativo. A escolha da constante δ não é no entanto pacífica. Com efeito, δ deve ser bastante pequeno principalmente quando a norma de H tem um valor próximo de um. No entanto um valor demasiado pequeno para δ poderá fazer com que o algoritmo nunca termine, por a desigualdade $||x^{(k)} - x^{(k-1)}|| \leq \delta$ não se verificar. Na prática, o erro relativo é usado em (8.4) em vez do absoluto. Por outro lado, há que ultrapassar os problemas decorrentes das soluções com normas muito próximas de zero. Daí o critério (8.4) ser substituído por

$$\frac{||x^{(k)} - x^{(k-1)}||}{\max\{1, ||x^{(k)}||\}} \leq \delta \tag{8.5}$$

A constante δ tem um valor $\delta = 10^{-t}$, com t um número inteiro positivo pequeno que é escolhido pelo utilizador de acordo com os comentários acima apresentados.

Tal como referimos em capítulos anteriores, a norma do resíduo $r^{(k)} = b - Ax^{(k)}$ é uma boa medida para verificar se $x^{(k)}$ é uma boa aproximação da solução do sistema $Ax = b$. Portanto um outro Critério de Paragem a utilizar é

$$||r^{(k)}|| = ||b - Ax^{(k)}|| \leq \delta \tag{8.6}$$

com δ uma quantidade pequena da forma $\delta = 10^{-t}$. Seguidamente iremos discutir o valor dessa tolerância. Se x^* é a solução única do sistema $Ax = b$, então

$$x^* - x^{(k)} = A^{-1}b - A^{-1}Ax^{(k)} = A^{-1}\left(b - Ax^{(k)}\right) = A^{-1}r^{(k)}$$

Donde

$$||x^* - x^{(k)}|| \leq ||A^{-1}||\ ||r^{(k)}||$$

e portanto se $x^* \neq 0$ vem

$$\frac{||x^* - x^{(k)}||}{||x^*||} \leq \frac{||A^{-1}||}{||x^*||}||r^{(k)}||$$

Portanto a verificação do Critério de Paragem (8.6) implica que

$$\frac{||x^* - x^{(k)}||}{||x^*||} \leq \frac{||A^{-1}||}{||x^*||}\delta$$

Se $||A^{-1}||$ é pequeno, um valor de δ relativamente elevado ($\delta \geq \sqrt{\epsilon_M}$) fornecerá uma boa aproximação da solução do sistema. Contudo se $||A^{-1}||$ é elevado, então δ tem de ser bastante pequeno para que a solução $x^{(k)}$ obtida pelo método seja uma boa aproximação da solução do sistema. No entanto, um valor muito pequeno para δ poderá implicar que o Critério de Paragem (8.6) nunca se verifique e o algoritmo nunca termine. Se recordarmos que

$$\text{cond}(A) = ||A||\ ||A^{-1}||$$

então facilmente concluimos que os métodos iterativos têm dificuldades em terminar com uma boa aproximação da solução do sistema $Ax = b$ quando A é uma matriz mal condicionada. Essa é aliás uma das grandes desvantagens dos métodos iterativos.

A eficiência de um método iterativo depende do número de iterações k necessárias para a obtenção de uma solução aproximada. Se x^* é a solução exacta do sistema, então tem-se

$$||x^{(k)} - x^*|| \leq ||H||^k ||x^{(0)} - x^*||$$

para uma qualquer norma vectorial e uma qualquer norma matricial que com ela seja compatível. Como $\rho(H) \leq ||H||$, podemos escrever

$$||x^{(k)} - x^*|| \approx [\rho(H)]^k\ ||x^{(0)} - x^*||$$

Suponhamos agora que $||x^{(0)}|| = 0$. Então tem-se

$$\frac{||x^{(k)} - x^*||}{||x^*||} \approx [\rho(H)]^k$$

Deste modo, se quisermos obter uma aproximação $x^{(k)}$ de x^* com um erro relativo inferior a 10^{-t}, o número k de iterações a efectuar terá que satisfazer a

$$[\rho(H)]^k \leq 10^{-t}$$

ou seja

$$k \geq \frac{t}{-\log_{10} \rho(H)}$$

Da fórmula anterior conclui-se que o número de iterações aumenta quando $\rho(H)$ percorre o intervalo aberto $]0, 1[$, sendo bastante elevado para matrizes H cujo raio espectral seja próximo de 1. Portanto nos métodos iterativos a matriz H deve ter raio espectral não só menor que um (para que o método seja convergente) mas também o menor possível (para ter a máxima eficiência). Além disso o número de iterações cresce com o aumento da precisão t da solução a obter pelo algoritmo.

O desenvolvimento de métodos iterativos para a resolução de sistemas de equações lineares reside na determinação de matrizes H que satisfaçam a equivalência (8.3) e cujo raio espectral (ou a norma) seja menor que um. O processo mais frequentemente usado para determinar H consiste em decompor a matriz A do sistema $Ax = b$ numa diferença de duas matrizes B e C tais que

$$A = B - C \tag{8.7}$$

com B não singular. Se tal acontecer, então podemos escrever o sistema na forma

$$x = B^{-1}Cx + B^{-1}b \tag{8.8}$$

e portanto

$$H = B^{-1}C \tag{8.9}$$

O método iterativo é convergente se $\rho(B^{-1}C) < 1$ e isso só é possível para determinadas escolhas das matrizes B e C. Em termos computacionais, a matriz B deverá ser escolhida de modo a que a sua inversa seja facilmente calculável. Nesta secção iremos apresentar dois processos que corresponderão a matrizes H que satisfazem (8.9), diferindo apenas na escolha das matrizes B e C que aparecem em (8.7).

O primeiro processo é conhecido por Método de Jacobi, e baseia-se na seguinte escolha:

$$B = D = diag(A) = \begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix} \tag{8.10}$$

$$C = -\begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ a_{21} & 0 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & 0 \end{bmatrix} = L + U \tag{8.11}$$

com

$$L = -\begin{bmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \vdots & \ddots & \\ a_{n1} & a_{n2} & \dots & 0 \end{bmatrix}, \quad U = -\begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ & 0 & \dots & a_{2n} \\ & & \ddots & \vdots \\ & & & 0 \end{bmatrix} \tag{8.12}$$

Portanto

$$x^{(k+1)} = D^{-1}(L+U)x^{(k)} + D^{-1}b \tag{8.13}$$

ou

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left[b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k)}\right], \ i = 1, 2, \dots, n \tag{8.14}$$

Por outro lado no Método de Gauss-Seidel tem-se

$$B = D - L, \quad C = U$$

com L e U dadas por (8.12) e portanto

$$x^{(k+1)} = (D-L)^{-1}Ux^{(k)} + (D-L)^{-1}b \tag{8.15}$$

ou

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right], \quad i = 1, \dots, n \tag{8.16}$$

Como vimos anteriormente, a eficiência de um método iterativo da forma (8.1) é tanto maior quanto menor for o valor do raio espectral $\rho(H)$ da matriz H. Uma prática corrente consiste em introduzir um parâmetro positivo ω nos métodos de Jacobi e Gauss-Seidel de modo a que o raio espectral da matriz H_ω dos métodos resultantes seja menor que os raios espectrais das matrizes

Algoritmo 41

Para $k = 1, 2, \ldots$

Para $i = 1, \ldots, n$

$$\bar{x}_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right]$$

$$x_i^{(k+1)} = \omega \bar{x}_i^{(k+1)} + (1 - \omega) x_i^{(k)}$$

Figura 8.1: Método JOR

H correspondentes aos métodos de Jacobi e Gauss-Seidel. Assim se obtêm os chamados Métodos JOR e SOR, que têm as formas apresentadas respectivamente nas figuras 8.1 e 8.2.

Portanto nestes dois métodos a solução obtida pelo método de Jacobi ou de Gauss-Seidel é corrigida usando um parâmetro $\omega > 0$. Quando $\omega = 1$, nenhuma correcção tem lugar e os processos JOR e SOR reduzem-se aos algoritmos de Jacobi e Gauss-Seidel respectivamente. De entre estes dois processos, o método JOR só pontualmente tem merecido algum destaque, pelo que neste capítulo apenas iremos estudar o método SOR.

A convergência e a eficiência do método SOR dependem do valor do parâmetro ω que é fornecido pelo utilizador. Seguidamente iremos demonstrar que ω tem de pertencer a um determinado intervalo real para se poder garantir convergência. As relações expressas no algoritmo 42 podem-se escrever na forma

$$a_{ii} x_i^{(k+1)} + \omega \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} = (1 - \omega) a_{ii} x_i^{(k)} - \omega \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} + \omega b_i, \quad i = 1, \ldots, n$$

Se D, L e U forem as matrizes definidas por (8.11) e (8.12), então podemos escrever as n igualdades anteriores na forma

$$Dx^{(k+1)} - \omega L x^{(k+1)} = (1 - \omega) D x^{(k)} + \omega U x^{(k)} + \omega b$$

ou ainda

$$x^{(k+1)} = H_\omega x^{(k)} + \omega (D - \omega L)^{-1} b \tag{8.17}$$

com

$$H_\omega = (D - \omega L)^{-1} \left[(1 - \omega) D + \omega U \right] \tag{8.18}$$

É evidente que H_1 é a matriz correspondente ao método de Gauss-Seidel. O método SOR é convergente para a solução do sistema $Ax = b$ independentemente do vector inicial escolhido se e só se $\rho(H_\omega) < 1$. Mas

$$\det(H_\omega) = \left[\det(D) \right]^{-1} (1 - \omega)^n \det(D) = (1 - \omega)^n$$

Algoritmo 42

Para $k = 1, 2, \ldots$

Para $i = 1, \ldots, n$

$$\bar{x}_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right]$$

$$x_i^{(k+1)} = \omega \bar{x}_i^{(k+1)} + (1 - \omega) x_i^{(k)}$$

Figura 8.2: Método SOR

Se λ_i, $i = 1, \ldots, n$ são os valores próprios de H_ω, então

$$|1 - \omega|^n = |\det(H_\omega)| = |\lambda_1 \ldots \lambda_n| = |\lambda_1| \ldots |\lambda_n|$$

Donde

$$\rho(H_\omega) = \max_{1 \leq i \leq n} |\lambda_i| \geq |\omega - 1|$$

Portanto $\rho(H_\omega) < 1$ implica $\omega \in]0, 2[$. Podemos deste modo enunciar a seguinte condição necessária para a convergência do método SOR:

Teorema 8.3 *Se o método SOR converge para a solução do sistema $Ax = b$ qualquer que seja o vector inicial $x^{(0)}$, então $0 < \omega < 2$.*

A convergência do método SOR para $\omega \in]0, 2[$ e do método de Jacobi está relacionada com a classe a que pertence a matriz do sistema. Esta questão irá ser analisada seguidamente.

8.2 Convergência dos métodos iterativos básicos

Teorema 8.4 *Seja $A = M - N$ com M e A não singulares e $H = M^{-1}N \geq 0$. Então*

$$\rho(H) < 1 \Leftrightarrow A^{-1}N \geq 0$$

Além disso, tem-se

$$\rho(H) = \frac{\rho(A^{-1}N)}{1 + \rho(A^{-1}N)} \tag{8.19}$$

Demonstração: [$\Rightarrow$] Se $A = M - N$ e $H = M^{-1}N$, então

$$\begin{aligned} A^{-1}N &= (M - N)^{-1}N = \left[M(I - M^{-1}N)\right]^{-1} N \\ &= (I - M^{-1}N)^{-1} M^{-1} N = (I - H)^{-1} H \end{aligned} \tag{8.20}$$

com I a matriz identidade. Se $\rho(H) < 1$, então dos teoremas 2.3 e 2.4 e de (8.20) vem

$$A^{-1}N = \sum_{k=1}^{\infty} H^k \geq 0$$

ficando assim demonstrada a implicação [$\Rightarrow$].

[$\Leftarrow$] Seja $A^{-1}N \geq 0$. Como $H \geq 0$, então pelo teorema 2.2 existe $0 \neq x \geq 0$ tal que $Hx = \rho(H)x$. Mas de (8.20) vem

$$0 \leq A^{-1}Nx = (I - H)^{-1}Hx \tag{8.21}$$

e

$$(I - H)x = x - Hx = x - \rho(H)x = (1 - \rho(H))x$$

ou ainda

$$(I - H)^{-1}x = \frac{1}{1 - \rho(H)}x$$

Portanto, de (8.21) vem

$$\begin{aligned} A^{-1}Nx &= (I - H)^{-1}Hx = \rho(H)(I - H)^{-1}x \\ &= \frac{\rho(H)}{1 - \rho(H)}x \geq 0 \end{aligned} \tag{8.22}$$

Como $0 \neq x \geq 0$, então $\rho(H) < 1$, o que demonstra a implicação [$\Leftarrow$].

Para demonstrar a igualdade (8.19), notemos que (8.22) implica que

$$\frac{\rho(H)}{1 - \rho(H)}$$

é valor próprio de $A^{-1}N$. Portanto

$$\rho(A^{-1}N) \geq \frac{\rho(H)}{1 - \rho(H)}$$

ou seja

$$\rho(H) \leq \frac{\rho(A^{-1}N)}{1 + \rho(A^{-1}N)} \tag{8.23}$$

Por outro lado, como $A^{-1}N \geq 0$, existe $0 \neq y \geq 0$ tal que

$$A^{-1}Ny = \rho(A^{-1}N)y$$

Mas

$$H = M^{-1}N = (A + N)^{-1}N = \left(A(I + A^{-1}N)\right)^{-1} N = (I + A^{-1}N)^{-1}A^{-1}N$$

e portanto fazendo como em (8.22) vem

$$Hy = (I + A^{-1}N)^{-1}A^{-1}Ny = \frac{\rho(A^{-1}N)}{1 + \rho(A^{-1}N)}y.$$

Donde

$$\frac{\rho(A^{-1}N)}{1 + \rho(A^{-1}N)} \leq \rho(H)$$

o que, conjuntamente com (8.23), estabelece a igualdade (8.19).

Na teoria da convergência dos métodos iterativos o conceito de partição regular tem um papel fundamental.

Definição 8.1 *Dada uma matriz A não singular, $A = M - N$ é uma Partição Regular se*

$$A = M - N,\ M^{-1} \geq 0,\ N \geq 0$$

Como consequência do teorema anterior e da definição apresentada podemos estabelecer o seguinte teorema:

Teorema 8.5 *Se $A = M - N$ é uma partição regular, então as seguintes propriedades são equivalentes:*

1. $A^{-1} \geq 0$.
2. $A^{-1}N \geq 0$.
3. $\rho(H) = \dfrac{\rho(A^{-1}N)}{1+\rho(A^{-1}N)} < 1$ *com* $H = M^{-1}N$.

Demonstração: Se $A = M - N$ é uma partição regular, então $H = M^{-1}N \geq 0$ e 2. $\Leftrightarrow$ 3. pelo teorema anterior. Além disso, como $N \geq 0$, então 1. $\Rightarrow$ 2.. Para completar a demonstração do teorema basta provar que 3. $\Rightarrow$ 1.. Mas se

$$\rho(H) = \rho(M^{-1}N) < 1$$

então

$$\begin{aligned} A^{-1} &= (M-N)^{-1} = \big(M(I - M^{-1}N)\big)^{-1} = (I-H)^{-1}M^{-1} \\ &= \sum_{k=0}^{\infty} H^k M^{-1} \geq 0 \end{aligned}$$

pois $H \geq 0$ e $M^{-1} \geq 0$. Portanto 1. é verdadeiro e isso prova o teorema.

Por este teorema, se $A = M - N$ é uma partição regular de A e $A^{-1} \geq 0$, então o método iterativo

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b \tag{8.24}$$

converge para a solução do sistema $Ax = b$. O próximo teorema investiga a rapidez de convergência de um tal processo.

Teorema 8.6 *Seja $A^{-1} \geq 0$ e $A = M_1 - N_1 = M_2 - N_2$ duas partições regulares com $N_2 \leq N_1$. Se $H_1 = M_1^{-1}N_1$ e $H_2 = M_2^{-1}N_2$, então $\rho(H_2) \leq \rho(H_1) < 1$.*

Demonstração: Como $A^{-1} \geq 0$ e $A = M_1 - N_1 = M_2 - N_2$ são partições regulares, então para $H_1 = M_1^{-1}N_1 \geq 0$ e $H_2 = M_2^{-1}N_2 \geq 0$ tem-se

$$\rho(H_1) = \frac{\rho(A^{-1}N_1)}{1+\rho(A^{-1}N_1)},\ \rho(H_2) = \frac{\rho(A^{-1}N_2)}{1+\rho(A^{-1}N_2)} \tag{8.25}$$

Além disso $A^{-1} \geq 0$ e $N_2 \leq N_1$. Portanto $A^{-1}N_2 \leq A^{-1}N_1$. Como $A^{-1}N_2 \geq 0$, então pelo teorema 2.5 vem

$$0 \leq \rho(A^{-1}N_2) \leq \rho(A^{-1}N_1) \tag{8.26}$$

Mas a função

$$f : \alpha \mapsto \frac{\alpha}{1+\alpha}$$

é crescente para $\alpha \geq 0$ e portanto de (8.25) e (8.26) obtém-se a desigualdade pretendida.

Os teoremas demonstrados nesta secção são fundamentais para o estudo da convergência dos métodos de Jacobi, Gauss-Seidel e SOR, quando a matriz A pertence às classes K e H.

(i) Matrizes K e H

Consideremos primeiramente o caso em que $A \in \mathrm{K}$. Sejam J e H_ω as matrizes correspondentes aos métodos de Jacobi e SOR respectivamente, isto é

$$\begin{aligned} J &= D^{-1}(L+U) \\ H_\omega &= (D-\omega L)^{-1}\left[(1-\omega)D+\omega U\right] \end{aligned} \tag{8.27}$$

com L, D e U dadas por (8.10) e (8.11). É então possível provar o seguinte teorema:

Teorema 8.7 *Se $A \in \mathrm{K}$, então*

1. $\rho(J) < 1$ *e* $\rho(H_\omega) < 1$ *para* $0 < \omega \leq 1$.
2. $\rho(H_\omega) \leq 1 - \omega + \omega\rho(J)$ *para* $0 < \omega \leq 1$.

Demonstração:

1. Se $A \in \mathrm{K}$ então $L + U \geq 0$ e $D^{-1} \geq 0$. Portanto

$$A = D - (L+U)$$

é uma partição regular. Além disso $A^{-1} \geq 0$ pelo teorema 3.27. Portanto, pelo teorema 8.5, $\rho(J) < 1$.

Para $0 < \omega \leq 1$, podemos escrever

$$A = \underbrace{\frac{1}{\omega}(D-\omega L)}_{M} - \underbrace{\frac{1}{\omega}\left[(1-\omega)D+\omega U\right]}_{N} \tag{8.28}$$

Mas $N \geq 0$ e $M \in \mathrm{K}$, o que implica que $M^{-1} \geq 0$, pelo teorema 3.27. Portanto (8.28) é uma partição regular e como $A^{-1} \geq 0$, tem-se

$$\rho(H_\omega) = \rho(M^{-1}N) < 1$$

2. Como $0 < \omega \leq 1$, então $H_\omega \geq 0$ e pelo teorema 2.2 existe $0 \neq x \geq 0$ tal que

$$H_\omega x = \rho(H_\omega)x \tag{8.29}$$

Seja $\lambda = \rho(H_\omega)$. Então (8.29) pode ser escrita na forma

$$\left[(1-\omega)D+\omega U\right]x = \lambda[D-\omega L]x$$

ou ainda

$$\omega(\lambda L + U)x = (\lambda + \omega - 1)Dx$$

Donde

$$D^{-1}(\lambda L+U)x = \frac{\lambda+\omega-1}{\omega}x \tag{8.30}$$

e portanto $\frac{\lambda+\omega+1}{\omega}$ é valor próprio da matriz $D^{-1}(\lambda L + U)$. Mas $0 < \lambda \leq 1$ e portanto $\lambda L + U \leq L + U$, o que implica que

$$0 \leq D^{-1}(\lambda L+U) \leq D^{-1}(L+U) = J$$

Então, pelo teorema 2.5, tem-se

$$\rho\left(D^{-1}(\lambda L+U)\right) \leq \rho(J) \tag{8.31}$$

De (8.30) e (8.31) vem

$$\frac{\lambda+\omega-1}{\omega} \leq \rho(J)$$

ou seja, $\lambda \leq 1 - \omega + \omega\rho(J)$. Como $\lambda = \rho(H_\omega)$, o resultado pretendido está provado.

O próximo teorema compara a rapidez de convergência dos métodos de Jacobi, Gauss–Seidel e SOR para $0 < \omega \leq 1$.

Teorema 8.8 *Se $A \in$ K, então*

1. $\rho(H_1) \leq \rho(J) < 1$.
2. *Se* $0 < \omega_1 < \omega_2 \leq 1$, *então*
$$\rho(H_{\omega_2}) \leq \rho(H_{\omega_1}) < 1$$

Demonstração:

1. É consequência imediata do teorema anterior.
2. Considerando a partição regular definida por (8.28), então
$$N = (\frac{1}{\omega} - 1)D + U$$
Como a função
$$f : \omega \mapsto \frac{1}{\omega}$$
é decrescente em $]0, 1]$, então $0 < \omega_1 < \omega_2 \leq 1$, vem $N_{\omega_2} \leq N_{\omega_1}$. Então, pelo teorema 8.6, tem-se $\rho(H_{\omega_2}) \leq \rho(H_{\omega_1})$.

Devido a este teorema, o método de Gauss–Seidel é preferível ao método de Jacobi e ao método SOR quando $0 < \omega < 1$. Portanto o método de Jacobi e o método SOR com $0 < \omega < 1$ não são normalmente usados quando $A \in$ K. Por outro lado, $\rho(H_\omega)$ é uma função contínua de ω, pois para $0 < \omega < 2$, H_ω existe sempre e $\rho(H_\omega)$ pode ser calculado. Como $\rho(H_\omega)$ é decrescente para $0 < \omega \leq 1$, então existe $\alpha > 1$ tal que $\rho(H_\omega) < 1$ para $0 < \omega \leq \alpha$ e para $1 < \omega \leq \alpha$ o método SOR pode ser mais eficiente que o método de Gauss-Seidel. O próximo teorema determina um valor para α.

Teorema 8.9 *Se $A \in$ K, então*

$$\rho(H_\omega) \leq \omega - 1 + \omega\rho(J) < 1$$

para

$$1 < \omega < \frac{2}{1 + \rho(J)}$$

Demonstração: Consideremos a partição (8.28). Então $H_\omega = M^{-1}N$ é uma matriz não positiva. Seja
$$\bar{N} = \frac{1}{\omega}\left[(\omega - 1)D + \omega U\right]$$
e $\bar{H}_\omega = M^{-1}\bar{N}$. Então $\bar{H}_\omega \geq 0$ e $|H_\omega| = \bar{H}_\omega$. Mas pelo teorema 2.2 existe um vector $x \geq 0$ tal que
$$\bar{H}_\omega x = \lambda x$$
com $\lambda = \rho(\bar{H}_\omega)$. Donde
$$\lambda(D - \omega L)x = \left[(\omega - 1)D + \omega U\right] x$$
ou seja
$$D^{-1}(U + \lambda L)x = \frac{\lambda + 1 - \omega}{\omega} x$$

Portanto $\frac{\lambda+1-\omega}{\omega}$ é valor próprio de $D^{-1}(U+\lambda L)$ e deste modo

$$\frac{\lambda+1-\omega}{\omega} \leq \rho\left(D^{-1}(U+\lambda L)\right) \tag{8.32}$$

Mas, se $\lambda \geq 1$, então $U+\lambda L \leq \lambda(U+L)$ e

$$D^{-1}(U+\lambda L) \leq \lambda D^{-1}(U+L) = \lambda J$$

Então pelo teorema 2.5 vem

$$\frac{\lambda+1-\omega}{\omega} \leq \lambda\rho(J)$$

Donde

$$\omega \geq \frac{1+\lambda}{1+\lambda\rho(J)} \geq \frac{2}{1+\rho(J)}$$

pois a função

$$f : \alpha \mapsto \frac{1+\alpha}{1+\beta\alpha},\ 0<\beta<1$$

é crescente. Mas, por hipótese,

$$\omega < \frac{2}{1+\rho(J)}$$

e portanto $\lambda \leq 1$. Donde

$$\rho(D^{-1}(U+\lambda L)) \leq \rho(J)$$

e substituindo em (8.32) obtém-se

$$\lambda \leq (\omega-1)+\omega\rho(J)$$

Além disso, para $0<\omega<\frac{2}{1+\rho(J)}$, tem-se

$$\begin{aligned} \lambda &\leq \omega(1+\rho(J))-1 \\ &< \frac{2}{1+\rho(J)}(1+\rho(J))-1 = 1 \end{aligned}$$

Como $|H_\omega| = \bar{H}_\omega$, então, pelo teorema 2.5, $\rho(H_\omega) \leq \lambda$ e isso demonstra o teorema.

Dos teoremas 8.7 e 8.9 podemos concluir que, se $A \in$ K, então

$$\left.\begin{array}{ll} \rho(H_\omega) < 1 & \text{para } \ 0<\omega<\dfrac{2}{1+\rho(J)} \\ \rho(H_\omega) \leq |\omega-1|+\omega\rho(J) & \end{array}\right\} \tag{8.33}$$

Além disso o método de Jacobi é menos eficiente que o método de Gauss-Seidel ($\omega = 1$). O método SOR só deve ser usado no caso de $\omega \geq 1$. A não ser em casos muito especiais não é possível determinar o valor de ω para o qual o método SOR é mais eficiente. Contudo, sabemos que em princípio esse valor pertence ao intervalo $[1, \frac{2}{1+\rho(J)}[$. E dizemos em princípio porque a condição apresentada no teorema anterior é suficiente , e como sabemos ω pode variar no intervalo $]0,2[$. Aliás, é possível encontrar matrizes K para as quais o método SOR converge com $\omega > \frac{2}{1+\rho(J)}$.

Seguidamente estudamos o caso em que $A \in$ H. Como vimos anteriormente, a matriz A é H se a matriz companheira $M(A)$ é K. Consideremos o sistema

$$M(A)x = b \tag{8.34}$$

Como $M(A) \in \mathrm{K}$, então, por (8.33) tem-se

$$\begin{cases} \rho(\bar{H}_\omega) < 1 \ \text{ para } \ 0 < \omega < \dfrac{2}{1+\rho(\bar{J})} \\ \rho(\bar{J}) < 1 \ \text{ e } \rho(\bar{H}_\omega) < |\omega - 1| + \omega\rho(\bar{J}) \end{cases} \tag{8.35}$$

onde $\bar{J}$ e $\bar{H}_\omega$ são respectivamente as matrizes dos métodos de Jacobi e SOR para a resolução do sistema (8.34). Se J e H_ω forem respectivamente as matrizes dos métodos de Jacobi e SOR associadas à matriz A, então tem-se

$$|J| = \bar{J}, \quad |H_\omega| = \bar{H}_\omega$$

e tendo em conta o teorema 2.5 e as fórmulas (8.35) podemos enunciar o seguinte teorema:

Teorema 8.10 *Se $A \in \mathrm{H}$, então*

1. $\rho(J) < 1$.
2. $\rho(H_\omega) \leq |\omega - 1| + \omega\rho(|J|) < 1$ *para* $0 < \omega < \frac{2}{1+\rho(|J|)}$

Como as matrizes estritamente diagonalmente dominantes por linhas e por colunas são matrizes H, então o teorema 8.10 é também válido para essas classes de matrizes.

Podemos assim afirmar que os métodos de Jacobi e Gauss-Seidel são convergentes quando $A \in \mathrm{H}$, e portanto quando A é estritamente diagonalmente dominante. O método SOR converge quando $\omega \in]0, \frac{2}{1+\rho(J)}[$, podendo convergir fora desse intervalo para valores superiores ou iguais ao extremo superior do intervalo e inferiores a 2. Em relação à eficiência do método SOR, não existem em geral resultados comparativos, mas, tal como nas matrizes K, a experiência tem mostrado que valores de $\omega \geq 1$ dão normalmente lugar a um menor número de iterações. Além disso o método de Jacobi é usualmente mais lento que o método SOR.

(ii) Matrizes simétricas positivas definidas

Na sequência deste estudo sobre a convergência dos métodos iterativos vamos agora analisar o caso em que $A \in \mathrm{SPD}$. Para isso, precisamos do seguinte resultado:

Teorema 8.11 *Se $A \in \mathrm{SPD}$ e H é uma matriz tal que $A - H^T AH \in \mathrm{SPD}$, então $\rho(H) < 1$.*

Demonstração: Seja λ um valor próprio qualquer de H e provemos que $|\lambda| < 1$. Suponhamos que u é o vector próprio de H correspondente ao valor próprio λ. Então, atendendo a que $A - H^T AH \in \mathrm{SPD}$, vem

$$\begin{aligned} u^T(A - H^T AH)u &= u^T Au - u^T H^T AHu \\ &= u^T Au - (Hu)^T A(Hu) \\ &= u^T Au - (\lambda u)^T A(\lambda u) \\ &= \left[1 - |\lambda|^2\right] u^T Au > 0 \end{aligned}$$

Como $A \in \mathrm{SPD}$ e $u \neq 0$, então $u^T Au > 0$ e portanto $|\lambda|^2 < 1$, ou seja, $|\lambda| < 1$.

Na análise da convergência dos métodos iterativos básicos, o conceito de partição P-regular desempenha um papel fundamental. Esse conceito é introduzido a seguir.

Definição 8.2 *Diz-se que $A = M - N$ é uma Partição P–Regular se $M + N \in$ PD.*

O próximo teorema mostra que, se $A \in$ SPD e $A = M - N$ é uma partição P–regular, então o método

$$x^{(k+1)} = M^{-1}Nx^{(k)} + M^{-1}b$$

é convergente.

Teorema 8.12 *Se $A \in$ SPD e $A = M - N$ é uma partição P–regular, então*

$$\rho(M^{-1}N) < 1$$

Demonstração: Devido ao teorema 8.11, basta provar que

$$Q = A - (M^{-1}N)^T A(M^{-1}N) \in \text{SPD}$$

Como $M^{-1}N = M^{-1}(M - A) = I - M^{-1}A$, então

$$\begin{aligned} Q &= A - (M^{-1}N)^T A(M^{-1}N) = A - (I - M^{-1}A)^T A(I - M^{-1}A) \\ &= A - \left[A - (M^{-1}A)^T A + (M^{-1}A)^T A(M^{-1}A) - A(M^{-1}A)\right] \\ &= (M^{-1}A)^T A - (M^{-1}A)^T A(M^{-1}A) + A(M^{-1}A) \\ &= A^T M^{-T} A + AM^{-1}A - (M^{-1}A)^T A(M^{-1}A) \\ &= A^T M^{-T} M M^{-1} A + AM^{-T} M^T M^{-1} A - (M^{-1}A)^T A(M^{-1}A) \\ &= (M^{-1}A)^T \left[M + M^T - A\right](M^{-1}A) \end{aligned}$$

Como $M^{-1}A$ é não singular, então $(M^{-1}A)x \neq 0$ para qualquer vector $x \neq 0$ e portanto $x^T Qx = y^T(M + M^T - A)y$, com $y = (M^{-1}A)x \neq 0$. Mas

$$y^T(M + M^T - A)y = y^T(M^T + N)y$$

e portanto tem de se provar que $y^T(M^T + N)y > 0$ para qualquer vector $y \neq 0$.

Como $A = M - N$ é uma partição P–regular, então $M + N \in$ PD e portanto $y^T(M + N)y > 0$, ou seja,

$$y^T My > -y^T Ny$$

Mas

$$y^T My = (y^T My)^T = y^T M^T y$$

e substituindo na desigualdade anterior, vem

$$y^T M^T y > -y^T Ny$$

ou ainda

$$y^T(M^T + N)y > 0$$

o que estabelece o resultado pretendido.

Como consequência deste teorema podemos estudar as convergências dos métodos de Jacobi e SOR para matrizes SPD. Em relação ao primeiro método pode-se estabelecer o seguinte resultado:

Teorema 8.13 *Se $A \in$ SPD e $(2D - A) \in$ SPD, então $\rho(J) < 1$.*

Demonstração: Como

$$A = \underbrace{D}_{M} - \underbrace{(D - A)}_{N}$$

e

$$M + N = D + D - A = 2D - A$$

então o resultado é uma consequência imediata do teorema anterior.

Contrariamente ao que se passa com as outras classes de matrizes aqui estudadas, o método de Jacobi nem sempre converge quando $A \in$ SPD. Por outro lado, o método SOR converge para todo o valor de ω pertencente ao intervalo $]0, 2[$, conforme mostra o seguinte teorema:

Teorema 8.14 *Se $A \in$ SPD e $0 < \omega < 2$, então $\rho(H_\omega) < 1$.*

Demonstração: Se $A \in$ SPD, então A é simétrica e portanto $U = L^T$ e a partição (8.28) tem a forma

$$A = \underbrace{\frac{1}{\omega}(D - \omega L)}_{M} - \underbrace{\frac{1}{\omega}\left[(1-\omega)D + \omega L^T\right]}_{N}$$

e $H_\omega = M^{-1}N$. Para estabelecer o resultado temos apenas que provar que a partição anterior é P–regular, isto é, que $M + N \in$ PD. Mas

$$M + N = \frac{1}{\omega}\left[(2-\omega)D - \omega(L - L^T)\right]$$

e portanto, para qualquer $x \neq 0$,

$$x^T(M+N)x = \frac{1}{\omega}(2-\omega)x^T Dx - \omega x^T(L - L^T)x$$

Por outro lado,

$$x^T(L - L^T)x = x^T Lx - x^T L^T x = x^T Lx - \left(x^T L^T x\right)^T = x^T Lx - x^T Lx = 0$$

Portanto

$$x^T Mx = \frac{1}{\omega}(2-\omega)x^T Dx > 0$$

pois D é uma matriz diagonal de elementos diagonais positivos.

Por este teorema, o método SOR é convergente em todo o intervalo $\omega \in]0, 2[$ quando $A \in$ SPD. Tal como para as outras classes de matrizes, a não ser em casos muito especiais, não é conhecido teoricamente um valor óptimo para o parâmetro ω. A experiência tem mostrado que valores de ω maiores do que um conduzem normalmente a um menor número de iterações.

É ainda de notar que o teorema anterior mostra que os resultados obtidos para as matrizes K e diagonalmente dominantes em que o método SOR converge são apenas condições suficientes. Com efeito, se $A \in$ K ou $A \in$ H$_+$ (SCDD$_+$) e é simétrica, então $A \in$ SPD e portanto ω pode ser superior aos valores referidos nos teoremas 8.9 e 8.10.

8.3 Características dos métodos iterativos básicos

Nesta secção iremos abordar os problemas da implementação e da utilidade dos métodos discutidos nas duas secções anteriores tanto para matrizes densas como para matrizes esparsas.

Se a matriz A do sistema é densa, então o espaço de armazenagem total para a implementação do método SOR (em particular para o método de Gauss–Seidel) é de $n^2 + 2n$. Com efeito, a matriz A necessita de n^2 elementos para ser armazenada, enquanto que os vectores b e $x^{(k)}$ são armazenados como vectores densos de ordem n.

Se A é esparsa, então pode ser armazenada usando um esquema de colecção de vectores. Se A tiver estrutura de banda ou tiver um invólucro relativamente pequeno, então poder-se-à tirar partido disso optando por esquemas de banda e banda variável respectivamente. De notar que o método de Jacobi requer adicionalmente a armazenagem de um vector denso, pois há que considerar em cada iteração k a armazenagem dos vectores $x^{(k)}$ e $x^{(k+1)}$.

Os métodos iterativos têm suscitado grande interesse devido à simplicidade das suas implementações tanto no caso denso como no caso esparso. Com efeito, o algoritmo 38 é muito fácil de implementar se a matriz A é densa. Isso também acontece no caso de A ser uma matriz esparsa não simétrica, se se usar um esquema de colecção de vectores orientado por linhas. Assim, por exemplo, se A está armazenada por linhas usando os vectores DIAGA, VALA, INDA e PNTA cujos significados foram explicados no capítulo 6, então a implementação do algoritmo SOR tem a forma apresentada na figura 8.3, onde se incluiu o critério de paragem(8.5) com a norma $||\cdot||_\infty$.

Algoritmo 43

Seja X um vector inicial, ε uma tolerância para zero e ERRO um número maior que ε.

Enquanto ERRO $> \varepsilon$ faça
- ERRO $= 0$
- NORMAX $= 0$
- Para $i = 1, \ldots, n$
  - INICIO $=$ PNTA(i)
  - FIM $=$ PNTA$(i+1) - 1$
  - Se INICIO $\leq$ FIM faça
    - SOMA $= b(i)$
    - Para $k =$ INICIO$, \ldots,$ FIM
      - SOMA $=$ SOMA $-$ VALA$(k) \times X($INDA$(k))$
    - SOMA $=$ SOMA$/$DIAGA(i)
  - aux $= \omega \times$ SOMA $+ (1 - \omega) \times X(i)$
  - Se $|$aux$| >$ NORMAX , faça NORMAX $=$ $|$aux$|$
  - Se $|$aux $- X(i)| >$ ERRO, faça ERRO $= |$aux $- X(i)|$
  - $X(i) =$ aux
- aux $= 1$
- Se NORMAX > 1, faça aux $=$ NORMAX
- ERRO $=$ ERRO$/$aux

Figura 8.3: Implementação do método SOR

Considere-se agora o caso de A ser uma matriz esparsa SPD. Como vimos anteriormente, apenas a diagonal e a parte abaixo da diagonal são armazenadas, pelo que o método SOR tem de ser implementado numa forma que não seja baseada no algoritmo 43. Mas de (8.18) tem-se

$$(D - \omega L)\, x^{k+1} = \omega b + (1 - \omega) D x^{(k)} + \omega L^T x^{(k)}$$

Portanto a determinação do vector x na iteração k é feita resolvendo um sistema triangular com a matriz $(D - \omega L)$. O termo independente desse sistema é fácil de obter usando a estrutura de dados que armazena A, o vector $x^{(k)}$ da iteração anterior e um algoritmo para o produto de uma matriz esparsa por um vector denso. Note-se que se A está armazenada por linhas, então $L^T x^{(k)}$ deve ser calculado por um processo orientado por colunas e o sistema triangular é resolvido por linhas. Os processos para cumprir esses objectivos foram descritos no capítulo 6, pelo que a implementação é deixada como exercício.

Se A é uma matriz densa de ordem n, então cada iteração dos métodos de Jacobi e Gauss-Seidel requer n^2 operações, enquanto que são necessárias $n(n+2)$ operações para o algoritmo SOR. Podemos então dizer que os métodos iterativos requerem um número de operações inferior aos métodos directos se tiverem que efectuar um número de iterações inferior ou igual a $\frac{1}{3}n$ ($\frac{1}{6}n$ no caso simétrico).

Se a matriz A é esparsa com η elementos não nulos, então os métodos de Jacobi e de Gauss-Seidel requerem $\eta + n$ operações por iteração, necessitando o método SOR de efectuar $\eta + 3n$ operações por iteração.

Em qualquer método iterativo o número de iterações a efectuar cresce com o grau de precisão desejada para a solução do sistema. Além disso, o número de iterações a efectuar depende também da norma de $||x^0 - x^*||$, com x^0 o vector inicial e x^* a solução do sistema.

As considerações apresentadas nesta secção permitem-nos concluir sobre a maior ou menor utilidade dos métodos iterativos em relação aos métodos directos. Assim, se a matriz A é densa de dimensão pequena, então os métodos directos são usualmente vantajosos, pelas seguintes razões:

1. A implementação do método directo é relativamente simples.

2. Do ponto de vista numérico, obtém-se usualmente uma boa solução. Tal não acontecerá se a matriz for muito mal condicionada, mas neste caso os métodos iterativos têm imensa dificuldade em convergir.

3. O número de iterações dos métodos iterativos está dependente da aproximação inicial que é fornecida pelo utilizador. Como não há em geral qualquer conhecimento sobre a solução do sistema, torna-se difícil encontrar uma boa solução inicial. Isso implica que em geral o método iterativo requer demasiadas iterações, o que o impede de ser vantajoso em relação aos métodos directos.

Por outro lado, se A é esparsa e de grande dimensão, então os métodos iterativos poderão em alguns casos ser vantajosos, devido às razões que a seguir expomos:

1. As implementações dos métodos directos são bastante mais complexas que as dos métodos iterativos.

2. Em alguns casos, poderão ocorrer muitos enchimentos, no decurso da fase de factorização, o que acarreta um aumento dos requerimentos de armazenagem e do número de operações. Nos métodos iterativos não há lugar a enchimentos, por não se efectuarem decomposições.

Do atrás exposto não se pode de modo nenhum concluir que os métodos iterativos são sempre preferíveis no caso de a matriz A ser esparsa e de grande dimensão. Com efeito, os métodos directos podem ser muito mais aconselháveis mesmo neste último caso. Isso acontece por exemplo se a matriz possuir uma estrutura de banda ou um invólucro bastante pequeno. Assim, se A é tridiagonal de ordem n, então os métodos directos requerem um espaço de armazenagem igual a $4n - 2$ e $5n - 4$ operações, ao passo que os métodos iterativos requerem um espaço de $5n - 2$, e $3n - 2$ operações por iteração. Assim o método directo requer menos espaço de armazenagem que o método iterativo. Além disso basta que se tenham que efectuar duas iterações do método iterativo para que o número de operações seja superior ao do método directo. Os métodos directos

são igualmente preferíveis quando é exigida uma precisão muito grande e a matriz é relativamente mal condicionada (desde que os requerimentos de armazenagem não sejam muito elevados).

Há contudo situações em que os métodos iterativos são preferíveis, nomeadamente quando se verificarem as seguintes condições:

1. A matriz A não é mal condicionada, ou então a precisão exigida é pequena.
2. É conhecida uma aproximação inicial $x^{(0)}$ suficientemente próxima de x^*.

Por essa razão os métodos iterativos têm sido usados isoladamente em processos sequenciais como o método de Newton para a resolução de sistemas de equações não lineares. Nestes casos, à medida que o processo se aproxima do fim, é fácil de obter boas soluções aproximadas iniciais para esses sistemas, o que torna o número de iterações do método iterativo bastante pequeno.

8.4 Método da Partição

Dado o sistema $Ax = b$, consideremos uma partição de A da forma

$$A = B - C \tag{8.36}$$

em que B é uma matriz não singular escolhida de modo a que um sistema envolvendo a matriz B seja relativamente fácil de resolver. Como mencionámos anteriormente, essa partição conduz ao esquema iterativo

$$x^{(k)} = B^{-1}Cx^{(k-1)} + B^{-1}b$$

obtendo-se assim o chamado Método da Partição, que se pode formular do modo seguinte:

Algoritmo 44 (Método da Partição)

Dada uma aproximação inicial $x^{(0)}$

Para $k = 1, 2, \ldots$

Resolver o sistema $Bx^{(k)} = Cx^{(k-1)} + b$

Se *Critério de paragem é satisfeito*

Então *Termine.*

É evidente que os algoritmos referidos nas secções anteriores são métodos de partição. Assim por exemplo, o método de Jacobi é o método de partição em que B é exactamente a diagonal da matriz A. Nesta secção estamos no entanto interessados em considerar escolhas mais gerais para a matriz B.

O teorema 8.1 fornece uma condição necessária e suficiente para a convergência do método da partição, nomeadamente, o algoritmo é convergente se e só se

$$\rho(B^{-1}C) < 1. \tag{8.37}$$

Contudo, a verificação dessa condição não é fácil de averiguar na prática. Seguidamente apresentamos algumas condições suficientes para a convergência do processo. Sugerimos [Axelsson] para uma discussão de outros resultados deste tipo.

Teorema 8.15 *Seja* $A \in \mathrm{K}$ *e* (B, C) *uma partição de* A *tal que*

(i) $b_{ii} = a_{ii}$, $i = 1, \ldots, n$.

(ii) $b_{ij} \neq 0 \Rightarrow c_{ij} = 0$, *para* $i \neq j$.

Então o método da partição converge para a solução do sistema $Ax = b$, *qualquer que seja o vector inicial* $x^{(0)}$.

Demonstração: Pelo teorema 3.27, se $A \in \mathrm{K}$, então $A^{-1} \geq 0$. Além disso, as hipóteses do teorema implicam que $b_{ij} = a_{ij}$ ou $b_{ij} = 0$ para todo o (i, j) e portanto $B \geq A$. Como $A \in \mathrm{K} = \mathrm{Z} \cap \mathrm{S}$, o mesmo acontece com B. Portanto $B^{-1} \geq 0$ pelo teorema 3.27. Finalmente $C \geq 0$ pelas hipóteses do teorema. Então, pelo teorema 8.5, $\rho(B^{-1}C) < 1$ e o algoritmo é convergente.

Este teorema mostra que se $A \in \mathrm{K}$, então há uma grande liberdade na escolha das matrizes B e C. Isso é muito importante na prática, pois permite tirar partido da possível estrutura da matriz A na construção da partição (B, C).

Consideremos agora o caso de $A \in \mathrm{SPD}$. Então da definição apresentada e do teorema 8.12 podemos concluir o seguinte resultado.

Teorema 8.16 *Se* $A \in \mathrm{SPD}$ *e* $B + C \in \mathrm{PD}$ *então o método da partição converge para a solução do sistema* $Ax = b$, *qualquer que seja o vector inicial* $x^{(0)}$.

Como consequência deste teorema podemos ainda estabelecer a convergência do método da partição para matrizes H_+ simétricas.

Teorema 8.17 *Se* $A \in \mathrm{H}_+$ *e é simétrica, e* B *uma matriz simétrica contendo a diagonal de* A, *então o método da patrição converge para a solução do sistema* $Ax = b$, *qualquer que seja o vector inicial* $x^{(0)}$.

Demonstração: Se $A = B - C \in \mathrm{H}$ é simétrica e tem elementos diagonais positivos, então, pelo teorema 3.32, $A \in \mathrm{P}$ e portanto $A \in \mathrm{SPD}$ pelo teorema 4.11. Por outro lado, a escolha de B implica que $B + C \in \mathrm{H}_+$. Então pelo teorema 3.31, $B + C \in \mathrm{P}$. Como $B + C$ é simétrica tem-se $B + C \in \mathrm{PD}$ e o resultado pretendido é uma consequência do teorema 8.16.

Portanto e à semelhança das matrizes K há uma grande liberdade na escolha da partição (B, C) quando A é uma matriz simétrica H, e em particular quando A é uma matriz estritamente diagonalmente dominante, com elementos diagonais positivos.

Seguidamente iremos discutir um processo de construir a partição (B, C) quando A é SPD. Para esse efeito seja R uma matriz não negativa e simétrica e D uma matriz diagonal de elementos diagonais positivos tal que

$$Dv > Rv \tag{8.38}$$

para um certo vector $v > 0$. Então

$$A = D + A - R - (D - R)$$

e portanto podemos escolher

$$B = D + A - R, \; C = D - R \tag{8.39}$$

Devido à condição (8.38) e às definições das matrizes D e R, tem-se

$$C = D - R \in \mathrm{Z} \cap \mathrm{S} = \mathrm{K}$$

Além disso C é simétrica e portanto $C \in \mathrm{PD}$. Donde

$$B = A + C \in \mathrm{PD}$$

e portanto o método da partição é convergente para as escolhas de B e C apresentadas em (8.39). É de notar que se R contém todos os elementos positivos de A então $B \in \mathrm{K}$. Na prática, uma

vez escolhida a matriz R é fácil de encontrar uma matriz D de elementos diagonais positivos que satisfaça a condição (8.38) e assim obter a partição pretendida. Este processo é normalmente denominado por Compensação na Diagonal, devido ao facto de a matriz D compensar de certa forma os elementos de A que não ficam em B.

A eficiência do método da partição depende da escolha da matriz B. Assim, por um lado B tem de ser relativamente simples para que o sistema

$$Bx^{(k)} = Cx^{(k-1)} + b$$

se possa resolver por um método directo. Por outro lado, se B é muito diferente de A, então o algoritmo pode requerer muitas iterações para obter uma solução com alguma precisão. O teorema 8.6 mostra que se $A \in \mathrm{K}$ o número de iterações varia na razão inversa do número de não zeros que são considerados em B. Portanto o método de Jacobi é nesse sentido o pior método de partição quando $A \in \mathrm{K}$. Apesar de não ser possível estabelecer um resultado semelhante para matrizes SPD e H_+, a prática indica que em geral este resultado é também válido. É portanto boa política tentar que B tenha o maior número possível de elementos de A, mas que ao mesmo tempo os enchimentos de B sejam muito menores que os enchimentos de A. Isso torna simples a resolução do sistema com a matriz B usando um método directo e ao mesmo tempo torna o número total de iterações relativamente pequeno. Apesar de existirem alguns processos de determinar B desse modo (ver por exemplo [Patricio]) não nos iremos debruçar sobre esse assunto.

8.5 Método dos Gradientes Conjugados

Consideremos o sistema $Ax = b$, com $A \in \mathbb{R}^{n\times n}$ não singular. Vamos começar esta secção introduzindo o conceito de vectores A–conjugados. Para simplificação de notação iremos nesta secção representar vectores na forma x_k em vez da notação usual $x^{(k)}$.

Definição 8.3 *Os vectores $p_0, p_1, \ldots, p_{m-1} \in \mathbb{R}^n$ são A-conjugados se*

$$p_i^T A p_j = 0, i \neq j$$

Como consequência da definição, vectores ortogonais são I-conjugados, sendo I a matriz identidade. O próximo resultado mostra que podemos resolver um sistema de equações lineares usando uma sequência finita de vectores A-conjugados.

Teorema 8.18 (Teorema das direcções conjugadas) *Seja $A \in \mathbb{R}^{n\times n}$ uma matriz* SPD*, e consideremos a sucessão $\{x_k\}_{k=0,\ldots,n-1}$ de vectores de $\mathbb{R}^n$ definida por*

$$x_{k+1} = x_k - \alpha_k p_k, \quad k = 0, \ldots, n-1$$

com p_k $(k = 0, 1, \ldots, n)$ vectores A-conjugados e

$$\alpha_k = \frac{p_k^T r_k}{p_k^T A p_k} \quad (r_k = Ax_k - b) \tag{8.40}$$

Então o vector x_n é a solução do sistema $Ax = b$.

Demonstração: Comecemos por provar que os vectores $p_0, p_1, \ldots, p_{n-1}$ são linearmente independentes, isto é,

$$\alpha_0 p_0 + \ldots + \alpha_{n-1} p_{n-1} = 0 \Rightarrow \alpha_i = 0, \quad i = 0, 1, \ldots, n-1$$

Com efeito tem-se

$$\sum_{k=0}^{n-1} \alpha_k p_k = 0 \Rightarrow \sum_{k=0}^{n-1} \alpha_k A p_k = A\left(\sum_{k=0}^{n-1} \alpha_k p_k\right) = 0$$

Multiplicando ambos os membros por p_i^T tem-se

$$\sum_{k=0}^{n-1} \alpha_k p_i^T A p_k = 0 \Rightarrow \alpha_i p_i^T A p_i = 0 \Rightarrow \alpha_i = 0$$

pois $A \in$ SPD. Então $\{p_0, p_1, \ldots, p_{n-1}\}$ é uma base de $\mathbb{R}^n$. Se x_* é a solução única do sistema $Ax = b$, então existem escalares $\beta_0, \beta_1, \ldots, \beta_{n-1}$ tais que

$$x_* - x_0 = \beta_0 p_0 + \ldots + \beta_{n-1} p_{n-1}$$

e

$$p_i^T A(x_* - x_0) = \beta_i p_i^T A p_i$$

Como $A \in$ SPD tem-se $p_i^T A p_i > 0$ e

$$\beta_i = \frac{p_i^T A(x_* - x_0)}{p_i^T A p_i}$$

Mas se $\{x_k\}$, $k = 0, 1, \ldots, n$ é o conjunto de vectores gerado pelo método, então

$$x_i - x_0 = -\left(\alpha_0 p_0 + \ldots + \alpha_{i-1} p_{i-1}\right)$$

Portanto

$$p_i^T A(x_i - x_0) = 0$$

e

$$\beta_i = \frac{p_i^T A(x_* - x_i) + p_i^T A(x_i - x_0)}{p_i^T A p_i} = \frac{p_i^T (b - A x_i)}{p_i^T A p_i} = -\frac{p_i^T r_i}{p_i^T A p_i} = -\alpha_i$$

Isso completa a demonstração do teorema.

Como consequência deste teorema, se $p_k, k = 0, 1, \ldots$ são direcções A-conjugadas, então a sucessão $\{x_k\}$ definida por

$$x_{k+1} = x_k - \alpha_k p_k$$

com α_k dado por (8.40), converge para a solução do sistema $Ax = b$ ($A \in$ SPD) em n iterações. Na prática, a ocorrência de erros de arredondamento torna inviável que isso aconteça. No entanto este teorema fornece a base teórica de um tipo de métodos iterativos. Com efeito, se for possível gerar uma sucessão $\{p_k\}$ de direcções A-conjugadas, então a sucessão $\{x_k\}$ definida anteriormente deve convergir para uma solução do sistema de equações $Ax = b$. O Método dos Gradientes Conjugados é um algoritmo em que as direcções A-conjugadas são definidas de um modo iterativo e tem a forma apresentada na figura 8.4.

Seguidamente iremos estabelecer a convergência deste algoritmo para matrizes SPD.

Teorema 8.19 *Se $A \in$ SPD e x_* é a solução de $Ax = b$, então os vectores p_k gerados pelo método dos gradientes conjugados são A-conjugados, isto é, satisfazem*

$$p_k^T A p_j = 0, \quad 0 \leq j < k, \quad k = 1, \ldots, n-1 \tag{8.41}$$

Além disso $x_* = x_m$ para algum $m \leq n$.

Demonstração: Provemos (8.41) e

$$r_k^T r_j = 0, \quad 0 \leq j < k, \quad k = 1, \ldots, n-1 \tag{8.42}$$

Algoritmo 45

Dado x_0

$p_0 = -r_0 = b - Ax_0$

Para $k = 0, 1, \ldots$

$$\alpha_k = \frac{r_k^T p_k}{p_k^T A p_k}$$

$$x_{k+1} = x_k - \alpha_k p_k$$

$$r_{k+1} = r_k - \alpha_k A p_k$$

$$\beta_k = \frac{p_k^T A r_{k+1}}{p_k^T A p_k}$$

$$p_{k+1} = -r_{k+1} + \beta_k p_k$$

Figura 8.4: Método dos Gradientes Conjugados

por indução. Para qualquer $j = 0, 1, \ldots, n-2$ tem-se

$$\begin{array}{l} p_j^T r_{j+1} = p_j^T r_j - \alpha_j p_j^T A p_j = 0 \\ p_j^T A p_{j+1} = -p_j^T A r_{j+1} + \beta_j p_j^T A p_j = 0 \end{array} \tag{8.43}$$

Como $p_0 = -r_0$ então a primeira igualdade de (8.43) mostra que (8.42) é verdadeira para $k = 1$. A segunda igualdade de (8.43) implica que (8.41) se verifica para $k = 1$.

Suponhamos que as condições (8.41) e (8.42) se verificam para $k < n - 1$ e provemos que também são verdadeiras para $k + 1$.

Para qualquer $j < k$ tem-se

$$\begin{aligned} r_j^T r_{k+1} &= r_j^T (r_k - \alpha_k A p_k) = r_j^T r_k - \alpha_k r_j^T A p_k \\ &= r_j^T r_k + \alpha_k p_k^T A (p_j - \beta_{j-1} p_{j-1}) \\ &= r_j^T r_k + \alpha_k p_k^T A p_j - \alpha_k \beta_{j-1} p_k^T A p_{j-1} \\ &= 0 \end{aligned}$$

pois $r_j^T r_k = 0$ pela hipótese de indução e as direcções p_k são A-conjugadas. Além disso

$$\begin{aligned} r_k^T r_{k+1} &= (-p_k + \beta_{k-1} p_{k-1})^T r_{k+1} \\ &= \beta_{k-1} p_{k-1}^T r_{k+1} - p_k^T r_{k+1} \end{aligned}$$

Mas por (8.43) $p_k^T r_{k+1} = 0$ e portanto

$$\begin{aligned} r_k^T r_{k+1} &= \beta_{k-1} p_{k-1}^T (r_k - \alpha_k A p_k) \\ &= \beta_{k-1} p_{k-1}^T r_k - \alpha_k \beta_{k-1} p_{k-1}^T A p_k \end{aligned}$$

Então por (8.43) tem-se $r_k^T r_{k+1} = 0$ e (8.42) é verdadeira para $k + 1$. Para estabelecer (8.41) tem-se de (8.43)

$$p_k^T A p_{k+1} = 0$$

e para $j < k$

$$\begin{aligned} p_j^T A p_{k+1} &= p_j^T A(-r_{k+1} + \beta_k p_k) \\ &= -p_j^T A r_{k+1} + \beta_k p_j^T A p_k \end{aligned}$$

Como $p_j^T A p_k = 0$ pela hipótese de indução, então por (8.42) vem

$$\begin{aligned} p_j^T A p_{k+1} &= -r_{k+1} A p_j \\ &= r_{k+1}^T (r_{j+1} - r_j) \frac{1}{\alpha_j} = 0 \end{aligned}$$

desde que $\alpha_j \neq 0$.

Provemos agora que $p_k \neq 0$ a não ser que $x_k = x_*$. Suponhamos então que $p_m = 0$ para $m < n$. Então

$$\begin{aligned} 0 = p_m^T p_m &= (-r_m + \beta_{m-1} p_{m-1})^T (-r_m + \beta_{m-1} p_{m-1}) \\ &= r_m^T r_m - 2\beta_{m-1} r_m^T p_{m-1} + \beta_{m-1}^2 p_{m-1}^T p_{m-1} \end{aligned}$$

Mas por (8.43) $r_m^T p_{m-1} = 0$ e

$$0 \geq r_m^T r_m$$

o que implica que

$$r_m = A x_m - b = 0 \Rightarrow x_m = x_*$$

Por outro lado, se $p_k \neq 0$ para todo o k, então $x_n = x_*$ pelo teorema anterior. Para demonstrar o teorema basta averiguar o caso de $\alpha_j = 0$. Mas

$$r_j^T p_j = r_j^T (-r_j + \beta_{j-1} p_{j-1}) = -r_j^T r_j$$

Portanto

$$\alpha_j = \frac{-r_j^T r_j}{p_j^T A p_j} \tag{8.44}$$

e

$$\alpha_j = 0 \Leftrightarrow r_j = 0 \Leftrightarrow x_j = x_*$$

Donde em todos os casos existe $m \leq n$ tal que $x_m = x_*$ e isso demonstra o teorema.

Note-se que os escalares β_k podem ser obtidos usando apenas os resíduos r_k e r_{k+1}. Com efeito, tem-se

$$\begin{aligned} r_{k+1}^T p_{k+1} &= (r_k - \alpha_k A p_k)^T p_{k+1} = r_k^T p_{k+1} \\ &= r_k^T (-r_{k+1} + \beta_k p_k) = \beta_k r_k^T p_k \end{aligned}$$

e como $r_j^T p_j = -r_j^T r_j$ vem

$$\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} = \frac{||r_{k+1}||^2}{||r_k||^2} \tag{8.45}$$

Esta fórmula para o cálculo de β_k tem vantagens em relação à anterior, pois permite reduzir o esforço computacional em cada iteração do algoritmo dos gradientes conjugados. Com efeito o critério de paragem para este processo é dado por $||r_{k+1}|| < \sqrt{\epsilon}$ para certa tolerância ϵ próxima da precisão da máquina ϵ_M. Se a norma ℓ_2 é usada, então esse critério tem a forma

$$\eta_k = ||r_{k+1}||_2^2 < \epsilon.$$

Algoritmo 46

Dado x_0

$p_0 = -r_0 = b - Ax_0$

Para $k = 0, 1, \ldots$

$\gamma_k = r_k^T r_k$

$v_k = Ap_k$

$\gamma_k = -\frac{\gamma_k}{p_k^T v_k}$

$x_{k+1} = x_k - \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k v_k$

Se $\eta_k = r_{k+1}^T r_{k+1} < \epsilon$ *Termine.*

Doutro modo

$\beta_k = \dfrac{\eta_k}{\gamma_k}$

$p_{k+1} = -r_{k+1} + \beta_k p_k$

Figura 8.5: Método dos gradientes conjugados (forma mais eficiente)

Portanto se $\eta_k < \epsilon$ então o método termina com a solução aproximada x_{k+1} do sistema. De outro modo, a quantidade η_k pode ser usada para calcular β_k e desse modo são poupadas algumas operações.

Tendo em conta as expressões (8.44) e (8.45) obtém-se a formulação do método dos gradientes conjugados que é apresentada na figura 8.5.

Na escolha do critério de paragem tem que se ter em consideração a questão da escolha do parâmetro ϵ. Usualmente considera-se $\epsilon \approx \epsilon_M$. Como referimos anteriormente, além do critério de paragem referido, poderá ser vantajoso considerar um critério da forma

$$\frac{||x_{k+1} - x_k||_\infty}{\max\{1, ||x_{k+1}||_\infty\}} \leq \epsilon$$

com $\epsilon \approx \sqrt{\epsilon_M}$. Com efeito, se este critério for satisfeito, então o algoritmo está a convergir para uma solução. De acordo com a discussão apresentada na secção 8.1, o uso deste critério poderá ter vantagens quando A é mal condicionada e a verificação do critério do resíduo só ser possível para valores bastante superiores à tolerância ϵ escolhida.

Iremos agora fazer incidir a nossa atenção sobre alguns detalhes da implementação. Note-se que este método apenas requer produtos de matrizes por vectores, produtos escalares e adições de vectores. Esse facto reflecte-se no tipo de armazenagem a considerar para a matriz do sistema e para os vários vectores envolvidos. Assim:

1. Se o problema tem dimensão reduzida, a matriz A deverá ser armazenada como densa.
2. Se a matriz tem uma estrutura de banda ou um invólucro pequeno, então a implementação do algoritmo deverà tirar partido desse facto.
3. Para problemas de dimensão elevada, a matriz deverá ser armazenada num esquema de colecção de vectores com diagonal separada.
4. Os vectores b, x_k, r_k, v_k e p_k são armazenados como vectores densos, requerendo-se para eles um espaço de armazenagem igual a $5n$.

Seguidamente iremos investigar a taxa de convergência do método dos gradientes conjugados, ou seja, iremos observar com que velocidade a sucessão $\{x_k\}$ de vectores gerada pelo método converge para a solução. O teorema seguinte mostra que essa convergência é linear.

Teorema 8.20 *Consideremos o sistema $Ax = b$, com $A \in$ SPD e solução única x_*. Se $\{x_k\}$ é a sucessão de vectores gerada pelo método dos gradientes conjugados, então*

$$||x_k - x_*||_2 < ||x_{k-1} - x_*||_2$$

Demonstração: Tem-se

$$\begin{aligned} ||x_{k-1} - x_*||_2^2 &= (x_{k-1} - x_*)^T (x_{k-1} - x_*) \\ &= (x_* - x_k + x_k - x_{k-1})^T (x_* - x_k + x_k - x_{k-1}) \\ &= (x_* - x_k)^T (x_* - x_k) + 2(x_* - x_k)^T (x_k - x_{k-1}) + (x_k - x_{k-1})^T (x_k - x_{k-1}) \\ &= ||x_* - x_k||_2^2 + 2(x_* - x_k)^T (x_k - x_{k-1}) + ||x_k - x_{k-1}||_2^2 \end{aligned}$$

Se $x_{k-1} \neq x_*$, então $x_k \neq x_{k-1}$ e

$$||x_k - x_{k-1}||_2^2 > 0$$

Portanto tem de se provar que

$$(x_* - x_k)^T (x_k - x_{k-1}) \geq 0$$

Mas pelo teorema 8.19, existe um $m \leq n$ tal que $x_m = x_*$. Além disso

$$x_m - x_k = x_m - x_{m-1} + x_{m-1} - \ldots - x_{k+1} + x_{k+1} - x_k$$

Como

$$x_{j+1} - x_j = -\alpha_j p_j$$

então

$$\begin{aligned} (x_m - x_k)^T (x_k - x_{k-1}) &= -(\alpha_{m-1} p_{m-1} + \ldots + \alpha_k p_k)^T (-\alpha_{k-1} p_{k-1}) \\ &= \alpha_{k-1} (\alpha_{m-1} p_{m-1}^T p_{k-1} + \ldots + \alpha_k p_k^T p_{k-1}) \end{aligned}$$

Mas

$$\alpha_j = -\frac{r_j^T r_j}{p_j^T A p_j} \leq 0$$

Portanto para mostrar que

$$(x_* - x_k)^T (x_k - x_{k-1}) \geq 0$$

basta provar que $p_j^T p_{k-1} \geq 0$, para todo $j = k, \ldots, m-1$. Como

$$\begin{aligned} p_j &= -r_j + \beta_{j-1} p_{j-1} = -r_j + \beta_{j-1}(-r_{j-1} + \beta_{j-2} p_{j-2}) = \ldots \\ &= -[r_j + \beta_{j-1} r_{j-1} + \ldots + (\beta_{j-1} \ldots \beta_k) r_k - (\beta_{j-1} \ldots \beta_{k-1}) p_{k-1}] \end{aligned}$$

então

$$p_j^T p_{k-1} = -\left[r_j^T p_{k-1} + \beta_{j-1} r_{j-1}^T p_{k-1} + \ldots + (\beta_{j-1} \ldots \beta_k) r_k^T p_{k-1}\right] + (\beta_{j-1} \ldots \beta_{k-1}) p_{k-1}^T p_{k-1} \geq 0$$

pois a primeira parcela é nula pelo teorema anterior e a segunda parcela é não negativa devido a $\beta_i \geq 0$ para todo o i. Isso demonstra o resultado pretendido.

Este teorema mostra que o erro absoluto reduz em cada iteração, o que confirma a convergência do método para matrizes SPD. Contudo a rapidez dessa convergência depende do número de condição de A. Com efeito [Ortega] é possível provar que

$$||x_k - x_*||_A \leq 2 \left(\frac{\sqrt{\text{cond}(A)} - 1}{\sqrt{\text{cond}(A)} - 1} \right)^k ||x_0 - x_*||_A$$

onde a norma $||.||_A$ é definida por

$$||y||_A = \left(y^T A y\right)^{1/2}$$

Esta fórmula mostra que para matrizes mal condicionadas o método dos gradientes conjugados pode ser extremamente ineficiente. O precondicionamento, a introduzir na secção seguinte, é considerada a técnica mais interessante de minorar essa deficiência do algoritmo.

O método dos gradientes conjugados pode ser ainda utilizado na resolução de um sistema $Ax = b$ com A uma matriz não simétrica ou simétrica indefinida não singular. Com efeito o sistema $Ax = b$ é equivalente a

$$A^T A x = A^T b \tag{8.46}$$

e a matriz $A^T A$ é SPD, pelo que o método pode ser aplicado para obter a solução desse sistema. É de notar que a matriz $A^T A$ não precisa de ser formada na implementação desse processo. Na realidade, o algoritmo dos gradientes conjugados só precisa da soma e produtos escalares de vectores e produtos da matriz A por um vector u. Mas

$$y = A^T A u = A^T (A u)$$

e portanto o vector y pode ser calculado em duas fases a partir de

(i) $z = Au$

(ii) $y = A^T z$

É de notar que se A está armazenada por linhas, isto é, se

$$A = \begin{bmatrix} A_1 \\ \ldots \\ A_n \end{bmatrix}$$

então o vector z é calculado a partir de

$$z_i = A_i \cdot u, \quad i = 1, \ldots, n$$

enquanto que y é obtido por

$$y = z_1 A_1 + \ldots + z_n A_n$$

Apesar de simples de implementar, este processo poderá encontrar dificuldades em obter uma solução aceitável. Com efeito verifica-se o seguinte resultado.

Teorema 8.21 $\text{cond}_2(A^TA) = [\text{cond}_2(A)]^2$

Demonstração: Como A^TA é uma matriz SPD, então os seus valores próprios são todos positivos e

$$\text{cond}_2(A^TA) = \frac{\lambda_{\max}(A^TA)}{\lambda_{\min}(A^TA)}$$

onde $\lambda_{\max}(B)$ e $\lambda_{\min}(B)$ representam o maior e o menor valores próprios de B respectivamente. Por outro lado

$$\begin{aligned}\text{cond}_2(A) &= ||A||_2||A^{-1}||_2 \\ &= \sqrt{\lambda_{\max}(A^TA)}\sqrt{\lambda_{\max}(A^TA)^{-1}} \\ &= \frac{\sqrt{\lambda_{\max}(A^TA)}}{\sqrt{\lambda_{\min}(A^TA)}} = \sqrt{\frac{\lambda_{\max}(A^TA)}{\lambda_{\min}(A^TA)}}\end{aligned}$$

O resultado é agora consequência imediata destas duas igualdades.

Assim o número de condição é ainda mais importante para a obtenção de uma boa solução do sistema $Ax = b$. Por isso é imperioso o uso de técnicas de precondicionamento muito eficazes capazes de reduzir o efeito do condicionamento da matriz A. Apesar de em alguns casos ter sido possível resolver eficientemente o sistema (8.46) pelo método dos gradientes conjugados com precondicionamento, em geral o algoritmo tem dificuldades em convergir. Daí ter havido nos últimos anos grande investigação no desenvolvimento de técnicas generalizadas de gradientes conjugados que procuram resolver o sistema $Ax = b$ sem explorar a redução a (8.46). No entanto, nao iremos discutir esses processos neste trabalho, sugerindo [Axelsson] como uma boa referência nesses assuntos.

É ainda de notar como consequência deste resultado, que o número de condição de uma matriz SPD se pode calcular a partir do número de condição da matriz L da sua decomposição LL^T. Com efeito, como $A = LL^T$, então $\text{cond}_2(A) = [\text{cond}_2(L)]^2$. De igual modo, como $A = LDL^T$, vem $\text{cond}_2(A) = \left[\text{cond}_2(LD^{\frac{1}{2}})\right]^2$. Estes resultados são importantes, pois como referimos anteriormente o estimador LINPACK é mais eficiente para matrizes triangulares.

8.6 A técnica de Precondicionamento

Se S é uma matriz não singular e

$$\hat{A} = SAS^T, \quad \hat{b} = Sb$$

então $\hat{x}$ é solução de $\hat{A}x = \hat{b}$ se e só se $\bar{x} = S^T\hat{x}$ é solução de $Ax = b$. Portanto poderá haver vantagem em resolver o sistema $\hat{A}x = \hat{b}$ em vez de $Ax = b$, desde que

$$\text{cond}(\hat{A}) < \text{cond}(A) \tag{8.47}$$

A técnica de precondicionamento consiste exactamente em considerar uma matriz S tal que a desigualdade (8.47) se verifique e resolver o sistema $\hat{A}x = \hat{b}$ em vez de $Ax = b$. A escolha dessa matriz S não é fácil de fazer na prática. Seguidamente apresentaremos duas hipóteses possíveis para essa escolha.

(i) $S = A^{-1/2}$. Neste caso temos $\hat{A} = I_n$ e portanto $\text{cond}(\hat{A}) = 1$. Do ponto de vista da redução do número de condição, esta é sem dúvida a situação ideal. Contudo, para calcular $\hat{b}$, teremos que resolver um sistema envolvendo a matriz $A^{1/2}$. É evidente que a dificuldade da resolução de um sistema com essa matriz é semelhante à da resolução do sistema com a matriz A.

(ii) $S = D = diag(A)$. Neste caso $\hat{A} = DAD$, bastando portanto escalonar A para obter $\hat{A}$. Este processo poderá dar bons resultados para problemas específicos, mas em geral não é aceitável.

Algoritmo 47

Dado $\hat{x}_0$

$\hat{p}_0 = -\hat{r}_0 = \hat{b} - \hat{A}\hat{x}_0$

Para $k = 0, 1, \ldots$

$$\hat{\alpha}_k = -\frac{\hat{r}_k^T \hat{p}_k}{\hat{p}_k^T \hat{A} \hat{p}_k}$$

$$\hat{x}_{k+1} = \hat{x}_k - \hat{\alpha}_k \hat{p}_k$$

$$\hat{r}_{k+1} = \hat{r}_k - \hat{\alpha}_k \hat{A} \hat{p}_k$$

$$\beta_{k+1} = \frac{\hat{r}_{k+1}^T \hat{r}_{k+1}}{\hat{r}_k^T \hat{r}_k}$$

$$\hat{p}_{k+1} = -\hat{r}_{k+1} + \hat{\beta}_k \hat{p}_k$$

Figura 8.6: Adaptação do método dos gradientes conjugados à resolução do sistema $\hat{A}\hat{x} = \hat{b}$.

Na prática a escolha da matriz S é de certo modo um compromisso entre essas duas hipóteses. Seguidamente iremos explicar como essa tarefa deve ser executada. O método dos gradientes conjugados para o sistema $\hat{A}x = \hat{b}$ tem a forma apresentada na figura 8.6.

É no entanto possível apresentar os passos deste algoritmo usando directamente os vectores b e x_k e a matriz A do sistema original. Com efeito, tem-se:

$$-\hat{r}_0 = \hat{b} - \hat{A}\hat{x}_0 = Sb - SAS^T S^{-T} x_0 = S(b - Ax_0) = -Sr_0$$

e portanto

$$\hat{r}_0 = Sr_0 \tag{8.48}$$

Seja agora

$$\hat{p}_0 = S^{-T} p_0$$

Então

$$\hat{p}_0^T \hat{A} \hat{p}_0 = \left(S^{-T} p_0\right)^T SAS^T \left(S^{-T} p_0\right) = p_0^T A p_0$$

$$\hat{r}_0^T \hat{r}_0 = (S^T r_0)^T (S r_0) = (S^T S r_0)^T r_0$$

Escrevendo

$$\tilde{r}_0 = S^T S r_0 \tag{8.49}$$

tem-se

$$\hat{\alpha}_0 = -\frac{\tilde{r}_0^T r_0}{p_0^T A p_0} \tag{8.50}$$

Donde

$$\begin{aligned} \hat{x}_1 = \hat{x}_0 - \hat{\alpha}_0 \hat{p}_0 &\Rightarrow S^T\hat{x}_1 = S^T\hat{x}_0 - \hat{\alpha}_0 S^T\hat{p}_0 \\ &\Rightarrow x_1 = x_0 - \hat{\alpha}_0 A p_0 \qquad (8.51) \\ \hat{r}_1 = \hat{r}_0 - \hat{\alpha}_0 \hat{A} p_0 &\Rightarrow S^{-1}\hat{r}_1 = S^{-1}\hat{r}_0 - \hat{\alpha}_0 S^{-1} A S^{-T} p_0 \\ &\Rightarrow r_1 = r_0 - \hat{\alpha}_0 A p_0 \qquad (8.52) \end{aligned}$$

Além disso

$$\hat{r}_i^T \hat{r}_i = (Sr_i)^T (Sr_i) = \tilde{r}_i^T r_i, \quad i = 0, 1$$

e portanto

$$\hat{\beta}_0 = \frac{\tilde{r}_1^T r_1}{\tilde{r}_0^T r_0} \qquad (8.53)$$

Por outro lado,

$$\hat{p}_1 = -\hat{r}_1 + \hat{\beta}_1 \hat{p}_0 \Rightarrow S^T\hat{p}_1 = -S^T\hat{r}_1 + \hat{\beta}_0 S^T\hat{p}_0$$

ou seja

$$p_1 = -S^T S r_1 + \hat{\beta}_0 p_0$$

Donde

$$p_1 = -\tilde{r}_1 + \hat{\beta}_0 p_0 \qquad (8.54)$$

Finalmente

$$p_0 = S^T\hat{p}_0 = -S^T\hat{r}_0 = -S^T S r_0 \Rightarrow p_0 = -S^T S r_0 \qquad (8.55)$$

As fórmulas (8.48) a (8.55) são também válidas, por indução, para qualquer $k > 0$. Se considerarmos a matriz $M = \left(S^T S\right)^{-1}$ então as fórmulas (8.48) a (8.55) permitem escrever os passos do algoritmo da figura 8.6 na sua forma equivalente apresentada na figura 8.7 que usa os vectores x_k e b e a matriz A originais.

A matriz $M = (S^T S)^{-1}$ desempenha um papel fundamental em todo este processo e é conhecida como Matriz de Precondicionamento. A matriz S não chega a ser calculada, sendo apenas necessário obter a decomposição LDL^T de M. Note-se que

$$S^T \hat{A} S^{-T} = S^T S A S^T S^{-T} = S^T S A = M^{-1} A$$

e portanto

$$\mathrm{cond}_2(\hat{A}) = \frac{|\lambda_{\max}(\hat{A})|}{|\lambda_{\min}(\hat{A})|} = \frac{|\lambda_{\max}(M^{-1}A)|}{|\lambda_{\min}(M^{-1}A)|} = \mathrm{cond}_2(M^{-1}A)$$

onde $\lambda_{\max}(A)$ e $\lambda_{\min}(A)$ representam os valores próprios de maior e menor valor absoluto. Se fizermos $M = A$, então $\mathrm{cond}(\hat{A}) = \mathrm{cond}(M^{-1}A) = 1$. Portanto M deve ser uma boa aproximação de A, e quanto melhor for essa aproximação, melhor será a convergência.

O Critério de paragem a implementar poderá ser o seguinte:

$$||r_{k+1}||_2 \leq \epsilon$$

com $\epsilon \approx \sqrt{\epsilon_M}$. Dado que M é não singular e $M\tilde{r}_k = r_k$, então $||r_{k+1}||_2$ é pequeno apenas quando $\gamma_k = \tilde{r}_{k+1}^T r_{k+1}$ também o é. Por isso na prática o critério de paragem só é verificado quando γ_k é pequeno. Além disso é conveniente usar também o critério

$$\frac{||x_{k+1} - x_k||_\infty}{\max\{1, ||x_{k+1}||_\infty\}} < \epsilon$$

pelas razões apresentadas anteriormente.

Existem vários processos para a escolha da matriz M [Axelsson]. A técnica mais usada na prática é a denominada Decomposição Incompleta e será discutida na secção seguinte.

Algoritmo 48

Dado x_0

$r_0 = Ax_0 - b$

Resolva $M\tilde{r}_0 = r_0$

$p_0 = -\tilde{r}_0$

Para $k = 0, 1, \ldots$

$v_k = Ap_k$

$\gamma_k = \tilde{r}_k^T r_k$

$\alpha_k = -\dfrac{\gamma_k}{p_k^T v_k}$

$x_{k+1} = x_k - \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k v_k$

Se o critério de paragem for satisfeito, Termine.

Doutro modo:

Resolva $M\tilde{r}_{k+1} = r_{k+1}$

$\beta_k = \dfrac{\tilde{r}_{k+1} r_{k+1}}{\gamma_k}$

$p_{k+1} = -\tilde{r}_{k+1} + \beta_k p_k$

Figura 8.7: Método dos gradientes conjugados com precondicionamento

8.7 Decomposição incompleta

Neste processo procura-se obter a decomposição LDL^T da matriz A de modo que em cada iteração k apenas os elementos a_{ij} de A correspondentes a um conjunto $X^{(k)}$ sejam modificados. Esse conjunto $X^{(k)}$ satisfaz as seguintes condições:

$$\left.\begin{array}{l} X^{(k)} \subseteq \{k,\ldots,n\} \times \{k,\ldots,n\} \\ (i,i) \in X^{(k)} \text{ para todo } i \geq k \\ (i,j) \in X^{(k)} \Rightarrow (j,i) \in X^{(k)} \end{array}\right\} \tag{8.56}$$

onde $\times$ representa produto cartesiano de dois conjuntos. O método pode ser descrito da seguinte forma

Algoritmo 49 (IDPCG)

Para $k = 1, 2, \ldots, n-1$
  Para $i = k+1, \ldots, n$
    Se $(i,k) \in X^{(k)}$ *faça*
      $aux = a_{ik}$
      $a_{ik} = \frac{a_{ik}}{a_{kk}}$
    Para $j = k+1, \ldots, i$
      Se $(i,j) \in X^{(k)}$ *e* $(j,k) \in X^{(k)}$ *faça*
        $a_{ij} = a_{ij} - aux\ a_{jk}$

Se os elementos a_{kk}, $k = 1, \ldots, n$, são todos positivos, então o processo termina com os factores L e D da decomposição LDL^T de uma matriz $M \in \text{SPD}$, onde

$$d_{kk} = a_{kk}, \quad l_{ik} = \begin{cases} a_{ik} & \text{se} \quad i \in X^{(k)} \\ 0 & \text{se} \quad i \notin X^{(k)} \end{cases}$$

Como exemplo de aplicação, consideremos a matriz SPD

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 \\ 1 & 0 & 3 & 2 \\ 1 & 0 & 2 & 4 \end{bmatrix} \tag{8.57}$$

e seja $X^{(k)} = \{(i,j) : a_{ij} \neq 0\}$ para cada iteração $k = 1, \ldots, n-1$. Então os elementos nulos não são transformados durante o processo de decomposição, pelo que, após a primeira iteração se tem

$$A^{(2)} = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ -1 & 0 & 2 & \\ 1 & 0 & 1 & 3 \end{bmatrix}$$

Na segunda iteração não é efectuada qualquer operação ($A^{(3)} = A^{(2)}$). Finalmente na última iteração obtém-se a matriz

$$A^{(4)} = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ -1 & 0 & 2 & \\ 1 & 0 & \frac{1}{2} & \frac{5}{2} \end{bmatrix}$$

e os factores L e D da matriz M são dados por

$$L = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ -1 & 0 & 2 & \\ 1 & 0 & \frac{1}{2} & 1 \end{bmatrix}, D = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 2 & \\ & & & \frac{5}{2} \end{bmatrix}$$

Para que o método possa terminar com sucesso, é necessário que sejam positivos todos os elementos diagonais das matrizes reduzidas $A^{(k)}$ que se vão obtendo durante as suas $(n-1)$ iterações. Isso pode não ser possível para algumas escolhas de conjuntos $X^{(k)}$. Seguidamente é estabelecida uma condição suficiente para a terminação do algoritmo.

Teorema 8.22 *Se A é uma matriz simétrica* H_+*, então o algoritmo termina com $M = LDL^T \in$* SPD*.*

Demonstração: Consideremos primeiramente o caso de A ser estritamente diagonalmente dominante com elementos diagonais positivos ($A \in \mathrm{SDD}_+$). Para provar o teorema, basta mostrar que após a primeira iteração do processo se obtém uma matriz $S \in \mathrm{SDD}_+$. Mas da definição do algoritmo tem-se

$$B = (A|a_{11}) = S + Q$$

em que Q corresponde à parte do Complemento de Schur que não é calculada. Como $(A|a_{11}) \in \mathrm{SDD}_+$ e

$$b_{ii} = a_{ii} - \frac{a_{i1}^2}{a_{11}}$$

então $s_{ii} \geq b_{ii}$ para todo o i. Donde

$$s_{ii} \geq b_{ii} \geq \sum_{j \neq i} |b_{ij}| \geq \sum_{j \neq i} |s_{ij}|$$

Portanto $S \in \mathrm{SDD}_+$ como desejávamos mostrar. O resultado é agora estabelecido por indução.

Seja agora $A \in \mathrm{H}_+$. Então existe uma matriz diagonal D de elementos diagonais positivos tal que $AD \in \mathrm{SRDD}_+$. Seja

$$D = \begin{bmatrix} d_{11} & \\ & \bar{D} \end{bmatrix}$$

Então como vimos na secção 3.7 do capítulo 3, tem-se

$$(A|a_{11})\bar{D} \in \mathrm{SRDD}_+$$

Mas

$$(A|a_{11})\bar{D} = S\bar{D} + Q\bar{D}$$

e procedendo como anteriormente chega-se à conclusão que $S\bar{D} \in \mathrm{SRDD}_+$. Donde $S \in \mathrm{H}_+$ e o teorema fica demonstrado por indução.

É importante notar que este teorema fornece uma condição suficiente que no entanto não é necessária para certas escolhas do conjunto $X^{(k)}$. Com efeito, é fácil de ver que a matriz (8.57) do exemplo anterior não é H_+, pois a sua matriz companheira não é K. Contudo o processo de decomposição incompleta terminou com sucesso nesse caso. Por outro lado, como toda a matriz K é H_+, então o processo IDPCG fornece uma matriz SPD se $A \in$ K. Além disso é possível demonstrar o seguinte resultado.

Teorema 8.23 *Se $A \in$ K então o algoritmo IDPCG termina com uma matriz $M = LDL^T \in$ K. Além disso $N = M - A \geq 0$.*

Demonstração: Tal como no teorema anterior, basta demonstrar que $(A|a_{11}) = Q - R$ com $R \geq 0$ e $Q \in \mathrm{K}$. Se $A \in \mathrm{K}$, então também

$$B = (A|a_{11}) = Q - R \in \mathrm{K}$$

Além disso, usando o mesmo tipo de raciocínio usado no teorema anterior, vem

$$r_{ii} = q_{ii} - b_{ii} \geq 0, \text{ para todo } i.$$

Por outro lado $B \in \mathrm{Z}$ e portanto os elementos não diagonais de Q e de $(-R)$ são não positivos. Portanto $R \geq 0$ e $Q \in \mathrm{Z}$. Como $(A|a_{11}) \in \mathrm{K} = \mathrm{Z} \cap \mathrm{S}$, então existe um vector $v > 0$ tal que

$$(A|a_{11})v = Qv - Rv > 0$$

Donde $Qv > Rv \geq 0$ e $Q \in \mathrm{Z} \cap \mathrm{S}$. Portanto $Q \in \mathrm{K}$ e isso demonstra o teorema.

Como consequência deste teorema e do teorema 8.5 podemos concluir que se $A \in \mathrm{K}$, então

$$\rho(M^{-1}A) < 1$$

Portanto

$$\mathrm{cond}_2(M^{-1}A) < \frac{1}{\lambda_{\min}(M^{-1}A)} \tag{8.58}$$

com $\lambda_{\min}(B)$ o menor valor próprio da matriz $B \in \mathrm{SPD}$. Assim a matriz M pode ser um bom precondicionamento para a resolução do sistema $Ax = b$ com o método PCG. Se $A \in \mathrm{H}_+$ e não pertence à classe K, não é possível obter o mesmo tipo de limite superior para o número de condição de $M^{-1}A$.

Consideremos agora uma matriz A que não pertença à classe H_+. Então o algoritmo IDPCG pode ser ainda aplicado com uma modificação segundo a qual sempre que ocorrer um elemento a_{kk} não positivo se lhe adiciona um número $\mu_k > 0$ tal que $a_{kk} + \mu_k > 0$. A escolha dessa constante μ_k é muito importante na eficiência do método IDPCG. Uma primeira hipótese consiste em escolher μ_k de modo a que

$$a_{kk} + \mu_k > \alpha\lambda$$

com α um número real positivo menor que um e $\lambda = \max_{i>k} |a_{ik}|$. Nesse caso o algoritmo proposto no capítulo 4 para matrizes indefinidas só usa pivots 1×1 e termina com uma matriz $M = LDL^T$ definida como anteriormente a partir da última matriz reduzida. Outros processos de efectuar a decomposição incompleta modificada podem ser baseados em critérios descritos em [Gill et al, cap. 4] e [Schnabel e Eskow], mas não serão discutidos neste trabalho.

Um outro processo de obter uma matriz de precondicionamento é baseado na ideia de compensação na diagonal discutida na secção 8.4. Assim considera-se primeiramente a matriz

$$B = Q + A - R$$

com R uma matriz simétrica constituída pelos elementos positivos de A e Q uma matriz diagonal de elementos diagonais positivos tal que $Qv > Rv$ para certo vector $v > 0$. Então $B \in \mathrm{K}$ e a sua decomposição incompleta pode ser obtida pelo algoritmo IDPCG, obtendo-se assim a matriz de precondicionamento $M = LDL^T$. É de notar que se $A \geq 0$, então B é uma matriz diagonal pelo que não é necessário usar o algoritmo IDPCG.

Da discussão do algoritmo IDPCG facilmente se conclui que a escolha dos conjuntos $X^{(k)}$ tem um papel fundamental na valia do precondicionamento $M = LDL^T$ que se obtém por esse processo. Se em cada iteração k o conjunto $X^{(k)}$ é escolhido tendo apenas em conta a estrutura da parte não nula da matriz A, diz-se que a decomposição incompleta é Por Posição. Por outro lado

Problema	Ordem	Número de elementos não nulos não diagonais	Origem
1	420	3720	Colecção Harwell-Boeing
2	1083	8677	
3	1473	16384	
4	1806	30824	
5	1000	1890	Matrizes de diferenças finitas
6	3000	5395	

Tabela 8.1: Primeiro conjunto de problemas teste para os métodos iterativos

a decomposição incompleta é Por Valores se os valores reais dos elementos das sucessivas matrizes reduzidas forem considerados na construção dos conjuntos $X^{(k)}$. A chamada Decomposição Incompleta Sem Enchimentos pertence à primeira categoria e corresponde a não autorizar quaisquer enchimentos na decomposição LDL^T de A. Portanto tem-se

$$X^{(k)} = \{(i,j) : a_{ij} \neq 0\}$$

em cada iteração k do processo IDPCG. Esta decomposição tem a grande vantagem de se poder implementar sem modificar a estrutura de dados que armazena a matriz original. Além disso é possível desenvolver uma técnica de grau mínimo sem enchimentos que permita obter uma decomposição LDL^T que seja pouco diferente da matriz A. Uma outra alternativa interessante é o uso de um processo de invólucro truncado, onde se procura obter uma decomposição de uma matriz cujo invólucro está contido no conjunto que se obteria se se aplicasse o método do invólucro à matriz A. Sugerimos [Patrício] para uma discussão desse processo.

8.8 Experiência computacional

Nesta secção relatamos a experiência efectuada com alguns métodos introduzidos neste capítulo, nomeadamente o método SOR (com parâmetro $\omega = 1.3$), o método dos gradientes conjugados (CG) e o método dos gradientes conjugados precondicionados com decomposição incompleta sem enchimentos (IDPCG). Começamos em primeiro lugar por investigar a sensibilidade do método CG à tolerância. Para esta primeira experiência recorremos a um conjunto de cinco problemas teste que apresentamos na tabela 8.1, em tudo semelhante ao usado na experiência computacional do capítulo anterior.

Para obter este primeiro conjunto de resultados considerámos tolerâncias iguais a 10^{-5} e 10^{-8} para os métodos referidos, e estabelecemos um número máximo de 50000 iterações para este método. Como computador, usámos uma Iris Indigo com processador R3000 a 25 Mhz e precisão da máquina igual a 10^{-16}. A nossa análise incidiu sobre o número de iterações, o tempo de execução em segundos de CPU e a norma ℓ_∞ do resíduo. Os resultados são apresentados sob a forma de gráfico na figura 8.8. Como seria de esperar, a diminuição do valor da tolerância implica um aumento no número de iterações e no tempo de CPU e uma diminuição no valor do resíduo da solução obtida. Salientamos que para o problema 3 não foi possível obter uma solução com uma tolerância de 10^{-8} em menos de 50000 iterações.

Na tabela 8.2 apresentamos uma comparação no desempenho do método dos gradientes conjugados com o do método directo MA27 apresentado no capítulo anterior, assim como procuramos discutir a eficácia do precondicionamento nos cinco primeiros problemas apresentados na tabela 8.1. Para os métodos iterativos consideramos uma tolerância igual a 10^{-8}. Como primeira conclusão podemos afirmar que o método directo obtém melhores resultados do ponto de vista do tempo de execução e do resíduo. Além disso, o precondicionamento traduz-se apenas numa ligeira melhoria do método de gradientes conjugados.

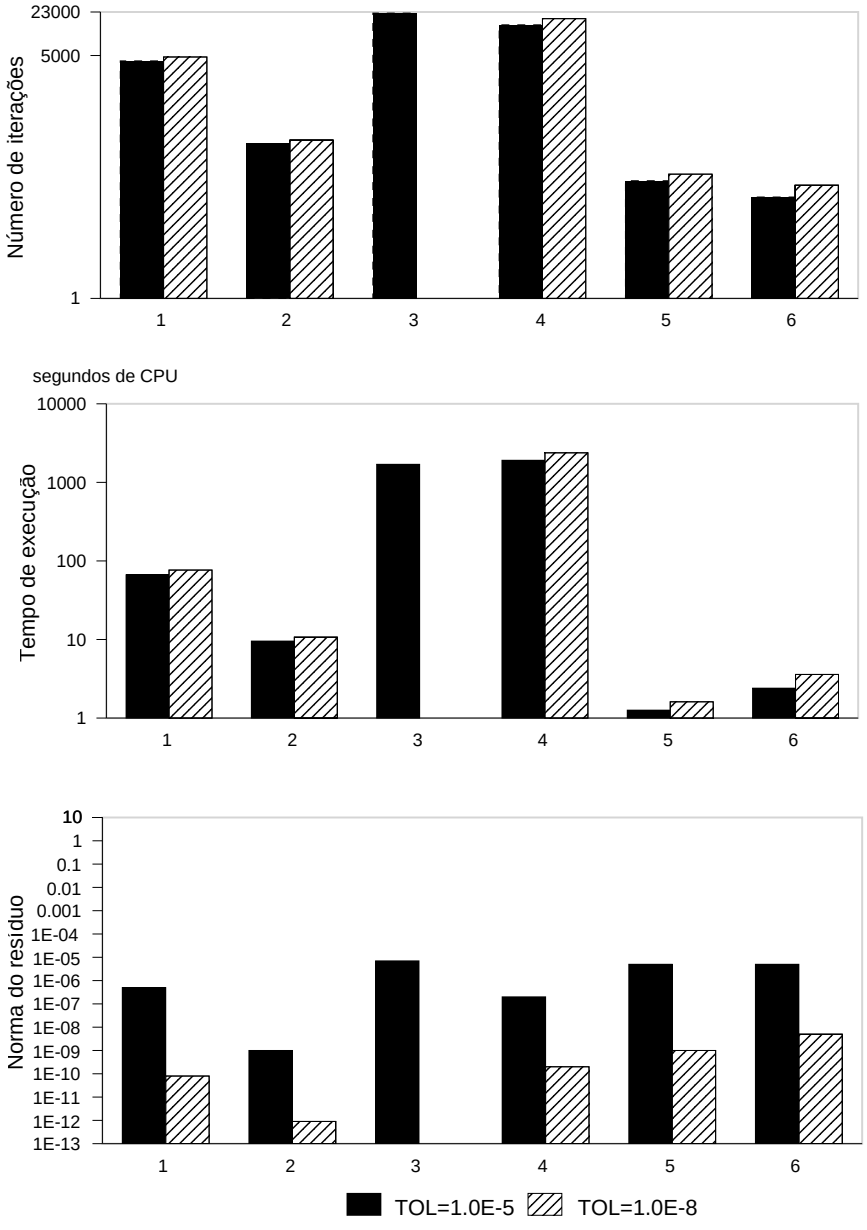


Figura 8.8: O papel da tolerância no desempenho do método dos gradientes conjugados

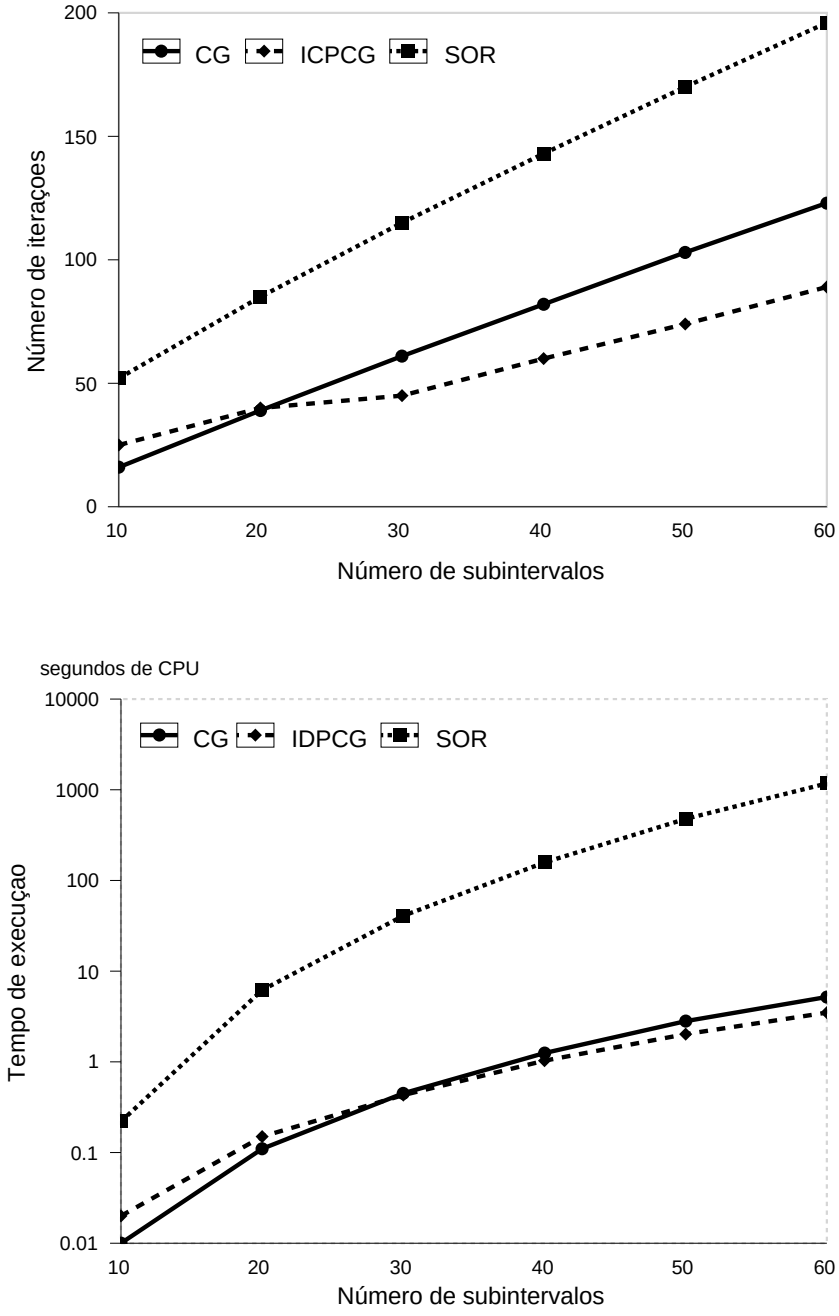


Figura 8.9: Comparação entre os métodos SOR, CG e IDPCG em problemas de derivadas parciais

Problema	Método	NI	T	ERR
1	MA27		0.58	6.7E-15
	CG	371	0.77	4.2E-10
	IDPCG	300	1.43	2.7E-11
2	MA27		0.80	5.6E-14
	CG	733	1.40	1.8E-08
	IDPCG	413	1.64	2.6E-10
3	MA27		0.49	8.8E-17
	CG	317	1.20	2.0E-09
	IDPCG	192	1.23	8.6E-10
4	MA27		0.27	8.8E-17
	CG	1372	4.02	4.4E-10
	IDPCG	1102	4.06	6.7E-10
5	MA27		0.75	7.8E-17
	CG	2098	6.00	8.1E-10
	IDPCG	1031	5.75	2.3E-10

Tabela 8.2: A importância do precondicionamento e a sua comparação com um método directo. (NI: número de iterações; T: tempo de CPU (em segundos); ERR: norma ℓ_∞ do resíduo)

No terceiro e último conjunto de resultados que aqui apresentamos, usámos como problemas teste as matrizes dos sistemas de equações lineares originários da resolução de equações de derivadas parciais com o método das diferenças finitas que discutimos no primeiro capítulo. Nessas experiências considerámos 10, 20, 30, 40, 50 e 60 subintervalos, de maneira a poder testar a importância do acréscimo de dimensão em sistemas com a mesma estrutura. A tolerância foi sempre igual a 10^{-8}. Os resultados, apresentados na figura 8.9, mostram que os métodos baseados em gradientes conjugados (CG e IDPCG) são claramente superiores ao método SOR. Por outro lado, estes resultados permitem-nos concluir que o método CG é superior ao método IDPCG para problemas de dimensão reduzida, invertendo-se as posições para problemas de dimensão mais elevada. O aumento dessa tendência é proporcional à ordem do problema.

Além disso a eficiência do método SOR é muito mais deteriorada com o aumento da dimensão do sistema. Assim, este tipo de sistemas é um bom exemplo de um caso onde o algoritmo de gradientes conjugados com precondicionamento sem enchimentos é particularmente recomendável. É de notar que um método directo não é muito eficiente, quando o número de intervalos é muito elevado, pois ocorrem muitos enchimentos durante a fase de factorização.

Exercícios

1. Desenvolva o algoritmo de implementação do método de Gauss-Seidel para a resolução de um sistema com uma matriz $A \in$ SDD de comprimentos de banda inferior e superior iguais a 2.

2. Desenvolva uma implementação do método SOR para a resolução de um sistema com uma matriz esparsa SPD armazenada num esquema de colecção de vectores com diagonal separada.

3. Desenvolva um algoritmo de implementação do método de Jacobi para a resolução de um sistema com uma matriz SPD de comprimento de banda igual a 5.

4. Considere as seguintes matrizes

$$A_1 = \begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & 2 & 1 \\ 0 & 2 & 5 & -1 \\ 0 & 1 & -1 & 3 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & -2 & -1 & 1 \\ -2 & 4 & 1 & -1 \\ 1 & 1 & 3 & 2 \\ -1 & 0 & 1 & 3 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & -1 & 3 \end{bmatrix}, \quad A_4 = \begin{bmatrix} -1 & 0 & 0 \\ 2 & 1 & -1 \\ 1 & -2 & 3 \end{bmatrix}, \quad A_5 = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & 0 \\ -1 & 0 & 3 \end{bmatrix}$$

(a) Verifique se os métodos de Jacobi e Gauss-Seidel podem ser usados para resolver sistemas com essas matrizes.

(b) Mostre que o método SOR pode ser usado para resolver o sistema

$$A_1 x = b$$

com $b = [1\ 2\ -1\ 3]^T$, e efectue duas iterações desse processo, usando o vector inicial $x^0 = [1\ 1\ -1\ 1]^T$ e o parâmetro $\omega = 1.3$.

(c) Efectue duas iterações dos métodos de Jacobi e Gauss-Seidel para resolver o sistema

$$A_3 x = b$$

com $b = [1\ -1\ 2]^T$, usando o vector inicial $x^0 = [1\ -1\ 2]^T$.

5. Seja D uma matriz de ordem m diagonal de elementos diagonais positivos, B uma matriz esparsa rectangular de ordem $m \times n$ e A a matriz tridiagonal simétrica de ordem n com elementos diagonais positivos a_i, $i = 1, \ldots, n$ e elementos subdiagonais positivos b_i, $i = 1, \ldots, n-1$ tais que $a_1 > b_1$, $a_i > b_i + b_{i+1}$, $i = 1, \ldots, n-1$ e $a_n > b_{n-1}$.

(a) Diga como armazena as matrizes D, A e B.

(b) Mostre que $D + BAB^T \in$ SPD.

(c) Desenvolva uma implementação do método de Jacobi para a resolução de um sistema com a matriz $D + BAB^T$ usando as estruturas de dados referidas em (a).

6. Considere as seguintes matrizes e vectores:

$$A^{(1)} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & 1 \\ -1 & 1 & 6 \end{bmatrix}, \quad A^{(2)} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 3 & -1 \\ -1 & -2 & 4 \end{bmatrix}, \quad A^{(3)} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

$$A^{(4)} = \begin{bmatrix} 1 & -1 & -2 & 0 \\ -1 & 2 & -1 & 1 \\ -2 & -1 & 5 & 0 \\ 0 & 1 & 0 & 3 \end{bmatrix}, \quad A^{(5)} = \begin{bmatrix} 2 & 0 & -1 & -2 \\ -1 & 3 & -2 & -1 \\ 0 & -1 & 4 & 0 \\ -1 & 0 & -2 & 5 \end{bmatrix}$$

$$b_1 = b_2 = b_3 = [1\ -1\ 2]^T, \quad b_4 = b_5 = [1\ -1\ -2\ 0]^T$$

(a) Determine um conjunto $X \supset \{(i,i) : i = 1, \ldots, n\}$ tal que o método da partição

$$Bx^{(k+1)} = Cx^{(k)} + b, \quad B = \left[a_{ij}^{(t)}\right], \ t = 1, \ldots, 5$$

é convergente.

(b) Efectue duas iterações desse processo para cada uma das matrizes $A^{(i)}$, $i = 1, \ldots, 5$ começando pelo vector nulo.

7.

(a) Mostre que se $A \in$ K e $B \in$ Z são duas matrizes da mesma ordem n tais que $A \leq B$, então $B \in$ K.

(b) Seja X um subconjunto de $\{1, \ldots, n\}^2$ que contenha os pontos (i,i) e a partição $A = LU - R$ onde L e U são obtidos a partir da decomposição incompleta em relação ao conjunto X.

i. Mostre que o método da partição

$$LUx^{(k+1)} = Rx^{(k)} + b$$

é convergente para qualquer $x^{(0)}$ inicial.

ii. Mostre que se X_1 e X_2 são dois subconjuntos de $\{1,\ldots,n\}^2$ tais que $X_1 \subset X_2$, então o método da partição anterior associado a X_2 tem uma taxa de convergência mais rápida do que o associado a X_1.

iii. Mostre que o método de Jacobi é o método da partição com taxa de convergência mais lenta de todos os processos definidos em (i).

8. Considere o sistema de equações lineares

$$\begin{bmatrix} B & -I_n \\ -I_n & B \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$

com b e d vectores de dimensão n, I_n a matriz identidade de ordem n e B uma matriz simétrica tridiagonal de ordem n com elementos diagonais $a_i > 0$, $i = 1,\ldots,n$ e elementos subdiagonais $c_i < 0$,$i = 2,\ldots,n$ satisfazendo

$$a_1 > 1 - c_2, \quad a_i > 1 - c_i - c_{i+1}, \quad a_n > 1 - c_n$$

(a) Mostre que o método definido por

$$\begin{aligned} Bx^{(k+1)} &= b + y^{(k)} \\ By^{(k+1)} &= d + x^{(k)} \end{aligned}$$

converge para a solução do sistema, quaisquer que sejam as aproximações iniciais $x^{(0)}$ e $y^{(0)}$.

(b) Desenvolva uma implementação desse método.

9. Descreva uma implementação do método dos gradientes conjugados para a resolução de um sistema com uma matriz esparsa SPD armazenada num esquema de colecção de vectores.

10. Desenvolva uma implementação do método dos gradientes conjugados com decomposição incompleta sem enchimentos para a resolução de um sistema com uma matriz SPD esparsa armazenada num esquema de colecção de vectores.

11. Efectue duas iterações do método dos gradientes conjugados para a resolução dos sistemas $A^{(i)}x = b_i$, $i = 1,\ldots,5$ do problema 6.

12. Considere o sistema de equações lineares $Ax = b$ com A uma matriz não simétrica não singular esparsa armazenada num esquema de colecção de vectores.

(a) Mostre que $A^TA \in$ SPD.

(b) Desenvolva uma implementação do método dos gradientes conjugados para a obtenção da solução de $Ax = b$ a partir da resolução do sistema equivalente $A^TAx = A^Tb$ e usando apenas a estrutura de dados da matriz A.

13. Sejam $B \in \mathbb{R}^{m\times n}$ e $M \in \mathbb{R}^{n\times n}$ com m e n bastante elevados. Diga como pode resolver um sistema com a matriz

$$\begin{bmatrix} M & -B^T \\ B & 0 \end{bmatrix}$$

usando um método iterativo, quando

(a) $M \in$ SPD.

(b) M é não simétrica PD.

14. Considere o sistema de equações lineares associado à resolução numérica da equação de Laplace e apresentado no capítulo 1.

(a) Mostre que a matriz desse sistema pertence à classe K.

(b) Desenvolva as seguintes implementações do método dos gradientes conjugados para a resolução desse sistema:

   i. sem precondicionamento;
   ii. com precondicionamento diagonal;
   iii. com decomposição incompleta com conjunto $X = \{(i, j) : |i - j| \leq 1\}$;
   iv. com decomposição incompleta sem enchimentos.

(c) Faça um estudo computacional comparativo desses quatro processos para vários valores de m e n.

Capítulo 9

Matrizes ortogonais

Neste capítulo começamos por apresentar a definição de matriz ortogonal e as principais propriedades dessas matrizes. Seguidamente discutimos as chamadas matrizes de Householder e Givens, a decomposição QR usando essas matrizes e a sua aplicação na resolução de sistemas de equações lineares.

9.1 Definição e propriedades

Diz-se que uma matriz Q quadrada de ordem n é Ortogonal se é não singular e a sua inversa é igual à sua transposta, isto é, $Q^T = Q^{-1}$. Portanto tem-se

$$QQ^T = Q^TQ = I_n$$

com I_n a matriz identidade de ordem n.

As matrizes ortogonais possuem algumas propriedades interessantes. Nesta secção iremos discutir as mais relevantes para a resolução de sistemas de equações lineares, deixando outros resultados importantes como exercícios deste capítulo. Assim, da definição de matriz ortogonal podemos estabelecer facilmente os seguintes resultados:

Teorema 9.1 *Se Q é uma matriz ortogonal, então*

1. *$det(Q) = \pm 1$.*
2. *Q^T é ortogonal.*

Teorema 9.2 *Se Q_1 e Q_2 são matrizes ortogonais, então Q_1Q_2 é ortogonal.*

As matrizes ortogonais preservam as normas ℓ_2 de vectores e de matrizes, isto é, verifica-se o seguinte teorema:

Teorema 9.3 *Se Q é uma matriz ortogonal, então*

1. *$||Qx||_2 = ||x||_2$ para todo o vector x.*
2. *$||QA||_2 = ||A||_2$ para toda a matriz A.*
3. *$||Q||_2 = 1$.*

Demonstração:

1. Se x é um vector qualquer, então

$$||Qx||_2^2 = (Qx)^T (Qx) = x^TQ^TQx = x^Tx = ||x||_2^2$$

2. Se $\rho(B)$ representa o raio espectral da matriz B, então

$$\begin{aligned} ||QA||_2 &= \sqrt{\rho\left[(QA)(QA)^T\right]} \\ &= \sqrt{\rho\left(QAA^TQ^T\right)} \\ &= \sqrt{\rho(AA^T)} = ||A||_2 \end{aligned}$$

3. É consequência imediata de 2. e de $||I||_2 = 1$.

Como referimos anteriormente, o número de condição de uma matriz A é um factor fundamental para a precisão do sistema $Ax = b$. Seguidamente mostramos que as matrizes ortogonais preservam o número de condição de uma matriz na norma ℓ_2, isto é, que se verifica o seguinte resultado:

Teorema 9.4 *Se Q é uma matriz ortogonal e $cond_2(B)$ é o número de condição da matriz B na norma ℓ_2, então*

$$\mathrm{cond}_2(QA) = cond_2(A)$$

Demonstração: Se A é uma matriz qualquer, então

$$\begin{aligned} cond_2(QA) &= ||(QA)^{-1}||_2||QA||_2 \\ &= ||A^{-1}Q^T||_2||QA||_2 \\ &= ||A^{-1}||_2||A||_2 = cond_2(A) \end{aligned}$$

É perfeitamente conhecido que toda a matriz A não singular admite uma decomposição QR da forma

$$QA = R$$

com Q uma matriz ortogonal e R triangular superior com elementos diagonais não nulos. Nas próximas seções iremos discutir duas maneiras de obter essa decomposição e a aplicação dessa decomposição à resolução de sistemas de equações lineares.

9.2 Matrizes de Householder

Seja $x \in \mathbb{R}^n$ e consideremos a matriz

$$P = I - \frac{2}{v^Tv}vv^T \tag{9.1}$$

onde I é a matriz identidade de ordem n e

$$v = x \pm ||x||_2e^1 \tag{9.2}$$

com $e^1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^n$. Então

$$Px = \pm||x||_2e^1 \tag{9.3}$$

e portanto todas as componentes do vector x à excepção da primeira são anuladas por premultiplicação da matriz P.

Das fórmulas (9.1), (9.2) e (9.3) conclui-se que é indiferente escolher o sinal + ou − no vector v para anular todas as componentes de Px. Por outro lado se $sign(x_1)$ representar o sinal da componente x_1 do vector x e se x difere pouco de e^1 então

$$v = x - sgn(x_1)||x||_2e^1$$

tem norma bastante reduzida. Isso implica que o produto escalar v^Tv possa ser uma quantidade muito pequena, o que acarreta problemas de instabilidade no cálculo da matriz P. Por essa razão o vector v é habitualmente definido a partir de

$$v = x + sgn(x_1)||x||_2e^1$$

Além disso como todos os elementos da matriz vv^T são produtos de duas componentes do vector x, então valores grandes dessas componentes podem implicar a ocorrência de fenómenos de *overflow*. Esse problema é resolvido considerando o vector $\frac{1}{||x||_\infty}x$ em vez de x na definição de v. Assim na definição da matriz P, o vector v é definido por

$$v = \frac{x}{||x||_\infty} + sgn(x_1)\left\|\frac{x}{||x||_\infty}\right\| e^1 \tag{9.4}$$

e tem-se

$$Px = sgn(x_1)\left\|\frac{x}{||x||_\infty}\right\|_2 e^1 = \begin{bmatrix} -sgn(x_1)\alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{9.5}$$

com

$$\alpha = \left\|\frac{x}{||x||_\infty}\right\|_2 = \sqrt{\sum_{i=1}^{n}\left(\frac{x_i}{vmax}\right)^2}$$

e

$$vmax = \max_i |x_i|$$

Seja ainda

$$\beta = \frac{2}{v^T v} \tag{9.6}$$

Se v e β são dados por (9.4) e (9.6) respectivamente, então

$$P = I - \beta vv^T$$

é chamada Matriz (ou Transformação) de Householder. Esta matriz é uma matriz ortogonal, pois

$$\begin{aligned} P^T P &= (I - \beta vv^T)^T(I - \beta vv^T) \\ &= (I - \beta vv^T)(I - \beta vv^T) \\ &= I - 2\beta vv^T + \beta^2 vv^T vv^T \\ &= I - 2\beta vv^T + \beta^2 v^T v(vv^T) \\ &= I - \frac{4}{v^T v}vv^T + \frac{4}{(v^T v)^2}v^T v(vv^T) \\ &= I - \frac{4}{v^T v}vv^T + \frac{4}{v^T v}vv^T = I \end{aligned}$$

A discussão apresentada mostra que as matrizes de Householder podem ser usadas para transformar uma matriz A numa matriz triangular. Com efeito se $A \in \mathbb{R}^{n\times n}$ e

$$P_1 = I - \beta_1 v^{(1)} {v^{(1)}}^T$$

com

$$v^{(1)} = \left[\frac{a_{11}}{vmax} + sgn(a_{11})\alpha, \frac{a_{21}}{vmax}, \ldots, \frac{a_{n1}}{vmax}\right]^T$$

$$vmax = \max\{|a_{i1}| : 1 \le i \le n\}$$

$$\alpha = \sqrt{\sum_{i=1}^{n}\left(\frac{a_{i1}}{vmax}\right)^2}, \beta_1 = \frac{2}{{v^{(1)}}^T v^{(1)}}$$

então

$$P_1 A = A^{(2)} = \begin{bmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \ldots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \ldots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \ldots & a_{nn}^{(2)} \end{bmatrix}$$

É agora fácil de concluir que, se A é não singular, então

$$P_{n-1}P_{n-2}\dots P_2P_1A = R$$

onde R é uma matriz triangular superior e cada P_k uma matriz de Householder da forma

$$P_k = I - \beta_k v^{(k)} {v^{(k)}}^T \tag{9.7}$$

com

$$v^{(k)} = \left[\begin{array}{ccccccc} 0 & \dots & 0 & \frac{a_{kk}^{(k)}}{vmax} + sgn(a_{kk}^{(k)})\alpha & \frac{a_{k+1,k}^{(k)}}{vmax} & \dots & \frac{a_{nk}^{(k)}}{vmax} \end{array}\right]^T,$$

$$vmax = \max\{\left|a_{ik}^{(k)}\right|, i = k, \dots, n\},$$

$$\alpha = \sqrt{\sum_{i=k}^{n}\left(\frac{a_{ik}^{(k)}}{vmax}\right)^2},$$

$$\beta_k = \frac{2}{{v^{(k)}}^T v^{(k)}}$$

Como todas as matrizes P_k são ortogonais, também

$$Q = P_{n-1}P_{n-2}\dots P_2P_1 \tag{9.8}$$

é ortogonal e podemos escrever

$$QA = R \tag{9.9}$$

que se chama Decomposição QR de A.

A título de ilustração, consideremos a seguinte matriz

$$A = \left[\begin{array}{ccc} 4 & 1 & -1 \\ -1 & 9 & 1 \\ 1 & -1 & 7 \end{array}\right]$$

e procuremos obter a sua decomposição QR. No primeiro passo da decomposição QR os elementos não diagonais da primeira coluna têm de ser anulados. Para isso constroi-se a matriz

$$P_1 = I - \beta_1 v^{(1)} {v^{(1)}}^T$$

com

$$v_1 = \left[\begin{array}{ccc} \frac{a_{11}}{vmax} + sgn(a_{11})\alpha & \frac{a_{21}}{vmax} & \frac{a_{31}}{vmax} \end{array}\right]^T,$$

$$vmax = \max\{|a_{i1}|, i = 1, 2, 3\},$$

$$\alpha = \sqrt{\sum_{i=1}^{3}\left(\frac{a_{i1}^{(1)}}{vmax}\right)^2},$$

$$\beta_1 = \frac{2}{{v^{(1)}}^T v^{(1)}}$$

Então $vmax = 4, \alpha = 1.0607$ e por isso

$$v^{(1)} = \left[\begin{array}{ccc} 2.0607 & -0.2500 & 0.2500 \end{array}\right]$$

Logo $\beta_1 = 0.4575$ e

$$P_1 = \left[\begin{array}{ccc} -0.9428 & 0.2357 & -0.2357 \\ 0.2357 & 0.9714 & 0.0286 \\ -0.2357 & 0.0286 & 0.9714 \end{array}\right]$$

Se agora multiplicarmos a matriz P_1 por A obtemos

$$A^{(2)} = P_1 A = \begin{bmatrix} -4.2426 & 1.4142 & -0.4714 \\ 0 & 8.9498 & 0.9359 \\ 0 & -0.9498 & 7.0641 \end{bmatrix}$$

o que está de acordo com o nosso objectivo. Na segunda iteração há que calcular a matriz P_2 que faz com que todos os elementos da segunda coluna de $P_2 A^{(2)}$ abaixo da diagonal sejam nulos. Como referimos anteriormente essa matriz tem a forma $P_2 = I - \beta_2 v^{(2)} {v^{(2)}}^T$, com

$$v^{(2)} = \frac{1}{vmax} \begin{bmatrix} 0, \frac{a_{22}^{(2)}}{vmax} + sgn(a_{22}^{(2)})\alpha, a_{32}^{(2)} \end{bmatrix}^T,$$

$$vmax = \max\{|a_{22}^{(2)}|, |a_{32}^{(2)}|\},$$

$$\alpha = \sqrt{\left(a_{22}^{(2)}\right)^2 + \left(a_{32}^{(2)}\right)^2},$$

$$\beta_2 = \frac{2}{{v^{(2)}}^T v^{(2)}}.$$

Então

$$v^{(2)} = \begin{bmatrix} 0 & 2.0056 & -0.1061 \end{bmatrix}^T,$$

$$\beta_2 = 0.4958$$

$$P_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.9944 & 0.1055 \\ 0 & 0.1055 & 0.9944 \end{bmatrix}$$

Assim tem-se

$$R = P_2 A^{(2)} = \begin{bmatrix} -4.2426 & 1.4142 & -0.4714 \\ 0 & -9.0000 & -0.1853 \\ 0 & 0 & 7.1234 \end{bmatrix}$$

e a matriz triangular superior da decomposição QR foi obtida. É de notar que a matriz Q é o produto de P_2 por P_1 e assim a sua forma explícita é

$$Q = P_2 P_1 = \begin{bmatrix} -0.9428 & 0.2357 & -0.2357 \\ -0.2593 & -0.9630 & 0.0741 \\ -0.2095 & 0.1309 & 0.9690 \end{bmatrix}.$$

Consideremos agora um sistema de equações lineares

$$Ax = b. \tag{9.10}$$

Se a decomposição QR de A foi calculada, então a resolução do sistema (9.10) consiste dos seguintes passos:

1. Determinar o vector $\bar{b} = Qb$;
2. Resolver o sistema $Rx = \bar{b}$.

Da descrição do processo de decomposição QR conclui-se que as matrizes P_k apenas são necessárias para premultiplicar A e as suas matrizes reduzidas $A^{(k)}$. Além disso, dada uma coluna $A_{.j}^{(k)}$ de $A^{(k)}$, então

$$\begin{aligned} P_k A_{.j}^{(k)} &= \left(I - \beta_k v^{(k)} {v^{(k)}}^T\right) A_{.j}^{(k)} \\ &= A_{.j}^{(k)} - \left(\beta_k {v^{(k)}}^T A_{.j}^{(k)}\right) v^{(k)} \end{aligned}$$

Assim as matrizes P_k não precisam de ser armazenadas nem calculadas explicitamente no processo de decomposição QR da matriz A. Em vez disso, apenas o escalar β_k e o vector $v^{(k)}$ são considerados e o produto $P_k A^{(k)}$ é feito de acordo com o apresentado acima. Os algoritmos para a construção dessas quantidades β_k e dos vectores $v^{(k)}$ e para o cálculo de $P_k A^{(k)}_{.j}$, $j \geq k$, são descritos a seguir.

Algoritmo 50 (Construção de P_k (isto é, de β_k e $v^{(k)}$)) *Seja $A = A^{(k)}$ a matriz obtida após $k-1$ iterações.*

$vmax = \max\{|a_{ik}| : k \leq i \leq n\}$

$\alpha = 0$

Para $i = k, \ldots, n$

$v_i = \frac{a_{ik}}{vmax}$

$\alpha = \alpha + v_i^2$

$\alpha = \sqrt{\alpha}$

$\beta = \frac{1}{\alpha(\alpha + |v_k|)}$

$v_k = v_k + sgn(v_k)\alpha$

Algoritmo 51 (Cálculo de $P_k A$)

Para $t = k, \ldots, n$

$s = \sum_{i=k}^{n} v_i a_{it}$

$s = \beta s$

Para $i = k, \ldots, n$

$a_{it} = a_{it} - s v_i$

Seguidamente é determinado o número de operações necessárias para obter a matriz P_k e calcular o produto $P_k A$. Do algoritmo 50 tem-se

$$\begin{cases} 1 \text{ raíz quadrada} \\ 2(n-k+2) \text{ multiplicações / divisões} \\ (n-k+3) \text{ adições} \end{cases}$$

e do algoritmo 51 tem-se

$$\begin{cases} (n-k+1)\,[2(n-k+1)+1] \text{ multiplicações /divisões} \\ (n-k+1)\,[(n-k)+(n-k+1)] \text{ adições} \end{cases}$$

Portanto o número total de multiplicações / divisões é dado por

$$\begin{aligned} \sum_{k=1}^{n-1} [2(n-k+1)(1+n-k+1)+(n-k+1)+2] &= \\ 2\sum_{k=1}^{n-1}(n-k)(n-k+1) + 5\sum_{k=1}^{n-1}(n-k+1) + 2\sum_{k=1}^{n-1} 1 &= \\ \frac{2}{3}n(n^2-1) + \frac{(7n+10)(n-1)}{2} & \end{aligned} \tag{9.11}$$

e o número total de adições é

$$\sum_{k=1}^{n-1}(n-k+1)\left[1+2(n-k)+1\right]+2=$$

$$2\sum_{k=1}^{n-1}(n-k)(n-k+1)+2\sum_{k=1}^{n-1}(n-k+2)+2\sum_{k=1}^{n-1}1=$$

$$\frac{2}{3}n(n^2-1)+(2n+4)(n-1) \tag{9.12}$$

Além disso o número total de raízes é $n-1$.

Estes números indicam que a decomposição QR necessita de aproximadamente $\frac{2}{3}n^3$ operações para ser calculada, isto é, requer sensivelmente o dobro das operações necessárias para efectuar a decomposição LU de A.

Se pretendermos resolver o sistema (9.10), então, além da decomposição QR, temos de determinar o vector Qb, o que pode ser feito usando o seguinte algoritmo:

Algoritmo 52 (Cálculo de Qb)

$s=\sum_{i=k}^{n} v_i b_i$

$s=\beta s$

Para $i=k,\ldots,n$

$\quad b_i=b_i-sv_i$

Portanto o número de adições para calcular Qb é dado por

$$\sum_{k=1}^{n-1}\left[2(n-k)+1\right]=2\sum_{k=1}^{n-1}(n-k)+(n-1)=n^2-1 \tag{9.13}$$

e o número de multiplicações / divisões é

$$\sum_{k=1}^{n-1}\left[2(n-k+1)+1\right]=2\sum_{k=1}^{n-1}(n-k+1)+(n-1)=(n-3)(n-1) \tag{9.14}$$

Além disso é ainda necessário resolver o sistema triangular superior $Rx=b$, pelo que o número total de operações para resolver o sistema $Ax=b$ se calcula a partir das fórmulas (9.11), (9.12), (9.13) e (9.14) e vem dado por

$$\begin{cases} \text{número de raízes quadradas:} & = & n-1 \\ \text{número de multiplicações / divisões:} & = & \frac{2}{3}n^3+(n-1)(5n+9)+\frac{n}{3} \\ \text{número de adições:} & = & \frac{2}{3}n^3+\frac{5(n-1)(n+2)}{2}-\frac{2n}{3} \end{cases} \tag{9.15}$$

Portanto a resolução de um sistema usando a decomposição QR requer aproximadamente o dobro das operações necessárias para a resolução do sistema com decomposição LU. Por essa razão as matrizes de Householder não são normalmente usadas na resolução de sistemas de equações lineares. Contudo o uso da decomposição QR não acarreta problemas de estabilidade. Com efeito, é possível provar que se E é a matriz erro definida por

$$Q^TR=A+E$$

então

$$||E||_2 \leq c\epsilon_M ||A||_2 \tag{9.16}$$

com c aproximadamente igual a n^2 e ϵ_M a precisão da máquina [Golub].

A armazenagem da decomposição QR da matriz A é feita usando o espaço reservado à matriz A e dois vectores DIAG e BETA de dimensão n e $n-1$ respectivamente de tal modo que:

1. A matriz A armazena os vectores $v^{(k)}$, $k = 1, \ldots, n-1$ das matrizes P_k e os elementos não diagonais da matriz R.

2. O vector DIAG armazena os elementos diagonais de R.

3. O vector BETA armazena os $n-1$ valores β_k associados às matrizes P_k.

Assim, se $P_{n-1} \ldots P_1 A = R$, $P_k = I - \beta_k v^{(k)} {v^{(k)}}^T$, então tem-se

$$\begin{array}{lcl} \text{DIAG} & = & \left[\begin{array}{cccc} r_{11} & r_{22} & \ldots & r_{nn} \end{array} \right] \\ \text{BETA} & = & \left[\begin{array}{cccc} \beta_1 & \beta_2 & \ldots & \beta_{n-1} \end{array} \right] \\ A & = & \left[\begin{array}{ccccc} v_1^{(1)} & r_{12} & \ldots & r_{1,n-1} & r_{1n} \\ v_2^{(1)} & v_2^{(2)} & \ldots & r_{2,n-1} & r_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ v_{n-1}^{(1)} & v_{n-1}^{(2)} & \ldots & v_{n-1}^{(n-1)} & r_{n-1,n} \\ v_n^{(1)} & v_n^{(2)} & \ldots & v_n^{(n-1)} & 0 \end{array} \right] \end{array} \tag{9.17}$$

onde o último elemento diagonal de A não é utilizado.

Chegamos assim à conclusão de que a decomposição QR de uma matriz A pode ser efectuada em $n-1$ iterações substituindo os elementos dessa matriz e usando dois vectores adicionais. O processo de implementação da decomposição QR de uma matriz A é feito usando este tipo de estrutura de dados e os algoritmos 50 a 52 descritos anteriormente e é deixado como exercício. Como veremos no capítulo seguinte, este processo de decomposição QR pode ser facilmente estendido para uma matriz rectangular. Aliás, as matrizes de Householder encontram grande aplicação na resolução de sistemas de equações lineares com matrizes rectangulares densas. A determinação dos valores próprios de uma matriz quadrada densa é outra área em que as matrizes de Householder têm vindo a ser muito usadas [Golub].

9.3 Matrizes de Givens

Uma Matriz de Givens tem a forma

$$J(k,i,\theta) = \left[\begin{array}{ccccccc} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \ldots & s & & \\ & & \vdots & \ddots & \vdots & & \\ & & -s & \ldots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{array} \right] \tag{9.18}$$

com $c = \cos\theta$ e $s = \sin\theta$ para um certo θ. A fórmula fundamental da trigonometria mostra que a matriz $J(k,i,\theta)$ é ortogonal. Além disso se $x \in \mathbb{R}^n$ e $y = J(k,i,\theta)x$, então

$$\left\{ \begin{array}{lcl} y_k & = & cx_k + sx_i \\ y_i & = & -sx_k + cx_i \\ y_j & = & x_j, j \neq i, k \end{array} \right.$$

Se pretendermos que a componente y_i de y seja nula, então basta escolher c e s a partir de

$$c = \frac{x_k}{\sqrt{x_i^2 + x_k^2}}, \quad s = \frac{x_i}{\sqrt{x_i^2 + x_k^2}} \tag{9.19}$$

Portanto se s e c forem escolhidos de modo a satisfazer (9.19), a multiplicação de $J(k, i, \theta)$ por x faz anular a componente i do vector obtido.

Consideremos agora um sistema $Ax = b$ com A uma matriz não singular de ordem n. Da definição anterior é fácil de obter uma matriz triangular R multiplicando A por

$$(n-1) + (n-2) + \ldots + 1 = \frac{n(n-1)}{2}$$

matrizes de Givens construídas de forma a anular sucessivamente os elementos abaixo da diagonal principal. Portanto podemos escrever

$$J\left(n-1, n, \theta_{\frac{n(n-1)}{2}}\right) \ldots J(1,3,\theta_2) J(1,2,\theta_1) A = R \tag{9.20}$$

que fornece a decomposição QR de A usando matrizes de Givens. Portanto em $\frac{n(n-1)}{2}$ passos é possível transformar a matriz A numa matriz triangular superior. Cada passo consiste em calcular uma matriz $J(k, i, \theta)$ e multiplicá-la por uma matriz A obtida da matriz A inicial por um número finito de passos anteriores. Seguidamente, apresentamos um algoritmo que executa esse passo.

Algoritmo 53 (Construção da matriz $J(k, i, \theta)$)

Se $|a_{ik}| \geq |a_{kk}|$ *faça*

$t = \frac{a_{kk}}{a_{ik}},\ s = \frac{1}{\sqrt{1+t^2}},\ c = st$

Se $|a_{ik}| < |a_{kk}|$ *faça*

$t = \frac{a_{ik}}{a_{kk}},\ c = \frac{1}{\sqrt{1+t^2}},\ s = ct$

As comparações efectuadas no algoritmo têm como fim o controle da grandeza dos números. De notar ainda que o algoritmo necessita de uma raíz quadrada, uma adição e quatro multiplicações / divisões. O algoritmo para efectuar o produto da matriz de Givens pela matriz A pode ser escrito da seguinte forma:

Algoritmo 54 (Multiplicação de $J(k, i, \theta)$ por A)

Para $j = k, \ldots, n$

$w = a_{kj}$

$v = a_{ij}$

$a_{kj} = cv + sw$

$a_{ij} = -sv + cw$

O número de adições é $2(n-k+1)$ e o número de multiplicações é $4(n-k+1)$. Assim, dos algoritmos apresentados chega-se à conclusão que o número total de operações para transformar a matriz A numa matriz triangular superior usando matrizes de Givens é dado por:

$$\begin{aligned}
\text{número de raízes quadradas} &= \frac{n(n-1)}{2} \\
\text{número de multiplicações / divisões} &= \sum_{k=1}^{n} (n-k)\left[4(n-k+1)+4\right] \\
&= \frac{4}{3}n(n^2-1) + \frac{n(n-1)}{3} \\
\text{número de adições} &= \sum_{k=1}^{n} (n-k)\left[2(n-k+1)+1\right] \\
&= \frac{2}{3}n(n^2-1) + \frac{n(n-1)}{2}
\end{aligned} \tag{9.21}$$

Portanto o processo de determinação da decomposição QR usando matrizes de Givens necessita de aproximadamente $\frac{4}{3}n^3$ operações, ou seja, do dobro das operações requeridas pelo processo de obtenção da decomposição QR usando matrizes de Householder. Por outro lado, contrariamente ao caso anterior, não é possível desenvolver o processo de decomposição QR usando a estrutura de dados que armazena a matriz A. Por essas razões, as matrizes de Givens não são normalmente usadas na obtenção da decomposição QR de uma matriz densa. No entanto essas matrizes têm vindo a ser utilizadas na decomposição QR de matrizes esparsas e principalmente em processos de actualização da decomposição QR incorporados na implementação de alguns algoritmos de optimização não linear [Dennis], [Gill].

Exercícios

1. Mostre que se Q_1 e Q_2 são duas matrizes ortogonais, então Q_1Q_2 é ortogonal.
2. Mostre que todos os valores próprios de uma matriz ortogonal têm módulo igual a 1.
3. Mostre que se Q é ortogonal, então A e Q^TAQ têm os mesmos valores próprios.
4. Seja A uma matriz simétrica de ordem n com valores próprios λ_i distintos, $i = 1, \ldots, n$.
   (a) Mostre que se $u^1, \ldots, u^n$ são vectores próprios associados a $\lambda_1, \ldots, \lambda_n$, então a matriz
   $$Q = \left[\frac{u^1}{||u^1||}, \ldots, \frac{u^n}{||u^n||} \right]$$
   é ortogonal.
   (b) Mostre que
   $$Q^TAQ = diag(\lambda_1, \ldots, \lambda_n).$$
5. Se S é uma matriz hemi-simétrica, mostre que $A = (I - S)(I + S)^{-1}$ é ortogonal.
6. Determine a decomposição QR das seguintes matrizes
   $$A_1 = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 2 & 0 & 1 \end{bmatrix}, A_2 = \begin{bmatrix} 4 & 1 & -1 & 1 \\ 1 & 3 & 1 & 1 \\ 0 & 1 & 2 & -1 \\ 0 & 0 & 1 & 2 \end{bmatrix}, A_3 = \begin{bmatrix} 2 & 0 & 1 & 1 \\ 0 & 3 & -1 & 0 \\ 1 & 0 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$
   $$A_4 = \begin{bmatrix} 1 & 1 & 2 \\ 2 & -1 & 0 \\ -1 & 3 & 1 \end{bmatrix}, A_5 = \begin{bmatrix} 2 & -1 & 0 \\ 1 & 1 & 4 \\ -1 & 3 & 1 \end{bmatrix}$$
   usando matrizes de Householder.
7. Determine a decomposição QR das matrizes A_i, $i = 1, 2, 3$ do exercício 6 usando matrizes de Givens.
8. Resolva os sistemas $A_ix_i = b^i$, $i = 1, 2$, com $b^1 = [3, 0, 3]^T$ e $b^2 = [3, 4, 4, -1]^T$, usando a decomposição QR dessas matrizes.
9. Mostre que toda a matriz de Hessenberg superior A ($a_{ij} = 0$ para $i > j + 1$) não singular admite uma decomposição QR, com Q produto de $n - 1$ matrizes de Givens e determine o número de operações necessárias para calcular essa decomposição.
10. Seja $A = \bar{Q}\bar{R}$, com $\bar{Q}$ uma matriz ortogonal e $\bar{R}$ uma matriz triangular superior não singular. Desenvolva um processo para a obtenção da decomposição QR da matriz $A + uv^T$ usando matrizes de Givens.

11. Desenvolva uma implementação do processo de decomposição QR com matrizes de Householder usando a estrutura de dados referida neste capítulo.

12.

(a) Mostre que se A é uma matriz simétrica não singular, existem $n-1$ matrizes de Householder $H_1, \ldots, H_{n-1}$ tal que $Q^T AQ = T$, com T uma matriz tridiagonal e $Q = H_1 \ldots H_{n-1}$.

(b) Explique como pode utilizar esta decomposição para resolver um sistema com a matriz A.

(c) Resolva o sistema

$$\begin{cases} x_1 & +2x_2 & -x_3 & = & 2 \\ 2x_1 & -x_2 & +x_3 & = & 2 \\ x_1 & & -3x_3 & = & -2 \end{cases}$$

usando a decomposição anterior.

Capítulo 10

Resolução de sistemas de equações lineares com matrizes rectangulares

Neste capítulo iremos estudar os algoritmos directos e iterativos para a resolução do sistema de equações lineares $Ax = b$ quando A é uma matriz rectangular. Tal como no caso dos sistemas de equações lineares com matrizes quadradas consideraremos apenas o caso em que A tem característica $c(A)$ completa, isto é, em que $c(A) = \min\{m, n\}$. Iremos designar esses sistemas por verticais ou horizontais, consoante o número de equações seja maior ou menor que o número de incógnitas. No primeiro caso o sistema não tem em geral uma solução que satisfaça todas as equações, pelo que nos preocupamos em obter a chamada solução dos mínimos quadrados. Um sistema horizontal com característica completa é indeterminado. Neste capítulo debruçar-nos-emos sobre a determinação da solução desse sistema cuja norma ℓ_2 é mínima.

10.1 Métodos directos para sistemas verticais

Consideremos o sistema

$$Ax = b \tag{10.1}$$

em que $A \in \mathbb{R}^{m\times n}$ com $m > n$, $b \in \mathbb{R}^m$ e $x \in \mathbb{R}^n$. Como referimos anteriormente, iremos apenas considerar o caso em que a característica de A é igual a m. Contrariamente ao que sucede quando A é quadrada, não existe em geral um solução desse sistema com resíduo $r = b - Ax$ nulo. É no entanto possível determinar a solução que minimiza a norma ℓ_2 desse resíduo, isto é, a solução do chamado Problema de Mínimos Quadrados Lineares

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2 \tag{10.2}$$

Se considerarmos o seguinte problema equivalente a (10.2)

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\|Ax - b\|_2^2$$

então tem-se

$$\begin{aligned} f(x) &= \frac{1}{2}(Ax - b)^T(Ax - b) \\ &= b^Tb - (A^Tb)^Tx + \frac{1}{2}x^T(A^TA)x \end{aligned}$$

Como $c(A) = n$ então $A^T A \in$ SPD. Portanto $\bar{x}$ é solução única do problema (10.2) se e só se o gradiente $\nabla f(x)$ de f em x satisfaz a equação $\nabla f(\bar{x}) = 0$ [Dennis]. Mas

$$\nabla f(x) = A^T Ax - A^T b$$

e portanto $\bar{x}$ satisfaz as equações normais

$$A^T Ax = A^T b \tag{10.3}$$

É evidente que a solução de (10.1) também é solução deste sistema (10.3), o que confirma a asserção anterior.

Nesta secção iremos apresentar processos para a resolução do sistema (10.1), ou equivalentemente para a solução do problema de mínimos quadrados lineares (10.2). Tal como no caso da resolução de sistemas de equações lineares com matrizes quadradas, o número de condição de A tem uma importância fundamental em relação à precisão da solução que se obtém por qualquer um destes processos. Para definir o número de condição de uma matriz A rectangular, introduzimos a chamada Pseudoinversa A^+ de A a partir de

$$A^+ = \left(A^T A\right)^{-1} A^T \tag{10.4}$$

É de notar que esta matriz tem ordem $n \times m$. Além disso $A^T A \in$ SPD pois $c(A) = n$. Portanto A^+ existe sempre e

$$A^+ A = \left(A^T A\right)^{-1} A^T A = I_n$$

Assim podemos definir o número de condição $cond(A)$ a partir de

$$cond(A) = ||A|| \cdot ||A^+|| \tag{10.5}$$

Se usarmos a norma ℓ_2 então tem-se

$$cond_2(A) = ||A||_2 \cdot ||A^+||_2 = \sqrt{\rho\left(A^T A\right)}\sqrt{\rho\left(A^+ \left(A^+\right)^T\right)}$$

Mas

$$\begin{aligned} A^+ \left(A^+\right)^T &= \left(A^T A\right)^{-1} A^T \left[\left(A^T A\right)^{-1} A^T\right]^T \\ &= \left(A^T A\right)^{-1} \left(A^T A\right) \left(A^T A\right)^{-T} \\ &= \left(A^T A\right)^{-1} \end{aligned}$$

pois $A^T A$ é simétrica. Se representarmos por $\lambda_{\max}(B)$ e $\lambda_{\min}(B)$ o maior e menor valor próprio da matriz B, e como $A^T A \in$ SPD, então

$$\text{cond}_2(A) = \sqrt{\frac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)}} \tag{10.6}$$

Após estas considerações iniciais estamos em condições de nos debruçarmos sobre os quatro processos para a determinação da solução do problema de mínimos quadrados lineares (10.2).

(i) Método das equações normais

Este processo consiste em resolver o sistema

$$A^T Ax = A^T b$$

Como $A^T A \in$ SPD então existem algoritmos directos e iterativos muito eficientes para resolver o sistema (10.3), tanto no caso denso como no caso esparso e daí o grande interesse que esta

abordagem tem despertado. O processo poderá no entanto incorrer em algumas dificuldades numéricas. Com efeito,como $A^TA \in \text{SPD}$, então

$$cond_2(A^TA) = \frac{\lambda_{\max}(A^TA)}{\lambda_{\min}(A^TA)} = [cond_2(A)]^2$$

Assim, se A é relativamente mal condicionada, a matriz A^TA pode ser muito mal condicionada, o que vai implicar a determinação de uma solução pouco precisa.

(ii) Transformadas ortogonais

Se $A \in \mathbb{R}^{m\times n}$ tem característica n, então existe uma matriz ortogonal Q tal que

$$QA = \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{10.7}$$

Esta matriz Q é o produto de n matrizes de Householder ou de $\frac{n(n+1)}{2}$ matrizes de Givens. Se escrevermos

$$Qb = \begin{bmatrix} b^1 \\ b^2 \end{bmatrix}$$

então o sistema $Ax = b$ é equivalente a

$$\begin{bmatrix} R \\ 0 \end{bmatrix} x = \begin{bmatrix} b^1 \\ b^2 \end{bmatrix} \Leftrightarrow Rx = b^1$$

Portanto o sistema $Ax = b$ pode ser resolvido a partir dos seguintes passos:

(i) Calcular

$$QA = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

(ii) Determinar

$$\begin{bmatrix} b^1 \\ b^2 \end{bmatrix} = Qb$$

(iii) Resolver $Rx = b^1$.

A decomposição QR permite ainda obter boas estimativas do número de condição de A. Com efeito, tem-se

$$cond_2(A) = cond_2(R) \tag{10.8}$$

pelo que o número de condição de A se pode calcular aproximadamente usando o estimador LINPACK para a matriz R. Para demonstrar (10.8), provemos primeiramente que

$$cond_2(A) = cond_2(QA) \tag{10.9}$$

para qualquer matriz ortogonal Q. De (10.6) vem

$$\begin{aligned} cond_2(QA) &= \sqrt{\frac{\lambda_{\max}\left[(QA)^T(QA)\right]}{\lambda_{\min}\left[(QA)^T(QA)\right]}} \\ &= \sqrt{\frac{\lambda_{\max}\left[A^TQ^TQA\right]}{\lambda_{\min}\left[A^TQ^TQA\right]}} \\ &= \sqrt{\frac{\lambda_{\max}\left(A^TA\right)}{\lambda_{\min}\left(A^TA\right)}} = cond_2(A) \end{aligned}$$

o que mostra a igualdade (10.9). Então de (10.7) tem-se

$$\begin{aligned} cond_2(A) &= cond_2\left(\begin{bmatrix} R \\ 0 \end{bmatrix}\right) \\ &= \sqrt{\frac{\lambda_{\max}\left(\begin{bmatrix} R^T & 0 \end{bmatrix}\begin{bmatrix} R \\ 0 \end{bmatrix}\right)}{\lambda_{\min}\left(\begin{bmatrix} R^T & 0 \end{bmatrix}\begin{bmatrix} R \\ 0 \end{bmatrix}\right)}} \\ &= \sqrt{\frac{\lambda_{\max}(R^TR)}{\lambda_{\min}(R^TR)}} = cond_2(R) \end{aligned}$$

e isso demonstra a igualdade (10.8).

Seguidamente vamos discutir a utilidade e implementação da decomposição QR na resolução do sistema $Ax = b$, ou equivalentemente na determinação do mínimo global do problema de mínimos quadrados (10.2). Se A é uma matriz densa, então n matrizes de Householder

$$P_i = I_m - \beta_i v^{(i)} \left(v^{(i)}\right)^T$$

com

$$\beta_i = \frac{2}{\left(v^{(i)}\right)^T v^{(i)}}$$

e $v^{(i)} \in \mathbb{R}^m$ são necessárias para determinar a decomposição QR, em que

$$Q = P_n P_{n-1} \ldots P_1$$

Tal como no caso em que a matriz A é quadrada, apenas as componentes não nulas dos vectores $v^{(i)}$ e os elementos β_i são armazenados, conjuntamente com a matriz triangular superior, de acordo com o seguinte esquema:

$$A \rightarrow \begin{bmatrix} v_1^{(1)} & r_{12} & r_{13} & \ldots & r_{1n} \\ v_2^{(1)} & v_2^{(2)} & r_{23} & \ldots & r_{2n} \\ v_3^{(1)} & v_3^{(2)} & v_3^{(3)} & \ldots & r_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ v_n^{(1)} & v_n^{(2)} & v_n^{(3)} & \ldots & v_n^{(n)} \\ v_{n+1}^{(1)} & v_{n+1}^{(2)} & v_{n+1}^{(3)} & \ldots & v_{n+1}^{(n)} \\ \vdots & \vdots & \vdots & & \vdots \\ v_m^{(1)} & v_m^{(2)} & v_m^{(3)} & \ldots & v_m^{(n)} \end{bmatrix} \tag{10.10}$$

$$\beta = \begin{bmatrix} \beta_1 & \beta_2 & \ldots & \beta_n \end{bmatrix} \; r = \begin{bmatrix} r_{11} & r_{22} & \ldots & r_{nn} \end{bmatrix}$$

Além disso o processo é estável e como vimos anteriormente não altera o número de condição de A. Por isso este processo é muito recomendado para a resolução do sistema (10.1) quando a matriz A é densa. É fácil de ver que o número de operações necessárias para obter a resolução do sistema é superior ao que é necessário para o método das equações normais e essa diferença aumenta com $m - n$. Apesar dessa desvantagem, a decomposição QR é normalmente mais utilizada que as equações normais quando a matriz A é densa de dimensão pequena.

Consideremos agora o caso em que A é esparsa de dimensão elevada. A necessidade de armazenar os vectores $v^{(i)}$ torna este processo impraticável para este tipo de problemas. O método das equações semi-normais apresentado a seguir procura ultrapassar este inconveniente.

(ii) Método das equações semi-normais

Seja

$$QA = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Então

$$A^T A = \begin{bmatrix} R^T & 0 \end{bmatrix} QQ^T \begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} R^T & 0 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} = R^T R$$

O método das equações semi-normais consiste em obter a decomposição de Cholesky $R^T R$ usando a factorização QR e depois usar essa decomposição no método das equações normais. Portanto o processo consiste dos seguintes passos:

(i) Determinar

$$QA = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

(ii) Resolver $R^T Rx = A^T b$.

É de notar que a matriz Q não precisa de ser armazenada, o que torna este processo bastante mais útil para matrizes esparsas. A precisão da solução obtida por este processo é normalmente inferior à que é obtida pela decomposição QR. Por essa razão este processo não é usado para matrizes densas. Contudo a experiência computacional mostra que o uso de uma iteração de refinamento iterativo é normalmente suficiente para melhorar bastante a precisão da solução. A utilidade deste processo na resolução de sistemas com matrizes esparsas de dimensão elevada prende-se com essa propriedade e com o facto de a decomposição de Cholesky $R^T R$ ser única. Segundo esse processo, a Fase de Análise para a matriz $A^T A$ é primeiramente efectuada e obtém-se uma matriz de permutação P tal que

$$P^T (A^T A) P$$

conduz a poucos enchimentos. Mas

$$P^T (A^T A) P = (AP)^T (AP)$$

portanto na Fase de Factorização deve-se determinar a decomposição QR da matriz AP, isto é, da matriz que se obtém de A por reordenação das suas colunas. Então tem-se

$$QAP = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Na Fase de Solução há que resolver o sistema

$$R^T R(P^T x) = b$$

ou seja,

$$R^T y = b,\ Rx = y,\ \bar{x} = Px$$

Como a decomposição $R^T R$ de Cholesky é única, então o número de elementos não nulos de R é o mesmo da matriz que se obteria aplicando o algoritmo de decomposição directamente à matriz $A^T A$. No entanto o número de operações para obter R é normalmente superior no caso das equações semi-normais. Isso implica que este processo é normalmente mais demorado que o método das equações normais. Contudo, as equações semi-normais com uma iteração de refinamento iterativo conduzem em geral a soluções mais precisas, particularmente em sistemas com matrizes mal condicionadas.

(iv) Sistema Aumentado

As equações normais (10.3) podem ser escritas na forma

$$A^T(b - Ax) = 0$$

ou ainda

$$\left\{ \begin{array}{rcl} A^T r & = & 0 \\ r & = & b - Ax \end{array} \right.$$

Este sistema pode por sua vez ser escrito na seguinte forma matricial

$$\left[\begin{array}{cc} I & A \\ A^T & 0 \end{array} \right] \left[\begin{array}{c} r \\ x \end{array} \right] = \left[\begin{array}{c} b \\ 0 \end{array} \right] \tag{10.11}$$

O processo do sistema aumentado consiste exactamente em resolver o sistema (10.11). A sua matriz é simétrica indefinida de ordem $m + n$. Por essa razão este processo não é utilizado no caso denso. Se A é uma matriz esparsa de ordem elevada, então este processo tem utilidade pelas seguintes razões:

- o número de condição da matriz do sistema (10.11) é da ordem do número de condição de A, o que vai implicar a determinação de soluções bastante precisas.
- a matriz é bastante esparsa.
- existe software de grande qualidade para matrizes esparsas simétricas indefinidas, como por exemplo o programa MA27 anteriormente referido, assim como uma sua extensão MA47 especialmente vocacionada para este tipo de sistemas.

Como conclusão final, podemos dizer que em situações em que a precisão da solução é o aspecto mais importante, a decomposição QR no caso denso e as equações semi-normais ou o sistema aumentado no caso esparso são aconselháveis. Se a preferência for para um menor tempo de execução, então o método das equações normais deve ser usado em ambos os casos.

10.2 Métodos directos para sistemas horizontais

Como é perfeitamente conhecido o sistema (10.1) tem uma infinidade de soluções se $c(A) = m < n$. Nesta secção iremos apenas considerar processos de determinar a solução de norma mínima, isto é, a solução do seguinte problema de optimização

$$\begin{array}{rl} \text{Minimize} & ||x||_2 \\ \text{sujeito a} & Ax = b \end{array} \tag{10.12}$$

que por sua vez é equivalente a

$$\begin{array}{rl} \text{Minimize} & \frac{1}{2}||x||_2^2 \\ \text{sujeito a} & Ax = b \end{array}$$

Usando a teoria dos multiplicadores de Lagrange podemos concluir que a solução óptima deste programa satisfaz as seguintes equações

$$\begin{array}{rcl} I_n x - A^T y & = & 0 \\ Ax & = & b \end{array} \tag{10.13}$$

ou ainda

$$\begin{array}{rcl} x & = & A^T y \\ A^T A y & = & b \end{array} \tag{10.14}$$

Como A tem característica completa ($c(A) = m$), então a matriz AA^T é SPD e o sistema $AA^T y = b$ tem solução única. A solução de norma mínima é obtida multiplicando a matriz A^T por esse vector y.

Antes de apresentarmos os algoritmos para a determinação da solução do sistema com norma mínima , iremos definir o número de condição da matriz rectangular desse sistema. Tal como anteriormente, consideremos a pseudoinversa A^+ de A a partir da seguinte definição

$$A^+ = A^T(AA^T)^{-1} \in \mathbb{R}^{n\times m}$$

Então

$$AA^+ = (AA^T)(AA^T)^{-1} = I_m$$

e

$$\begin{aligned} cond_2(A) &= ||A||_2||A^+||_2 \\ &= \sqrt{\rho(AA^T)}\sqrt{\rho((A^+)^TA^+)} \end{aligned}$$

Mas como AA^T é simétrica, tem-se

$$\begin{aligned} (A^+)^TA^+ &= (AA^T)^{-T}AA^T(AA^T)^{-1} \\ &= (AA^T)^{-1}(AA^T)(AA^T)^{-1} = (AA^T)^{-1} \end{aligned}$$

Donde

$$\begin{aligned} cond_2(A) &= \sqrt{\rho(AA^T)}\sqrt{\rho((A^+)^TA^+)} \\ &= \sqrt{\frac{\lambda_{\max}(AA^T)}{\lambda_{\min}(AA^T)}} \end{aligned} \tag{10.15}$$

com $\lambda_{\max}(B)$ e $\lambda_{\min}(B)$ o maior e menor valores próprios da matriz B.

Tal como no caso anterior, existem quatro processos para a resolução do sistema $Ax = b$, que são discutidos a seguir.

(i) Método das equações normais

Este processo consiste em usar as fórmulas (10.14). Como

$$AA^T \in \text{SPD}$$

então o sistema $AA^Ty = b$ pode ser resolvido eficientemente, tanto no caso denso como esparso, usando a decomposição LDL^T dessa matriz.

A desvantagem desse processo reside na possibilidade da matriz desse sistema poder ser mal condicionada. Com efeito, da fórmula (10.15) vem

$$cond_2(AA^T) = \frac{\lambda_{\max}(AA^T)}{\lambda_{\min}(AA^T)} = [cond_2(A)]^2 \tag{10.16}$$

Tal como no caso dos sistemas verticais essa é a única razão para a existência dos processos alternativos, que iremos descrever a seguir.

(ii) Transformadas Ortogonais

Como vimos na secção anterior, tem-se

$$QA^T = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

ou seja

$$AQ = \begin{bmatrix} R^T & 0 \end{bmatrix}$$

Então$Ax = b$ é equivalente a

$$AQQ^T x = b$$

ou ainda

$$\left[\begin{array}{cc} R^T & 0 \end{array}\right] (Q^T x) = b$$

Se escrevermos

$$y = Q^T x = \left[\begin{array}{c} y^{(1)} \\ y^{(2)} \end{array}\right],$$

tem-se

$$\left[\begin{array}{cc} R^T & 0 \end{array}\right] \left[\begin{array}{c} y^{(1)} \\ y^{(2)} \end{array}\right] = b$$

Portanto

$$R^T y^{(1)} = b \tag{10.17}$$

e $y^{(2)}$ pode ser um vector qualquer. Fazendo $y^{(2)} = 0$, uma solução do sistema $Ax = b$ é calculada por

$$x = Q \left[\begin{array}{c} y^{(1)} \\ 0 \end{array}\right]. \tag{10.18}$$

Além disso x é solução de norma mínima, pois a norma ℓ_2 de um vector não é alterada por premultiplicação por uma matriz ortogonal.

Tal como na secção anterior, é possível provar que

$$cond_2(R) = cond_2(A)$$

pelo que este processo tem bastante interesse do ponto de vista das suas propriedades numéricas, possibilitando a obtenção de soluções bastante precisas. O número de operações requeridas por este processo é superior ao das equações normais sendo essa diferença tanto maior quanto maior for a diferença $n - m$. Esta desvantagem não obsta a que este processo das transformadas ortogonais seja o mais recomendado para a resolução do sistema (10.1) quando a matriz A tem dimensão relativamente pequena. Nesse caso são usadas matrizes de Householder, sendo a sua armazenagem feita segundo um esquema semelhante ao apresentado em (10.10) considerando linhas em vez de colunas.

A necessidade de armazenagem da matriz ortogonal Q (ou dos vários vectores $v^{(i)}$) torna este processo impraticável para matrizes esparsas de dimensão elevada. O processo das equações semi-normais foi desenvolvido para esse caso e é discutido a seguir.

(iii) Método das equações semi-normais

Este processo resolve o sistema $Ax = b$ a partir das equações normais (10.14), mas usando a decomposição QR de A para obter a decomposição de Cholesky de AA^T. Com efeito, tem-se

$$QA^T = \left[\begin{array}{c} R \\ 0 \end{array}\right] \Leftrightarrow AA^T = AQ^T QA^T = \left[\begin{array}{cc} R^T & 0 \end{array}\right] \left[\begin{array}{c} R \\ 0 \end{array}\right] = R^T R$$

e desta forma o processo de resolução de (10.1) é dado por

(i) Calcular a decomposição QR de A

$$QA^T = \left[\begin{array}{c} R \\ 0 \end{array}\right]$$

sem armazenar a matriz Q.

(ii) Resolver $R^T R y = b$.

(iii) Calcular $x = A^T y$.

Este processo é particularmente útil para matrizes esparsas. Neste caso o algoritmo do grau mínimo deve ser aplicado para determinação de uma ordenação definida por uma matriz de permutação P. Como

$$P(AA^T)P^T = (PA)(PA)^T$$

então as linhas de A devem ser dispostas segundo essa ordenação, sendo a decomposição QR de A obtida em seguida. Além disso, as matrizes A (ou A^T) e R devem ser armazenadas em esquemas de colecção de vectores. Este processo é normalmente mais demorado do que as equações normais, mas conduz a soluções mais precisas, principalmente se a solução obtida por este processo for melhorada por uma iteração de refinamento iterativo. Isso torna este processo mais recomendado do que as equações normais quando se pretende uma solução com uma boa precisão.

(iv) Sistema aumentado

As equações (10.13) podem ser escritas na forma

$$\begin{array}{rcl} x - A^T y & = & 0 \\ -Ax & = & -b \end{array}$$

ou ainda

$$\left[\begin{array}{cc} I_n & -A^T \\ -A & 0 \end{array} \right] = \left[\begin{array}{c} x \\ y \end{array} \right] = \left[\begin{array}{c} 0 \\ -b \end{array} \right] \tag{10.19}$$

O método do sistema aumentado consiste em resolver este sistema (10.19). Como a matriz deste sistema tem ordem $m+n$, este processo não deve ser utilizado para matrizes densas. Além disso, a matriz deste sistema é simétrica e o seu número de condição é sensivelmente da ordem do da matriz A. Por outro lado existe software de grande qualidade, nomeadamente as já mencionadas MA27 e MA47, e por essa razão alguns autores aconselham este processo na resolução de sistemas horizontais.

10.3 Métodos iterativos para sistemas rectangulares

Estes algoritmos procuram resolver os sistemas

$$A^T Ax = A^T b \text{ ou } AA^T x = b$$

sem calcular as suas matrizes explicitamente. Iremos apenas considerar o primeiro tipo de sistemas, pois a discussão dos algoritmos para sistemas horizontais de norma mínima é semelhante. Como

$$cond_2(A^T A) = [cond_2(A)]^2$$

então estes processos têm grandes dificuldades de convergência quando A é relativamente mal condicionada e por isso raramente são usados para sistemas de pequenas dimensões. No entanto têm utilidade quando as dimensões são elevadas, particularmente quando A é bem condicionada ou é conhecido um bom precondicionamento.

(i) Método de Jacobi

Como no caso quadrado, o método de Jacobi é baseado na partição

$$A^T A = D - E,\ D = diag(A^T A) \tag{10.20}$$

Então

$$\begin{array}{rcl} Dx^{(k+1)} & = & A^T b + Ex^{(k)} \\ & = & A^T b + (D - A^T A)x^{(k)} \\ & = & Dx^{(k)} + A^T(b - Ax^{(k)}) \end{array}$$

Donde

$$x^{(k+1)} = x^{(k)} + D^{-1}A^T(b - Ax^{(k)})$$

e deste modo o método de Jacobi pode ser escrito na forma

$$\left\{ \begin{array}{rcl} r^{(k)} & = & b - Ax^{(k)} \\ x^{(k+1)} & = & x^{(k)} + D^{-1}A^T r^{(k)} \end{array} \right. \tag{10.21}$$

Para completar a descrição do processo é necessário explicar como se calcula a matriz $D = diag(A^T A)$. Se A está armazenada por colunas, a matriz D é muito fácil de obter. Com efeito, se

$$A = \left[\begin{array}{cccc} A_{.1} & A_{.2} & \ldots & A_{.n} \end{array} \right]$$

então

$$A^T = \left[\begin{array}{c} A_{.1}^T \\ A_{.2}^T \\ \vdots \\ A_{.n}^T \end{array} \right]$$

e por isso

$$D = diag(A^T A) = \left[\begin{array}{cccc} diag(A_{.1}^T A_{.1}) & & & \\ & diag(A_{.2}^T A_{.2}) & & \\ & & \ddots & \\ & & & diag(A_{.n}^T A_{.n}) \end{array} \right]$$

Tendo em conta essa armazenagem da matriz, é fácil de calcular o vector $r^{(k)}$ a partir do produto da matriz A pelo vector denso $x^{(k)}$. O vector $x^{(k+1)}$ é então determinado por

$$x_j^{(k+1)} = x_j^{(k)} + \frac{1}{A_{.j}^T A_{.j}} A_{.j}^T r^{(k)},\ j = 1, 2, \ldots, n$$

(ii) Métodos de Gauss-Seidel e SOR

A partição associada ao método de Gauss-Seidel é dada por

$$A^T A = (L + D) + L^T$$

com D e L respectivamente a diagonal e o triângulo inferior de $A^T A$. Então

$$(L + D)x^{(k+1)} = -L^T x^{(k)} + A^T b$$

e portanto

$$x^{(k+1)} = x^{(k)} + D^{-1}\left\{ A^T b - \left[Lx^{(k+1)} + (D + L^T)x^{(k)} \right] \right\}$$

Se $z \in \mathbb{R}^n$ é o vector correspondente à iteração $k + 1$ então tem-se

$$\begin{array}{rcl} z^{(1)} & = & x^{(k)} \\ z^{(j+1)} & = & z^{(j)} + \left[D^{-1}A^T(b - Az^{(j)}) \right]_j e^j, j = 1, \ldots, n \\ x^{(k+1)} & = & z^{(n+1)} \end{array}$$

com e^j o vector j da base canónica. O segundo grupo de equações pode ser escrito na forma

$$z^{(j+1)} = z^{(j)} + \frac{A_{.j}^T(b - Az^{(j)})}{A_{.j}^T A_{.j}} e^j, j = 1, \ldots, n$$

Introduzindo agora o vector

$$r^{(j)} = b - Az^{(j)}$$

então tem-se

$$z^{(j+1)} = z^{(j)} + \frac{A_{.j}^T r^{(j)}}{A_{.j}^T A_{.j}} e^j, j = 1, \ldots, n$$

com $A_{.j}$ a coluna j da matriz A. Além disso

$$r^{(j+1)} = b - Az^{(j+1)} = b - Az^{(j)} - \delta_j Ae^j = r^{(j)} - \delta_j Ae^j = r^{(j)} - \delta_j A_{.j}$$

e portanto o algoritmo para o método de Gauss-Seidel para sistemas verticais pode ser apresentado da seguinte forma

Algoritmo 55 *[Método de Gauss-Seidel]*

Para $k = 0, 1, \ldots$

$z^{(1)} = x^{(k)}$

$r^{(1)} = b - Az^{(1)}$

Para $j = 1, \ldots, n$

$\delta_j = \dfrac{A_{.j}^T r^{(j)}}{A_{.j}^T A_{.j}}$

$z^{(j+1)} = z^{(j)} + \delta_j e^{(j)}$

$r^{(j+1)} = r^{(j)} - \delta_j A_{.j}$

$x^{(k+1)} = z^{(n+1)}$

É agora relativamente simples ver que o método SOR tem a mesma forma, com δ_j agora dado por

$$\delta_j = \frac{\omega A_{.j}^T r^{(j)}}{A_{.j}^T A_{.j}}, \ \omega \in]0, 2[$$

em que ω é o parâmetro de relaxação.

Para finalizar, há a referir que, em virtude de $A^T A \in$ SPD, o método SOR converge para qualquer $0 < \omega < 2$. Além disso, a matriz A deve estar armazenada por colunas numa implementação do método quando A é uma matriz esparsa de dimensão elevada.

(iii) Método dos gradientes conjugados

Como $A^T A \in$ SPD, então o método dos gradientes conjugados pode ser utilizado para resolver o sistema $A^T Ax = A^T b$. De acordo com a notação usada na secção 8.5 esse processo tem a forma:

Algoritmo 56

Dado x_0

Calcule $r_0 = b - Ax_0, p_0 = A^T r_0$

Para $k = 0, 1, \ldots$

$\alpha_k = \dfrac{||A^T r_k||_2^2}{||Ap_k||_2^2}$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k - \alpha Ap_k$

$\beta_{k+1} = \dfrac{||A^T r_{k+1}||_2^2}{||A^T r_k||_2^2}$

$$p_{k+1} = A^T r_{k+1} + \beta_{k+1} p_k$$

Tal como anteriormente, na implementação desse processo há que evitar o cálculo da matriz $A^T A$, para o que é importante considerar dois vectores $v_k \in \mathbb{R}^m$ e $s_k \in \mathbb{R}^n$ tais que

$$v_k = Ap_k, s_k = A^T r_k = -\left[A^T A x^{(k)} - A^T b\right]$$

Tendo em conta as definições desses vectores, podemos escrever o algoritmo na seguinte forma:

Algoritmo 57 *[Método dos Gradientes Conjugados (CG)]*

Dado x_0.

Calcule

$r_0 = b - Ax_0$
$s_0 = p_0 = A^T r_0$
$\gamma_0 = s_0^T s_0$

Para $k = 0, 1, \ldots$

$v_k = Ap_k$
$\alpha_k = \dfrac{\gamma_k}{v_k^T v_k}$
$x_{k+1} = x_k + \alpha_k p_k$
$r_{k+1} = r_k - \alpha_k v_k$
$s_{k+1} = A^T r_{k+1}$
Se $\gamma_{k+1} = s_{k+1}^T s_{k+1} < \epsilon$ *termine. De outro modo*
$\beta_k = \dfrac{\gamma_{k+1}}{\gamma_k}$
$p_{k+1} = s_{k+1} + \beta_k p_k$

Neste algoritmo, a tolerância ϵ é normalmente escolhida ligeiramente superior á precisão da máquina ϵ_M. De notar igualmente que a prestação do método depende bastante do número de condição de A, uma vez que $cond_2(A^T A) = [cond_2(A)]^2$.

(iv) Precondicionamento

Tal como para as matrizes quadradas, a técnica de precondicionamento consiste em introduzir uma matriz M não singular tal que AM^{-1} seja melhor condicionada que A. Mas

$$\begin{aligned} \min_{x \in \mathbb{R}^n} ||Ax - b||_2 \quad &\Leftrightarrow \quad \min_{y \in \mathbb{R}^n} ||AM^{-1}y - b||_2, \ Mx = y \\ &\Leftrightarrow \quad \left(AM^{-1}\right)^T \left(AM^{-1}\right) y = \left(AM^{-1}\right)^T b, \ Mx = y \end{aligned}$$

e por isso o método dos gradientes conjugados precondicionado (PCG) tem a seguinte forma:

Algoritmo 58 *[Método PCG]*

Dado x_0

Calcule $r_0 = b - Ax_0$

Resolva

$M^T s_0 = A^T r_0$

$Mp_0 = s_0$.

Calcule $\gamma_0 = s_0^T s_0$

Para $k = 0, 1, \ldots$

$v_k = Ap_k$

$\alpha_k = \dfrac{\gamma_k}{v_k^T v_k}$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k v_k$

Resolva $M^T s_{k+1} = A^T r_{k+1}$

Se $\gamma_{k+1} = s_{k+1}^T s_{k+1} < \epsilon$ *termine. De outro modo*

$\beta_k = \dfrac{\gamma_{k+1}}{\gamma_k}$

Resolva $M\tilde{s}_{k+1} = s_{k+1} \rightarrow s_{k+1}$

$p_{k+1} = s_{k+1} + \beta_k p_k$

Tal como no caso de um sistema com uma matriz quadrada, a matriz M deve ser escolhida por forma a que os sistemas lineares com essa matriz sejam de fácil resolução. Existem alguns casos possíveis entre os quais distinguimos a já analisada Decomposição Incompleta Sem Enchimentos. As matrizes ortogonais podem também ser úteis neste contexto. Suponhamos que A tem um número reduzido p de linhas pouco esparsas, e que sem perda de generalidade essas linhas são justamente as últimas p linhas da matriz. Se

$$A = \begin{bmatrix} B \\ E \end{bmatrix}, B \in \mathbb{R}^{(m-p)\times n}, E \in \mathbb{R}^{p\times n}$$

então

$$A^T A = B^T B + E^T E$$

e por isso podemos usar $B^T B$ como aproximação para $A^T A$. Se

$$QB = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

então $B^T B = R^T R$ e a matriz de precondicionamento será justamente $M = R^T R$.É evidente que o precondicionamento será tanto mais eficiente quanto menor for o número p de linhas que não se consideram na formação da matriz B. Sugerimos [Portugal], [Matstoms] para bons estudos desse tipo de precondicionamento e dos outros algoritmos directos e iterativos para a resolução do problema de mínimos quadrados lineares com matrizes esparsas de dimensão elevada.

Exercícios

1. Determine a solução dos mínimos quadrados para os seguintes sistemas:

   (a)
   $$\begin{cases} x_1 - x_2 = 1 \\ x_1 + x_2 = 2 \\ 2x_1 - 3x_2 = 1 \\ 3x_1 - x_2 = 3 \end{cases}$$

   (b)
   $$\begin{cases} x_1 + x_3 = 2 \\ x_2 - x_3 = 0 \\ x_1 - 2x_3 = -1 \\ x_2 + x_3 = 2 \end{cases}$$

usando os quatro métodos directos discutidos neste capítulo e determine os resíduos associados a essas soluções.

2. Determine a equação da recta que aproxima no sentido dos mínimos quadrados os pontos $(x_i, y_i) \in \mathbb{R}^2$ dados pela seguinte tabela

x_i	1	2	3	4	5	6
y_i	1	1.3	1.8	2.5	3.2	4.0

usando um método à sua escolha.

3. Determine a equação do polinómio de grau 2 que aproxima no sentido dos mínimos quadrados os pontos $(x_i, y_i) \in \mathbb{R}^2$ dados na tabela da alínea anterior usando um método à sua escolha.

4. Desenvolva uma implementação do método de Gauss-Seidel para a determinação da solução dos mínimos quadrados de um sistema vertical com uma matriz esparsa armazenada numa colecção de vectores.

5. Considere o sistema vertical $Ax = b$, com $A \in \mathbb{R}^{m\times n}$ uma matriz de característica completa $n < m$. Mostre que se perturbarmos o vector b em $\delta b \in \mathbb{R}^m$ então a solução do respectivo sistema $(x + \delta x)$ satisfaz
$$\frac{||x + \delta x||}{||x||} \leq cond(A)\frac{||b + \delta b||}{||b||}$$
com $||\cdot||$ uma norma subordinada.

6. Determine as soluções de norma mínima dos seguintes sistemas horizontais:

   (a)
   $$\begin{cases} x_1 + x_2 + x_3 + x_4 = 4 \\ -x_1 + x_2 + x_3 = 2 \end{cases}$$

   (b)
   $$\begin{cases} x_1 - x_2 + x_4 = 2 \\ -x_1 + x_3 - x_4 = -2 \\ x_2 - x_3 + x_4 = 1 \end{cases}$$

   usando os quatro métodos directos discutidos neste capítulo.

7. Desenvolva processos baseados na decomposição LU e na decomposição QR para a determinação de uma solução básica de um sistema horizontal (recorde a noção de solução básica apresentada no capítulo 3).

8. Determine uma solução básica dos sistemas da pergunta 6 que não tenha norma mínima, usando os processos discutidos na pergunta anterior.

9. Descreva os métodos de Jacobi, Gauss-Seidel, SOR e Gradientes Conjugados para a determinação da solução de norma mínima de um sistema horizontal.

10. Desenvolva uma implementação do método de gradientes conjugados para a determinação da solução de norma mínima de um sistema horizontal com uma matriz esparsa armazenada num esquema de colecção de vectores.

11. Seja H uma matriz SPD. Indique os processos de determinação da solução de norma $||\cdot||_H$ mínima ($||y||_H = \sqrt{y^T H y}$) quando:

    (a) H é uma matriz diagonal.

    (b) H é uma matriz não diagonal.

12. Determine as soluções de norma $||\cdot||_H$ mínima dos sistemas horizontais da pergunta 6, com $H = diag(2, 1, 3, 1)$.

Bibliografia

1. O. Axelsson, Iterative Solution Methods, Cambridge University Press, New York, 1994.
2. R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine e H. Van der Vost, Templates for the Solution of Linear Systems, SIAM, Philadelphia, 1994.
3. A. Berman e R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.
4. A. Bjorck, Algorithms for linear least squares problems, em "Computer Algorithms for Solving Linear Algebraic Problems", editado por E. Spedicato, NATO ASI Series, Springer-Verlag, Berlim, 1991.
5. J. Bunch e L. Kaufman, Some stable methods for calculating inertia and solving symmetric linear systems, *Mathematics of Computation* **31** (1977) 163–179.
6. R. L. Burden, J. D. Faires e A. C. Reynolds, *Numerical Analysis*, Prindle, Weber & Schmidt, Boston, 1981.
7. R. W. Cottle, J. S. Pang e R. E. Stone, The Linear Complementarity Problem, Academic Press, 1992.
8. J. E. Dennis e R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
9. I. S. Duff, A. M. Erisman e J. K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, 1986.
10. M. Fiedler, *Special Matrices and their Applications in Numerical Mathematics*, Kluwer, Boston, 1986.
11. G. E. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, New Jersey, 1967.
12. A. George e J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, New Jersey, 1981.
13. P. E. Gill, W. Murray e M. H. Wright, *Practical Optimization*, Academic Press, New York, 1981.
14. P. E. Gill, W. Murray e M. H. Wright, *Numerical Linear Algebra and Optimization*, Addison-Wesley, New York, 1991.
15. G. H. Golub e C. F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, 1983.
16. L. A. Hageman e D. M. Young, *Applied Iterative Methods*, Academic Press, New York, 1981.
17. W. Hager, *Applied Numerical Linear Algebra*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.
18. A. Jennings e J. C. McKeown, *Matrix Computations*, John Wiley & Sons, New York, 1992.

19. P. Matstoms, Sparse QR Factorization with applications to Linear Least Squares Problems, Tese de Doutoramento, Departamento de Matemática, Universidade de Linkoping, Suécia, 1994.

20. L. Mirsky, An Introduction to Linear Algebra, Clarendon Press, Oxford, 1972.

21. J. M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1988.

22. J. M. Patrício, *Um estudo sobre o Método do Invólucro para Matrizes Esparsas Simétricas Positivas Definidas*, Departamento de Matemática, Universidade de Coimbra, 1991.

23. S. Pissanetzky, *Sparse Matrix Technology*, Academic Press, New York, 1984.

24. L. F. Portugal, Resolução de Problemas de Mínimos Quadrados Lineares, Tese de Mestrado, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, 1992.

25. R. Schnabel e E. Eskow, A new modified Cholesky factorization, *SIAM Journal on Scientific and Statistical Computing* **11**(1990)1136-1158.

26. G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

27. R. S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, New Jersey, 1962.

28. D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

29. Z. Zlatev, *Computational Methods for General Sparse Matrices*, Kluwer, London, 1991.