Esparsidade, Estrutura, Estabilidade e Escalamento em Álgebra Linear Computacional

por

Julio M. Stern

Departamento de Ciência de Computação do
Instituto de Matemática e Estatística da
Universidade de São Paulo

Abril de 1994

Conteúdo

Propomos, para a IX Escola de Computação, o curso acompanhado de livro texto: **Esparsidade, Estrutura, Escalamento e Estabilidade em Álgebra Linear Computacional**. Embora abrangente o suficiente para a compreensão dos principais problemas da área, a tônica do curso é a solução de sistemas lineares esparsos e-ou estruturados. Na primeira seção da introdução damos uma visão panorâmica sobre o tema e a organização do livro. Neste prefácio ressaltamos algumas motivações para a inclusão do curso na IX Escola de Computação, e esclarecemos a forma e o conteúdo das palestras a serem dadas durante o mesmo.

0.1 Importância da Área

Grande parte do processamento de dados em ciência envolve computação numérica, e a maior parte desta computação numérica são rotinas básicas de álgebra linear. Várias aplicações em engenharia, física, pesquisa operacional, administração ou matemática, geram problemas lineares cujas matrizes são esparsas (entre tantos outros: resolução de equações diferenciais, análise de sistemas, circuitos ou estruturas, programação linear e não linear, otimização de fluxos em redes, etc...). Ademais, observa-se que a densidade destas matrizes decresce com a dimensão do problema: tipicamente apenas $n^{\frac{3}{2}}$ ou $n\log(n)$ de seus n^2 elementos são não nulos. Assim, quanto maiores e mais complexas os problemas (e usuários sempre querem resolver modelos maiores), mais importante se torna usar eficientemente a esparsidade e a estrutura das matrizes envolvidas. Por exemplo, na área de otimização, problemas hoje considerados de grande porte envolvem milhões de equações, sendo a maioria destes problemas altamente estruturados, muito esparsos, e usando mais de 99% do tempo de resolução em rotinas básicas de álgebra linear!

0.2 Interdisciplinaridade

Um aspecto que, a nosso ver, torna o tema interessante para um evento como a Escola de Computação é sua interdisciplinaridade, vejamos:

- A resolução dos problema de álgebra linear envolve os aspectos clássicos de complexidade, convergência e estabilidade de análise numérica.

- O tratamento de esparsidade e estrutura é um problema essencialmente combinatório, envolvendo teoria de grafos, hipergrafos, e heurísticas para solução de problemas de programação matemática discreta.

- A paralelização destes algoritmos, ou o desenvolvimento de novos métodos, em ambientes tão diversos como máquinas de memória compartilhada ou redes de estações de trabalho, vem liderando as pesquisas na área nos últimos anos.

0.3 Serventia do Livro Texto

O curso que se segue foi recentemente montado como a disciplina MAC-795, Métodos Computacionais da Álgebra Linear, no programa de mestrado do Departamento de Ciência da Computação da Universidade de São Paulo. Cursos de métodos computacionais da álgebra linear tem se popularizado nos últimos anos em muitos departamentos de computação, engenharia e matemática aplicada. Era minha intenção escrever mais um capítulo sobre métodos iterativos mas, ao perceber que meu rascunho duplicava o tamanho do presente livro, resolvi postergar a tarefa para outra ocasião.

0.4 Plano de Aulas

Em cinco palestras de 90 minutos é possível dar um bom panorama da área, sua importância, problemas, métodos, e perspectivas. O material das palestras foi distribuído da seguinte forma:

1. **Introdução:** Visão geral da área, sua importância, origem de problemas de grande porte, e aspectos essenciais para a sua solução.

2. **Esparsidade, Eliminação Assimétrica:** Minimização de preenchimento local, outras heurísticas, atualizações de base.

3. **Esparsidade, Eliminação Simétrica:** Árvores de eliminação, heurísticas de ordenação, heurísticas de dissecção.

4. **Estrutura:** Esparsidade macroscópica, métodos de blocos, heurísticas de redução a formas blocadas.

5. **Paralelismo:** Uso de esparsidade $\times$ uso de estrutura, granularidade, e potencialidades de ambientes com diferentes coeficientes entre velocidade de processamento e velocidade de comunicação.

0.5 Comentário sobre a Bibliografia de Suporte

Para a parte clássica de análise numérica há uma farta variedade de livros texto, como: [Stewart-73] e [Golub-83]. Existem alguns livros ou coletâneas sobre matrizes esparsas em ambiente seqüencial, sendo os principais: [Rose-72], [Tewarson-73], [Bunch-76], [Duff-79], [George-81] e também [Pissan-84] e [Duff-86]. Com exceção dos dois últimos, estes livros já estão bastante desatualizados e esgotados.

Álgebra Linear Computacional, sob o enfoque de Computação Paralela, é um campo de intensa pesquisa atual: Exclusivamente para matrizes densas há varias obras publicadas, x1como:

[Bertsekas-89] e [Dongarra-91]. Coletâneas de artigos sobre álgebra linear computacional em ambiente paralelo, trazendo alguns artigos sobre matrizes esparsas, são muito poucos; como: [Carey-89], [Vorst-89] e [Gallivan-90]. Nestas resenhas encontra-se também uma extensiva bibliografia comentada da área.

0.6 Agradecimentos

No DCC-IME-USP, Departamento de Ciência da Computação da Instituto de Matemática e Estatística da Universidade de São Paulo, contei sempre com a ajuda e o encorajamento de muitos colegas, como os Professores Marcos D. Gubitoso, Arnaldo Mandel, Kunio Okuda, Siang W. Song e Routo Terada. Sou especialmente grato ao coordenador do grupo de Programação Matemática, Professor Carlos Humes Jr.

No CEMCAP-IME-USP, Centro de Matemática e Computação Aplicadas, tive sempre o apoio dos Professores Marco Antonio Raupp, Pedro A. Morettin e Carlos A. B. Pereira. No NOPEF-USP, Núcleo de Otimização e Processos Estocásticos Aplicados à Economia e Finanças, contei com o companheirismo dos Professores Marcos Eugênio da Silva, José Carlos Santos e João Carlos Prandini. Em várias oportunidades recebi a colaboração do Eng. Fábio Nakano e do Prof. Jacob Zimbarg sobrinho.

Partes deste livro baseiam-se em trabalhos feitos com o Professor Stephen A. Vavasis, da Universidade de Cornell. A apresentação de alguns tópicos foi inspirada em disciplinas ministradas pelo Professor Thomas F. Coleman, na mesma Universidade. Em 1992 montamos a disciplina MAC-795, Métodos Computacionais da Álgebra Linear, para o programa de mestrado do DCC-IME-USP. A apostila que acompanhou o curso serviu de base para este livro. Devo a Paulo Régis Zanjácomo, presentemente na Universidade de Cornell, e a Professora Celma O. Ribeiro, da Escola Politécnica da USP, várias críticas, comentários, e sugestões que em muito melhoraram aquela primeira versão deste texto.

Dos promotores da IX Escola de Computação, inclusive do anônimo, crítico e bem humorado referee, e da Universidade Federal de Pernambuco (UFP-Recife), recebi todo o necessário suporte.

A todos estes amigos, à minha esposa, Marisa, e a meus filhos, Rafael, Ana Carolina e Deborah, minha gratidão.

Capítulo 1

INTRODUÇÃO

1.1 Panorama do Livro e seu Tema

O objetivo deste curso é o estudo de métodos computacionais para a resolução de sistemas lineares, $Ax = b$. Enfatizaremos quatro aspectos da construção de bons métodos computacionais:

- **Esparsidade**: Como resolver eficientemente sistemas cujas matrizes tenham baixa densidade de elementos não nulos.
- **Estrutura**: Como resolver eficientemente sistemas esparsos cujos elementos não nulos estão dispostos com uma certa regularidade na matriz de coeficientes.
- **Escalamento**: Como minimizar erros de arredondamento gerados ao operarmos com números de ordem de grandeza muito diferente, numa representação de precisão limitada.
- **Estabilidade**: Como lidar com o efeito cumulativo dos erros de arredondamento na solução final gerada pelo algoritmo.

As ferramentas relevantes para a análise e implementação de algoritmos eficientes de álgebra linear estão espalhados entre diversas áreas, áreas estas afins-mas-nem-tanto, como Teoria de Grafos e Hipergrafos, Álgebra Linear, Análise Numérica, e Teoria de Processamento Paralelo. O propósito destas notas é apresentar este material de forma razoavelmente coerente e didática.

O Capítulo 2 expõe o método de Gauss para solução de sistemas lineares, inversão, ou fatoração de matrizes. Os métodos para matrizes esparsas lidam com a estrutura da disposição dos elementos não nulos dentro da matriz, e esta estrutura é convenientemente descrita e manipulada em termos da teoria de grafos. O Capítulo 3 é um resumo de conceitos básicos desta teoria. Conceitos mais avançados (ou menos usuais) como grafos cordais, separadores, e hipergrafos, são tratados em outros capítulos.

No Capítulo 4 estudamos técnicas para tratar sistemas esparsos assimétricos, principalmente preenchimento local, e a heurística P3. Desenvolvimentos posteriores deste tema encontram-se também no capítulo 7. O Capítulo 5 expõem as Fatorações QR e de Cholesky, e sua utilização na solução de problemas de quadrados mínimos, programação quadrática, e construção de projetores. No Capítulo 6 estudamos técnicas para tratar sistemas esparsos simétricos; incluindo toda a necessária teoria de grafos cordais.

O capitulo 7 trata da estrutura de um sistema, que pode ser vista como regularidades no padrão de esparsidade, ou como a decomposição do sistema em sub-sistemas acoplados. Neste capítulo estudamos brevemente a paralelização de alguns dos algoritmos, tema que já aparece implicitamente em capítulos anteriores. O Capitulo 8 é dedicado à análise do efeito dos inevitáveis erros de arredondamento nos procedimentos computacionais. No Capitulo 9 analisamos quanto estes erros podem degradar a solução final de um problema. Estes dois capítulos tratam dos aspectos de Análise Numérica que, embora não sendo a tônica do curso, não podem ser desprezados.

O Capítulo 10 trata do problema de mudança de base, i.é., de atualizar a inversa de uma matriz quando nesta se substitui uma única coluna. Este problema é de fundamental importância para muitos algoritmos de Otimização.

Parte da avaliação numa disciplina como a proposta deve ser feita com exercícios-programa, como os dados ao longo das notas. É recomendável o uso de uma linguagem estruturada, que inclua entre seus tipos básicos, números reais de precisão simples e dupla, inteiros, campos de bits e ponteiros, que permita a fácil construção e manipulação de estruturas, e para a qual haja compiladores em computadores dos mais diversos portes. As linguagens C, C++ e FORTRAN-90 são sugestões naturais. Ambientes iterativos para cálculo matricial, como Matlab, são um grande estímulo à experimentação e comparação de diversos de métodos numéricos, facilitando a rápida prototipagem e teste de algoritmos. Apresentamos uma introdução a este tipo de ambiente como apêndice.

1.2 Notações

Utilizaremos letras maiúsculas, $A, B, \ldots$, para denotar matrizes, e letras minúsculas, $a, b, \ldots$, para vetores. Numa matriz ou vetor, os índices de linha ou coluna serão, respectivamente, subscritos ou superscritos à direita. Assim: A_i^j é o elemento na i-ésima linha e na j-ésima coluna da matriz A; b_i é o elemento na i-ésima linha do vetor-coluna b; e c^j é o elemento na j-ésima coluna do vetor-linha c.

Índices ou sinais à esquerda, superscritos ou subscritos, identificam vetores ou matrizes distintas. Assim:A, $'A$, ${}^{j}A$, ${}_{i}A$, ${}_{i}^{j}A$, $\ldots$ são matrizes distintas. Também a letra δ forma, à esquerda de uma letra latina o nome de um vetor ou matriz, como δa ou δA. As demais letras gregas usaremos geralmente para escalares ou funções.

Freqüentemente nos referimos aos blocos componentes de vetores ou matrizes, como por exemplo: se b e c são vetores linha $a = \begin{bmatrix} b & c \end{bmatrix}$ é um vetor linha cujos primeiros elementos são os elementos de b, e os elementos seguintes são os elementos de c.

Analogamente,

$$A = \begin{bmatrix} {}^1_1A & {}^2_1A & {}^3_1A \\ {}^1_2A & {}^2_2A & {}^3_2A \\ {}^1_3A & {}^2_3A & {}^3_3A \end{bmatrix}$$

representa uma matriz blocada, desde que as dimensões dos blocos sejam compatíveis.

Se A é uma matriz, A_i representa a i-ésima linha de A, e A^j sua j-ésima coluna, de modo que se A é mxn,

$$A = \begin{bmatrix} A^1 & A^2 & \dots & A^n \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{bmatrix}$$

Uma **matriz diagonal** será denotada como

$$diag(d) = D = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} , \quad d = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix}$$

O **vetor unitário**, $\mathbf{1}$, é o vetor, linha ou coluna, em que todos os elementos são iguais a 1, isto é, $\mathbf{1} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$. Assim a matriz identidade é $I = diag(\mathbf{1})$. O j-ésimo **versor** de dimensão n é I^j, a j-ésima coluna da matriz identidade de dimensão n. A transposta, a inversa e a transposta da inversa de uma matriz A são denotadas, respectivamente, por A', A^{-1} e A^{-t}.

O **determinante** de uma matriz A, nxn, é

$$det(A) = \sum_p S(p) A^1_{p(1)} A^2_{p(2)} \dots A^n_{p(n)}$$

onde $p = \begin{bmatrix} p^1 & p^2 & \dots & p^n \end{bmatrix}$ é uma **permutação** dos elementos de ${}^0p = \begin{bmatrix} 1 & 2 & \dots & n \end{bmatrix}$. O sinal de permutação, $S(p)$, é +1 ou -1 conforme o número de trocas de pares de elementos que é necessário fazer em p para retornar a ${}^0p$, seja par ou ímpar. O conjunto dos elementos em cada um dos termos na somatória da definição do determinante é denominado uma **diagonal** da matriz. A diagonal correspondente a pemutação ${}^0p$ é denominada **diagonal principal**. Uma matriz quadrada é singular se tiver determinante nulo. O **posto** de uma matriz A, $rank(A)$, é a dimensão de sua maior sub-matriz quadrada não singular.

Dado um sistema $Ax = b$, com matriz de coeficientes A nxn não singular, a **Regra Cramer** nos diz que o sistema tem por única solução

$$x \mid x_i = det(\begin{bmatrix} A^1 & \dots & A^{i-1} & b & A^{i+1} & A^n \end{bmatrix})/det(A).$$

O **traço** de uma matriz A, nxn, é a soma de seus elementos na diagonal principal:

$$tr(A) = \sum_{i=1}^{n} A_i^i \quad .$$

A **matriz booleana** associada à matriz A, $B(A)$, é a matriz, da mesma dimensão de A, em que $B(A)_i^j = 1$ se $A_i^j \neq 0$, e $B(A)_i^j = 0$ se $A_i^j = 0$. O complemento de uma matriz booleana B, é a matriz booleana $\bar{B}$, da mesma dimensão de B, tal que $\bar{B}_i^j = 1 \Leftrightarrow B_i^j = 0$.

Os conjuntos $N = \{1, 2, ..., n\}$ e $M = \{1, 2, ..., m\}$ serão freqüentemente usados como **domínios de índices**. Assim, se A é uma matriz mxn, faz sentido falar dos elementos A_i^j, $i \in M$, $j \in N$,

O número de **elementos não nulos**, ENNs, numa matriz A, mxn, é

$$enn(A) = \sum_{i,j=1}^{m,n} B(A)_i^j = \mathbf{1}'B(A)\mathbf{1}$$

de modo que $enn(A_i)$ e $enn(A^j)$ são, respectivamente, o número de elementos não nulos na i-ésima linha e na j-ésima coluna de A.

Dada uma função $\varphi()$, definida num domínio D, e $X \subset D$, definimos seu **argumento mínimo** $V = arg\min_{x \in X} \varphi(x)$ e **argumento máximo** $U = arg\max_{x \in X} \varphi(x)$ como, respectivamente, os conjuntos $V,\ U \subset D$ que minimizam ou maximizam a função φ em X. Assim, se por exemplo, $\varphi = x^2$, $x \in \Re$, $X = [-5, 5]$ e $Y =]-5, 5[$, então $arg\min_X \varphi(x) = arg\min_Y \varphi(x) = \{0\}$, $arg\max_X \varphi(x) = \{-5, 5\}$, $arg\max_Y \varphi(x) = \emptyset$.

Para realizar experiências computacionais utilizaremos por vezes os sistemas lineares de dimensão n e solução $x = \mathbf{1}$, com as seguintes matrizes de coeficientes e vetores independentes:

- **Binomial**:

$$^{n}B_i^j = \begin{cases} \begin{pmatrix} i \\ j-1 \end{pmatrix} & \Leftrightarrow \quad j = 1, 2, \ldots i \\ 0 & \Leftrightarrow \quad j > i \end{cases} \quad , \quad ^{n}b = \begin{bmatrix} 2^1 \\ 2^2 \\ \vdots \\ 2^n \end{bmatrix}$$

- **Hilbert**:

$$^{n}H_i^j = 1/(i + j - 1) \quad , \quad ^{n}h_i = \sum_{j=1}^{n} 1/(i + j - 1)$$

- **Tridiagonal**:

$$^{n}T_i^j = \begin{cases} -2 & \Leftrightarrow \quad i = j \\ -1 & \Leftrightarrow \quad |i - j| = 1 \\ 0 & \text{caso contrario} \end{cases} \quad , \quad ^{n}t = \begin{cases} 1 & \Leftrightarrow \quad i \in \{1, n\} \\ 0 & \text{caso contrario} \end{cases}$$

1.3 Representação de Matrizes Esparsas

Ao representarmos uma matriz esparsa gostaríamos de utilizar o mínimo de memória, idealmente apenas a posição e o valor dos ENNs, mas ao mesmo tempo ter a maior flexibilidade possível para acessar, modificar, inserir ou remover um elemento qualquer. Estes objetivos são algo conflitantes, o que motiva o uso de diversas representações:

Representação **estática por linhas**: Para uma matriz A, $m \times n$ com $enn(A) = l$, são utilizados um vetor real, $aias(k)$ $k \in L$, um vetor inteiro $aijs(k)$, $k \in L = \{1, \ldots l\}$, e um segundo vetor de inteiros, $aif(i)$, $i \in M$. Os vetores aias e aijs listam os valores e os índices de coluna dos ENN's, linha por linha. $aif(i)$ nos dá o fim, ou a posição do último elemento da linha i em $aias$ e $aijs$.

Representação **estática por colunas**: Para uma matriz A, $m \times n$ com $enn(A) = l$, são utilizados um vetor real, $ajas(k)$ $k \in L$, um vetor inteiro $ajis(k)$, $k \in L$, e um segundo vetor de inteiros, $ajf(j)$, $j \in N$. Os vetores $ajas$ e $ajis$ listam os valores e os índices de linha dos ENNs, coluna por coluna. $ajf(i)$ nos dá a posição do último elemento da coluna j em $ajas$ e $ajis$.

Representação de **lista ligada por linhas**: Cada ENN corresponde a uma célula contendo: O valor do ENN, o índice de coluna e um ponteiro para a célula do próximo ENN na linha. As células dos últimos ENNs em cada linha contém o ponteiro nulo, e um vetor de m ponteiros, $ancorai(i)$, nos dá a primeira célula de cada linha.

Representação de **lista ligada por colunas**: Cada ENN corresponde a uma célula contendo: O valor do ENN, o índice de linha e um ponteiro para a célula do próximo ENN na coluna. As células dos últimos ENNs em cada coluna contém o ponteiro nulo, e um vetor de n ponteiros, $ancoraj(j)$, nos dá a primeira célula de cada coluna.

Representação de **rede**: Cada ENN corresponde a uma célula contendo o valor do ENN A_i^j, os índices i e j, e ponteiros para a célula do próximo ENN na linha e na coluna. Dois vetores $ancorai(i)$ e $ancoraj(j)$ contém ponteiros para as primeiras células em cada linha e coluna.

Representação de **rede dupla**: Análoga a representação de rede, mas cada célula contém além de ponteiros para as próximas células ao **sul** (próximo ENN na coluna) e ao **leste** (próximo ENN na linha), também ponteiros para as células ao **oeste** e ao **norte**, i.e. para os ENN anteriores na linha e na coluna.

No exemplo 1 temos as diversas representações da matriz:

$$A = \begin{bmatrix} & 102 & & 104 & \\ 201 & & & & \\ 301 & 304 & & & \\ & & & & 405 \\ 501 & 502 & 503 & & \end{bmatrix} \quad , \quad l = enn(A) = 9 \ .$$

Representação estática por linhas:

$$aias = \left[\begin{array}{ccccccccc} 102 & 104 & 201 & 301 & 304 & 405 & 501 & 502 & 503 \end{array} \right] ,$$

$$aijs = \left[\begin{array}{ccccccccc} 2 & 4 & 1 & 1 & 4 & 5 & 1 & 2 & 3 \end{array} \right] ,$$

$$aif = \left[\begin{array}{ccccc} 2 & 3 & 5 & 6 & 9 \end{array} \right] .$$

Representação estática por colunas:

$$ajas = \left[\begin{array}{ccccccccc} 201 & 301 & 501 & 102 & 502 & 503 & 104 & 304 & 405 \end{array} \right] ,$$

$$ajis = \left[\begin{array}{ccccccccc} 2 & 3 & 5 & 1 & 5 & 5 & 1 & 3 & 4 \end{array} \right] ,$$

$$ajf = \left[\begin{array}{ccccc} 3 & 5 & 6 & 8 & 9 \end{array} \right] .$$

Representação de lista ligada por linhas:

$$\begin{array}{llll}
\text{ancorai}(1) \rightarrow & (2,\ 102,\ \rightarrow) & (4,\ 104,\ \dashv) & \\
\text{ancorai}(2) \rightarrow & (1,\ 201,\ \dashv) & & \\
\text{ancorai}(3) \rightarrow & (1,\ 301,\ \rightarrow) & (4,\ 304,\ \dashv) & \\
\text{ancorai}(4) \rightarrow & (5,\ 405,\ \dashv) & & \\
\text{ancorai}(5) \rightarrow & (1,\ 501,\ \rightarrow) & (2,\ 502,\ \rightarrow) & (3,\ 503,\ \dashv)
\end{array}$$

Representação de lista ligada por colunas:

$$\begin{array}{ccccc}
\text{ancoraj}(1) & \text{ancoraj}(2) & \text{ancoraj}(3) & \text{ancoraj}(4) & \text{ancoraj}(5) \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
(2,\ 201,\ \downarrow) & (1,\ 102,\ \downarrow) & (5,\ 503,\ \perp) & (1,\ 104,\ \downarrow) & (4,\ 405,\ \perp) \\
(3,\ 301,\ \downarrow) & (5,\ 502,\ \perp) & & (3,\ 304,\ \perp) & \\
(5,\ 501,\ \perp) & & & &
\end{array}$$

Representação em rede:

$$\begin{array}{cccccc}
 & \text{ancoraj}(1) & \text{ancoraj}(2) & \text{ancoraj}(3) & \text{ancoraj}(4) & \text{ancoraj}(5) \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
\text{ancorai}(1) \rightarrow & & \begin{pmatrix} 1 & 2 & 102 \\ & \downarrow & \rightarrow \end{pmatrix} & & \begin{pmatrix} 1 & 4 & 104 \\ & \downarrow & \rightarrow \end{pmatrix} & \\
\text{ancorai}(2) \rightarrow & \begin{pmatrix} 2 & 1 & 201 \\ & \downarrow & \dashv \end{pmatrix} & & & & \\
\text{ancorai}(3) \rightarrow & \begin{pmatrix} 3 & 1 & 301 \\ & \downarrow & \rightarrow \end{pmatrix} & & & \begin{pmatrix} 3 & 4 & 304 \\ & \perp & \dashv \end{pmatrix} & \\
\text{ancorai}(4) \rightarrow & & & & & \begin{pmatrix} 4 & 5 & 405 \\ & \perp & \dashv \end{pmatrix} \\
\text{ancorai}(5) \rightarrow & \begin{pmatrix} 5 & 1 & 501 \\ & \perp & \rightarrow \end{pmatrix} & \begin{pmatrix} 5 & 2 & 502 \\ & \perp & \rightarrow \end{pmatrix} & \begin{pmatrix} 5 & 3 & 503 \\ & \perp & \dashv \end{pmatrix} & &
\end{array}$$

Exercícios

1. Prove que

   (a) $(AB)' = B'A'$.

   (b) $(AB)^{-1} = A^{-1}B^{-1}$.

   (c) $(A')^{-1} = (A^{-1})'$.

2. Na representação estática por linhas, é realmente necessário termos aif além de $aias$ e $aijs$?

3. Considere uma matriz esparsa de estrutura regular e conhecida, por exemplo tri-diagonal, para a qual bastaria conhecermos o valor dos ENNs numa dada seqüência, por exemplo linha por linha. Suponha que um inteiro ou ponteiro ocupa 2 bytes e que um real ocupa 4 bytes. Dê o coeficiente de uso de memória de cada uma das outras representações em relação a está representação minimal.

4. (a) Considere uma matriz esparsa A de densidade, i.e. fração de elementos não nulos, α^2. Suponha que A está representada em rede. Queremos adicionar um novo ENN A_i^j à matriz. Supondo que os ENN estão aleatoriamente distribuídos, e tomando acesso a uma célula como operação elementar, qual a complexidade esperada desta operação? Explique sucintamente como realizar a operação.

   (b) Nas mesmas condições, queremos substituir duas linhas A_i e A_{i+1}, respectivamente, pelas combinações lineares $\tilde{A}_i = c * A_i + s * A_{i+1}$ e $\tilde{A}_{i+1} = c * A_{i+1} - s * A_i$. Novamente queremos um algoritmo, descrito sumária mas claramente, e a complexidade esperada. Dicas e curiosidades:

      i. Assumindo que $\alpha << 1$, o caso em que estamos interessados, a densidade esperada das novas linhas é $2 * \alpha$. Por quê?

      ii. O algoritmo para a segunda questão deve ser algo melhor que a mera repetição do algoritmo da primeira questão.

      iii. Esta transformação linear, tomando as constantes c e s como o coseno e o seno de um ângulo θ, é uma "rotação de Givens", a ser estudada no capítulo 3.

5. Dados u e w $n \times 1$, A e B $n \times n$, e $k \in N^*$, prove que

   (a) $tr(A + B) = tr(A) + tr(B)$.

   (b) $tr(AB) = tr(BA)$.

   (c) $\mathrm{tr}(uw') = w'u$.

   (d) $\mathrm{tr}(Auw') = \mathrm{tr}(uw'A) = w'Au$.

   (e) $(uw'A)^k = (\mathrm{tr}(uw'A))^{k-1}\ (uw'A)$.

   (f) $(Auw')^k = (\mathrm{tr}(Auw'))^{k-1}\ (Auw')$.

Capítulo 2

FATORAÇÃO LU

Solução de Sistemas Lineares

2.1 Permutações e Operações Elementares

Uma **matriz de permutação** é uma matriz obtida pela permutação de linhas ou colunas na matriz identidade. Realizar, na matriz identidade, uma dada permutação de linhas, nos fornece a correspondente matriz de permutação de linhas; Analogamente, uma permutação de colunas da identidade fornece a correspondente matriz de permutação de colunas.

Dada uma (matriz de) permutação de linhas, P e uma (matriz de) permutação de colunas, Q, o correspondente vetor de índices de linha (coluna) permutados são

$$p = (P \begin{bmatrix} 1 \\ 2 \\ \vdots \\ m \end{bmatrix})'$$

$$q = \begin{bmatrix} 1 & 2 & \dots & n \end{bmatrix} Q$$

Lema 2.1 *Realizar uma permutação de linhas (de colunas) numa matriz qualquer A, de modo a obter a matriz permutada $\tilde{A}$, equivale a multiplicá-la, à esquerda (à direita), pela correspondente matriz de permutação de linhas (de colunas). Ademais, se p (q) é o correspondente vetor de índices de linha (de coluna) permutados,*

$$\tilde{A}_i^j = (PA)_i^j = A_{p(i)}^j$$

$$\tilde{A}_i^j = (AQ)_i^j = A_i^{q(j)} \ .$$

Exemplo 1: Dadas as matrizes

$$A = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix} \; , \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \; , \quad Q = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \; ,$$

$$p = q = \begin{bmatrix} 3 & 1 & 2 \end{bmatrix} \; , \quad PA = \begin{bmatrix} 31 & 32 & 33 \\ 11 & 12 & 13 \\ 21 & 22 & 23 \end{bmatrix} \; , \quad AQ = \begin{bmatrix} 13 & 11 & 12 \\ 23 & 21 & 22 \\ 33 & 31 & 32 \end{bmatrix} \; .$$

Uma matriz quadrada, A, é **simétrica** sse for igual a transposta, isto é, sse $A = A'$. Uma **permutação simétrica** de uma matriz quadrada A é uma permutação da forma $\tilde{A} = PAP'$, onde P é uma matriz de permutação. Uma matriz quadrada, A, é **ortogonal** sse sua inversa for igual a sua transposta, isto é, sse $A^{-1} = A'$.

Lema 2.2 *(a) Matrizes de permutação são ortogonais. (b) Uma permutação simétrica de uma matriz simétrica é ainda uma matriz simétrica.*

Estudaremos a seguir alguns algoritmos para solução do sistema $Ax = b$. Tais algoritmos são denominados diretos pois, a menos do erro de arredondamento, fornecem diretamente a solução do sistema. A essência destes algoritmos são transformações sobre o sistema que deixam inalteradas sua solução.

São **operações elementares** sobre uma matriz, $n \times m$, qualquer:

1. Multiplicar uma linha por um escalar não nulo.
2. Adicionar, a uma linha, uma outra linha da matriz.
3. Subtrair de uma linha, uma outra linha da matriz multiplicada por um escalar não nulo.

Uma operação elementar aplicada à matriz identidade I, $n \times n$, produz a **matriz elementar** correspondente, E.

Lema 2.3 *Realizar uma operação elementar sobre uma matriz qualquer, A, equivale a multiplicá-la, à esquerda, pela matriz elementar correspondente.*

Exemplo 3.

$$A = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix} \; , \quad E = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -31/11 & 0 & 1 \end{bmatrix} \; ,$$

$$EA = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 0 & 32 - 12 * 31/11 & 33 - 13 * 31/11 \end{bmatrix} \; .$$

Lema 2.4 *Toda matriz elementar é inversível. Ademais, $Ax = b \Leftrightarrow EAx = Eb$.*

A matriz aumentada correspondente ao sistema $Ax = b$, A $n \times n$ é a matriz $[A\ b]$, $n \times n+1$. Obviamente a matriz aumentada do sistema $EAx = Eb$ é a matriz $E[A\ b]$.

Uma matriz quadrada A é dita **triangular** superior se todos os elementos abaixo da diagonal principal são nulos, e é dita triangular estritamente superior se também os elementos da diagonal são nulos. Analogamente, definimos matriz triangular inferior e estritamente inferior. Assim, A é

- Triangular Superior $\Leftrightarrow (i > j \Rightarrow A_i^j = 0)$.
- Triangular Estritamente Superior $\Leftrightarrow (i \geq j \Rightarrow A_i^j = 0)$.
- Triangular Inferior $\Leftrightarrow (i < j \Rightarrow A_i^j = 0)$.
- Triangular Estritamente Inferior $\Leftrightarrow (i \leq j \Rightarrow A_i^j = 0)$.

Os métodos diretos que estudaremos funcionam por ser fácil resolver um sistema Ux = b se U é triangular superior, isto é Em um tal sistema podemos calcular por substituição, nesta ordem, as componentes $x_n,\ x_{n-1}, \ldots, x_1$, pois

$$\begin{aligned} x_n &= b_n / U_n^n \\ x_{n-k} &= (b_{n-k} - \sum_{j=n-k+1}^{n} U_{n-k}^j x_j) / U_{n-k}^{n-k} \end{aligned}$$

Tal método funciona se $U_j^j \neq 0,\ \forall j \in N$. Como $det(U) = \prod_{j=1}^n U_j^j$, esta condição equivale a termos um sistema bem determinado. Estudaremos a seguir como levar, através de operações elementares, um sistema qualquer à forma triangular.

2.2 Método de Gauss

O método de Gauss leva o sistema à forma triangular através de operações elementares do tipo 3. Tentaremos fazê-lo usando, nesta ordem, a primeira linha para anular os elementos abaixo da diagonal na primeira coluna, a segunda linha para anular os elementos abaixo da diagonal na segunda coluna, etc... O exemplo 2 ilustra o processo. Indicamos também os fatores pelos quais multiplicamos cada linha antes de subtraí-la de outra.

Exemplo 4.

$$\left[{}^0A\ {}^0b\right] = \begin{array}{cc} \begin{bmatrix} 2 & 1 & 3 & 1 \\ 2 & 3 & 6 & 2 \\ 4 & 4 & 6 & 6 \end{bmatrix} & \begin{array}{c} \\ 1 \\ 2 \end{array} \end{array} \rightarrow \left[{}^1A\ {}^1b\right] = \begin{array}{cc} \begin{bmatrix} 2 & 1 & 3 & 1 \\ 0 & 2 & 3 & 1 \\ 0 & 2 & 0 & 4 \end{bmatrix} & \begin{array}{c} \\ \\ 1 \end{array} \end{array} \rightarrow$$

$$\left[{}^2A\ {}^2b\right] = \begin{bmatrix} 2 & 1 & 3 & 1 \\ 0 & 2 & 3 & 1 \\ 0 & 0 & -3 & 3 \end{bmatrix} \quad \rightarrow \quad x = \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}$$

Em cada etapa denominaremos a linha que está sendo multiplicada e subtraída às demais de **linha pivô**, seu elemento diagonal de **elemento pivô** e os fatores de multiplicação de **multiplicadores**. Posteriormente faremos uso dos multiplicadores utilizados no processo e, portanto, devemos armazená-los. Com este fim, definimos as matrizes, $n \times n$, ${}^1M$, ${}^2M$, $\ldots {}^{n-1}M = M$, onde: Se $j \leq k$ e $i > j$, então ${}^kM_i^j$ é o multiplicador utilizado na k-ésima etapa para anular o elemento na i-ésima linha e j-ésima coluna; Caso contrario, ${}^kM_i^j = 0$.

Observe que os elementos não nulos de ${}^kM$ correspondem a elementos nulos em ${}^kA$, e vice-versa. Podemos pois poupar memória, guardando uma única matriz, ${}^kA + {}^kM$. Assim, no Exemplo 4, $\left[{}^kA +^k M \mid {}^kb\right]$, $k = 0, 1, 2$, pode ser representado como:

$$\begin{bmatrix} 2 & 1 & 3 & 1 \\ 2 & 3 & 6 & 2 \\ 4 & 4 & 6 & 6 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 3 & 1 \\ \mathit{1} & 2 & 3 & 1 \\ \mathit{2} & 2 & 0 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & 1 & 3 & 1 \\ \mathit{1} & 2 & 3 & 1 \\ \mathit{2} & \mathit{1} & -3 & 3 \end{bmatrix}$$

2.3 Pivoteamento

Consideremos a hipótese do surgimento de um zero na posição do pivô da etapa seguinte do processo de triangularização, isto é, em ${}^{k-1}A$ anula-se o elemento ${}^{k-1}A_k^k$. Se na coluna ${}^{k-1}A^k$ houver algum elemento não nulo na linha l, $l > k$, podemos permutar as linhas k e l e continuar o processo de triangularização. Uma permutação de linhas, para trocar o elemento pivô a ser utilizado, denomina-se um **pivoteamento**. A cada etapa queremos lembrar-nos de quais os pivoteamentos realizados durante o processo. Para tanto, guardamos os vetores de índices de linha permutados da permutação corrente em relação ao sistema original. Estes são os vetores de permutação, ${}^1p$, ${}^2p$, $\ldots {}^{n-1}p = p$. No que tange ao armazenamento dos multiplicadores, devemos convencionar se estes serão ou não permutados, junto com as respectivas linhas de $A + M$, nas operações de pivoteamento. Adotaremos, por hora, a convenção de SIM, permutá-los, junto com os pivoteamentos.

O exemplo 5 ilustra o processo para uma matriz de dimensão $k = 4$, apresentando a matriz $\left[{}^kA +^k M\right]$ juntamente com o vetor de permutação, ${}^kp$ para cada etapa da triangularização:

$$\begin{bmatrix} 2 & 1 & 9 & -1 \\ 1 & 3 & 7 & 7 \\ 2 & 8 & 4 & 2 \\ 3 & 9 & 6 & 6 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \rightarrow \begin{bmatrix} 3 & 9 & 6 & 6 \\ \mathit{1/3} & 0 & 5 & 5 \\ \mathit{2/3} & 2 & 0 & -2 \\ \mathit{2/3} & -5 & 5 & -5 \end{bmatrix} \begin{matrix} 4 \\ 2 \\ 3 \\ 1 \end{matrix} \rightarrow$$

$$\left[\begin{array}{cccc|c} 3 & 9 & 6 & 6 & 4 \\ 2/3 & -5 & 5 & -5 & 1 \\ 2/3 & -2/5 & 2 & -4 & 3 \\ 1/3 & 0 & 5 & 5 & 2 \end{array}\right] \rightarrow \left[\begin{array}{cccc|c} 3 & 9 & 6 & 6 & 4 \\ 2/3 & -5 & 5 & -5 & 1 \\ 1/3 & 0 & 5 & 5 & 2 \\ 2/3 & -2/5 & 2/5 & -6 & 3 \end{array}\right]$$

Observação 2.1 Note que se o sistema é bem determinado, um pivoteamento que permite o prosseguimento do processo de triangularização é sempre possível: Se na k-ésima etapa, para $l = k \ldots n$, ${}^{k-1}A_l^k = 0$, isto é o elemento na posição pivô e todos os elementos abaixo dele na coluna ${}^{k-1}A^k$ se anulam, então as linhas ${}^{k-1}A_l$, $l = k \ldots n$, são linearmente dependentes e $det(\,{}^{k-1}A) = 0 \Rightarrow det(\,{}^{0}A) = 0$, o que contraria a hipótese do sistema ser bem determinado.

Observação 2.2 Assuma sabermos de antemão um vetor de permutação, ${}^{n-1}p = p$, viável no processo de triangularização de um dado sistema $[{}^0A\ {}^0b]$. Neste caso, poderíamos permutar as linhas do sistema original como indicado no vetor de permutação, obtendo um novo sistema $P\,[{}^0A\ {}^0b]$, cujas equações são as mesmas que as do sistema original, a menos da ordem em que estão escritas. Poderíamos então triangularizar este sistema sem necessidade de nenhum pivoteamento, obtendo ao final o mesmo sistema equivalente, $[{}^{n-1}A\ {}^{n-1}b]$.

Note que a triangularização da matriz dos coeficientes de um sistema é completamente independente do vetor dos termos independentes. Suponha termos triangularizado o sistema $[{}^0A\ {}^0b]$, isto é, que temos um sistema equivalente e triangular $[{}^{n-1}A = U\ {}^{n-1}b]$, tendo sido preservados os multiplicadores e as permutações utilizadas, M e p. Se for necessário resolver um segundo sistema $[{}^0A\ {}^0b]$, que difere do primeiro apenas pelo vetor dos termos independentes, não será necessário retriangularizar a matriz A; Bastará efetuar no novo vetor de termos independentes, ${}^0c$, as operações indicadas pelo vetor de permutação p e pela matriz de multiplicadores M.

Uma política, que mais tarde demonstraremos útil, é realizarmos pivoteamentos para colocar como elemento pivô, em cada etapa, o elemento de máximo módulo. Esta estratégia, o **pivoteamento parcial**, garante termos todos os multiplicadores $\mid M_i^j \mid \leq 1$. O pivoteamento parcial é de fundamental importância para a estabilidade numérica do processo, controlando a propagação de erros de arrendondamento.

Observação 2.3 Para realizar uma operação de pivoteamento não é necessário efetivamente permutar linhas da matriz A: Basta, na k-ésima etapa da triangularização utilizar o índice ${}^{k-1}p(i)$ ao invés do índice de linha i.

Observação 2.4 Uma maneira alternativa de guardar pivoteamentos realizados ao longo do processo de triangularização é o **vetor de pivoteamentos**, t, onde $t(j) = i$ significa que ao eliminarmos a coluna j permutamos para a posição de pivo (linha j) o elemento na linha i. O vetor de pivoteamentos do exemplo anterior é $\left[\begin{array}{cccc} 4 & 1 & 2 & 3 \end{array}\right]'$. Com o vetor de pivoteamentos adotaremos a convenção de NÃO permutar os multiplicadores na coluna ${}^jM$, nos pivoteamentos $k > j$.

2.4 Lema da Fatoração

Lema 2.5 (da Fatoração) *Seja ${}^{0}A$ uma matriz inversível triangularizável pelo método de Gauss sem nenhum pivoteamento e sejam, conforme a nomenclatura adotada, $A = {}^{o}A$, ${}^{1}A$, ... ${}^{n-1}A$, e ${}^{1}M$, ${}^{2}M$, ... ${}^{n-1}M = M$, respectivamente, a k-transformada da matriz A e a k-ésima matriz dos multiplicadores. Fazendo $U = {}^{n-1}A$ e $L = M + I$, temos que L e U são matrizes triangulares, respectivamente inferior e superior, e que $A = LU$.*

Demonstração.

Verifiquemos inicialmente que a transformação linear que leva ${}^{k-1}A$ em ${}^{k}A$ é dada por ${}^{k}T$, i.e. ${}^{k}A = {}^{k}T\ {}^{k-1}A$, onde

$$
{}^{k}T = \begin{bmatrix} 1 & & & & 0 \\ 0 & \ddots & & & \\ & & 1 & & \vdots \\ \vdots & & -M^{k}_{k+1} & & \\ & & \vdots & \ddots & 0 \\ 0 & & -M^{k}_{n} & & 1 \end{bmatrix},
$$

de modo que $U = {}^{n-1}A = {}^{n-1}T\ {}^{n-2}T \ldots {}^{2}T\ {}^{1}T\ A = TA$. Observando ainda que ${}^{k}T$ é inversível e que

$$
{}^{k}T^{-1} = {}^{k}L = \begin{bmatrix} 1 & & & & 0 \\ 0 & \ddots & & & \\ & & 1 & & \vdots \\ \vdots & & +M^{k}_{k+1} & & \\ & & \vdots & \ddots & 0 \\ 0 & & +M^{k}_{n} & & 1 \end{bmatrix},
$$

Ademais, temos que $T^{-1} = {}^{1}T^{-1}\ {}^{2}T^{-1} \ldots {}^{n-1}T^{-1} = {}^{1}L \ldots {}^{n-1}L$ e é fácil verificar que ${}^{1}L\ {}^{2}L \ldots {}^{n-1}L = L$, donde $A = LU$. Q.E.D.

Teorema 2.1 (LU) *Seja A uma matriz inversível $n \times n$. Então existe uma (matriz de) permutação de linhas P de modo que $\tilde{A} = PA$ é triangularizável pelo método de Gauss e $\tilde{A} = LU$.*

Demonstração.

O teorema segue trivialmente do lema da fatoração e das observações 2.1 e 2.2.

A solução de um sistema linear $Ax = b$ pode ser expressa diretamente em termos da fatoração LU da matriz dos coeficientes, i.e., se $\tilde{A} = PA = LU$, então $P'LUx = b$, e $x = U^{-1}L^{-1}Pb$. Assim, em muitas aplicações, o conhecimento explícito de A^{-1} pode ser substituído com vantagem pelo conhecimento da fatoração da matriz.

Observação 2.5 Considere os dois algoritmos seguintes, o de substituição e o de multilpicação pela inversa, para solução do sistema $Ux = b$: O primeiro algoritmo acessa a matriz U por linha, enquanto o segundo acessa a matriz U por coluna.

```
x = b ;
x(n) = x(n) / U(n,n) ;
for i = n-1:-1:1
  x(i) = ( x(i) - U(i,i+1:n)*x(i+1:n) ) / U(i,i) ;
end
```

```
x = b ;
for j = n:-1:2
  x(j) = x(j) / U(j,j) ;
  x(1:j-1) = x(1:j-1) - x(j)*U(1:j-1,j) ;
end
x(1) = x(1) / U(1,1) ;
```

2.5 O Método de Doolittle

Usando o Teorema LU podemos determinar L e U diretamente da equação de decomposição, $LU = A$, tomando, nesta ordem, para $i = 1 \ldots n$, as equações das componentes de A na i-ésima linha e na i-ésima coluna, isto é,

$$\text{para } i = 1 \ldots n \left\{ \begin{array}{l} L_i U = A_i \\ L U^i = A^i \end{array} \right.$$

temos

$$\text{para } i = 1 \ldots n \left\{ \begin{array}{ll} \text{para } j = i \ldots n & U_i^j = A_i^j - \sum_{k=1}^{i-1} M_i^k U_k^j \\ \text{para } j = i+1 \ldots n & M_j^i = (A_j^i - \sum_{k=1}^{i-1} M_j^k U_k^i)/U_i^i \end{array} \right.$$

Note que só escrevemos as equações para os termos incógnitos, isto é, ou acima da diagonal em U, e abaixo da diagonal em L. Também as somatórias foram interrompidas quando os termos remanescentes são todos nulos.

O cálculo do elemento $(M+U)_i^j$, envolve, na ordem prescrita, apenas elementos já calculados de $(M+U)$. Ademais, o cálculo de $(M+U)_i^j$ é a última ocasião em que se necessita os elementos A_i^j; portanto podemos armazenar $(M+U)_i^j$ no lugar de A_i^j. Estas são as matrizes $A = {}^0D, \ldots, {}^nD = M + U$.

Exemplo 6 - Usando o método de Doolittle para triangularizar a matriz do Exemplo 4, onde não há necessidade de pivoteamento, temos

$$
{}^{0}D = A = \begin{bmatrix} 2 & 1 & 3 \\ 2 & 3 & 6 \\ 4 & 4 & 6 \end{bmatrix} \quad \rightarrow
$$

$$
\begin{bmatrix} 2 & 1 & 3 \\ 2 & 3 & 6 \\ 4 & 4 & 6 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} 2 & 1 & 3 \\ 1 & 3 & 6 \\ 2 & 4 & 6 \end{bmatrix} = {}^{1}D \quad \rightarrow
$$

$$
\begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 3 \\ 2 & 4 & 6 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 3 \\ 2 & 1 & 6 \end{bmatrix} = {}^{2}D \quad \rightarrow
$$

$$
\begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 3 \\ 2 & 1 & -3 \end{bmatrix} = {}^{3}D = M + U
$$

Para realizar os pivoteamentos necessários à passagem de ${}^{k}D$ a ${}^{k+1}D$ é necessário examinar os possíveis elementos pivôs nas linhas $i = k \ldots n$, isto é, os elementos em ${}^{k}A^{k}$. Para tanto, basta calcular, para $i = k \ldots n$,

$$
{}^{k}v_i = {}^{0}A_i^k - \sum_{l=1}^{k-1} M_i^l U_l^k \ .
$$

Exemplo 7 - Triangularizando pelo método de Doolittle a matriz, A, com pivoteamento parcial, temos os ${}^{k}p$, ${}^{k}D$, ${}^{k}v$, para $k = 0 \ldots n$, como segue:

$$
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{bmatrix} 2 & 1 & 9 & -1 \\ 1 & 3 & 7 & 7 \\ 2 & 8 & 4 & 2 \\ 3 & 9 & 6 & 6 \end{bmatrix} \begin{array}{c} 2 \\ 1 \\ 2 \\ 3 \end{array} \quad \rightarrow \quad \begin{array}{c} 4 \\ 2 \\ 3 \\ 1 \end{array} \begin{bmatrix} 3 & 9 & 6 & 6 \\ 1/3 & 3 & 7 & 7 \\ 2/3 & 8 & 4 & 2 \\ 2/3 & 1 & 9 & -1 \end{bmatrix} \begin{array}{c} * \\ 0 \\ 2 \\ -5 \end{array} \quad \rightarrow
$$

$$
\begin{array}{c} 4 \\ 1 \\ 3 \\ 2 \end{array} \begin{bmatrix} 3 & 9 & 6 & 6 \\ 2/3 & -5 & 5 & -5 \\ 2/3 & -2/5 & 4 & 2 \\ 1/3 & 0 & 7 & 7 \end{bmatrix} \begin{array}{c} * \\ \\ 2 \\ 5 \end{array} \quad \rightarrow \quad \begin{array}{c} 4 \\ 1 \\ 2 \\ 3 \end{array} \begin{bmatrix} 3 & 9 & 6 & 6 \\ 2/3 & -5 & 5 & -5 \\ 1/3 & 0 & 5 & 5 \\ 2/3 & -2/5 & 2/5 & 2 \end{bmatrix} \quad \rightarrow
$$

$$
\begin{array}{c} 4 \\ 1 \\ 2 \\ 3 \end{array} \begin{bmatrix} 3 & 9 & 6 & 6 \\ 2/3 & -5 & 5 & -5 \\ 1/3 & 0 & 5 & 5 \\ 2/3 & -2/5 & 2/5 & -6 \end{bmatrix}
$$

2.6 Complexidade

Contemos agora o número de operações aritméticas necessárias à triangularização de uma matriz de coeficientes e à solução de um sistema linear. Na fase de triangularização da matriz dos coeficientes, vemos que o cálculo de ${}^{k}A$ a partir de ${}^{k-1}A$ requer $(n-k)$ divisões e $(n-k)^2$ somas e produtos. No total são portanto requeridos

$$\sum_{k=1}^{n-1}(n-k) = n(n-1)/2 \qquad \text{divisões, e}$$

$$\sum_{k=1}^{n-1}(n-k)^2 = n(n^2-1)/3 + n(n-1)/2 \qquad \text{produtos e subtrações.}$$

isto é, são necessários da ordem de $n^3/3 \ + O(n^2)$ produtos e subtrações, e $n^2/2 \ + O(n)$ divisões.

Analogamente, dada a matriz dos multiplicadores e o vetor das permutações, M e p, o tratamento de um vetor de termos independentes requer $n(n-1)/2$ produtos e subtrações. Finalmente, a solução do sistema linear triangularizado requer n divisões e $n(n-1)/2$ produtos e subtrações.

Exercícios

1. Comente como cada forma de representação da matriz A, por linhas ou por colunas, favorece o uso de um dos algoritmos apresentados na observação 2.5. Escreva algoritmos similares para a solução do sistema $Lx = b$.

2. Programe e implemente, em linguagem C, C++, ou FORTRAN-90, funções para:

   (a) Fatoração LU com pivoteamento parcial.

   (b) Solução dos sistemas $Lx = b$ e $Ux = b$,

   (c) Um argumento adicional deve indicar a forma de representação das matrizes, densa, estática por linha, ou estática por coluna.

   (d) No caso da representação densa, um segundo argumento deve indicar o uso do vetor de permutações ou pivoteamentos. No caso das representações estáticas, escreva o programa como lhe parecer mais conveniente.

3. Mostre que calcular explicitamente a inversa de A implica saber resolver n sistemas lineares, $Ax = I^j$. Dada a fatoração $A = LU$, qual o custo de computar explicitamente a inversa A^{-1}?

Capítulo 3

RESUMO DE TEORIA DOS GRAFOS

3.1 Conceitos Básicos

Um **grafo** é um par ordenado $G = (V_G, \Gamma_G)$ onde o primeiro elemento é um conjunto finito, o conjunto de **vértices**, e o segundo elemento é uma função $\Gamma_G : V_G \mapsto P(V_G)$,no conjunto das partes de V_G, a **função de filiação**. Tomaremos V_G um conjunto indexado, $V_G = \{v_1, v_2, \ldots v_n\}$ ou então tomaremos V_G como sendo o próprio conjunto dos n primeiros inteiros positivos, $N = \{1, 2, \ldots n\}$. Para não sobrecarregar a notação escreveremos, quando não houver ambigüidade, (V_G, Γ_G) como (V, Γ).

É comum representarmos um grafo por conjunto de pontos (os vértices) e um conjunto de **arestas** (setas) que vão de cada vértice para os seus filhos, isto é, de cada $v \in V$ para os elementos em $\Gamma(v)$. Podemos definir o grafo através dos seus vértices e de suas arestas, $G = (V_G, A_G)$, onde cada aresta é um par ordenado de vértices e $(i, j) \in A_G \Leftrightarrow j \in \Gamma_G(i)$. Uma terceira maneira de definir um grafo é pelo par $G = (V_G, B_G)$ onde B_G é a **matriz de adjacência**, a matriz Booleana tal que $B_i^j = 1 \Leftrightarrow j \in \Gamma(i)$.

Exemplo 1:

Considere o grafo de vértices $N = \{1, 2, 3, 4, 5, 6\}$ e função de filiação $\Gamma(1) = \emptyset$, $\Gamma(2) = \{2\}$, $\Gamma(3) = \emptyset$, $\Gamma(4) = \{3, 5, 6\}$, $\Gamma(5) = \{3, 4, 5\}$ e $\Gamma(6) = \{5\}$. Suas arestas e matriz de adjacência são, $A_G = \{(2, 2), (4, 3), (4, 5), (4, 6), (5, 3), (5, 4), (5, 5), (6, 5)\}$, e

$$B_G = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{array}{cccc} 1 & 3 & \leftarrow & 5 \hookleftarrow \\ & \uparrow & \nearrow\!\swarrow & \uparrow \\ \hookrightarrow 2 & 4 & \rightarrow & 6 \end{array}$$

Dado um subconjunto de vértices $W \subset V$, definimos $\Gamma(W) \equiv \cup_{w \in W} \Gamma(w)$. Definimos também

$\Gamma^0(i) = \{i\}$, $\Gamma^1(i) \equiv \Gamma(i)$ e, para $k > 1$, $\Gamma^k(i) \equiv \Gamma(\Gamma^{k-1}(i))$. Estes são, para $k = 1, 2, 3, \ldots$ os filhos, netos, bisnetos, etc. do vértice i. Finalmente definimos os **descendentes** de i por $\bar{\Gamma}(i) = \cup_{k=0}^{\infty} \Gamma^k(i)$, e o **fecho transitivo** $\bar{G} = (V_G, \bar{\Gamma}_G)$.

Exemplo 2:

Damos a matriz de adjacência de um grafo, e do respectivo fecho transitivo:

$$B = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad , \quad \bar{B} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dado o grafo $G = (V, \Gamma)$ definimos a **função de paternidade** $i \in \Gamma^{-1}(j) \Leftrightarrow j \in \Gamma(i)$. Definimos também seu **grafo inverso**, $G^{-1} = (V, \Gamma^{-1})$.

Lema 3.1 *Dado um grafo definido por sua matriz de adjacência, $G = (N, B)$, seu grafo inverso é $G^{-1} = (N, B')$.*

Um **sub-grafo** de $G = (V, A)$ é um grafo $G' = (V', A')$, onde $V' \subset V$ e A' é um subconjunto de arestas de A que tem ambos os vértices em V'. O sub-grafo induzido por um subconjunto de vértices V' é $G = (V', A')$ onde A' é máximo, e o sub-grafo induzido por um subconjunto de arestas A' é $G = (V', A')$ onde V' é mínimo.

Uma aresta que parte e chega no mesmo vértice é dita um **loop**. Duas arestas, (i, j) e (k, h), são ditas **contíguas** se a primeira chega no vértice de que parte a segunda, isto é se $j = k$. Um **passeio** é uma seqüência não vazia e finita de arestas contíguas. Um **circuito** é um passeio que parte e chega no mesmo vértice, isto é, o primeiro vértice da primeira aresta é o segundo vértice da última aresta. Uma **trilha** é um passeio onde não se repete nenhuma aresta e um **caminho** é uma trilha na qual não há (subseqüências que sejam) circuitos. Uma trilha na qual o único circuito é toda a trilha é uma **ciclo**.

Notemos que um passeio $C = (w_0, w_1), (w_1, w_2), \ldots (w_{k-1}, w_k))$, é igualmente bem determinado pela seqüência de seus vértices $C = (w_0, w_1, w_2, \ldots w_k)$.

Teríamos assim, no Exemplo 1, exemplos de:

- loop: $((2, 2))$.
- passeio: $((4, 5), (5, 5), (5, 5), (5, 4), (4, 5)$.
- circuito: $(5, 5, 4, 6, 5, 5)$.
- trilha: $(5, 4, 6, 5, 5, 3)$.
- caminho: $(6, 5, 4, 3)$.

- ciclo: $(5,4,6,5) ou (5,5)$.

Um grafo é **acíclico** se não contém ciclos. Uma **árvore** de **raiz** v é um grafo acíclico, $H = (V_H, \Gamma_H)$, $v \in V_H$, onde todos os vértices têm no máximo um pai e apenas a raiz não tem pai. Os vértices sem filhos de uma árvore dizem-se folhas da árvore. Uma coleção de árvores é denominada **floresta**. Uma floresta cobre um grafo G sse é um subgrafo de G que contém todos os seus vértices. Seguem exemplos de florestas que cobrem o grafo do Exemplo 1, estando assinaladas as raízes.

$$
\begin{array}{cccc}
\mathbf{1} & 3 & & 5 \\
 & \uparrow & \swarrow & \uparrow \\
\mathbf{2} & 4 & & \mathbf{6}
\end{array}
\quad , \quad
\begin{array}{cccc}
\mathbf{1} & 3 & & 5 \\
 & \uparrow & \nearrow & \\
\mathbf{2} & \mathbf{4} & \rightarrow & 6
\end{array}
$$

Lema 3.2 *Dada uma árvore, $H = (V, \Gamma)$, de raiz v, e um vértice $w \in V$, existe um único passeio que parte de v e termina em w, a saber, $(v, \Gamma^{-k}(w), \ldots \Gamma^{-2}(w), \Gamma^{-1}(w), w)$. Ademais, este passeio é um caminho.*

3.2 Relações de Ordem

Uma **ordem** num conjunto S é uma relação, $\leq$, tal que $\forall a, b, c \in S$, temos que

1. $a \leq a$.
2. $a \leq b \ \wedge \ b \leq c \Rightarrow a \leq c$.

i.e., uma relação **reflexiva** e **transitiva**. Alternativamente denotaremos $a \leq b$ por $b \geq a$.

Sendo $=$ a relação identidade e $\not\leq$ a negação da relação de ordem, dizemos que a ordem é

- **total**, sse $a \not\leq b \Rightarrow b \leq a$.
- **parcial**, sse $(a \leq b \ \wedge \ b \leq a) \Rightarrow a = b$.
- **boa**, se é parcial e total.

Lema 3.3 *Dado um grafo, $G = (N, \Gamma)$, a função de descendência, $\bar{\Gamma}$, define uma ordem em seus vértices, a ordem natural, $\leq$, definida por por $j \geq i \Leftrightarrow j \in \bar{\Gamma}(i)$. Note, porém, que a ordem natural não é, em geral, nem parcial nem total.*

Exemplo 3:

Nos grafos seguintes, de matrizes de adjacência B_1, B_2, B_3 e B_4,

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

temos ordens naturais

- em G_1: Nem parcial, pois $1 \leq 2 \leq 1$, nem total, pois $1 \not\leq 3 \not\leq 1$.
- em G_2: Total, mas não parcial, pois $1 \leq 4 \leq 1$.
- em G_3: Parcial, mas não total, pois $2 \not\leq 3 \not\leq 2$.
- em G_4: Boa.

Uma **equivalência** num conjunto S é uma relação, $\sim$, tal que $\forall a, b, c \in S$:

1. $a \sim a$.
2. $a \sim b \ \wedge \ b \sim c \Rightarrow a \sim c$.
3. $a \sim b \Rightarrow b \sim a$.

i.e., uma relação reflexiva, transitiva e **simétrica**.

A **classe de equivalência** de um elemento qualquer, $a \in S$, é o sub-conjunto de S: $[a] = \{x \in S \mid x \sim a\}$.

Uma **partição** de um conjunto S é uma coleção P de sub-conjuntos não vazios de S tal que:

1. $\forall X, Y \in P,\ X \neq Y \Rightarrow X \cap Y = \emptyset$.
2. $\cup_{X \in P} X = S$.

i.e., uma coleção de conjuntos disjuntos que reunidos é igual a S.

Lema 3.4 *Dado S, um conjunto munido de uma ordem, $\forall a, b \in S$, $a \sim b \Leftrightarrow a \leq b \wedge b \leq a$, define uma relação de equivalência. Esta é a equivalência induzida pela ordem. Dado S um conjunto munido de uma equivalência, o conjunto das classes de equivalência em S é uma partição de S.*

As **componentes fortemente conexas**, CFC, de um grafo G são as classes de equivalência induzida pela ordem natural nos vértices de G. G é dito fortemente conexo sse possui uma única CFC.

Exemplo 4:

As CFC dos grafos do Exemplo 3 são:

- em G_1: $\tilde{V} = \{\{1,2\},\{3\},\{4\}\}$.
- em G_2: $\tilde{V} = \{\{1,2\},\{3,4\}\}$.
- em G_3 e G_4: $\tilde{V} = \{\{1\},\{2\},\{3\},\{4\}\}$.

O **grafo reduzido**, de um grafo $G = (V, \Gamma)$, é o grafo $G = (\tilde{V}, \tilde{\Gamma})$, que tem por vértices as CFCs de G: $\tilde{V} = \{V_1, V_2, \ldots V_k\}$, e a função de filiação, $\tilde{\Gamma}$, definida por

$$V_s \in \tilde{\Gamma}(V_r) \Leftrightarrow \exists i \in V_r,\ j \in V_s \mid j \in \Gamma(i),\ (V_r \neq V_s)\ .$$

Exemplo 5:

As matrizes de adjacência dos grafos reduzidos dos grafos do Exemplo 3, com os vértices correspondendo as CFCs na ordem em que aparecem no exemplo anterior, são $\tilde{B}_1$, $\tilde{B}_2$, $\tilde{B}_3$ e $\tilde{B}_4$:

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},\quad \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},\quad \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},\quad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Dado um conjunto S e $\leq$ uma ordem em S, a **ordem reduzida** é a relação de ordem no conjunto das classes (da equivalência induzida pela ordem $\leq$), $\tilde{S}$, definida por

$$\forall X, Y \in \tilde{S}\ ,\quad X \leq Y \Leftrightarrow \exists x \in X, \wedge \exists y \in Y \mid x \leq y\ .$$

Lema 3.5 *A ordem reduzida (nas CFCs) de um grafo $G = (V, \Gamma)$, é a ordem natural do grafo reduzido $\tilde{G} = (\tilde{V}, \tilde{\Gamma})$, i.e., se $\tilde{V} = \{V_1, \ldots V_k\}$,*

$$\begin{aligned} V_r \leq V_s &\Leftrightarrow V_s \in \bar{\tilde{\Gamma}}(V_r) \\ &\Leftrightarrow \exists i \in V_r,\ j \in V_s \mid j \in \bar{\Gamma}(i) \\ &\Leftrightarrow \exists i \in V_r,\ j \in V_s \mid j \geq i\ . \end{aligned}$$

Ademais,

1. *A ordem natural de G é parcial sse, a menos de loops, G é acíclico.*
2. *Grafos reduzidos são acíclicos e ordens reduzidas são parciais.*

Teorema 3.1 (Hoffman) : *Um grafo $G = (V, \Gamma)$ é fortemente conexo sse dado qualquer subconjunto próprio dos vértices $W \subset V$, $W \neq V$, houver uma aresta de um vértice em W para um vértice fora de W.*

Demonstração:

Se houver W um subconjunto próprio de V tal que $\Gamma(W) = \emptyset$, então não há caminho de nenhum vértice $w \in W$ para nenhum vértice $v \in V$, e G não é fortemente conexo. Presupondo a condição $\nexists W \neq V \mid \Gamma(W) = \emptyset$, provemos que existe um caminho do vértice w ao vértice v, $\forall w, v \in V$. Tomemos inicialmente $W_0 = \{w\}$. A condição garante a existência de um caminho de comprimento (número de arestas) 1 de w a algum vértice $x_1 \neq w$. Se $x_1 = v$ a prova está completa. Caso contrário tomemos $W_1 = \{w, x_1\}$. A condição garante a existência de um caminho c_2 de comprimento $|c_2| \leq 2$ de w para algum vértice $x_2 \neq w, x_1$. Repetindo o argumento até obtermos $x_k = v$, $k < n$, concluímos a demonstração, QED.

Dado um conjunto S, munido de uma ordem $\leq$, a ordem de S, uma boa ordem, $<=$, em S é dita uma **ordem coerente** (com $\leq$) sse:

1. $\forall a, b \in S,\ a \leq b\ \wedge\ b \not\leq a \Rightarrow a <= b$.
2. $\forall a, b, c \in S,\ a <= b <= c\ \wedge\ a \sim c \Rightarrow a \sim b \sim c$.

O primeiro critério de coerência determina que o reordenamento se subordine à ordem parcial do grafo reduzido; O segundo critério determina que se **discriminem** (não se misturem) vértices em CFCs incomparáveis porém distintas.

Uma boa ordem, ou reordenamento, no conjunto $N = \{1, 2, \ldots n\}$, $q_1 <= q_2 <= \ldots <= q_n$, corresponde a um vetor de permutação $q = [q_1, q_2, \ldots q_n] = [\sigma(1), \sigma(2), \ldots \sigma(n)]$. Um reordenamento coerente dos vértices de um grafo G é um reordenamento coerente com a ordem natural do grafo. Um reordenamento coerente num grafo acíclico é também dito um (re)**ordenamento topológico** (dos vértices) deste grafo.

Exemplo 6:

Listamos a seguir alguns reordenamentos nos grafos do Exemplo 3, indicando se satisfazem aos dois critérios de coerência. No grafo G_1: $q = [4; 1, 2; 3]$ $(1, 2)$, $q = [1, 3, 2, 4]$ (1), $q = [3, 4, 1, 2]$ (2), $q = [1, 3, 4, 2]$ $()$. No grafo G_2: $q = [3, 4, 1, 2]$ $(1, 2)$, $q = [1, 3, 2, 4]$ (1). No grafo G_3: $q = [1, 3, 2, 4]$ $(1, 2)$, $q = [1, 2, 3, 4]$ $(1, 2)$, e estes são os únicos reordenamentos coerentes. No grafo G_4: O único reordenamento coerente é $q = [1, 2, 3, 4]$.

Para reordenar coerentemente os vértices de um grafo precisamos pois determinar o grafo reduzido, e no grafo reduzido uma ordem topológica. Esta tarefa pode ser levada a cabo com o algoritmo de Tarjan, a ser visto a seguir.

3.3 Busca em Profundidade

A busca em profundidade é um procedimento que nos permite visitar, a partir de um determinado vértice v de $G = (N, \Gamma)$, todos os seus descendentes. Cada vértice será marcado como "já visitado" ou "não visitado", sendo "não visitado" o estado inicial de todos os vértices.

Na busca em profundidade, partindo de v, seguiremos um caminho, sempre por vértices não visitados, o mais longe possível. Ao passar por um vértice, marcá-lo-emos como "já visitado". Não sendo mais possível prosseguir de um dado vértice, isto é, quando estivermos num vértice sem filhos não visitados, retornaremos ao vértice de onde este foi atingido e prosseguimos na busca em profundidade. Quando tivermos voltado ao vértice inicial, v, e já tivermos visitado todos os seus filhos, teremos terminado a busca em profundidade. Os vértices visitados e as arestas utilizadas para atingí-los formam uma árvore de raiz v, que é a **árvore da busca**.

Iniciando novas buscas em profundidade, nas quais consideramos "não visitados" apenas os vértices que não pertencem a nenhuma árvore previamente formada, teremos uma floresta que cobre o grafo G, a floresta de busca.

Uma maneira de rotularmos (ou reordenarmos) os vértices de G durante a busca em profundidade é pela **ordem de visitação**, $Ov()$, onde $Ov(i) = k$ significa que o vértice i foi o k-ésimo vértice a ser visitado na formação da floresta de busca. Uma maneira alternativa de rotular (re-ordenar) os vértices de uma floresta de busca é a **ordem de retorno**: $Or()$, é a ordem em que verificamos já termos visitado todos os filhos de um vértice e retornamos ao seu pai na árvore (ou terminamos a árvore se se tratar de uma raiz).

Estando os vértices de um grafo bem ordenados por algum critério, por exemplo pelos seus índices, a floresta de busca em profundidade canônica é aquela em que tomamos os vértices, tanto para raízes de novas árvores quando para visitação dentro de uma busca, na ordem estabelecida por este critério. No exemplo 7, a primeira é a floresta canônica.

Exemplo 7:

Um grafo G, e várias das possíveis florestas de busca que o cobrem, estão dadas na figura seguinte. Apresentamos estas florestas com os vértices numerados primeiro pela ordem de visitação, depois pela ordem de retorno. Indicamos com um circunflexo as raizes da busca.

$$
\begin{array}{ccccc}
 & & 3 & \to & 6 \\
 & \swarrow & \downarrow & \nwarrow & \downarrow \\
1 & \to & 4 & \leftarrow & 7 \\
 & \swarrow & \downarrow & \nwarrow & \\
2 & \leftarrow & 5 & \to & 8
\end{array}
$$

$$
\begin{array}{ccccc}
 & & \hat{6} & \to & 7 \\
 & & & & \downarrow \\
\hat{1} & \to & 2 & & 8 \\
 & \swarrow & \downarrow & & \\
3 & & 4 & \to & 5
\end{array}
\qquad
\begin{array}{ccccc}
 & & \hat{1} & \to & 7 \\
 & \swarrow & \downarrow & & \downarrow \\
6 & & 2 & & 8 \\
 & \swarrow & \downarrow & & \\
3 & & 4 & \to & 5
\end{array}
\qquad
\begin{array}{ccccc}
 & & \hat{6} & \to & 7 \\
 & & & & \downarrow \\
\hat{5} & & 4 & & 8 \\
 & & & \nwarrow & \\
\hat{1} & & \hat{2} & \to & 3
\end{array}
$$

$$
\begin{array}{ccccc}
 & & \hat{8} & \rightarrow & 7 \\
 & & & & \downarrow \\
\hat{5} & \rightarrow & 4 & & 6 \\
 & \swarrow & \downarrow & & \\
1 & & 3 & \rightarrow & 2
\end{array}
\qquad
\begin{array}{ccccc}
 & & \hat{8} & \rightarrow & 7 \\
 & \swarrow & \downarrow & & \downarrow \\
5 & & 4 & & 6 \\
 & \swarrow & \downarrow & & \\
1 & & 3 & \rightarrow & 2
\end{array}
\qquad
\begin{array}{ccccc}
 & & \hat{8} & \rightarrow & 7 \\
 & & & & \downarrow \\
\hat{5} & & 2 & & 6 \\
 & & & \nwarrow & \\
\hat{1} & & \hat{4} & \rightarrow & 3
\end{array}
$$

Descrevemos agora o **algoritmo de Tarjan** para determinação das componentes fortemente conexas de um grafo G.

1. Considerando a boa ordem dos índices, construa a floresta de busca canônica em G, marcando seus vértices pela ordem de retorno;

2. Considere G^{-1} com os vértices reordenados, isto é rotulados, na ordem inversa da ordem de retorno estabelecida no passo 1;

3. Considerando a boa ordem estabelecida no passo 2, construa a floresta de busca canônica em G^{-1}.

Teorema 3.2 (Tarjan) : *Cada árvore em G^{-1} construída no passo 3 do algoritmo de Tarjan cobre os vértices de exatamente uma componente fortemente conexa de G. Mais ainda, a ordem de visitação (bem como a ordem de retorno), na floresta canônica do passo 3 do Algoritmo de Tarjan, é um reordenamento coerente dos vértices de G.*

Demonstração:

Se $v, w \in V$ estão numa mesma componente de G então certamente pertencem a uma mesma árvore em G^{-1} (bem como em G). Se x é um vértice de G, denotaremos o número que marca o vértice x, pela ordem de retorno em G, por $Or(x)$. Se w é um vértice de uma árvore de raiz v, em G^{-1}, então:

1. v descende de w, em G, pois w descende de v em G^{-1}.

2. w descende de v, em G.

Para justificar a segunda afirmação notemos que, por construção $Or(v) > Or(w)$. Isto significa que ou w foi visitado durante a busca em profundidade a partir de v, ou w foi visitado antes de v. A primeira hipótese implica em (2) enquanto a segunda hipótese é impossível pois, como por (1) v descende de w, jamais poderíamos ter deixado v “não visitado” após concluir uma busca em profundidade que visitasse w. Q.E.D.

Exemplo 8:

A figura seguinte apresenta os passos na determinação pelo algoritmo de Tarjan, das CFCs do exemplo 7, e nos dá um reordenamento coerente de seus vértices. A floresta de BEP em G está rotulada pela ordem de retorno, e a floresta em G^{-1} pela ordem de visitação. O reordenamento coerente dos vértices do grafo original obtido neste exemplo é $q = [3, 6, 7 \mid 1 \mid 4, 5, 8 \mid 2]$.

$$
\begin{array}{ccccc}
 & & 3 & \rightarrow & 6 \\
 & \swarrow & \downarrow & \nwarrow & \downarrow \\
1 & \rightarrow & 4 & \leftarrow & 7 \\
 & \swarrow & \downarrow & \nwarrow & \\
2 & \leftarrow & 5 & \rightarrow & 8
\end{array}
\qquad\qquad
\begin{array}{ccccc}
 & & \hat{8} & \rightarrow & 7 \\
 & & & & \downarrow \\
\hat{5} & \rightarrow & 4 & & 6 \\
 & \swarrow & \downarrow & & \\
1 & & 3 & \rightarrow & 2
\end{array}
$$

$$
\begin{array}{ccccc}
 & & 1 & \leftarrow & 2 \\
 & \nearrow & \uparrow & \searrow & \uparrow \\
4 & \leftarrow & 5 & \rightarrow & 3 \\
 & \nearrow & \uparrow & \searrow & \\
8 & \rightarrow & 6 & \leftarrow & 7
\end{array}
\qquad\qquad
\begin{array}{ccccc}
 & & \hat{1} & & 3 \\
 & & & \searrow & \uparrow \\
\hat{4} & & \hat{5} & & 2 \\
 & & & \searrow & \\
\hat{8} & & 7 & \leftarrow & 6
\end{array}
$$

3.4 Grafos Simétricos e Casamentos

Um grafo $G = (N, \Gamma)$ é dito **simétrico** sse $\forall i, j \in N,\ j \in \Gamma(i) \Rightarrow i \in \Gamma(j)$. É usual representarmos um grafo simétrico substituindo cada par de arestas, (i, j) e (j, i), por uma aresta sem orientação ou **lado**, $\{i, j\}$. Podemos definir um grafo simétrico através de seus vértices e dos seus lados, $G = (N, E)$, onde cada lado é um conjunto de dois vértices e $\{i, j\} \in E \Leftrightarrow i \in \Gamma(j)$.

Um m-**casamento**, num grafo simétrico, é um conjunto de m lados onde são distintos todos os vértices; $M = \{\{u_1, u_2\}, \{u_3, u_4\}, \ldots \{u_{2m-1}, u_{2m}\}\}$. Os vértices em M dizem-se casados, os de mesmo lado dizem-se companheiros, e os vértices fora do casamento dizem-se solteiros. Um m-casamento é máximo em G se não houver em G um $m+1$ casamento e é perfeito se não deixar nenhum vértice solteiro.

Dado um grafo simétrico $G = (V, E)$ e nele um m-casamento, M, um **caminho M-alternado** é um caminho $C = (\{w_0, w_1\}, \{w_1, w_2\}, \ldots \{w_{k-1}, w_k\})$ que tem lados, alternadamente, dentro e fora do casamento. Um caminho M-alternado diz-se um **caminho de aumento** se começa e termina em vértices solteiros.

Teorema 3.3 (Berge) : *Dado um grafo simétrico, $G = (V, E)$ e nele um m-casamento, $M = \{\{u_1, u_2\}, \{u_3, u_4\}, \ldots \{u_{2m}, u_{2m}\}\}$, este casamento é máximo se não houver caminho de aumento.*

Demonstração:

Suponha que existe um caminho de aumento
$C = (\{w_0, w_1\}, \{w_1, w_2\}, \ldots \{w_{2k-1}, w_{2k+1}\})$ então

$$
\begin{aligned}
M' = M \oplus C \quad = \quad & M - \{\{w_1, w_2\}, \{w_3, w_4\}, \ldots \{w_{2k-1}, w_{2k}\}\} \\
& + \{\{w_0, w_1\}, \{w_2, w_3\}, \ldots \{w_{2k}, w_{2k+1}\}\}
\end{aligned}
$$

é um $m+1$-casamento.

Se, por outro lado, M não é máximo, seja M'um m+1-casamento e considere o grafo auxiliar H de lados $E_H = M' \oplus M$, sendo $V_H = V_G$. Cada vértice de H é isolado, ou pertence a algum lado de E_H e no máximo a dois lados, um de M e outro de M'. Cada componente de H é pois, ou um vértice isolado, ou um ciclo de lados alternadamente em M e M', ou um caminho de lados alternadamente em cada um dos casamentos. Como H tem mais lados de M'que de M, há ao menos uma componente de tipo caminho que começa e termina com lados de M'. Este caminho é M-alternado e seus vértices extremos são,por construção, solteiros em M. Temos assim um caminho de aumento para M. Q.E.D.

Exemplo 9:

Apresentamos agora: Na primeira linha: M: um m-casamento num grafo, C: um caminho de aumento, e M': o $m+1$-casamento $M' = M \oplus C$. Na segunda linha: M: um m-casamento, M': um $m+1$-casamento, e H: o grafo $H = M' \oplus M$.

$$
\begin{array}{ccccc}
1 & \cdots & 2 & \cdots & 3 \\
| & & | & & \vdots \\
4 & \cdots & 5 & \cdots & 6 \\
\vdots & & \vdots & & | \\
7 & \cdots & 8 & \cdots & 9
\end{array}
\qquad
\begin{array}{ccccc}
1 & \cdots & 2 & & 3 \\
| & & | & & \\
4 & & 5 & \cdots & 6 \\
\vdots & & & & | \\
7 & & 8 & \cdots & 9
\end{array}
\qquad
\begin{array}{ccccc}
1 & - & 2 & \cdots & 3 \\
\vdots & & \vdots & & \vdots \\
4 & \cdots & 5 & - & 6 \\
| & & \vdots & & \vdots \\
7 & \cdots & 8 & - & 9
\end{array}
$$

$$
\begin{array}{ccccc}
1 & \cdots & 2 & \cdots & 3 \\
| & & | & & \vdots \\
4 & \cdots & 5 & \cdots & 6 \\
\vdots & & \vdots & & | \\
7 & \cdots & 8 & \cdots & 9
\end{array}
\qquad
\begin{array}{ccccc}
1 & - & 2 & \cdots & 3 \\
\vdots & & \vdots & & | \\
4 & - & 5 & \cdots & 6 \\
\vdots & & \vdots & & \vdots \\
7 & \cdots & 8 & - & 9
\end{array}
\qquad
\begin{array}{ccccc}
1 & \cdots & 2 & & 3 \\
| & & | & & \vdots \\
4 & \cdots & 5 & & 6 \\
 & & & & | \\
7 & & 8 & \cdots & 9
\end{array}
$$

Um grafo $G = (V, \Gamma)$ é **bipartido** se houver uma bipartição de seus vértices tal que todos os lados tenham um vértice em cada pedaço da bipartição, i.e. se for possível encontrar conjuntos $X, Y \mid V = X \cup Y \ \wedge \ X \cap Y = \emptyset \ \wedge \ \forall e \in E, e = \{x, y\} \ x \in X, y \in Y$.

Teorema 3.4 (Hall) : *Dado $G = (V, \Gamma)$ um grafo simétrico e bipartido, com bipartição $V = X \cup Y$, existe em G um casamento que casa todos os vértices de X sse vale a condição de Hall: $\forall S \subseteq X, \#\Gamma(S) \geq \#S$. isto é, qualquer subconjunto de X tem, coletivamente, ao menos tantos filhos quanto elementos.*

Demonstração

Se existe um casamento M que casa todos os vértices em X, então os lados do casamento garantem ao menos a igualdade na condição de Hall. Por outro lado, se houver um casamento máximo, M, que deixe algum $x \in X$ solteiro, exibiremos um $S \subseteq X$ que viola a condição de Hall.

Seja Z o conjunto dos vértices conexos a x por caminhos M-alternados. Pelo teorema de Berge sabemos que não há solteiro em Z, pois caso contrário teríamos um caminho de aumento e M não seria máximo. Assim, um caminho M-alternado maximal, isto é, que não pode ser continuado, que começa no solteiro $x \in S \subseteq X$, deve necessariamente terminar, com um número par de lados, num casado $x' \in S \subseteq X$.

Considerando, pois, $S = (Z \cap X) + x$ e $T = Z \cap Y$, temos que $\#S = \#T + 1$ e $\Gamma(S) = T$, donde $\#\Gamma(S) = \#T = \#S - 1 < \#S$, mostrando que S viola a condição de Hall. Q.E.D.

3.5 O Algoritmo Húngaro

Dado um grafo simétrico $G = (V, E)$, bipartido em conjuntos de mesma cardinalidade X e Y, o algoritmo húngaro fornece um casamento perfeito ou encontra um conjunto $S \subset X$ que viola a condição de Hall.

Seja M um casamento que deixa $x \in X$ solteiro. Uma árvore de raiz x, $H = (V_H, E_H)$, subgrafo de G, é M-alternada se for uma árvore onde qualquer caminho partindo de x, $C = (x, w_1, w_2, \ldots)$ for M-alternado. Observemos que se uma dada árvore M-alternada de raiz x tiver uma folha solteira, então o caminho C que vai da raiz a esta folha é um caminho de aumento e $M' = M \oplus C$ é um novo casamento onde, além de todos os vértices casados em M, x também é casado.

Se, todavia, encontrarmos uma árvore M-alternada de raiz solteira, onde todas as folhas são casadas e que seja máxima, isto é, tal que não exista nenhum vértice em $V_G - V_H$, adjacente a uma folha de H teremos encontrado um $S \subset X$ que viola a condição de Hall, $S = V_H \cap X$ e $T = V_H \cap Y = \Gamma(S)$.

O algoritmo Húngaro faz o seguinte: dado um grafo bipartido e simétrico com um casamento que deixa $x \in X$ solteiro, gerar a árvore canônica de busca em profundidade M-alternada, até encontrar um caminho de aumento, isto é, uma folha solteira. Caso isto não seja possível constatarmos a violação da condição de Hall.

Exemplo 10:

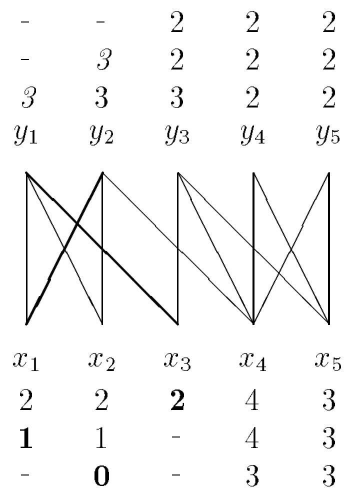
No exemplo 10 temos dois exemplos de grafo e casamento, nos quais está assinalado um vértice w solteiro e sem mais nenhum pretendente disponível. Temos as respectivas árvores M-alternadas, marcadas pela ordem de visitação, os caminhos de aumento, C, e os novos casamentos, $M' = M \oplus C$, que casam o vértice w.

O algoritmo Húngaro, bem como qualquer algoritmo casamenteiro conhecido, não é linear; Portanto vale a pena procurar heurísticas mais eficientes para a procura de um casamento perfeito. Dado um grafo e um casamento, os pretendentes de um vértice solteiro são os solteiros em seu conjunto de adjacência, e o peso de um solteiro, $\rho(x)$, é seu numero de pretendentes. A idéia subjacente em todas estas heurísticas é a de casar primeiro os vértices mais exigentes, i.e. de menor peso, de modo a preservar o máximo de pretendentes para os vértices ainda solteiros.

Na **Heurística do Mínimo Simples**, HMS, dado um m-casamento imperfeito, procuraremos obter um m+1-casamento adicionando ao casamento um lado que una vértices ainda solteiros. Escolhemos este lado, de modo a casar um vértice de mínimo peso, entre os ainda solteiros, com um de mínimo peso dentre seus pretendentes. Sempre que houver um vértice isolado, i.e. sem pretendentes ou de peso zero, procuraremos casá-lo pelo algoritmo Húngaro. Na **Heurística do Mínimo Par**, HMP, escolhemos um lado a ser acrescentado ao casamento onde um dos vértices é de peso mínimo, e o outro tenha peso mínimo dentre todos os pretendentes a vértices de peso mínimo. Como a HMP pode ser bem mais custosa podemos, por exemplo, utilizar a HMS enquanto o peso mínimo dos vértices ainda solteiros for grande, i.e. acima de um dado limite, e a HMP caso contrário.

Exemplo 11:

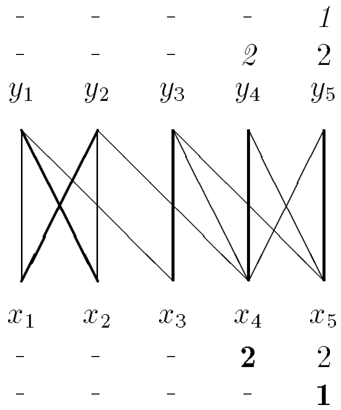
O exemplo seguinte ilustra aaplicação do algoritmo húngaro com a heurística min-min. No exemplo indicamos o peso, ou número de pretendentes, de cada vértice. Indicamos ainda, a cada passo, o vértice de nínimo peso a ser casado, bem como, em itálico, seu pretendente de mínimo peso.



A árvore M-alternada com raiz em x_2, que ficou isolado, nos fornece o caminho de aumento

$$c = \quad x_2 \; - \; y_1 - x_3 \; - \; y_3$$

de modo que podemos recomeçar a heurrística em $M \oplus c$:


- - - - *1*
- - - *2* 2
y_1 y_2 y_3 y_4 y_5
x_1 x_2 x_3 x_4 x_5
- - - **2** 2
- - - - **1**

Capítulo 4

ELIMINAÇÃO ASSIMÉTRICA

Esparsidade na Fatoração LU

A maioria dos sistemas lineares encontrados na prática, especialmente os sistemas grandes, são esparsos, isto é, apenas uma pequena parte dos coeficientes do sistema não são nulos. Ao tratar um sistema esparso, devemos tentar manter a esparsidade do sistema ao longo das tranformações necessárias à sua solução. Tal medida visa, primordialmente, minimizar o número de operações aritméticas a serem realizadas, na própria transformação e na solução do sistema. Economia de memória também é um fator importante.

4.1 Preenchimento Local

No método de Gauss as tranformações ${}^kA \rightarrow {}^{k+1}A$ podem diminuir o número de elementos nulos. Se tal ocorrer, dizemos que houve **preenchimento** de algumas posições. Queremos escolher o elemento pivô em ${}^kA$ de modo a minimizar este preenchimento.

Teorema 4.1 (Tewarson) *Seja*

$$\begin{bmatrix} {}^kA_1^1 & & \dots & & {}^kA_1^n \\ 0 & \ddots & & & \\ \vdots & 0 & {}^kA_{k+1}^{k+1} & \dots & {}^kA_{k+1}^n \\ & \vdots & \vdots & & \vdots \\ 0 & 0 & {}^kA_n^{k+1} & \dots & {}^kA_n^n \end{bmatrix}$$

seja ${}^kB$, $(n-k)\times(n-k)$, *a matriz Booleana associada a submatriz das* $n-k$ *últimas linhas e colunas de* ${}^kA$, *i.e.*

$$ {}^kB = B(A_{k+1:n}^{k+1:n}), \quad ou \quad {}^kB_p^q = 1 \Leftrightarrow {}^kA_{k+p}^{k+q} \neq 0 \ ,$$

e seja

$$ {}^kG = {}^kB({}^k\bar{B})'\,{}^kB\ . $$

onde o operador complemento aplicado à matriz B, $\bar{B}$, troca 0's por 1's e vice-versa.

A escolha do pivô ${}^kA_{k+i}^{k+j} \neq 0$ implica no prenchimento de exatamente ${}^kG_i^j$ posições.

Demonstração:

Seja ${}^k\tilde{A}$ a matriz obtida de ${}^kA$ por permutação da $k+1$-ésima linha e coluna com, respectivamente, a $k+i$-ésima linha e a $k+j$-ésima coluna.

Os novos elementos de ${}^{k+1}A$ serão, para $p,q\ = 2\ldots(n-k)$,

$$
\begin{aligned}
&{}^{k+1}A_{k+p}^{k+q} = \\
&= {}^k\tilde{A}_{k+p}^{k+q} - {}^{k+1}M_{k+p}^{k+1}\ {}^k\tilde{A}_{k+1}^{k+q} \\
&= {}^k\tilde{A}_{k+p}^{k+q} - {}^k\tilde{A}_{k+p}^{k+1}\ {}^k\tilde{A}_{k+1}^{k+q}\ /\ {}^k\tilde{A}_{k+1}^{k+1} \\
&= {}^kA_{k+r}^{k+s} - {}^kA_{k+r}^{k+j}\ {}^kA_{k+i}^{k+s}\ /\ {}^kA_{k+i}^{k+j}
\end{aligned}
$$

onde

$$ r = p \text{ se } p \neq i,\ \text{e } r = 1 \text{ se } p = i\ ; \quad \text{e} \quad s = q \text{ se } q \neq j,\ \text{e } s = 1 \text{ se } q = j\ . $$

Haverá preenchimento de uma posição em ${}^{k+1}A$ sempre que,
para $r,s = 1\ldots(n-k)$, $r \neq i$ e $s \neq j$,

$$ {}^kB_r^s = 0 \quad \wedge \quad {}^kB_r^j = 1 \quad \wedge \quad {}^kB_i^s = 1 $$

e portanto o total de preenchimentos, correspondente ao pivô $(k+i,\ k+j)$, é:

$$
\begin{aligned}
&\sum_{r=1,\ r\neq i}^{n-k}\ \sum_{s=1,\ s\neq j}^{n-k} {}^k\bar{B}_r^s\ {}^kB_r^j\ {}^kB_i^s = \\
&= \sum_{r,s=1}^{n-k} {}^kB_r^j(({}^k\bar{B})')_s^r\ {}^kB_i^s \\
&= ({}^kB({}^k\bar{B})'\,{}^kB)_i^j = G_i^j
\end{aligned}
$$

Na penúltima passagem usamos que os termos $r = i$ ou $s = j$ são todos nulos,
pois $B_p^q\bar{B}_p^q = 0$. QED.

Observação 4.1 Uma aproximação para a matriz ${}^kG$ é a **matriz de Markowitz**:

$$ {}^kF_i^j = ({}^kB\ \mathbf{1}\ {}^kB)_i^j\ , $$

onde denotamos por **1** a matriz quadrada de 1's. A aproximação F é exatamente o número de multiplidores não nulos vezes o número de elementos não nulos, fora o pivô, na linha pivô. Esta aproximação é bastante boa se ${}^kB$ é muito esparsa.

Exemplo 1:

Dada a matriz ${}^0A$, cujos elementos não nulos estão indicados na matriz boleana associada $B({}^0A)$, indique uma escolha de pivôs que minimize preeenchimentos locais. Assuma que ao longo do processo de triangularização não há cancelamentos, i.e., que um elemento não nulo uma vez preenchido, não volta a se anular.

Tomando

$$
{}^0B = B({}^0A) = {}^0B = \begin{bmatrix} \mathbf{1} & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix} \quad , \quad {}^0G = \begin{bmatrix} \mathbf{0} & 3 & 1 & 5 & 3 \\ 0 & 1 & 0 & 3 & 2 \\ 2 & 3 & 3 & 1 & 2 \\ 3 & 6 & 5 & 1 & 1 \\ 1 & 6 & 3 & 6 & 3 \end{bmatrix}
$$

Assim, $arg\min_{(i,j)\,|\,{}^0B_i^j=1} \ {}^0G_i^j \ = \{(1,1),(2,3)\}$.

Escolhendo $(i,j) = (1,1)$, i.e. ${}^0A_1^1$ como pivô, temos o preenchimento de ${}^0G_1^1 = 0$ posições em ${}^1A$,

$$
{}^1B = B({}^1A + {}^1M) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ \mathit{0} & 1 & \mathbf{1} & 0 & 0 \\ \mathit{0} & 1 & 0 & 1 & 0 \\ \mathit{0} & 0 & 0 & 1 & 1 \\ \mathit{1} & 1 & 1 & 0 & 1 \end{bmatrix} \quad , \quad {}^1G = \begin{bmatrix} 1 & \mathbf{0} & 3 & 2 \\ 2 & 2 & 1 & 2 \\ 4 & 3 & 1 & 1 \\ 3 & 1 & 4 & 2 \end{bmatrix}
$$

Assim, $arg\min_{(i,j)\,|\,{}^1B_i^j=1} \ {}^1G_i^j \ = \{(1,2)\}$.

Escolhendo $(i,j) = (1,2)$, i.e. ${}^1A_2^3$ como pivô, temos o preenchimento de ${}^1G_1^2 = 0$ posições em ${}^2A$,

$$
{}^2B = B({}^2A + {}^2M) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ \mathit{0} & 1 & 1 & 0 & 0 \\ \mathit{0} & \mathit{0} & \mathbf{1} & 1 & 0 \\ \mathit{0} & \mathit{0} & 0 & 1 & 1 \\ \mathit{1} & \mathit{1} & 1 & 0 & 1 \end{bmatrix} \quad , \quad {}^1G = \begin{bmatrix} \mathbf{1} & 1 & 2 \\ 2 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}
$$

Assim, $arg\min_{(i,j)\,|\,{}^2B_i^j=1} \ {}^2G_i^j \ = \{(1,1),(1,2),(2,2),(2,3),(3,1),(3,3)\}$.

Escolhendo $(i,j) = (1,1)$, i.e. ${}^2A_3^3$ como pivô, temos o preenchimento de ${}^2G_1^1 = 1$ posições em ${}^3A$,

$$
{}^3B = B({}^3A + {}^3M) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ \mathit{0} & 1 & 1 & 0 & 0 \\ \mathit{0} & \mathit{0} & 1 & 1 & 0 \\ \mathit{0} & \mathit{0} & \mathit{0} & 1 & 1 \\ \mathit{1} & \mathit{1} & \mathit{1} & \mathbf{1} & 1 \end{bmatrix} \quad , \quad {}^3B = \begin{bmatrix} 1 & 1 \\ \mathbf{1} & 1 \end{bmatrix}
$$

Como ${}^3B = \mathbf{1}$, é obvio que não haverá mais preenchimento, quaisquer que que sejam os pivôs doravante selecionados. Tomando, por exemplo, o pivô ${}^3A_5^4$, temos finalmente temos a fatoração

$\tilde{A} = PAQ = LU$, com

$$B(\tilde{A}) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad , \quad B(M+U) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ \mathit{0} & 1 & 1 & 0 & 0 \\ \mathit{0} & \mathit{0} & 1 & 1 & 0 \\ \mathit{1} & \mathit{1} & \mathit{1} & 1 & 1 \\ \mathit{0} & \mathit{0} & \mathit{0} & \mathit{1} & 1 \end{bmatrix}$$

sendo os vetores de indices permutados ${}^4p = [1,2,3,5,4]$ e ${}^4q = [1,3,2,4,5]$. Neste processo de triangulaziração, tivemos assim o preenchimento de apenas 1 posição, em ${}^3A_5^4$. //

Observação 4.2 Para não fazer a seleção de pivôs apenas pelos critérios para minimização de preenchimento local, desprezando assim completamente os apectos de estabilidade numérica, podemos "vetar" escolhas de pivôs muito pequenos, i.é., que impliquem no uso de multiplicadores muito grandes, $|M_i^j| \geq multmax \gg 1$. Se nosso sistema for bem equilibrado, podemos usar como critério de veto o tamanho do proprio pivô, $|U_i^i| \leq pivomin \ll 1$.

4.2 Pré-Posicionamento de Pivôs

Minimizar o preenchimento local é um processo custoso, tanto no número de operações envolvidas no cálculo de ${}^kG$, como por termos de acessar toda a matriz ${}^kA$, o que é um grande inconveniente para o armazenamento eficiente da matriz. Ademais, minimizar o preenchimento local, i.e. aquele causado por cada pivô individualmente, é uma estratégia gulosa que pode ter um fraco desempenho global, i.e. causar muito mais preenchimento que a seqüência ótima de n pivôs. Visando superar as deficiências dos métodos locais estudaremos a **heurística P3**, ou Procedimento de Pré-determinação de Pivôs.

A heurística P3 procura uma permutação de linhas e colunas, $B = PAQ$, que reduza o preenchimento na fatoração $B = LU$. O procedimento que implementa a heurística P3 posiciona as colunas de A em B e numa matriz temporária, S, e permuta linhas de todas as colunas, tentando produzir uma B que seja quase triangular inferior. As colunas de B que não tem ENNs acima da diagonal denominam-se **colunas triangulares**. Colunas triangulares são sempre posicionadas com um ENN na diagonal, que será usado como pivô da coluna na fatoração $B = LU$. As colunas de B que tem ENNs acima da diagonal denominam-se **espinhos**. Na fatoração LU de B, somente poderá ocorrer preenchimento dentro dos espinhos. Ademais, não haverá preenchimento acima do mais alto (menor índice de linha) ENN num espinho. Portanto, queremos minimizar o número de espinhos e, como objetivo secundário, minimizar a altura dos espinhos acima da diagonal.

A heurística P3 procede como segue:

1. Compute o número de ENNs em cada linha e coluna de A, ou os pesos de linha e coluna, respectivamente, $pr(i)$ e $pc(j)$.

2. Compute $h = arg\min_i pr(i)$ e $\rho = pr(h)$.

3. Compute a ρ-**altura** de cada coluna, $\Theta_\rho(j)$,
o numero de ENNs na coluna j em linhas de peso ρ.

4. Compute $t = arg\max_j \Theta_\rho(j)$.

   (a) Se $\rho = 1$, seja $h \in \{i \mid pr(i) = 1 \wedge A(i,t) \neq 0\}$; posicione t como a primeira coluna de A, h como a primeira linha, e aplique P3 recursivamente a A(2:m,2:n). A coluna excluída torna-se a última coluna de B.

   (b) Se $\rho > 1$, posicione a coluna t como a última coluna de A, e aplique P3 recursivamente a A(:,1:n-1). A coluna excluída torna-se a primeira coluna de S.

   (c) Se $\rho = 0$, posicione h como a primeira linha, reposicione a primeira coluna de S (última a ser excluída de A) como a última coluna de B, e aplique P3 recursivamente a A(2:m,:).

Em P3, o caso

- $\rho = 1$ corresponde ao posicionamento de uma coluna triangular em B.

- $\rho > 1$ corresponde a impossibilidade de posicionar uma coluna triangular. Assim (temporariamente) eliminamos uma coluna de A e prosseguimos com P3. O critério de seleção para a coluna a eliminar visa produzir o caso $\rho = 1$ nos próximos passos de P3. Em caso de empate na ρ-altura, poderíamos usar como desempate a $(\rho + 1)$-altura.

- $\rho = 0$ corresponde a não haver em A uma coluna que pudéssemos posicionar em B de modo a continuar formando B não singular. Então reintroduzimos, como próxima coluna de B, a primeira coluna em S. Escolhemos a primeira em S visando minimizar a altura do espinho acima da diagonal. Existe o perigo de que este espinho não venha a prover um pivô não nulo, ou que após cancelamentos, o pivô seja muito pequeno para garantir a estabilidade numérica da fatoração. Se A é esparsa e tivermos apenas alguns espinhos, a melhor solução é intercalar a fatoração numérica e simbólica, o que nos dá a chance de rejeitar pivôs instáveis. Métodos para lidar com pivôs inaceitáveis na fase de fatoração numérica, após termos completado uma fase independente de fatoração simbólica, são discutidos posteriormente.

Exemplo

Apliquemos o P3 à matriz booleana A. À medida que o P3 prossegue representaremos a matriz particionada $\left[\ B, A, S\ \right]$, cujos ENNs serão denotados respectivamente b, a e s. Os índices de linha e coluna estão a esquerda e acima da matriz. Os pesos, ρ, e as ρ-alturas, Θ_ρ, a direita e abaixo da matriz. Os pivôs, ou espinhos, escolhidos a cada passo estão em negrito.

$p\backslash q$	1	**2**	3	4	5	ρ
1	a	a	a	a		4
2	a	a				2
3			a	a		2
4		a		a	a	3
5		a		a	a	3
Θ_2	1	1	1	1	0	
Θ_3	1	3	1	3	.	
Θ_4	2	4	2	4	.	

,

$p\backslash q$	1	3	4	5	2	ρ
1	a	a	a		s	3
2	$\mathbf{a}$				s	1
3		a	a			2
4			a	a	s	2
5			a	a	s	2
Θ_1	1	0	0	0	.	

$p\backslash q$	1	3	**4**	5	2	ρ
2	$\mathbf{b}$				s	.
1	b	a	a		s	2
3		a	a			2
4			a	a	s	2
5			a	a	s	2
Θ_2	.	2	4	2	.	

,

$p\backslash q$	1	3	5	4	2	ρ
2	$\mathbf{b}$				s	.
1	b	$\mathbf{a}$		s	s	1
3		a		s		1
4			a	s	s	1
5			a	s	s	1
Θ_1	.	2	2	.	.	

$p\backslash q$	1	3	5	4	2	ρ
2	$\mathbf{b}$				s	.
1	b	$\mathbf{b}$		s	s	.
3		b		$\mathbf{s}$		0
4			a	s	s	1
5			a	s	s	1

,

$p\backslash q$	1	3	4	5	2	ρ
2	$\mathbf{b}$				s	.
1	b	$\mathbf{b}$	b		s	.
3		b	$\mathbf{b}$			.
4			b	$\mathbf{a}$	s	1
5			b	a	s	1
Θ_1	.	.	.	2	.	

$p\backslash q$	1	3	4	5	2
2	$\mathbf{b}$				b
1	b	$\mathbf{b}$	b		b
3		b	$\mathbf{b}$		$+$
4			b	$\mathbf{b}$	b
5			b	b	$\mathbf{b}$

Na permutação final obtida pela P3, dada pelos vetores de permutação p e q, indicamos com um $+$ as posiccões a serem preenchidas no processo de eliminação. Uma notação mais compacta para o mesmo exemplo seria:

s		1		2							
$\bar{p}\backslash\bar{q}$	1	5	2	3	4			ρ			
2	x	x	$\mathbf{x}$	x		4	3	2	1	.	.
1	$\mathbf{x}$	x				2	1	.	.	.	.
3			x	$\mathbf{x}$		2	2	2	1	0	.
4		x		x	$\mathbf{x}$	3	2	2	1	1	1
5		$\mathbf{x}$		x	x	3	2	2	1	1	1
Θ_2	1	1	1	1	0						
Θ_3	1	3	1	3	.						
Θ_2	.	.	1	4	1						

No exemplo seguinte reintroduzimos um espinho com 0 na posição pivô. Todavia, no processo de eliminação, esta posição será preenchida antes da sua utilização como pivô.

s		1	2										
$\bar{p}\backslash\bar{q}$	1	6	4	5	3	2							
1	$\mathbf{x}$	x					2	1	.	.	.	.	.
2		x	x			$\mathbf{x}$	3	2	2	1	.	.	.
3		x	x		$\mathbf{x}$		3	2	2	1	1	.	.
5	x		x	$\mathbf{x}$	x		4	4	3	2	2	1	.
4	x	x	**0**		x	x	4	3	2	2	1	0	.
6		$\mathbf{x}$	x	x		x	4	3	3	2	2	1	0
Θ_2	1	1	0	0	0	0							
Θ_3	1	3	.	.	.	.							
Θ_2	.	.	2	0	2	2							
Θ_3	.	.	4	.	3	3							

$p\backslash q$	1	6	5	3	4	2
1	$\mathbf{x}$					x
2		$\mathbf{x}$		x		x
3			$\mathbf{x}$	x		x
5	x	x	x	$\oplus$		x
4	x		x	x	$\mathbf{x}$	$+$
6		x		x	x	$\mathbf{x}$

Observação 4.3 Da maneira como descrevemos o P3, poderíamos aplicá-lo também a uma matriz retangular. No caso de Programação linear é comum ordenarmos pelo P3 todas as colunas da matriz de restrições, A, e depois tomarmos, a cada reinversão, as colunas na base, B, conforme a ordem estabelecida por P3 em A [Orchard-Hays68].

Exercícios

1. Implemente a fatoração LU usando a heurística de Markowitz. Antes de aceitar um pivô, assegure-se que este tem módulo maior que $pivmin = 10^{-4}$, substituindo-o caso contrário. Use a representação estática por colunas e de rede da matriz.

2. Implemente o P3 para ordenar as colunas de uma matriz retangular, conforme a observação 6.3. Use a representação estática por coluna da matriz.

3. Considere a possibilidade de termos um espinho problemático que, após o processo de eliminação das colunas precedentes, apresente um 0 na posição pivô. Mostre que necessariamente existe um espinho, à direita do problemático, cujo elemento na mesma linha do pivô do espinho problemático, neste estágio da fatoração da matriz, é diferente de zero. Descreva um procedimento para lidar com estes casos excepcionais através da permutação de colunas espinhos. Discuta a viabilidade de resolver o problema através de permutação de linhas.

Capítulo 5

FATORAÇÕES SIMÉTRICAS e ORTOGONAIS

Projetores e Problemas Quadráticos

5.1 Matrizes Ortogonais

Dizemos que uma matriz quadrada e real é **ortogonal** sse sua transposta é igual a sua inversa.Dada Q uma matriz ortogonal, suas colunas formam uma base ortonormal de $\Re^n$, como pode ser visto da identidade $Q'Q = I$. O quadrado da **norma quadrática** de um vetor v,

$$\|v\|^2 \equiv \sum_{i=1}^{n} (v_i)^2 = v'v$$

permanece inalterada por uma transformação ortogonal, pois

$$\|Qv\|^2 = (Qv)'(Qv) = v'Q'Qv = v'Iv = v'v = \|v\|^2 .$$

5.2 Fatoração QR

Dada uma matriz real A $m \times n$, $m \geq n$, podemos existe uma matriz ortogonal Q tal que $A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}$, onde R é uma matriz quadrada e triangular superior. Esta decomposição é dita uma fatoração QR, ou **fatoração ortogonal**, da matriz A. Descrevemos a seguir um método para fatoração ortogonal.

A **rotação** de um vetor $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \Re^2$ por um ângulo θ é dada pela transformação linear

$$rot(\theta)x = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} .$$

Notemos que a rotação é uma transformação ortogonal, pois

$$rot(\theta)'rot(\theta) = \begin{bmatrix} \cos(\theta)^2 + \sin(\theta)^2 & 0 \\ 0 & \cos(\theta)^2 + \sin(\theta)^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} .$$

Além disso podemos tomar o ângulo $\theta = -\arctan(x_2/x_1)$ de modo que a rotação correspondente anule a segunda componente do vetor rodado (se $x_1 = 0$, tome $\theta = \pi/2$).

Como o produto de transformações ortogonais continua ortogonal (prove), podemos usar uma seqüência de rotações para levar a matriz A à forma triangular superior, como veremos a seguir.

A **rotação de Givens** é um operador linear cuja matriz coincide com a identidade, exceto num par de linhas onde imergimos uma matriz de rotação bidimencional:

$$G(i,j,\theta) = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \cos(\theta) & & \sin(\theta) & & \\ & & & \ddots & & & \\ & & -\sin(\theta) & & \cos(\theta) & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} .$$

Dizemos que a aplicação deste operador numa matriz A, GA, roda as linhas i e j de A de um ângulo θ.

Abaixo ilustramos uma seqüência de rotações de linhas que leva uma matriz 5×3 à forma triangular superior. Cada par de índices, (i, j), indica que rodamos estas linhas do ângulo apropriado para zerar a posição na linha i, coluna j. Supomos que, inicialmente, a matriz é densa, i.e. todos os seus elementos são diferentes de zero, e ilustramos o padrão de esparsidade da matriz nos estágios assinalados com um asterisco na seqüência de rotações.

$$(1,5)*(1,4)(1,3)(1,2)*(2,5)(2,4)(2,3)*(3,5)(3,4)*$$

$$\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \\ 0 & x & x \end{bmatrix} \quad \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{bmatrix} \quad \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & x \\ 0 & 0 & x \end{bmatrix} \quad \begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Tomando Q como a produtória das rotações de Givens, temos a fatoração $A = QR$, como procurada.

5.3 Espaços Vetoriais com Produto Interno

Dados dois vetores $x, y \in \Re^n$, o seu **produto escalar** é definido como

$$< x \mid y > \equiv x'y = \sum_{i=1}^{n} x_i y^i.$$

Com esta definição, o produto escalar é um operador que satisfaz as propriedades fundamentais de **produto interno**, a saber:

1. $< x \mid y >=< y \mid x >$, **simetria**.
2. $< \alpha x + \beta y \mid z >= \alpha < x \mid z > + \beta < y \mid z >$, **linearidade**.
3. $< x \mid x > \geq 0$, não negatividade.
4. $< x \mid x >= 0 \Leftrightarrow x = 0$, **definitividade**.

Através do produto interno, definimos a norma:

$$\|x\| \equiv < x \mid x >^{1/2};$$

e definimos também o ângulo entre dois vetores não nulos:

$$\Theta(x, y) \equiv \arccos(< x \mid y > / \|x\| \|y\|).$$

5.4 Projetores

Consideremos o subespaço linear gerado pelas colunas de uma matriz A, $m \times n$, $m \geq n$:

$$C(A) = \{y = Ax, x \in \Re^n\}.$$

Denominamos $C(A)$ de **imagem** de A, e o complemento de $C(A)$, $N(A')$, de **espaço nulo** de A',

$$N(A') = \{y \mid A'y = 0\}.$$

O fator ortogonal $Q = [C \mid N]$ nos dá uma base ortonormal de $\Re^m$ onde as n primeiras colunas são uma base ortonormal de $C(A)$, e as $m - n$ últimas colunas são uma base de $N(A')$, como pode ser visto diretamente da identidade $Q'A = \begin{bmatrix} R \\ 0 \end{bmatrix}$.

Definimos a **projeção** de um vetor $b \in \Re^m$ no espaço das colunas de A, pelas relações:

$$y = P_{C(A)} b \Leftrightarrow y \in C(A) \wedge (b - y) \perp C(A)$$

ou, equivalentemente,

$$y = P_{C(A)}b \Leftrightarrow \exists x \mid y = Ax \ \wedge A'(b - y) = 0.$$

No que se segue suporemos que A tem posto pleno, i.e. que suas colunas são linearmente independentes. Provemos que o projetor de b em $C(A)$ é dado pela aplicação linear

$$P_A = A(A'A)^{-1}A'.$$

Se $y = A((A'A)^{-1}A'b)$, então obviamente $y \in C(A)$. Por outro lado, $A'(b - y) = A'(I - A(A'A)^{-1}A')b = (A' - IA')b = 0$.

5.5 Quadrados Mínimos

Dado um sistema superdeterminado, $Ax = b$ onde a matriz A $m \times n$ tem $m > n$, dizemos que x^* "resolve" o sistema no sentido dos **quadrados mínimos**, ou que x^* é a "solução" de quadrados mínimos, sse x^* minimiza a norma quadrática do resíduo,

$$x^* = Arg \min_{x \in \Re^n} \|Ax - b\|,$$

Dizemos também que $y = Ax^*$ é a melhor aproximação, no sentido dos quadrados mínimos de b em $C(A)$.

Como a multiplicação por uma matriz ortogonal deixa inalterada a norma quadrática de um vetor, podemos procurar a solução deste sistema (no sentido dos quadrados mínimos) minimizando a transformação ortogonal do resíduo usada na fatoração QR de A,

$$\|Q'(Ax - b)\|^2 = \| \begin{bmatrix} R \\ 0 \end{bmatrix} x - \begin{bmatrix} c \\ d \end{bmatrix} \|^2 = \|Rx - c\|^2 + \|0x - d\|^2.$$

Da última expressão vê-se que a solução, a aproximação e o resíduo do problema original são dados, respectivamente, por

$$x^* = R^{-1}c \ , \quad y = Ax^* \quad \text{e} \quad z = Q \begin{bmatrix} 0 \\ d \end{bmatrix}.$$

Como já havíamos observado, as $m - n$ últimas colunas de Q formam uma base ortonormal de $N(A')$, logo $z \perp C(A)$, de modo que concluímos que $y = P_A b$!

5.6 Programação Quadrática

O problema de **programação quadrática** consiste em minimizar a função

$$f(y) \equiv (1/2)y'Wy + c'y \ , \quad W = W'$$

sujeitos às **restrições**

$$g_i(y) \equiv N_i' y = d_i.$$

Os gradientes de f e g_i são dados, respectivamente, por

$$\nabla_y f = y'W + c' \ , \ \text{e} \ \nabla_y g_i = N_i' \ .$$

As **condições de otimalidade** de primeira ordem (condições de Lagrange) estabelecem que as restrições sejam obedecidas, e que o gradiente da função sendo minimizada seja uma combinação linear dos gradientes das restrições. Assim a solução pode ser obtida em função do **multiplicador de Lagrange**, i.e. do vetor l de coeficientes desta combinação linear, como

$$N'y = d \ \wedge \ y'W + c' = l'N' \ ,$$

ou em forma matricial,

$$\begin{bmatrix} N' & 0 \\ W & N \end{bmatrix} \begin{bmatrix} y \\ l \end{bmatrix} = \begin{bmatrix} d \\ c \end{bmatrix} .$$

Este sistema de equações é conhecido como o **sistema normal**. O sistema normal tem por matriz de coeficientes uma matriz simétrica. Se a forma quadrática W for **positiva definida**, i.e.se $\forall x \ x'Wx \geq 0 \ \wedge \ x'Wx = 0 \Leftrightarrow x = 0$, e as restrições N forem lineramente independentes, a matriz de coeficientes do sistema normal será também positiva definida. Estudaremos a seguir como adaptar a fatoração de Gauss a matrizes simétricas e positivas definidas.

5.7 Fatoração de Cholesky

Após a fatoração de Gauss de uma matriz simétrica, $S = LU$, podemos pôr em evidência os elementos diagonais de U obtendo $S = LDL'$. Se S for positiva definida assim o será D, de modo que podemos escrever $D = D^{1/2}D^{1/2}$, $D^{1/2}$ a matriz diagonal contendo a raiz dos elementos em D. Definindo $C = LD^{1/2}$, temos $S = CC'$, a **fatoração de Cholesky** de S.

Analisemos agora algumas relações entre as fatorações de Cholesky e ortogonal (QR), e de que maneiras a fatoração ortogonal nos dá uma representação da inversa. Primeiramente observemos que se $A = QR$,

$$A'A = (QR)'QR = R'Q'QR = R'IR = L'L$$

i.e., o fator triangular da fatoração ortogonal é o transposto do fator de Cholesky da matriz simétrica $A'A$.

Veremos agora que, no caso de termos A quadrada e de posto pleno, o produto pela inversa, $y = A^{-1}x$, pode ser calculado sem o uso explícito do fator Q: Transpondo $AR^{-1} = Q$ obtemos $Q' = R^{-t}A'$, donde

$$y = A^{-1}x = (QR)^{-1}x = R^{-1}Q'x = R^{-1}R^{-t}A'x \ .$$

Exercícios

1. Use as propriedades fundamentais do produto interno para provar:

   (a) A **desigualdade de Cauchy-Scwartz**: $| < x \mid y > | \leq \|x\|\|y\|$. Sugestão: Calcule $\|x - \alpha y\|^2$ para $\alpha =< x \mid y >^2 / \|y\|$.

   (b) A **Desigualdade Triangular**: $\|x + y\| \leq \|x\| + \|y\|$.

   (c) Em que caso temos igualdade na desigualdade de Cauchy-Schwartz? Relacione sua resposta com a definição de ângulo entre vetores.

2. Use a definição do produto interno em $\Re^n$ para provar a Lei do Paralelogramo: $\|x + y\|^2 + \|x - y\|^2 = 2\|x\|^2 + 2\|y\|^2$.

3. Uma matriz P é **idempotente**, ou um projetor não ortogonal, sse $P^2 = P$. Se P é idempotente prove que:

   (a) $R = (I - P)$ é idenpotente.

   (b) $\Re^n = C(P) + C(R)$.

   (c) Todos os autovalores de P são 0 ou $+1$. Sugestão: Mostre que se 0 é uma raíz do polinômio característico de P, $\varphi_P(\lambda) \equiv \det(P - \lambda I)$, então $(1 - \lambda) = 1$ é raíz de $\varphi_R(\lambda)$.

4. Prove que $\forall P$ idempotente e simétrico, $P = P_{C(P)}$. Sugestão: Mostre que $P'(I - P) = 0$.

5. Prove que o operador de projeção num dado sub-espaço vetorial V, P_V, é único e simétrico.

6. Prove o teorema de Pitágoras: $\forall b \in \Re^m, u \in V$ temos que $\|b - u\|^2 = \|b - P_V b\|^2 + \|P_V b - u\|^2$.

7. Formule o problema de quadrados mínimos como um problema de programação quadrática.

   (a) Assuma dada uma base N de $N(A')$.

   (b) Calcule diretamente o resíduo, $z = b - y$, em função de A.

8. O **traço** de uma matriz A é definido por $tr(A) \equiv \sum_i A_i^i$. Mostre que

   (a) Se A, $m \times n$, tem posto pleno $\rho(A) = n$, então $tr(P_A) = n$.

   (b) Nas condições do item anterior, definindo $R_A = (I - P_A)$, temos que $tr(R_A) = m - n$.

9. Método de Hauseholder: A **reflexão** definida por um vetor unitário $u \mid u'u = 1$, é a transformação linear $H = I - 2uu'$.

   (a) Interprete geometricamente a opereação de reflexão.

   (b) Prove que $H = H'$, $H' = H^{-1}$, e $H^2 = I$.

(c) Dado $x \neq 0$ um vetor em R^n, tome

$$v = x \pm \|x\| \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad , \quad u = v/\|v\| \ \text{ e } \quad H = I - 2uu' \ .$$

Mostre que $(Hx)_1 = \|x\|$, e que todas as demais componentes de Hx se anulam.

(d) Discuta como poderiamos usar uma série de reflexões para obter a fatoração QR de uma matriz.

Capítulo 6

ELIMINAÇÃO SIMÉTRICA

Esparsidade na Fatoração de Cholesky.

6.1 Grafos de Eliminação

Na eliminação assimétrica, estudada no capítulo 4, procuramos uma permutação geral, $QPAQ'$ que minimizasse o preenchimento durante a fatoração $QPAQ' = LU$. No caso simétrico queremos preservar a simetria da matriz, e restringir-nos-emos a permutações simétricas, $QAQ' = A_{q(i)}^{q(j)} = LL'$.

Exemplo 1:

Neste exemplo mostramos as posições preenchidas na fatoração de Cholesky de uma matriz simétrica A, bem como o preenchimento na fatoração de duas permutações simétricas da mesma matriz:

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \left[\begin{array}{cccccc} 1 & x & x & & & x \\ x & 2 & x & & & 0 \\ x & x & 3 & x & & 0 \\ & & x & 4 & & 0 \\ & & & & 5 & x \\ x & 0 & 0 & 0 & x & 6 \end{array}\right] \left[\begin{array}{cccccc} 1 & x & x & x & & \\ x & 3 & 0 & x & x & \\ x & 0 & 6 & 0 & 0 & x \\ x & x & 0 & 2 & 0 & 0 \\ & x & 0 & 0 & 4 & 0 \\ & & x & 0 & 0 & 5 \end{array}\right] \left[\begin{array}{cccccc} 5 & & & x & & \\ & 4 & & & x & \\ & & 2 & & x & x \\ x & & & 6 & & x \\ & x & x & & 3 & x \\ & & x & x & x & 1 \end{array}\right]$$

Assim como no capítulo 4 a linguagem de teoria de grafos foi útil para descrever o processo de eliminação simétrica, usaremos agora grafos simétricos para estudar a eliminação simétrica. O primeiro destes conceito a ser definido é o de **grafos de eliminação**, que nada mais são que os grafos que tem por matriz de adjacência as submatrizes ${}^{k}A$ do processo de fatoração (veja capítulo 2): Dado um grafo simétrico $G = (N, E)$, $N = \{1, 2, \ldots n\}$, e uma ordem de eliminação $q = [\sigma(1), \ldots \sigma(n)]$, onde σ é uma permutação de $[1, 2, \ldots n]$, definimos o processo de eliminação dos vértices de G na ordem q como a seqüência de grafos de eliminação $G_i = (N_i, E_i)$ onde, para

$i = 1 \dots n$,

$$N_i = \{q(i), q(i+1), \dots q(n)\}, \quad E_1 = E, \quad \text{e, para } i > 1 \ ,$$

$$\{a,b\} \in E_i \Leftrightarrow \ \{a,b\} \in E_{i-1} \ \text{ ou } \ \{q(i-1), a\}, \ \{q(i-1), b\} \in E_{i-1}$$

Definimos também o inverso ordem de eliminação, $\bar{q}(i) = k \Leftrightarrow q(k) = i$, significando que i foi o k-ésimo vértice de G a ser eliminado.

O **grafo preenchido** é o grafo $P = (N, F)$, onde $F = \cup_1^n E_i$. Os lados originais e os lados preenchidos em F são, respectivamente, os lados em E e em $F - E$.

Observação 6.1 Ao eliminar a j-ésima coluna na fatoração de Cholesky da matriz $QAQ' = A_{q(i)}^{q(j)} = LL'$ preenchemos exatamente as posições correspondentes aos lados preenchidos em F quando da eliminação do vértice $q(j)$.

Exemplo 2:

Os grafos de eliminação e o grafo preenchido correspondentes à segunda ordem de eliminação do exemplo 1, $q = [1, 3, 6, 2, 4, 5]$, são:

$$\begin{array}{cccc} \mathbf{1} & - & 3 & \\ | & \times & & \backslash \\ 2 & & 6 & | \\ & / & & / \\ 5 & & 4 & \end{array} \quad \begin{array}{cccc} & & \mathbf{3} & \\ & / & | & \backslash \\ 2 & - & 6 & | \\ & / & & / \\ 5 & & 4 & \end{array} \quad \begin{array}{ccc} 2 & - & \mathbf{6} \\ & \times & | \\ 5 & & 4 \end{array} \quad \begin{array}{ccc} \mathbf{2} & & \\ | & \backslash & \\ 5 & - & 4 \end{array} \quad \begin{array}{ccc} 5 & - & \mathbf{4} \end{array} \quad \begin{array}{cccc} 1 & - & 3 & \\ | & \times & | & \backslash \\ 2 & - & 6 & | \\ | & \times & | & / \\ 5 & - & 4 & \end{array}$$

Lema 6.1 (Parter) *Se $f = \{i,j\} \in F$, então ou f é um lado original, i.é. $F \in E$, ou f foi preenchido quando da eliminação de um vértice $k \mid \bar{q}(k) < \min\{\bar{q}(i), \bar{q}(j)\}$.*

Lema 6.2 (do caminho) *Considere a eliminação dos vértices de $G = (N, E)$ na ordem q. Temos que $\{i,j\} \in F$ sse existe um caminho de i a j em G, passando apenas por vértices eliminados antes de i ou j, i.e., $\exists C = (i, v_1, v_2, \dots v_p, j)$, em G, com $\bar{q}(1) \dots \bar{q}(p) < \min\{\bar{q}(i), \bar{q}(j)\}$.*

Demonstração:

$\Leftarrow$: trivial.

$\Rightarrow$: por aplicação recursiva do lema de Parter.

Dada a matriz A, $G = (N, E) = (N, B(A))$, a ordem de eliminação q, e o respectivo grafo preenchido, consideremos o conjunto de índices de linha de ENNs na coluna j do fator de Cholesky, $L^j \mid QAQ' = LL'$:

$$enn(L^j) = \{i \mid i > j \wedge \{q(i), q(j)\} \in F\} + \{j\} \ .$$

Definimos a **árvore de eliminação**, H, por

$$\Gamma_H^{-1}(j) = \left\{ \begin{array}{l} j, \quad \text{se} \quad enn(L^j) = \{j\}, \quad \text{ou} \\ \min\{i > j \mid i \in enn(L^j)\} \end{array} \right. , \quad \text{caso contrário}$$

Para não sobrecarregar a notação usaremos $h() = \Gamma_H^{-1}(\ )$ e $g() = \Gamma_H(\ )$. Assim $h(j)$, o pai de j em H, é simplesmente o primeiro ENN (não diagonal) na coluna j de L.

Exemplo 3: As as árvores de eliminação correspondentes ao exemplo 1 são:

$$\begin{array}{ccccccccc} 6 & \rightarrow & 5 & & & & & & \\ & \searrow & 4 & \rightarrow & 3 & \rightarrow & 2 & \rightarrow & 1 \end{array} \ , \quad \begin{array}{ccccc} 6 & \rightarrow & 5 & \rightarrow & 4 \\ & & & & \downarrow \\ 1 & \leftarrow & 2 & \leftarrow & 3 \end{array} \ , \quad \begin{array}{ccccccc} & & & & \nearrow & 2 \\ 6 & \rightarrow & 5 & \rightarrow & 3 \\ & \searrow & 4 & \rightarrow & 1 \end{array} \ .$$

Teorema 6.1 (da árvore de eliminação) *Dado $i > j$,*

$$i \in enn(L^j) \Rightarrow j \in \bar{g}(i) \ ,$$

i.e., qualquer índice de linha abaixo da diagonal na coluna j de L é um ascendente de de j na árvore de eliminação.

Demonstração:

$$\begin{array}{ccccccccccc} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & j & & & & & & & & \\ & & \vdots & \ddots & & & & & & & \\ & & x & \ldots & k & & & & & & \\ & & & & & \ddots & & & & & \\ & & & & & & l & & & & \\ & & & & & & \vdots & \ddots & & & \\ & & \bullet & & \bullet & & x & \ldots & i & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & n \end{array}$$

Se $i = h(j)$ o resultado é trivial. Caso contrário (vide figura 1) seja $k = h(j)$. Mas $L_i^j \neq 0 \wedge L_k^j \neq 0 \Rightarrow L_i^k \neq 0$, pois $\{q(j), q(i)\}, \{q(j), q(k)\} \in G_j \Rightarrow \{q(k), q(i)\} \in G_{j+1}$. Agora, ou $i = h(k)$, ou aplicamos o argumento recursivamente, reconstruindo o ramo de H, $(i, l, \ldots k, j)$, $i > l > \ldots > k > j$. QED.

Pela demonstração do teorema vemos que a árvore de eliminação retrata as dependências entre as colunas para o processo de fatoração numérica da matriz. Mais exatamente, podemos eliminar a coluna j de A (i.e. calcular todos os multiplicadores na coluna j, denotados por M^j no capítulo 2, e atualizar os elementos afetados por estes multiplicadores) sse já tivermos eliminado todos os descendentes de j na árvore de eliminação.

Se pudermos realizar processamento paralelo (veja capítulo 8), podemos eliminar simultaneamente todas as colunas em um mesmo nível da árvore de eliminação, começando pelas folhas, e terminando por eliminar a raiz.

Exemplo 4:

Consideremos a eliminação de uma matriz com o mesmo padrão de esparsidade da última permutação do exemplo 1. Sua árvore de eliminação é a última apresentada no exemplo 3. Esta árvore tem 3 níveis que, das folhas para a raiz são: $\{1,3,2\}$, $\{4,5\}$, e $\{6\}$. Assim, podemos fatorar uma matriz com este padrão de esparsidade em apenas 3 etapas, como ilustrado no exemplo numérico seguinte:

$$\begin{bmatrix} \mathbf{1} & & & 7 & & \\ & \mathbf{2} & & & 8 & \\ & & \mathbf{3} & & 6 & 9 \\ 7 & & & 53 & & 2 \\ & 8 & 6 & & 49 & 23 \\ & & 9 & 2 & 23 & 39 \end{bmatrix} \begin{bmatrix} 1 & & & 7 & & \\ & 2 & & & 8 & \\ & & 3 & & 6 & 9 \\ \mathit{7} & & & \mathbf{4} & & 2 \\ & \mathit{4} & \mathit{2} & & \mathbf{5} & 5 \\ & & \mathit{3} & 2 & 5 & 12 \end{bmatrix} \begin{bmatrix} 1 & & & 7 & & \\ & 2 & & & 8 & \\ & & 3 & & 6 & 9 \\ \mathit{7} & & & 4 & & 2 \\ & \mathit{4} & \mathit{2} & & 5 & 5 \\ & & \mathit{3} & \mathit{\frac{1}{2}} & \mathit{1} & \mathbf{6} \end{bmatrix}$$

O próximo teorema mostra uma forma computacionalmente mais eficiente de obter o grafo preenchido, $P = (N, F)$, e a árvore de eliminação, H, dado o padrão de esparsidade da matriz original, $G = (N, E)$, e a ordem de eliminação, q. Nesta versão simplificada dos grafos de eliminação, G^*_j, ao eliminarmos o vértice $q(j)$, preenchemos apenas os lados incidentes ao seu vizinho mais próximo de ser eliminado.

Teorema 6.2 (da fatoração simbólica) *Definimos agora a versão simplificada do processo de eliminação:*

$$\begin{aligned} G^*_j &= (N_j, E^*_j), \\ E^*_1 &= E, \\ h^*(j) &= \min\{i > j \mid \{q(j), q(i)\} \in E^*_j\} \\ E^*_{j+1} &= \{\{a,b\} \in E^*_j \mid \bar{q}(a), \bar{q}(b) > j\} \\ & \quad \cup \ \{\{q(h(j)), v\},\ v \mid \bar{q}(v) \geq j \wedge \{q(j), v\} \in E^*_j\} \\ F^* &= \cup_1^n E^*_j \end{aligned}$$

Afirmamos que o grafo preenchido e a árvore de eliminação obtidos no processo simplificado coincidem com a definição anterior, i.e., $F^* = F$ e $h^* = h$.

Exemplo 5: Os grafos de eliminação simplificados e o grafo preenchido referentes a segunda ordem de eliminação no exemplo 1, $q = [1, 3, 6, 2, 4, 5]$, são:

$$
\begin{array}{cccc}
\mathbf{1} & - & \mathit{3} & \\
| & \times & & \backslash \\
2 & & 6 & | \\
& / & & / \\
5 & & 4 &
\end{array}
\quad
\begin{array}{cccc}
 & & \mathbf{3} & \\
 & / & | & \backslash \\
2 & & \mathit{6} & | \\
 & / & & / \\
5 & & 4 &
\end{array}
\quad
\begin{array}{cccccc}
\mathit{2} & - & \mathbf{6} & \mathbf{2} & & \\
 & / & | & | & \backslash & \\
5 & & 4 & 5 & & \mathit{4}
\end{array}
\quad
\begin{array}{ccc}
5 & - & \mathbf{4}
\end{array}
\quad
\begin{array}{ccccc}
1 & - & 3 & & \\
| & \times & | & \backslash & \\
2 & - & 6 & & | \\
| & \times & | & / & \\
5 & - & 4 & &
\end{array}
$$

O lema seguinte demonstra o teorema da fatoração simbólica:

Lema 6.3

$$enn(L^j) = \cup_{k\in g(j)} enn(L^k) \ \cup \ enn(A^j) - J + \{j\} \quad , \quad J = \{1, 2, \ldots j\} \ .$$

Demonstração:

$\supseteq$:

$$
\begin{array}{cccccccc}
1 & & & & & & & \\
 & \ddots & & & & & & \\
 & & k & - & x & & & \\
 & & \vdots & \ddots & | & & & \\
 & & x & \ldots & j & & & \\
 & & & & & \ddots & & \\
 & & \bullet & & \bullet & & \bullet & \\
 & & & & & & & \ddots \\
 & & & & & & & & n
\end{array}
$$

Consideremos $k \in g(j)$ (vide figura 2). Se $i > j \in enn(L^k)$, então se L^j_i já não é um ENN, será preenchido ao eliminarmos L^k.

$\subseteq$:

$$
\begin{array}{ccccccccccc}
1 & & & & & & & & & & \\
 & \ddots & & & & & & & & & \\
 & & l & & & - & & & \bullet & & \\
 & & \vdots & \ddots & & & & & & & \\
 & & x & \ldots & h(l) & & & & & & \\
 & & & & \vdots & \ddots & & & | & & \\
 & & & & x & \ldots & h^2(l) & & & & \\
 & & & & & & & \ddots & & & \\
 - & & - & & & - & & - & j & & \\
 & & \vdots & \mapsto & \vdots & \mapsto & \vdots & \ldots & \vdots & \ddots & \\
 & & & & & & & & & & n
\end{array}
$$

Seja $l < j \mid j \in enn(L^l)$, i.e., consideremos uma coluna e cuja eliminação poderia causar preenchimentos na coluna j (vide figura 3). Pelo teorema da árvore de eliminação, j é um ascendente de l, i.e., $\exists p \leq n \mid j = h^{p+1}(l)$. Mas pela primeira parte da prova,

$$enn(L^l) - J \subseteq enn(L^{h(l)}) - J \subseteq \ldots \subseteq enn(L^{h^p(j)}) \subseteq enn(L^j) - \{j\} \ ,$$

de modo que qualquer vértice que poderia causar, durante a eliminação da coluna l, uma preenchimento na coluna j, deve necessariamente aparecer em $h^p(l) \in g(j)$. QED.

6.2 Grafos Cordais

Em um dado grafo simétrico, $G = (N, E)$, definimos os seguintes termos: $G = (N, E)$ é **cordal** sse para qualquer ciclo $C = (v_1, v_2, \ldots v_p, v_1)$, $p \geq 3$, existe uma corda, i.e., um lado $e \in E$ ligando dois vértices não consecutivos em C. Um vértice é **simplicial** sse sua eliminação não causa preenchimento. Uma ordem de eliminação, q, é **perfeita** sse a eliminação de nenhum dos vértices, na ordem q, causa preenchimento. Um subconjunto dos vértices, $S \subset N$, é um *separador* entre os vértices a e b, sse a remoção de S deixa a e b em componentes conexas distintas. Dizemos que $S \subset N$ é um **separador** entre $A \subset N$ e $B \subset N$ sse, $\forall a \in A, \ b \in B$, S separa a de b. Um conjunto, C, satisfazendo uma propriedade, P, é *minimal* (em relação a P) sse nenhum subconjunto próprio de C satisfaz P. Analogamente, um conjunto é **maximal**, em relação a P, sse nenhum conjunto que o contenha propriamente satisfaz P. Um **clique** é um conjunto de vértices $C \subset N$ onde todos os vértices são adjacentes, i.e., $\forall i, j \in C, \ \{i, j\} \in E$. indexClique

Exemplo 6:

No 1o grafo do exemplo 2 vemos que 5 e 6 são vértices simpliciais, $q = [5, 4, 2, 6, 3, 1]$ é uma ordem de eliminação perfeita. No último grafo do exemplo 2 vemos que $S = \{2, 3, 6, 4\}$ é um separador entre os vértices $a = 1$ e $b = 5$ (não minimal), $S' = S - \{3\}$ é um separador minimal, e $C = \{1, 2, 3, 6\}$ é um clique.

Teorema 6.3 (da caracterização de grafos cordais) *Dado $G = (N, E)$, as três propriedades seguintes são equivalentes:*

1. *Existe em G uma ordem de eliminação perfeita.*

2. *G é cordal.*

3. *Se S é um separador minimal entre $a, b \in N$, então S é um clique.*

Demonstração:

$1 \Rightarrow 2$:

Seja $q = [\sigma(1), \sigma(2), \ldots \sigma(n)]$ uma ordem de eliminação perfeita em G, e seja $C = (v_1, v_2, \ldots v_p, v_1)$ um ciclo em G. Consideremos v_k o primeiro vértice de C a ser eliminado, $k = arg\min_{1 \leq i \leq p}\{\bar{q}(v_i)\}$. Como q é uma ordem de eliminação perfeita, a corda $\{v_{k-1}, v_{k+1}\} \in E$.

$2 \Rightarrow 3$:

Seja S um separador minimal entre A e B, $a \in A$, $b \in B$, $v, w \in S$. Como S é minimal, existe um caminho $C = (a, c_1, \ldots c_p, v) \mid c_1, \ldots c_p \in A$, pois, caso contrário, $S - v$ continuaria separando a de b. Analogamente existe um caminho D conectando a a w, com os vértices intermediários todos em A. Existe pois um caminho de v a w, cujos vértices intermediários estão todos em A. Seja P um tal caminho de comprimento mínimo, $P = (v, a_1, \ldots a_p, w)$. Analogamente seja $Q = (w, b_1, \ldots b_q, v)$ um caminho de w a v através de B com comprimento mínimo.

Concatenando P e Q obtemos um ciclo $R = (v, a_1, \ldots a_p, w, b_1, \ldots b_q, v)$. Como G é cordal, R contém ao menos uma corda, $\{x, y\}$. Como S é um separador, não podemos ter $x \in A$ e $y \in B$. Como P tem comprimento mínimo, não podemos ter $x, y \in P$ e, analogamente não podemos ter $x, y \in Q$. Assim, $\{v, w\}$ é a única corda possível em R, e concluímos que $\forall v, w \in S, \ \{v, w\} \in G$.

Antes de $3 \Rightarrow 1$, provemos 2 lemas auxiliares:

Lema 6.4 (hereditariedade) *Se G satisfaz a propriedade 3, então qualquer subgrafo de G induzido por um subconjunto de vértices também a satisfaz.*

Demonstração:

Seja $\tilde{G} = (\tilde{N}, \tilde{E})$ o subgrafo de G induzido por $\tilde{N} \subset N$, e $\tilde{S}$ minimal em $\tilde{G}$ separando a de b. Podemos, em G, completar $\tilde{S}$, com vértices de $N - \tilde{N}$, de modo a obter S, um separador minimal em G entre a e b. Mas se G satisfaz a propriedade 3, S é um clique, e como $\tilde{S}$ é um subgrafo de S, $\tilde{S}$ também é um clique.

Lema 6.5 (Gavril) *Se G satisfaz a propriedade 3 então ou G é completo, ou G tem ao menos dois vértices simpliciais não adjacentes.*

Demonstração:

Provemos o lema por indução no número de vértices de G, n. Para $n = 1$ o lema é trivial. Se $G = (N, E)$, $n > 1$, não é completo, $\exists a, b \in G \mid \{a, b\} \notin E$.

Seja S um separador minimal entre a e b, partindo $G - S$ em ao menos 2 componentes conexas distintas, A e B, $a \in A$, $b \in B$. $G(S \cup A)$, o subgrafo induzido pelos vértices de $S \cup A$, hereditariamente satisfaz a propriedade 3, e pela hipótese de indução, ou $G(S \cup A)$ é um clique, ou contém (ao menos) 2 vértices simpliciais não adjacentes.

Se $G(S \cup A)$ não for completo, então um dos seus 2 vértices simpliciais não adjacentes deve estar em A (pois S é um clique). Se $G(S \cup A)$ é completo, então qualquer vértice de A é simplicial (pois S é um separador). Logo, existe ao menos um vértice simplicial, $v \in A$. Analogamente, existe um vértice simplicial $w \in B$, e v não é adjacente a w (pois S separa v de w).

$3 \Rightarrow 1$:

Seja $G = (N, E)$ satisfazendo a propriedade 3. Se G é completo, qualquer ordem q é perfeita. Caso contrário, existem 2 vértices simpliciais não adjacentes. Podemos pois eliminar um deles, $q(1)$, sem preenchimento, e pelo lema da hereditariedade o subgrafo resultante continua satisfazendo a propriedade 3. Podemos então usar o argumento recursivamente para obter uma ordem de eliminação perfeita. QED.

6.3 Ordenações por Dissecção

Consideremos em $G = (N, E)$ um separador S que parte $N - S$ com componentes conexas $N_1, N_2, \ldots, N_k$. Em cada uma desta componentes podemos considerar um novo separador que a reparte, e assim recursivamente. Podemos representar este processo pela **árvore de dissecção**, D, onde S é a raiz, uma componente separada por S, ou o separador dentro dela, é filha de S, e as componentes não repartidas são as folhas da árvore.

Uma **pós-ordem** dos vértices de uma árvore H de raiz r é uma ordem q, dos vértices de H, que lista os vértices de cada uma das árvores de $H - r$, recursivamente em pós-ordem, e, finalmente, r. Seja $\tilde{q}$ uma pós-ordem numa árvore de dissecção, D, de G. Substituindo em $\tilde{q}$ cada nó, d, de D pelos vértices de G em d, temos uma **ordem de dissecção**, q.

Lema 6.6 (Dissecção) *Consideremos a eliminação dos vértices de G na ordem de dissecção q. A eliminação de um dado vérice, $v \in d$, só pode preencher lados dentro do seu nó em D, d, ou entre (vértices de) d e (vértices em) nós ancestrais de d (em D), ou ainda entre (vértices em) ascendente de d (em D).*

Demonstração: Trivial, pelo lema do caminho.

Exemplo 7:

Vejamos um exemplo de eliminação usando uma ordem de dissecção

```
           -  -  -  1      2                      {1,2}
         /        /  |      |  \                 /    |     \
 3       |     4     5      6     7     {3,4}        {5,6}       {7}
 |       |     |     |  \   |  \          |            |     \
 8   -   9  -  10    11     12    13   {8,9,10}     {11,12}      {13}
```

1	9	x	x									x	
2	x	8	0	x								0	
3	x	0	10	0	x							0	
4		x	0	3	0							0	
5			x	0	4							x	
6						11			x				
7							12		x	x			
8								13		x			
9						x	x		5	0		x	
10							x	x	0	6		0	x
11											7		x
12	x	0	0	0	x				x	0		1	0
13										x	x	0	2

O último conjunto da partição, S_k ou a raiz de $\tilde{G}$, é em, G, um separador entre cada uma das componentes de $\tilde{G} - k$. Para minimizar a região de possível preenchimento (em G ou QAQ') gostaríamos de ter o separador S_k "pequeno e balanceado", i.e. tal que

- $\#S_k$ seja o menor possível.
- Sub-árvores tenham aproximadamente o mesmo número de vértices de G.

Recursivamente, gostaríamos de ter como raiz de cada uma das sub-árvores um bom separador, i.e., pequeno e balanceado. Veremos a seguir várias heurísticas para obter num grafo qualquer, G, um bom separador.

Heurística de **Busca em Largura**:

Uma busca em largura, BEL, a partir uma raiz $v \in N$, particiona os vértices de G em níveis $L_0, L_1, \ldots L_k$, definidos por

$$L_0 = \{v\} \ , \quad L_{i+1} = adj(L_i) - L_{i-1} \ .$$

A **profundidade** do nível L_i é i, e a **largura** do nível L_i é $\#L_i$. A profundidade e a largura da BEL são, respectivamente, a máxima profundidade e a máxima largura nos níveis da BEL.

Lema 6.7 *O nível L_i separa, em G, os vértices em níveis mais profundos dos em níveis menos profundos que i.*

A heurística de BEL procura um separador balanceado $S \subseteq L_i$, tomando $i \approx k/2$, ou então tomando $i \mid \sum_1^{i-1} \#L_j < n/2 \ \wedge \ \sum_{i+1}^{n} \#L_j < n/2$.

Para obter um separador pequeno a heurística procura uma raiz, v, que gere uma BEL de máxima profundidade, com o intuito de reduzir a largura da BEL. A **distância** (em G) de um vértice v a um vértice w, $dist(v,w)$, é o comprimento, ou número de lados, do caminho mais curto entre ambos os vértices. A **excentricidade** de um vértice v é $exc(v) = \max_{w \in N} dist(v,w)$. Um vértice de máxima excentricidade se diz **periférico**, e sua excentricidade é o **diâmetro** de G.

Uma BEL com raiz v terá profundidade igual a excentricidade de v. Isto motiva querermos iniciar a BEL por um vértice periférico. Encontrar um vértice periférico é um problema computacionalmente difícil. A **heurística de Gibbs** encontra um vértice **quase-periférico** como segue:

1. Escolha como raiz um vértice de grau mínimo.

2. Forme os níveis da BEL com raiz v, $L_1 \ldots L_k$. Particione o nível mais profundo em suas componentes conexas, $L_k = \cup_1^l S_j$, e tome um vértice de grau mínimo, v_j, em cada componente.

3. Para $j = 1 : l$

   - Tome v_j como nova raiz e encontre os níveis da BEL, $L_1 \ldots L_{k'}$

   Ate que $k' > k$ ou $j = l$.

4. Se o passo 3 terminou com $k' > k$, volte ao passo 2. Caso contrário a atual raiz é um vértice quase-periférico.

Exemplo 8:

Retomando o exemplo 7, a heuristica de Gibbs encontra 3 como vértice quase-periférico. Tomando 3 como raiz geramos a árvore H por BEL:

$$
\begin{array}{ccccccccccccccccccc}
 & & & & & & & 10 & - & 4 & & & & & & 13 & & \\
 & & & & & & / & & & & & & & & / & & & \\
H = & 3 & - & 8 & - & 9 & - & 1 & - & 5 & - & 12 & - & 6 & - & 2 & - & 7 \\
 & & & & & & & & & & & & & & & & & \\
L = & 1 & & 2 & & 3 & & 4 & & 5 & & 6 & & 7 & & 8 & & 9
\end{array}
$$

Escolhendo $S_1 = L_5$ como primeiro separador, e depois $S_2 = L_3$ e $S_3 = L_7$ como separadores dentro de cada uma das componetes separadas por S_1, obtemos a odem por dissecção $q = [3, 8, 1, 10, 9, 11, 12, 2, 13, 7, 6, 4, 5]$.

$$
\begin{array}{r|ccccccccccccc}
1 & 3 & x & & & & & & & & & & & \\
2 & x & 8 & & & x & & & & & & & & \\
3 & & & 1 & & & & & & & & & x & x \\
4 & & & & 10 & x & & & & & & & x & \\
5 & & x & & x & \mathbf{9} & & & & & & & 0 & \\
6 & & & & & & 11 & & & & & & & x \\
7 & & & & & & & 12 & & & & x & & x \\
8 & & & & & & & & 2 & & x & x & & \\
9 & & & & & & & & & 13 & & x & & \\
10 & & & & & & & & x & & 7 & 0 & & \\
11 & & & & & & & x & x & x & 0 & \mathbf{6} & & 0 \\
12 & & & x & x & 0 & & & & & & & \mathbf{4} & 0 \\
13 & & & x & & & x & x & & & & 0 & 0 & \mathbf{5}
\end{array}
$$

Note que nas linhas (colunas) correspondentes aos vértices do primeiro separador, $S_1 = \{1\}$, pode haver ENN's em qualquer posição. Note também que o resto da matriz está em forma diagonal blocada (vide definição no capítulo 8), onde cada bloco correspnde a uma das componentes separadas por S_1. Esta estrutura se repete em cada bloco, formando a estrutura "espinha de peixe" característica de ordens por dissecção. Note que esta estrutura é preservada pela fatoração de Cholesky.

Exercícios

1. Implemente a eliminação simbólica num grafo G, com a ordem q, computando F e H, em tempo $O(\#enn(L))$.

2. Quão eficiente (k em tempo $O(n^k)$) poderíamos implementar o algoritmo implícito na última parte do teorema de caracterização de grafos cordais?

3. Considere o seguinte algoritmo para numerar os vértices de um grafo na ordem $n, n-1, \ldots 2, 1$:

   Ordenamento Reverso por Grau Máximo (ORGM):

   (a) Escolha, como vértice n, um vértice qualquer.

   (b) Escolha, como vértice seguinte um vértice ainda não numerado adjacente a um número máximo de vértices já numerados

   Prove que ORGM define uma ordem perfeita.

4. Quão eficientemente podemos implementar o ORGM?

5. De um exemplo de vértice quase-periférico que não seja periférico.

Capítulo 7

ESTRUTURA

Acoplamento de Sub-Sistemas

Estruturas Blocadas

Consideraremos neste capítulo matrizes com blocos de elementos nulos dispostos de forma regular. Estudaremos duas estruturas: a **triangular-blocada** superior, e a **angular blocada** por colunas. O bloco ${}^{r}_{s}B$ tem dimensão $m(r) \times n(s)$, e na estrutura triangular blocada $m(k) = n(k)$.

$$
\begin{bmatrix}
{}^{1}_{1}B & {}^{2}_{1}B & {}^{3}_{1}B & \dots & {}^{h}_{1}B \\
0 & {}^{2}_{2}B & {}^{3}_{2}B & \dots & {}^{h}_{2}B \\
0 & 0 & {}^{3}_{3}B & \dots & {}^{h}_{3}B \\
\vdots & & & \ddots & \vdots \\
0 & 0 & 0 & \dots & {}^{h}_{h}B
\end{bmatrix}
,
\begin{bmatrix}
{}^{1}_{1}B & 0 & \dots & 0 & {}^{h}_{1}B \\
0 & {}^{2}_{2}B & & 0 & {}^{h}_{2}B \\
\vdots & & \ddots & & \vdots \\
0 & 0 & & {}^{h-1}_{h-1}B & \\
0 & 0 & \dots & 0 & {}^{h}_{h}B
\end{bmatrix}
$$

Estas estruturas blocadas propiciam grandes facilidades computacionais. Em particular triangularizá-las corresponde a triangularizar os blocos diagonais, sendo que nenhum elemento não nulo é criado, durante a triangularização, nos blocos nulos.

7.1 Estrutura Triangular Blocada

Como já visto no estudo de matrizes de permutação, dado $G = (N, B)$, $N = \{1, \dots n\}$, B, $n \times n$, sua matriz de adjacência, e um reordenamento de seus vértices, $q = [q_1, \dots q_n]$, $q_i \in N$, então a matriz de adjacência do grafo G com os vértices reordenados (i.e., reindexados) por q é $\tilde{B} = B^{q(j)}_{q(i)} = QBQ'$.

Lema 7.1 *Considere o reordenamento coerente dos vértices de $G = (N, B)$ numa ordem coerente $q = [{}^1q, {}^2q \ldots {}^hq]$. Conforme a definição de ordem coerente, cada bloco, ${}^kq = [{}^kq_1, \ldots {}^kq_{n(k)}]$, contém os vértices de uma CFC (componente fortemente conexa) de G, v_k, e $(v_1, \ldots v_h)$ estão topologicamente ordenados. Neste caso a matriz de adjacência do grafo reordenado, $\tilde{B} = QBQ'$, é triangular blocada superior, de blocos ${}^r_sB$, $n(r) \times n(s)$.*

Uma matriz que não possa ser, por permutações de linhas e de colunas, reduzida, isto é, posta na forma triangular-blocada, é dita **irredutível**.

Dada uma matriz A, não singular, procuraremos permutações de linhas e colunas, isto é, por matrizes de permutação R e Q, encontrar $\tilde{A} = RAQ$ que seja a "mais fina" partição possível de A. Observemos que a hipótese de não singularidade de A implica na não singularidade dos blocos diagonais, pois

$$det(A) = det(\tilde{A}) = \prod_{k=1}^{h} det({}^k_k\tilde{A}) \ .$$

Sejam R e Q matrizes de permutação e seja $P = QR$,

$$\tilde{A} = RAQ = Q^{-1}QRAQ = Q'PAQ$$

ou seja, podemos escrever qualquer transformação do tipo RAQ como uma permutação de linhas, PA, seguida de uma permutação simétrica $Q'(PA)Q$.

Consideremos agora o grafo associado à matriz PA, $G(PA)$, que tem por matriz de adjacência o padrão de esparsidade de PA, $B(PA)$, i.e.,

$$G(PA) = (N, B(PA)) = (N, \Gamma), \ j \in \Gamma(i) \Leftrightarrow (PA)_i^j \neq 0 \ .$$

Do último lema sabemos que a permutação simétrica que dá a mais fina partição de PA é um ordenamento coerente dos vértices de $G(PA)$. Ademais, PA é irredutível se $G(PA)$ é fortemente conexo. Resta, portanto, analisar o papel da permutação de linhas, P, na permutação geral de linhas e colunas de $\tilde{A} = Q'PAQ$, onde a permutação simétrica é dada por um ordenamento coerente em $G(PA)$.

Pela não singularidade, A deve possuir ao menos uma diagonal de elementos não nulos (não necessariamente a diagonal principal, veja as definições de determinante e diagonal). Portanto existe uma permutação de linhas, P, que posiciona esta diagonal de elementos não nulos na diagonal principal de PA. Uma tal permutação P será dita uma **permutação própria**, ao passo que uma permutação que coloque algum elemento nulo na diagonal principal será dita **imprópria**. Isto posto, mostraremos que:

Teorema 7.1

1. *Qualquer permutação própria induz a mesma estrutura de partição, isto é, h blocos de dimensões $n(1)\dots n(h)$, com os mesmos índices em cada bloco.*

2. *Qualquer permutação imprópria não pode induzir uma partição mais fina em $\tilde{A}$.*

Demonstração:

Pelo lema de Hoffmann $G(PA)$ é fortemente conexo, i.e., PA é irredutível se qualquer conjunto de $k < n$ linhas tiver elementos não nulos em ao menos $k+1$ colunas. Como esta caracterização independe da ordem das linhas, mostramos que a irredutibilidade da matriz PA independe da permutação própria considerada.

Da mesma forma, a irredutibilidade do bloco sudeste, ${}^{h}_{h}\tilde{A}$ bem como a existência da CFC v_h entre as "últimas" (no sentido da ordem natural do grafo reduzido) componentes associadas a qualquer outra permutação própria, P, está garantida, de modo que a invariância da estrutura de $\tilde{A}$ segue por indução no número de componente fortemente conexas (blocos).

Mostramos agora que se A tiver algum elemento diagonal nulo, então as componentes fortemente conexas de $G(A)$ são fusões de componentes fortemente conexas de $G(PA)$: Distinguamos os zeros na diagonal de A e consideremos o grafo $G^{+}(PA)$ obtido de $G(PA)$ ao adicionarmos as arestas correspondentes aos zeros distinguidos em PA. As CFCs em $G(A)$ são exatamente as CFCs em $G^{+}(PA)$, sendo que as arestas adicionais (se quando adicionadas ao grafo reduzido de $G(PA)$ formarem ciclos) só podem tornar equivalentes alguns vértices antes em CFCs distintas. QED.

A permutação própria, P, pode ser vista como um casamento perfeito entre linhas e colunas, onde casamos a coluna j com a linha $p(j)$, entre seus pretendentes $\Gamma^{-1}(j) = \{i \mid B_i^j \neq 0\}$. Podemos portanto encontrá-la através do algoritmo Húngaro, de preferência com o emprego de alguma heurística eficiente para evitar a freqüente geração de árvores de caminhos de aumento.

Exemplo 1:

Considere as matrizes de adjacência B, sua permutação própria PB, e o posterior ordenamento coerente $Q'PBQ$.

$$B = \begin{bmatrix} \mathbf{0} & \mathbf{1} & 0 \\ 0 & 1 & \mathbf{1} \\ \mathbf{1} & 1 & 1 \end{bmatrix} \quad , \quad PB = B^j_{p(i)} = \begin{bmatrix} \mathbf{1} & 1 & 1 \\ \mathbf{0} & \mathbf{1} & 0 \\ 0 & 1 & \mathbf{1} \end{bmatrix} \quad ,$$

$$Q'PAQ = B^{q(j)}_{p(q(i))} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ \mathbf{0} & 0 & 1 \end{bmatrix} \quad .$$

Observe que $G(B)$ é fortemente conexo. Em B destacamos a diagonal a ser posicionada por P na diagonal principal de PB, e o zero inicialmente na diagonal principal. Em PB destacamos o mesmo zero originalmente na diagonal de B. $G(PB)$ tem 3 CFCs, que voltariam a fundir-se numa só, caso adicionássemos a aresta correspondente ao zero destacado em PB. Neste exemplo tivemos $p = [3, 1, 2]$, $q = [1, 3, 2]$.

O **procedimento P4** (Partição e Pré-Posicionamento de Pivôs) explora esparsidade estrutural, e posteriormente a esparsidade local a cada bloco, como segue:

1. Encontre uma permutação própria, PA, através do algoritmo húngaro complementado com uma heurística eficiente.

2. Encontre, através do algoritmo de Tarjan, um ordenamento coerente $Q'PAQ$.

3. Inverta este último ordenamento, $QPAQ'$, de modo a colocar a matriz na forma triangular blocada inferior. Em seguida aplique a heurística P3 a cada bloco diagonal.

São apresentados três exemplos de aplicação do P4. A disposição original dos ENNs da matriz original, A, corresponde aos sinais $+$, $*$, ou números de 1 a 9 no corpo da matriz. Como prescrito no primeiro passo do P4, primeiramente encontramos uma diagonal não nula, ou equivalentemente uma permutação própria PA. O vetor inverso de índices de linha permutados, $\bar{p}$, é dado à esquerda da numeração original das linhas. Os asteriscos indicam os elementos desta diagonal.

Como prescrito no segundo passo do P4, devemos em seguida encontrar um reordenamento coerente, q, em $G(PA)$. Para tanto aplicamos o algoritmo de Tarjan: Primeiramente fazemos a busca em profundidade canônica em $G(PA)$. Para tanto percorremos as linhas de PA, gerando a primeira floresta de cada exemplo. As raízes desta busca são assinaladas por um acento circunflexo, tanto na floresta como no vetor $\bar{p}$. Os números no corpo da matriz correspondem a ordem de visitação nesta busca. A inversa da ordem de retorno é dada pelo vetor $\bar{b}$.

O algoritmo de Tarjan exige em seguida a busca em profundidade canônica no grafo inverso reordenado por b, $G((BPAB)')$; na verdade basta tomarmos as raízes desta busca na ordem canônica. Percorrendo as colunas da matriz, construímos a segunda floresta em cada exemplo, cujas raízes estão assinaladas por um acento circunflexo, tanto na floresta como no vetor $\bar{b}$. A ordem de visitação nesta segunda busca em profundidade nos dá o reordenamento coerente $QPAQ'$; apresentado explicitamente em cada exemplo para que se reconheça a estrutura triangular superior.

Finalmente, ao final dos exemplos, apresentamos o terceiro passo do P4: a inversão deste ordenamento coerente, $Q'PAQ$, (portanto triangular inferior), com posterior aplicação do P3 a cada bloco diagonal. Neste ponto fica claro que o segundo e o terceiro exemplos diferem apenas pela permutação original da matriz. Neste ponto indicamos também os zeros preenchidos durante a fatoração.

Exemplo 2:

			$\bar{q}$	1	9	4	7	2	3	5	8	6
			$\bar{b}$	$\hat{1}$	9	3	$\hat{7}$	4	2	6	$\hat{8}$	5
$\bar{q} \circ \bar{p}$	$\bar{b} \circ \bar{p}$	$\bar{p}$		1	2	3	4	5	6	7	8	9
7	7	4	1				$*$				6	
2	4	5	2	+				$*$		4		8
6	5	9	3		+					+		$*$
1	1	$\hat{1}$	4	$*$		2			9	+		
8	8	8	5		7						$*$	
5	6	7	6	+		+	5			$*$		
3	2	6	7			+		+	$*$			+
4	3	3	8			$*$		3		+		
9	9	2	9		$*$						+	

$$\begin{array}{ccccccccccccc}
\hat{1} & \rightarrow & 3 & \rightarrow & 5 & \rightarrow & 7 & \rightarrow & 4 & \rightarrow & 8 & \rightarrow & 2 \\
\downarrow & & & & \downarrow & & & & & & & & \\
6 & & & & 9 & & & & & & & &
\end{array}$$

$$\begin{array}{ccccccc}
\hat{1} & & & & & \hat{7} & \hat{8} \\
\downarrow & & & & & & \downarrow \\
4 & \rightarrow & 3 & & & & 9 \\
\downarrow & & \downarrow & & & & \\
2 & & 6 & \rightarrow & 5 & &
\end{array}$$

p/q	1	2	3	4	5	6	7	8	9
1	$*$		+	+	+				
2	+	$*$			+	+			
3		+	$*$	+		+			
4		+		$*$	+				
5	+			+	$*$		+		
6					+	$*$			+
7							$*$	+	
8								$*$	+
9								+	$*$

Exemplo 3:

			$\bar{q}$	1	3	4	2	5	8	7	9	6
			$\bar{b}$	$\hat{1}$	6	2	9	3	$\hat{7}$	5	8	$\hat{4}$
$\bar{q}\circ\bar{p}$	$\bar{b}\circ\bar{p}$	$\bar{p}$		1	2	3	4	5	6	7	8	9
1	1	$\hat{1}$	1	$*$		2	$+$					
9	8	8	2						$+$		$*$	
6	4	9	3							9	$+$	$*$
3	6	2	4		$*$		4		5			
2	9	4	5	$+$		$+$	$*$					
7	5	7	6							$*$		$+$
8	7	6	7						$*$		6	
5	3	5	8			$+$		$*$				8
4	2	3	9		3	$*$	$+$	7				

$$
\begin{array}{ccccccc}
\hat{1} & \to & 3 & \to & 2 & \to & 4 \\
 & & & & \downarrow & & \\
 & & \downarrow & & 6 & \to & 8 \\
 & & & & & & \\
 & & 5 & \to & 9 & \to & 7
\end{array}
\qquad
\begin{array}{cccccc}
\hat{1} & & & & \hat{4} & \hat{7} \\
\downarrow & & & & \downarrow & \downarrow \\
9 & & & & 5 & 8 \\
\downarrow & & & & & \\
6 & & & & & \\
\downarrow & & & & & \\
2 & \to & 3 & & &
\end{array}
$$

p/q	1	2	3	4	5	6	7	8	9
1	$*$	$+$		$+$					
2	$+$	$*$		$+$					
3		$+$	$*$					$+$	
4		$+$	$+$	$*$	$+$				
5				$+$	$*$	$+$			
6						$*$	$+$		$+$
7						$+$	$*$		
8								$*$	$+$
9								$+$	$*$

Exemplo 4:

			$\bar{q}$	6	1	4	5	8	2	7	3	9
			$\bar{b}$	$\hat{6}$	$\hat{1}$	2	5	$\hat{8}$	4	7	3	9
$\bar{q} \circ \bar{p}$	$\bar{b} \circ \bar{p}$	$\bar{p}$		1	2	3	4	5	6	7	8	9
1	1	$\hat{2}$	1		$*$	6			$+$			
8	8	5	2					$*$				4
7	7	7	3	$+$				3		$*$		
3	3	8	4						$+$		$*$	$+$
2	4	6	5		$+$	$+$			$*$			
6	6	$\hat{1}$	6	$*$						2		
9	9	9	7					$+$				$*$
5	5	4	8			$+$	$*$			$+$		
4	2	3	9			$*$	7		8		9	

$$\hat{1} \to 7 \to 5 \to 9$$

$$\begin{tikzcd}
\hat{2} \arrow[r] & 3 \arrow[r] \arrow[d] \arrow[dr] & 4 \\
 & 6 & 8
\end{tikzcd}$$

$$\begin{tikzcd}
\hat{1} \arrow[d] & \hat{6} \arrow[d] & \hat{8} \arrow[d] \\
4 \arrow[d] & 7 & 9 \\
3 \arrow[d] & & \\
2 \arrow[r] & 5 &
\end{tikzcd}$$

p/q	1	2	3	4	5	6	7	8	9
1	$*$	$+$		$+$					
2	$+$	$*$		$+$					
3		$+$	$*$						$+$
4		$+$	$+$	$*$	$+$				
5				$+$	$*$		$+$		
6						$*$	$+$		
7						$+$	$*$	$+$	
8								$*$	$+$
9								$+$	$*$

Exemplos 2 a 4; Passo 3 do P4:

p/q	1	2	3	4	5	6	7	8	9
1	$*$	$+$							
2	$+$	$*$							
3		$+$	$*$						
4	$+$	0		$*$					$+$
5			$+$		$*$		$+$		$+$
6						$*$	$+$		$+$
7				$+$	$+$	$+$	0		$+$
8				$+$	$+$	$+$	0	$*$	0
9					$+$		$+$	$+$	$*$

p/q	1	2	3	4	5	6	7	8	9
1	$*$	$+$							
2	$+$	$*$							
3			$*$	$+$					
4	$+$	0	$+$	$*$					
5				$+$	$*$				$+$
6					$+$	$*$	$+$		$+$
7		$+$				$+$	$*$		0
8							$+$	$*$	$+$
9							$+$	$+$	$*$

p/q	1	2	3	4	5	6	7	8	9
1	$*$	$+$							
2	$+$	$*$							
3		$+$	$*$	$+$					
4			$+$	$*$					
5			$+$	0	$*$				$+$
6					$+$	$*$	$+$		$+$
7	$+$	0				$*$	$+$		0
8							$+$	$*$	$+$
9							$+$	$+$	$*$

7.2 Estrutura Angular Blocada

Suponhamos dada uma matriz quadrada e não singular na forma angular blocada, com blocos diagonais ${}^1B, \ldots {}^hB$, ${}^kB$ $m(k) \times n(k)$, $d(k) \equiv m(k) - n(k) \geq 0$, e os correspondentes blocos nas colunas residuais ${}^1C \ldots {}^hC$, ${}^kC$ $m(k) \times n(h+1)$. Podemos usar rotações de Givens para fatorar cada um dos blocos diagonais, ${}^kB = {}^hQ \begin{bmatrix} {}^kV \\ 0 \end{bmatrix}$, onde ${}^kV$ é triangular superior $n(k) \times n(k)$, e o bloco de zeros é $d(k) \times n(k)$.

$$\begin{bmatrix} {}^1B & & & {}^1C \\ & \ddots & & \vdots \\ & & {}^hB & {}^hC \end{bmatrix} , \quad \begin{bmatrix} {}^1V & & & {}^1W \\ 0 & & & {}^1Z \\ & \ddots & & \vdots \\ & & {}^hV & {}^hW \\ & & 0 & {}^hZ \end{bmatrix} .$$

Para completar a fatoração QR da matriz original, respeitando a estrutura de blocos, permutamos os blocos ${}^kZ$ para as últimas linhas da matriz, formando o bloco quadrado Z de dimensão $\sum_1^h d(k) = n(h+1)$. Finalmente completamos a fatoração QR do bloco sudeste, $Z = QS$.

$$\begin{bmatrix} {}^1V & & & {}^1W \\ & \ddots & & \vdots \\ & & {}^hV & {}^hW \\ & & & {}^1Z \\ & & & \vdots \\ & & & {}^hZ \end{bmatrix} , \quad \begin{bmatrix} {}^1V & & & {}^1W \\ & \ddots & & \vdots \\ & & {}^hV & {}^hW \\ & & & S \end{bmatrix} .$$

Como permutações são apenas um tipo especial de transformações ortogonais, obtivemos o fator triangular da fatoração QR respeitando a estrutura diagonal blocada da matriz original, $B = QU$. A inversa da matriz original seria dada por $B^{-1} = U^{-1}Q'$, mas usando que $Q' = U^{-t}B'$ temos $B^{-1} = U^{-1}U^{-t}B'$. Isto é, obtivemos uma fatoração de B onde todos os fatores herdam (e são computados de acordo com) a estrutura angular blocada da matriz original.

Observando que $B'B = U'Q'QU = U'U$, vemos que uma maneira alternativa de computar o fator triangular da fatoração ortogonal de B, é computar o fator de Cholesky da matriz simetrizada $B'B$:

$$\begin{bmatrix} {}^1B'{}^1B & & & {}^1B'{}^1C \\ & \ddots & & \vdots \\ & & {}^hB'{}^hB & {}^hB'{}^hC \\ {}^1C'{}^1B & \ldots & {}^hC'{}^hB & {}^0Z \end{bmatrix} .$$

Ao eliminarmos os h blocos das linhas residuais de $B'B$, formamos o bloco sudeste $Z = {}^0Z + \sum_1^h {}^kZ$, a ser fatorado na última etapa do processo, $Z = S'S$. Ao final, obtemos exatamente o fator triangular da fatoração QR da matriz original.

7.3 Partição de Hipergrafos

Um **hipergrafo** é um par ordenado $G = (V, C)$ onde o primeiro elemento é um conjunto finito, o conjunto de vértices, e o segundo elemento é uma matriz booleana, a **matriz de incidência**. Se $|V| = m$, cada coluna de C, $m \times n$, nos dá os vértices sobre os quais incide o correspondente (hiper)lado. No caso particular de todos as colunas terem exatamente 2 ENN's temos na matriz de incidência mais uma representação de um grafo simétrico. Em hipergrafos todavia um lado genérico pode incidir sobre mais de 2 vértices.

O problema de permutar PAQ para forma angular blocada pode ser visto como um problema de partição no hipergrafo $G = (M, B(A))$, $M = \{1, \dots m\}$, $B(A)$ a matriz booleana associada à matriz A, $m \times n$. No problema de partição damos a cada linha $i \in M$ de $B(A)$ uma cor $p(i) \in H = \{1, \dots h\}$. A cor de cada lado é definida como o conjunto de cores dos vértices sobre os quais este incide, $q(j) = \{p(i) \mid A_i^j \neq 0\}$. Lados multicoloridos correspondem a colunas residuais na forma angular blocada, e os lados de cor $q(j) = k$ correspondem às colunas no bloco angular formado pelos ENN's $A_i^j \mid p(i) = k \wedge q(j) = \{k\}$, $k \in H = \{1, \dots h\}$.

Para definir o problema de partição falta-nos uma função objetivo a ser minimizada. Em vista das aplicações já apresentadas, e outras a serem apresentadas no próximo capitulo, queremos ter:

- Aproximadamente o mesmo número de linhas em cada bloco.
- Poucas colunas residuais.

Com estes propósitos é natural considerar a seguinte função de custo de uma dada partição de linhas em cores $p : M \mapsto H$:

$$f(p) = c(p) + \alpha \sum_{k=1}^{h} (m/h - s(k))^2 \ ,$$

$$\text{onde} \quad c(p) = |\{j \in N \mid |q(j)| \geq 2\}| \ , \text{e} \quad s(k) = |\{i \in M \mid p(i) = k\}| \ .$$

Mesmo casos especiais deste problema são NP-difíceis. Por exemplo: Seja A a matriz de incidência de um grafo, m um múltiplo exato de $h = 2$, e faca α suficientemente grande para garantir que todos os blocos tenham exatamente m/h linhas. Este é o problema exato de 2-partição em grafos, e a versão de reconhecimento deste problema é NP-Completa; veja problema ND14 em [Garey79]. Um algoritmo de anulamento simulado com perturbações métricas para resolver este problema é apresentado em [Stern92].

7.4 Paralelismo

Um dos fatores mais importantes no desenvolvimento de algoritmos é a possibilidade de realizar várias etapas de um procedimento em paralelo. No restante deste capítulo adaptaremos algumas das fatorações anteriormente estudadas para as estruturas blocadas, visando paralelizar etapas independentes. Vejamos a seguir alguns conceitos básicos de computação paralela.

Existem vários modelos teóricos de computador paralelo, e inúmeras instâncias e implementações destes modelos em máquinas reais. O modelo mais simples é o de **memória compartilhada**. Neste modelo vários processadores, têm acesso a uma memória comum. Neste modelo, a descrição de uma algoritmo paralelo envolve basicamente dois fatores:

- Como distribuir o trabalho entre os processadores.
- Como sincronizar as diversas etapas do algoritmo.

Este modelo é conceitualmente simples e elegante; todavia limitações da nossa tecnologia inviabilizam a construção de máquinas de memória compartilhada com mais de uns poucos (da ordem de dez) processadores.

O **modelo de rede** é mais genérico. Nele o computador é visto como um grafo: cada vértice, **nó**, ou **processador** representa: um processador propriamente dito, uma **memória local**, i.e., acessível somente a este processador, e portas de comunicação. Cada aresta representa uma **via de comunicação** inter-nós. Note que no modelo de memória compartilhada, a comunicação entre os processadores podia ser feita de maneira trivial através da memória; todavia no modelo de rede é preciso saber os detalhes da arquitetura da máquina para especificar um terceiro aspecto do algoritmo:

- A comunicação entre os processadores.

Estes detalhes incluem a disposição das vias de comunicação, ou **topologia**, a velocidade de comunicação em relação a velocidade de processamento, a possibilidade ou não de haver comunicações simultâneas em vias distintas, etc. Algumas destas topologias comumente empregadas, são: **Estrela**, **Barra**, **Anel**, **Grade**, **Toro**, **Hipercubo** e **Borboleta**.

Como exemplo de algoritmo paralelo, calculemos a média de n números numa rede com p processadores. Suponhamos que inicialmente tenhamos n/p destes números em cada uma das memórias locais. Por simplicidade suponhamos que n é um múltiplo de p, e que $n >> p$. Na primeira fase do algoritmo cada processador, k, calcula a média dos n/p números em sua memória local, $m(p)$. Se os processadores são todos iguais (rede homogênea), cada processador completa sua tarefa em $1/p$ do tempo necessário para calcular a média geral num computador com apenas um processador deste mesmo tipo. Na segunda fase do algoritmo reunimos as médias parciais para calcular a média geral, $m(0) = (1/p)\sum_{k=1}^{p} m(k)$. Examinemos como calcular esta média geral em duas redes com topologia de anel, onde cada qual:

1. Não permite comunicações em paralelo.

2. Permite comunicações em paralelo via segmentos de arco não superpostos.

Novamente por simplicidade, suporemos que $p = 2^q$.

Na primeira rede, para $k = 1 : p$,

1. Calcule no, nó k, $s(k) = s(k-1) + m(k)$.

2. Transmita $s(k)$ ao processador $k+1$.

Neste procedimento temos as somas parciais das médias $s(k) = \sum_{i=1}^{k} m(k)$, a condição de inicialização é $s(0) = 0$, e ao término do algoritmo podemos computar, no nó p, a média $m(0) = s(p)/p$.

Na segunda rede, para $i = 1 : q$,

- $j = 2^i$; em paralelo, para $k = j : p$,
  1. Calcule, no nó k, $r(i,k) = r(i-1,k) + r(i-1,k-j/2)$.
  2. Transmita $r(i,k)$ do nó k para o nó $k+j$.

Neste procedimento temos as somas parciais das médias $r(i,k) = \sum_{l=k-j+1}^{k} m(l)$, a condição de inicialização é $r(0,k) = m(k)$, e ao término do algoritmo podemos computar, no nó p, a média $m(0) = r(q,p)/p$.

Em geral, a transmissão de dados entre nós é muito mais lenta que a manipulação destes dados localmente e, ao medir a complexidade de um algoritmo, contamos separadamente os trabalhos de processamento e comunicação.

7.5 Fatorações Blocadas

Das seções anteriores vemos que quase todo o trabalho na fatoração QR de uma matriz angular blocada, $A = QU$, ou da fatoração de Cholesky $A'A = U'U$, consiste na aplicação repetitiva de algumas operações simples sobre os blocos. Para tirar vantagem desta modularidade em algoritmos para fatoração e atualização de matrizes com estrutura angular blocada, definimos a seguir algumas destas operações, e damos sua complexidade em número de operações de ponto flutuante.

1. Compute a *fatoração de Cholesky parcial*, eliminando as primeiras n colunas da matriz blocada

$$\begin{bmatrix} F & G \\ G^t & 0 \end{bmatrix}$$

para obter

$$\begin{bmatrix} V & W \\ 0 & Z \end{bmatrix}$$

onde $F = F^t$ é $n \times n$, e G é $n \times l$. Isto requer $(1/6)n^3 + (1/2)n^2 l + (1/2)nl^2 + O(n^2 + l^2)$ FLOPs.

2. Compute a *transformação inversa parcial*, i.e. u, em

$$\begin{bmatrix} V & W \\ O & I \end{bmatrix}^t \begin{bmatrix} u^1 \\ u^2 \end{bmatrix} = \begin{bmatrix} y^1 \\ y^2 \end{bmatrix}$$

onde V é $n \times n$ triangular superior, W é $n \times l$, 0 e I são as matrizes zero e identidade, e u e y são vetores coluna. Isto requer $(1/2)n^2 + nl + O(n + l)$ FLOPs.

3. *Reduza a triangular superior* uma matriz de Hessenberg, i.e., aplique a seqüência de rotações de Givens $G(1, 2, \theta), G(2, 3, \theta) \ldots G(n - 1, n, \theta)$ à matriz blocada

$$\begin{bmatrix} V & W \end{bmatrix}$$

onde V é $n \times n$ Hessenberg superior, e W é $n \times l$, para reduzir V a triangular superior. Isto requer $2n^2 + 4nl + O(n^2 + l^2)$ FLOPs.

4. *Reduza a triangular superior* uma matriz blocada coluna – triângulo superior, i.e., aplique a seqüência de rotações de Givens $G(n - 1, n, \theta), G(n - 2, n - 1, \theta), \ldots G(1, 2, \theta)$ à matriz blocada

$$\begin{bmatrix} u & V \end{bmatrix}$$

onde u é um vetor coluna $n \times 1$, e V é $n \times n$ triangular superior, de modo a reduzir u à um único ENN na primeira linha, assim transformando V de triangular para Hessenberg superior. Isto requer $2n^2 + O(n)$ FLOPs.

Em ambas as fatorações, $A = QU$ e $A'A = U'U$, muitas das operações nos blocos podem ser feitas independentemente. Portanto a estrutura angular blocada não só nos dá a possibilidade de preservar esparsidade, mas também a oportunidade de fazer várias operações em paralelo.

Descreveremos uma forma de paralelizar a fatoração de Cholesky numa rede de $h+1$ nós. Para $k = 1 \ldots h$ alocamos blocos das matrizes A e U a nós específicos, como segue:

- Os blocos D^k E^k, V^k e W^k são alocados ao nó k.
- Os blocos sudeste, Z e S, são alocados ao nó 0 (ou $h + 1$).

Expressaremos a complexidade do algoritmo em termos da soma e do máximo das dimensões dos blocos.

$$dbsum = \sum_1^h m(k)$$

$$dbmax = \max\{m(1), \ldots, m(h), n(h+1)\}.$$

Na análise de complexidade contabilizaremos o tempo de processamento, medido em FLOPs, $pTime$, bem como a comunicação inter-nós, INC. Quando h operações sobre blocos, $bop^1 \ldots bop^h$, podem ser efetuadas em paralelo (em nós distintos), contamos seu tempo de processamento por $\wedge_1^h flops(bop^k) = \ flops(bop^1) \wedge \ldots \wedge flops(bop^h)$, onde $\wedge$ é o operador *máximo*, e $flops(bop^k)$ é o numero de operações de ponto flutuante necessário para a operação no bloco k, $bop(k)$. Nas equações que seguem, $\wedge$ tem precedência menor que qualquer operador multiplicativo ou aditivo. As expressões "No nó *k=1:h* compute" ou "Do nó *k=1:h* envie" significam, "Em (de) todos os nós $1 \leq k \leq h$, em paralelo, compute (envie)". Nas expressões de complexidade ignoraremos termos de ordem inferior.

Damos agora uma descrição algorítmica da fatoração de Cholesky blocada $bch()$:

1. No nó *k=1:h* compute os blocos $(B^k)^t B^k$, $(B^k)^t C^k$, e $(C^k)^t C^k$.
   $pTime = m(k)n(k)^2 + m(k)n(k)n(h+1) + m(k)n(h+1)^2 \leq 3dbmax^3$,
   $INC = 0$.

2. Envie $(C^k)^t C^k$ do nó k para o nó 0, onde acumulamos $Z^0 = \sum_1^h (C^k)^t C^k$. $pTime = h\ n(h+1)^2 \leq h\ dbmax^2$, $INC = h\ n(h+1)^2 \leq h\ dbmax^2$

3. No nó k compute a fatoração de Cholesky parcial, eliminando as primeiras $n(k)$ colunas, da matriz blocada
   $$\begin{bmatrix} (B^k)^t B^k & (B^k)^t C^k \\ (C^k)^t B^k & 0 \end{bmatrix}$$
   obtendo
   $$\begin{bmatrix} V^k & W^k \\ 0 & Z^k \end{bmatrix}$$
   $pTime = (1/6)n(k)^3 + (1/2)n(k)^2 n(h+1) + (1/2)n(k)n(h+1)^2 \leq (7/6)dbmax^3$,
   $INC = 0$.

4. Envie Z^k do nó k para o nó 0, onde acumulamos $Z = \sum_0^h Z^k$.
   $pTime = h\ n(h+1)^2 \leq h\ dbmax^2$, $INC = h\ n(h+1)^2 \leq h\ dbmax^2$.

5. No nó 0 fatore o bloco sudeste $S = chol(Z)$, onde $chol()$ indica a fatoração de Cholesky padrão.
   $pTime = (1/6)n(h+1)^3 \leq (1/6)dbmax^3$, $INC = 0$.

Teorema 7.2 *A fatoração de Cholesky blocada, $bch()$, requer não mais de $(4+1/3)dbmax^3 + h\ dbmax^2$ tempo de processamento, e $h\ dbmax^2$ tempo de comunicação inter-nós.*

Nos passos 2 e 4, se a rede permite comunicações em paralelo, a reunião da matriz acumulada pode ser feita em $log(h)$ passos, e podemos substituir h por $log(h)$ no último teorema.

Capítulo 8

ESCALAMENTO

Representação em Ponto Flutuante

8.1 O Sistema de Ponto Flutuante

A representação de um número real, $\zeta \in R$, em um computador tem, usualmente, precisão finita. A inexatidão desta representação introduz erros no resultado final do processamento e o objetivo desta seção é obter limites máximos para estes erros quando da aplicação do método de Gauss. A representação normalmente utilizada para números reais é o sistema de representação em **ponto flutuante normalizado** de t dígitos e base b, SPF, isto é,

$$\begin{aligned} fl(\zeta) &= \pm 0.d_1 d_2 \ldots d_t * b^{\pm n}, \text{ou} \\ & \pm 0.d_1 d_2 \ldots d_t \;\; E \pm n \end{aligned}$$

onde

$$\begin{aligned} & d_k, n, b \in N \\ & 0 \leq d_k \leq b, \;\; d_1 \neq 0 \\ & 0 \leq n \leq emax, \;\; b \neq 0 \;. \end{aligned}$$

Dado um real ν com representação exata num dado SPF, a fração normalizada será denominada mantissa. A **mantissa** e o **expoente** de ν serão denotados, respectivamente,

$$\begin{aligned} mant(\nu) &= \pm 0.d_1 \ldots d_t, \\ expo(\nu) &= \pm n \end{aligned}$$

Há duas maneiras normalmente empregados para obter $fl(\zeta)$ a partir de ζ: **truncamento** e **arredondamento**. Para $trunc(\zeta)$ simplesmente truncamos a representação em base b de ζ

obtendo uma mantissa de t digitos. No arredondamento tomamos a mantissa de t digitos que melhor aproxima ζ. Assim, para $\zeta = \Pi = 3.141592653\ldots$, e o SPF com $b = 10$ e $t = 5$, $trunc(\zeta) = +0.31415\ E + 1$ e $round(\zeta) = +0.31416\ E + 1$.

Note que $\zeta = 0$ não pode ser representado devido a condição $d_1 \neq 0$. Portanto, alguma representação inambígua deve ser convencionada, por exemplo, $fl(0) = +0.00\ldots0\ E + 0$. Se $n > emax$ não há representação possível no SPF, e dizemos que houve "overflow" ou transbordamento.

8.2 Erros no Produto Escalar

Definimos a **unidade de erro** do SPF, u, como b^{1-t} no caso de usarmos truncamento, e $b^{1-t}/2$ no caso de usarmos arredondamento.

Lema 8.1 *Dado $\zeta \in R$ e $fl(\zeta)$ sua representação, num dado SPF de unidade de erro u, $\exists \delta \in [-u, u] \mid fl(\zeta) = \zeta(1+\delta)$.*

Demonstração:

Pela definição de SPF, se $expo(fl(\zeta)) = e$, vê-se que

$$\frac{|fl(\zeta) - \zeta|}{|\zeta|} = |\delta| \leq \frac{ub^{e-1}}{b^{e-1}} = u\ .$$

Se ν e μ são números em ponto flutuante, isto é números reais com representação exata num dado SPF, é perfeitamente possível que uma operação aritmética elementar entre eles resulte num número sem representação exata. Do lema anterior, porém, sabemos que, qualquer que seja a operação aritmética, $\star \in \{+, -, *, /\}$, $fl(\nu \star \mu) = (\nu \star \mu)(1+\delta)$, para algum $\delta \mid |\delta| \leq u$. QED.

Exemplo 1:

Consideremos um computador que armazena um número real em 4 bytes, sendo 3 bytes para os dígitos da mantissa e 1 byte para o sinal do número, o sinal do expoente e os 6 bits restantes para o módulo do expoente como um inteiro em base 2. Supondo que todos os cálculos são feitos com arredondamento, calculamos, a unidade de erro do SPF, u, e o maior real representável, $rmax = b^{emax}$, no caso de usarmos base $b = 256$, $b = 16$ ou $b = 2$.

b	t	$u = b^{t-1}/2$	$rmax = b^{64}$
2	24	$2^{-24} < 10^{-7}$	$2^{64} > 10^{19}$
$16 = 2^4$	6	$2^{-21} < 10^{-6}$	$2^{256} > 10^{75}$
$256 = 2^8$	3	$2^{-17} < 10^{-5}$	$2^{512} > 10^{150}$

Um recurso freqüentemente disponível, concomitantemente ao uso de um SPF de base b e t dígitos, é o sistema de representação em ponto flutuante normalizado de **precisão dupla**, SPFD, com mantissa de $2t$ dígitos, que denotamos $fld(\zeta)$.

Este recurso é extremamente útil, se utilizado parcimoniosamente. Podemos trabalhar com a maior parte dos dados no SPF, de precisão simples, e utilizar a precisão dupla apenas nas passagens mais críticas do procedimento. Em analogia ao SPF, definimos a unidade de erro do SPFD, ud como b^{1-2t} no caso de usarmos truncamento, e $b^{1-2t}/2$ no caso de usarmos arredondamento.

É interessante notar que se ν e μ são números reais com representação exata no SFP, de precisão simples, a representação do produto destes números no SPFD é exata, isto é $fld(\nu * \mu) = \nu * \mu$, pois o produto de dois inteiros de t dígitos tem, no máximo, $2t$ dígitos.

Estudamos agora o efeito cumulativo dos erros de representação, em todas as passagens intermediárias num produto interno. Sejam x, $1 \times n$ e y, $n \times 1$, vetores cujas componentes têm representação exata, num dado SPF. Definimos

$$fl(xy) = fl(fl(x_n y^n) + fl(fl(x_{n-1} y^{n-1}) + \ldots + fl(fl(x_2 y^2) + fl(x_1 y^1)) \ldots)) \ .$$

Lema 8.2 *Dados $u \in [0, 1[$, $n \in \mathbf{N}$ tq $nu \leq \alpha < 1$, e $\delta_i \mid |\delta_i| \leq u$, $i = 1 \ldots n$, então*

$$1 - nu \leq \prod_1^n (1 + \delta_i) \leq 1 + (1 + \alpha) nu \ .$$

Demonstração:

Como

$$(1 - nu)^n \leq \prod_1^n (1 + \delta_i) \leq (1 + nu)^n \ ,$$

basta provar que

- $(1 - u)^n \geq 1 - nu$.
- $(1 + u)^n \leq 1 + nu + \alpha nu$.

Considerando a função $f(u) = (1 - u)^n$ temos que $\exists \theta \in]0, 1[\ \mid$

$$\begin{aligned} f(u) &= f(0) + u f'(u) + u^2 f''(\theta u)/2 \\ &= 1 - nu + n(n-1)(1 - \theta u)^{n-2} u^2/2 \ . \end{aligned}$$

Da não negatividade do último termo segue a primeira inequação.

Considerando que $\forall \beta \in [0, \alpha]$,

$$1 + \beta \leq e^\beta \leq 1 + \beta + \alpha\beta \ ,$$

temos que

$$(1 + u)^n \leq e^(nu) \leq 1 + nu + \alpha nu \ .$$

QED.

Corolário:

Nas condições do lema 2, $\exists \theta \in [-1,1]$ tq

$$\prod_1^n (1+\delta_i) = 1 + \theta(1+\alpha)nu \ .$$

Lema 8.3 *Se x, $1 \times n$ e y, $n \times 1$, são vetores cujas componentes tem representação exata num dado SPF, de unidade de erro u, com $nu < 1$, então*

$$fl(xy) = \sum_{i=1}^{n} x_i y^i (1 + \theta_i (1+\alpha)(n-i+2)u) \ .$$

Demonstração:

$\exists \delta_i, \ |\delta_i| \le ud$, e $\theta_i, \ |\theta_i| \le 1 \quad |$

$$\begin{aligned}
fl(xy) &= fl(fl(x_n y^n) + fl(fl(x_{n-1}y^{n-1}) + \ldots \\
&\quad +(x_3 y^3 (1+\delta_4) + (x_2 y^2 (1+\delta_2) + x_1 y^1 (1+\delta_1))(1+\delta_3))(1+\delta_5)\ldots)) \\
&= fl(fl(x_n y^n) + fl(fl(x_{n-1}y^{n-1}) + \ldots + x_3 y^3 (1+\delta_4)(1+\delta_5) \\
&\quad +x_2 y^2 (1+\delta_2)(1+\delta_3)(1+\delta_5) + x_1 y^1 (1+\delta_1)(1+\delta_3)(1+\delta_5)\ldots)) \\
&= x_n y^n (1+\delta_{2n-2})(1+\delta_{2n-1}) + \\
&\quad +x_{n-1} y^{n-1} (1+\delta_{2n-4})(1+\delta_{2n-3})(1+\delta_{2n-1}) + \ldots \\
&\quad +x_2 y^2 (1+\delta_2)(1+\delta_3)\ldots(1+\delta_{2n-3})(1+\delta_{2n-1}) \\
&\quad +x_1 y^1 (1+\delta_1)(1+\delta_3)\ldots(1+\delta_{2n-3})(1+\delta_{2n-1}) \\
&= x_n y^n (1+\theta_n(1+\alpha)2u) + x_{n-1}y^{n-1}(1+\theta_{n-1}(1+\alpha)3u) + \ldots \\
&\quad +x_2 y^2 (1+\theta_2(1+\alpha)nu) + x_1 y^1 (1+\theta_1(1+\alpha)(n+1)u) \ .
\end{aligned}$$

Lema 8.4 *Se x, $1 \times n$ e y, $n \times 1$, são vetores cujas componentes têm representação exata num SPF, em precisão simples, de unidade de erro u, sendo $n * ud \le \gamma < 1$, então*

$$\begin{aligned}
fld(xy) &\equiv fld(fld(x_n y^n) + fld(fld(x_{n-1}y^{n-1}) + \ldots + fld(fld(x_2 y^2) + fld(x_1 y^1))\ldots)) \\
&= \sum_1^n x_i y^i (1+\theta_i(1+\gamma)(n-i+1)ud)
\end{aligned}$$

Demonstração:

$\exists \epsilon_i, \ |\epsilon_i| \le ud$, e $\theta_i, \ |\theta_i| \le 1 \quad |$

$$\begin{aligned}
fld(xy) &= fld(x_n y^n + fld(x_{n-1}y^{n-1} + \ldots \\
&\quad +(x_3 y^3 + (x_2 y^2 + x_1 y^1)(1+\epsilon_1))(1+\epsilon_2)\ldots)) \\
&= x_n y^n (1+\epsilon_{n-1}) + x_{n-1}y^{n-1}(1+\epsilon_{n-2})(1+\epsilon_{n-1}) + \ldots \\
&\quad x_2 y^2 (1+\epsilon_1)\ldots(1+\epsilon_{n-1}) + x_1 y^1 (1+\epsilon_1)\ldots(1+\epsilon_{n-1}) \\
&= x_n y^n (1+\theta_n(1+\gamma)ud) + x_{n-1}y^{n-1}(1+\theta_{n-1}(1+\gamma)2ud) + \\
&\quad x_2 y^2 (1+\theta_2(1+\gamma)(n-1)ud) + x_1 y^1 (1+\theta_1(1+\gamma)nud)
\end{aligned}$$

É freqüente o cálculo em dupla precisão, de produtos de vetores armazenados em precisão simples, e o posterior armazenamento do resultado em precisão simples, isto é o cálculo $fl(fld(xy))$.

Do último lema vemos que

$$fl(fld(xy)) = (1+\delta)\sum_1^n x_i y^i (1+\theta_i(1+\gamma)(n-i+1)ud) \ .$$

Observação 8.1 Se $xy \gg \sum x_i y^i \theta_i (1+\gamma)(n-i+1)ud$, o que ocorre se $nu \ll 1$ e não houver "cancelamentos críticos" na somatória, o resultado final do produto é afetado de um erro da ordem do erro introduzido por uma única operação aritmética, em precisão simples. Assumiremos esta hipótese no restante do capítulo.

Exemplo 2:

Calculemos, no SPF decimal de 2 dígitos com arredondamento, $fl(xy)$, $fld(xy)$ e $fl(fld(xy))$, onde $x = [7.5,\ 6.9,\ 1.3]$ e $y = [0.38,\ -0.41,\ 0.011]'$.

$$fld(xy) = fld(2.85 - 2.829 + 0.0143) = fld(0.021 + 0.0143) = 0.0353 \ .$$

$$fl(fld(xy)) = fl(0.0353) = 0.035 \ .$$

$$fl(xy) = fl(2.9 - 2.8 - 0.014) = fl(0.1 + 0.014) = 0.11 \ .$$

8.3 Escalamento de Matrizes

Da análise de erros no produto interno, e considerando que a maioria das operações nos algoritmos de triangularização podem ser agrupadas na forma de produtos internos, fica evidente que é conveniente termos todos os elementos da matriz da mesma ordem de grandeza, i.e., termos a matriz bem **equilibrada**. Evitamos assim ter multiplicadores muito pequenos e soma de termos de ordem de grandeza muito diferentes. Se tal não ocorre para a matriz de um dado sistema, $Ax = b$, podemos procurar x através da solução de um outro sistema melhor equilibrado.

Se $E = diag(e_1, e_2, \ldots e_n)$ e $D = diag(d_1, d_2, \ldots d_n)$, onde $e_i, d_i \neq 0$, o sistema $EADy = Eb$ é obtido multiplicando-se a i-ésima equação do sistema por e_i, e efetuando-se a substituição de variáveis $x = Dy$, o que equivale a multiplicar a j-ésima coluna de A por d_j. Uma transforamação $\tilde{A} = EAD$, A $m \times n$, E e D diagonais com $e_i, d_i \neq 0$, é um **escalamento** da matriz A.

Exemplo 3:

$$EAD = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 11 & 0 & 0 \\ 0 & 12 & 0 \\ 0 & 0 & 13 \end{bmatrix} = \begin{bmatrix} 11 & 12 & 13 \\ 22 & 24 & 26 \\ 31 & 36 & 39 \end{bmatrix}$$

Estudaremos o problema de escolher um escalamento que "melhor equilibre" uma dada matriz. Em primeiro lugar, vale notar que, estando num SPF de base b, a escolha de E e D da forma $E_i^i = b^{e_i}$. $D_j^j = b^{d_j}$, onde e e d são vetores de elementos inteiros, é muito conveniente, pois $\tilde{A} = EAD$ e A terão elementos de mesma mantissa, sendo o efeito do escalamento apenas o de alterar os expoentes dos elementos da matriz,

$$mant(\tilde{A}_i^j) = mant(A_i^j) \quad , \quad expo(\tilde{A}_i^j) = expo(A_i^j) + e_i + d^j \ .$$

Observação 8.2 Para implementar eficientemente a função $expo()$ e a operação de soma de um inteiro ao expoente de um número real, devemos conhecer detalhadamente o SPF usado e manipular diretamente os campos de bits envolvidos.

Uma maneira de medir o grau de desequilibrio de uma matriz, A é atravez da média e da variância dos expoentes de seus elementos:

$$mex(A) = \sum_{i,j|A_i^j \neq 0} expo(A_i^j)/enn(A) \ ,$$

$$vex(A) = \sum_{i,j|A_i^j \neq 0} (expo(A_i^j) - mex)^2/enn(A) \ .$$

Nas somatórias que definem mex e vex excluimos os elementos nulos da matriz, os elementos nulos da matriz podem ser eliminados das operações de produto interno.

Para não sobrecarregar a notação, doravante escrevemos

$$\sum_i{}' = \sum_{i=1 \ |A_i^j \neq 0}^{m} \ , \quad \sum_j{}' = \sum_{j=1 \ |A_i^j \neq 0}^{n} \ , \quad \sum_{i,j}{}' = \sum_{i,j=1 \ |A_i^j \neq 0}^{m,n} \ .$$

O primeiro método de escalamento que estudaremos é justamente o **método da redução de variância** em que procuramos minimizar a variância dos expoentes de EAD.

Tomemos as matrizes de escalamento esquerda e direita como, respectivamente, $E = diag(round(e_i^*))$ e $D = diag(round(d_j^*))$, sendo os vetores e^* e d^* argumentos que minimizão a variância dos expoentes da matriz escalada,

$$vex(EAD) = \sum_{i,j}{}'(expo(A_i^j) + e_i + d_j - mex(A))^2 \ .$$

Um ponto de mínimo deve obedecer ao sistema

$$\frac{\partial vex(EAD)}{\partial e_i^*} = 0 = 2\sum_j{}'(expo(A_i^j) + e_i^* + d_j^* - mex(A))$$
$$\frac{\partial vex(EAD)}{\partial d_j^*} = 0 = 2\sum_i{}'(expo(A_i^j) + e_i^* + d_j^* - mex(A))$$

ou, fazendo a substituicao $e_i^+ = e_i^* - mex(A)/2$, $d_j^+ = d_j^* - mex(A)/2$

$$\begin{aligned} \sum_j {}' e_i^+ + d_j^+ &= -\sum_j {}' expo(A_i^j) \\ \sum_i {}' e_i^+ + d_j^+ &= -\sum_i {}' expo(A_i^j) \end{aligned}$$

Se a matriz A não tiver elementos nulos, então a solução e^*, d^* do sistema é imediata:

$$\begin{aligned} e_i^* &= (1/n)\sum_j (mex(A) - expo(A_i^j)) \\ d_i^* &= (1/m)\sum_i (mex(A) - expo(A_i^j)) \end{aligned}$$

Esta equação pode nos dar uma boa aproximação da solução em matrizes densas, i.é, com poucos elementos nulos, como se considerássemos o "expoente dos elementos nulos" como sendo expoente médio de A. Este é o **método aproximado da redução de variância**. Procuremos agora métodos heurísticos para determinação de escalamentos, que sejam menos trabalhosos que o método da redução de variância.

O **método da média geométrica** consiste em tomar

$$\begin{aligned} d_j &= -int((\sum_i {}' expo(A_i^j))/enn(A^j)) \\ e_i &= -int((\sum_j {}'(expo(A_i^j) + d_j))/enn(A^i)) \end{aligned}$$

Assim os fatores de quilibramento são uma aproximação do inverso da média geométrica dos elementos não nulos das colunas, ou linhas, i.e.,

$$\begin{aligned} d_j &= -int((\sum_i {}' \log(|A_i^j|))/enn(A^j)) \\ e_i &= -int((\sum_j {}'(\log(|A_i^j|) + d_j))/enn(A^i)) \end{aligned}$$

Uma variante do método da média geométrica é o **método da média max-min**, no qual tomamos

$$\begin{aligned} d_j &= -int((\max_i {}' expo(A_i^j) + \min_i {}' expo(A_i^j))/2) \\ e_i &= -int((\max_j {}'(expo(A_i^j) + d_j) + \min_j {}'(expo(A_i^j) + d_j))/2) \end{aligned}$$

Finalmente, o **método da norma infinito** consiste em tomar

$$\begin{aligned} e_i &= -\max_j {}' expo(A_i^j) \\ d_j &= -\max_i {}'(expo(A_i^j) + e_i) \end{aligned}$$

A escolha do particular método a ser empregado depende bastante do tipo de matriz a ser equilibrada e da exigência que temos sobre $vex(A)$. Em pacotes comerciais de otimização é comum a aplicação do método max-min um número pré-determinado de vezes, ou até que variância dos expoentes se reduza a um valor limite aceitável. Este limite deve ser tomado em função das condições do problema, como por exemplo o número de condição da matriz, ser definido no capítulo 9, e da unidade de erro do SPF.

Exemplo 4:

Equilibremos a matriz A, dada abaixo, num SPF de base 10,

1. pelo método aproximado de redução de variância,
2. pelo método da média geométrica,
3. pelo método max-min,
4. pelo método da norma ∞.

Para A e para cada um dos escalamentos, calculemos mex, vex, e o diâmetro da matriz, definido como a diferença entre o maior e o menor expoente dos elementos de A.

$$A = \begin{bmatrix} 1 & 0.7E-4 & 0.5E-1 & 0.9E0 & -0.2E2 \\ 0 & 0 & 0.3E-1 & 0 & 0.3E4 \\ 1 & -0.8E-3 & 0 & 0 & 0.3E6 \\ 0 & 0.3E-1 & -0.8E0 & 0.1E3 & 0.7E9 \end{bmatrix}$$

$$expo(A) = \begin{bmatrix} 0 & -4 & -1 & 0 & 2 \\ x & x & -1 & x & 4 \\ 0 & -3 & x & x & 6 \\ x & -1 & 0 & 3 & 9 \end{bmatrix} \quad \begin{matrix} +1 \\ 160 \\ 13 \end{matrix}$$

Pelo método aproximado de redução de variância, temos e, d, $expo(EAD)$ e $[mex, vex, diam]$, respectivamente

$$\begin{bmatrix} int(8/5) = 2 \\ int(2/5) = 0 \\ int(2/5) = 0 \\ int(-6/5) = -1 \end{bmatrix} \begin{bmatrix} int(4/4) = 1 \\ int(12/4) = 3 \\ int(6/4) = 2 \\ int(1/4) = 0 \\ int(-17/4) = -4 \end{bmatrix} \begin{bmatrix} 3 & 1 & 3 & 2 & 0 \\ x & x & 1 & x & 0 \\ 0 & 0 & x & x & 2 \\ x & 3 & 1 & 2 & 4 \end{bmatrix} \quad \begin{matrix} 1.57 \\ 23.4 \\ 4 \end{matrix}$$

Analogamente, pelo método da média geométrica, temos

$$\begin{bmatrix} -int(4/5) = 1 \\ -int(-1/2) = 1 \\ -int(1/3) = 0 \\ -int(8/4) = -2 \end{bmatrix} \begin{bmatrix} -int(0/2) = 0 \\ -int(-8/3) = 3 \\ -int(-2/4) = 1 \\ -int(3/2) = -2 \\ -int(21/4) = -5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 1 & -2 \\ x & x & 1 & x & 0 \\ 0 & 0 & x & x & 1 \\ x & 0 & -1 & -1 & 2 \end{bmatrix} \quad \begin{matrix} 0.214 \\ 14.4 \\ 4 \end{matrix}$$

Analogamente, pelo método max-min, temos

$$\begin{bmatrix} -int(-4/2) = 2 \\ -int(-2/2) = 1 \\ -int(0/2) = 0 \\ -int(4/2) = -2 \end{bmatrix} \quad \begin{bmatrix} -int(0/2) = 0 \\ -int(-5/2) = 3 \\ -int(-1/2) = 1 \\ -int(3/2) = -2 \\ -int(11/2) = -6 \end{bmatrix} \quad \begin{bmatrix} 2 & 1 & 2 & & -2 \\ x & x & 1 & x & -1 \\ 0 & 0 & x & x & 0 \\ x & 0 & -1 & & 1 \end{bmatrix} \quad \begin{matrix} 0.143 \\ 14.3 \\ 4 \end{matrix}$$

Analogamente, pelo método da norma-∞, temos

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ -3 \\ -9 \end{bmatrix} \quad \begin{bmatrix} 0 & -3 & -1 & -3 & -7 \\ x & x & 0 & x & -4 \\ 0 & -2 & x & x & -3 \\ x & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{matrix} -1.64 \\ 59.2 \\ 7 \end{matrix}$$

Escalamentos visando equilibrar as matrizes do problema são uma etapa importante na solução de sistemas lineares de grande porte. Note que operações de escalamento não afetam os elementos nulos de uma matriz, e portanto não alteram sua estrutura e esparsidade. O desempenho das heurísticas estudadas variam conforme a área de aplicação; vale pois testar experimentalmente as heurístcas escalamento e suas variações.

Capítulo 9

ESTABILIDADE

Erros e Perturbações da Solução

9.1 Normas e Condição

Uma **norma**, num dado espaço vetorial E, é uma função

$$|| \, . \, || \; : \; E \Rightarrow \mathbf{R} \mid \forall x, y \in E, \alpha \in \mathbf{R} \; :$$

1. $||x|| \geq 0$, e $||x|| = 0 \Leftrightarrow x = 0$.
2. $||\alpha x|| = |\alpha| \; ||x||$.
3. $||x + y|| \leq ||x|| + ||y||$, a desigualdade triangular.

São de grande interesse em $\mathbf{R}^n$ as p-**normas**

$$||x||_p = (\sum_1^n |x_i|^p)^{1/p} \; ,$$

e de particular interesse a norma 1, a norma 2 (ou norma quadrática, ou Euclidiana) e a norma $p = +\infty$. No caso da norma infinito, devemos tomar o limite da definição de p-norma para $p \rightarrow +\infty$, ou seja,

$$||x||_p = \max_{i=1}^n |x_i| \; .$$

Dado um espaço vetorial normado $(E, \; || \; ||)$ definimos a **norma induzida** sobre as transformações lineares limitadas, $T : \; E \rightarrow E$ tq $\exists \alpha \in \mathbf{R} \mid \forall x \in E, \; ||T(x)|| \leq \alpha ||x||$ como sendo

$$||T|| \equiv \max_{x \neq 0} (\; ||T(x)|| \; / \; ||x|| \;)$$

ou equivalentemente, por linearidade

$$||T|| \equiv \max_{x|\ ||x||=1} ||T(x)|| \ .$$

Em $(\mathbf{R}^n,\ ||\ ||)$ falamos da norma induzida sobre as matrizes $n \times n$ como sendo a norma da transformação associada, isto é $||A|| = ||T||$, onde $T(x) = Ax$,

Lema 9.1 *A norma induzida sobre as matrizes em* $(\mathbf{R}^n,\ ||\ ||)$*, goza das propriedades, para* $\forall A, B\ n \times n,\ \alpha \in \mathbf{R}$,

1. $||A|| \geq 0$ *e* $||A|| = 0 \Leftrightarrow A = 0$
2. $||A + B|| \leq ||A|| + ||B||$
3. $||AB|| \leq ||A||\ ||B||$

Lema 9.2 *Para a norma 1 e para norma* ∞ *temos as seguintes expressões explícitas da norma induzida sobre as transformações, ou matrizes,*

$$\begin{aligned} ||A||_1 &= \max_{j=1}^{n} \sum_{i=1}^{n} |A_i^j| \\ ||A||_\infty &= \max_{i=1}^{n} \sum_{j=1}^{n} |A_i^j| \end{aligned}$$

Demonstração:

Para verificar a validade das expressões observe que

$$\begin{aligned} ||Ax||_1 &= \sum_{i=1}^{n} |\sum_{j=1}^{n} A_i^j x_j| \leq \sum_{i=1}^{n}\sum_{j=1}^{n} |A_i^j|\ |x_j| \\ &\leq \sum_{j=1}^{n} |x_j| \max_{j=1}^{n} \sum_{i=1}^{n} |A_i^j| = ||A||_1\ ||x||_1 \end{aligned}$$

$$\begin{aligned} ||Ax||_\infty &= \max_{i=1}^{n} |\sum_{j=1}^{n} A_i^j x_j| \leq \max_{i=1}^{n} \sum_{j=1}^{n} |A_i^j|\ |x_j| \\ &\leq \max_{j=1}^{n} |x_j| \max_{i=1}^{n} \sum_{j=1}^{n} |A_i^j| = ||x||_\infty\ ||A||_\infty \end{aligned}$$

e que, se k é o índice que realiza o máximo na definição da norma, então as igualdades são realizadas pelos vetores $x = I^k$, para a norma 1, e $x \mid x_j = sig(A_i^j)$, para a norma ∞.

Definimos o **número de condição** de uma matriz como

$$cond(A) = ||A||\ ||A^{-1}|| \ .$$

Exemplo 1:

Calcule a norma da matriz de binomial de dimensão 3, e de sua inversa, nas normas 1 e ∞. Para cada uma das normas calculadas exiba um vetor x que, multiplicado pela matriz, seja "esticado" de um fator igual à própria norma, i.e. $x \mid \|Ax\| = \|A\|\|x\|$.

$$
{}^3B = \begin{bmatrix} \binom{1}{0} & \binom{1}{1} & 0 \\ \binom{2}{0} & \binom{2}{1} & \binom{2}{2} \\ \binom{3}{0} & \binom{3}{1} & \binom{3}{2} \end{bmatrix}
$$

$$
{}^3B = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 1 & 3 & 3 \end{bmatrix} \quad {}^3B^{-1} = \begin{bmatrix} 3 & -3 & 1 \\ -2 & 3 & -1 \\ 1 & -2 & 1 \end{bmatrix}
$$

Para ${}^3B$ a soma da norma dos elementos $|A_i^j|$ por linha e por coluna são, respectivamente, $\begin{bmatrix} 7 & 6 & 4 \end{bmatrix}$ e $\begin{bmatrix} 6 & 8 & 3 \end{bmatrix}$. Para ${}^3B^{-1}$, analogamente, temos $\begin{bmatrix} 7 & 6 & 4 \end{bmatrix}$ e $\begin{bmatrix} 7 & 6 & 4 \end{bmatrix}$. Assim, para a norma 1, $||{}^3B||$, $||{}^3B^{-1}||$, e $cond({}^3B)$ são respectivamente 6, 8, e 48. Analogamente, para norma ∞, temos 7, 7, e 49. Finalmente, $\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}'$ e $\begin{bmatrix} 1 & -1 & 1 \end{bmatrix}'$ são vetores como os procurados.

9.2 Perturbações

Estudaremos agora uma maneira de limitar o efeito de uma pertubação nos dados do sistema linear, $Ax = b$, sobre a sua solução.

Teorema 9.1 (da pertubação) *Se os dados do sistema $Ax = b$ forem pertubados, isto é, alterados de uma matriz δA de um vetor δb, a resposta será pertubada por um δx, isto é $(A + \delta A)(x + \delta x) = (b + \delta b)$, tal que, se $||\delta A||\ ||A^{-1}|| < 1$, então*

$$
\frac{||\delta x||}{||x||} \leq \frac{cond(A)}{1 - cond(A)\ ||\delta A||/||A||} \left(\frac{||\delta b||}{||b||} + \frac{||\delta A||}{||A||} \right) .
$$

Demonstração.

$(A + \delta A)(x + \delta x) = Ax + A\delta x + \delta Ax + \delta A\delta x = b + \delta b$, portanto $A\delta x = \delta b - \delta Ax - \delta A\delta x$, e $\delta x = A^{-1}(\delta b - \delta Ax - \delta A\delta x)$. Assim,

$$
||\delta x|| \leq ||A^{-1}||\,(||\delta b|| + ||\delta A||||x|| + ||\delta A||||\delta x||) .
$$

Como também $||b|| \leq ||A||||x||$, temos que

$$\left(1 - ||A^{-1}||||\delta A||\right) \frac{||\delta x||}{||x||} \leq ||A^{-1}|| \left(\frac{||\delta b||}{||x||} + ||\delta A|| \right)$$
$$\leq ||A^{-1}|| \left(\frac{||A||||\delta b||}{||b||} + ||\delta A|| \right) .$$

Usando a hipótese $||\delta A||||A^{-1}|| \leq 1$, temos o teorema, QED.

Exemplo 2:

Considere o sistema $(B + \delta A)x = (b + \delta b)$, onde a matriz de coeficientes, e o vetor de termos independentes correspondem à matriz binomial de dimensão 3, ${}^3B$, e ${}^3b \mid {}^3B\mathbf{1} = {}^3b$. Os elementos da matriz e do vetor de pertubação, δA_i^j e δb_i, são aleatórios. A distribuição de cada um destes elementos de perturbação é independente e é uma função de dois parametros (α, p): Tomemos cada elemento da perturbação é no cojunto $\{0, -\alpha, \alpha\}$ com probabilidade, respectivamente, $[1 - p, p/2, p/2]$.

Com os dados do Exemplo 1, determine um limite máximo para $0 \leq \alpha \leq alfamax$ que garanta a hipótese do teorema da pertubação. Faça uma experiência comparando limites de erro e pertubação da solução do sistema proposto.

Na norma 1, $||\delta A|| \leq 3\alpha$, e do Exemplo 1 sabemos que $||B^{-1}|| \leq 7$. Assim, a condição $||\delta A||||A^{-1}|| \leq 1$ está garantida para $\alpha \leq 0.04 < 1/21$.

Tomando como experimento de perturbação,

$$\delta A = \begin{bmatrix} 0 & 0.01 & -0.01 \\ 0 & 0.01 & 0 \\ 0 & 0.01 & -0.01 \end{bmatrix} \quad e \quad \delta b = \begin{bmatrix} 0 \\ -0.01 \\ 0 \end{bmatrix}$$

a solução do sistema pertubado é $x = [1.0408, 0.9388, 1.0622]'$, i.e. $\delta x = [0.04, -0.06, 0.06]'$ e $||\delta x|| = 0.16$.

Por outro lado, o Teorema da pertubação nos dá o limite

$$\frac{||\delta x||}{||x||} \leq \frac{49}{1 - 49 * 0.03/7} (\frac{0.01}{7} + \frac{0.03}{7}) = 0.35$$

um limite superior de acordo com o resultado obtido no experimento.

9.3 Erro na Fatoração LU

Consideremos a resolução de um sistema, $Ax = b$, pelo método de Doolittle, isto é, a decomposição $A = LU$, e a solução do sistema $Ly = b$ e $Ux = b$.

Teorema 9.2 (Wilkinson) *A solução, afetada pelos erros de arredondamento, pode ser escrita como a solução exata, x, mais um termo de pertubação δx. Assumiremos que um produto escalar em dupla precisão é afetado de um erro da ordem do erro de passagem para precisão simples, conforme a observação 8.1. Nestas condições,, a solução calculada, $(x + \delta x)$, é solução exata de um sistema. $(A + \delta A)(x + \delta x) = b$, tal que $||\delta A||_\infty \leq (2n + 1)gu$, onde $g = \max_{i,j} |U_i^j|$.*

Demonstração.

Consideremos as linhas da matriz A já ordenadas de modo a não haver necessidade de pivoteamentos. A decomposição da matriz A será dada por, para $i = 1 \dots n$

$$\begin{aligned} L_i U &= A_i \\ L U^i &= A^i \end{aligned}$$

ou, para $i = 1 \dots n$, para

$$\begin{aligned} j = i \dots n \qquad & U_i^j = A_i^j - \sum_{k=1}^{i-1} M_i^k U_k^j \\ j = i+1 \dots n \qquad & M_j^i = (A_j^i - \sum_{k=1}^{i-1} M_j^k U_k^i)/U_i^i \end{aligned}$$

Na realidade, obteremos as matrizes afetadas de erro, para $i = 1 \dots n$, para

$$\begin{aligned} j = i \dots n \qquad & \tilde{U}_i^j = fl(fld(A_i^j - \sum_{k=1}^{i-1} \tilde{M}_i^k \tilde{U}_k^j)) \\ j = i+1 \dots n \qquad & \tilde{M}_j^i = fl(fld((A_j^i - \sum_{k=1}^{i-1} \tilde{M}_j^k \tilde{U}_k^i)/\tilde{U}_i^i))) \end{aligned}$$

Como supomos desprezíveis os termos de $O(ud)$ (vide observação 8.1) temos, para $i = 1 \dots n$, para

$$\begin{aligned} j = i \dots n \qquad & \tilde{U}_i^j = (1 + \delta_i^j)(A_i^j - \sum_{k=1}^{i-1} \tilde{M}_i^k \tilde{U}_k^j) \\ j = i+1 \dots n \qquad & \tilde{M}_j^i = (1 + \delta_j^i)(A_j^i - \sum_{k=1}^{i-1} \tilde{M}_j^k \tilde{U}_k^i)/\tilde{U}_i^i \end{aligned}$$

Portanto, para $i = 1 \dots n$, para

$$\begin{aligned} j = i \dots n \qquad & \sum_{k=1}^{i-1} \tilde{M}_i^k \tilde{U}_k^j + 1 U_i^j = \tilde{L}_i \tilde{U}^j = A_i^j + \tilde{U}_i^j \delta_i^j / (1 + \delta_i^j) \\ j = i+1 \dots n \qquad & \sum_{k=1}^{i-1} \tilde{M}_j^k \tilde{U}_k^i + \tilde{M}_j^i \tilde{U}_i^i = \tilde{L}_j \tilde{U}^i = A_j^i + \tilde{M}_j^i \tilde{U}_i^i \delta_j^i / (1 + \delta_j^i) \end{aligned}$$

Notemos agora que: $|\delta_i^j|/(1+\delta_i^j) \sim |\delta_i^j| \leq u$, que $|M_i^j| \leq 1$, pelo pivoteamento parcial, e definindo $g = \max_{i,j} |U_i^j|$, temos que $\tilde{L}\tilde{U} = A + E$, onde $|E_i^j| \leq gu$, donde $||E||_\infty \leq ngu$.

Na solução do sistema $\tilde{L}y = b$ e $\tilde{U}x = y$ calculamos, para $i = 1 \ldots n$,

$$\tilde{y}_i = fl(fld((b_i - \sum_{j=1}^{i-1} \tilde{L}_i^j \tilde{y}_j)/\tilde{L}_i^i))$$

e novamente, para $i = 1 \ldots n$,

$$\tilde{x}_i = fl(fld((\tilde{y}_i - \sum_{j=i+1}^{n} \tilde{U}_i^j \tilde{x}_j)/\tilde{U}_i^i)) \ .$$

Supondo desprezíveis os termos de $0(ud)$

$$\begin{aligned} \tilde{y}_i &= (1+\delta_i)(b_i - \sum_{j=1}^{i-1} \tilde{L}_i^j \tilde{y}_j)/\tilde{L}_i^i \\ \tilde{x}_i &= (1+\delta_i')(y_i - \sum_{j=i+1}^{n} \tilde{U}_i^j \tilde{x}_j)/\tilde{U}_i^i \end{aligned}$$

ou

$$\begin{aligned} \sum_{j=1}^{i} \tilde{L}_i^j \tilde{y}_j &= \tilde{L}_i \tilde{y} = b_i + \tilde{L}_i^j \tilde{y}_i \delta_i/(1+\delta_i) \\ \sum_{j=i}^{n} \tilde{U}_i^j \tilde{x}_j &= \tilde{U}_i \tilde{x} = \tilde{y}_i + U_i^j \tilde{x}_i \delta_i'/(1+\delta_i') \end{aligned}$$

isto é

$$\begin{aligned} (\tilde{L} + \delta L)\tilde{y} &= b \\ (\tilde{U} + \delta U)\tilde{x} &= \tilde{y} \end{aligned}$$

onde para as matrizes diagonais δL e δU, temos $|\delta L_i^i| \leq u$ e $|\delta U_i^i| \leq gu$.

Em suma,

$$\begin{aligned} b &= (\tilde{L} + \delta L)\tilde{y} = (\tilde{L} + \delta L)(\tilde{U} + \delta U)\tilde{x} \\ &= (\tilde{L}\tilde{U} + \tilde{L}\delta U + \delta L\tilde{U} + \delta L\delta U)\tilde{x} \\ &= (A + E + \tilde{L}\delta U + \delta L\tilde{U} + \delta L\delta U)\tilde{x} \\ &= (A + \delta A)\tilde{x} \end{aligned}$$

Desprezando o termo $\delta L\delta U$, de $O(ud)$, $\delta A = E + \tilde{L}\delta U + \delta L\tilde{U}$, donde

$$||\delta A||_\infty = ||E|| + ||\tilde{L}\delta U + \delta LU\tilde{U}|| \leq ngu + (n+1)gu$$

pois

$$
\begin{aligned}
&||\tilde{L}\delta U + \delta L\tilde{U}||_\infty \le \\
&\le \left\| \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 1 & & 1 \end{bmatrix} \begin{bmatrix} gu & & 0 \\ & \ddots & \\ 0 & & gu \end{bmatrix} + \begin{bmatrix} u & & 0 \\ & \ddots & \\ 0 & & u \end{bmatrix} \begin{bmatrix} g & & g \\ & \ddots & \\ 0 & & g \end{bmatrix} \right\|_\infty \\
&\le \left\| \begin{bmatrix} gu & & 0 \\ & \ddots & \\ gu & & gu \end{bmatrix} + \begin{bmatrix} gu & & gu \\ & \ddots & \\ 0 & & gu \end{bmatrix} \right\| \\
&\le \left\| \begin{bmatrix} 2gu & & gu \\ & \ddots & \\ gu & & 2gu \end{bmatrix} \right\| = (n+1)gu \quad QED.
\end{aligned}
$$

Para frisar a importância do uso conjugado do método de Doolittle e dupla precisão daremos, sem demonstração, a versão do teorema ora demonstrado para o método de Gauss, que equivale ao método de Doolittle sem o recurso da dupla precisão: Definindo $h = \max_{i,j,k} \mid {}^k A_i^j|$, teríamos $||E||_\infty \le O(n^2 hu)$ e $||\delta A||_\infty \le O(n^3 hu)$. Estes resultados tornam evidente a obrigatoriedade de calcularmos os produtos escalares envolvidos no processo, sempre em dupla precisão, a menos que tratemos de sistemas de pequena dimensão.

Além do pivoteamento parcial, isto é, da escolha como elemento pivô do maior elemento em módulo na coluna, poderíamos usar o **pivoteamento total**, isto é, encolher como elemento pivô o maior elemento em módulo em qualquer linha ou coluna utilizável (isto é, poderiamos tomar por pivô da transformação ${}^{k-1}A \rightarrow {}^kA$, um elemento em $arg\max_{k\le i,j \le n} \mid {}^{k-1}A_i^j$. Usando pivoteamento total, podemos demonstrar a existência de um limite superior para a constante h, supondo que a matriz original A é tal que $|A_i^j| \le 1$, da ordem de $O(n^{(1/4)\ln(n)})$, contra limites de $O(2^n)$ para pivoteamento parcial. O uso do pivoteamento total é todavia, extremamente custoso, pois exige a cada passo da $O(n^2)$ comparações, contra $O(n)$ no pivoteamento parcial. Ademais, estes limites de h tendem a ser extremamente pessimistas, principalmente se A for bem equilibrada.

Exercícios

1. Estabilidade de Fatorações Simétricas.

   (a) Adapte os resultados de estabilidade da faoração LU para o caso particular da fatoração de Cholesky.

   (b) Qual o número de condição de uma matriz ortogonal? Usando a relação entre o fator triangular da fatoração QR e a fatoração de Cholesky, discuta a estabilidade da fatoração QR.

2. Prove que $|| \; ||_1$, $|| \; ||_2$ e $|| \; ||_\infty$ são efetivamente normas.

3. Desenhe em $\mathbf{R}^2$ a regiao $||x|| \leq 1$, para as normas 1, 2 e ∞.

4. Prove o Lema 1.

5. Equivalência das normas 1, 2 e ∞: Prove em $\mathbf{R}^n$ que

   (a) $||x||_\infty \leq ||x||_1 \leq n||x||_\infty$

   (b) $||x||_\infty \leq ||x||_2 \leq n^{1/2}||x||_\infty$

   (c) $||A||_2 = \lambda$, onde λ^2 é o maior autovalor de $A'A$.

   (d) $||A'A||_2 = ||A||_2^2$

   (e) $||A||_2 \leq (\; ||A||_1||A||_\infty)^{1/2}$

6. Calcule computacionalmente $cond(A)$, para as matrizes de teste ${}^nT$, $A = {}^nB$ e $A = {}^nH$, para $n = 2, 4, 8, 16$. Estime graficamente o crescimento do número de condição das matrizes teste em função da dimensão.

7. Resolva computacionalmente os sistemas teste ${}^8Tx = {}^8t$, ${}^8Bx = {}^8b$ e ${}^8Hx = {}^8h$.

   (a) Pelo método de Gauss com pivoteamento parcial;

   (b) Pelo método de Gauss com pivoteamento total;

   (c) Pelo método de Doolittle com pivoteamento parcial e dupla precisão;

   (d) Pelo método de Doolittle com pivoteamento total e dupla precisão.

   No item c, verifique se o erro final está dentro do esperado, em função da unidade de erro do SPF utilizado.

Capítulo 10

MUDANÇA de BASE

Atualizações de Posto 1

Em Otimização, principalmente em Programação Linear, o termo base significa uma matriz quadrada de posto pleno. Estudaremos agora o problema de **mudança de base**, aqui posto na seguinte forma: Seja $\hat{A}$ a nova base, obtida da base original, A, substituindo-se a coluna A^s por uma nova coluna, a, isto é $\hat{A} = [A^1, \ldots A^{s-1}, a, A^{s+1}, \ldots A^n]$. Se já dispusermos da inversa de A (ou na prática mais comumente de uma fatoração de A), como atualizar a inversa (ou a fatoração)? Isto é, como, a partir da inversa de A obter a inversa de $\hat{A}$ com um trabalho muito menor que o necessário para **reinverter** $\hat{A}$, i.e., computar a inversa de novo, sem aproveitar a informação contida em A^{-1} ou, na fatoração $A = LU$ ou $A = QR$?

10.1 Fórmulas de Modificação

Teorema 10.1 (fórmula geral de modificação) *Dada A, $n \times n$ e inversível, $V = A^{-1}$, δA, $n \times n$, e $\alpha \in R$ suficientemente pequeno, podemos fazer a expansão*

$$(A + \alpha\delta A)^{-1} = V + \sum_{k=1}^{\infty} (-\alpha)^k \, (V \, \delta A)^k \, V \ .$$

Demonstração:

Em virtude da regra de Cramer podemos, para $\alpha \in [0, \text{alphamax}]$, escrever a série de Taylor para cada elemento de $(A + \alpha\delta A)^{-1}$,

$$(A + \delta A)^{-1} = V + \sum_{k=1}^{\infty} \frac{\alpha^k}{k!} \, \delta^k V$$

Para determinar a matriz que dá a perturbação de ordem k da inversa $\delta^k V$ observemos que

$$(A + \alpha\delta A)(V + \alpha \, \delta V + \frac{\alpha^2}{2} \, \delta^2 V + \frac{\alpha^3}{3!} \, \delta^3 V + \ldots) = I$$

donde

$$\alpha(\delta A\ V + A\ \delta V) + \alpha^2(\frac{1}{2}A\ \delta^2 V + \delta A\ \delta V) + \alpha^3(\frac{1}{6}A\ \delta^3 V + \frac{1}{2}\delta A\ \delta^2 V) + \ldots = 0$$

de modo que

$$\delta V = -V\ \delta A\ V\ ,\quad \delta^2 V = 2V\ \delta A\ V\ \delta A\ V\ ,\quad \delta^3 V = (V\ \delta A)^3\ V\ ,\ldots$$

$$\delta^k V = (-1)^k\ k!\ (V\ \delta A)^k\ V\ .$$

Substituindo esta fórmula geral de $\delta^k V$ na série de Taylor de $(A + \alpha\ \delta A)$, segue diretamente o teorema. QED.

Teorema 10.2 (fórmula de Sherman e Morrison) *Dada A, $n \times n$ e inversível, u e w $n \times 1$, e $\alpha \in R^*$, então a inversa de $\hat{A} = A + \alpha u w'$ é dada por*

$$\hat{A}^{-1} = A^{-1} + \beta A^{-1} u w' A^{-1}\ ,$$

$$\beta = -(\alpha^{-1} + w' A^{-1} u)^{-1}\ .$$

Demonstração:

Da fórmula geral de modificação, supondo que $\alpha <$ alfamax, e das propriedades do operador traço (exercício 1.5) temos:

$$\begin{aligned}
&(A + \alpha u w')^{-1} = \\
&\quad = V + \sum_{k=1}^{\infty} (-\alpha)^k (V u w') V \\
&\quad = V - \alpha V u w' V \sum_k (-\alpha \mathrm{tr}(V u w'))^k \\
&\quad = V - \frac{\alpha V u w' V}{1 + \alpha \mathrm{tr}(V u w')} \\
&\quad = V - \frac{V u w' V}{\alpha^{-1} + \mathrm{tr}(V u w')} \\
&\quad = V - (\alpha^{-1} + w' V u)^{-1}\ V u w' V\ .
\end{aligned}$$

o que prova o teorema para $0 < \alpha <$ alphamax.

A fórmula de Sherman e Morrison é todavia uma identidade que podemos provar diretamente para qualquer $\alpha > 0 \mid (\alpha^{-1} + w' A^{-1} u) \neq 0$: Usando a formula de Sherman e Morisson para desenvolver a identidade $\hat{A}^{-1}\hat{A} = I$, obtemos,

$$\begin{aligned}
(V + \beta V u w' V)(A + \alpha u w') &= I \quad \Leftrightarrow \\
\beta V u w' + \alpha\beta (V u w')^2 &= -\alpha V u w' \quad \Leftrightarrow \\
(\alpha^{-1} + w' V u) \alpha V u w' &= -\beta^{-1} \alpha V u w'
\end{aligned}$$

sendo esta última identidade trivialmente verdadeira. Q.E.D.

Exemplo 1:

Tomando

$$A = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \ , \quad u = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \ , \quad w = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \ , \quad \alpha = 1 \ ,$$

podemos calcular a inversa de

$$\hat{A} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 8 & 9 \end{bmatrix}$$

calculando

$$\beta = -(1+3)^{-1} = -1/4 \ ,$$

$$\hat{A}^{-1} = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} - (1/4) \begin{bmatrix} -5 & 4 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 9/4 & -1 \\ -2 & 1 \end{bmatrix}$$

Observação 10.1 Note que uma mudança de base poderia ser feita pela fórmula de Sherman e Morrison, pois se $\hat{A} = [A^1, \ldots A^{j-1}, a, A^{j+1}, \ldots A^n]$, então, $\hat{A} = A + (a - A^j)\ I_j$, de modo que a inversa da nova base seria, tomando $V = A^{-1}$,

$$\begin{aligned} \hat{A}^{-1} &= V + \beta V (a - A^j) I_j V \\ &= V + V \beta (a - A^j) V_j \end{aligned}$$

$$\text{onde} \ \ \beta = -(1 - I_j V (a - A^j))^{-1} = -(V_j a)^{-1} \ .$$

Exemplo 2:

Se

$$A = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \ , \quad a = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \ \text{ e } \ j = 2 \ ,$$

então a inversa da nova base, $\hat{A}^{-1}$, pose ser calculada como segue

$$\beta = -(\begin{bmatrix} -2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix})^{-1} = -1$$

$$\begin{aligned} \hat{A}^{-1} &= (\begin{bmatrix} 1 & 1 \\ 2 & 3 \end{bmatrix})^{-1} = \\ &= \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} (-1) (\begin{bmatrix} 1 \\ 3 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix}) \begin{bmatrix} -2 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix} + \begin{bmatrix} -1 \\ 0 \end{bmatrix} \begin{bmatrix} -2 & 1 \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ -2 & 1 \end{bmatrix} \end{aligned}$$

10.2 Atualizações Estáveis

O **Algoritmo de Bartels e Golub**, que apresentaremos a seguir, nos dá uma atualização estável da fatoração LU de uma matriz. Tomando $T = L^{-1}$, $A = LU$,

$$T\hat{A} = [U^1, \dots U^{s-1}, Ta, U^{s+1}, \dots U^n] \ .$$

Consideremos agora a permutação de colunas que comuta a s-ésima e a n-ésima colunas, $\hat{Q}$ Esta permutação leva $\hat{A}$ em $\bar{A} = \hat{A}\hat{Q}$, e $T\hat{A}$ em

$$T\hat{A}\hat{Q} = \begin{bmatrix} U_1^1 & \dots & U_1^{s-1} & T_1 a & U_1^{s+1} & \dots & U_1^n \\ \vdots & \ddots & \vdots & \vdots & & & \vdots \\ 0 & & U_{s-1}^{s-1} & T_{s-1}a & U_{s-1}^{s+1} & & U_{s-1}^n \\ 0 & & 0 & T_s a & U_s^{s+1} & & U_s^n \\ 0 & & 0 & T_{s+1}a & U_{s+1}^{s+1} & & U_{s+1}^n \\ \vdots & & & \vdots & & \ddots & \\ 0 & \dots & 0 & T_n a & 0 & \dots & U_n^n \end{bmatrix} Q =$$

$$H = \begin{bmatrix} U_1^1 & \dots & U_1^{s-1} & U_1^{s+1} & \dots & U_1^{n-1} & U_1^n & T_1 a \\ 0 & \ddots & & & & & & \vdots \\ \vdots & \ddots & U_{s-1}^{s-1} & U_{s-1}^{s+1} & & U_{s-1}^{n-1} & U_{s-1}^n & T_{s-1}a \\ 0 & & 0 & U_s^{s+1} & & U_s^{n-1} & U_s^n & T_s a \\ 0 & & 0 & U_{s+1}^{s+1} & \ddots & U_{s+1}^{n-1} & U_{s+1}^n & T_{s+1}a \\ \vdots & & & & \ddots & \ddots & & \vdots \\ 0 & & 0 & 0 & & U_{n-1}^{n-1} & U_{n-1}^n & T_{n-1}a \\ 0 & \dots & 0 & 0 & \dots & 0 & U_n^n & T_n a \end{bmatrix}$$

H é uma matriz de **Hessenberg** superior, isto é, apenas os elementos paralelos à diagonal principal podem ser não nulos no triângulo inferior. O método de Bartels e Golub consiste na aplicação de método de Gauss, com pivotamento parcial, à matriz H.

Observemos que a aplicação do método de Gauss à matriz H é extremamente simples, pois

1. As primeiras $s-1$ etapas de transformação não são necessárias, pois nas colunas $1, \dots s-1$, H já é triangular superior.

2. As etapas $j = s, \dots n-1$, que transformam

$$^0H = \ ^{s-1}H \rightarrow \ ^sH \rightarrow \dots \ ^jH \rightarrow \dots \ ^{n-1}H = \ \hat{U}$$

resumem-se em

(a) Permutar as linhas j e $j+1$ se $\mid {}^{j-1}H_{j+1}^j \mid > \mid {}^{j-1}H_j^j \mid$, obtendo uma nova matriz ${}^{j-1}\tilde{H}$.

(b) Calcular um único multiplicador ${}^{j}N^{j}_{j+1} = {}^{j-1}\tilde{H}^{j}_{j+1} / {}^{j-1}\tilde{H}^{j}_{j}$.

(c) Atualizar uma única linha ${}^{j}H_{j+1} = {}^{j-1}\tilde{H}_{j+1} - {}^{j}N^{j}_{j+1}\, {}^{j-1}\tilde{H}_{j}$.

A aplicação do metodo de Gauss nos dá a fatoração $\tilde{H} = \hat{L}\hat{U}$, onde $\tilde{H} = \hat{R}H$ é a matriz obtida de H pelas permutações de linhas realizadas durante a triangularização. Assim,

$$\begin{aligned} \tilde{H} &= \hat{R}H = \hat{R}T\bar{A} = \hat{L}\hat{U} \ , \quad \text{donde} \\ \bar{A} &= L\hat{R}^{t}\hat{L}\hat{U} \ , \quad \text{ou} \\ \hat{V} = \bar{A}^{-1} &= \hat{U}^{-1}\hat{T}\hat{R}T \end{aligned}$$

Após uma seqüência de mudanças de base, nossa representação da inversa teria a forma

$${}^{k}V = {}^{k}U^{-1}\, {}^{k}T\, {}^{k}R \ldots {}^{1}T\, {}^{1}R\, T \ .$$

10.3 Preservando Esparsidade

O **algoritmo de Saunders**, que agora examinamos, pode ser visto como uma adaptação do algoritmo de Bartels e Golub visando aproveitar a prévia estruturação da base pelo algoritmo P4.

Suponhamos termos a fatoração $A = LU$ obtida pela aplicação do método de Gauss (sem pivotamento de linhas mas com possíveis permutações de colunas) à matriz A previamente estruturada pelo P4, por exemplo

Exemplo 3:

	1	2	3	4	5	6	7	8	9	10
1	x									
2		x		x						
3	x	x	x	0						
4			x	x						
5	x			x	x					
6	x		x	0		x				x
7		x		x	x		x	x		x
8		x		0			x	x		x
9		x		x		x		x	x	x
10	x				x			x	x	0

No Exemplo 3, "x" indica as posições originalmente não nulas de A e "0" indica as posições preenchidas durante a triangularização.

Seja $\tilde{U}$ a matriz obtida de U pela permutação simétrica, $Q'UQ$, que leva os espinhos, preservando sua ordem de posicionamento, para as últimas colunas à direita. No Exemplo 3, teríamos,

$$\tilde{U} = Q'UQ = \begin{array}{c||ccccccc|ccc}
 & 1 & 2 & 3 & 5 & 6 & 7 & 9 & 4 & 8 & 10 \\
\hline\hline
1 & x & & & & & & & & & \\
2 & & x & & & & & & x & & \\
3 & & & x & & & & & x & & \\
5 & & & & x & & & & & & \\
6 & & & & & x & & & & & \\
7 & & & & & & x & & & & x \\
9 & & & & & & & x & & x & x \\
\hline
4 & & & & & & & & x & & \\
8 & & & & & & & & & x & x \\
10 & & & & & & & & & & x
\end{array}$$

Esta matriz tem estrutura $\tilde{U} = \begin{bmatrix} D & E \\ 0 & F \end{bmatrix}$, onde D é diagonal e F é triangular superior.

O algoritmo de Saunders atualiza a decomposição $A = LQ\tilde{U}Q'$ para a nova base

$$\begin{array}{rcl}
\hat{A} & = & \begin{bmatrix} A^1 & \ldots & A^{s-1} & a & A^{s+1} & \ldots & A^n \end{bmatrix} \quad \text{ou} \\
\hat{A}\hat{Q} & = & \begin{bmatrix} A^1 & \ldots & A^{s-1} & A^{s+1} & \ldots & A^n & a \end{bmatrix}
\end{array}$$

como segue:

1. Forma, pela permutação $\tilde{U}\hat{Q}$, e substituição da última coluna,

$$W = T\hat{A}\hat{Q} = Q'TAQ\hat{Q} = \begin{bmatrix} \tilde{U}^1 & \ldots & \tilde{U}^{s-1} & \tilde{U}^{s+1} & \ldots & \tilde{U}^n & Q'Ta \end{bmatrix} .$$

2. Forma, pela permutação simétrica,

$$\hat{W} = \hat{Q}'T\hat{A}\hat{Q} == \hat{Q}'Q'T\hat{A}Q\hat{Q} = \hat{W} =$$
$$\begin{bmatrix} (W_1)' & \ldots & (W_{s-1})' & (W_{s+1})' & \ldots & (W_n)' & (W_s)' \end{bmatrix}' .$$

Note que $\hat{W}$ tem uma estrutura do tipo

$$\hat{W} = \begin{bmatrix} \hat{D} & \hat{E} & \hat{W}^n_{1:n-c-1} \\ 0 & \hat{F} & \hat{W}^n_{n-c:n-1} \\ 0 & W_s & \hat{W}^n_n \end{bmatrix}$$

onde $\hat{D}$ é diagonal e W_s só pode ter elementos não nulos nas últimas posições à direita (sob $\hat{F}$ e $\hat{W}^n$).

3. Triangulariza $\hat{W}$, isto é, $\hat{N}$, pelo método de Gauss com pivotamento parcial. A matriz $\hat{N}$, constituída de $\hat{F}$ e dos últimos elementos da linha W_s e da coluna $\hat{W}^n$, é denominada **núcleo** de $\hat{W}$. Nas rotinas numéricas pode ser conveniente guardar apenas o núcleo, $c \times c$, $c << n$ como uma matriz densa na memória principal, enquanto o resto da matriz, usada apenas para leitura, pode ser guardada em uma representação esparsa, e na memória secundária.

No Exemplo 3 teríamos, ao mudar a coluna 3 da base A,

$\hat{A} =$

	1	2	3	4	5	6	7	8	9	10
1	x		y							
2		x		x						
3	x	x								
4			y	x						
5	x			x	x					
6	x					x				x
7		x	y	x	x		x	x		x
8		x					x	x		x
9		x		x		x		x	x	x
10	x		y		x			x	x	

$W =$

	1	2	5	6	7	9	4	8	10	3
1	x									y
2		x					x			
3							x			
5			x							
6				x					x	
7					x			x	x	y
9						x			x	
4							x			y
8								x	x	
10									x	y

$$
\hat{W} = \begin{array}{r||cccccc|ccc|c}
 & 1 & 2 & 5 & 6 & 7 & 9 & 4 & 8 & 10 & 3 \\
\hline\hline
1 & x & & & & & & & & & y \\
2 & & x & & & & & x & & & \\
5 & & & x & & & & & & & \\
6 & & & & x & & & & & x & \\
7 & & & & & x & & & x & x & y \\
9 & & & & & & x & & & x & \\
\hline
4 & & & & & & & x & & & y \\
8 & & & & & & & & x & x & \\
10 & & & & & & & & & x & y \\
\hline
3 & & & & & & & x & & & 0
\end{array}
$$

Após a triangularização de $\hat{W}$ teremos, sendo $\hat{R}$ as permutações de linhas realizadas ao triangularizar $\hat{W}$,

$$
\begin{aligned}
\hat{R}\hat{W} &= \hat{L}\hat{U} = \hat{R}\hat{Q}'Q'T\hat{A}Q\hat{Q} \quad \text{portanto} \\
\hat{A} &= LQ\hat{Q}\hat{R}'\hat{L}\hat{U}\hat{Q}'Q' \quad \text{e} \\
\hat{A}^{-1} &= Q\hat{Q}\hat{U}^{-1}\hat{T}\hat{R}\hat{Q}'Q'T \; .
\end{aligned}
$$

Em geral, após uma seqüência de mudanças de base, nossa representação da inversa terá a forma

$$
{}^kA^{-1} = Q\;{}^1Q \ldots {}^kQ\;{}^kU^{-1}\;{}^kT\;{}^kR\;{}^kQ' \ldots {}^1T\;{}^1R\;{}^1Q\;Q\;T \; .
$$

Observação 10.2

1. A região dos preenchimentos em ${}^kW$ está restrita ao triângulo superior do núcleo e sua última linha.

2. A matriz ${}^kL$ só terá elementos nulos na última linha do núcleo.

3. A cada nova mudança de base, a dimensão do núcleo:

   (a) Aumenta de 1, se a coluna que sai da base é uma coluna triangular da base original.

   (b) Permanece constante, se a coluna que sai da base é um espinho, ou uma coluna já anteriormente substituída.

4. Cada mudança de base aumenta um termo na fatoração da base, o que leva a uma gradativa perda de eficiência e acúmulo de erros. Depois de um número pré-determinado de mudanças sobre uma base original, ou quando o acúmulo de erros assim o exigir, devemos fazer uma reiversão, i.é., reiniciar o processo aplicando a P4 e triangularizando a próxima base desejada.

10.4 Preservando Estrutura

Consideraremos agora o problema de atualizar a fatoração $A = QU$ de uma base na forma angular blocada. Conforme argumentamos no final do capítulo 7, estamos apenas interessados no fator U, sendo as transformações ortogonais descartadas logo após o seu uso. No que segue, a coluna a sair da base será a coluna $outj$ do bloco $outk$, $1 \leq outk \leq h+1$. Em seu lugar será introduzida na base uma coluna com a estrutura do bloco ink, $1 \leq ink \leq h+1$.

Apresentamos agora um procedimento de atualização de U por blocos, *bup*(), que usa explicitamente a estrutura angular blocada da base A e do fator U. Este procedimento será descrito em termos das operações simples em cada bloco apresentadas na seção 7.5.

Em *bup*() consideraremos cinco casos distintos:

Caso I $ink \neq outk, \quad ink \neq h+1, \quad outk \neq h+1.$

Caso II $ink = outk, \quad ink \neq h+1.$

Caso III $ink \neq h+1, \quad outk = h+1.$

Caso IV $ink = h+1, \quad outk \neq h+1.$

Caso V $ink = outk, \quad ink = h+1.$

Vejamos em detalhe o caso I, quando ink and $outk$ são blocos diagonais distintos, como mostrado na figura 1. Neste caso os únicos ENN's na coluna saindo estão no bloco ${}^{\text{outk}}B_{\bullet}^{\text{outj}}$. Analogamente os únicos ENN's na coluna entrando na base, a, estão em ${}^{\text{ink}}a$.

Definimos $y \equiv A'a$, e $u \equiv Q'a = U^{-t}A'a = U^{-t}y$. Notamos que os vetores y e u preservam a estrutura blocada da base. Assim, os ENN's em u estão nos blocos ${}^{ink}u$ e ${}^{h+1}u$,

$$\begin{bmatrix} {}^{ink}u \\ {}^{h+1}u \end{bmatrix} = \begin{bmatrix} {}^{ink}V & {}^{ink}W \\ 0 & S \end{bmatrix}^{-t} \begin{bmatrix} {}^{ink}y \\ {}^{h+1}y \end{bmatrix}$$

Para atualizar U removemos a coluna $outj$ do bloco $outk$, e inserimos u como a última coluna de ${}^{ink}U$. Depois apenas temos que reduzir U a uma matriz triangular superior através de transformações ortogonais.

1. Leve $\begin{bmatrix} {}^{outk}V & {}^{outk}W \end{bmatrix}$ de Hessenberg a triangular superior.

2. Leve $\begin{bmatrix} {}^{h+1}u & S \end{bmatrix}$ a triangular.

3. Insira a primeira linha de ${}^{h+1}U$, como a última linha de ${}^{ink}U$.

4. Insira a última linha de ${}^{outk}U$ como a primeira de ${}^{h+1}U$.

5. Leve S de Hessenberg a triangular superior.

Os outros casos são bastante similares. Os esquemas nas figuras 2, 3 e 4 são os análogos do esquema na figura 1, e dão uma descrição sumária de cada um dos casos.

Estes são os passos para o caso I :

1. No nó ink, compute $y^{ink} = (B^{ink})^t a^{ink}$ e $y^{b+1} = (C^{ink})^t a^{ink}$.
   $pTime = m(ink)n(ink) + m(ink)n(b+1) \leq 2dbmax^2$, $INC = 0$.

2. No nó ink, compute a transformação inversa parcial

   $$\begin{bmatrix} u^{ink} \\ z \end{bmatrix} = \begin{bmatrix} V^{ink} & W^{ink} \\ 0 & I \end{bmatrix}^{-t} \begin{bmatrix} y^{ink} \\ y^{b+1} \end{bmatrix}$$

   Então insira u^{ink} como a última coluna de V^{ink}.
   $pTime = (1/2)n(ink)^2 + n(ink)n(b+1) \leq (3/2)dbmax^2$, $INC = 0$.

3. Do nó ink envie z ao nó 0.
   $pTime = 0$, $INC = n(b+1) \leq dbmax$.

4. No nó 0 compute $u^{b+1} = S^{-t}z$.
   $pTime = (1/2)n(b+1)^2 \leq (1/2)dbmax^2$, $INC = 0$.

5. (a) No nó $outk$, remova a coluna $V^{outk}_{\bullet,outj}$ de V^{outk}. Então reduza $\begin{bmatrix} V^{outk} & W^{outk} \end{bmatrix}$ de Hessenberg para triangular superior.
   (b) No nó 0, reduza $\begin{bmatrix} u^{b+1} & S \end{bmatrix}$ para triangular superior.

   Observe que as operações nos passos $5a$ e $5b$ são independentes, portanto
   $pTime = 2n(ink)^2 + 4n(ink)n(b+1) \wedge 2n(b+1)^2 \leq 6dbmax^2$, $INC = 0$.

6. Do nó 0 envie o vetor $S_{1,\bullet}$ ao nó ink, onde ele é inserido como a última coluna de W^{ink}. Do nó 0 envie o elemento u_1^{b+1} ao nó ink, onde ele é inserido como $U^{ink}_{n(ink)+1,n(ink)+1}$. Do nó $outk$ envie vetor $W^{outk}_{n(outk),\bullet}$ ao nó 0, onde ele é inserido como a primeira linha de S.
   $pTime = 0$, $INC = 2n(b+1) + n(outk) \leq 3dbmax$.

7. No nó 0, reduza S de Hessenberg para triangular superior.
   $pTime = 2n(b+1)^2 \leq 2dbmax^2$, $INC = 0$.

Estes são os passos para o caso II :

Os Passos 1—5 são exatamente como no caso I.

6. (a) Do nó ink envie $V^{ink}_{n(ink),n(ink)}$ e $W^{ink}_{n(ink),\bullet}$ ao nó 0.

(b) No nó 0 reduza para triangular superior a matriz $2 \times n(b+1)+1$

$$\begin{bmatrix} V^{ink}_{n(ink),n(ink)} & W^{ink}_{n(ink),\bullet} \\ u_1^{b+1} & S_{1,\bullet} \end{bmatrix}$$

$pTime = 4n(b+1) \leq 4dbmax$, $INC = n(b+1) \leq dbmax$.

7. (a) Do nó 0 envie o vetor modificado $\begin{bmatrix} V^{ink}_{n(ink),n(ink)} & W^{ink}_{n(ink),\bullet} \end{bmatrix}$ de volta ao nó ink.

   (b) No nó 0, reduza S de Hessenberg para triangular superior.

   $pTime = 2n(b+1)^2 \leq 2dbmax^2$, $INC = n(b+1) \leq dbmax$.

Estes são os passos do caso III :

Os passos 1—4 são exatamente como no caso I.

5. (a) No nó $k = 1 : b$ remova a coluna $W^k_{\bullet,outj}$ de W^k.

   (b) No nó 0 reduza $\begin{bmatrix} u^{b+1} & S \end{bmatrix}$ para triangular superior. Remova $S_{\bullet,outj}$ de S.

   $pTime = 2n(b+1)^2 \leq 2dbmax^2$, $INC = 0$.

6. Do nó 0 envie ao nó ink, u_1^{b+1} para ser inserido em V^{ink} como $V^{ink}_{n(ink)+1,n(ink)+1}$, e $S_{1,\bullet}$ para ser inserido como a última coluna de W^{ink}.
   $pTime = 0$, $INC = n(b+1) \leq dbmax$.

7. No nó 0, reduza S de Hessenberg para triangular superior.
   $pTime = 2n(b+1)^2 \leq 2dbmax^2$, $INC = 0$.

Estes são os passos do caso IV :

1. No nó *k=1:b*, compute $y^k = (B^k)^t a^k$ e $x^k = (C^k)^t a^k$.
   $pTime = \wedge_1^b \ m(k)n(ink) + m(k)n(b+1) \leq 2dbmax^2$, $INC = 0$.

2. No nó *k=1:b*, compute a transformação inversa parcial

$$\begin{bmatrix} u^k \\ z^k \end{bmatrix} = \begin{bmatrix} V^k & W^k \\ 0 & I \end{bmatrix}^{-t} \begin{bmatrix} y^{ink} \\ x^k \end{bmatrix}$$

   e insira u^k como a última coluna de W^k.
   $pTime = \wedge_1^b \ (1/2)n(k)^2 + n(k)n(b+1) \leq (3/2)dbmax^2$, $INC = 0$.

3. Do nó *k=1:b* envie z^k ao nó 0, onde acumulamos $z = \sum_1^b z^k$.
   $pTime = b\ n(b+1) \leq b\ dbmax$, $INC = b\ n(b+1) \leq b\ dbmax$.

4. No nó 0 compute $u^{b+1} = S^{-t}z$, e insira u^{b+1} como a última coluna de S.
   $pTime = (1/2)n(b+1)^2 \leq (1/2)dbmax^2$, $INC = 0$.

5. remova a coluna $V^{outk}_{\bullet,outj}$ de V^{outk} , e reduza $\left[\begin{array}{cc} V^{outk} & W^{outk} \end{array} \right]$ para triangular superior.
$pTime = 2n(outk)^2 + 4n(outk)n(b+1) \leq 6dbmax^2$, $INC = 0$.

6. Envie o vetor $W^{outk}_{n(ouk),\bullet}$ do nó $outk$ ao nó 0, onde o inserimos como a primeira linha de S, e reduza S a triangular.
$pTime = 2n(b+1)^2 \leq 2dbmax^2$, $INC = n(b+1) \leq dbmax$.

Estes são os passos do caso V :

Os passos 1—4 são exatamente como no caso IV.

5. No nó *k=1:b*, remova a coluna $W^k_{\bullet,outj}$ de W^{outk} , e insira u^k como a última coluna de W^k. No nó 0 remova a coluna $S_{\bullet,outj}$ de S , e insira u^{b+1} como a última coluna de S.
$pTime = 0$, $INC = 0$.

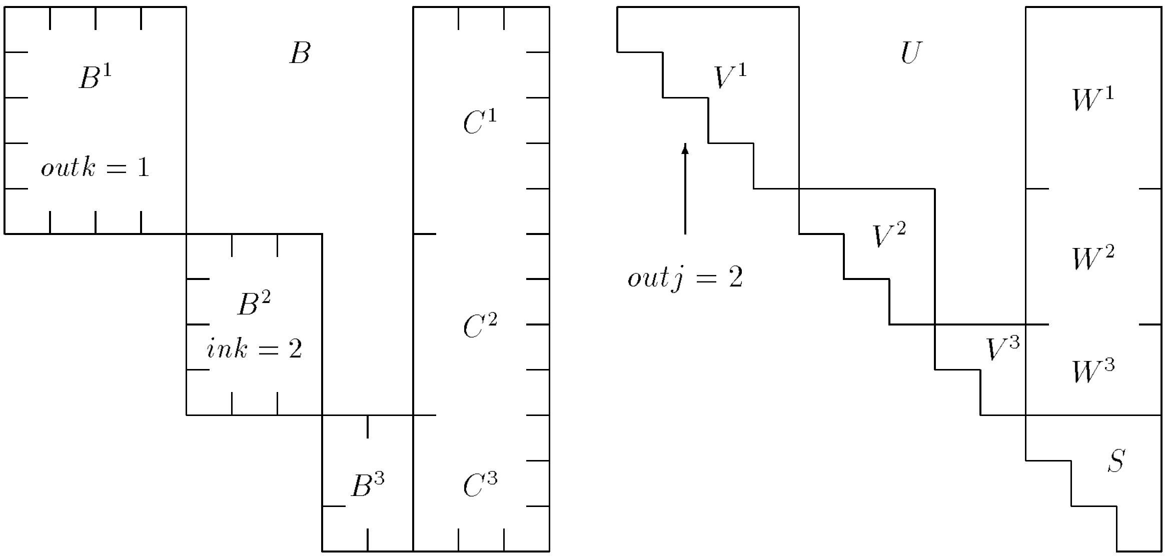
6. No nó 0, reduza S de Hessenberg para triangular superior.
$pTime = 2n(b+1)^2 \leq 2dbmax^2$, $INC = 0$.

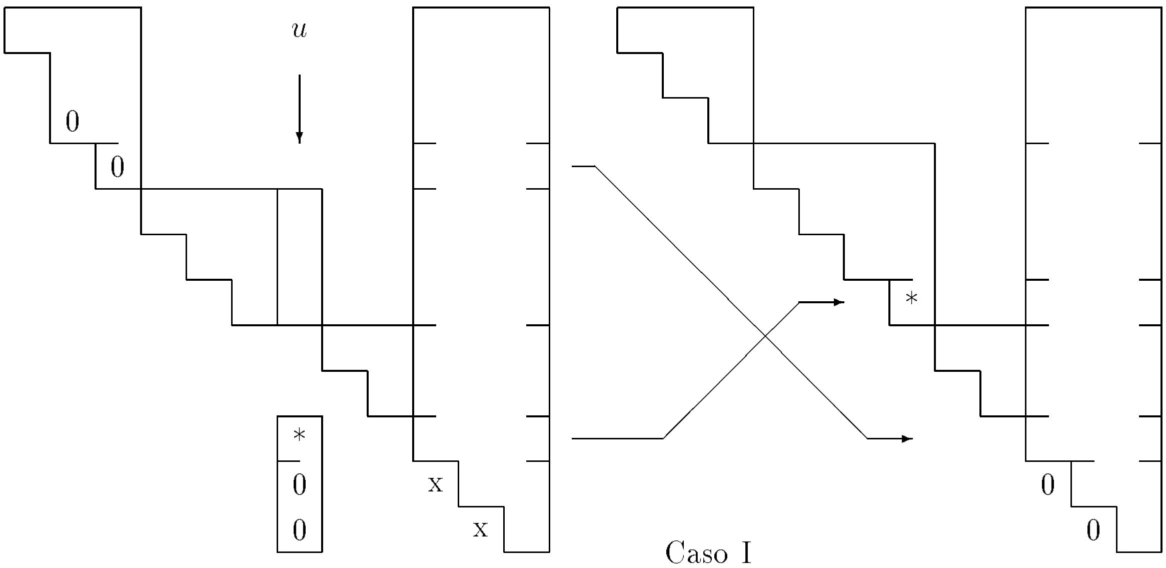
A complexidade do procedimento de atualização de U por blocos, *bup*(), é dada pelo seguinte teorema: Como na seção 7.4, $dbmax = \max_{k=1}^{h} n(k)$.

Teorema 10.3 *A complexidade de bup(), desconsideramos termos de ordem inferior, tem os seguintes limitantes superiores para tempo de processamento e comunicação:*

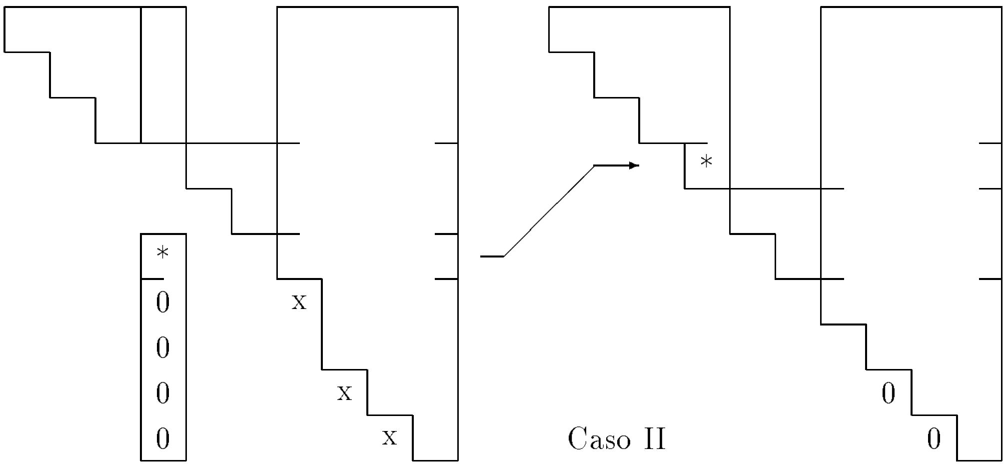
Caso	*Tempo de Processamento*	*Comunicação*
I	$12dbmax^2$	$3dbmax$
II	$12dbmax^2$	$3dbmax$
III	$8dbmax^2$	$2dbmax$
IV	$12dbmax^2 + b\ dbmax$	$h\ dbmax$
V	$6dbmax^2 + b\ dbmax$	$h\ dbmax$

Caso o ambiente permita comunicações em paralelo, podemos substituir h por $log(h)$ nas expressões de complexidade.

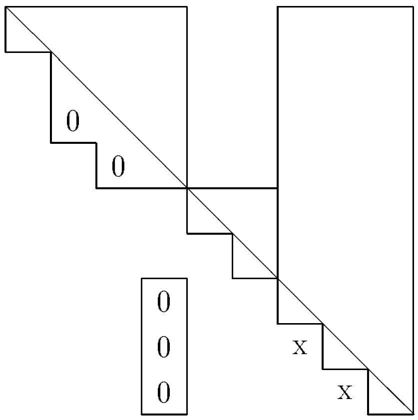

B^1
$outk = 1$
B
C^1
B^2
$ink = 2$
C^2
B^3
C^3
V^1
U
W^1
$outj = 2$
V^2
W^2
V^3
W^3
S


u
0
0
*
0
0
x
x
*
0
0

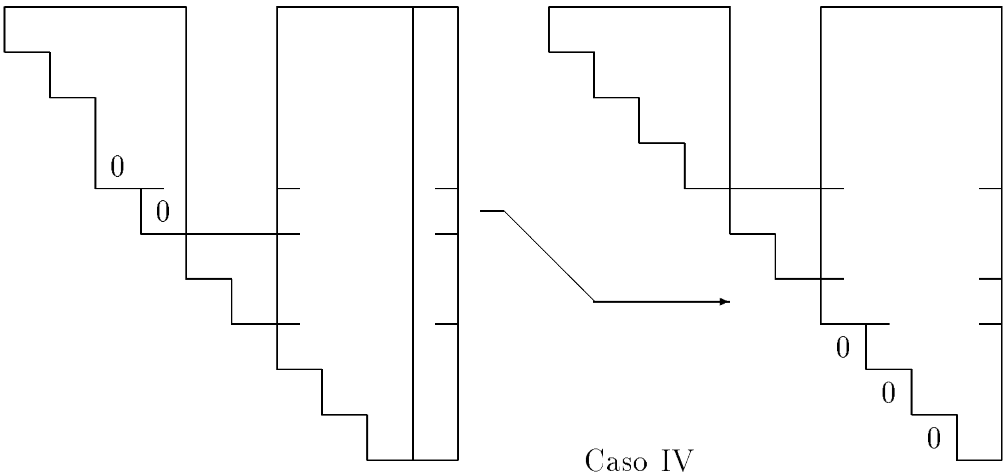
Caso I



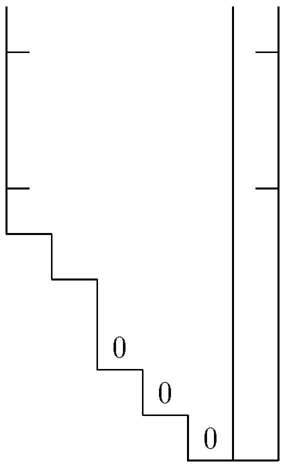
Caso II



Caso III



Caso IV



Caso V

Matlab

Histórico

Matlab, de Matrix Laboratory, é um ambiente interativo para computação envolvendo matrizes. Matlab foi desenvolvido no inicio da década de 80 por Cleve Moler, no Departamento de Ciência da Computação da Universidade do Novo México, EUA.

Matlab coloca à disposição do usuário, num ambiente interativo, as bibliotecas desenvolvidas nos projetos LINPACK e EISPACK. Estes projetos elaboraram bibliotecas de domínio público para Álgebra Linear. LINPACK tem rotinas para solução de sistemas de equações lineares, e EISPACK tem rotinas para cálculo de autovalores. Os manuais destes projetos são portanto documentação complementar à documentação do Matlab.

Versões posteriores de Matlab, atualmente na versão 4.0, foram desenvolvidas na firma comercial MathWorks Inc., que detêm os direitos autorais destas implementações. As versões recentes do produto Matlab melhoram significativamente o ambiente interativo, incluindo facilidades gráficas de visualização e impressão; todavia a "Linguagem Matlab" manteve-se quase inalterada. Existem vários interpretadores da linguagem Matlab em domínio publico, como Matlab 1.0, Octave e rlab. Existem também outros interpretadores comerciais de Matlab, como CLAM. Existem ainda várias Tool Boxes, bibliotecas vendidas pela MathWorks e por terceiros, com rotinas em Matlab para áreas específicas.

Usaremos a grafia de nome próprio, **Matlab**, como referência a linguagem, o nome em maiúsculas, **MATLAB**, como referência ao produto comercial da MathWorks, e a grafia em minúsculas, **matlab**, como referência a um interpretador genérico da linguagem Matlab.

O Ambiente

Para entrar no ambiente matlab, simplesmente digite "matlab". O prompt do matlab é `>>` , que espera por comandos. Para sair use o comando `quit`. Dentro do ambiente matlab, um comando precedido do bang, `!`, é executado pelo sistema operacional, assim: usando `!dir` ou `!ls` ficaremos sabendo os arquivos no diretório corrente, e usando `!edit` , `!vi` ou `!emacs` , seremos capazes de editar um arquivo. Normalmente Matlab distingue maiúsculas de minúsculas.

O comando `help` exibe todos os comandos e símbolos sintáticos disponíveis. O comando `help nomecom` fornece informações sobre o comando de nome *nomecom*. O comando `diary nomearq` abre um arquivo de nome *nomearq*, e ecoa tudo que aparece na tela para este arquivo. Repetindo o comando `diary` fechamos este arquivo. O formato dos números na tela pode ser alterado com o comando `format`.

Os comandos `who` e `whos` listam as variáveis em existência no espaço de trabalho. O comando `clear` limpa o espaço de trabalho, extinguindo todas as variáveis. O comando `save nomearq` guarda o espaço de trabalho no arquivo *nomearq*, e o comando `load nomearq` carrega um espaço de trabalho previamente guardado com o comando `save`.

Em Matlab há dois terminadores de comando: a vírgula, `,` , e o ponto-e-vírgula, `;` . O resultado de um comando terminado por vírgula é ecoado para a tela. O terminador ponto-e-vírgula não causa eco. Resultados ecoados na tela são atribuídos a variável do sistema `ans` (de answer, ou resposta). O terminador vírgula pode ser suprimido no último comando da linha. Para continuar um comando na linha seguinte devemos terminar a linha corrente com três pontos, `...` , o símbolo de continuação. O sinal de porcento, `%` , indica que o resto da linha é comentário.

Matrizes

O tipo básico do Matlab é uma matriz de números complexos. Uma matriz real é uma matriz que tem a parte imaginária de todos os seus elementos nula. Um vetor linha é uma matriz $1 \times n$, um vetor coluna uma matriz $n \times 1$, e um escalar uma matriz 1×1.

As atribuições
`A = [1, 2, 3; 4, 5, 6; 7, 8, 9];` ou
`A = [1 2 3; 4 5 6; 7 8 9];` ,
são equivalentes, e atribuem à variável A o valor

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

Matrizes são delimitadas por colchetes, elementos de uma mesma linha são separados por vírgulas (ou apenas por espaços em branco), e linhas são separadas por ponto-e-vírgula (ou pelo caracter nova-linha). O apóstrofe, `'` , transpõem uma matriz. É fácil em Matlab compor matrizes blocadas, desde que os blocos tenham dimensões compatíveis! Por exemplo:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \; , \; B = \begin{bmatrix} 5 \\ 6 \end{bmatrix} \; , \; C = \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \; , \; \texttt{D=[A,B;C'];} \; D = \begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Cuidado ao concatenar matrizes com os espacos em branco, pois estes são equivalentes a vírgulas, separando elementos. Assim: `[1,2+3]==[1 5]` mas `[1,2 +3]==[1,2,3]`.

Há várias funções para gerar matrizes e vetores especiais: `zeros(m,n)`, `ones(m,n)` e `rand(m,n)` são matrizes de dimensão $m \times n$ com, respectivamente, zeros, uns, e números aleatórios em [0,1]. O vetor `i:k:j` é o vetor linha $[i, i+k, i+2k, \ldots i+nk]$, onde $n = \max m \mid i+mk \leq j$. Podemos suprimir o "passo" $k = 1$, escrevendo o vetor `i:1:j` simplesmente como `i:j`. Se v é um vetor, `diag(v)` é a matriz diagonal com diagonal v, se A é uma matriz quadrada, `diag(A)` é o vetor com os elementos da diagonal principal de A. A matriz identidade de ordem n é `eye(n)`, o mesmo que `diag(ones(1,n))`.

`A(i,j)` é o elemento na i-ésima linha, j-ésima coluna de A, $m \times n$. Se vi e vj são vetores de índices em A, i.e. vetores linha com elementos inteiros positivos, em vi não excedendo m, e em vj não excedendo n, `A(vi,vj)` é a sub-matriz do elementos de A com índice de linha em vi e índice de coluna em vj. Em particular `A(1:m,j)`, ou `A(:,j)` é a j-ésima coluna de de A, e `A(i,1:j)`, ou `A(i,:)`, é a i-ésima linha de A. Exemplo:

$$A = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{bmatrix} \quad vi = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \quad vj = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \quad A(vi, vj) = \begin{bmatrix} 32 & 33 \\ 12 & 13 \end{bmatrix}$$

As operações de adição, subtração, produto, divisão e potência, `+ - * / ^`, são as usuais da álgebra linear. O operador de divisão à esquerda, `\`, fornece em `x = A\b;` uma solução $x \mid A*x = b$. O operador de divisão a direita, `/` , fornece em `a = b/A;` uma solução $x \mid x * A = b$. Quando o sistema é bem determinado isto é o mesmo que, respectivamente, `inv(A)*b` ou `b*inv(A)`. (Quando o sistema é super-determinado x é uma solução no sentido dos mínimos quadrados.) Os operadores aritméticos de produto, potência e divisão tem a versão pontual, `.* .^ ./ .\` , que são executados elemento a elemento. Exemplo:

$$x = \begin{bmatrix} 5 & 5 & 5 \end{bmatrix}, \quad y = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}, \quad x .* y = \begin{bmatrix} -5 & 0 & 5 \end{bmatrix}$$

Matlab é uma linguagem declarativa (em oposição a imperativa) onde as variáveis são declaradas, dimensionadas e redimensionadas dinamicamente pelo interpretador. Assim, se A presentemente não existe, `A=11;` declara e inicializa uma matriz real 1×1. Em seguida, o comando `A(2,3)=23;` redimensionaria A como a matriz 2×3 `[11, 0, 0; 0, 0, 23]`. A nome da matriz nula é `[ ]`. A atribuição `A(:,2)=[];` anularia a segunda coluna de A, tornado-a a matriz 2×2 `[11, 0; 0, 23]`.

Controle de Fluxo

Os operadores relacionais da linguagem são `< <= > >= == ~=`, que retornam valor 0 ou 1 conforme a condição seja verdadeira ou falsa. Os operadores lógicos, *não e ou*, são, respectivamente, `~ & |` .

Matlab possui os comandos de fluxo *for – end*, *while – end* e *if – elseif – else – end*, que tem a mesma sintaxe do Fortran, exemplificada a seguir. Lembre-se de **não** escrever a palavra `elseif` como duas palavras separadas.

```
for i=1:n
  for j=1:m
    H(i,j)=1/(i+j-1);
  end
end
```

```
if(x<0)
  signx=-1;
elseif(x>0)
  signx=1;
else
  signx=0;
end
```

```
fatn=1;
while(n>1)
  fatn=fatn*n;
  n=n-1;
end
```

Uma consideração sobre eficiência: Matlab é uma linguagem interpretada, e tempo de interpretação de um comando simples pode ser bem maior que seu tempo de execução. Para tornar o programa rápido, tente operar sobre matrizes ou blocos, evitando loops explícitos que operem numa matriz elemento por elemento. Em outras palavras, tente evitar loops que repetem um comando que atribui valores a elementos, por atribuições a vetores ou matrizes. As facilidades no uso de vetores de índices, os operadores aritméticos pontuais, e funções como `max` , `min` , `sort` , etc., tornam esta tarefa fácil, e os programas bem mais curtos e legíveis.

Scripts e Funções

O fluxo do programa também é desviado pela invocação de subrotinas. A subrotina de nome *nomsubr* deve estar guardada no arquivo `nomsubr.m`; por isto subrotinas são também denominadas M-arquivos (M-files). Há dois tipos de subrotinas: Scripts e Funções.

Um script é simplesmente uma seqüência de comandos, que serão executados como se fossem digitados ao prompt do matlab. Subrotinas podem invocar outras subrotinas, inclusive recursivamente.

Um M-arquivo de função em Matlab começa com a declaração da forma
`[ps1, ps2,... psm] = nomefunc( pe1, pe2,... pen )`
A lista entre parênteses é a lista de parâmetros de entrada da função, e a lista entre colchetes são os parâmetros de saída. Parâmetros são variáveis locais, assim como são variáveis locais todas as variáveis no corpo da função.

Ao invocarmos a função *nomefunc* com o comando
`[as1,... asm] = nomefunc(ae1,... aen);`
Os argumentos de entrada, $ae1, \ldots aen$, são passados por valor aos (i.e. copiados nos) parâmetros de entrada, e ao fim do M-arquivo, ou ao encontrar o comando `return` , os parâmetros de saída são passados aos argumentos de saída. Exemplos:

```
function [mabel, mabin] = vmax(v)
% procura o maior elemento, em valor absoluto,
% dentro de um vetor, linha ou coluna.
%Obs: Esta funcao NAO segue os conselhos
```

```
%para operar sobre blocos, e nao elementos!

[m,n]=size(v);
if( m ~= 1  &  n ~= 1 )
  erro;
else
  K=max([m,n]);
  mabel= abs(v(1)); mabin= 1;
  for k=2:K
    if( abs(v(k)) > mabel )
      mabel= abs(v(k)); mabin= k;
    end%if
  end%for
end%else
```

Para referir-mo-nos, dentro de uma função, a uma variável externa, esta deve ter sido declarada uma variável global com o comando `global`. A forma de especificar uma variável como global muda um pouco de interpretador para interpretador, e mesmo de versão para versão: Diga `help global` para saber os detalhes de como funciona a sua implementação.

Bibliografia

[Colema-88] T.F.Coleman and C.F.van Loan. *A Matrix Computation Handbook*. SIAM Publications, Philadelphia.

[Moler-81] C.B.Moler. *Matlab Manual*. Department of Computer Science, University of New Mexico.

[Dongar-79] J.J.Dongarra, J.R.Bunch, C.B.Moler, G.W.Stewart. *LINPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia.

[Smith-76] B.T.Smith, J.M.Boyle, J.J.Dongarra, B.S.Garbow, Y.Ikebe, V.C.Klema, C.B.Moler. *Matrix Eigensystem Routines: EISPACK Guide*. Lecture Notes in Computer Science, volume 6, second edition, Springer-Verlag.

[Garbow-77] B.S.Garbow, J.M.Boyle, J.J.Dongarra, C.B.Moler. *Matrix Eigensystem Routines: EISPACK Guide Extension*. Lecture Notes in Computer Science, volume 51, Springer-Verlag.

[Moler-92] C.B.Moler, J.N.Litte and S.Bangert. *PC-MATLAB User's Guide*. The MathWorks Inc. Sherborn, Massachusetts.

[Eaton-92] J.W.Eaton. *Octave Manual*. Chemical Engineering Department, University of Texas at Austin. Austin, Texas.

Bibliografia

Bibliografia SUMARIA de suporte para o curso:

[Bertsekas-89] D.P.Bertsekas J.N.Tsitsiklis. *Parallel and Distributed Computation, Numerical Methods.* Prentice Hall, 1989.

[Bunch-76] J.R.Bunch D.J.Rose. *Sparse Matrix Computations.* Academic Press, 1976.

[Carey-89] G.F.Carey. *Parallel Supercomputing.* John Wiley, 1989.

[Dongarra-91] J.J.Dongarra et all. *Solving Linear Systems on Vector and Shared Memory Computers.* SIAM, 1991.

[Duff-79] I.S.Duff G.W.Stewart. *Sparse Matrix Proceedings 1978.* SIAM, 1979.

[Duff-86] I.S.Duff A.M.Erisman J.K.Reid. *Direct Methods for Sparse Matrices.* Oxford, 1986

[Gallivan-90] K.A.Gallivan et all. *Parallel Algorithms for Matrix Computations.* SIAM, 1990.

[George-81] A.George J.W.H.Liu. *Computer Solution of Large Sparse Positive-Definite Systems.* Prentice Hall, 1981.

[Golub-83] G.H.Golub C.F.van Loan. *Matrix Computations.* John Hopkins, 1983.

[Pissan-84] S.Pissanetzky. *Sparse Matrix Technology.* Academic Press 1984.

[Rose-72] D.J.Rose R.A.Willoughby. *Sparse Matrices.* Plenum Press, 1972.

[Stern-92] J.M.Stern. *Simulated Annealing with a Temperature Dependent Penalty Function.* ORSA Journal on Computing, V-4, N-3, p-311-319, 1992.

[Stern-93] J.M.Stern S.A.Vavasis. *Nested Dissection for Sparse Nullspace Bases.* SIAM Journal on Matrix Analysis and Applications, V-14, N-3, p-766-775, 1993.

[Stern-94] J.M.Stern S.A.Vavasis. *Active Set Algorithms for Problems in Block Angular Form.* Computational and Applied Mathematics, V-13, 1994.

[Stewart-73] G.W.Stewart. *Introduction to Matrix Computations.* Academic Press, 1973.

[Tewarson-73] R.P.Tewarson. *Sparse Matrices.* Academic Press, 1973.

[Vorst-89] H.A.van der Vorst P.van Dooren. *Parallel Algorithms for Numerical Linear Algebra.* North Holland, 1989.