PRODUTOS DE KRONECKER, SIMETRIZADORAS E ALGORÍTMOS PARALELOS E SEQUENCIAIS NA ÁLGEBRA LINEAR

KARABI DATTA

Orientador

Prof. Dr. TENKASI M. VISWANATHAN

Dissertação apresentada no Instituto de Matemática, Estatística e Ciência da Computação, como requisito parcial para obtenção do título de Doutor em Matemática.

Agosto - 1982

AGRADECIMENTOS

Ao Professor T.M. Viswanathan, pelos novos rumos que a tese tomou na sua fase final e pelo apoio constante e orientação ao longo dos últimos dois anos.

Ao Professor José V. Zago, pelas discussões e leitura cuidadosa do manuscrito.

Ao Professor Janos Simon, pela proposta do estudo de algorítmos paralelos e pela sua orientação inicial.

Ao Professor Roberto Cignoli pela sua compreensão e colaboração.

Ao meu marido, Biswa N. Datta, pela paciência e sacrifício no decorrer deste trabalho e, particularmente, durante o período do meu afastamento dos familiares.

Â minha amiga Prem, pelo apoio moral e emocional durante as fases críticas.

Â minha amiga Maria de Lourdes, pelo excelente trabalho de datilografia da tese.

Aos colegas de Pós-Graduação pelo incentivo dado no decorrer deste trabalho.

Â FAPESP, pelo auxílio financeiro.

NOTAÇÕES

NOTAÇÃO		EXPLICAÇÃO
log n	-	$\log_2 n$, todos os algoritmos são tomados sobre base 2.
[n]	-	o menor número interior maior ou igual a' n'
F	-	Corpo Numérico
E	-	Anel Comutativo de Matrizes de ordem m.
0(f(n))	-	Ordem de f(n)
R	-	Corpo dos Números Reais.
C	-	Corpo dos Números Complexos.

ÍNDICE

CAPÍTULO I

INTRODUÇÃO

Esta tese nasceu de uma tentativa de desenvolver algorítmos em paralelo na álgebra linear. Nesta tentativa constatamos à nossa surpresa o papel importante que o produto de Kronecker de duas matrizes B e A pode desempenhar em algorítmos paralelos. Ele está presente em quase toda tese quer no conceito da matriz-determinante do Capítulo III, quer na equação matricial LA - BL = R do Capítulo V ou nas equações HA + A*H = N do Capítulo VI.

Muitos algorítmos nossos são baseados em simetrizadoras e a implementação em paralelo utiliza a matriz-determinante do produto de Kronecker.

Como a nossa atenção é voltada para os aspectos computacionais, quase sempre trabalhamos com uma matriz $A = (a_{ij})$ de Hessenberg inferior em forma normalizada:

$$a_{ij} = 0 \quad \text{se} \quad j > i+1 \quad \text{e} \quad a_{i,i+1} = 1.$$

Isto é equivalente a dizer que A é uma matriz não-derrogatória ou seja o polinômio característico de A é igual ao polinômio mínimo de A. O último fato já garante que a menos de similaridade uma matriz arbitrária pode ser escrita como soma direta de matrizes de Hessenberg inferiores em forma normalizada. Esta redução consta também no Capítulo IV. Computacionalmente é interessante trabalhar

com matrizes de Hessenberg, pois uma matriz arbitrária pode ser transformada em uma matriz de Hessenberg por similaridade, usando o método eficiente e numericamente estável de Householder [41].

Muitos algorítmos podem ser executados mais rapidamente se certos passos são executados ao mesmo tempo. A idéia de usar paralelismo para melhorar o tempo de execução de programas é muito atrativa; em certos casos, n processadores serão n vezes mais rápidos do que um único processador. Ao mesmo tempo, certos algorítmos não se prestam a este tipo de execução, e parecem inerentemente sequenciais: a disponibilidade de um número arbitrário de processadores não resulta em melhor tempo de execução. A investigação deste fenômeno é interessante, tanto do ponto de vista teórico, quanto na prática. A questão: qual é a melhora que pode ser conseguida, em geral, ao se passar de um modelo sequencial para um modelo paralelo de computação é uma das grandes perguntas em aberto da Teoria da Computação. Do ponto de vista prático, vários computadores paralelos foram construidos nos últimos anos. Alguns destes são "super-computadores", projetados para resolver rapidamente problemas extremamente complexos (O ILLIAC IV [25] e o STARAN [25] são alguns exemplos das primeiras máquinas deste tipo). Outras máquinas foram projetadas para tentar explorar o baixo custo de mini e microcomputadores (o computador experimental C*, em funcionalmento, é um exemplo [33]). O baixo custo de novas tecnologias (VLSI) permitirá, em breve, a construção de computadores altamente paralelos, com um número muito grande de processadores (10.000 ou mais) a um preço atrativo.

Existem ainda vários obstáculos à construção e uso destas máquinas: os problemas de sincronização, interconexão e resistência a falhas em processadores são algumas das questões importantes ainda por resolver. Há, no entanto, um outro problema básico: a inexistência de algorítmos paralelos para muitos problemas. Os algorítmos sequenciais existentes, não se prestam, em geral, à execução paralela. Nossa instrução e nossas técnicas de prova, que são a base para a síntese de algorítmos, são muitas vezes sequenciais. Assim, o problema de desenvolver algorítmos paralelos, é importante e não-trivial, e uma melhor compreensão do problema é necessária.

No Capítulo II, estudamos os algorítmos em paralelo existentes. Para nós, o mais importante desses algorítmos é um algorítmo de Csanky para resolução do sistema: $Ax = b$. Csanky [8] mostrou que o sistema $Ax = b$ pode ser resolvido em paralelo usando somente $O(\log^2 n)$ passos, onde n é a ordem do sistema. O mesmo número de passos é suficiente para inverter uma matriz A de ordem n ou computar o determinante de A.

O Capítulo III parte do princípio que multiplicação de matrizes em paralelo é uma coisa simples, fazendo com que matrizes sejam vistas como números. Assim trabalhamos sobre um anel comutativo E de matrizes $m \times m$ sobre um corpo F. Se A é uma matriz $n \times n$ sobre E, então indicamos por matriz-determinante de A (M-det A) o determinante de A sobre E. Gostaríamos de ter conseguido a computação da M-det A à maneira de Csanky. Há dificuldades técnicas nesta direção e conseguimos fazê-lo apenas quando a matriz A é

uma matriz em forma de blocos de um produto de Kronecker. Se A e B são duas matrizes $m \times m$ e $n \times n$ respectivamente, então a matriz $nm \times nm$ indicada por $B \otimes I_m - I_n \otimes A$ é chamada do produto de Kronecker de B e A. Note que se $A = (a_{ij})$ e $B = (b_{ij})$ então o produto de Kronecker de B e A pode ser representada em blocos por

$$K = \begin{bmatrix} b_{11}I-A & b_{12}I & \cdots & b_{1n}I \\ b_{21}I & b_{22}I-A & & b_{nn}I \\ & \cdots & & \\ b_{n1}I & \cdots & & b_{nn}I-A \end{bmatrix}$$

(onde I é a matriz identidade $m \times m$).

Assim K é uma matriz $n \times n$ sobre o anel $E = k[A]$ e a matriz determinante de K pode ser computada usando o método de Csanky. Esta computação é importante para a implementação em paralelo de vários algorítmos desenvolvidos nos Capítulos IV, V e VI.

No Capítulo IV tratamos de simetrizadoras de uma matriz de Hessenberg. Uma matriz X é dita simetrizadora de uma matriz A se, X é simétrica e

$$XA = A^tX \tag{1.1}$$

Simetrizadoras tem um papel significante na resolução do problema de autovalores. Por exemplo, com o conhecimento de simetrizadoras, o problema de autovalores para uma matriz não simétrica

A, (isto é $Ax = \lambda x$) pode ser transformada no seguinte problema de autovalores para matrizes simétricas:

$$XAx = \lambda Xx$$

ou seja

$$Cx = \lambda Xx \qquad (1.2)$$

onde ambas as matrizes C e X são simétricas. Em particular, se a simetrizadora X é positiva definida, pode se mostrar [10] que (1.2) reduz-se ao problema padrão para matrizes simétricas da forma

$$Ez = \lambda z$$

onde E é uma matriz simétrica. Assim, neste caso, o problema de computar os autovalores de uma matriz não simétrica A reduz-se ao problema de computar os autovalores de uma matriz simétrica.

Taussky e Zassenhaus [38] mostraram em 1959 que, se uma matriz A é não-derrogatória, cada solução X de (1.1) é simétrica, e que existem simetrizadoras de uma matriz não-derrogatória A. Todavia, não havia na literatura um algorítmo numérico para computar simetrizadoras. Em 1972, Datta [10] propôs um algorítmo para computar uma classe de simetrizadoras não singulares. Observou-se que o algorítmo é rápido, e estável numericamente [10].

Um dos resultados mais importantes do Capítulo IV é o Teorema 4.2, já provado por Taussky e Zassenhaus [38] dando uma decomposição canônica de uma simetrizadora X na forma $T_oP(A)$, onde T_o têm uma forma especial e P(A) é uma matriz polinômio em A. A

unicidade desta decomposição parece nova e ela é usada no Capítulo V (Teorema 5.3) para achar um algorítmo novo para a computação do polinômio característico de uma matriz de Hessenberg.

Uma das boas aplicações de simetrizadoras se encontra no Capítulo V (Teorema 5.2), onde damos um algorítmo para que duas matrizes A e B não tenham nenhum autovalor em comum. Este problema surge para matrizes sobre R ou C em várias situações práticas.Por exemplo, sabe-se [26] que a equação matricial

$$XA - BX = C \tag{1.3}$$

admite uma solução única se, e somente se, A e B não têm nenhum autovalor em comum. A equação matricial surge em várias aplicações de Física e Mecânica.

A simetrizadora S é construida seguindo o Algorítmo de Datta do Capítulo IV, enquanto a última linha s_n de S é computada usando a equação matricial

$$LA - BL = R;$$

a matriz R tem suas primeiras (n-1) linhas nulas, enquanto a solução L é uma matriz triangular inferior com diagonal unitária. A teoria atrás desta construção se baseia mais uma vez no produto de Kronecker. Um aspecto significativo do nosso algorítmo é o seguinte: *não computamos os polinômios característicos de A e B.*

Ainda no Capítulo V, damos um algorítmo muito interessante (Teorema 5.3) para a computação do polinômio característico de

uma matriz de Hessenberg B em forma normalizada. Ela se baseia na computação da solução L da equação matricial LA - BL = R, onde A é escolhida como a matriz nilpotente:

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

Mais uma vez no Capítulo VI destacamos o papel de simetrizadoras na localização de raízes de polinômios - um problema importante no estudo da estabilidade do sistema de equações diferenciais $\dot{x}(t) = Ax(t)$.

Nestes problemas é importante saber o *número* de autovalores de A com partes reais positivas, negativas e nulas. Este problema é conhecido como o problema de inércia da matriz A. A resposta é dada em termos de autovalores de uma matriz hermitiana H associada a A, já que os autovalores de H podem ser computados pelo método de LDL^t ou pelo método de Jacobi. Entre todos os algorítmos na busca de uma matriz hermitiana H adequada, o mais citado é o método de Fujiwara (Teorema 6.6). Há um método recente de Carlson-Datta para a computação da inércia de uma matriz de Hessenberg A em forma normalizada. Mostramos que o método de Carlson-Datta é baseado na forma hermitiana de Fujiwara (Teorema 6.6). Mostramos

também que a simetrizadora S do Capítulo V é baseada na forma quadrática de Bezout (Teorema 6.4).

Há ainda o problema do círculo unitário, onde queremos saber o número de autovalores de uma matriz A dentro do círculo unitário. Aqui também existe um método de Fujiwara (Teorema 6.7) dando a resposta em termos do número de autovalores positivos de uma matriz hermitiana F_2, desde que a matriz A seja uma matriz companheira. Damos dois novos algorítmos para o caso onde A é uma matriz de Hessenberg em forma normalizada (Algorítmos 6.5.1 e 6.5.2). Estes algorítmos usam o nosso algorítmo do Capítulo V para computação do polinômio característico de A. Será muito interessante desenvolver um algorítmo baseado diretamente na forma hermitiana de Fujiwara denida por F_2.

Achamos importante salientar que as demonstrações dadas neste capítulo, inclusive dos teoremas de Fujiwara são baseadas na equação matricial TA - BL = R. Assim as provas são diferentes e mais atrativas do que as provas conhecidas. Para todos os algorítmos abordados temos uma versão em paralelo.

Os problemas propostos neste tese herdam paralelismo suficiente e os algorítmos que propomos possuem tempo de execução em paralelo de $O(\log^2 n)$, (sendo n as ordens das matrizes envolvidas) com um um número de processadores polinomial em n. Os algorítmos sequenciais correspondentes usam em contraste um tempo polinomial com um único processador.

CAPÍTULO II

UM ESTUDO DOS ALGORITMOS EM PARALELO EXISTENTES

2.1. INTRODUÇÃO.

Neste capítulo introduzimos alguns conceitos básicos de computação em Paralelo e fazemos um breve estudo dos algoritmos paralelos existentes de Álgebra Linear. Nosso estudo é baseado no artigo recente de Heller [21] e numa bibliografia preparada por Poole e Vogt [21].

Notamos aqui que ainda não existe nenhum algoritmo em paralelo na literatura para resolver o problema de Localização de raízes de um polinômio ou separação de autovalores de uma matriz. Nesta Tese tratamos destes problemas num Capítulo posterior.

Fazemos as seguintes hipótese básicas:

i. Temos acesso a um número de processadores idênticos. (embora em muitos casos daremos limitações ao número de processadores).

ii. Qualquer processador pode dar acesso à memória de qualquer outro processador (desde que isto não resulte em conflitos).

iii. Qualquer processador pode executar qualquer uma das quatro operações aritméticas em qualquer tempo, mas processadores diferentes podem fazer operações diferentes ao mesmo tempo.

iv. Não é gasto nenhum tempo para comunicação de dados entre processadores e memórias.

v. Instruções são sempre disponíveis para execução quando necessário.

vi. Cada operação gasta o mesmo tempo, a que nos referimos como um passo unitário.

Uma motivação básica para o desenvolvimento de algorítmos em paralelo é se conseguir uma maior velocidade de computação. Então, a eficiência de um algorítmo paralelo depende de quão rápido o algorítmo é em comparação com o algorítmo ordinário (sequencial) correspondente.

Nós definimos a velocidade de um algorítmo sequencial como o número de operações aritméticas necessárias ao processo, enquanto a velocidade de um algorítmo paralelo será o número de passos unitários. Um resultado básico para os dois é:

TEOREMA 2.1. *Se pelo menos q operações são necessárias para computar um número Q então qualquer algorítmo usando P processador para computar Q precisa de pelo menos* $m(P,q+1)$ *passos, onde* $m(P,N)$ *é aproximadamente*

$$m(P,N) = \frac{N}{P} + \log(P/2) \quad \text{para} \quad P < N$$

e

$$m(P,N) = \lceil \log N \rceil \quad \text{para} \quad P \geq N.$$

A demonstração do resultado acima pode ser encontrada em [21].

Se T_1 é o tempo de execução de um algorítmo (sequencial) e T_p o tempo de execução do mesmo algorítmo usando P processadores,

então

$$S_p = T_1/T_p$$

é uma medida da melhora do tempo de execução usando-se paralelismo e

$$E_p = S_p/P$$

é chamada a medida da eficiência.

2.2. ALGUNS RESULTADOS BÁSICOS.

2.2.1. Dado $\{x\}$, x^n pode ser computado em tempo $\lceil \log n \rceil$ usando-se apenas dois processadores.

DEMONSTRAÇÃO: Um processador computa sucessivamente $x^2, x^4, x^8, \ldots, x^{2r}, \ldots,$ enquanto outro processador acumula o produto das potências apropriadas de x à medida que elas são geradas. Cada termo no produto acumulado corresponde a um 1 na representação binária de n.

OBSERVAÇÃO: A computação de x^n precisa de log n passos mesmo que um número infinito de processadores seja usado. A mesma delimitação pode ser conseguida apenas com 2 processadores.

2.2.2. Sejam a_i números dados. A soma $\sum_{i=1}^{N} a_i$ pode ser computada em $\lceil (\log N) \rceil$ passos usando-se $\lceil \frac{N}{2} \rceil$ processadores.

DEMONSTRAÇÃO: Escreve-se

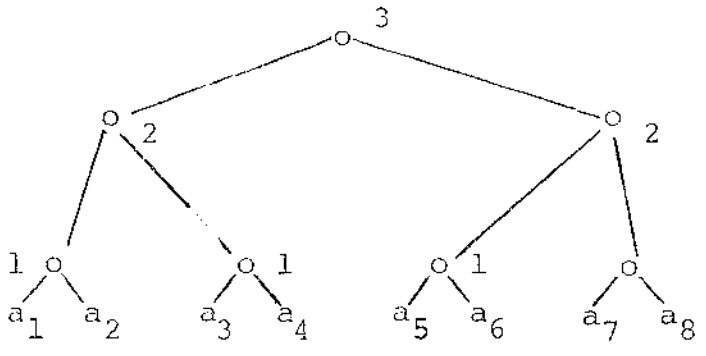
$$\sum_{i=1}^{N} a_i = \sum_{i=1}^{n-1} a_i + \sum_{i=n}^{N} a_i \quad , \quad n = \lfloor \frac{N}{2} \rfloor .$$

Aplicando essa decomposição repetidamente, a soma $\sum_{i=1}^{N} a_i$ é computada em $\lceil \log N \rceil$ passos e $\lfloor \frac{N}{2} \rfloor$ processadores são necessários.

2.2.3. Algorítmo do tipo "Fan-in" : Computação de recursão:

$A_N = a_1 \circ a_2 \circ , \ldots , a_N$ onde 'o' é uma operação aritmética (+ ou x) pode ser computado em $\lceil \log N \rceil$ passos se $\lfloor \frac{N}{2} \rfloor$ processadores são usados na computação.

DEMONSTRAÇÃO: Daremos uma demonstração no caso especial N = 8. O caso geral é análogo. A demonstração é melhor entendida seguindo-se a seguinte árvore:



2.2.4. Computação de Iterações

Dada a relação recursiva $y_i = f(y_{i-1})$ onde $f(y)$ é uma função racional e $y_o = x$, y_n pode ser computado em $0(\log n)$ passos.

DEMONSTRAÇÃO: Consideremos primeiro o caso quando $f(y)$ é linear, isto é, $f(y) = ay + b$, neste caso

$$y_n = a(\ldots a(ax + b) + b \ldots) + b$$

$$= a^n x + b \sum_{i=0}^{n-1} a^i$$

$$= a^n x + b \frac{(a^n - 1)}{a-1}$$

$$= a^n \left(x + \frac{b}{a-1}\right) - \frac{b}{a-1}$$

É claro que y_n pode ser computado em $[\log n] + 2$ passos usando n processadores.

Se f é em função linear de $y_{i-1}, \ldots, y_{i-m}$, a computação é mais complicada, mas ainda $0(\log n)$ passos são necessários [3].

2.2.5. Computação de uma função racional:

Seja $f(y)$ uma função racional e d = grau de $f(y)$. Então dada a relação recursiva

$y_i = f(y_{i-1})$ e a condição inicial $y_o = x$, pelo menos $\lfloor n \log d \rfloor$ passos são necessários para computar y_n.

DEMONSTRAÇÃO: O resultado segue do resultado anterior observando-se o fato de que o grau de y_n é d^n ou d^n+1 [3].

2.2.6. O Produto escalar de dois Vetores:

O produto escalar de dois vetores com n componentes pode ser computado em (log n) + 1 passos.

DEMONSTRAÇÃO: Computamos n multiplicações no primeiro passo, depois adicionamos os n produtos em (log n) passos.

2.2.7. O Produto de duas Matrizes:

O produto de duas matrizes A e B; A de ordem $m \times n$ e B de ordem $n \times p$ pode ser computado em (log n) + 1 passos.

DEMONSTRAÇÃO: A matriz do produto é de ordem $m \times p$ e cada componente dessa matriz é um produto escalar de dois vetores com n componentes. O resultado então segue do resultado anterior.

OBSERVAÇÃO: O resultado acima foi generalizado por Kuck e Maruyama [28] da seguinte maneira:

Se t_A, t_M e t_I passos são necessários para fazer adições, multiplicações e inversões de matrizes de ordem N, então qualquer expressão matricial envolvendo n matrizes de ordem N (sem inversão) pode ser computada em $[2 \log n]\ [t_A + t_M]$ passos. Se inversões são necessárias então,

$[\log n]\ (2t_A + t_M + t_I) - t_M + t_I$ passos são necessários.

2.3. COMPUTAÇÃO DE UM POLINÔMIO EM PARALELO: (a implementação do algorítmo de Horner).

O algorítmo de Horner para computar o valor de um polinômio p(x) num ponto dado foi um dos primeiros métodos implementados em paralelo.

Estrin [3] deu um algorítmo para computar p(x) em a proximadamente 2⌈log n⌉ passos.

Seja $$p(x) = \sum_{i=0}^{n} a_i x_i$$

então o algorítmo de Estrin computa recursivamente

$$p(x) = q(x)x^{\lceil n/2 +1\rceil} + r(x)$$

onde

$$q(x) = \sum_{i=0}^{\lceil n/2 -1\rceil} a_{i+\lceil n/2+1\rceil} x^i$$

e

$$r(x) = \sum_{i=0}^{\lceil n/2\rceil} a_i x^i$$

obviamente este processo precisa de 2⌈log n⌉ passos.

Dorn [3] descreveu um algorítmo que decompõe o polinômio em k sub-polinômios e computa cada polinômio usando o método padrão de Horner:

Seja $m = \lceil (n-i)/k \rceil$ e

$$q_i(x) = \sum_{j=0}^{m} a_{j+kj} x^{kj} \quad , \quad i = 0,\ldots,k-1$$

então $p(x) = \sum_{i=0}^{k-1} q_i(x)x^i$. Com k processadores este algorítmo pode ser implementado em $2n/k + 2\log k$ passos.

Munro e Paterson [3] mostraram que $p(x)$ pode ser computado em menos que $\log n + 0((\log n)^{1/2})$ passos.

O algorítmo pode ser descrito da seguinte maneira: Seja

$$D_r = \frac{1}{2} r(r+1) + 1, \quad \text{e}$$

$$p = \lceil \log(n+1) \rceil .$$

Suponha que

$$D_{r-1} < p \leq D_r .$$

O polinômio pode ser escrito na forma:

$$p(x) = q_o(x) + q_1(x)x^{2^{p-r}} + q_2(x)x^{2.2^{p-r}} +$$

$$+ \ldots + q_{2^r-1}(x)x^{(2^r-1).2^{p-r}}$$

onde $q_i(x)$ são polinômios de grau menor que 2^{p-r}.

Então para computar $p(x)$, computamos $\{q_i\}$, no passo seguinte multiplicamos por potências apropriadas de x e depois usamos mais r passos para adicionar 2^r termos.

Podemos mostrar por indução que o algorítmo precisa de (p+r+1)

passos no máximo.

PROVA: Para $r=0$ e $p=1$ e o polinômio de grau 1 pode ser computado em 2 passos.

Suponha que o resultado é verdadeiro para todos os graus menores que n; então $\{q_i\}$ pode ser computado em tempo

$$(p-r) + (r-1) + 1 = p$$

desde que

$$p-r \leq D_r - r = D_{r-1} .$$

As potências de x também são obtidas neste tempo. Então, o algorítmo precisa de $p+r+1$ passos. Como

$$(r(r-1))/2 < p$$

podemos afirmar que

o tempo total $\leq \log n+(2\log n)^{1/2} + 0(1)$.

2.4. COMPUTAÇÃO DA INVERSA DE UMA MATRIZ E SOLUÇÃO DE SISTEMA DE EQUAÇÕES:

Por algum tempo se acreditava que o processo de eliminação de Gauss-Jordan seria o melhor método para resolução (em paralelo) do sistema $Ax = b$ e inversão da matriz A.

Intuitivamente, é claro que qualquer algorítmo recursivo como o processo de eliminação precisa de pelo menos n passos.

Recentemente foi mostrado por Csanky [8] que o número de passos necessários para inverter uma matriz de ordem n, o número de passos para computar o determinante de uma matriz de ordem n e o número de passos para resolver n equações lineares com n incógnitas, são todos da mesma ordem de magnitude.

Em cada caso, são necessários, $0(\log^2 n)$ passos, dado um número finito de processadores. Nenhum resultado melhor do que esse foi encontrado ainda.

Abaixo nós explicamos um método de Csanky para resolução do sistema

$$Ax = b$$

Seja

$$f(z) = z^n + c_1 z^{n-1} \quad \ldots + c_{n-1} z + c_n.$$

O polinômio característico de A e seja $s_i = tr(A^i)$

i. Compute o vetor $(c_1, c_2, \ldots, c_n)^T$ resolvendo o sistema triangular

$$\begin{bmatrix} 1 & 0 & 0 \ldots 0 & 0 \\ s_1 & 2 & & \\ s_2 & s_1 & & \\ \vdots & & & \\ s_{n-1} & s_{n-2} & s_{\cdot} & n \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} -s_1 \\ \vdots \\ -s_n \end{bmatrix}$$

ii. Compute A^{-1} usando a fórmula

$$A^{-1} = -(A^{n-1} + c_1 A^{n-2} + \ldots + c_{n-2} A + c_{n-1} I)/\ n$$

iii. Compute $A^{-1}b$.

Para determinar o número de passos, notamos que A^k, $1 \leq k \leq n$ pode ser computado em $\log n\lceil(\log n+1)\rceil$ passos com $\lceil \frac{n(n^3)}{2} \rceil$ processadores. Como $tr(A) = \sum_{i=1}^{n} a_{ii}$, podemos computar s_i, $1 \leq i \leq n$ em $\log\ n$ passos usando n^2 processadores.

O sistema triangular pode ser resolvido em $[(\log n+1)]$, $[(\log n+2)/2]$ passos usando $\lceil \frac{n^3}{68} \rceil + 0(n^2)$ processadores e A^{-1} pode ser computado em $(\log^2 n+2)$ passos usando n^3 processadores. Finalmente, a computação de $x = A^{-1}b$ precisa de $\lceil(\log n+1)\rceil$ passos e n^2 processadores. Então, concluimos que, o processo precisa de apenas $0(\log^2 n)$ passos usando $0(n^4)$ processadores. Recentemente Preparata e Sarwate [31] mostraram que o número de processadores acima pode ser reduzido (mantendo o mesmo tempo de $0\log(n)$ passos), se a multiplicação de duas matrizes de ordem pode ser feito em paralelo em tempo $0(\log(n))$ usando $(n^2/\log n)$ processadores, para algum α real que satisfaz $2 < \alpha < 3$. O número de processadores neste caso é $2n^{\alpha+(1/2)} / (\log n)^2$.

Embora, o método do Csanky citado acima é realmente interessante teoricamente, ele depende da computação exata em todos os passos. Os erros de arredondamento cometidos na computação do traço de A causam problemas de instabilidade em muitos casos [41] .

Então, o método é instável numericamente.

Para se obter estabilidade, temos de usar os métodos padrões de eliminação: Gauss-Jordan, LU ou decomposição QR.

2.5. SOLUÇÃO DE SISTEMA TRIANGULAR

Nessa secção consideramos o problema de computar a solução do sistema $Ax = b$, em paralelo; onde A é uma matriz triangular. É fácil ver que a solução sequencial do sistema acima precisa de n^2 operações aritméticas. A solução em paralelo do sistema foi considerada primeiro por Heller [3] que mostrou que pode ser computada em $0(\log^2 n)$ passos com $0(n^4)$ processadores.

Este resultado foi melhorado depois. Agora sabemos que a solução x pode ser computada em $0(\log^2 n)$ passos usando-se $0(n^3)$ processadores. Na realidade é possível mostrar [21] que com algumas restrições, x pode ser computado em $0(\log^2 n)$ passos com o número de processadores abaixo de n^3.

Discutiremos agora mais alguns algorítmos.

1) Algorítmo (Borodin e Munro [3]):

Seja

$$A = \begin{bmatrix} A_1 & \\ A_2 & A_3 \end{bmatrix}$$

onde A_1 e A_3 são matrizes triangulares inferiores e A_2 de ordem n/2, então

$$A^{-1} = \begin{bmatrix} A_1^{-1} & 0 \\ -A_3^{-1}A_2A_1^{-1} & A_3^{-1} \end{bmatrix}$$

FASE I - Compute simultaneamente

$$A_1^{-1} \quad \text{e} \quad A_3^{-1} \quad \text{e resolva} \quad A_3 y = A_2 .$$

FASE II - Multiplique y por A_1^{-1} .

Então t(n), o tempo necessário para inverter uma matrix triangular A de ordem n de acordo com o algorítmo acima, é dado por:

$$t(n) \leq T(\tfrac{n}{2}) + \lceil 2\log n \rceil = O(\log^2 n).$$

Observe que o determinante de uma matriz triangular pode ser computado em paralelo em log n passos.

2) Outro é o método criado independentemente por Heller [21].

Suponha que A tem a diagonal unitária e seja A = I - L, onde L é uma matriz triangular inferior

$$x = A^{-1}b = (I - L)^{-1}.b \text{ , já que } L^n = 0$$

$$= (I + L + L^2 + \ldots + L^{n-1}) \, . \, b$$

$$= (I + L^{2p-1})(I + L^{2p-2}) \ldots (I + L)b$$

onde $p = \lceil \log n \rceil$.

x é computado fazendo-se o quadrado de L repetidamente e acumulando-se os produtos de acordo com a fórmula acima.

O algorítmo pode ser efetuado em p^2+p passos usando no máximo $n^2(n+1)$ processadores.

2.6. SOLUÇÃO DE SISTEMA TRIDIAGONAL:

O problema de resolver o sistema Ax = v onde A é uma matriz tridiagonal surge em várias aplicações práticas e por isso, daremos aqui um tratamento separado deste problema.

Na literatura existem várias maneiras de se resolver este problema em paralelo. O problema foi estudado primeiramente por Stone [21] que mostrou que o problema pode ser resolvido usando somente $0(\log n)$ passos.

Seja A = LDU; onde é matriz triangular inferior, D é matriz diagonal e U é matriz triangular superior. Os elementos de D são:

$$d_1 = b_1$$

$$d_j = b_j - a_j c_{j-1}/d_{j-1} \qquad 2 \leq j \leq n$$

$$\ell_j = a_j/d_{j-1} \qquad 2 \leq j \leq n \qquad \text{onde } T = \begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & c_3 & \\ & & \ddots & \ddots & \ddots \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{bmatrix}$$

$$u_j = c_j/d_j \qquad 1 \leq j \leq n$$

Stone sugere a seguinte maneira de computar os elementos da diagonal D (em paralelo).

Defina $p_o = 1$

$p_1 = b_1$

$p_j = b_j p_{j-1} - a_j c_{j-1} p_{j-2}$

$d_j = p_j / p_{j-1}$

Uma vez a matriz D é computada; as matrizes L e U são completamente determinadas por D e, o sistema Ax = v, pode então ser resolvido; resolvendo-se os dois sistemas bidiagonais:

$$Lw = v \quad e \quad Ux = D^{-1}w$$

A computação de D usando o método acima exige 0(log n) passos. Os sistemas bidiagonais podem ser resolvidos usando-se relações recursivas de primeiro grau, e então a solução de cada sistema precisa de 0(log n) passos. Finalmente $D^{-1}w$ pode ser computado em apenas um passo, usando-se n processadores. Então o número total de passos neste processo é de 0(log n).

O processo de Stone falha quando o pivotamento é necessário na fatorização de A. Isso pode ser evitado usando-se a fatorização QR de A, em vez da fatorização LDU. Num artigo recente, Sameh e Kuck [21] mostraram que a fatorização QR de uma matriz tridiagonal pode ser obtida em 0(log n) passos com n processadores. Então neste caso também, podemos resolver um sistema tridiagonal em 0(log n) passos usando n processadores.

Alternadamente, Traub [39] sugeriu dois métodos iterativos:

O método LR em paralelo e o método de Gauss em paralelo. Descrevemos abaixo o método de Gauss:

O Método de Gauss em Paralelo:

Suponha que os elementos da diagonal principal de A são não nulos. De fato, podemos assumir sem perda de generalização que eles são unitários.

Seja $A = A_L + I + A_R$

onde A_L é uma matriz cujos elementos não nulos são somente os que ficam na primeira subdiagonal inferior e os elementos não nulos de A_R ficam na primeira subdiagonal superior.

Sejam $E^{(o)}$, $f^{(o)}$ e $x^{(o)}$ dados

Então

i) $(I - A_L E^{(i-1)})E^{(i)} = A_R$, $i = 1 \ldots M$

ii) $(I - A_L E^{(M)})f^{(i)} = v - A_L f^{(i-1)}$, $i = 1 \ldots N$

iii) $x^{(i)} = f^{(N)} - E^{(M)} x^{(i-1)}$, $i = 1 \ldots P$

dados $e_j^{(o)}$, $j = 2, \ldots, n-1$

$$e_1^{(i)} = c_1 \text{ , } i = 0,1 \ldots M$$

iv) $$e_j^{(i)} = \frac{c_j}{1 - a_j e_{j-1}^{(i-1)}} \text{ , } \quad \begin{matrix} i = 1 \ldots M \\ j = 1 \ldots n-1 \end{matrix}$$

dados $f_j^{(o)}$, $j = 2 \ldots n$

$f_1^{(i)} = b_1$, $i = 0 \ldots N$

v) $$f_j^{(i)} = \frac{b_j - a_j f_{j-1}^{(i-1)}}{1 - a_j e_{j-1}^{(M)}}, \quad \begin{matrix} i = 1 \ldots N \\ j = 2 \ldots n \end{matrix}$$

dados $x_j^{(o)}$ $j = 1, \ldots m-1$

$x_m^{(i)} = f_m$, $i = 0, 1 \ldots p$

vi) $$x_j^{(i)} = f_j^{(N)} - e_j^{(M)} x_{j+1}^{(i-1)}, \quad \begin{matrix} i = 1, \ldots p \\ j = 1, \ldots n-1 \end{matrix}$$

É claro que o algorítmo acima é um algorítmo paralelo, pois todos as componentes de $E^{(i)}$ podem ser computadas em paralelo a partir das componentes de $E^{(i-1)}$.

É interessante observar que o algorítmo paralelo acima nos permite resolver um sistema tridiagonal num tempo independente de n, o número de equações e incógnitas. O tempo depende somente da dominância diagonal da matriz A, os erros iniciais e acuidade final exigida. Mas ainda, a norma do erro é reduzida a cada passo da iteração.

CAPÍTULO III

A MATRIZ DETERMINANTE E O PRODUTO DE KRONECKER

3.1. INTRODUÇÃO

Como apontamos na Introdução, nossa meta é descobrir novos algorítmos para implementação em paralelo. Diferentemente de implementação sequencial, a multiplicação de duas matrizes envolve apenas (log n+1) passos usando-se n^4 processadores. Isto sugere que em paralelo, matrizes podem ser vistas como simples números o que a multiplicação de matrizes exige quase o mesmo número de passos que a multiplicação de números comuns. Dado este ponto de vista, o antigo corpo de números cede lugar para um anel adequado de matrizes. Assim podemos considerar uma matriz $n \times n$ cujas entradas são matrizes $m \times m$! Do ponto de vista prático isto conduz a uma simplificação enorme: uma matriz $n^2 \times n^2$ A' por exemplo pode ser vista como uma matriz $n \times n$ A, cada entrada de A sendo um bloco $n \times n$, desde que os blocos individuais comutem entre si.

Portanto vamos trabalhar sobre um anel comutativo E de matrizes $m \times m$ sobre um corpo F fechado algebricamente. Já existe na literatura [19] o conceito do determinante de matrizes sobre E. Repare que uma matriz $n \times n$ A sobre E pode ser vista também como uma matriz $nm \times nm$ A' sobre F. Temos o determinante de A sobre E o qual é uma matriz $m \times m$ em E; portanto chamamo-lo de *matriz determinante* de A. Por outro lado tem-se o determinante de A' sobre F,

o qual é um escalar. A Proposição 10 proporciona a importante ligação, entre os dois conceitos.

Do ponto de vista de aplicação o que é importante é a matriz $nm \times nm$ A'. É imprescindível que os blocos $m \times m$ de A' produzindo a matriz $n \times n$ A comutem entre si. É importante também que a equação característica de A se fatoriz linearmente sobre E. No final deste capítulo damos um exemplo, onde esta fatoração não é possível.

A maior aplicação desta técnica envolve o produto de Kronecker. Como apontamos na Introdução o produto de Kronecker de duas matrizes B e A de ordem n e m respectivamente se enquadra em nossos estudos idealmente, uma vez que os blocos comutam entre si, podendo assim trabalhar apenas com a matriz $n \times n$ K de blocos. Neste capítulo calculamos a matriz determinante de K em paralelo usando uma técnica direta que segue imediatamente dos resultados de Csanky.

3.2. A MATRIZ DETERMINANTE E SUAS PROPRIEDADES

Sejam F um anel comutativo e E um anel comutativo de matrizes $m \times m$ sobre F. O caso interessante é quando $E = k[A]$ sendo k um corpo e A uma matriz $m \times m$.

Seja A uma matriz $n \times n$ sobre E. Então $A = (A_{ij})$, onde cada A_{ij} é uma matriz $m \times m$ sobre F. Como os A_{ij} comutam podemos

falar no determinante de A. Observe que o determinante de A é um elemento de E - portanto uma matriz $m \times m$ sobre F. Por isso referimo-nos ao determinante de A como a *matriz determinante* de A e escrevemos M - det A.

Assim

$$M - \det A = |A_{ij}| = \sum_{\pi} (\operatorname{Sgn} \pi) A_{1\pi(1)} \cdots A_{m\pi(n)}$$

onde a soma é tomada sobre todas as permutações de elementos de

$$S = \{1, \ldots, n\}.$$

Queremos ressaltar que a matriz $n \times n$ A sobre E pode também ser vista como uma matriz $nm \times nm$ A' e referimo-nos a esta notação sem confusão.

Aqui apresentamos algumas propriedades da matriz determinante. As demonstrações podem ser encontradas em [19].

PROPOSIÇÃO 1:

$$M.\det A^{T} = M.\det A.$$

PROPOSIÇÃO 2: Se B é uma matriz obtida de A pela multiplicação de uma linha (coluna) de A por uma matriz $C \in E$ então

$$M.\det B = C(M.\ \det A).$$

PROPOSIÇÃO 3: Se a matriz B sobre E é obtida de A pela troca de quaisquer duas linhas (ou colunas) de A, então

$$M.\det B = -M \det A.$$

PROPOSIÇÃO 4: Se A tem duas linhas (ou colunas) iguais, então

$$M.\det A = 0.$$

PROPOSIÇÃO 5: Se B é uma matriz (sobre E) obtida de A adicionando-se um múltiplo de uma linha (coluna) com outra, então

$$M \det B = M \det A.$$

PROPOSIÇÃO 6: Se A tem uma linha (ou coluna) nula então

$$M.\det A = 0.$$

PROPOSIÇÃO 7: Se A e B são quaisquer duas matrizes sobre E, então

$$M.\det(A.B) = (M.\det A)(M.\det B) = M.\det(BA).$$

A proposição mostra a relação entre det A' e det(M.det A). Se n = 2, esta relação é mais forte e é bem conhecida:

PROPOSIÇÃO 8: Seja

$$A = \begin{bmatrix} X & Y \\ P & Q \end{bmatrix}$$

onde X, Y, P, Q são blocos de partição de E e são comutativas. Então

$$\det A' = \det(M \det A) ,$$

onde A' é a matriz $2m \times 2m$ associada a A.

DEMONSTRAÇÃO: A prova se encontra em [18, Vol. I].

PROPOSIÇÃO 9: Seja $A = (a_{ij})$ uma matriz $n \times n$ sobre anel comutativo E. Indicamos por c_{ij} o cofator de a_{ij} em A, isto é $c_{ij} = (-1)^{i+j} \det A_{ij}$ onde A_{ij} a matriz $(n-1) \times (n-1)$ obtida de A suprindo a i-ésima linha e a j-ésima coluna de A.

Seja

$$C = (c_{ij})$$

Então

$$\det A = \sum_{j=1}^{n} a_{ij} c_{ij}$$

e $AC^t = C^t A = (\det A) I_n$.

DEMONSTRAÇÃO: ver Godement [19] pp. 321-323.

O seguinte resultado mostra a relação entre o determinante da matriz $nm \times nm$ A' e a matriz determinante de A.

PROPOSIÇÃO 10: Sejam A uma matriz $n \times n$ no anel comutativo E e A' a matriz $nm \times nm$ definida por A sobre um corpo F. Então a matriz $nm \times nm$ A' é não singular se, e somente se a matriz $m \times m$ M det A em E é não-singular.

DEMONSTRAÇÃO: Escrevamos

$$A = (A_{ij})$$

onde $A_{ij} \in E$. Seja $C = (c_{ij})$ onde c_{ij} é o cofator de A_{ij} em A. Então pela Proposição 9

$$AC^t = (M - \det A) I_n ,$$

sendo I_n a matriz identidade $n \times n$ em E.

Podemos considerar cada uma destas matrizes como matrizes $nm \times nm$ sobre F e ainda continuamos a ter

$$A'(C^t)' = (M - \det A\ I_n)' .$$

Tomando-se determinantes tem-se

$$\det A' \det (C^t)' = (\det(M \det A))^n$$

Assim $\det A' \neq 0$ se, e somente se $\det(M \det A) \neq 0$ ou seja A' é não-singular se, e somente se $M - \det A$ é não singular.

OBSERVAÇÃO 1: Se $n = 2$ então

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad e \quad B = \begin{bmatrix} A_{22} & -A_{21} \\ -A_{12} & A_{11} \end{bmatrix}$$

$$AB^t = \begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda \end{bmatrix} = \Lambda^2 I_2 \quad \text{onde} \quad \Lambda = M - \det A.$$

Tomando-se determinantes, observe que

$$\det A' = \det(B^t)' \quad (2m \times 2m \text{ matrizes})$$

$$(\det A')^2 = (\det \Lambda) \ ;$$

logo $$\det \Delta = \pm \det A' \ .$$

Comparando-se o termo típico dos dois lados, tem-se:

$$\det \Delta = \det A' \ ,$$

ou seja $$\det \Lambda' = \det(M - \det A)$$

como já tínhamos observado na Proposição 8.

3.3. O PRODUTO DE KRONECKER

Vamos usar a Proposição 10 para provar de novo um Teorema clássico sobre um critério para duas matrizes $m \times m$ A e $n \times n$ B não terem nenhum auto-valor em comum. Mais adiante no Capítulo V voltaremos a este problema e daremos um outro critério computacionalmente mais interessante (Teorema 5.2).

Se A e B são duas matrizes $m \times m$ e $n \times n$ respectivamente, então a matriz $nm \times nm$ indicada por

$$B \otimes I_m \quad {}_n \otimes A$$

é chamada de produto de Kronecker de B e A. Note que se $A = (a_{ij})$ e $B = (b_{ij})$ então o produto de Kronecker de B e A pode ser representado em blocos por

$$K = \begin{bmatrix} b_{11} I & a_{12} I & \cdots & b_{1n} I \\ \vdots & & & \vdots \\ a_{n1} I & & & b_{nn} I \end{bmatrix} -$$

$$= \begin{bmatrix} A & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & A \end{bmatrix}$$

$$= \begin{bmatrix} b_{11} I-A & b_{12} I & \cdots & b_{1n} I \\ b_{21} I & b_{22} I-A & \cdots & b_{nn} I \\ & & \ddots & \\ b_{n1} I & \cdots & \cdots & b_{nn} I - A \end{bmatrix}$$

(onde I é a matriz identidade $m \times m$).

Note que K está na forma de blocos e que as entradas de K são matrizes $n \times n$ que comutam entre si, sendo pertencentes ao anel comutativo $k[A]$. Note também que a matriz determinante de K é $\varphi(A)$, onde $\varphi(x)$ é o polinômio característico de B. Pela Proposição 10 o produto de Kronecker (uma matriz $nm \times nm$) é não-singular se, e somente se a matriz determinante $\varphi(A)$ é não-singular. Ora, se $\lambda_1, \lambda_2, \ldots, \lambda_m$ são os valores característicos de A (no corpo complexo C ou num corpo k fechado algebricamente), então $\varphi(\lambda_1), \varphi(\lambda_2), \ldots, \varphi(\lambda_m)$ são os valores característicos de $\varphi(A)$ e $\varphi(A)$ é não-singular se, e somente se $\varphi(\lambda_i) \neq 0$ para todo $i = 1, 2, \ldots, m$. Note que $\mu_1, \mu_2, \ldots, \mu_n$ são valores característicos de B se, e somente se $\varphi(\mu_i) = 0$. Assim temos demonstrado o seguinte importante resultado clássico [2].

TEOREMA 3.1: *Duas matrizes complexas* n × nA e m×mB *nenhum autovalor em comum se, e somente se o produto de Kronecker é não-singular.*

3.4. OS MÉTODOS DE CSANKY PARA O *PRODUTO DE KRONECKER.*

Seja A uma matriz n × n sobre E. Então M $\det(A - \lambda I_n)$ será chamada de polinômio caracte rístico em λ de A. Observe que como trabalhamos sobre um anel , o *polinômio característico* de A indicada por $f(\lambda)$ é da forma

$$f_A^E(\lambda) = (-1)^n [I_m \lambda^n + C_{n-1}\lambda^{n-1} + C_{n-2}\lambda^{n-2} + \ldots + C_o]$$

onde os C_i são matrizes de E. A propriedade importante do polinômio característico de A é proporcionada pelo Teorema de Cayley-Hamilton:

Se $f_A(\lambda)$ é o polinômio característico da matriz n × nA, então $f_A(A) = 0$. Este fato é bem conhecido.

Esta é a equação que, como no Capítulo II conduzirá ao cálculo da inversa do produto de Kronecker K, se a inversa existe.

A maior dificuldade teórica é a não-válidade do Teorema Fundamental de Álgebra no anel E. Portanto não se pode fatorizar o polinômio $f(\lambda)$ sobre E.

Mesmo assim, pode-se aplicar os métodos de Csanky no caso importante do produto de Kronecker. Nossa observação consiste em aproveitar a forma em blocos do produto de Kronecker.

No fim deste Capítulo daremos um exemplo de um anel E de matrizes mostrando a falta da fatoração do $f(\lambda)$ sobre E.

Sejam B uma matriz $n \times n$ sobre F e A uma matriz $m \times m$ sobre E. Então a matriz $nm \times nm$ indicada por

$$K = B \otimes I_m - I_n \otimes A$$

$$= \begin{bmatrix} b_{11}I_m - A & b_{12}I_m & \cdots & b_{1n}I_m \\ b_{21}I_m & b_{22}I_m - A & \cdots & \\ \vdots & & & \vdots \\ b_{n1}I_m & \cdots & & b_{nn}I_m - A \end{bmatrix}$$

é o produto de Kronecker de B e A em forma de blocos.

Olhamos K como uma matriz $n \times n$ sobre o anel $E = F[A]$. Observe que as entradas K comutam entre si.

Seja $\phi_B^F(\lambda)$ o polinômio característico de B sobre F.

Como $\phi_B^F(\lambda) = \det \begin{bmatrix} b_{11}-\lambda & b_{12} & \cdots & b_{1n} \\ \vdots & & & \\ b_{n1} & b_{n2} & & b_{nn}-\lambda \end{bmatrix}$

$$\text{e} \qquad \phi_K^E(\lambda) = M\,\det \begin{bmatrix} b_{11}I-A-\lambda & b_{12}I & \cdots & b_{1n}I \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ b_{n1}I & b_{n2}I & \cdots & b_{nn}I-A-\lambda \end{bmatrix}$$

Segue logo que $\phi_K^E(\lambda) = \phi_B^F(\lambda+A)$ onde identificamos $\alpha \in F$ com $\alpha I_m \in E$. Assim

se $\phi_B^F(\lambda) = (-1)^n(\lambda^n+\beta_{n-1}\lambda^{n-1}+\ldots+\beta_o) \in F[\lambda]$

então $\phi_K^E(\lambda) = (-1)^n\{I_m(\lambda+A)^n+\beta_{n-1}I_m(\lambda+A)^{n-1}+\ \ldots\ +\beta_o I_m\}$.

Primeiramente estamos interessados na M-det do produto de Kronecker K. Repare que M-det $K = \phi_A^F(A)$ (entendida a identificação natural de F em E).

TEOREMA 3.2: *Seja K o produto de Kronecker de n×n e m×m matrizes B e A respectivamente. Então K pode ser visto como um elemento do elemento do anel* $E = F[A]$. *Se* $\phi(x)$ *é o polinômio característico de B sobre F, então* M-det K sobre $E = \phi(B)$. M-det K pode ser computado em paralelo em $0(\log^2 n)$ passos usando $0(n^4)$ processadores.

DEMONSTRAÇÃO:

$$M\text{-det } K = \phi_B^F(A) = (-1)^n\{A^n+\beta_{n-1}A^{n-1}+\ldots+\beta_o I_m\} \qquad (3.1)$$

Usando o método de Csanky diretamente podemos computar todos os β_i $(i = 0,1,2,\ldots,n-1)$ em $0(\log^2 n)$ passos com $0(n^4)$ processadores [8]. Para determinar o número de passos, notamos que A^k, $1 \leq k \leq n$ pode ser computado em $\log k\lceil \log m+1\rceil$ passos com $\lceil k \frac{n^3}{2}\rceil$ processadores. Então para computar M det k precisamos $0(\log^2 n)$ passos com $0(n^4)$ processadores.

Passamos a computar agora o polinômio característico de K sobre E. Como apontamos ele é

$$\phi_K^E(\lambda) = \phi_B^F(\lambda+A)$$

$$= (-1)^n\{I_m(\lambda+A)^n + \beta_{n-1}I_m(\lambda+A)^{n-1} + \ldots + \beta_o Im$$

$$= (-1)^n(I_m\lambda^n + D_{n-1}\lambda^{n-1} + \ldots + D_o\} \qquad (3.2)$$

onde os D_i são matrizes $m \times m$ em E.

Ora, D_i = coeficiente de $(-1)^n\lambda^i$

$$= \binom{n}{i}A^{n-i} + \beta_{n-1}\binom{n-1}{i}A^{n-1-i} + \ldots + \beta_j\binom{j}{i}A^{j-i}$$

$$+ \ldots + \beta_i\binom{i}{i}A^{i-1} \qquad (3.3)$$

onde $0 \leq i \leq n-1$.

A computação dos coeficientes D_i pode ser feita em paralelo por um método análogo em $0(\log^2 n)$ passos com máximo número de processadores $0(n^5)$. Feito isto, se A e B não tem nenhum autovalor

em comum, então a inversa de K em E existe e pode ser computado em $0(\log^2 mn)$ passos com $0([\frac{m.n}{2}].m^2.n^3)$ processadores no máximo . Por ver isto, basta observar que

$$K^{-1}(\text{em } E) = -(D_o)^{-1}(I_m K^{n-1} + D_{n-1}K^{n-2} + \ldots + D_1).$$

Podemos resumir estes resultados da seguinte maneira.

TEOREMA 3.3. *Os coeficientes do polinômio característico do produto de Kronecker* K *de* B *e* A *sobre* E = F[A] *podem ser computados em* $0(\log^2 n)$ *passos com* $0(n^4)$ *processadores também se* A *e* B *não têm nenhum autovalor em comum, então a inversa de* K *em* E *pode ser feita em* $0(\log^2 mn)$ *passos com* $0([\frac{m.n}{2}]\, m^2 n^3)$ *processadores.*

OBSERVAÇÕES: Recentemente, Preparata e Sarwate [31] mostraram que os limites de Csanky acima podem ser atingidos, usando somente $\frac{2n^{\alpha+\frac{1}{2}}}{(\log n)^2}$ processadores, se a multiplicação de duas matrizes (sobre um corpo F) pode ser feita em paralelo em tempo $0(\log n)$ usando $\frac{n^\alpha}{\log n}$ processadores para algum real satisfazendo $2 \leq \alpha \leq 3$.

Assim podemos reduzir o número de processadores citados acima se usarmos o resultado de Preparata e Sarwate.

3.5. UM EXEMPLO:

Se tivermos a fatoração de polinômios sobre E em fatores lineares, poderíamos ter trabalhado com qualquer matriz sobre E em

vez do produto de Kronecker. Por exemplo

$$\phi_K^E(\lambda) = (-1)^n \prod_{i=1}^{n} \{\lambda - (\lambda_i I_m - A)\}$$

onde os λ_i são valores característicos de B sobre F. Portanto, o seguinte exemplo é interessante.

EXEMPLO:

Seja $A = \begin{bmatrix} 0 & N \\ I_2 & 0 \end{bmatrix}$ onde $N = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

e N, I_2 em E.

$$\phi(\lambda) = M \det(\lambda I - A) = M \det \begin{bmatrix} \lambda & -N \\ -I_2 & \lambda \end{bmatrix}$$

$$= I_2\lambda^2 - N$$

Fácil ver que

$$\phi(A) = A^2 - N$$

$$= \begin{bmatrix} N & 0 \\ 0 & N \end{bmatrix} - \begin{bmatrix} N & 0 \\ 0 & N \end{bmatrix} = 0$$

Então $\phi(\lambda)$ é o polinômio característico de A, mas $\phi(\lambda)$ não tem fatorização sobre E, pois não existe nenhuma $B \in k[N]$ tal que $B^2 - N = 0$.

CAPÍTULO IV

SIMETRIZADORES

4.1. INTRODUÇÃO

A partir deste capítulo passamos a estudar matrizes simetrizadoras de matrizes de Hessenberg em forma normalizada (4.2). Vários de nossos algoritmos paralelos bem como sequênciais se baseiam no estudo e cálculo de simetrizadoras relevantes. A implementação em paralelo destes algorítmos se torna viável graças a computação da matriz determinante abordada no Capítulo anterior. Como destacamos na Introdução desta tese, o interesse de nossos algorítmos se deve ao fato que eles são implementados em $O(\log^2 n)$ passos, sendo n a ordem das matrizes em estudo.

Neste capítulo apresentamos primeiro um método de Datta [10] para achar uma família de simetrizadoras de uma matriz de Hessenberg em forma normalizada e em seguida a sua implementação em paralelo.

Seja X uma simetrizadora de uma matriz de Hessenberg A em forma normalizada. Os Teoremas 4.1 e 4.2 são interessantes também do ponto de vista técnico da álgebra linear. A cada simetrizadora X, associamos uma matriz polinômio P(A) (em A) e fornecemos uma decomposição de X na forma $X = T_o P(A)$ onde T_o é uma simetrizadora canônica de A gozando de uma propriedade simples. A unicidade desta decomposição é o conteúdo do Teorema 4.2.

Damos também um método recursivo para computar a matriz polinômio

P(A). Finalmente, damos um método para construir uma simetrizadora positiva definida de uma matriz companheira (quando existe) e apresentamos a sua implementação em paralelo. Encerramos o Capítulo com uma pergunta cuja resposta seria certamente interessante do ponto de vista teórico.

4.2. MATRIZES DE HESSENBERG E O PROBLEMA DE AUTOVALORES

Uma matriz $A = (a_{ij})$ é dita matriz de Hessenberg inferior (superior) se $a_{ij} = 0$ sempre que $i \geq j+2$ ($j \geq i+2$).

Uma matriz arbitrária pode ser transformada em uma matriz de Hessenberg por similaridade e essa transformação pode ser feita usando os métodos eficientes e numericamente estáveis de Householder [36] ou de Givens [36]. Assim, ao tratar o problema de autovalores ou problemas afins, pode-se assumir, sem perda de generalidade, que a matriz dada é uma matriz de Hessenberg . De fato, pode-se assumir ainda mais que a matriz de Hessenberg em questão não tem nenhum elemento nulo na codiagonal. Isto pode ser visto da seguinte maneira: Seja $A = (a_{ij})$ uma matriz de Hessenberg inferior. Sabemos que se todos os elementos na diagonal superior são zero, então A é uma matriz triangular inferior e os autovalores de A são os elementos da diagonal. Assim podemos nos restringir ao caso, onde A é uma matriz de Hessenberg com um ou mais elementos não-nulos na sua diagonal superior. Senão, A pode ser particionada na forma

$$A = \begin{bmatrix} A_1 & 0 \\ A_3 & A_2 \end{bmatrix}$$

onde A_1 e A_2 são matrizes de Hessenberg inferiores e a matriz A_1 é ou uma matriz 1×1 ou tem todos os elementos não-nulos na sua diagonal superior; se algum elemento na diagonal superior de A_2 é ainda nulo, podemos ainda particionar A_2 da maneira indicada acima; o processo pode ser continuado e depois de alguns passos finitos, chegaremos a uma partição:

$$A = \begin{bmatrix} A_{11} & & \\ A_{21} & A_{22} & 0 \\ \vdots & \ddots & \\ * & & A_{kk} \end{bmatrix}$$

onde as matrizes $A_{11}, A_{22}, \ldots, A_{kk}$ são ou matrizes 1×1 ou matrizes de Hessenberg inferiores, tendo todo elemento da diagonal superior não-nulo.

Ora,

$$\det(\lambda I - A) = \det(\lambda I - A_{11})\det(\lambda I - A_{22}) \ \ldots \ \det(\lambda I - A_{kk})$$

donde concluimos o seguinte: sabendo resolver o problema de autovalores de matrizes de Hessenberg tendo todo elemento não-nulo na sua codiagonal não nulo, sabemos também resolver o problema de autovalores

de matrizes de Hessenberg quaisquer. Feita esta observação, convém fazer a seguinte definição:

DEFINIÇÃO 1. Uma matriz $n \times n$ (com $n \geq 2$) inferior (superior) com com todo elemento na diagonal superior (inferior) não-nulo é chamada de matriz de *Hessenberg* não-reduzida.

Observe que, uma matriz de Hessenberg não-reduzida pode ser ainda mais reduzida a uma matriz de Hessenberg com codiagonal unitária (tendo todo o elemento da codiagonal igual a 1) via uma transformação diagonal de similaridade. Assim, se $A = (a_{ij})$ é uma matriz de Hessenberg inferior não-reduzida e

$$D = \text{diag}\left(\frac{1}{a_{12}a_{23}a_{34}\cdots a_{n-1,n}}, \frac{1}{a_{23}a_{34}\cdots a_{n-1,n}}, \ldots, \frac{1}{a_{n-1,n}}, 1\right)$$

então DAD^{-1} é uma matriz de Hessenberg inferior, tendo todo o elemento na diagonal superior unitário. Uma observação análoga vale sobre matrizes de Hessenberg superiores não-reduzidas. Isto nos conduz à seguinte definição:

DEFINIÇÃO 2: Uma matriz $n \times n$ (com $n \geq 2$) de Hessenberg inferior com codiagonal (superior) unitária será chamada de matriz de *Hessenberg normalizada*.

Nosso interesse em simetrizadoras se deve às suas aplicações aos problemas relacionados com autovalores. Em tais situações, portanto, podemos sempre trabalhar, face às observações feitas acima, com matrizes de Hessenberg normalizadas, quer inferiores, quer

superiores.

4.3.0 ALGORÍTMO DE DATTA PARA CONSTRUIR SIMETRIZADORAS DE MATRIZES DE HESSENBERG.

Seja

$$A = \begin{bmatrix} a_{11} & 1 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 1 & \cdots & 0 \\ \vdots & & & & \\ & & & & 1 \\ a_{n1} & \cdots & \cdots & \cdots & a_{nn} \end{bmatrix}$$

uma matriz de Hessenberg normalizada.

Sabemos que [38] cada solução X da equação matricial $XA=A^TX$ é simétrica e que a equação admite soluções se, A é uma matriz não-derrogatória.

Sejam $x_1, x_2, \ldots, x_n$ as linhas sucessivas de uma simetrizadora de A. Então segue o Algorítmo de Datta:

PASSO 1. Escolher x_n arbitrariamente

PASSO 2. Computar $x_{n-1}, x_{n-2}, \ldots, x_2$ e x_1 recursivamente por:

$$x_i = x_{i+1}A - a_{i+1i+1}\, x_{i+1} - \cdots - a_{ni+1}x_n \qquad (4.1)$$

onde $i = n-1, n-2, \ldots, 2, 1$.

DEMONSTRAÇÃO. A equação matricial é equivalente ao seguinte sistema de equações, ao notar que A^t é uma matriz de Hessenberg superior:

$$\begin{aligned} x_1A &= a_{11}x_1 + a_{21}x_2 + \ldots + a_{n1}x_n \\ x_2A &= x_1 + a_{22}x_2 + \ldots + a_{22}x_n \\ &\vdots \\ x_nA &= x_{n-1} + a_{nn}x_n \end{aligned} \tag{4.2}$$

Vamos olhar para estas equações como equações lineares com coeficientes no anel comutativo $K[A]$. Assim, se indicamos por I a matriz identidade $n \times n$, teríamos o seguinte sistema:

$$\left.\begin{aligned} x_1(a_{11}I-A) + x_2(a_{21}I) + \ldots + x_n(a_{n1}I) &= 0 \quad \ldots (1) \\ x_1I + x_2(a_{22}I - A) + \ldots + x_n(a_{n2}I) &= 0 \quad \ldots (2) \\ 0 + x_2I + \ldots + x_n(a_{n3}I) &= 0 \quad \ldots (3) \\ &\vdots \\ x_{n-1}I + x_n(a_{nn}I-A) &= 0 \ldots (n) \end{aligned}\right\} \tag{4.2}'$$

Seja B a matriz de coeficientes:

$$B = \begin{bmatrix} (a_{11}I - A) & a_{21}I & \cdots & a_{n1}I \\ I & (a_{22}I-A) & \cdots & a_{n2}I \\ 0 & I & \cdots & a_{n3}I \\ & & & \\ & I & & (a_{nn}I-A) \end{bmatrix} \qquad (4.3)$$

Observe que o sistema de equações (2), (3),...,(n) de (4.2') pode ser visto com um sistema em $x_1, x_2, \ldots, x_{n-1}$ transpondo-se os termos de x_n ao lado direito. A matriz $(n-1) \times (n-1)$ dos coeficientes deste novo sistema é:

$$D = \begin{bmatrix} I & a_{22}I - A & \cdots & a_{n-1,2}I \\ 0 & I & & a_{n-1,3}I \\ 0 & 0 \quad I & & a_{n-1,4}I \\ & \cdot \quad \cdot \quad \cdot & & \\ & & & \\ 0 & & 0 & I \end{bmatrix} \qquad (4.2)''$$

cuja M-determinante, é I. Assim, $x_1, x_2, \ldots, x_{n-1}$ são unicamente determinados por x_n. Ainda precisamos mostrar que a solução assim obtida satisfaz a equação (1) de (4.2)' .

Em notação matricial, o novo sistema é equivalente a:

$$D' = \left[\begin{array}{c|c} D & \begin{array}{c} -x_n a_{n2} I \\ -x_n a_{n3} I \\ \vdots \\ -x_n (a_{nn} I - A) \end{array} \end{array}\right]$$

enquanto (4.2)' é equivalente a

$$B' = \left[\begin{array}{c|c} \begin{array}{ccc} A_{11} I - A & a_{21} I \cdots & a_{n-1,1} I \end{array} & \begin{array}{c} -x_n a_{n1} I \\ -x_n a_{n2} I \\ \vdots \\ -x_n (a_{nn} I - A) \end{array} \\ D & \end{array}\right]$$

Ora, se multiplicamos a primeira, segunda,...,n-ésima linhas de B' pelos cofatores de $a_{n1} I$, $a_{n2} I, \ldots, (a_{nn} I - A)$ em B, respectivamente e somamos, obtemos o sistema equivalente:

$$\left[\begin{array}{c|c} \begin{array}{ccc} 0 & \cdots & 0 \end{array} & -x_n \varphi(A) \\ D & \begin{array}{c} -x_n a_{n2} I \\ \vdots \\ -x_n (a_{nn} I - A) \end{array} \end{array}\right]$$

(o novo sistema é equivalente, pois o cofator de $a_{n1} I$ é I o que é inversível no anel $k[A]$).

Ora $\varphi(X)$ é o polinômio característico de A^t e portanto também de A. Logo $-x_n \varphi(A) = 0$, pelo Teorema de Cayley-Hamilton, mostrando que a equação (1) do sistema (4.2)' é consistente com as demais equações. Assim, mostramos que x_n pode ser escolhido arbitrariamente e uma vez que x_n é dado as demais linhas, x_{n-1}, x_{n-2}, ..., x_1 são determinadas de (4.2) e satisfazem (4.1).

DEFINIÇÃO 3: Como uma simetrizadora X de uma matriz de Hessenberg inferior normalizada A é unicamente determinada pela sua última linha x_n, referimo-nos a X como *a simetrizadora de* A *associada ao vetor* x_n.

4.4. IMPLEMENTAÇÃO DO ALGORÍTMO DE DATTA EM PARALELO:

É claro que as fórmulas apresentadas em (4.1) não são adequadas para computar eficientemente simetrizadoras em paralelo.

Vamos agora reformulá-las.

PASSO I . Escolher x_n arbitrariamente.

PASSO II . De (4.2)' temos

$$x_{n-1} = -x_n (A_{nn} I - A)$$

$$x_{n-2} = +x_n \text{ M-det} \begin{bmatrix} a_{n-1n-1} I - A & a_{nn-1} I \\ I & a_{nn} I - A \end{bmatrix}$$

.

.

.

$$x_1 = (-1)^{n-1} x_n \cdot \text{Mdet} \begin{bmatrix} a_{22}I-A & a_{32}I & a_{42}I \ldots & a_{n2}I \\ I & (a_{33}I-A) & a_{43}I \ldots & a_{n3}I \\ 0 & I & (a_{22}I-A)\ldots & a_{n4}I \\ & & & \\ & & I & (a_{nn}I-A) \end{bmatrix}$$

$$= x_n \ \text{M-det}(B_1)$$

Ora, $T(n)$ é o número de passos para calcular a matriz-determinante B_1 cujos elementos são $n \times n$ matrizes sobre o corpo k, pertencendo ao anel comutativo $E = k[A]$ sendo k um corpo fechado algebricamente de característica 0 ou $p > n$. Assim, podemos apelar ao Teorema 3.1 com "n" = n-1 e "m" = n. Concluimos então que, $T(n) = "0(\log^2 mn)" = 0(\log^2 n)$ usando no máximo n^7 processadores. Portando a computação de uma simetrizadora pode ser feita em $0(\log^2 n)$ passos.

OBSERVAÇÃO 1:

Se $x_n = (\alpha, 0, 0, 0 \ldots 0)$, com $\alpha \neq 0$, então a simetrizadora resultante é não singular, pois neste caso a simetrizadora X tem a forma:

$$X = \begin{bmatrix} * & * & * & \cdots & * & \alpha \\ * & \cdots & \cdots & \cdots & *\alpha & 0 \\ & & & & & \vdots \\ & & & & & \vdots \\ \alpha & - & - & - & - & 0 \end{bmatrix}$$

4.5. MATRIZES POLINÔMIOS P(A) EM UMA MATRIZ DE HESSENBERG NORMALIZADA.

No que segue neste capítulo, supomos que k seja um corpo fechado algebricamente, A uma matriz $n \times n$ (com $n \geq 2$) de Hessenberg e que a característica p de k é maior que n ou $p = 0$. Muitas vezes esta condição sobre a característica p não é necessário.

PROPOSIÇÃO 4.1. *Sejam* A *uma matriz* $n \times n$ *de Hessenberg inferior normalizada e* $\varepsilon = (1,0,0,\ldots,0)$ *em* k^n. *Então os vetores* ε, εA, $\varepsilon A^2, \ldots, \varepsilon A^{n-1}$ *são linearmente independentes em* k^n. *Se* x_n *é um vetor qualquer de* k^n *então existe um polinômio* $P(X) \in k[X]$ *tal que* x_n *é a primeira linha da matriz polinômio* P(A) . *Mais ainda, o polinômio* P(X) *é unicamente determinado por esta propriedade , se o grau de* P(X) *for menor que ou igual a* n-1. *Se* A *é uma matriz* $n \times n$ *de Hessenberg superior normalizada, um resultado análogo vale com* $\varepsilon = (0,0,\ldots,1)$ *e com a última linha em vez da primeira linha.*

PROVA: Provamos o resultado para A uma matriz de Hessenberg inferior.

Observe que

$$(A)_{ij} = 1 \quad \text{se} \quad j = i+1$$

$$= 0 \quad \text{se} \quad j > i+1$$

$$(A)^2_{ij} = 1 \quad \text{se} \quad j = i+2$$

$$= 0 \quad \text{se} \quad j > i+2$$

$$(A)^k_{ij} = 1 \quad \text{se} \quad j = i+k \qquad \text{onde} \quad i \geq 1;\ i,k \leq n$$

$$\text{se} \quad j > i+k$$

Seja

$$\varepsilon = (1,0,\ldots,0),$$

então

$$\varepsilon A = (*,1,0,\ldots,0)$$

$$\varepsilon A^2 = (*,*,1,0,\ldots,0)$$

$$\vdots$$

$$\varepsilon A^{n-1} = (*,*,*,\ldots,*,1)$$

É evidente que todos os vetores ε, εA, $\varepsilon A^2,\ldots,$ εA^{n-1} são linearmente independentes em k^n. Eles são primeiras linhas de I, A, $A^2,\ldots,A^{n-1}$.

Ora, se $x_n = a_o\varepsilon + a_1\varepsilon A + \ldots + a_{n-1}\varepsilon A^{n-1}$, então simplesmente tomamos $P(X) = a_o + a_1x + \ldots + a_{n-1}x^{n-1}$ e $\varepsilon \cdot P(A) = x_n$.

Como $\varepsilon \cdot P(A)$ é a primeira linha de $P(A)$, o resultado segue.

Para a segunda parte, observe que o polinômio $P(X)$ pode ser escolhido tal que o grau $\partial P \leq n-1$. Se $Q(X) = b_o + b_1 x + \ldots + \ell_{n-1}$ é é um outro polinômio com grau $Q \leq n-1$, tendo x_n como a primeira linha, então $x_n = \varepsilon Q(A) = b_o\varepsilon + b_1\varepsilon A + \ldots + b_{n-1}\varepsilon A^{n-1}$. O fato de $\varepsilon, \varepsilon A, \ldots, \varepsilon A^{n-1}$ determinarem uma base de k^n mostra que $a_i = b_i$ para todo $i = 0,1,2,\ldots,n-1$. Assim $P(X) = Q(X)$.

TEOREMA 4.1. [13] *Seja* A *uma matriz* n × n *de Hessenberg inferior normalizada e* x_n *um vetor não nulo.*

Defina recursivamente os vetores

$$p_1 = x_n$$

$$p_{i+1} = p_i B_i - \sum_{j=1}^{i-1} a_{ij} p_j \quad , \; i=1,2,\ldots,n-1 \ldots \quad (4.6)$$

onde $B_i = A - a_{ii} I$.

Então existe um polinômio $P(X) \in k[X]$ *tal que* $P(A)$ *é a matriz cujas linhas são* $p_1, p_2, \ldots, p_n$.

DEMONSTRAÇÃO. Indique por a_i e e_i as i-ésimas linhas de A e I respectivamente. Pela Proposição 4.1 existe um polinômio $P(X) \in K[x]$ tal que a primeira linha de $P(A)$ é

$p_1 = x_n$. Ora,

$$p_2 = p_1 B_1 = e_1 P(A) B_1$$

$$= e_1 B_1 P(A) = e_2 P(A).$$

Assim p_2 é a segunda linha de P(A). Supomos agora que p_i $(i \geq 2)$ é a i-ésima linha de P(A).

Então

$$p_{i+1} = p_i B_i - \sum_{j=1}^{i-1} a_{ij} p_j$$

$$= [e_i B_i - (a_{i1}, a_{i2}, \ldots, a_{ii-1}, 0 \ldots 0)] P(A) ,$$

uma vez que P(A) e B_i comutam

$$= [(a_i - a_{ii} e_i) - (a_i - a_{ii} e_i - e_{i+1})] P(A) ,$$

notando-se que a_i é a i-ésima linha da matriz de Hessenberg A.

$$= e_{i+1} P(A).$$

Assim p_{i+1} é a (i+1)-ésima linha de P(A), i=1,2,...,n-1.

Observamos que uma vez conhecida a primeira linha da matriz polinômio P(A), as demais linhas são determinadas por relações recursivas (4.6). Isto nos conduz imediatamente ao seguinte resultado.

COROLÁRIO 4.1. *Uma matriz polinômio em uma matriz de Hessenberg inferior normalizada é unicamente determinada pela sua primeira linha. Mais especificamente se* A *é uma matriz de Hessenberg inferior normalizada e* g(x) *e* h(x) *são dois polinômios tais que as matrizes polinômios* g(A) *e* h(A) *tem a mesma primeira linha não-nula,*

então temos necessariamente

$$g(A) = h(A).$$

OBSERVAÇÃO 2: O resultado do Corolário 4.1 não é válido se A é uma matriz de Hessenberg qualquer. Para ver isso, tome

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

$$g(x) = x^2 + 4x + 4$$

$$h(x) = x^2 + 3x + 5$$

Então $g(A) = \begin{bmatrix} 90 \\ 69 \end{bmatrix}$ $h(A) = \begin{bmatrix} 90 \\ 59 \end{bmatrix}$

$g(A)$ e $h(A)$ têm a mesma primeira linha, mas $g(A) \neq h(A)$.

OBSERVAÇÃO 3: Note que na observação 2 a matriz A é uma matriz de Hessenberg superior normalizada. Neste caso, tanto o Teorema 4.1 como o Colorário 4.1 sofrem modificações relevantes. Por exemplo:

COROLÁRIO 4.1'. *Uma matriz polinômio em uma matriz de Hessenberg superior normalizada é unicamente determinada pela sua última linha.*

OBSERVAÇÃO 4. Note que no Corolário 4.1 pode bem acontecer que $g(x) \neq h(x)$, embora $g(A) = h(A)$. No entanto, isto não é possível, se exigimos que o grau $\partial g \leq n-1$. Isto é, se $g(x)$ e $h(x)$ são polinômios de grau menor que ou igual a n-1 tais que $g(A)$ e $h(A)$ tem as mesmas primeiras linhas, sendo que A é uma matriz de Hessenberg inferior normalizada, então $g(x) = h(x)$ e logo $g(A) = h(A)$. Isto segue da unicidade que consta no enunciado da Proposição 4.1.

DEFINIÇÃO 3: Seja A uma matriz de Hessenberg inferior normalizada. Dado um vetor x_n, dizemos que *uma matriz polinômio* P(A) em A é *associada ao vetor* x_n se a primeira linha de P(A) é vetor x_n.

4.6. UMA DECOMPOSIÇÃO CANÔNICA DE SIMETRIZADORAS DE A:

TEOREMA 4.2. Sejam T_o um simetrizadora de A associada ao vetor (1,0,...,0) e X. Então existe uma matriz polinômio P(A) tal que

$$X = T_o P(A).$$

Mais ainda esta decomposição de X é única no segundo sentido. Se $X = T_o Q(A)$ sendo Q(A) uma matriz polinômio, então P(A) = Q(A).

DEMONSTRAÇÃO: Primeiro queremos provar que T é um simetrizador de A. Então TA, TA^2,...,TA^i,... (i=1,2,...) são todos simetrizadores de A.

Pois

$$(TA) \cdot A = (A^t T)A$$

Seja TA^i uma simetrizadora de A, queremos mostrar que TA^{i+1} é é também uma simetrizadora de A.

$$TA^{i+1} \cdot A = A^t\ TA^i\ \ A = A^t\ TA^{i+1}$$

Notamos que 1) se x,y são duas simetrizadoras de A, então (x+y) é também simetrizador de A . Pois,

$$(x+y))A = xA + yA = A^tX + A^ty = A^t(x+y).$$

2) Para cada $a \in k$, aT é uma simetrizadora de A. Destas observações segue que para toda matriz polinômio P(A), TP(A) é também uma simetrizadora de A.

Sejam T_o uma simetrizadora de A associada com um vetor (1,0,...,0) e indique por x_n a última linha de X.

Seja P(A) uma matriz polinômio de A com primeira linha x_n.

Note que pela observação 1,

$$T_o = \begin{bmatrix} * & \cdots & 1 \\ * & \cdots & 1\ 0 \\ \vdots & & 0 \\ \vdots & \iddots & \vdots \\ 1 & & \end{bmatrix}$$

Vê-se que a última linha de $T_oP(A)$ é também x_n. Assim X e $T_oP(A)$ são duas simetrizadoras de A com mesma última linha. Logo,

$$X = T_oP(A) \tag{4.7}$$

Para mostrar que esta decomposição de X, P(A) é única: Suponha que $X = T_oQ(A)$ onde $Q(A)$ é uma matriz polinômio. Segue imediatamente que a primeira linha de $Q(A)$ é x_n. Apelando ao Corolário (4.1), tem-se

$$Q(A) = P(A).$$

OBSERVAÇÃO 5. Note que na decomposição de X na forma $X = T_oP(A)$, a primeira linha de $P(A)$ é a última linha de X.

OBSERVAÇÃO 6. Originalmente tentamos conseguido a decomposição de X na forma

$$X = TQ(A)$$

onde T é forma especial

$$\begin{bmatrix} t_{11} & t_{12} \cdots t_{1n-1} & 1 \\ t_{21} & t_{22} \quad\quad 1 & 0 \\ \vdots & & \vdots \\ t_{n-1\,1} & 1 & \vdots \\ 1 & 0 \cdots\cdots\cdots 0 & 0 \end{bmatrix}$$

e $Q(A)$ uma matriz polinomial em A. Uma pergunta natural é se

$$X = TQ(A) = T_o P(A)$$

sendo a decomposição canônica do Teorema 4.2, será que $T = T_o$ e $Q(A) = P(A)$?

Primeiro observamos que a primeira linha de $Q(A)$ e $P(A)$ são iguais. Logo pelo Corolário 4.1

$$Q(A) = P(A)$$

Não é verdade que $T = T_o$ sempre. Por exemplo se $X = 0$ pode se tomar qualquer T da forma indicada e $T \neq T_o$. Como exemplo de um $X \neq 0$ damos o seguinte caso interessante:

Tome

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

$$T = \begin{bmatrix} t_1 & -t_1 & 1 \\ -1 & 1 & \\ 1 & 0 & 0 \end{bmatrix}$$ onde t_1 é arbitrário e

$P(x) = 1 + x^2$.

Pelo Teorema 4.2 temos

$$X = T_o P(A) = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

é uma simetrizadora de A.

Ora,

$$TP(A) = \begin{bmatrix} t_1 & -t_1 & 1 \\ -1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

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

Evidentemente

$$TP(A) = T_o P(A)$$

observe que P(A) é uma matriz singular. Em visto disto o seguinte resultado é interessante pela unicidade de T para algumas matrizes P(A) mesmo singulares.

TEOREMA 4.3. *Sejam*

$$X = TQ(A) = T_o P(A)$$

decomposições de uma simetrizadora X *de* A *nas formas indicadas acima.*

Então Q(A)=P(A) *e se as primeiras* r *linhas de* P(A) *são linearmente independentes* ($r \leq n-1$) *então as últimas linhas* (r+1) *linhas de* T *e* T_o *são iguais.*

DEMONSTRAÇÃO: Seja p_i a i-ésima linha de P.

Sejam $p_1,\ldots,p_r$ linearmente independentes. De $TP(A)=T_oP(A)$ tem-se:

$$\begin{bmatrix} t_{11} & t_{12} \cdots t_{1n-1} & & 1 \\ t_{r1} & t_{r2} \cdots t_{rr-1} & 1 & \\ t_{n-11} & 1 & & \\ 1 & & & \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_r \\ p_{r+1} \\ \vdots \\ p_r \end{bmatrix}$$

$$= \begin{bmatrix} t^o_{11} & t^o_{12} \cdots t^o_{1n-1} & & 1 \\ \vdots & & & \\ t^o_{r1} & t^o_{r2} \cdots 1 & & \\ t^o_{n-1} & 1 & & \\ 1 & & & \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_r \\ p_{r+1} \\ \vdots \\ p_r \end{bmatrix}$$

Consideremos a i-ésima linha do produto dos dois lados.

Observe que

$$t_{ij} = 1 \quad \text{se} \quad i+j = n+1$$

e

$$t_{ij} = 0 \quad \text{se} \quad i+j > n+1 \ .$$

Comparando-se os dois lados teríamos

$$t_{i1}p_1 + t_{i2}p_2 + \cdots + t_{in-i}p_{n-i} + p_{n+1-i}$$

$$= t^{o}_{i1}p_1 + t^{o}_{i2}p_2 + \cdots + p_{n+1-i}$$

Observe que p_{n+1-i} é cancelado e portanto se

$$i \leq r \quad \text{então} \quad p_1, p_2 \cdots p_{n+1-i}$$

são linearmente independentes. Logo se $i \geq n-r$ então as i-ésimas linhas de T e T_o so iguais. Portanto

(n-(r+1) ésima, (n-r) ésima,..., n-ésima linhas

são iguais para T e T_o .

Isto é, as últimas (r+1) linha de T e T_o são iguais.

OBSERVAÇÃO 7. Vê-se no Teorema 4.3, $T = T_o$ se P(A) é inversível. Mas P(A) pode ser singular.

Por exemplo

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

Neste caso $TA = T_oA \implies T = T_o$.

Note por exemplo que, se $X = 0$, então $r = 0$ e $r+1 = 1$, neste caso como se vê, logo T e T_o somente podem ter a última linha igual.

PERGUNTA 1. É possível afirmar que dada a matriz polinômio $P(A)$, $TP(A) = T_oP(A)$ implica em $T = T_o$ para todo T se, e somente se o posto de $P(A) > n-1$?

EXEMPLO 1. Seja C a matriz companheira do polinômio

$$x^n - c_nx^{n-1} - \ldots - c_2x - c_1 :$$

$$C = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & & \\ c_1 & c_2 & c_3 & \cdots & c_n \end{bmatrix} \qquad (4.8)$$

Seja X a simetrizadora de C associada com o vetor x_n (não nulo), então temos

$$x_{i-1} = x_iC - c_ix_n \quad , \qquad i = n, n-1, \ldots, 3, 2.$$

essas equações podem ser escritas com

$$x_{n-1} = x_n C - c_n x_n$$

$$x_{n-2} = x_{n-1} C - c_{n-1} x_n$$

$$= (x_n C - c_n x_n) C - c_{n-1} x_n$$

$$= x_n C^2 - c_n x_n C - c_{n-1} x_n$$

$$x_{n-3} = x_{n-2} C - c_n x_n$$

$$= x_n C^3 - c_n x_n C^2 - c_{n-1} x_n C - c_n x$$

$$\vdots$$

$$x_2 = x_3 C - c_3 x_n$$

$$= x_n C^{n-2} - c_n x_n C^{n-3} - \ldots - c_3 x_n$$

Analogamente

$$x_1 = x_n C^{n-1} - c_n x_n C^{n-2} - \ldots - c_3 x_n C - c_2 x_n$$

Portanto,

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} -c_2 & -c_3 & \cdots & -c_n & 1 \\ -c_3 & & & 1 & 0 \\ \vdots & & & & \\ -c_n & 1 & & 0 & \\ 1 & & & 1 & \end{bmatrix} \begin{bmatrix} x_n \\ x_n C \\ x_n C^2 \\ \vdots \\ x_n C^{n-2} \\ x_n C^{n-1} \end{bmatrix}$$

Pelo Teorema (4.2) e pelas relações recursivas lá encontradas, segue que

$$\begin{bmatrix} x_n \\ x_n C \\ x_n C^2 \\ \vdots \\ x_n C^{n-2} \\ x_n C^{n-1} \end{bmatrix} = Q(C)$$

para algum polinômio $Q(x) \in k[x]$

Então, temos

$$X = T_o Q(C)$$

onde T_o é simetrizadora de C com a última linha $(1,0,0,0)$.

O seguinte resultado é útil do ponto de vista teórico, pois ele analisa o relacionamento de simetrizadoras de A com a da matriz companheira C de A.

PROPOSIÇÃO 4.2. Sejam C a matriz companheira de uma matriz A de Hessenberg inferior normalizada e Y' e Y simetrizadoras de C de A associada ao mesmo vetor y_n. Então existe uma matriz triangular inferior com diagonal (1,1,...,1) tal que

$$Y = L^t Y' L$$

DEMONSTRAÇÃO. Para A, uma matriz de Hessenberg inferior normalizada, existe uma matriz triangular L com diagonal (1,1,...,1) tal que

$$A = L^{-1} C L$$

onde C é a matriz companheira da forma (4.8) do polinômio característico de A. Como Y é uma simetrizadora de A, temos

$$YL^{-1}CL = L^t C^t (L^t)^{-1} Y$$

onde B^t é a transposta de B.

ou

$$(L^t)^{-1} Y L^{-1} C = C^t (L^t)^{-1} Y L^{-1} .$$

Isso mostra que $(L^t)^{-1} Y L^{-1}$ é uma matriz simetrizadora de C. Como $(L^t)^{-1} Y L^{-1}$ e Y' tem mesmas últimas linhas decorre que

$$Y' = (L^t)^{-1} Y L^{-1} \quad \text{ou seja}$$

$$Y = L^t Y' L \tag{4.10}$$

EXEMPLO 2. Seja

$$A = \begin{bmatrix} a_{11} & 1 & 0 \\ a_{21} & a_{22} & 1 \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Seja X última linha de simetrizadora de A é (1,0,0) , então

$$X = \begin{bmatrix} (a_{11}-a_{33})(a_{11}-a_{22}) + a_{21}-a_{32} & a_{11}-a_{33} & 1 \\ a_{11} - a_{33} & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Seja X_1 uma simetrizadora de A com última linha (0,1,0).

Pelo Teorema 4.2.,

$$X_1 = X_o \cdot P(A) \tag{4.11}$$

onde $P(A) = (b_o I + b_1 A + b_2 A^2)$ é primeira linha de $P(A)$ é última linha de X ,

então $(0,1,0) = b_o(1,0,0) + b_1(a_{11}\ 1,0) + b_2(a_{11}^2+a_{21}\quad a_{11}+a_{22}\ 1)$

Comparando os dois lados

$$b_o + a_{11}\, b_1 + (a_{11}^2 + a_{21})b_2 = 0$$

$$1 = b_1 + b_2(a_{11} + a_{22})$$

$$0 = b_2$$

então $b_1 = 1$, $b_0 = - a_{11}$.

De (4.11) temos

$$X_1 = X_0 \cdot [-a_{11}I + A]$$

$$= \begin{bmatrix} (a_{11}-a_{33})(a_{11}-a_{22}) & a_{11}-a_{33} & 1 \\ +a_{21}-a_{32} & & \\ a_{11}-a_{33} & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ a_{21} & a_{22}-a_{11} & 1 \\ a_{31} & a_{32} & a_{33}-a_{11} \end{bmatrix}$$

$$= \begin{bmatrix} a_{21}(a_{11}-a_{33})+a_{31} & a_{21} & 0 \\ a_{21} & (a_{22}-a_{33}) & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad (4.12)$$

Obtemos o mesmo resultado, usando as relações recursivas em (4.1).

$$x_3 = (0 \quad 1 \quad 0)$$

$$x_2 = x_3 A - a_{33} x_3$$

$$= (a_{21} \quad a_{22}-a_{33} \quad 1)$$

$$x_1 = x_2 A - a_{22} x_2 - a_{32} x_1$$

$$= [a_{21}(a_{11}-a_{33})+a_{31} \quad a_{21} \quad 0]$$

onde

$$e_1 P(A) = p_1 = y_n \qquad (4.13)$$

4.7. SIMETRIZADORA POSITIVA DEFINIDA E SUA IMPLEMENTAÇÃO EM PARALELO:

Nessa secção mostramos que a inversa da matriz de Henkel das somas de Newton é uma simetrizadora de uma matriz companheira A.

Essa matriz é positiva definida se, e somente se os autovalores de A são reais e distintos. Damos também uma técnica conveniente para construir em paralelo a matriz de Hankel das somas de

Newton.

Seja

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ & & & \\ a_1 & a_2 & \cdots\cdots & a_n \end{bmatrix}$$

uma matriz companheira associada ao polinômio

$$f(x) = x^n - a_n x^{n-1} \ldots -a_2 x - a_1 .$$

Definimos

$$s_k = tr(A^k), \; k = 0,1,2,\ldots$$

A matriz

$$H = \begin{bmatrix} s_o & s_1 & \cdots & s_{n-1} \\ s_1 & s_2 & \cdots & s_n \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ s_{n-1} & s_n & \cdots & s_{2n-2} \end{bmatrix}$$

é chamada da matriz de Hankel das somas de Newton.

TEOREMA 4.4. [9] A inversa da matriz de Hankel das somas de Newton é uma simetrizadora da matriz companheira definida acima, isto é

$$H^{-1} A = A^T H^{-1}$$

ou,

$$AH = H A^T .$$

DEMONSTRAÇÃO. É bem conhecido [18] que as somas de Newton satisfazem as seguintes relações:

$$s_i = a_n s_{i-1} + a_{n-1} s_{i-2} + \ldots + a_2 s_1 + a_1 s_o \quad (i=1,2,\ldots,n)$$

$$s_i = a_n s_{i-1} + a_{n-1} s_{i-2} + \ldots + a_2 s_{i-n+1} + a_1 s_{i-n} \quad (i > n)$$

usando essas relações, é fácil verificar que

$$AH = \begin{bmatrix} s_1 & s_2 & \cdots & s_n \\ s_2 & \cdots & \cdots & s_{n+1} \\ \vdots & & & \\ s_n & \cdots & \cdots & s_{2n-1} \end{bmatrix}$$

é simétrica. Como H também é uma matriz simétrica, H^{-1} é uma simetrizadora de A.

TEOREMA 4.5. A matriz de Hankel das somas de Newton é positiva definida se todos os autovalores de A são reais e distintos.

DEMONSTRAÇÃO: Sejam $\lambda_1, \lambda_2, \ldots, \lambda_n$ os autovalores de A. Então podemos escrever

$$H = \begin{bmatrix} s_o & s_1 & \cdots & s_{n-1} \\ s_1 & s_2 & \cdots & s_n \\ \vdots & & & \\ s_{n-1} & \cdots & \cdots & s_{2n-2} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \lambda_1 & \lambda_2 & \cdots & \lambda_n \\ \lambda_1^2 & & & \lambda_n^2 \\ \lambda_1^{n-1} & \lambda_2^{n-1} & & \lambda_n^{n-1} \end{bmatrix} \begin{bmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{n-1} \\ 1 & \lambda_2 & & \lambda_2^{n-1} \\ & & & \\ 1 & \lambda_n & \cdots & \lambda_n^{n-1} \end{bmatrix}$$

$= V \cdot V^T$ onde V é a matriz de Vandermonde. Como $\lambda_1, \lambda_2, \ldots, \lambda_n$ são reais e distintos, V é não singular, e então H é positiva definida. Como a inversa de uma matriz positiva definida é positiva definida, temos dos teoremas acima:

TEOREMA 4.6. A inversa da matriz de Henkel das somas de Newton é uma simetrizadora positiva definida de uma matriz companheira.

4.7.1. CONSTRUÇÃO DA MATRIZ DE HENKEL EM PARALELO.

Apresentamos agora um método eficiente para computar a matriz

de Hankel das somas de Newton em paralelo. Recentemente, Datta [9] mostrou que a matriz de Hankel, das somas de Newton é igual à matriz de Hankel dos parâmetros de Markov associada a $f(x)$ e $f'(x)$, onde $f(x)$ é o polinômio característico de A e $f'(x)$ é a primeira derivada de $f(x)$.

Como existe uma simples relação recursiva para gerar os coeficientes de uma matriz de Hankel dos parâmetros de Markov, podemos utilizar esse fato conveniente para construirmos a matriz de Hankel das somas de Newton em paralelo; como segue:

Sejam

$$f(x) = x^n - a_n x^{n-1} \ldots - a_2 x - a_1$$

$$f'(x) = nx^{n-1} - (n-1)a_n x^{n-2} - \ldots - a_2 .$$

Então

$$s_o = n$$

$$s_1 + s_o(-a_n) = -(n-1)a_n$$

$$\text{ou,} \quad s_1 = a_n$$

$$s_2 - a_n s_1 - a_{n-1} s_o = -(n-2)a_{n-1}$$

então

$$s_2 = a_n s_1 + 2a_{n-1}$$

$$= a_n^2 + 2a_{n-1}$$

$$= \det \begin{bmatrix} a_n & -2a_{n-1} \\ 1 & a_n \end{bmatrix}$$

$$s_3 = a_n s_2 - a_{n-1} s_1 - a_{n-2} s_o = -(n-3)a_{n-2}$$

$$s_3 = a_n s_2 + a_{n-1} s_1 + 3a_{n-2}$$

$$= a_n(a_n^2 + 2a_{n-1}) + a_n a_{n-1} + 3a_{n-2}$$

$$= a_n^3 + 3a_n\, a_{n-1} + 3a_{n-2}$$

$$= \begin{bmatrix} a_n & -2a_{n-1} & 3a_{n-2} \\ 1 & a_n & -a_{n-1} \\ 0 & 1 & a_n \end{bmatrix}$$

e assim

$$s_k = \begin{bmatrix} a_n & -2a_{n-1} & +3a_{n-2} & \cdots & +(-1)^{k-1} k\, a_{n-(k-1)} \\ 1 & a_n & -a_{n-1} & & +(-1)^{k-2} a_{n-(k-2)} \\ 0 & & & & \vdots \\ \vdots & & & & \\ 0 & & & 1 & a_n \end{bmatrix}$$

onde $k = 1,2,\ldots,n$.

Para $k = n+1, n+2, \ldots, 2n-2,$

temos

$$s_k = a_n s_{k-1} + a_{n-1} s_{k-2} + \cdots + a_1 s_{k-n}$$

que pode ser escrito na forma

$$s_k = \begin{bmatrix} a_n & -2a_{n+1} & & +(-1)^{n-1} n a_1 & 0 & \cdots & 0 \\ 1 & a_n & -a_{n-1} & \cdots & +(-1)^{n-2} a_2 & +(-1)^{n-1} a & 0 \\ \vdots & & & & & & \\ 0 & & & & 1 & & a_n \end{bmatrix}$$

Cada matriz s_k acima é uma matriz de Hessenberg e a computação do determinante de uma matriz de Hessenberg deste tipo precisa de $0(\log^2 n)$ passos usando n^4 processadores (durante estes passos, os menores principais líderes são computados também [20].

Como $s_1, s_2, \ldots, s_{n-1}$ são computados como menores principais líderes de s_n e também $s_{n+1}, s_{n+2}, \ldots, s_{2n-3}$ são computados como menores principais líderes de s_{2n-2}, concluimos que a matriz de Hankel pode ser computado em $0(\log^2 n)$ passos usando n^4 processadores.

PERGUNTA 2. Sejam A uma matriz companheira e H a simetrizadora que é inversa da matriz de Hankel. Então, pelo Teorema 4.2

$$H^{-1} = T_o P(A)$$

Qual o polinômio $P(x)$ se $\partial P \leq n-1$?

CAPÍTULO V

A EQUAÇÃO MATRICIAL LA - BL = R

5.1. INTRODUÇÃO

Uma simetrizadora X de uma matriz $n \times n$ pode ser vista como uma solução da equação matricial $XA - A^tX = 0$. Como apontamos no capítulo anterior, existe uma infinidade de soluções X todas em função da sua última linha. Neste capítulo consideramos uma generalização da equação acima, a saber

$$XA - BX = R$$

onde B é uma matriz $n \times n$ de Hessenberg inferior normalizada, A uma matriz $n \times n$ qualquer e R uma matriz tendo as suas primeiras (n-1) linhas nulas. Admitimos uma liberdade na escolha da última linha r_n de R. Em particular r_n pode ser escolhido de maneira que $r_n = \varepsilon(-1)\phi(A)$, onde $\varepsilon = (1,0,0,\ldots,0)$ e $\varphi(x)$ é o polinômio característico de B. Assim existe uma simetrizadora $X = T_o(-1)^n\phi(A)$ de A tendo r_n como sua última linha e a não-singularidade desta simetrizadora fornece um critério para que as duas matrizes A e B tenham um autovalor em comum.

Nosso método possibilita um algorítmo para a computação do vetor r_n e junto com o algorítmo da construção da simetrizadora, temos um método construtivo para o problema de autovalor em comum.

Como apontamos na Introdução desta tese, este último problema é de grande interesse em várias situações práticas. Os dois

métodos conhecidos até agora empregam o resultado de polinômios característicos das matrizes envolvidas e o produto de Kronecker de A e B respectivamente. Como observamos no Capítulo III, o determinante do produto de Kronecker de A e B é a M-determinante de uma matriz conveniente no anel comutativo k [A] de matrizes. Este relacionamento é de fato o que está atrás da demonstração do Teorema onde fornecemos o referido critério. Dado o nosso ponto de vista o trabalho sobre o anel comutativo k[A] se torna mais interessante computacionalmente tanto em paralelo como sequencialmente.

Enquanto o método envolvendo o produto de Kronecker exige o cálculo de um determinante de ordem n^2, nosso método divide o problema em três passos computacionalmente interessante:

1) o cálculo do vetor r_n;

2) construção da simetrizadora X;

3) testar a não singularidade da matriz simétrica X.

O último ponto é facilmente verificado, pois X, sendo simétrica, admite uma decomposição $X = LDL^T$ onde L é uma matriz triangular inferior com diagonal unitária e D uma matriz diagonal. Observe também que existem métodos eficientes para conseguirmos esta decomposição [15a] .

Assim o nosso algorítmo é interessante sequencialmente, embora os dois métodos tenha a mesma complexidade em paralelo.

Observe que no método envolvendo o resultante, é necessário calcular os polinômios característicos das matrizes envolvidas. No

estudo de nossa equação $XA - BX = R$, uma escolha adequada de A proporciona um algorítmo novo para o cálculo explícito do polinômio característico de uma matriz B de Hessenberg inferior em forma normalizada. Este algoritmo sequencial parace muito interessante, uma vez que ele consegue abaixar o número de passos necessários de $\frac{1}{6} n^3$ para $\frac{1}{6} (n^3 - n)$ (ver [41]). (Esperamos ainda melhorar esta última limitação. O Teorema 5.3 trata deste algorítmo bem simples.

Finalmente cabe apontar que usamos a forma da equação $LA - BL = R$, pois muitas vezes o caso interessante é de L ser triangular inferior.

5.2. A EQUAÇÃO MATRICIAL LA - BL = R

TEOREMA 5.1. *Sejam* B *uma matriz* n × n *de Hessenberg inferior normalizada,* A *uma matriz* n × n *qualquer. Sejam* $\phi(x)$ *o polinômio característico de* B *e* ℓ_1 *um vetor arbitrário qualquer. Então existem matrizes* n × n L *e* R *tais que*

(1) ℓ_1 *é a primeira linha de* L;

(2) *as primeiras* (n-1) *linhas de* R *são nulas;*

(3) *a n-ésima linha* r_n *de* R *é dada por* $(-1)^n r_n = \ell_1 \varphi(A)$

e

(4) LA - BL = R. *Mais ainda,* L *é unicamente determinada por* ℓ_1.

DEMONSTRAÇÃO: De

$$LA - BL = R$$

tem-se

$$LA = BL + R \tag{5.1}$$

ou

$$\begin{bmatrix} \ell_1 \\ \ell_2 \\ \cdot \\ \cdot \\ \cdot \\ \ell_n \end{bmatrix} A = \begin{bmatrix} b_{11} & 1 & 0 & \cdots & 0 \\ b_{21} & b_{22} & 1 & \cdots & 0 \\ \cdot & & & & \\ \cdot & & \cdots & & \\ \cdot & & & & \\ b_{n1} & b_{n2} & \cdots\cdots & & b_{nn} \end{bmatrix} \begin{bmatrix} \ell_1 \\ \ell_2 \\ \cdot \\ \cdot \\ \cdot \\ \ell_n \end{bmatrix} + \begin{bmatrix} \\ 0 \\ \\ \\ r_n \end{bmatrix} \tag{5.2}$$

Comparando-se os dois lados da equação (5.2) têm-se:

$$\left.\begin{array}{ll}
\ell_1(b_{11}I - A) + \ell_2 I = 0 \;\dots\dots\dots\dots\dots\dots & (1)\\
\ell_1 b_{21}I + \ell_2(b_{22}I - A) + \ell_3 I = 0 \;\dots\dots & (2)\\
 & \\
b_{n-1,1}\ell_1 + b_{n-12}\ell_2 + \dots + \ell_{n-1}(b_{n-1n-1}I - A) + \ell_n I = 0 & \\
b_{n1}\ell_1 + b_{n2}\ell_2 + \dots + b_{n,n-1}\ell_{n-1} + \ell_n(b_{nn}I - A) = -r_n \dots & (n)
\end{array}\right\} \quad (5.3)$$

Seja

$$D = \begin{bmatrix}
b_{11}\, I - A & I & 0 & & 0\\
b_{21}\, I & b_{22}I - A & I & \ddots & \\
\vdots & & & (b_{n-1n-1}I-A) & I\\
b_{n1}I & \dots & \dots & \dots & (b_{nn}I-A)
\end{bmatrix} \quad (5.4)$$

Multiplicando-se a primeira, a segunda,..., a n-ésima equações de (5.3) pelos cofatores de $(b_{11}\, I - A)$, $b_{21}I,\dots,b_{n1}I$ em D e somando-se têm-se:

$$\ell_1 \phi(A) = (-1)^n r_n$$

onde $\phi(x)$ é o polinômio característico de B.

Do sistema (5.3) se vê que as linhas $\ell_2, \ldots, \ell_n$ de L são unicamente determinadas pelo ℓ_1 e que r_n satisfaz (3) e (4) é válido, que L é unicamente determinada por ℓ_1 é imediato.

COROLÁRIO 1. Com as mesmas hipóteses do teorema, se tomamos $A = B^t$, então L é uma simetrizadora de B^t, sendo escolhida a primeira linha ℓ_1 de L arbitrariamente.

DEMONSTRAÇÃO: Se $A = B^t$, então $\phi(A) = 0$ logo $R = 0$ e

$$LA - BL = 0 \quad \text{ou}$$

$$LB^t - BL = 0 \quad \text{ou}$$

$$LB^t = BL$$

definindo L como uma simetrizadora de B^t.

COROLÁRIO 2. Com as mesmas hipóteses sobre B e A, a equação matricial

$$XA - BX = 0 \ldots$$

admite uma solução não nula se, e somente se $\phi(A)$ é singular.

DEMONSTRAÇÃO: Como $\phi(A)$ é singular existe um vetor não nulo tal que $\ell_1 \phi(A) = 0$. Se construirmos L partindo de ℓ_1 na maneira

citada na demonstração do Teorema 5.1, então

$LA - BL = 0$, pois $r_n = 0$. Logo

$X = L$ é uma solução de

$XA - BX = 0$.

Reciprocamente se existe uma solução $X = L \neq 0$ da equação citada, então a Demonstração do Teorema 5.1 mostra que $\ell_1 = 0$, pois neste caso as relações recursivas de (5.3) definindo L demonstraram que $L = 0$, é um absurdo.

Logo $\ell_1 \neq 0$ e $\ell_1 \phi(A) = (-1)^n r_n = 0$

Portanto $\phi(A)$ é singular.

Um problema importante na álgebra linear trata de matrizes que comutam: Se A e B são duas matrizes que comutam, é possível descrever alguma relação entre A e B? Por exemplo, se T é uma matriz normal em C^n com um produto interno, então T* comuta com T e T* é uma matriz polinômio em T. Em geral, se AB = BA não decorre que A e B tem uma relação polinomial.

Por exemplo: Tome

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

e

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

Então $AB = BA = 0$. Mas $A \neq f(B)$ e $B \neq g(A)$ qualquer que sejam os polinômios A e B.

Em visto destas observações o seguinte resultado é interessante.

COROLÁRIO 3: Se B é uma matriz de Hessenberg normalizada e L uma matriz que comute com B, então L é uma matriz polinomial em B. Mais ainda, pode-se calcular o polinômio envolvido facilmente.

DEMONSTRAÇÃO: Não há perda de generalidade ao tomar B como matriz de Hessenberg inferior. Se L comuta com B, então

$$LB - BL = 0$$

sendo um caso particular da equação (5.1) com $A = B$ e $R = 0$.

As relações recursivas de (5.3) definindo as linhas de L tomam a seguinte forma:

$$\ell_2 = \ell_1(B - b_{11}I)$$

$$\ell_3 = \ell_2(B - b_{22}I) - b_{21}\ell_1$$

e

$$\ell_{i+1} = \ell_i(B - b_{ii}I) - \sum_{j=1}^{i-1} b_j \ell_j \qquad (5.5)$$

onde $i = 1,2,\ldots,n$.

As relações (5.5) junto com o Teorema 4.1 mostram que L é uma matriz polinômio em B com primeira linha ℓ_1.

Para computar o polinômio basta observar que

se $\ell_1 = a_o\varepsilon + a_1\varepsilon B + \ldots + a_{n-1}\varepsilon B^{n-1}$,

então $P(X) = a_o + a_1X + \ldots + a_{n-1}X^{n-1}$.

COROLÁRIO 4: Com as mesmas hipóteses do Teorema 5.1 sobre A e B a equação matricial

$$XA - BX = 0$$

admite apenas a solução trivial se, e somente se $\phi(A)$ é não singular ou equivalentemente se, e somente se A e B não têm autovalores comum.

DEMONSTRAÇÃO: O resultado é imediato do Corolário 3.

O interesse deste Corolário realmente está na segunda afirmação o que resulta do Teorema 5.2 que segue mais adiante.

OBSERVAÇÃO 1: É interessante observar que se A e B são ambas matrizes de Hessenberg inferiores normalizada e $\ell_1 = (1,0,0\ldots0)$, então a matriz L do Teorema 5.1 é uma matriz triangular inferior com a diagonal (1,1,... 1).

5.3. UM CRITÉRIO PARA O PROBLEMA DE AUTOVALOR COMUM

Partindo de duas matrizes $n \times n$ A e B de Hessenberg inferiores normalizadas, construimos uma simetrizadora S de A cuja não singularidade fornece um critério para que A e B não tenham nenhum autovalor em comum. Como destacamos na introdução deste capítulo, o algorítmo proposto aqui tem suas vantagens tanto na sua forma sequencial como na sua implementação em paralelo. A escolha da última linha de S é crucial na determinação de S conforme diz o Algorítmo de Datta. Esta escolha como a computação explícita desta última linha s_n de S é dirigida pelo Teorema 5.1.

TEOREMA 5.2. *Sejam* A *e* B *duas matrizes de Hessenberg inferiores normalizadas. Seja* L *a única solução da Equação Matricial* (5.1) *com* $\ell_1 = (1,0,0,\ldots,0)$ *e* LA - BL = R, *tendo* R *as suas primeiras* (n-1) *linhas nulas. Sejam* r_n *a última linha de* R *e* S *a simetrizadora de* A *associada no vetor* r_n. *Então* S *é não-singular se, e somente se* A *e* B *não tem nenhum auto valor em comum.*

DEMONSTRAÇÃO: Seja P(A) a matriz polinômio associada à simetrizadora S. Pelo Teorema 5.1 $(-1)^n r_n = \ell_1 \phi(A)$ = a primeira linha de $\phi(A)$, onde $\phi(x)$ é o polinômio característico de B. Pelo Teorema 4.2 e Corolário 4.1, concluimos agora que,

$$P(A) = \phi(A). \quad \text{Pelo Teorema 4.2 ,}$$

$$S = T_o P(A) = T_o (-1)^n \phi(A)$$

e S é não singular se, e somente se, $\phi(A)$ é não singular.

Por outro laso, se $\lambda_1,\lambda_2,\ldots,\lambda_n$ são os autovalores de A e $\mu_1,\mu_2,\ldots,\mu_n$ os de B, os autovalores de $\phi(A)$ são $\phi(\lambda_1),\ldots,\phi(\lambda_n)$. $\phi(A)$ é não-singular se, e somente se nenhum $\phi(\lambda_i)$ é nulo. Como

$$\phi(\mu) = \pm\ (\mu-\mu_1)(\mu-\mu_2)\ \ldots\ (\mu-\mu_n)\ ,$$

temos

$$\phi(\lambda_j) = \pm\ (\lambda_j-\mu_1)(\lambda_j-\mu_2)\ \ldots\ (\lambda_j-\mu_n).$$

Então $\phi(A)$ é não-singular se, e somente se, $\lambda_j \neq \mu_i$ para todo i e j.

5.4. UM EXEMPLO

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2i & -1 & 2i \end{bmatrix} \quad e \quad B = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

duas matrizes de ordem 3. Os autovalores de A são i, -i, 2i e os de B são $0,\ \frac{3+\sqrt{5}}{2}\ ,\ \frac{3+\sqrt{5}}{2}$.

Verificamos agora que A e B não tem autovalor em comum usando nosso algorítmo

$$\ell_1 = (1,0,0)$$

$$\ell_2 = \ell_1 A - b_{11}\ell_1$$

$$= (-1,1,0)$$

$$\ell_3 = \ell_2 A - \sum_{j=1}^{2} b_{2j}\ell_j$$

$$= (0,0,1)$$

então

$$LA - BL$$

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

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

Seja $s_3 = -(\ 2i \quad -2 \quad 2i-1)$

$$= (-2i \quad +2 \quad 1-2i)$$

$$s_2 = (-2i \quad 2 \quad 1-2i)\ A - \ 2i(-2i \quad 2 \quad 1-2i)$$

$$= (2i \quad -i-4i \quad 2)$$

$$s_1 = (2i \quad 2i \quad -2i)$$

$$S = \begin{bmatrix} 2i & 2i & -2i \\ 2i & -1-4i & 2 \\ -2i & 2 & 1-2i \end{bmatrix}$$

e det $S \neq 0$.

Então A e B não tem nenhum autovalor em comum.

5.5. O ALGORÍTMO PROPOSTO PARA O PROBLEMA DE AUTOVALOR COMUM

PASSO 1. Seja $\ell_1 = (1,0,0 \ldots)$

PASSO 2. Compute

$$\ell_{k+1} = \ell_k A - \sum_{j=1}^{k} b_{kj} \ell_j$$

$$k = 1,2,\ldots,n-1 .$$

PASSO 3. Compute

(LA - BL) para achar última linha r_n de R.

PASSO 4. Construa uma matriz S com linhas $s_1, s_2 \ldots s_n$ tal que

$$_n = r_n$$

e

$$s_i = s_i \quad a_{i+1i+1} s_{i+1} \cdots -a_{ni+1} s_n$$

onde i = n-1, n-2. ., 3,2,1.

Vamos implementar nossa técnica de implementação de algorítmos em paralelos do Capítulo III.

Portanto

$$\ell_1 = (1,0\ 0\ 0\ )$$

$$\ell_2 = -\ell_1(b_{11}I - A)$$

$$\ell_3 = (-1)^2\ell_1 \text{ M det} \begin{bmatrix} (b_{11}I-A) & I \\ b_{21}I & (b_{22}I-A) \end{bmatrix}$$

.

.

.

analogamente

$$\ell_n = (-1)^{n-1}\ell_1 \text{ M det} \begin{bmatrix} (b_{11}I-A) & I & 0 & \cdots & \\ \vdots & & \ddots & & \\ & & & & I \\ b_{n-11}I & & & & (b_{n-1n-1}I-A) \end{bmatrix}$$

Podemos computar todas as linhas de L em $0(\log^2(n-1)$ passos com $[0(\frac{n^7}{2})]$ processadores [Cap. III].

Uma vez L é construida, compute a última linha de R, via LA - BL = R em $[0(\log n)+2]$ passos com $(2n^2)$ processadores [Cap.II].

Também já sabemos do Capítulo IV que a construção das linhas de S pode ser feita em $0(\log^2 n)$ passos com $[0(\frac{n^7}{2})]$ processadores.

Concluimos que a construção da matriz S em paralelo pode ser feita

em $O(\log^2 n)$ passos usando somente $[O(\frac{n^7}{2})]$ processadores.

5.6. DOIS MÉTODOS PARA COMPUTAR O POLINÔMIO CARACTERÍSTICO DE UMA MATRIZ DE HESSENBERG NORMALIZADA.

Neste parágrafo descrevemos dois novos algorítmos sequenciais para a computação explícita do polinômio característico de uma matriz de Hessenberg inferior normalizada. Ambos os algorítmos dependem do fato de que a matriz é de Hessenberg normalizada. As observações feitas no Capítulo IV mostram que isto não acarreta a perda da generalidade. A justificativa para o primeiro algorítmo se baseia na solução da Equação Matricial (5.1) dada no Teorema 5.1.

I - TEOREMA 5.3. *Seja* B *uma matriz de Hessenberg inferior e normalizada e* A *a matriz nilpotente dada por*

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

e seja L *a única solução da Equação Matricial* LA - BL = R *do Teorema* 5.1 *com* $\ell_1 = (1,0,\ldots,0)$.

Seja $\phi(x) = (-1)^n x^n + b_{n-1}x^{n-1} + \ldots + b_o$ (5.6) *o polinômio característico de* B. *Se* r_n *é a última linha de* R *então*

$$(-1)^n r_n = (b_o, b_1, \ldots, b_{n-1}).$$

DEMONSTRAÇÃO: Do Teorema 5.1 sabemos que

$$(-1)^n r_n = \ell_1 \phi(A) \ldots \tag{5.7}$$

onde ℓ_1 é a primeira linha de L, e $\phi(x)$ o polinômio característico de B.

$$\phi(x) = (-1)^n x^n + p(x)$$

onde o grau de $p(x) < n-1$.

Observe que $A^n = 0$; logo

$\phi(A) = p(A)$. Ora $(-1)^n r_n = \ell_1 p(A) = \ell_1 p(A) = b_o \varepsilon + b_1 \varepsilon A + \ldots + b_{n-1} \varepsilon A^{n-1}$

Como $\{\varepsilon, \varepsilon A, \varepsilon A, \ldots, \varepsilon A^{n-1}\}$ é de fato a base canônica de k^n, temos que

$$(-1)^n r_n = (b_o, b_1, \ldots, b_{n-1})$$

como queríamos.

II - ALGORÍTMO (SEQUENCIAL) PROPOSTO.

PASSO 1. Seja $\ell_1 = (1,0 \ldots 0)$

PASSO 2. Compute

$$\ell_{k+1} = \ell_k A - \sum_{j=1}^{k} b_{kj} \ell_j$$

$$k = 1,=,\ldots,n-1$$

PASSO 3. Compute (LA - BL) para achar a última linha do R.

Seja $r_n = (c_o, c_1, \ldots, c_n)$

então o polinômio característico de B é

$$\phi(x) = (-1)^n x^n + c_{n-1} x^{n-1} + \ldots + c_o .$$

III - NÚMERO DE PASSOS NA IMPLEMENTAÇÃO SEQUENCIAL

Mostramos agora que o nosso algorítmo acarreta uma ligeira melhora no número de passos. O melhor resultado até agora diz que o polinômio característico pode ser computado em $\frac{1}{6} n^3$ passos, sendo n a ordem da matriz envolvida.

Mostramos que se A é uma matriz de Hessenberg inferior normalizada, então o algorítmo proposto possibilita o mesmo resultado em um número de passos menor ou igual a $\frac{1}{6}(n^3-n)$.

Note que na computação da (k+1)-ésima linha de L, para efeito do cálculo de número de passos, $\ell_k A$ não entra, pois as entradas de A são 0 ou 1. Observe que $L = \ell_{ij}$ é uma matriz triangular inferior com diagonal (1,1,1,...,1); portanto a multiplicação por ℓ_{ii} não entra no nosso cálculo. Assim os números de passos necessários para a computação de $\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ldots, \ell_n$ são respectivamente 0,0,1, 1+2, 1+2+3,..., e 1+2+...+ (n-2). Assim o número de passos para a computação de L é $\sum_{n=3}^{n} \frac{(n-1)(n-2)}{2}$ .Na base das observações feitas acima, a computação da última linha de LA - BL pode ser feita em $(n-1)+(n-2) + \ldots + 1 = \frac{n(n-1)}{2}$. Portanto, o

número de passos para a computação do polinômio característico é

$$\left(\sum_{n=3}^{n} \frac{(n-1)(n-2)}{2}\right) + \left[\frac{n(n-1)}{2}\right]$$

$$= \left(\sum_{n=1}^{n-2} \frac{n(n+1)}{2}\right) + \frac{n(n-1)}{2}$$

$$= \frac{1}{2}\left(\sum_{1}^{n-2} n^2\right) + \frac{1}{2}\sum_{1}^{n-2} n + \frac{n(n-1)}{2}$$

$$= \frac{1}{2}\,\frac{1}{6}(n-2)(n-1)(2n-3) + \frac{1}{2}\,\frac{(n-2)(n-1)}{2} + \frac{n(n-1)}{2}$$

$$= \frac{1}{12}(n-1)\{(n-1)(2n-3)+3(n-2)+6n\}$$

$$= \frac{1}{12}(n-1)(2n^2+2n)$$

$$= \frac{1}{6}(n^3-n)$$

IV - O nosso próximo algorítmo (sequencial) segue o mesmo espírito do algorítmo anterior. Ele é uma ligeira modificação de um algorítmo de Datta [12]. Na forma discreta aqui apresentada ele aparece mais interessante que na forma existente na literatura. Damos aqui um exemplo de algorítmo *ineficiente*.

ALGORÍTMO II. *Sejam* B *uma matriz de Hessenberg normalizada e* $\varphi(x) = (-1)^n x^n + b_{n-1}x^{n-1} + \ldots + b_1 x + b_o$ *seu polinômio característico.*

PASSO 1. Computar recursivamente

$$\ell_o = \varepsilon = (1,0,\ldots,0)$$

$$\ell_1 = \varepsilon B = (b_{11}, 1, \ldots, 0) = \ell_o B$$

$$\ell_2 = \ell_1 B = \ell_o B^2 = \varepsilon B^2$$

$$\ell_3 = \ell_2 B = \ell_o B^3 = \varepsilon B^3$$

$$\vdots$$

$$\ell_{n-1} = \varepsilon B^{n-1} = \ell_{n-2} B$$

PASSO 2. Computar

$$s = \varepsilon B^n = (c_o, c_1, \cdots, c_{n-1})$$

$$= b_{n-1} B$$

PASSO 3. Computar $- b_{n-1} = c_{n-1}$

$$- b_{n-2} = c_{n-2} + b_{n-1} b_{n-1,n-1}$$

$$- b_{n-3} = c_{n-3} + b_{n-2} b_{n-2,n-2} + b_{n-1} b_{n-1,n-2}$$

$$\vdots$$

$$- b_k = c_k + \sum_{j=k+1}^{n-1} b_j b_{j,k+1} \qquad k = n-1, n-2, \ldots, 0$$

Então $b_o, b_1, \ldots, b_{n-1}$ são os coeficientes desejados do polinômio característico de B.

DEMONSTRAÇÃO: Temos que justificar o algorítmo. Ora $\phi(B) = 0$. Conduza $s = \varepsilon B^n = -b_o\varepsilon - b_1\varepsilon B - b_2\varepsilon B^2 - \ldots - b_{n-1}\varepsilon B^{n-1}$

$= (c_o, c_1, \ldots, c_{n-1})$. Notando-se que

$\varepsilon A^j = (\ell_{11}, \ell_{12}, \ldots, 1 = \ell_{1j+1}, 0, 0, \ldots 0)$ seguem as equações do Passo 3 por comparação.

Vamos agora mostrar que o algorítmo requer um número bem elevado de passos na sua implementação sequencial. Como no algorítmo que segue o Teorema 5.3, o cálculo de $\ell_1, \ell_2, \ldots, \ell_{n-1}$ e s exigem $\frac{1}{6}(n^3-n)$ passos. O passo 3 requer $\frac{(n-1)n}{2}$ passos. Assim obtemos ao total o seguinte número de passos:

$$\frac{1}{6}(n^3-n) + \frac{n(n-1)}{2}$$

$$= \frac{(n-1)(n^2+4n)}{6} = \frac{n^3+3n^2-4n}{6} \geq \frac{n^3+2n^2}{6}$$

se $n \geq 4$ e $\geq \frac{n^3+n^2}{6}$ se $n \geq 2$. Portanto, este último algorítmo é ineficiente e o nosso primeiro algorítmo, deduzido do Teorema 5.3 é preferível.

V. Finalmente é evidente como implementar o primeiro algorítmo em paralelo. O número de passos neste caso é $0(\log^2 n)$.

VI . UM EXEMPLO

Retomamos o exemplo do Parágrafo 5.4.

Sejam

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad e \quad B = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$\ell_1 = (1,0,0)$

$\ell_2 = (1,0,0)A - b_{11}\ell_1$

$= (0,1,0) - (1,0,0)$

$= (-1, 1, 0)$

$\ell_3 = (-1, 1, 0)A - (b_{21}\ell_1 + b_{22}\ell_2)$

$= (0, -1, 1) - [(1, 0, 0) + (-1, 1, 0)]$

$= (0, -2, 1)$

Então

$$LA - BL =$$

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

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

$\phi(x)$ = polinômio característico de B.

$= x^3 + 3x^2 - 1x$.

Logo o polinômio característico de B é $x^3 + 3x^2 - x$.

Repare que os autovalores de B são $0, \frac{3+\sqrt{5}}{2}, \frac{3-\sqrt{5}}{2}$.

CAPÍTULO VI

SIMETRIZADORAS NO ESTUDO DE LOCALIZAÇÃO DE RAÍZES DE UM POLINÔMIO E SEPARAÇÃO DE AUTOVALORES DE UMA MATRIZ

6.1. INTRODUÇÃO: O sistema de *equações diferenciais*

$$\dot{x}(t) = Ax(t) \tag{I}$$

Surge em várias aplicações práticas. O *sistema* é dito estável, se, cada solução $x(t) \longrightarrow 0$ com $t \longrightarrow \infty$. Como uma solução arbitrária do sistema pode ser escrita na forma:

$$x(t) = e^{At}x(0)$$

é fácil ver que o sistema será estável se, e somente se, as partes reais de todos os autovalores de A são negativas. Por outro lado, em matemática aplicada o sistema (I) é frequentemente aproximado pelo sistema de equações diferenças da forma

$$x_{k+1} = Ax_k \tag{II}$$

e este sistema é estável se, e somente se, todos os autovalores de A estão dentro do círculo unitário.

De fato, esses problemas de estabilidade são casos especiais de dois problemas antigos de matemática: o problema do círculo unitário e o problema da inércia de uma matriz. A inércia de uma matriz A denotado por In(A) é uma tripla $(\pi(A),\nu(A),\delta(A))$ onde $\pi(A)$, $\nu(A)$ e $\delta(A)$ são respectivamente os números de autovalores de A com partes reais positivas, negativas e nulas. Então o problema de estabilidade do sistema (I) é um caso especial do problema de inércia. Nos termos de inércia, podemos dizer que o sistema (I) é estável se, e somente se

$$In(A) = (0,\ n,\ 0).$$

A matriz A com In(A) = (0,n,0) é dita *matriz estável*.

Analogamente, o problema da estabilidade do sistema (II) é um caso especial do problema de círculo unitário. No caso particular em que a matriz A é uma matriz companheira de um polinômio f(x), os problemas de inéricia e círculo unitário são conhecidas como os problemas de localização de raízes de f(x) e foram tratados separadamente na literatura de matemática e teoria de controle. Existem muitos métodos para resolução dos problemas de localização de raízes de f(x). Entre esses, o método mais citado em literatura é um método antigo de Fujiwara [17] que deu uma solução unificada de ambos os problemas usando a forma bilinear de Bezout. O método de Fujiwara é considerado como um método original e muitos outros métodos desenvolvidos posteriormente são considerados como variações do método de Fujiwara. O método de Fujiwara precisa dos coeficientes exatos de f(x). O método de Fujiwara consiste em expressar o número de raízes com partes reais negativas de um

polinômio f(x) em termos de inércia de uma matriz hermitiana F_1.

Neste capítulo daremos um tratamento unificado dos resultados de Fujiwara e Carlson-Datta. A nossa primeira observação consiste em ligar a simetrizadora, S do Parágrafo 6.3 à forma quadrática de Bezout. De fato, S é nada mais do que uma matriz representando a forma quadrática de Bezout numa base conveniente; obtida pela mudança da base a partir da matriz de Bezout.

Dada esta observação encaramos a matriz hermitiana, F_1 de Fujiwara como a matriz de uma forma hermitiana F(f), a qual por sua vez sugere a possibilidade de mudança da base. Esta observação nossa, torna-se muito proveitosa, pois conseguimos mostrar que a matriz H de Carlson-Datta descrevendo a inércia de uma matriz de Hessenberg A é também uma matriz representando a forma hermitiana F(f) de Fujiwara.

Neste Capítulo também implementamos em paralelo o método de Fujiwara e de Carlson-Datta.

Finalmente sob este ponto de vista conseguimos achar um novo algorítmo baseado no método de Fujiwara para o problema do círculo unitário. Este algorítmo é o conteúdo de 6.5.1 e 6.5.2.

6.2. ALGUNS TEOREMAS DE INÉRCIA E DO CÍRCULO UNITÁRIO

Nesta secção citamos dois teoremas, um de inércia e outro um teorema de círculo unitário, que usaremos mais tarde neste capítulo.

TEOREMA 6.1. (Carlson e Schneider [6]).

Seja A *uma matriz complexa com* $\delta(A) = 0$.

Sejam H *uma matriz hermitiana não singular e* N *uma matriz positiva semidefinida tal que* $HA + A^*H = N$.

Então $In(A) = In(A^*) = In(H)$.

TEOREMA 6.2. (Datta [11]).

Seja A *uma matriz complexa com nenhum autovalor de módulo um.*

Sejam H *uma matriz hermitiana não singular e* N *uma matriz positiva semidefinida.*

$$(A^*HA - H) = N.$$

Então o número de autovalores de dentro (fora) do círculo unitário é $\nu(A)$ $(\pi(H))$.

6.3. A FORMA BILINEAR DE BEZOUT

Dados dois polinômios f(x) e g(x) a forma Bilinear de Bezout associada a

$$f(x) = a_n x^n - a_{n-1} x^{n-1} - \ldots - a_1 x - a_o \qquad a_n \neq 0 \tag{6.1}$$

$$g(x) = b_n x^n - b_{n-1} x^{n-1} - \ldots - b_1 x - b_o \qquad b_n \gtreqless 0 \tag{6.2}$$

é definida da seguinte maneira

$$B(f.g. : x,y) = \frac{f(x)g(y) - f(y)g(x)}{x - y}$$

$$= \sum_{i,k=0}^{n-1} b_{ik} x^i y^k$$

A matriz $B_{fg} = (b_{ik})$ é simétrica e é chamada de matriz de Bezout. Esta matriz simétrica define a forma Bilinear de Bezout.

TEOREMA 6.3. *O determinante da matriz de Bezout é igual ao resultante de* f(x) *e* g(x) *multiplicado por* $(-1)^n$. *Assim* $|B_{fg}| = (-1)^n$ *(o resultante de* f(x) *e* g(x)). *Daí* B_{fg} *é não singular se, e somente*

se f(x) *e* g(x) *não tem nenhum zero em comum.*

É evidente portanto, que a forma bilinear de Bezout pode ser usada para problema de autovalor comum de duas matrizes A e B, desde que conheçamos seus polinômios característicos. De fato, o algorítmo proposto no capítulo V (Teorema 5.2) é nada mais de que uma reformulação da "matriz de Bezout" numa outra base mais conveniente do ponto de vista computacional. Passamos a esclarecer isto mais precisamente.

A primeira observação é um fato devido a Datta e Barnett [11].

PROPOSIÇÃO 1. (Barnett-Datta): Sejam f(x) e g(x) polinômios como (6.1) e (6.2) respectivamente e C matriz companheira de $\frac{1}{a_n}$ f(x). Então a matriz de Bezout é

$$B_{fg} = X \quad \text{onde}$$

X é simetrizadora de C associada ao vetor $a_n(\varepsilon g(C))$.

$$(b_n a_o - a_n b_o,\ b_n a_1 - a_n b_1, \ldots,\ b_n a_{n-1} - a_n b_{n-1}).$$

Então a forma bilinear de Bezout é *representada por* X *(numa base conveniente)*.

A matriz X é chamada por Fujiwara de matriz de Bezout. Mas reparamos que esta matriz depende de base, e portanto a forma bilinear de Bezout pode ser representada por várias matrizes.

Vamos supor que f e g são polinômios característicos de duas matrizes A e B respectivamente (não necessariamente matrizes companheiras).

Nossa meta é computar a forma bilinear de Bezout de f e g partir das matrizes A e B sem computarmos f e g. Portanto examinemos o efeito da mudança de base sobre simetrizadoras. Encaramos uma simetrizadora como uma forma bilinear.

PROPOSIÇÃO 2. Se A e C são semelhantes, isto é, se $A = DCD^{-1}$ e se X é uma simetrizadora de A então D^tXD é uma simetrizadora de C. Portanto, as duas simetrizadoras definam a mesma forma bilinear.

DEMONSTRAÇÃO: Sejam $A = DCD^{-1}$

então

$XA = A^tX$, ou seja $XDCD^{-1} = (D^t)^{-1}C^tD^tX$. Daí, $D^tXDC = C^tD^tXD$. Portanto D^tXD é simetrizadora de C. É evidente que X e D^tXD definem a mesma forma bilinear.

O próximo resultado descreve a conexão do algorítmo do Parágrafo 5.2 com a forma bilinear de Bezout.

TEOREMA 6.4: *Sejam* A e B *duas matrizes de Hessenberg em forma normalizada e* S *a matriz simetrizadora de* A *associada ao vetor* r_n *como no teorema* 5.2 . *Então* S *também representa a forma bilinear de Bezout (na base de* A*) dos polinômios característicos de* A e B.

DEMONSTRAÇÃO: Por construção S é uma simetrizadora de A. Se C é a matriz companheira de A então

$$C = D^{-1}AD \quad \text{e} \quad X = D^{t}SD$$

é uma simetrizadora de C.

Ora,

$$S = T_o(-1)^n\phi(A) \quad \text{de onde}$$

$$D^tSD = D^tT_o(-1)^n\phi(A)D$$

Logo,

$$X = D^tSD = D^tT_oDD^{-1}(-1)^n\phi(A)D$$

$$= T'_o(-1)^n\phi(C)$$

Sejam

$$f(x) = (-1)^nx^n - a_{n-1}x^{n-1} - \ldots - a_o$$

e

$$\phi(x) = (-1)^nx^n - b_{n-1}x^{n-1} - \ldots - b_o$$

os polinômios característicos de A e B respectivamente.

Apelamos ao Teorema 5.2 para concluir o seguinte: Sejam C e Y as matrizes companheiras de f(x) e $\phi(x)$ respectivamente. Então IC - YI = R onde I é a matriz identidade $n \times n$ e R é uma matriz tendo as suas primeiras (n-1) linhas nulas e a n-ésima linha r_n = $= (-1)^n(a_o-b_o, a_1-b_1,\ldots,a_{n-1} - b_{n-1})$.

Logo

$$\varepsilon(-1)^n\phi(c) = r_n = (-1)^n(a_o-b_o,\ldots,a_{n-1}-b_{n-1}).$$

Portanto a última linha da simetrizadora X de C é igual a

$$(-1)^n(a_o-b_o,\ a_1-b_1,\ldots,a_{n-1}-b_{n-1})$$

$$= \text{a última linha de } B_{fg}.$$

6.4. A PRIMEIRA FORMA HERMITIANA DE FUJIWARA

O problema de inércia de uma matriz é de fato um problema de localização de raízes do seu polinômio característico. Este último problema é respondido por dois teoremas de Fujiwara, associando matrizes hermitianas a polinômios dados e recolocando o problema em termos de valores característicos destas matrizes hermitianas. Como existem métodos eficientes para a computação de valores característicos de matrizes hermitianas, esta colocação de Fujiwara responde ao problema de localização de raízes de polinômios de modo eficiente.

No caso geral, o problema de inércia de uma matriz qualquer se reduz a inércia de matrizes de Hessenberg (inferiores) em forma normal, como aplicamos no Parágrafo 4.2. No caso então, de uma matriz de Hessenberg A em forma normalizada, o interesse se transfere para um método que não envolve o cálculo explícito do polinômio característico de A. Este algorítmo é proporcionado por um teorema de Carlson-Datta.

Mostramos neste parágrafo que a matriz hermitiana de Carlson Datta é nada mais do que a forma hermitiana de Fujiwara numa outra base. Conseguimos fazê-lo porque olhamos "a matriz de Fujiwara" como uma forma hermitiana, a qual por sua vez sugere a possibilidade da mudança da base.

Dado um polinômio complexo $f(x) = a_n x^n - a_n x^{n-1} - \ldots - a_o$ com $a_n \neq 0$, definimos $g(x)$ por $g(x) = \overline{f}(-x)$.

Sejam $D = \text{diag}(1,-1,1,-1,\ldots,(-1)^{n-1})$ e $B_{f,g} = B_{f,\overline{f}(-x)}$ a "matriz de Bezout" de f e g. Então, a "matriz de Fujiwara" é dada por $F_1 = DB_{f,\overline{f}(-x)}$. Esta matriz é hermitiana, pois se $F_1 = (c_{ij})$ e $F_1^* = \overline{B}_{f,g}D = (d_{ij})$ e $B_{f,g} = (a_{ij})$

$$c_{ij} = (-1)^{i-1} a_{ij} \quad \text{e} \quad d_{ij} = \overline{a}_{ij}(-1)^{j-1}$$

e

$$\overline{c}_{ji} = (-1)^{j-1}\overline{a}_{ji} = (-1)^{j-1}\overline{a}_{ij} = d_{ij}.$$

A matriz F_1 define uma forma hermitiana que indicaremos por $F(f)$.

O Teorema de Fujiwara segue . A prova nossa usa a equação (5.1) e o Teorema de Carlson-Schneider.

TEOREMA 6.5. (Fujiwara [17]). *Suponha que F_1 é não singular.*

Então,

(a) *o número de zeros de f(x) com parte real negativa (positiva) é igual ao número de autovalores positivos (negativos) de F_1.*

(b) *em particular, f(x) é estável se, e somente se, F_1 é positiva definida.*

DEMONSTRAÇÃO: Seja A a matriz companheira de f(x).

$$F_1A + A^*F_1 = DB_{f,g}A + A^*DB_{f,g}$$

$$= DA^tB_{f,g} + A^*DB_{f,g}$$

pois B_{fg} é uma simetrizadora de A

$$= (DA^t + A^*D)B_{f,g}$$

$$= (DA + \bar{A}D)^*B_{f,g} \qquad (6.3)$$

É fácil verificar que

$DA + \bar{A}D = R$ é uma matriz cujas primeiras (n-1) linhas são nulas e a última linha r_n é

$$(-a_o + \bar{a}_o(-1)^n,\ -a_1 - \bar{a}_1(-1)^n, \ldots, -a_{n-1} - (-1)^{n-1}\bar{a}_{n-1}(-1)^n)$$

que é (-1) vezes a última linha de B_{fg}.

Então de (6.3) temos

$$F_1A + A^*F_1 = R^*B_{fg}$$

$$= -r_n^* r_n \qquad (6.4)$$

A matriz no lado direito é claramente uma matriz hermitiana e negativa definida.

A não singularidade de F_1 implica que B_{fg} é não singular e pelo Teorema 6.4 temos que f(x) e g(x) não tem nenhum zero em comum, isto é $\delta(A) = 0$.

Aplicando o ao Teorema de Inércia de Carlson e Schneider a (6.1), obtemos o Teorema (6.1) de Fujiwara.

Vamos agora descrever o método de Carlson-Datta para a inércia de uma matriz de Hessenberg em forma normalizada e em seguida mostramos que a matriz H de Carlson-Datta simplesmente representa a forma hermitiana F(f) de Fujiwara (numa outra base).

6.4.1. O MÉTODO DE CARLSON e DATTA PARA COMPUTAR A INÉRCIA

Seja A uma matriz de Hessenberg inferior normalizada.

PASSO 1: Construir uma matriz triangular inferior L com diagonal $(1, -1, 1, \ldots, (-1)^{n-1})$ tal que as primeiras (n-1) linhas de

$$LA + \bar{A}L = R \qquad (6.10)$$

são nulas.

PASSO 2: Computar a última linha r_n de R.

PASSO 3: Construir a simetrizadora S de A associada com r_n.

PASSO 4: Computar H = L*S

TEOREMA 6.6 (i) *a matriz* H *é hermitiana e* H *é não singular se, e somente se a matriz* F_1 *de Fujiwara é não singular. Se* H *é não singular, então* H *representa a forma hermitiana de Fujiwara e* In(A) = In(H).

(ii) H *é não singular se, e somente se,* A e $-\bar{A}$ *tem pelo menos um autovalor em comum.*

DEMONSTRAÇÃO. Sejam A uma matriz de Hessenberg normalizada inferior e C a matriz companheira de A.

Então existe uma matriz triangular T com diagonal unitária tal que

$$C = T^{-1}AT$$

Pelo Teorema 5.1, existem matrizes L_1 e R_1, onde L_1 é triangular inferior com diagonal $(1, -1, 1, \ldots, (-1)^{n-1})$ e R_1 é uma matriz tendo as primeiras (n-1) linhas nulas tais que $L_1C + \bar{C}L_1 = R_1$.

Ora,

$$L_1(T)^{-1}AT + (\bar{T})^{-1}\bar{A}\,\bar{T}L_1 = R_1, \quad \text{donde}$$

$$(\bar{T})L_1(T)^{-1}\,A + \bar{A}\,\bar{T}L_1(T)^{-1} = \bar{T}\,R_1(T)^{-1} \tag{6.5}$$

Sejam

$$L = \bar{T}\,L_1T^{-1} \quad \text{e}$$

$$R = \bar{T}\,R_1(T)^{-1}\,.$$

Note que R têm a mesma forma de R_1 e que L é uma matriz triangular inferior com diagonal $(1, -1, \ldots, (-1)^{n-1})$. Pelo (6.5) temos

$$LA + \bar{A}L = R \tag{6.6}$$

Pelo Teorema 6.5

$$S = (T^t)^{-1} B_{fg}(T)^{-1}$$

ora

$$H = L^*S = (T^*)^{-1} L_1^* T^t (T^t)^{-1} B_{fg}(T)^{-1}$$

$$= (T^*)^{-1} L_1^* B_{fg} T^{-1} .$$

Pela construção L_1 é a mesma matriz D de (6.3); então

$$H = (T^*)^{-1} D B_{fg} T_1$$

$$= T_1^* DB_{fg} T_1 \quad \text{onde} \quad T_1 = T^{-1}$$

$$= T_1^* F_1 T_1 .$$

Logo, H é hermitiana e pelo Teorema de Fujiwara segue o resultado de Carlson-Datta, ao notar que $InH = InF_1 = InC = InA$.

6.5. A SEGUNDA FORMA HERMITIANA DE FUJIWARA

Neste parágrafo consideramos o método de Fujiwara que descreve o número de raízes de um polinômio f(x) dentro do círculo unitário. Aplicamos este método a uma matriz de Hessenberg A e damos aqui um algorítmo descrevendo o número de valores característicos

de A dentro do círculo unitário. Nosso algorítmo usa o método de Fujiwara via o Algorítmo 5.3. Infelizmente não conseguimos um método computacional que use diretamente a forma hermitiana de Fujiwara via mudança da base. Tal algorítmo seria certamente mais interessante.

TEOREMA 6.7 (Fujiwara [17]). *Sejam* $f(x) = (-1)^n x^n - a_{n-1}x^{n-1} - \ldots - a_o$ *um polinômio e* $h(x) = x^n \overline{f}(\frac{1}{x})$.

Considere a matriz de Bezout B_{fh}, *e a matriz de permutação*

$$P = \begin{bmatrix} 0 & 0 & & 1 \\ 0 & 0 & 1 & 0 \\ & \cdot & \cdot & \cdot \\ 1 & 0 & & 0 \end{bmatrix}$$

é a matriz hermitiana $F_2 = \overline{B}_{fh}\, P$. Suponha que F_2 é não singular. Então

(a) o número de zero de f(x) dentro (fora) do círculo unitário é igual ao número de autovalores positivos (negativos) de F_2.

(b) em particular todos os zeros de f(x) estão dentro do círculo unitário se, e somente se, F_2 é positiva definida.

DEMONSTRAÇÃO. Seja

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ & & & \ddots & \vdots \\ & & & & 1 \\ (-1)^n a_o & (-1)^n a_1 & \cdots & & (-1)^n a_n \end{bmatrix}$$

a matriz companheira de $f(x)$.

Logo, $A^*F_2 - F_2 = A^*\bar{B}_{fh}PA - \bar{B}_{fh}P$

$$= \bar{B}_{fh}\bar{A}\ PA - \bar{B}_{fh}P$$

ao usar o fato de que

$$= \bar{B}_{fh}(\bar{A}\ PA - P) \qquad (6.6)$$

É fácil verificar diretamente que $R = \bar{A}PA - P$ é uma matriz cujas primeiras $(n-1)$ linhas são nulas a última linha

$$r_n \text{ é } (\bar{a}_o a_o - 1,\ \bar{a}_o a_1 + \bar{a}_{n-1}(-1)^n, \ldots, \bar{a}_o a_{n-1} + \bar{a}_1(-1)^n).$$

É fácil verificar que a última linha de $\bar{B}_{fh} = (-1)(\bar{r}_n)$

Então de (6.6) temos

$$A^*F_2A - F_2 = (-1)\ r_n^*\ r_n$$

A não singularidade de

$$F_2 = \bar{B}_{fh}\, P$$

implica que $\bar{B}_{fh}$ é não singular, então pelo Teorema 6.1, temos que $\bar{f}(x)$ e $h(x)$ não tem nenhum zero em comum. Isto por sua vez significa que A não têm nenhum zero com módulo um.

O Teorema 6.8 agora segue imediatamente do Teorema 6.2.

Portanto, de uma matriz de Hessenberg A, descrevemos dois mé todos para a computação do número de autovalores de A dentro (ou fora) do círculo unitário: os dois métodos passam pela matriz companheira C de A.

6.5.1. ALGORÍTMO I.

Sejam A uma matriz de Hessenberg inferior normalizada e N a matriz nilpotente do Teorema 5.3.

PASSO 1: Construir uma matriz triangular inferior com primeira li nha (1,0 ... 0) tal que as primeiras (n-1) linhas de

$$LN - AL = R \quad \text{são nulas} \tag{6.7}$$

PASSO 2: Computar a última linha r_n de R,tal que

$$(-1)^n r_n = (c_o, c_1, \ldots, c_{n-1}).$$

(assim $(-1)^n r_n$ vai ser a última linha da matriz companheira C de A, via o Teorema 5.3).

PASSO 3: Computar

$$\bar{C}PC - P = R_1 \qquad \text{onde}$$

$$P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ & \cdots & & \\ 1 & 0 & & 0 \end{bmatrix} .$$

Seja r'_n a última linha de R_1.

PASSO 4: Construir uma simetrizadora S_1 com última linha $(-1)(\bar{r}'_n)$

PASSO 5: Computar $F_2 = S_1 P$.

Se S_1 é não singular. Então o número de valores característicos de A dentro (fora) do círculo unitário é igual ao número de autovalores positivos (negativos) de F_2.

DEMONSTRAÇÃO DO ALGORÍTMO. A validade deste algorítmo segue imediatamente da demonstração do Teorema 6.8 de Fujiwara via o Teorema 5.3.

6.5.2. ALGORITMO 2. Sejam A uma matriz de Hessenberg inferior normalizada e N a matriz nilpotente do Teorema 5.3.

PASSO 1 e PASSO 2: como no algorítmo anterior.

PASSO 3: Construir uma matriz triangular inferior T com diagonal (1,1, ...,1) tal que

$$TC - AT = 0$$

PASSO 4: Computar

$$P_1 = \bar{T} P T^{-1}$$

PASSO 5: Computar

$$\bar{A} P_1 A - P_1 = R_1$$

onde R_1 é uma matriz cujas (n-1) linhas são nulas.

Seja r_n' a última linha de R_1.

PASSO 6: Construir uma simetrizadora S_1 com última linha $(-r_n')$.

PASSO 7: Computar

$$H_1 = \bar{S}_1 P_1.$$

Se S_1 é não singular, então o número de autovalores de A dentro (fora) do círculo unitário é igual ao número de autovalores positivos (negativos) de H_1.

DEMONSTRAÇÃO DO ALGORÍTMO

Observe que C é a matriz companheira de A e que $C = T^{-1}AT$.

Temos $\bar{C}PC - P = P = R$, como na prova do Teorema 6.8 de Fujiwara. Substituindo-se temos:

$((\bar{T})^{-1}\bar{A}\ \bar{T})P(T^{-1}AT) - P = R$; isto é,

$\bar{A}\ \bar{T}PT^{-1}A - \bar{T}P\bar{T}^{-1} = \bar{T}RT^{-1}$ donde

$\bar{A}P_1A - P_1 = R_1$, onde R_1 tem as suas primeiras (n-1) linhas nulas.

Ora,

$$H = \bar{S}P_1 = ((\bar{T})^{t})^{-1}\ \bar{B}_{fh}(\bar{T})^{-1}\ \bar{T}P\bar{T}^{-1}$$

$$= (T^*)^{-1}\ \bar{B}_{fg}PT^{-1} = T_1^*\ F_2T_1 \text{ , onde}$$

$$T_1 = T^{-1} .$$

6.6. MÉTODO DE FUJIWARA EM PARALELO

Para implementar o método de Fujiwara em paralelo, usamos o fato de que a matriz de Bezout B_{fg} associada a dois polinômios $f(x)$ e $g(x)$, respectivamente de grau n e $m \leq n$, é uma simetrizadora da matriz companheira A de $f(x)$. Como uma matriz companheira é um caso particular de uma matriz de Hessenberg com codiagonal unitária, pelo nosso método de construção de uma simetrizadora em paralelo usando matriz determinante, vemos que as matrizes de Bezout dos teorema de Fujiwara podem ser computada em $0(\log^2 n)$ passos com $[\frac{n^7}{2}]$ processadores. Os produtos DB_{fg} e $\bar{B}_{fg}P$ precisam de um

passo.

Os menores principais líderes F_{11}, F_{12}, F_{13}, etc., de F_1 (ou de F_2) podem ser computados em $0(\log^2 n)$ passos com $0(n^4)$ processadores [20]. Finalmente, para achar o número de variações (permanências) de sinais de $(1, F_{11}, F_{12}, \ldots)$ sequências de menores principais líderes de F_1 (ou analogamente de F_2), nós seguimos o seguinte caminho:

Seja

$$x_o = 1 \ , \quad x_1 = F_{11}, \ x_2 = F_{12}, \ldots, x_n = F_{1n}$$

Inicialmente designamos um mini-computador tendo cada elemento x_i de sequência. Esses mini-computadores contêm as seguintes informações:

(i) O número de N de variações de sinais da sequência acima;

(ii) O sinal do primeiro elemento P da sequência;

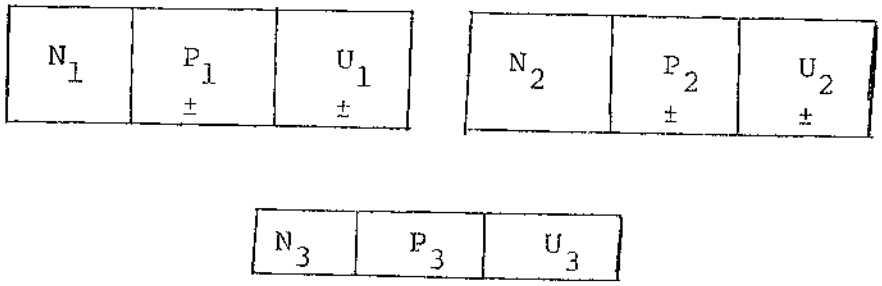
(iii) O sinal do último elemento U da sequência.

Inicialmente seja 2^i o comprimento da sequência $(i = 0, \ldots, k)$ onde

$$2^k \leq \{x_i\} \quad , \quad 0 \leq i \leq n$$

Seja N = 0, neste caso P = U ao sinal do elemento no computador.

Se o computador representa a sequência de comprimento 2^i



O comprimento da próxima sequência será 2^{i+1} e $U_3 = U_1$, $P_3 = P_2$

$$N_3 = N_1 + N_2 + 1 \quad \text{se} \quad P_1 \neq U_2$$

$$N_3 = N_1 + N_2 \quad \text{se} \quad P_1 = U_2$$

Se o comprimento da sequência é n, precisamos de log n passos em paralelo com n processadores.

Então, o número total de passos para a implementação de ambos os métodos de Fujiwara é $0(\log^2 n)$ com $[\frac{n^7}{2}]$ processadores.

6.6.2. MÉTODO DE CARLSON E DATTA EM PARALELO:

NO PASSO 1, precisamos resolver um sistema triangular de ordem $\frac{n(n-1)}{2}$ e precisa-se de $0(\log^2 n)$ passos com $\frac{n^3(n-1)^3}{8}$ processadores.

NO PASSO 2, para achar a última linha r de R precisamos de $0(\log n)$ passos.

NO PASSO 3, construímos uma simetrizadora e nós já sabemos que a construção de uma simetrizadora precisa de $0(\log^2 n)$ passos com $[\frac{n^7}{2}]$ processadores.

NO PASSO 1, para achar H precisamos de mais $0(\log n)$ passos.

Então, para achar H nós precisamos de um total de $0(\log^2 n)$ passos com $[\frac{n^7}{2}]$ processadores.

6.6.3. MÉTODO DE FUJIWARA PARA CÍRCULO UNITÁRIO EM PARALELO.

Como constatamos em vários algorítmos em paralelo nesta tese, a implementação de ambos os algorítmos do Parágrafo 6.5 baseado no método de Fujiwara leva $0(\log^2 n)$ passos com $0[\frac{n^7}{2}]$ processadores. Isto é, uma computação rotineira, visto o papel da M-determinante e da simetrizadora S_1.

BIBLIOGRAFIA

[1] S. BARNETT and C. STORY, Matrix Methods in Stability Theory, Thomas Nelson and Sons Ltd., London, 1970.

[2] R. BELLMAN, Introduction to Matrix Analysis, McGraw-Hill,New York, 1960.

[3] A. BORODIN and I. MUNRO, The Complexity of Algebraic and Numeric Problems, American Elsevier, New York, 1975.

[4] DAVID CARLSON and B.N. DATTA, On the Effective Computation of the Inertia of a Non-Hermitian matrix, Numer.Math.,33, 315-322 (1979).

[5] ———————— , The Lyapunov Matrix Equation SA + A*S = S*B*BS , Linear Algebra Appl., 28 (1979), 43-52.

[6] DAVID CARLSON and H. SCHNEIDER, Inertia Theorems for Matrices: The Semidefinite Case, J. Math. Anal. Appl. 6 (1963), 430-446.

[7] S.C. CHEN and D.J. KUCK, Time and Parallel Process Bounds for Linear Recurrence Systems, IEEE Trans-Computers, C - 24 (1975), 701-717.

[8] L. CSANKY, Fast Parallel Matrix Inversion Algorithms, SIAM J. Comput., 5 (1976), 618-623.

[9] B.N. DATTA, Application of Hankel Matrices of Markov Parameters to the Solutions of the Routh-Hurwitz and Schur-Cohn Problems, J. Math. Anal. Appl. (1979), 276-290.

[10] ————————, An Algorithm for Computing Symmetrizers of an Arbitrary Hessenberg Matrix, Computer Based Numerical Algorithms, E.V. Krishnomarthy and S.K. Sen, Von Nostrand

East and West Press, New Delhi, 261-263 (1976).

[11] ———, On the Routh-Hurwitz-Fujiwara and the Schur-Cohn Fujiwara theorems for the root-separation problem, Linear Alegbra Appl., 22 (1978), 235-246.

[12] ———, On the computation of the characteristic polynomial of a Hessenberg Matrix, the Jour.of the Insdustrial Mathematics Soc., vol. 30, part 1, 1980, 55-60.

[13] B.N. DATTA and KARABI DATTA, An Algorithm for Computing Powers of a Hessenberg Matrix and its Applications, Linear Algebra Appl. 14, 273-284 (1976).

[14] KARABI DATTA, A Implementação em Parelelo de Algoritmo de Álgebra Linear, apresentado no 2º Simpoósio Nacional de Cálculo Numérico, São Carlos, setembro (1979).

[15] ———, An Algorithm to Determine if two Matrices Have Common Eigenvalues, to published in IEEE Trans. On Automatic Control, Vol. Ac-27, No. 5, Oct. 1982.

[15a] ver no fim.

[16] R.J. DUFFIN, Algorithms for Classical Stability Problems, SIAM Rev. 11 (1969), 196-213.

[17] M. FUJIWARA, Uber die Algebraischen Gleichungen, deren Wurzeln in einem Kreise oder in einer Halbebene liegen , Math. Z. 24 (1926), 161-169.

[18] F.R. GANTMACHER, The Theory of Matrices, Vol. I, Vol. II , Chelsar Publishing Company, New York, (1959).

[19] R. GODEMENT, Cours d'algebre, Hermann, Paris, 1966.

[20] DON HELLER, A determinant Theorem with Applications to Parallel Algorithms, SIAM J. Anal., 11 (1974), 559-568.

[21] ———, A Survey of Parallel Algorithms in Numerical Linear Algebra, SIAM Rev. Vol. 20, Nº 4, oct. (1978).

[22] K. HOFFMAN and R. KUNZE, Linear Algebra, Prentice Hall, N.J. 1961.

[23] M.G. KREIN and M.A. NAIMARK, The method of Symmetric and Hermitian Forms in the Theory of the Separation of the Roots of Algebraic Equations, originally published in Kharkov (1936), Translation prepared by O. Boshko and J.L. Howland, Linear and Multilinear Algebra, 1981, vol. 10, 265-308.

[24] L.G. KHACHIAN, Doklady Akademia Nouk USSR, 1979, vol. 224, nº 5, 1093-1096.

[25] D.G. KUCK, Parallel processors, Architecture, A Survey, Sagamore Computer Conference on Parallel proceesing, 1975.

[26] PETER LANCASTER, Theory of Matrices, Academic Press, New York, 1969.

[27] W.L. MIRANKER, A survey of Parallelism in Numerical Analysis, SIAM REV, 13 (1971), 524-547.

[28] Y. MUROKA and D.J. KUCK, On the time Required for a sequence of matrix products, Comm. ACM, 16 (1973), 22-26.

[29] M.C. PEASE, Matrix Inversion using Parallel Processing, J. Assoc. Comp. Mach. 14 (1967), 757-764.

[30] ————, Inversion of Matrices by Partitions, Ibid., 16 (1969), 302-314.

[31] F.P. PREPARATA and D.V. SARWATE, An Improved Parallel Processor Bound in Fast Matrix Inversion, Information Processing letters, (1978), 148-150.

[32] A.H. SAMEH and D.J. KUCK, A Parallel QR Algorithm for Symmetric tridiagonal Matrices, IEEE. Trans. Comp. C - 26 (1977), 147-153.

[33] DAVID, H. SCHAEFER, NASA End to End Data System (NEED), Massively Parallel Processor (MPP) Concept Document, Project Report, Prepared by Computer Development Section, Goddard Space Flight Centre, Greenbelt, MD, USA.

[34] H.R. SCHWARZ, Ein Verfahren Zur Stabilitatsfragbei matrizereigenwerte Problema. Z. Angun. Math. Phys. (1956), 473-500.

[35] DAVID STEVENSON, Parallel Computers in the 1980s, Technical Report, Institute for Advanced Computation, NASA Amer. Research Centre, Moherr field, California, USA.

[36] G.W. STWEART, Introduction to Matrix Computations, Academic Press, New York, 1973.

[37] H.S. STONE, An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations, J. Assoc. Comput. Mach., 20 (1973), 27-38.

[38] O. TAUSSKY and A. ZASSENHAUS, On the Similarity Tranformation Between a Matrix and its Transpose, Pacific J. Math. Vol. 7 (1959), 893-896.

[39] J.F. TRAUB, Complexity of Sequential and Parallel Numerical Algorithms, Academic Press, New York, 1973.

[40] B.L. VANDER WAERDEN, Algebra, vol.1, Frederick Ungar Publishing Co. New York, 1970.

[41] J.H. WILKINSON, The Algebraic Eigenvalue Problem, Oxford University Press, London, (1965).

[15a] J.J. DONGARRA, C.B. MOLER, J.R. BUNCH and G.W. STEWART, Linpack User's Guide, SIAM, Philadelphia, 1979.