UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INSTITUTO DE MATEMATICA

ALGORITMOS DA RECONSTRUÇÃO ESPECTRAL PARA MATRIZES DE JACOBI

por

INÊS FERREIRA MORAES

Dissertação submetida ao curso de Pós-graduação em Matemática como requisito parcial para a obtenção do grau de mestre.

Orientadora

Profa. Dra. Teresa Tsukazan de Ruiz

Porto Alegre, Março de 1993.

Dedico este trabalho
ao meu marido e aos
meus pais.

AGRADECIMENTOS

A Professora Teresa Tsukazan de Ruiz pela orientação na elaboração deste trabalho e ao Professor Julio Cesar Ruiz Claeyssen pelo auxílio.

Aos meus pais Daito e Ana, pela ajuda, compreensão e estímulo.

Ao meu irmão Ivan e minha cunhada Rosana, pelo carinho e incentivo.

As amigas Rosemaira e Sandra, pelo apoio e amizade.

RESUMO

A reconstrução de matrizes de Jacobi a partir de dados espectrais é de grande importância na Teoria Vibracional. Usamos três métodos distintos para tal reconstrução: a aproximação por polinômios ortogonais, a iteração de Lanczos e a redução de Householder. Eles são aplicados a um sistema massa-mola supondo conhecidos os pólos e zeros da função frequência resposta correspondente a uma força senoidal aplicada em um dos extremos ou em um ponto interior. Resultados numéricos são obtidos com o software MATLAB e a linguagem FORTRAN.

ABSTRACT

The reconstruction of Jacobi matrices from spectral data is of great importance in vibration Theory. We use three distinct methods for such reconstruction: the orthogonal polynomial approach, the Lanczos iteration and the Householder reduction. They are applied to a spring-mass system by assuming to know the poles and zeros of the frequency response function corresponding to a sinusoidal forcing at an end or an interior point. Numerical results are obtained with MATLAB and FORTRAN codes.

INDICE

Capítulo 3

RECONSTRUÇÃO DE UMA MATRIZ DE JACOBI

UTILIZANDO-SE O ALGORITMO DE LANCZOS

Capítulo 4

RECONSTRUÇÃO DE UMA MATRIZ DE JACOBI

USANDO TRANSFORMAÇÃO DE HOUSEHOLDER

Capítulo 5

RECONSTRUÇÃO DE UMA MATRIZ DE JACOBI

A PARTIR DE DADOS ESPECTRAIS

Capítulo 6

APLICAÇÃO DOS MÉTODOS DE RECONSTRUÇÃO DE UMA MATRIZ DE JACOBI A UM SISTEMA VIBRACIONAL

Capítulo 7

RESULTADOS NUMÉRICOS

Apêndice A

INTRODUÇÃO

Neste trabalho descreveremos alguns métodos de reconstrução de matrizes de Jacobi. Estas matrizes além de serem tridiagonais possuem a característica de que os elementos das duas codiagonais possuem o mesmo sinal.

Nosso interesse em estudar este tipo de matriz está na sua aplicabilidade principalmente a problemas vibracionais.

Na teoria clássica de vibrações os problemas inversos preocupam-se com a reconstrução de um modelo dado, este pode ser um sistema mola-massa, uma corda, um sistema vibracional de um prédio, etc., tendo como dados os autovalores e/ou autovetores, isto é, os dados espectrais. Pode ocorrer que esta construção do sistema seja única ou não.

Veremos que tipos de dados espectrais são necessários e suficientes para podermos reconstruir uma matriz de Jacobi.

No capítulo 1 trataremos das propriedades destas matrizes e observaremos que os seus autovalores são todos distintos e que os autovalores de duas de suas submatrizes principais sucessivas se entrelaçam.

Descreveremos a relação existente entre os polinômios característicos das submatrizes principais de uma matriz de Jacobi com uma sequência de polinômios mônicos ortogonais. É por esta relação, que podemos gerar uma sequência de polinômios mônicos ortogonais de forma que estes representem os polinômios característicos das submatrizes. Tal propriedade permitiu que De Boor e Golub desenvolvessem um método de reconstrução de uma matriz de Jacobi a partir de dados espectrais através de polinômios ortogonais. Este método será visto com detalhes no capítulo 2.

No capítulo 3 apresentaremos um outro processo também numericamente estável que faz a reconstrução utilizando o algoritmo de Lanczos. Para realizarmos tal construção é necessário e suficiente que conheçamos os autovalores da matriz a ser reconstruída e as componentes da última (ou primeira) linha da matriz dos autovetores. Este método foi originalmente estudado para reconstrução de matrizes banda, mas como uma matriz de Jacobi é uma matriz tridiagonal (banda 2) pode ser adaptado, desde que se leve em conta que queremos reconstruir uma matriz que tenha os mesmos sinais nos elementos das duas codiagonais.

No quarto capítulo descreveremos a reconstrução através das transformações de Householder. Mas percebemos que devido ao grande número de operações que se deve realizar para chegar a matriz desejada, ele não é o método mais indicado, embora seja numericamente estável. Sua dificuldade também está na escolha adequada de sinais que

tem que ser feita logo no início do processo.

No capítulo 5 analisaremos uma reconstrução de uma matriz de Jacobi sendo que conhecemos os valores espectrais da matriz a ser reconstruída e os autovalores de duas de suas submatrizes principais, sendo que uma corresponde ao lado esquerdo superior da matriz original e a outra ao lado direito inferior.

Vemos que neste tipo de problema inverso existem dois casos a se analisar: quando os autovalores destas duas submatrizes são disjuntos e quando possuem um par ou mais de autovalores idênticos. Na primeira situação podemos garantir a unicidade da reconstrução, mas no segundo caso existe uma família de soluções que satisfazem os dados iniciais .

Devido a importância desse tipo de matriz em problemas inversos resolvemos ilustrar tais reconstruções fazendo uma aplicação destas ao modelo vibracional de molas. Assim a partir da reconstrução de uma matriz de Jacobi onde os seus autovalores corresponderão as frequências naturais do sistema, obtemos as matrizes inércia e de rigidez.

Tal obtenção poderá ser única desde que a reconstrução da matriz de Jacobi desejada seja única.

Como último item faremos uma análise dos resultados numéricos obtidos a partir da implementação destas reconstruções.

CAPITULO 1

MATRIZES DE JACOBI

1.1 - INTRODUÇÃO

Neste primeiro capítulo, introduziremos as matrizes de Jacobi e suas propriedades, o que será utilizado no desenvolvimento deste trabalho.

Estabeleceremos que os polinômios característicos das submatrizes principais de uma matriz de Jacobi formam uma sequência de Sturm. Através das propriedades que esta sequência possui caracterizaremos o espectro da mesma e de suas submatrizes principais. Em suma, concluíremos que todos os seus autovalores são distintos e que os autovalores de duas de suas submatrizes principais sucessivas se entrelaçam.

Os autovetores associados a autovalores de uma matriz de Jacobi possuem a propriedade de que a sua última (ou primeira) componente é não nula. Tal característica será apresentada na seção 1.3.

Na seção 1.4 trataremos de polinômios

ortogonais. Veremos que existe uma única sequência de polinômios mônicos ortogonais que pode ser gerada a partir de uma fórmula de recorrência, a qual satisfazem os polinômios característicos das submatrizes principais de uma matriz de Jacobi, desde que se utilize um produto interno discreto com peso adequado.

1.2 - DEFINIÇÃO DE MATRIZ DE JACOBI E CARACTERIZAÇÃO DOS POLINÔMIOS CARACTERÍSTICOS DE SUAS SUBMATRIZES PRINCIPAIS

DEFINIÇÃO 1.2.1 - Uma matriz simétrica tridiagonal A de ordem n, onde os elementos das diagonais secundárias, adjacentes a principal, são não nulos e de mesmo sinal denomina-se *matriz de Jacobi*. Sem perda de generalidade, consideremos os elementos não nulos fora da diagonal principal negativos, que é o caso de matrizes que provém da discretização numérica do laplaciano unidimensional.

Denotemos A por

$$A = \begin{bmatrix} a_1 & -b_1 & 0 & \cdots & 0 \\ -b_1 & a_2 & -b_2 & & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & a_{n-2} & -b_{n-2} & 0 \\ 0 & & -b_{n-2} & a_{n-1} & -b_{n-1} \\ 0 & \cdots & 0 & -b_{n-1} & a_n \end{bmatrix}, \text{ com } b_r > 0. \qquad (1.2.1)$$

Seja A uma matriz de Jacobi da forma (1.2.1). Consideremos os menores principais de (A-λI) denotados por

$$P_0(\lambda) \equiv 1 \; ; \quad P_1(\lambda) = a_1 - \lambda \; ; \quad P_2(\lambda) = \begin{vmatrix} a_1 - \lambda & -b_1 \\ -b_1 & a_2 - \lambda \end{vmatrix} \; ; \quad \ldots \tag{1.2.2}$$

Assim,

$$P_n(\lambda) = |A - \lambda I| \; . \tag{1.2.3}$$

Estes menores principais são exatamente os polinômios característicos das submatrizes principais esquerdas de A, e os denotemos por

$$P_r(\lambda) = |A_r - \lambda I_r| \; , \quad r = 1, 2, \ldots, n \; . \tag{1.2.4}$$

A sequência $(P_r)_0^n$ de polinômios característicos satisfaz a fórmula de recorrência,

$$P_{r+1}(\lambda) = (a_{r+1} - \lambda) P_r(\lambda) - b_r^2 P_{r-1}(\lambda) \quad r = 1, 2, \ldots, n-1$$
$$P_0(\lambda) \equiv 1 \; . \tag{1.2.5}$$

De fato, para r = 1 temos

$$P_2(\lambda) = (a_1 - \lambda)(a_2 - \lambda) - b_1^2 = (a_2 - \lambda) P_1(\lambda) - b_1^2 P_0(\lambda)$$

e, para k qualquer, com $2 \leq k \leq n-1$, temos

$$P_{k+1}(\lambda) = |A_{k+1} - \lambda I_{k+1}| = \left|\begin{array}{ccc:c} & A_k - \lambda I_k & & \begin{matrix}0\\ \vdots \\ 0\\ -b_k\end{matrix} \\ \hdashline 0 & \cdots & 0 & -b_k \quad (a_{k+1} - \lambda) \end{array}\right| =$$

$$= (a_{k+1} - \lambda)|A_k - \lambda I_k| + b_k \left|\begin{array}{cccc:c} a_1 - \lambda & -b_1 & \cdots & 0 & 0 \\ -b_1 & a_2 - \lambda & -b_2 & \vdots & \vdots \\ \vdots & & \ddots & -b_{k-2} & \\ 0 & \cdots & & a_{k-1} - \lambda & 0 \\ \hdashline 0 & & & -b_{k-1} & -b_k \end{array}\right| =$$

$$= (a_{k+1} - \lambda)P_k(\lambda) + b_k(-b_k)|A_{k-1} - \lambda I_{k-1}| =$$

$$= (a_{k+1} - \lambda)P_k(\lambda) - b_k^2 P_{k-1}(\lambda).$$

Portanto, a recorrência (1.2.5) permiti-nos calcular P_2, P_3, ..., P_n sucessivamente a partir de P_0 e P_1.

1.3 - SEQUÊNCIAS DE STURM

DEFINIÇÃO 1.3.1 - Uma sequência de Sturm é uma sequência de polinômios reais $P_0(\lambda)$, $P_1(\lambda)$, ..., $P_n(\lambda)$, a qual satisfaz as seguintes condições:

1. $P_0(\lambda)$ tem em toda parte o mesmo sinal ($P_0(\lambda) \equiv 1$);

2. Quando $P_r(\lambda)$ se anula, $P_{r+1}(\lambda)$ e $P_{r-1}(\lambda)$

são não nulos e têm sinais opostos.

TEOREMA 1.3.1 - Se A é uma matriz de Jacobi, então os polinômios característicos de suas submatrizes principais formam uma sequência de Sturm.

Prova:

Seja A uma matriz de Jacobi de ordem n. De acordo com a seção anterior temos que os polinômios característicos das submatrizes principais de A satisfazem a recorrência (1.2.5), donde vem que $P_0(\lambda) \equiv 1$. Dessa forma a primeira condição das sequências de Sturm já está estabelecida.

Quanto a segunda condição, observemos que dois sucessivos P_r não podem ser simultaneamente zero, isto é, para o mesmo $\lambda = \lambda_0$, pois se $P_{s+1}(\lambda_0) = 0 = P_s(\lambda_0)$ a recorrência (1.2.5) mostra-nos que $P_{s-1}(\lambda_0) = 0$, de maneira que finalmente devemos ter P_1 e P_0 nulos; mas $P_0(\lambda) = 1$, chegando-se a uma contradição.

A última parte da condição segue, agora, diretamente da recorrência (1.2.5). ■

Será através das propriedades, que as sequências de Sturm conferem à sequência de polinômios característicos das submatrizes principais de A, que chegaremos a resultados importantes sobre o espectro das matrizes de Jacobi.

DEFINIÇÃO 1.3.2 - Definimos a função de Sturm, denotada por $s_r(\lambda)$, como sendo a função que corresponde ao número cumulativo de mudanças de sinais na sequência $P_0, P_1(\lambda), \ldots, P_r(\lambda)$.

TEOREMA 1.3.2 - A função de Sturm $s_r(\lambda)$ muda somente quando λ passa através de um zero do último polinômio, $P_r(\lambda)$.

Prova:

É claro que $s_r(\lambda)$, conforme definida anteriormente, pode mudar somente quando λ passa através de um zero de um dos $P_s(\lambda)$ $(s \leq r)$. É, portanto, suficiente mostrar que $s_r(\lambda)$ não muda quando λ passa através de um zero de um dos $P_s(\lambda)$ intermediários $(s < r)$. Suponhamos que $P_s(\lambda_0) = 0$, onde $1 \leq s < r$, então $P_{s-1}(\lambda_0)$ e $P_{s+1}(\lambda_0)$ possuem sinais opostos. Os sinais do conjunto $P_{s-1}(\lambda_0)$, $P_s(\lambda_0)$, $P_{s+1}(\lambda_0)$ são, portanto, + 0 - ou - 0 + . Suponhamos que o primeiro caso ocorre e que $P_s(\lambda)$ cresce conforme λ passa através de λ_0. Portanto para valores de λ suficientemente próximos de λ_0, porém menores, os sinais são + - -, ao passo que para valores de λ suficientemente próximos de λ_0, porém maiores, os sinais são + + - . Logo não importa se λ é maior ou menor que λ_0, ocorrendo apenas uma mudança de sinal no conjunto. Em outras palavras, tais polinômios não contribuem em qualquer mudança de $s_r(\lambda)$ conforme λ passa através de λ_0. Mas nenhum outro elemento da sequência contribuirá em qualquer mudança de $s_r(\lambda)$

conforme λ passa através de λ_0 (se λ_0 for um zero de outro polinômio, $P_t(\lambda)$, $|t-s| \geq 2$, não haverá novamente mudança em $s_r(\lambda)$) de modo que $s_r(\lambda)$ não mudará no todo. As outras três possibilidades produziram a mesma conclusão para argumentos análogos, completando assim a demonstração. ■

TEOREMA 1.3.3 - Os zeros de $P_r(\lambda)$ são reais e simples. Além disso, se $P_r(\lambda_0) \neq 0$ e $s_r(\lambda_0) = k$, então $P_r(\lambda)$ tem k zeros menores do que λ_0.

A demonstração será feita para o caso em que os polinômios da sequência de Sturm são os polinômios característicos das submatrizes principais de uma matriz de Jacobi.

Prova:

Como $P_s(\lambda) = (-)^s\lambda^s + \ldots$, todos $P_s(\lambda)$ serão positivos para λ negativo e suficientemente grande em módulo, isto é, $\lambda \leq \alpha$, de modo que $s_r(\alpha) = 0$: α pode ser tomado igual a zero se A for positiva definida. Por outro lado, para λ positivo e suficientemente grande, isto é, $\lambda \geq \beta$, o $P_s(\lambda)$ alternará em sinal, de modo que $s_r(\beta) = r$. Desde que $s_r(\lambda)$ cresce somente quando λ passa através de um zero de $P_r(\lambda)$, todos os zeros de $P_r(\lambda)$ devem ser distintos, pois se λ_0 for um zero de multiplicidade par, então $P_r(\lambda) = (\lambda_0-\lambda)^{2m}(-1)^{r-2m}\lambda^{r-2m} + \ldots$, para algum $m \in Z^+$, de modo que $s_r(\lambda)$ não cresceria em absoluto conforme

λ passasse através de λ_o. Se λ_o fosse um zero de multiplicidade ímpar, $P_r(\lambda) = (\lambda_o-\lambda)^{2m-1}(-1)^{r-2m+1}\lambda^{r-2m+1} + \ldots$, para algum $m \in Z^+$ de modo que $s_r(\lambda)$ cresceria somente por unidade conforme λ passasse através de λ_o.

A segunda parte do teorema segue imediatamente. ■

COROLÁRIO 1.3.3.1 - Os autovalores de uma matriz de Jacobi são distintos.

COROLÁRIO 1.3.3.2 - O número de zeros de $P_r(\lambda)$ satisfazendo $\alpha < \lambda < \beta$ é igual a $s_r(\beta) - s_r(\alpha)$.

COROLÁRIO 1.3.3.3 - Se λ_o é um zero de $P_r(\lambda)$ então conforme λ passa através de λ_o- e λ_o+ o sinal de $P_{r-1}(\lambda)P_r(\lambda)$ muda de + para - e $s_r(\lambda)$ cresce por unidade.

Prova:

Seja λ_o um zero de $P_r(\lambda)$ e consideremos $s_r(\lambda_o-) = k$ $(k < r)$. Temos, pelo teorema 1.3.2, que $P_{r-1}(\lambda_o-)$ e $P_r(\lambda_o-)$ possuem o mesmo sinal, ou seja, $P_{r-1}(\lambda_o-)P_r(\lambda_o-) > 0$, da mesma forma, $P_{r-1}(\lambda_o+)$ mantém o sinal mas $P_r(\lambda_o+)$ mudou. Logo $P_{r-1}(\lambda_o+)P_r(\lambda_o+) < 0$ e $s_r(\lambda_o+) = k+1$, pois as raízes de $P_r(\lambda)$ são simples. ■

TEOREMA 1.3.4 - Entre quaisquer dois zeros vizinhos de $P_r(\lambda)$ existe um e somente um zero de $P_{r-1}(\lambda)$, e um e somente um

zero de $P_{r+1}(\lambda)$.

Prova:

Sejam μ_1, μ_2 dois zeros vizinhos de $P_r(\lambda)$. Suponhamos que $P_r(\mu_1-) > 0$, então $P_r(\mu_1+) < 0$ e $P_r(\mu_2-) < 0$. Pelo corolário 1.3.3.3, $P_{r-1}(\mu_1+) > 0$ e $P_{r-1}(\mu_2-) < 0$, de modo que $P_{r-1}(\lambda)$ muda de sinal entre μ_1+ e μ_2-, e portanto tem no mínimo um zero em (μ_1, μ_2).

A condição 2 das sequências de Sturm mostra que $P_{r+1}(\mu_i)$ e $P_{r-1}(\mu_i)$; $(i=1,2)$ têm sinais opostos. Assim $P_{r+1}(\mu_1+) < 0$, $P_{r+1}(\mu_2-) > 0$ de modo que $P_{r+1}(\lambda)$ tem no mínimo um zero em (μ_1, μ_2). Agora, suponhamos que $P_{r-1}(\lambda)$ (ou $P_{r+1}(\lambda)$) tenha duas raízes (ou mais) em (μ_1, μ_2) então $P_r(\lambda)$ tem um zero em (μ_1, μ_2), o que contraria a hipótese de que μ_1 e μ_2 são zeros vizinhos. ■

Este teorema nos permite afirmar que:

"Os autovalores de duas submatrizes principais sucessivas de uma matriz de Jacobi se entrelaçam."

1.4 - AUTOVETORES DE UMA MATRIZ DE JACOBI

Os autovetores associados a autovalores de uma matriz de Jacobi possuem a propriedade de que a sua última (ou primeira) componente é não nula. Esta característica será uma condição necessária para a reconstrução da matriz

de Jacobi através do algoritmo de Lanczos no capítulo 3. Mais precisamente,

PROPOSIÇÃO 1.4.1 - Seja A uma matriz de ordem n. Se A for uma matriz de Jacobi, então a primeira (ou última) componente de qualquer autovetor é necessariamente não nula.

Prova:

Suponhamos que $u^{(j)}$ é um autovetor de uma matriz de Jacobi A. Se a primeira componente de $u^{(j)}$, $u_1^{(j)}$, for nula, então a primeira equação correspondente ao problema de autovalor,

$$(A-\lambda_j I)u^{(j)} = 0, \qquad (1.4.1)$$

escreve-se,

$$b_1 u_2^{(j)} = 0 . \qquad (1.4.2)$$

Como b_1 é não nulo, decorre que $u_2^{(j)} = 0$.

O restante das componentes de $u^{(j)}$ podem ser determinadas recursivamente usando a r-ésima equação de (1.4.1):

$$-b_{r-1}u_{r-1}^{(j)} + (a_r-\lambda_j)u_r^{(j)} - b_r u_{r+1}^{(j)} = 0 \ , \ r=2,3,\ldots,n \ . \qquad (1.4.3)$$

Como $u_1^{(j)} = u_2^{(j)} = 0$, é claro que a equação (1.4.3) exige que

$u^{(j)} = 0$, desde que $b_r \neq 0$, $\forall\ r=1,2,\ldots,n-1$. Logo $u^{(j)}$ é um vetor nulo, o que não pode ser. Analogamente, se $u_n^{(j)} = 0$, então a equação (1.4.3), com $b_n = 0$ exige que $u_{n-1}^{(j)} = 0$ e, da fórmula recursiva resulta novamente que $u^{(j)}$ deve ser um vetor nulo. ■

1.5 - POLINÔMIOS ORTOGONAIS

Existe uma relação íntima entre matrizes de Jacobi e os polinômios ortogonais. Veremos inicialmente algumas propriedades básicas desses polinômios.

Denotemos por P_n, o espaço linear de todos os polinômios reais com grau $i < n$, de ordem n.

Para dois polinômios quaisquer $p(x)$ e $q(x)$ no espaço P_n, definimos

$$\langle p,q\rangle \equiv \sum_{i=1}^{n} \omega_i\, p(\xi_i)q(\xi_i)\ , \qquad (1.5.1)$$

com $\omega_i > 0$ para $i=1,2,\ldots,n$ e, n pontos $\xi_1<\xi_2<\ldots<\xi_n$.

No espaço P_n, $\langle\ ,\ \rangle$ define um produto interno, desde que ele seja positivo definido, bilinear e simétrico, isto é,

1) $\langle p,p\rangle \equiv \|p\|^2 > 0$ se $p(x) \neq 0$,

2) $\langle \alpha p, q \rangle = \alpha \langle p, q \rangle$, para α real

$\langle p+q, r \rangle = \langle p, q \rangle + \langle q, r \rangle$,

3) $\langle p, q \rangle = \langle q, p \rangle$.

Além disso,

4) $\langle xp, q \rangle = \langle p, xq \rangle$.

Dois polinômios $p(x)$, $q(x)$ são ditos ortogonais se $\langle p, q \rangle = 0$.

TEOREMA 1.5.1 - Existe uma única sequência de polinômios mônicos, isto é, $(q_i(x))_0^n$, tal que $q_i(x)$ tem grau i e o coeficiente líder (de x^i) é unitário, os quais são ortogonais com respeito ao produto interno $\langle \ , \ \rangle$, ou seja,

$$\langle q_i , q_j \rangle = 0 \ , \ i \neq j \ . \qquad (1.5.2)$$

Prova:

Os $q_i(x)$ podem ser construídos pela aplicação do processo de ortogonalização de Gram-Schmidt para polinômios linearmente independentes $(x^i)_0^{n-1}$. Assim

$$q_0 \equiv 1 \ , \quad q_i(x) = x^i - \sum_{j=0}^{i-1} \alpha_{i\,j} q_j(x) \ , \ i=1,2,\ldots,n-1 \qquad (1.5.3)$$

onde

$$\langle q_i , q_j \rangle = \langle x^i , q_j \rangle - \alpha_{i\,j} \langle q_j , q_j \rangle = 0, \qquad (1.5.4)$$

de modo que

$$\alpha_{i\,j} = \frac{\langle x^i , q_j \rangle}{\|q_j\|^2} \quad , \; j=0,1,\ldots,i-1. \qquad (1.5.5)$$

Observemos que o polinômio

$$q_n(x) = \prod_{i=1}^{n} (x-\xi_i), \qquad (1.5.6)$$

é um polinômio mônico de grau n na sequência, e também ortogonal a $(q_i)_0^{n-1}$. De fato é ortogonal a todas as funções, desde que $q_n(\xi_i) = 0$ $(i=1,2,\ldots,n)$. ■

O processo de Gram-Schmidt não proporciona um meio computacionalmente estável para a construção da sequência de polinômios ortogonais. Portanto usaremos o método desenvolvido por Forsythe [5] para a construção dessa sequência.

TEOREMA 1.5.2 - No espaço P_{n+1}, os polinômios mônicos ortogonais, $(q_i)_0^n$ satisfazem a relação de recorrência de três termos:

$$q_i(x) = (x-\alpha_i)q_{i-1}(x) - \beta_{i-1}^2 q_{i-2}(x), \quad i=1,2,\ldots,n \qquad (1.5.7)$$

com valores iniciais,

$$q_{-1}(x) = 0 \ , \ q_0(x) = 1. \qquad (1.5.8)$$

Prova:

O polinômio $(q_i(x)-xq_{i-1}(x))$ é de grau $i-1$, pode portanto, ser expresso em termos de $q_0, q_1, \ldots, q_{i-1}$. Assim,

$$q_i(x) - xq_{i-1}(x) = c_0q_0 + c_1q_1 + \ldots + c_{i-1}q_{i-1} \ . \qquad (1.5.9)$$

Fazendo o produto interno dessa equação com $q_j(x)$, $(j=0,1,\ldots,i-1)$, temos

$$\langle q_i, q_j\rangle - \langle q_{i-1}, xq_j\rangle = \sum_{k=0}^{i-1} c_k\langle q_k, q_j\rangle = c_j \|q_j\|^2 \ , \qquad (1.5.10)$$

onde o segundo termo do lado esquerdo foi reescrito usando a propriedade 4 do produto interno. Mas se $j=0,1,\ldots,i-1$, o primeiro termo do lado esquerdo é zero e, se $j=0,1,\ldots,i-3$, xq_j é de grau no máximo $i-2$ e assim é ortogonal a q_{i-1}. Temos então $c_j = 0$ se $j=0,1,\ldots,i-3$, e há dessa forma somente dois termos não nulos, c_{i-1} e c_{i-2}, no lado direito de (1.5.9), ou seja,

$$q_i(x) - xq_{i-1}(x) = c_{i-2}q_{i-2}(x) + c_{i-1}q_{i-1}(x). \qquad (1.5.11)$$

Logo, da equação (1.5.10) temos

$$\alpha_i = -c_{i-1} = \frac{\langle q_{i-1}, xq_{i-1}\rangle}{\|q_{i-1}\|^2}, \qquad (1.5.12)$$

$$c_{i-2} = -\frac{\langle q_{i-1}, xq_{i-2}\rangle}{\|q_{i-2}\|^2}. \qquad (1.5.13)$$

Mas xq_{i-2} é um polinômio mônico de grau i-1, podendo ser expresso como

$$xq_{i-2}(x) = q_{i-1}(x) + \sum_{j=0}^{i-2} d_j q_j(x) \qquad (1.5.14)$$

de tal forma que

$$\langle q_{i-1}, xq_{i-2}\rangle = \|q_{i-1}\|^2. \qquad (1.5.15)$$

Reescrevendo (1.5.13) obtemos

$$c_{i-2} = -\frac{\|q_{i-1}\|^2}{\|q_{i-2}\|^2} \qquad (1.5.16)$$

e, como c_{i-2} é negativo e igual a $-\beta_{i-1}^2$ temos

$$\beta_i = \frac{\|q_i\|}{\|q_{i-1}\|} \cdot \blacksquare \qquad (1.5.17)$$

As equações (1.5.7) e (1.5.8) com (1.5.12) e (1.5.17) permiti-nos calcular os polinômios $(q_i)_0^{n-1}$ etapa por etapa: com $q_{-1} = 0$ e $q_0 = 1$; primeiro calcula-se α_1 a partir de (1.5.12) e o qual substituído em (1.5.7) dá q_1. Calcula-se depois α_2 e β_1, e encontra-se q_2, assim por diante.

CAPITULO 2

RECONSTRUÇÃO DE UMA MATRIZ DE JACOBI ATRAVÉS DOS POLINÔMIOS CARACTERÍSTICOS DE SUAS SUBMATRIZES

2.1 - INTRODUÇÃO

Estamos interessados em resolver o seguinte problema inverso:

Dadas as sequências $\lambda = (\lambda_i)_1^n$ *e* $\mu = (\mu_i)_1^{n-1}$ *com*

$$\lambda_i < \mu_i < \lambda_{i+1} \; ; \qquad i=1,2,\ldots,n-1, \tag{2.1.1}$$

construir uma matriz de Jacobi A de ordem n que tenha como autovalores $\lambda_1, \lambda_2, \ldots, \lambda_n$ *e autovalores de sua submatriz principal esquerda* A_{n-1}, *de ordem n-1, os valores* $\mu_1, \mu_2, \ldots, \mu_{n-1}$.

Observemos que (2.1.1) é uma condição necessária para a existência de solução do problema.

O problema foi estudado primeiramente por Hochstadt [15]. Ele provou que se os elementos das

codiagonais de uma matriz de Jacobi são fixados então os elementos das diagonais são unicamente determinados através dos autovalores dados. O resultado completo é apresentado em [16], onde Hochstadt abandona a técnica anterior em favor de uma aproximação construtiva a qual, entretanto presupõe a existência de uma solução. Na tentativa de construir a matriz unicamente, não mostrou que seu método levaria sempre a valores reais os elementos b_i, das codiagonais. Provou, ainda neste trabalho, que a matriz de Jacobi depende continuamente sobre λ e μ.

Hald [13] apresentou outra construção e mostrou que, em teoria, Hochstadt sempre trabalhou sob a condição que os autovalores satisfizessem a condição de entrelaçamento. Na prática a construção feita por Hald não é estável. Hald também mostrou que a construção de Hochstadt conduzia a elementos b_i reais. Ele mostrou também com mais detalhes a dependência contínua de A sobre λ e μ e fez ainda uma prova independente de unicidade.

Descreveremos sucintamente, nesse capítulo, o método de Hald. Embora não seja estável tal método prova que o problema possui solução e esta é única.

Demonstraremos que a matriz de Jacobi a ser construída é única, com base na recorrência que seus polinômios característicos satisfazem. Devemos observar que é somente possível garantir a unicidade da construção quando os sinais dos elementos fora da diagonal principal são fixados. Neste trabalho, como já foi comentado

anteriormente, estamos considerando os elementos fora da diagonal principal todos negativos. Na prática, fazer tal escolha não é uma dificuldade séria, porque estes sinais são normalmente dados pelo modelo físico.

Nos deteremos em expor o método de construção desenvolvido por De Boor e Golub [4]. Este método constrói uma sequência de polinômios mônicos ortogonais, de forma que estes coincidam com os polinômios característicos das submatrizes principais esquerdas da matriz de Jacobi A; sendo que eles são gerados através de uma relação de recorrência desenvolvida por Forsythe [5], a qual fornece um meio estável de cálculo dos valores da matriz.

2.2 - RESOLUÇÃO DO PROBLEMA POR HALD

Da análise feita no capítulo 1, obtemos que a sequência de polinômios característicos das submatrizes principais esquerdas satisfazem a fórmula de recorrência dada por

$$p_{r+1}(\lambda) = (\lambda - a_{r+1})p_r(\lambda) - b_r^2 p_{r-1}(\lambda), \quad r=1,2,\ldots,n-1 \quad (1)$$
$$p_0(\lambda) \equiv 1 \qquad (2.2.1)$$

---

(1) - Por conveniência utilizaremos a partir de agora $p_r(\lambda) = |\lambda I - A|$ no lugar de $p_r(\lambda) = |A - \lambda I|$, trabalharemos assim com polinômios mônicos.

Da mesma, forma se tivermos uma sequência de polinômios mônicos, onde cada p_i possui grau i, que satisfaça a fórmula de recorrência (2.2.1), então pode-se afirmar que existe uma matriz de Jacobi onde esses polinômios são exatamente os polinômios característicos de suas submatrizes principais esquerdas. Como os zeros de p_r são os autovalores de A_r, para todo r, podemos propor o problema anterior da seguinte maneira:

Dadas as sequências $\lambda = (\lambda_i)_1^n$ *e* $\mu = (\mu_i)_1^{n-1}$ *com*

$$\lambda_i < \mu_i < \lambda_{i+1} ; \qquad i=1,2,\ldots,n-1, \tag{2.2.2}$$

construir sequências $a = (a_r)_1^n$ *e* $b = (b_r)_1^{n-1}$ *de modo que a sequência* $(p_r)_1^n$ *de polinômios obtida a partir de* (2.2.1) *satisfaça*

$$p_{n-1}(\lambda) = \prod_{j=1}^{n-1}(\lambda-\mu_j) \quad e \quad p_n(\lambda) = \prod_{j=1}^{n}(\lambda-\lambda_j). \tag{2.2.3}$$

Segundo Hald, esse problema tem no máximo uma solução, pois se já conhecemos os polinômios mônicos p_{r+1} e p_r então a_{r+1} é unicamente determinado, bastando exigir que $q(\lambda) = p_{r+1}(\lambda) - (\lambda-a_{r+1})p_r(\lambda)$ seja um polinômio de grau (r-1), pois pela recorrência devemos ter $q(\lambda) = -b_r{}^2 p_{r-1}(\lambda)$, onde $p_{r-1}(\lambda)$ por definição deverá ser um polinômio de grau (r-1). Assim para $p_{r-1}(\lambda)$ ficar completamente determinado basta dividir $q(\lambda)$ por $-b_r{}^2$. Temos então que, em cada etapa

encontra-se a_{r+1} e b_r.

Essa construção de $(p_r)_1^n$ através de (2.2.1), a partir de p_n e p_{n-1} resolve o problema proposto, numericamente, mas este processo é naturalmente instável, porque os polinômios p_{n-2}, p_{n-3}, ..., p_1 são encontrados cancelando-se sucessivamente os termos principais no par anterior de polinômios.

Provaremos a seguir, formalmente, a unicidade da construção de uma matriz de Jacobi a partir dos seus autovalores e dos autovalores de sua submatriz principal, de ordem n-1. Utilizaremos para tal, o método de construção realizado por Hald, pois baseia-se no fato de que os polinômios característicos de uma matriz de Jacobi satisfazem a fórmula de recorrência (2.2.1) e vice-versa.

TEOREMA 2.2.1. - Uma matriz de Jacobi é unicamente determinada através dos seus autovalores e dos autovalores de sua submatriz principal de ordem n-1.

Prova:

Suponhamos que existe duas matrizes de Jacobi, J_1 e J_2, tal que seus autovalores são λ_1, λ_2, ..., λ_n e os autovalores de suas submatrizes principais, de ordem n-1, são μ_1, μ_2, ..., μ_{n-1}. Sendo que esses dois conjuntos de autovalores satisfazem a desigualdade $\lambda_i < \mu_i < \lambda_{i+1}$. Mostraremos então que $J_1 = J_2$.

Por hipótese temos que

$$p_n(\lambda) = q_n(\lambda)$$
$$p_{n-1}(\lambda) = q_{n-1}(\lambda), \qquad (2.2.4)$$

onde $p_n(\lambda)$ e $q_n(\lambda)$ correspondem, respectivamente, aos polinômios característicos de J_1 e J_2 e, por sua vez, $p_{n-1}(\lambda)$ e $q_{n-1}(\lambda)$ os polinômios característicos das submatrizes principais, de ordem n-1, de J_1 e J_2.

Se tomarmos r = n-1, na recorrência (2.2.1), temos que

$$p_n(\lambda) = (\lambda-a_n)p_{n-1}(\lambda) - b_{n-1}^2 p_{n-2}(\lambda)$$
$$q_n(\lambda) = (\lambda-a'_n)q_{n-1}(\lambda) - b'^2_{n-1} q_{n-2}(\lambda), \qquad (2.2.5)$$

e por (2.2.4) obtemos que

$$(a'_n-a_n)p_{n-1}(\lambda) = b_{n-1}^2 p_{n-2}(\lambda) - b'^2_{n-1} q_{n-2}(\lambda). \qquad (2.2.6)$$

Mas $p_{n-2}(\lambda)$ e $q_{n-2}(\lambda)$ são polinômios mônicos de ordem n-2 e $p_{n-1}(\lambda)$ também é mônico, mas de ordem n-1, concluímos dessa forma que $a_n = a'_n$.

Portanto, de (2.2.6) temos

$$b_{n-1}^2 p_{n-2}(\lambda) = b'^2_{n-1} q_{n-2}(\lambda). \qquad (2.2.7)$$

Como $b_{n-1} \neq 0$, podemos escrevê-la como

$$p_{n-2}(\lambda) = \frac{b'^{2}_{n-1}}{b^{2}_{n-1}} q_{n-2}(\lambda), \qquad (2.2.8)$$

e novamente por $q_{n-2}(\lambda)$ e $p_{n-2}(\lambda)$ serem polinômios mônicos temos

$$\frac{b'^{2}_{n-1}}{b^{2}_{n-1}} = 1. \qquad (2.2.9)$$

Assim, $b^{2}_{n-1} = b'^{2}_{n-1}$. Como os elementos fora da diagonal principal de uma matriz de Jacobi são todos de mesmo sinal e, estamos considerando matrizes de Jacobi da forma (1.2.1) temos que $b_{n-1} = b'_{n-1}$.

Obtemos assim, que $p_{n-2}(\lambda) = q_{n-2}(\lambda)$.

Suponhamos agora que

$$\begin{aligned} p_k(\lambda) &= q_k(\lambda) \\ p_{k-1}(\lambda) &= q_{k-1}(\lambda), \quad \text{com } 1 \leq k \leq n-1. \end{aligned} \qquad (2.2.10)$$

Tomando $r = k-1$ em (2.2.1) obtemos

$$\begin{aligned} p_k(\lambda) &= (\lambda - a_k)p_{k-1}(\lambda) - b^{2}_{k-1}p_{k-2}(\lambda) \\ q_k(\lambda) &= (\lambda - a'_k)q_{k-1}(\lambda) - b'^{2}_{k-1}q_{k-2}(\lambda), \end{aligned} \qquad (2.2.11)$$

mas pela hipótese (2.2.10) temos a igualdade

$$(a'_k - a_k)p_{k-1}(\lambda) = b^{2}_{k-1}p_{k-2}(\lambda) - b'^{2}_{k-1}q_{k-2}(\lambda). \qquad (2.2.12)$$

Mas $p_{k-2}(\lambda)$ e $q_{k-2}(\lambda)$ são polinômios mônicos de ordem k-2 e $p_{k-1}(\lambda)$ também é mônico, mas de ordem k-1, assim $a_k = a'_k$.

Logo (2.2.11) torna-se

$$b_{k-1}^2 p_{k-2}(\lambda) = b'^2_{k-1} q_{k-2}. \qquad (2.2.13)$$

Como $b_{k-1} \neq 0$, $\forall$ k, temos

$$p_{k-2}(\lambda) = \frac{b'^2_{k-1}}{b_{k-1}^2} q_{k-2}(\lambda) \qquad (2.2.14)$$

e sendo $q_{k-2}(\lambda)$ e $p_{k-2}(\lambda)$ polinômios mônicos

$$\frac{b'^2_{k-1}}{b_{k-1}^2} = 1. \qquad (2.2.15)$$

Obtemos assim, que $b_{k-1} = b'_{k-1}$. E consequentemente temos $p_{k-2}(\lambda) = q_{k-2}(\lambda)$, para k arbitrário. Provamos dessa forma que $J_1 = J_2$. ■

2.3 - RESOLUÇÃO DO PROBLEMA POR DE BOOR E GOLUB

De Boor e Golub [4] geraram através da recorrência (2.2.1) uma sequência de polinômios mônicos ortogonais de forma que os dois últimos elementos da sequência sejam, respectivamente, os polinômios

característicos da matriz a ser reconstruída e de sua submatriz principal de ordem n-1, isto é,

$$p_n(\lambda) = \prod_{i=1}^{n} (\lambda-\lambda_i) \quad e \quad p_{n-1}(\lambda) = \prod_{i=1}^{n-1} (\lambda-\mu_i). \qquad (2.3.1)$$

Eles usaram para gerar essa sequência a construção desenvolvida por Forsythe [5] que foi apresentada no capítulo 1, na seção de polinômios ortogonais. Mas para fazê-la é necessário um produto interno com peso adequado. Como $p_n(\lambda)$ e $p_{n-1}(\lambda)$ são conhecidos obteremos um peso conveniente a partir deles. Será usada a mesma notação do capítulo 1.

Observemos que se $f(x)$ é um polinômio qualquer em P_{n-1}, isto é, de grau n-2, ou menor, então

$$\langle p_{n-1}, f \rangle \equiv \sum_{i=1}^{n} p_{n-1}(\lambda_i) f(\lambda_i) \omega_i = 0. \qquad (2.3.2)$$

Mas se a combinação

$$\sum_{i=1}^{n} m_i f(\lambda_i), \qquad (2.3.3)$$

com $m_i = \omega_i p_{n-1}(\lambda_i)$ é zero para qualquer $f(x)$ em P_{n-1}, então

$$\sum_{i=1}^{n} m_i \lambda_i^{\,k} = 0; \quad k=0,1,\ldots,n-2 \ , \qquad (2.3.4)$$

desde que cada x^k está em P_{n-1}. Assim temos que

$$Bm = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 & \dots & \lambda_n \\ (\lambda_1)^2 & (\lambda_2)^2 & (\lambda_3)^2 & \dots & (\lambda_n)^2 \\ \cdot & \cdot & \cdot & & \cdot \\ (\lambda_1)^{n-2} & (\lambda_2)^{n-2} & (\lambda_3)^{n-2} & \dots & (\lambda_n)^{n-2} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ .. \\ m_n \end{bmatrix} = 0. \qquad (2.3.5)$$

Para resolver essa equação matricial, devemos escrevê-la da seguinte forma:

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ \lambda_1 & \lambda_2 & \lambda_3 & \dots & \lambda_{n-1} \\ (\lambda_1)^2 & (\lambda_2)^2 & (\lambda_3)^2 & \dots & (\lambda_{n-1})^2 \\ \cdot & \cdot & \cdot & & \cdot \\ (\lambda_1)^{n-2} & (\lambda_2)^{n-2} & (\lambda_3)^{n-2} & \dots & (\lambda_{n-1})^{n-2} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ \dots \\ m_{n-1} \end{bmatrix} =$$

$$= -m_n \begin{bmatrix} 1 \\ \lambda_n \\ (\lambda_n)^2 \\ \cdot \\ (\lambda_n)^{n-2} \end{bmatrix} . \qquad (2.3.6)$$

Então por Cramer temos que

$$m_i = \frac{\Delta_i}{\Delta}(-m_n) \qquad i=1,2,\ldots,n-1, \tag{2.3.7}$$

onde Δ e Δ_i são determinantes de matrizes de Vandermonde[2], sendo assim

$$\Delta = \frac{\Gamma}{\prod_{j=1}^{n-1}(\lambda_n-\lambda_j)}, \quad \Delta_i = \frac{\Gamma}{\prod_{\substack{j=1\\ j\neq i}}^{n}(\lambda_i-\lambda_j)}, \quad i=1,2,\ldots,n-1 \tag{2.3.8}$$

com

$$\Gamma = \prod_{j=2}^{n}\prod_{k=1}^{j-1}(\lambda_j-\lambda_k). \tag{2.3.9}$$

Logo

$$m_i = \prod_{j=1}^{n-1}(\lambda_n-\lambda_j)m_n \frac{1}{\prod_{\substack{j=1\\ j\neq i}}^{n}(\lambda_i-\lambda_j)}, \quad i=1,2,\ldots,n-1. \tag{2.3.10}$$

Denotemos $\gamma = \prod_{j=1}^{n-1}(\lambda_n-\lambda_j)m_n$, temos como solução de (2.3.5),

---

(2) - Veja matrizes de Vandermonde em [6].

$$m_i = \frac{\gamma}{\prod_{j=1}^{n}{}' (\lambda_i - \lambda_j)} , \quad i=1,2,\ldots,n. \qquad (2.3.11)$$

A menos da arbitrariedade do fator γ, essa solução é única. O apóstrofe em (2.3.11) significa que o termo j=i foi omitido.

Como $p_n(\lambda) = \prod_{j=1}^{n} (\lambda-\lambda_j)$, temos

$$p_n'(\lambda_i) = \prod_{j=1}^{n}{}' (\lambda_i - \lambda_j), \qquad (2.3.12)$$

onde o apóstrofe do lado esquerdo denota diferenciação. Retornando a equação (2.3.11), temos que para γ adequado,

$$\omega_i p_{n-1}(\lambda_i) = \frac{\gamma}{p_n'(\lambda_i)} , \quad i=1,2,\ldots,n ; \qquad (2.3.13)$$

ou seja,

$$\omega_i = \frac{\gamma}{p_{n-1}(\lambda_i) p_n'(\lambda_i)} , \quad i=1,2,\ldots,n. \qquad (2.3.14)$$

Como a sequência de polinômios $(p_i)_0^n$ satisfaz a relação de recorrência (2.2.1), segue pelos argumentos usados na seção 1.3 que os zeros de $p_n(\lambda)$ e $p_{n-1}(\lambda)$ devem entrelaçarem-se e dessa forma $(-)^{n-i} p_{n-1}(\lambda_i) > 0$. E como $(\lambda_i)_1^n$ estão em ordem crescente $(-)^{n-i} p_n'(\lambda_i) > 0$.

Portanto, $p_{n-1}(\lambda_i)p_n'(\lambda_i) > 0$, significando que os pesos dados por (2.3.14) são positivos, para γ escolhido adequadamente.

De Boor e Golub construíram dessa forma uma sequência de polinômios mônicos ortogonais, onde estes correspondem aos polinômios característicos de uma matriz de Jacobi e de suas submatrizes principais.

Os polinômios são gerados na ordem natural usando a fórmula de recorrência,

$$p_{-1}(\lambda) \equiv 0,\ p_0(\lambda) = 1$$
$$p_r(\lambda) = (\lambda - a_r)p_{r-1}(\lambda) - b_{r-1}^2 p_{r-2}(\lambda),\ r=1,2,\ldots,n \qquad (2.3.15)$$

com os números a_r, b_r calculados por

$$a_r = \frac{\langle p_{r-1}, \lambda p_{r-1}\rangle}{\|p_{r-1}\|^2}, \qquad r=1,2,\ldots,n \qquad (2.3.16)$$

e

$$b_r = \frac{\|p_r\|}{\|p_{r-1}\|}, \qquad r=1,2,\ldots,n-1. \qquad (2.3.17)$$

Esse processo é numericamente estável.

Desde que a e b determinados por (2.3.15) e (2.3.16)-(2.3.17) dependem continuamente sobre λ e μ, segue que a matriz A depende continuamente sobre λ e μ.

A maior dificuldade encontrada por De Boor e Golub está no cálculo dos pesos ω_i.

Procurando superar essa dificuldade eles usaram a reflexão de A sobre a sua segunda diagonal.

Seja S a matriz permutação levando $(1,2,\ldots,n)$ em $(n,n-1,\ldots,1)$, isto é,

$$S = (\delta_{i+j,n+1})_{i,j=1}^{n} . \qquad (2.3.18)$$

Como S é ortogonal e simétrica então $S^2 = I$. Ela transforma A em

$$\bar{A} = SAS = \begin{bmatrix} a_n & -b_{n-1} & 0 & \cdots & 0 \\ -b_{n-1} & a_{n-1} & -b_{n-2} & & 0 \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & & & a_2 & -b_1 \\ 0 & \cdots & & -b_1 & a_1 \end{bmatrix} . \qquad (2.3.19)$$

Denotemos os elementos de $\bar{A}$ por $\bar{a}_r$ e $\bar{b}_r$, então

$$\begin{aligned} \bar{a}_r &= a_{n+1-r} , \\ \bar{b}_r &= b_{n-r} . \end{aligned} \qquad (2.3.20)$$

A matriz A é dita persimétrica se $A = \bar{A}$, isto é, $a_r = a_{n+1-r}$ e $b_r = b_{n-r}$.

O menor principal de ordem n-1 de $(\lambda I - A)$ é denotado por $p_{n-1}(\lambda)$, assim o correspondente menor de $\lambda I - \bar{A}$ é

dado por $\bar{p}_{n-1}(\lambda)$. Provemos o seguinte lema.

LEMA 2.3.1 - Considerando a notação acima temos para $i=1,2,\ldots,n$ que

$$p_{n-1}(\lambda_i)\bar{p}_{n-1}(\lambda_i) = (b_1 b_2 \ldots b_{n-1})^2 = b^2. \qquad (2.3.21)$$

Prova:

Observemos que $p_{n-1}(\lambda_i)\bar{p}_{n-1}(\lambda_i)$ é o produto do menor principal esquerdo de ordem n-1 com o menor principal direito de ordem n-1 da matriz singular $C = \lambda_i I - A$.

Utilizaremos a seguinte notação

$$C\begin{bmatrix} i_1, & \ldots, & i_r \\ j_1, & \ldots, & j_s \end{bmatrix} = \left[c_{i_\rho j_\sigma}\right]_{\rho=1}^{r} \quad {}_{\sigma=1}^{s}.$$

Assim, $p_{n-1}(\lambda_i) = \left|C\begin{bmatrix} 1,2,\ldots,n-1 \\ 1,2,\ldots,n-1 \end{bmatrix}\right|$ e

$$\bar{p}_{n-1}(\lambda_i) = \left|C\begin{bmatrix} 2,3,\ldots,n \\ 2,3,\ldots,n \end{bmatrix}\right|.$$

Usando a identidade de Sylvester[9], com $C\begin{bmatrix} 2,3,\ldots,n-1 \\ 2,3,\ldots,n-1 \end{bmatrix}$ como bloco pivotal, obtemos

---

(3) - Conforme [7].

$$0 = \left|C\begin{bmatrix}2,3,\ldots,n-1\\2,3,\ldots,n-1\end{bmatrix}\right||C| = \begin{vmatrix}\left|C\begin{bmatrix}1,2,\ldots,n-1\\1,2,\ldots,n-1\end{bmatrix}\right| & \left|C\begin{bmatrix}1,2,\ldots,n-1\\2,3,\ldots,n\end{bmatrix}\right| \\ \left|C\begin{bmatrix}2,3,\ldots,n\\1,2,\ldots,n-1\end{bmatrix}\right| & \left|C\begin{bmatrix}2,3,\ldots,n\\2,3,\ldots,n\end{bmatrix}\right|\end{vmatrix}$$

onde os elementos da diagonal principal da matriz são $p_{n-1}(\lambda_i)$ e $\bar{p}_{n-1}(\lambda_i)$ e, os elementos da segunda diagonal são

$$\left|C\begin{bmatrix}1,2,\ldots,n-1\\1,2,\ldots,n-1\end{bmatrix}\right| = \begin{vmatrix} b_1 & 0 & \cdots & 0 \\ \lambda_i - a_2 & b_2 & & 0 \\ b_2 & \lambda_i - a_3 & \ddots & \vdots \\ \vdots & & b_{n-2} & 0 \\ 0 & \cdots & \lambda_i - a_{n-1} & b_{n-1} \end{vmatrix} =$$

$$= b_1 b_2 \ldots b_{n-1};$$

$$\left|C\begin{bmatrix}2,3,\ldots,n\\1,2,\ldots,n-1\end{bmatrix}\right| = \begin{vmatrix} b_1 & \lambda_i - a_2 & b_2 & & 0 \\ 0 & b_2 & \lambda_i - a_3 & & 0 \\ \vdots & & \ddots & & \vdots \\ & & & b_{n-2} & \lambda_i - a_{n-1} \\ 0 & \cdots & 0 & & b_{n-1} \end{vmatrix} =$$

$$= b_1 b_2 \ldots b_{n-1}.$$

Logo $p_{n-1}(\lambda_i)\bar{p}_{n-1}(\lambda_i) = (b_1 b_2 \ldots b_{n-1})^2 = b^2$. ■

Este resultado permiti-nos concluir que os polinómios $\bar{p}_n(\lambda)$ $(=p_n(\lambda))$, $\bar{p}_{n-1}(\lambda)$, ..., $\bar{p}_1(\lambda)$, 1 são os polinómios obtidos através de (2.3.15), com pesos dados por

$$\omega_i = \frac{b^2}{\bar{p}_{n-1}(\lambda_i)\bar{p}_n'(\lambda_i)} = \frac{p_{n-1}(\lambda_i)}{p_n'(\lambda_i)} . \qquad (2.3.22)$$

Esses pesos podem ser mais facilmente construídos do que os pesos dados por (2.3.14).

Nesse processo os termos da matriz A são calculados na ordem $\bar{a}_1$, $\bar{b}_1$, ..., $\bar{b}_{n-1}$, $\bar{a}_n$, isto é, a_n, b_{n-1}, ..., b_1, a_1.

O uso desses pesos tem mais vantagens computacionais do que os outros dados por (2.3.14), fato comprovado por De Boor e Golub, pois por causa da condição de entrelaçamento (2.1.1), temos que

$$\frac{\lambda_i - \mu_{i-1}}{\lambda_i - \lambda_1} \frac{\mu_i - \lambda_i}{\lambda_n - \lambda_i} < \frac{p_{n-1}(\lambda_i)}{p_n'(\lambda_i)} < 1 , \qquad (2.3.23)$$

onde o primeiro (último) fator do lado esquerdo da primeira desigualdade é omitido quando i = 1 (i=n); assim os valores dos pesos ω_i, $\forall$ i, estão limitados inferiormente e superiormente. Isso mostra que *overflow* ou *underflow*, é altamente improvável de ocorrer no cálculo dos pesos (2.3.22). Ao contrário, o cálculo dos pesos (2.3.14) tem que ser cuidadosamente monitorado, em geral, pela ocorrência de overflow ou underflow; senão têm-se que calcular os logaritmos desses pesos, um procedimento bem mais trabalhoso.

Demonstraremos a seguir que no caso particular

em que a matriz de Jacobi é persimétrica é suficiente conhecer a sequência de autovalores $\lambda = (\lambda_i)_1^n$, para a reconstrução da mesma.

TEOREMA 2.3.1 - Existe uma única matriz de Jacobi persimétrica A com os autovalores $(\lambda_i)_1^n$ dados, satisfazendo $\lambda_1 < \lambda_2 < \ldots < \lambda_n$.

Prova:

Como os autovalores são conhecidos é possível encontrar os pesos para a construção dos polinômios ortogonais $p_r(\lambda)$. De fato, se A é persimétrica então o menor principal $p_r(\lambda)$ é igual a $\bar{p}_r(\lambda)$ e, do lema 2.3.1 temos que $(p_{n-1}(\lambda_i))^2 = b^2$, isto é, $p_{n-1}(\lambda_i) = \pm b$, de modo que a equação (2.3.22) produz

$$\omega_i = \frac{\pm b}{p_n'(\lambda_i)} . \qquad (2.3.24)$$

Desde que os sinais de $p_n'(\lambda_i)$ alternarão deve-se considerar os sinais de $\pm b$ de modo que ω_i seja positivo. A magnitude de b é irrelevante para a construção dos $p_r(\lambda)$. ■

CAPITULO 3

RECONSTRUÇÃO DE UMA MATRIZ DE JACOBI UTILIZANDO-SE O ALGORITMO DE LANCZOS

3.1 INTRODUÇÃO

Assim, como no capítulo 2 desejamos reconstruir uma matriz de Jacobi da forma (1.2.1).

O método que abordaremos foi desenvolvido por Boley e Golub [9] para resolver problemas inversos de autovalores associados a matrizes banda, ou seja, matrizes reais, simétricas e com $a_{ij} = 0$ quando $|i-j| \geq p$, onde p especifica a largura da banda. No nosso caso p = 2, pois a matriz de Jacobi é uma matriz tridiagonal. Mas teremos que garantir que os elementos fora da diagonal principal sejam negativos, já que em uma matriz banda isso não está caracterizado.

Para realizar tal reconstrução, suponhamos que os autovalores $(\lambda_i)_1^n$ de A sejam conhecidos, com $\lambda_1 < \lambda_2 < \ldots < \lambda_n$ e dos autovetores normalizados $u^{(i)}$, conhecemos os valores $(u_1^{(i)})_1^n$ (ou $(u_n^{(i)})_1^n$), isto é, a

sua primeira (ou última) componente.

Nas seções 3.2 e 3.3 analisaremos os valores estacionários de uma forma quadrática sujeita a restrições lineares e sua aplicação a problemas inversos de autovalores, cujo trabalho foi desenvolvido por Golub [10]. Nosso interesse em tal assunto está em mostrar como a primeira (ou última) componente dos autovetores pode ser obtida quando se conhece o conjunto de autovalores da submatriz principal direita (ou esquerda) de A, de ordem n-1, sendo que na reconstrução da matriz consideraremos a última linha da matriz dos autovetores, a qual será obtida explicitamente na seção 3.4.

A reconstrução é descrita na seção 3.5, onde utilizaremos o algoritmo de Lanczos, cfe. [12], [18] e [25], que tem a vantagem de ser numericamente bem condicionado. Juntamente a esta mostraremos que os vetores construídos a partir do algoritmo, chamados vetores de Lanczos, são realmente ortonormais.

3.2 - VALORES ESTACIONÁRIOS DE UMA FORMA QUADRÁTICA SUJEITA A RESTRIÇÕES LINEARES

Seja A uma matriz real, simétrica, de ordem n, e c um vetor dado, com $c^Tc = 1$. Determinaremos os valores estacionários de

$$x^T Ax, \qquad (3.2.1)$$

sujeita a restrições lineares

$$\begin{aligned} x^T x &= 1, \\ c^T x &= 0. \end{aligned} \qquad (3.2.2)$$

Consideremos, dessa forma, a função φ dada por

$$\varphi(x,\lambda,\mu) = x^T Ax - \lambda(x^T x-1) + 2\mu x^T c, \qquad (3.2.3)$$

onde λ e μ são as constantes de Lagrange. Diferenciando (3.2.3) obtemos a equação

$$Ax - \lambda x + \mu c = 0 . \qquad (3.2.4)$$

Multiplicando (3.2.4) por c^T e usando a condição $\|c\|_2 = 1$, temos

$$\mu = - c^T Ax. \qquad (3.2.5)$$

Substituindo-a em (3.2.4) ficamos com

$$PAx = \lambda x, \qquad (3.2.6)$$

onde $P = I - cc^T$.

Observemos que o parâmetro λ é valor um

estacionário de x^TAx, pois de (3.2.6) temos que $Ax - cc^TAx = \lambda x$. Pré-multiplicando-a por x^T obtemos

$$x^TAx = \lambda x^Tx + x^Tcc^TAx. \qquad (3.2.7)$$

Logo, pelas restrições lineares (3.2.2) temos que

$$x^TAx = \lambda. \qquad (3.2.8)$$

Embora P e A sejam simétricas, PA pode não ser. Mas observemos que $P^2 = P$, de modo que P é uma matriz projeção. Sabemos (cfe. [23]) que para duas matrizes quadradas arbitrárias F e G os autovalores de FG são iguais aos autovalores de GF. Assim,

$$\lambda(PA) = \lambda(P^2A) = \lambda(PAP)^{(1)}. \qquad (3.2.9)$$

A matriz PAP é simétrica e, portanto, podemos usar um dos algoritmos usuais para encontrar seus autovalores.

Consideremos $K = PAP$. Como K é simétrica e real existem n autovetores associados aos seus autovalores, então

$$Kz_i = \lambda_i z_i, \qquad (3.2.10)$$

---

(1) $\lambda(X)$ denota o conjunto dos autovalores da matriz X.

e segue que

$$x_i = Pz_i \ , \ i=1,2,\ldots,n \ , \qquad (3.2.11)$$

onde x_i é o autovetor que satisfaz (3.2.6).

Como $P = I - cc^T$ e $\|c\|_2 = 1$, temos $Pc = 0$, ou seja, c é um autovetor associado a um autovalor nulo de P. Mas $(PAP)c = 0$, logo existe pelo menos um autovalor nulo de K.

Sendo P uma matriz simétrica real, existe uma matriz Q tal que $Q^TQ = I_n$, de forma que

$$P = Q^TJQ, \qquad (3.2.12)$$

onde J é uma matriz diagonal com elementos da diagonal específicos. Isso ocorre porque P é uma matriz projeção, sendo

$$J = \begin{bmatrix} I_{n-1} & 0 \\ 0 & 0 \end{bmatrix}. \qquad (3.2.13)$$

Assim,

$$\lambda(PAP) = \lambda(Q^TJQAQ^TJQ) = \lambda(JQAQ^TJ). \qquad (3.2.14)$$

Considerando

$$G = QAQ^T = \begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix}, \qquad (3.2.15)$$

onde G_{11} é uma matriz de ordem n-1 e G_{22} um escalar, temos

$$JQAQ^TJ = JGJ = \begin{bmatrix} G_{11} & 0 \\ 0 & 0 \end{bmatrix}. \qquad (3.2.16)$$

Portanto os valores estacionários de x^TAx sujeita a retrições lineares (3.2.2) são simplesmente os autovalores da matriz G_{11} de ordem n-1. Finalmente, se

$$G_{11}v_i = \lambda_i v_i, \quad i=1,2,\ldots,n-1, \qquad (3.2.17)$$

então

$$x_i = Q^T \begin{bmatrix} I_{n-1} \\ 0 \end{bmatrix} v_i, \qquad (3.2.18)$$

onde x_i representa o autovetor que satisfaz (3.2.6).

Mais detalhes do algoritmo é dado em [11].

Por (3.2.15) temos que $\lambda(G) = \lambda(A)$. Então pelo teorema de Courant - Fischer, cfe. [24], obtemos

$$\lambda_j(A) \leq \lambda_j(G_{11}) \leq \lambda_{j+1}(A), \quad j=1,2,\ldots,n-1, \qquad (3.2.19)$$

quando $\lambda_j(A) \leq \lambda_{j+1}(A)$ e $\lambda_j(G_{11}) \leq \lambda_{j+1}(G_{11})$.

Vemos assim que existe uma forte relação entre os autovalores de A e os valores estacionários da função

$$\varphi(x,y,\mu) = x^TAx - \lambda(x^Tx - 1) + 2\mu c^Tx. \qquad (3.2.20)$$

3.3 - APLICAÇÃO A PROBLEMAS INVERSOS DE AUTOVALOR

Dada uma matriz simétrica A com autovalores $(\lambda_i)_1^n$ $(\lambda_i < \lambda_{i+1})$ e um conjunto de valores $(\nu_i)_1^{n-1}$ $(\nu_i < \nu_{i+1})$ com

$$\lambda_i < \nu_i < \lambda_{i+1}; \qquad (3.3.1)$$

desejamos determinar a restrição linear $c^Tx = 0$ de modo que os valores estacionários de x^TAx sujeita a $x^Tx = 1$ e $c^Tx = 0$ (com $c^Tc = 1$) seja o conjunto $(\nu_i)_1^{n-1}$.

De (3.2.4) temos $x = -\mu(A-\lambda I)^{-1}c$ e, portanto,

$$c^Tx = -\mu c^T(A-\lambda I)^{-1}c = 0. \qquad (3.3.2)$$

Consideremos $\mu \neq 0$ e seja

$$A = U\Lambda U^T, \qquad (3.3.3)$$

onde Λ é a matriz diagonal de autovalores de A e U é a matriz de autovetores ortonormalizados.

Substituindo (3.3.3) em (3.3.2) obtemos

$$c^T U(\Lambda-\lambda I)^{-1} U^T c = 0, \qquad (3.3.4)$$

onde

$$(\Lambda - \lambda I)^{-1} = \mathrm{diag}\left[\frac{1}{\lambda_1-\lambda}, \frac{1}{\lambda_2-\lambda}, \ldots, \frac{1}{\lambda_n-\lambda}\right]. \qquad (3.3.5)$$

Seja

$$Ud = c, \qquad (3.3.6)$$

que aplicada em (3.3.4) resulta em

$$d^T\left[\mathrm{diag}\left[\frac{1}{\lambda_1-\lambda}, \frac{1}{\lambda_2-\lambda}, \ldots, \frac{1}{\lambda_n-\lambda}\right]\right]d = 0, \text{ ou seja,}$$

$$\sum_{i=1}^{n} \frac{d_i^2}{(\lambda_i-\lambda)} = 0. \qquad (3.3.7)$$

Além disso, de (3.3.6) obtemos que

$$\sum_{i=1}^{n} d_i^2 = 1. \qquad (3.3.8)$$

Fazendo $\lambda = \nu_j$, $j=1,2,\ldots,n-1$, recaímos num sistema de n equações lineares da forma

$$\sum_{i=1}^{n} \frac{d_i^2}{(\lambda_i - \nu_j)} = 0, \text{ com } j=1,2,\ldots,n-1$$

$$\sum_{i=1}^{n} d_i^2 = 1. \qquad (3.3.9)$$

Com isso daremos uma solução explícita para este sistema.

Seja o polinômio característico

$$\varphi(\lambda) = \prod_{j=1}^{n-1} (\nu_j - \lambda). \qquad (3.3.10)$$

Coloquemos (3.3.7) na forma polinomial

$$\psi(\lambda) \equiv \prod_{j=1}^{n} (\lambda_j - \lambda) \left[\sum_{i=1}^{n} \frac{d_i^2}{(\lambda_i - \lambda)} \right] = \sum_{i=1}^{n} d_i^2 \prod_{\substack{j=1 \\ j \neq i}}^{n} (\lambda_j - \lambda). \qquad (3.3.11)$$

Desejamos calcular d ($d^T d = 1$) de modo que $\psi(\lambda) \equiv \varphi(\lambda)$, e então igualamos os dois polinômios em n pontos, dessa forma,

$$\varphi(\lambda_k) = \prod_{j=1}^{n-1} (\nu_j - \lambda_k) \text{ e } \psi(\lambda_k) = d_k^2 \prod_{\substack{j=1 \\ j \neq i}}^{n} (\lambda_j - \lambda_k). \qquad (3.3.12)$$

Portanto $\psi(\lambda_k) = \varphi(\lambda_k)$, para $k=1,2,\ldots,n$, se

$$d_k^2 = \frac{\prod_{j=1}^{n-1}(\nu_j - \lambda_k)}{\prod_{\substack{j=1 \\ j\neq i}}^{n}(\lambda_j - \lambda_k)} . \qquad (3.3.13)$$

A condição (3.3.1) garante que o lado direito de (3.3.13) será positivo. Sendo assim, observemos que se pode considerar d_k positivo ou negativo, de modo que existem 2^n diferentes soluções. Uma vez calculado o vetor d é fácil calcular c através de (3.3.6).

3.4 - OBTENÇÃO DA ÚLTIMA LINHA DA MATRIZ DOS AUTOVETORES ASSOCIADOS A UMA MATRIZ DE JACOBI

A condição de entrelaçamento dada por (3.3.1) é satisfeita para os conjuntos de autovalores $\lambda = (\lambda_i)_1^n$ e $\mu = (\mu_i)_1^{n-1}$ que representam, respectivamente, autovalores de A e de A_{n-1}, pois a matriz A, a ser construída, deve ser de Jacobi.

De acordo com as seções anteriores basta tomar $c = e_n$, onde $e_n^T = (0,\ldots,0,1)$ e $x_n = 0$, para que os valores estacionários de x^TAx, sujeita a restrições lineares $x^Tx=1$ e $e_n^Tx = 0$, sejam os autovalores de A_{n-1}.

Dessa forma, de (3.3.6), temos que $d = U^Te_n$,

ou seja,

$$d_i = u_n^{(i)} \text{ para } i=1,2,\ldots,n , \qquad (3.4.1)$$

representa a última componente dos autovetores $u^{(i)}$.

Desde que a matriz A é de Jacobi, $u_n^{(i)} \neq 0$ para todo $i=1,2,\ldots,n$, conforme proposição 1.4.1 e de acordo com (3.3.13) temos

$$(u_n^{(i)})^2 = \frac{\prod_{j=1}^{n-1} (\mu_j - \lambda_i)}{\prod_{j=1}^{n}{}' (\lambda_j - \lambda_i)} , \text{ com } i=1,2,\ldots,n. \qquad (3.4.2)$$

O apóstrofe no lado direito da equação (3.4.2) significa que o termo $j = i$ foi omitido e, ainda, sem perda de generalidade escolheremos $u_n^{(i)}$ positivo, para todo i.

3.5 - RECONSTRUÇÃO DA MATRIZ DE JACOBI ATRAVÉS DO ALGORITMO DE LANCZOS

Os autovetores $u^{(i)}$ satisfazem

$$Au^{(i)} = \lambda_i u^{(i)} , \quad i=1,2,\ldots,n. \qquad (3.5.1)$$

Daí consideremos $U = [u^{(1)}, u^{(2)}, \ldots, u^{(n)}]$.

Então a condição de ortogonalidade $u^{(i)T}u^{(j)} = \delta_{ij}$ produz $U^TU = I$. Porém $UU^T = I$, de modo que

$$\sum_{i=1}^{n} u_r^{(i)} u_s^{(i)} = \delta_{rs}. \qquad (3.5.2)$$

Escrevendo, $x^{(r)T} = [u_r^{(1)} u_r^{(2)} \ldots u_r^{(n)}]$, teremos

$$x^{(r)T}x^{(s)} = \delta_{rs}. \qquad (3.5.3)$$

O conjunto de equações (3.5.1) pode ser escrito como $AU = U\Lambda$, ou equivalentemente,

$$XA = \Lambda X, \qquad (3.5.4)$$

onde $X = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}] = U^T$.

Suponhamos que o vetor $x^{(n)T} = [u_n^{(1)} u_n^{(2)} \ldots u_n^{(n)}]$ seja dado. Portanto, de (3.5.4) temos

$$[x^{(1)}\ x^{(2)} \ldots\ x^{(n)}] \begin{bmatrix} a_1 & -b_1 & 0 & \cdots & 0 \\ -b_1 & a_2 & -b_2 & & 0 \\ & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & a_{n-2} & -b_{n-2} \\ & & & -b_{n-2} & a_{n-1} \\ 0 & \cdots & 0 & -b_{n-1} & a_n \end{bmatrix} =$$

$$= \Lambda[x^{(1)}\ x^{(2)}\ \ldots\ x^{(n)}]. \qquad (3.5.5)$$

A última coluna desta equação é

$$-b_{n-1}x^{(n-1)} + a_n x^{(n)} = \Lambda x^{(n)}, \qquad (3.5.6)$$

de modo que se $x^{(n)T}x^{(n-1)}$ é zero, então

$$a_n = x^{(n)T}\Lambda x^{(n)}. \qquad (3.5.7)$$

Fazendo

$$r^{(n-1)} = a_n x^{(n)} - \Lambda x^{(n)} \qquad (3.5.8)$$

e se $\|x^{(n-1)}\|_2 = 1$, b_{n-1} deve ser escolhido de tal forma que

$$b_{n-1} = \|r^{(n-1)}\|_2. \qquad (3.5.9)$$

Logo,

$$x^{(n-1)} = \frac{r^{(n-1)}}{b_{n-1}}. \qquad (3.5.10)$$

Tendo encontrado a_n e b_{n-1}, consideremos a (n-1)-ésima coluna da equação (3.5.5),

$$-b_{n-2}x^{(n-2)} + a_{n-1}x^{(n-1)} - b_{n-1}x^{(n)} = \Lambda x^{(n-1)}, \qquad (3.5.11)$$

de modo que se $x^{(n-1)T}x^{(n-2)}$ seja zero, então

$$a_{n-1} = x^{(n-1)T} A x^{(n-1)}, \qquad (3.5.12)$$

onde foi usado que $x^{(n)T} x^{(n-1)} = 0$.

Façamos

$$r^{(n-2)} = a_{n-1} x^{(n-1)} - A x^{(n-1)} - b_{n-1} x^{(n)}, \qquad (3.5.13)$$

b_{n-2} deve ser escolhido de modo que

$$b_{n-2} = \| r^{(n-2)} \|_2. \qquad (3.5.14)$$

Logo,

$$x^{(n-2)} = \frac{r^{(n-2)}}{b_{n-2}}. \qquad (3.5.15)$$

Consideremos a i-ésima etapa, ou seja, (n-i+1)-ésima coluna da equação (3.5.5),

$$-b_{n-i} x^{(n-i)} + a_{n-i+1} x^{(n-i+1)} - b_{n-i+1} x^{(n-i+2)} = A x^{(n-i+1)}. \qquad (3.5.16)$$

Na etapa anterior $x^{(n-i+1)}$ foi encontrado de modo que $x^{(n-i+1)T} x^{(n-i+2)} = 0$. Assim, a_{n-i+1} deve ser escolhido de forma que

$$a_{n-i+1} = x^{(n-i+1)T} A x^{(n-i+1)}. \qquad (3.5.17)$$

Então,

$$r^{(n-i)} = a_{n-i+1}x^{(n-i+1)} - Ax^{(n-i+1)} - b_{n-i+1}x^{(n-i+2)}, \qquad (3.5.18)$$

e b_{n-i} deve ser da forma

$$b_{n-i} = \|r^{(n-i)}\|_2. \qquad (3.5.19)$$

Portanto,

$$x^{(n-i)} = \frac{r^{(n-i)}}{b_{n-i}}, \text{ com } i=3,4,\ldots,n-1. \qquad (3.5.20)$$

No próximo teorema mostraremos que os vetores $x^{(i)}$ construídos a partir do algoritmo de Lanczos satisfazem $x^{(i)T}x^{(j)} = \delta_{ij}$, para $i,j=1,2,\ldots,n$; embora esta ortogonalidade seja aparentemente estabelecida somente para $|i-j| \leq 1$.

TEOREMA 3.5.1 - Dado um conjunto de vetores $(x^{(i)})_1^n$ construídos a partir do algoritmo de Lanczos, do qual consideremos que $x^{(-1)} = x^{(0)} = 0$, temos que esses vetores satisfazem

$$x^{(i)T}x^{(j)} = \delta_{ij} \qquad i,j=1,2,\ldots,n. \qquad (3.5.21)$$

Prova:

A equação (3.5.21), pelo algoritmo de Lanczos, já está explicitamente estabelecida para $|i-j| \leq 1$.

Verificaremos a equação (3.5.21), para $|i-j|=2$ por indução.

A segunda coluna da equação (3.5.5) é dada por

$$-b_1 x^{(1)} + a_2 x^{(2)} - b_2 x^{(3)} = \Lambda x^{(2)}, \qquad (3.5.22)$$

a qual após a multiplicação por $x^{(1)T}$, pela esquerda, produz

$$-b_2 x^{(1)T} x^{(3)} = x^{(1)T} \Lambda x^{(2)} + b_1, \qquad (3.5.23)$$

onde usamos a ortonormalidade dos vetores para $|i-j| \leq 1$. Como $b_2 \neq 0$ temos

$$x^{(1)T} x^{(3)} = -\frac{1}{b_2} (x^{(1)T} \Lambda x^{(2)} + b_1). \qquad (3.5.24)$$

Mas a primeira coluna de (3.5.5) é

$$a_1 x^{(1)} - b_1 x^{(2)} = \Lambda x^{(1)}, \qquad (3.5.25)$$

que multiplicada por $x^{(2)T}$ nos dá

$$-b_1 = x^{(2)T} \Lambda x^{(1)}, \qquad (3.5.26)$$

onde usamos novamente a ortonormalidade dos vetores para

$|i-j| \leq 1$.

Assim $x^{(1)T}x^{(3)} = 0$, conforme desejado.

Suponhamos que

$$x^{(i-2)T}x^{(i)} = 0 \qquad \text{para } i=3,4,\ldots,k-1. \tag{3.5.27}$$

Tomando a (k-1)-ésima coluna de (3.5.5), ou seja,

$$-b_{k-2}x^{(k-2)} + a_{k-1}x^{(k-1)} - b_{k-1}x^{(k)} = \Lambda x^{(k-1)} \tag{3.5.28}$$

e multiplicando-a por $x^{(k-2)T}$, obtemos

$$-b_{k-1}x^{(k-2)T}x^{(k)} = x^{(k-2)}\Lambda x^{(k-1)} + b_{k-2}\,. \tag{3.5.29}$$

Agora, multiplicando a (k-2)-ésima coluna de (3.5.5) por $x^{(k-1)T}$ produz

$$-b_{k-3}x^{(k-1)T}x^{(k-3)} + a_{k-2}x^{(k-1)T}x^{(k-2)} - b_{k-2}x^{(k-1)T}x^{(k-1)} = x^{(k-1)T}\Lambda x^{(k-2)}\,. \tag{3.5.30}$$

Entretanto, o primeiro termo do lado esquerdo é zero, pela hipótese de indução. Assim temos que

$$-b_{k-3}x^{(k-1)T}x^{(k-3)} - b_{k-2} = x^{(k-1)T}\Lambda x^{(k-2)}\,. \tag{3.5.31}$$

Como o primeiro termo do lado esquerdo é zero o resultado indutivo segue em (5.3.29), pois $b_{k-1} \neq 0$.

Usaremos novamente a indução para provar que a equação (3.5.21) é satisfeita inteiramente. Suponhamos que (3.5.21) é satisfeita $\forall\ |i-j| \leq k-1$ e verifiquemos se esta é satisfeita para $|i-j| = k$. Para esse fim, consideremos a s-ésima coluna da equação (3.5.5) e multiplicando-a por $x^{(s+1-k)T}$, temos

$$-b_{s-1}x^{(s+1-k)T}x^{(s-1)} + a_s x^{(s+1-k)T}x^{(s)} - b_s x^{(s+1-k)T}x^{(s+1)} = x^{(s+1-k)T}\Lambda x^{(s)}. \qquad (3.5.32)$$

Os dois primeiros termos do lado esquerdo desaparecem pela hipótese de indução e ficamos com

$$-b_s x^{(s+1-k)T}x^{(s+1)} = x^{(s+1-k)T}\Lambda x^{(s)}. \qquad (3.5.33)$$

Agora, multiplicando a (s+1-k)-ésima coluna da equação (3.5.5) por $x^{(s)T}$, obtemos

$$-b_{s-k}x^{(s)T}x^{(s-k)} + a_{s+1-k}x^{(s)T}x^{(s+1-k)} - b_{s+1-k}x^{(s)T}x^{(s+2-k)} = x^{(s)T}\Lambda x^{(s+1-k)}. \qquad (3.5.34)$$

Os dois últimos termos do lado esquerdo novamente desaparecem pela hipótese de indução. Ficamos então com

$$-b_{s-k}x^{(s)T}x^{(s-k)} = x^{(s)T}\Lambda x^{(s+1-k)}. \qquad (3.5.35)$$

Fazendo s=k e como por hipótese $x^{(0)} = 0$, obtemos

$$x^{(k)T}\Lambda x^{(1)} = 0. \qquad (3.5.36)$$

Substituindo-a em (3.5.33) quando s=k, obtemos que

$$x^{(s+1-k)}x^{(s+1)} = 0, \qquad (3.5.37)$$

pois $b_s \neq 0$. Esse último resultado sugere que novamente devemos proceder por indução.

Imaginemos que

$$x^{(r)T}x^{(r-k)} = 0 \ , \ r=k,k+1,\ldots,s-1. \qquad (3.5.38)$$

Substituindo s por s-1 na equação (3.5.34) e usando as hipóteses indutivas anteriores, obtemos

$$x^{(s-1)T}\Lambda x^{(s-k)} = 0. \qquad (3.5.39)$$

Usando este resultado na equação (3.5.33) onde novamente substituindo s por s-1 nos dás o resultado

$$x^{(s-k)T}x^{(s)} = 0. \qquad (3.5.40)$$

Assim, por indução, (3.5.40) está satisfeita para $s \geq k$, com s arbitrário. Esse fato completa a prova indutiva geral, desde que (3.5.40) implica que $x^{(i)}x^{(j)} = 0$, $\forall\ |i-j| = k$. Portanto, por indução, $k > 0$ arbitrário e (3.5.21) está completamente satisfeita. ■

Devemos salientar que na prática as condições de ortogonalidade $x^{(i)T}x^{(j)} = 0$ para $|i-j| > 1$ não serão satisfeitas precisamente porque em cada etapa do processo aumentarão alguns erros de arredondamento presentes. Mas essa dificuldade pode ser vencida extendendo o algoritmo anterior de modo que $x^{(n-i)}$ seja ortogonalizado com respeito aos vetores $x^{(n)}$, $x^{(n-1)}$, ..., $x^{(n-i+2)}$. Quando modificado, o algoritmo executará o processo satisfatoriamente.

CAPITULO 4

RECONSTRUÇÃO DE UMA MATRIZ DE JACOBI USANDO TRANSFORMAÇÃO DE HOUSEHOLDER

4.1 - INTRODUÇÃO

Originalmente esse algoritmo foi desenvolvido por Biegler-König [1] para reconstrução de matrizes banda. Analogamente ao método descrito no capítulo anterior, consideraremos apenas a reconstrução de uma matriz de Jacobi da forma

$$A = \begin{bmatrix} a_1 & -b_1 & 0 & \cdots & 0 \\ -b_1 & a_2 & -b_2 & & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & & a_{n-2} & -b_{n-2} & 0 \\ 0 & & -b_{n-2} & a_{n-1} & -b_{n-1} \\ 0 & \cdots & 0 & -b_{n-1} & a_n \end{bmatrix} \quad \text{, com } b_r > 0. \tag{4.1.1}$$

Tal método é numericamente estável e é realizado a partir dos valores espectrais $(\lambda_i)_1^n$ e $(\mu_i)_1^{n-1}$,

que são dados. O primeiro conjunto representa os autovalores da matriz A de ordem n a ser construída e, o segundo, os autovalores da submatriz principal esquerda de A de ordem n-1. Os dois conjuntos são reais e satisfazem a desigualdade,

$$\lambda_i < \mu_i < \lambda_{i+1} \quad (1) \quad , \text{ com } i=1,2,\ldots,n-1. \qquad (4.1.2)$$

Assim, o problema se resume em determinar uma matriz simétrica de ordem n, $A = A_n$. com autovalores $\lambda_1, \lambda_2, \ldots, \lambda_n$, sendo $\mu_1, \mu_2, \ldots, \mu_{n-1}$ autovalores de A_{n-1}, de forma que $a_{ij} = 0$ para $|i-j| \geq 2$. Notemos que o número de elementos não nulos de A é igual ao número de autovalores dados.

Na seção 4.2 construíremos uma matriz simétrica B, onde seus autovalores serão $(\lambda_i)_1^n$ e sua submatriz principal esquerda B_{n-1} terá como autovalores $(\mu_i)_1^{n-1}$. Mas B não será uma matriz tridiagonal.

Na seção 4.3 descreveremos as transformações de Householder que serão usadas na seção 4.4 para reduzir a matriz B a forma tridiagonal e, com modificações adequadas, a forma de Jacobi, de modo que permaneçam invariantes os autovalores de A e de A_{n-1}.

---

(1) - Já vimos no capítulo 1 que esta condição é necessária e suficiente para garantir a existência de uma matriz de Jacobi.

4.2 - CONSTRUÇÃO DE UMA MATRIZ SIMÉTRICA A PARTIR DOS VALORES ESPECTRAIS DADOS

Seja B_{n-1} uma matriz diagonal de ordem n-1 da forma

$$B_{n-1} = \text{diag}\,(\mu_1, \mu_2, \ldots, \mu_{n-1}). \qquad (4.2.1)$$

Consideremos, além da condição (4.1.2), que

$$\lambda_i < \lambda_{i+1},\ i=1,2,\ldots,n-1 \text{ e } \mu_j < \mu_{j+1},\ j=1,2,\ldots,n-2. \qquad (4.2.2)$$

Descreveremos a seguir como construir $B = B_n$ a partir de B_{n-1}.

Seja

$$B_n = \left[\begin{array}{c:c} B_{n-1} & \beta \\ \hdashline \beta^T & \delta \end{array}\right], \qquad (4.2.3)$$

onde

$$\beta^T = [\beta_1 \ \beta_2 \ \ldots \ \beta_{n-1}], \qquad (4.2.4)$$

e δ escalar.

Assim, desejamos obter β e δ de modo que B_n tenha como autovalores $\lambda_1, \lambda_2, \ldots, \lambda_n$. Calculemos, para

isso, o polinômio característico de B_n através da última linha de

$$B_n-\lambda I = \left[\begin{array}{cccc|c} \mu_1-\lambda & 0 & \cdots & 0 & \beta_1 \\ 0 & \mu_2-\lambda & & \vdots & \beta_2 \\ \vdots & & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & \mu_{n-1}-\lambda & \beta_{n-1} \\ \hline \beta_1 & \beta_2 & \cdots & \beta_{n-1} & \delta-\lambda \end{array}\right]. \qquad (4.2.5)$$

Logo

$$|B_n-\lambda I| = (\delta-\lambda)(-1)^{2n}\prod_{i=1}^{n-1}(\mu_i-\lambda) + \ldots +$$

$$+\beta_k(-1)^{n+k}\begin{vmatrix} \mu_1-\lambda & 0 & & \cdots & & \beta_1 \\ 0 & \mu_2-\lambda & & & & \beta_2 \\ \vdots & & \ddots & & & \vdots \\ & & 0 & & & \\ & & \mu_{k+1}-\lambda & \ddots & & \\ 0 & & \cdots & 0 & \mu_{n-1}-\lambda & \beta_{n-1} \end{vmatrix} + \ldots =$$

$$(\delta-\lambda)\prod_{i=1}^{n-1}(\mu_i-\lambda) + \ldots + (\beta_k)^2(-1)^{2(n+k)-1}\begin{vmatrix} \mu_1-\lambda & & 0 \\ & \ddots & \\ 0 & & \mu_{n-1}-\lambda \end{vmatrix} +$$

$$+ \ldots = (\delta-\lambda)\prod_{i=1}^{n-1}(\mu_i-\lambda) + \ldots - (\beta_k)^2\prod_{\substack{j=1 \\ j\neq k}}^{n-1}(\mu_j-\lambda) + \ldots .$$

Então o polinômio característico de B_n deve ser da forma

$$\prod_{i=1}^{n}(\lambda_i-\lambda) = (\delta-\lambda)\prod_{i=1}^{n-1}(\mu_i-\lambda) - \sum_{i=1}^{n-1}(\beta_i)^2 \prod_{\substack{j=1\\ j\neq i}}^{n-1}(\mu_j-\lambda). \qquad (4.2.6)$$

Em particular, para $\lambda = \mu_k$, com k=1,2,...,n-1 temos

$$\prod_{i=1}^{n}(\lambda_i-\mu_k) = -(\beta_k)^2 \prod_{\substack{j=1\\ j\neq k}}^{n-1}(\mu_j-\mu_k). \qquad (4.2.7)$$

Dessa forma obtemos que

$$\beta_k = \pm \left[\frac{-\prod_{i=1}^{n}(\lambda_i-\mu_k)}{\prod_{\substack{j=1\\ j\neq k}}^{n-1}(\mu_j-\mu_k)} \right]^{1/2}, \quad k=1,2,\ldots,n-1, \qquad (4.2.8)$$

com $\beta \in R^{n-1}$ pois, pelas condições (4.1.2) e (4.2.2), temos que

$$\text{sinal}\left(\prod_{i=1}^{n}(\lambda_i-\mu_k)\right) = (-1)^{k-1},$$

$$\text{sinal}\left(\prod_{\substack{j=1\\ j\neq k}}^{n-1}(\mu_j-\mu_k)\right) = (-1)^{k}. \qquad (4.2.9)$$

Como traço B_n = traço B_{n-1} + δ, temos

$$\delta = \sum_{i=1}^{n}\lambda_i - \sum_{i=1}^{n-1}\mu_i. \qquad (4.2.10)$$

Observemos que podemos escolher cada β_k com valor positivo ou negativo de modo que existem 2^n soluções. Assim obtemos explicitamente uma matriz B, a qual satisfaz todas as condições da matriz A procurada, exceto a forma tridiagonal.

4.3 - REDUÇÃO DE UMA MATRIZ SIMÉTRICA A UMA FORMA TRIDIAGONAL ATRAVÉS DE TRANSFORMAÇÕES DE HOUSEHOLDER

As matrizes de transformação de Householder possuem a seguinte forma

$$P = I - 2ww^T, \qquad (4.3.1)$$

onde w é um vetor unitário, ou seja,

$$w^Tw = \|w\|_2^2 = 1. \qquad (4.3.2)$$

Essas matrizes são ortogonais pois,

$$P^TP = (I-2ww^T)(I-2ww^T) = I \qquad (4.3.3)$$

e, evidentemente simétricas. Portanto

$$P = P^T = P^{-1}. \qquad (4.3.4)$$

As matrizes de Householder transformam um vetor real arbitrário em um outro vetor desejado de mesmo comprimento. Geometricamente, uma matriz desse tipo reflete cada vetor sobre o lado oposto de um plano - o plano perpendicular a w. Em geral, o vetor w é determinado em função de um vetor v que se deseja transformar mediante a matriz P em um vetor y, com alguma componente nula.

Assim, utilizando as matrizes de Householder, podemos reduzir uma matriz simétrica real a forma tridiagonal, mantendo seus autovalores. Isso é feito através de transformações ortogonais de semelhança, sendo que em cada etapa uma coluna inteira é deixada na forma desejada.

Seja A uma matriz real, simétrica,de ordem n. Definimos (n-2) matrizes ortogonais $P^{(1)}$, $P^{(2)}$, ..., $P^{(n-2)}$ e as matrizes $A^{(1)} = A$, $A^{(2)}$, ..., $A^{(n-1)}$ por

$$A^{(i+1)} = P^{(i)} A^{(i)} P^{(i)T}, \quad i=1,2,\ldots,n-2 , \qquad (4.3.5)$$

tal que $A^{(n-1)}$ seja uma matriz tridiagonal. Como cada $P^{(i)}$ é ortogonal, todas as matrizes $A^{(1)}$, $A^{(2)}$, ..., $A^{(n-1)}$ são semelhantes e, portanto, tem os mesmos autovalores. Além disso, cada $A^{(i)}$ mantem a simetria de $A^{(1)}$.

4.4 - REDUÇÃO DA MATRIZ SIMÉTRICA B A UMA MATRIZ DE JACOBI

Para transformar a matriz simétrica B em

matriz tridiagonal, onde os elementos fora da diagonal principal sejam todos negativos, é necessário escolher adequadamente os sinais de β_k, $\forall$ k, de forma que as transformsações de Householder gerem além de uma matriz tridiagonal uma matriz de Jacobi. Começamos a zerar os elementos fora da banda na n-ésima coluna (ou linha), ou seja, os elementos b_{ij} tal que $|i-j| \geq 2$. E continuamos com os da (n-1)-ésima, (n-2)-ésima,... , terceira coluna (ou linha), usando transformações de Householder.

Trabalhando da direita para a esquerda nas colunas (ou linhas) não destruímos, dessa forma, os autovalores de B nem de sua submatriz principal esquerda de ordem n-1.

Construíremos matrizes semelhantes $B = A^{(n-1)}$, $A^{(n-2)}$, ..., $A^{(1)} = A$, a partir das matrizes de transformação $P^{(n-1)}$, $P^{(n-2)}$, ..., $P^{(2)}$.

Nosso objetivo agora será o de obter o vetor w para que a matriz ortogonal $P^{(n-1)}$ aja sobre $A^{(n-1)}$, através da transformação de semelhança

$$A^{(n-2)} = P^{(n-1)} A^{(n-1)} P^{(n-1)T}, \qquad (4.4.1)$$

afim de que $A^{(n-2)}$ tenha zeros na sua n-ésima coluna e linha fora das posições tridiagonais.

Seja o vetor unitário w da forma

$$w^T = [w_1 \; w_2 \; \ldots \; w_{n-1} \; 0], \qquad (4.4.2)$$

em consequência,

$$P^{(n-1)} = \begin{bmatrix} 1-2w_1^2 & -2w_1w_2 & \cdots & -2w_1w_{n-1} & 0 \\ -2w_2w_1 & 1-2w_2^2 & & & \vdots \\ \vdots & & \ddots & & \\ -2w_{n-2}w_1 & & & -2w_{n-2}w_{n-1} & 0 \\ -2w_{n-1}w_1 & \cdots & -2w_{n-1}w_{n-2} & 1-2w_{n-1} & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} . \qquad (4.4.3)$$

Assim por (4.4.1) temos que a n-ésima coluna de $A^{(n-2)}$ é dada por

$$\begin{aligned} a_{1n}^{(n-2)} &= (1-2w_1^2)a_{1n}^{(n-1)} - 2w_1w_2a_{2n}^{(n-1)} \cdots - 2w_1w_{n-1}a_{n-1n}^{(n-1)} \\ a_{2n}^{(n-2)} &= -2w_2w_1a_{1n}^{(n-1)} + (1-2w_2^2)a_{2n}^{(n-1)} \cdots - 2w_2w_{n-1}a_{n-1n}^{(n-1)} \\ &\vdots \\ a_{n-2n}^{(n-2)} &= -2w_{n-2}w_1a_{1n}^{(n-1)} - 2w_{n-2}w_2a_{2n}^{(n-1)} \cdots - 2w_{n-2}w_{n-1}a_{n-1n}^{(n-1)} \\ a_{n-1n}^{(n-2)} &= -2w_{n-1}w_1a_{1n}^{(n-1)} - 2w_{n-1}w_2a_{2n}^{(n-1)} \cdots + (1-2w_{n-1}^2)a_{n-1n}^{(n-1)} \\ a_{nn}^{(n-2)} &= a_{nn}^{(n-1)} . \end{aligned} \qquad (4.4.4)$$

Mas os elementos fora da diagonal na última coluna de $A^{(n-2)}$ devem ser todos nulos, exceto o elemento da diagonal superior a diagonal principal, ou seja, $a_{n-1n}^{(n-2)}$. De (4.4.4) temos então que

$$
\begin{aligned}
a_{1n}^{(n-2)} &= a_{1n}^{(n-1)} - 2w_1 h = 0 \\
a_{2n}^{(n-2)} &= a_{2n}^{(n-1)} - 2w_2 h = 0 \\
&\vdots \\
a_{n-2n}^{(n-2)} &= a_{n-2n}^{(n-1)} - 2w_{n-2} h = 0 \\
a_{n-1n}^{(n-2)} &= a_{n-1n}^{(n-1)} - 2w_{n-1} h = s,
\end{aligned}
\qquad (4.4.5)
$$

onde

$$
h = a_{1n}^{(n-1)} w_1 + a_{2n}^{(n-1)} w_2 + \ldots + a_{n-1n}^{(n-1)} w_{n-1}. \qquad (4.4.6)
$$

Mas pela condição (4.3.2) temos

$$
w_1^2 + w_2^2 + \ldots + w_{n-1}^2 = 1. \qquad (4.4.7)
$$

Essas equações são suficientes para determinarmos o vetor w e também s. Elevando ao quadrado as equações de (4.4.5) e adicionando-as temos

$$
\begin{aligned}
&(a_{1n}^{(n-2)})^2 + (a_{2n}^{(n-2)})^2 + \ldots + (a_{n-1n}^{(n-2)})^2 = \\
&= (a_{1n}^{(n-1)})^2 + (a_{2n}^{(n-1)})^2 + \ldots + (a_{n-1n}^{(n-1)})^2 = s^2 .
\end{aligned}
\qquad (4.4.8)
$$

Vemos assim, que a soma dos quadrados dos elementos não diagonais da última coluna é invariante. E como

$$
a_{kn}^{(n-2)} = 0 \ , \ k=1,2,\ldots,n-2 \ , \qquad (4.4.9)
$$

obtemos

$$a_{n-1\,n}^{(n-2)} = s = \pm\left[\sum_{k=1}^{n-1} (a_{kn}^{(n-1)})^2\right]^{1/2}. \qquad (4.4.10)$$

De (4.4.5) vem que

$$a_{n-1\,n}^{(n-1)} - 2w_{n-1}h = s. \qquad (4.4.11)$$

A condição (4.4.9) nos leva às equações

$$a_{kn}^{(n-1)} - 2w_k h = 0, \quad k=1,2,\ldots,n-2. \qquad (4.4.12)$$

Multiplicando (4.4.11) por w_{n-1} e cada uma das equações (4.4.12) pelos correspondentes w_k e adicionando as equações resultantes obtemos

$$\sum_{k=1}^{n-1} (w_k a_{kn}^{(n-1)} - 2hw_k^2) = sw_{n-1}. \qquad (4.4.13)$$

Utilizando novamente a condição de normalização $w^T w = 1$ e a equação (4.4.6) obtemos

$$h = -sw_{n-1}. \qquad (4.4.14)$$

Das equações (4.4.11) e (4.4.14) resulta que

$$w_{n-1} = \left[\frac{1}{2}\left[1 - \frac{a^{(n-1)}_{n-1\,n}}{s}\right]\right]^{1/2} . \qquad (4.4.15)$$

Então (4.4.14) nos dá o valor de h e, finalmente por (4.4.12) temos os valores de $w_1, w_2, \ldots, w_{n-2}$ que são dados por

$$w_k = \frac{a^{(n-1)}_{k\,n}}{2h} , \quad k=1,2,\ldots,n-2. \qquad (4.4.16)$$

Em (4.4.10) o sinal de s é escolhido a fim de ser o mesmo de $-a^{(n-1)}_{n-1\,n}$ para que não haja, em (4.4.15), nenhuma perda de sentido no radicando. Mas como $a^{(n-1)}_{n-1\,n} = \beta_{n-1}$ e estamos considerando β_{n-1} positivo devemos tomar s negativo.

De acordo com (4.4.5) temos que

$$w^T = \left[\frac{a^{(n-1)}_{1\,n}}{2h}, \frac{a^{(n-1)}_{2\,n}}{2h}, \ldots, \frac{a^{(n-1)}_{n-2\,n}}{2h}, \frac{a^{(n-1)}_{n-1\,n} - s}{2h}, 0\right]. \qquad (4.4.17)$$

Porém de acordo com (4.4.14) e (4.4.15) podemos reescrevê-lo como

$$w^T = \mu \left[a^{(n-1)}_{1\,n} \quad a^{(n-1)}_{2\,n} \quad \ldots \quad a^{(n-1)}_{n-2\,n} \quad a^{(n-1)}_{n-1\,n} - s \quad 0\right] , \qquad (4.4.18)$$

onde

$$\mu = \frac{1}{\left[2s(s-a_{n-1\,n}^{(n-1)})\right]^{1/2}} . \qquad (4.4.19)$$

Portanto, a última coluna de $A^{(n-2)}$ terá a forma desejada desde que o vetor unitário w da matriz transformação $P^{(n-1)}$ seja dado por (4.4.18). Como $A^{(n-2)}$ é simétrica, a sua última linha também terá a mesma forma, onde

$$\begin{aligned} a_{nn}^{(n-2)} &= a_{nn}^{(n-1)}, \\ a_{n-1\,n}^{(n-2)} &= a_{nn-1}^{(n-2)} = s. \end{aligned} \qquad (4.4.20)$$

Continuando a redução de $B = A^{(n-1)}$ a forma tridiagonal, observemos que a próxima etapa é completamente análoga a esta.

Definimos

$$P^{(n-2)} = I - 2ww^T, \qquad (4.4.21)$$

onde w é um vetor unitário com suas duas últimas componentes iguais a zero. Logo, a transformação

$$A^{(n-3)} = P^{(n-2)} A^{(n-2)} P^{(n-2)\,T}, \qquad (4.4.22)$$

não alterá a última coluna e linha de $A^{(n-2)}$, afetando somente a submatriz definida pelas n-1 primeiras colunas de $A^{(n-2)}$. Portanto os elementos do vetor w são definidos analogamente aos correspondentes a w na etapa anterior, ou seja, é da forma

$$w^T = \mu\left[a_{1\,n-1}^{(n-2)} \quad a_{2\,n-1}^{(n-2)} \quad \cdots \quad a_{n-2\,n-1}^{(n-2)} - s \quad 0 \quad 0\right], \qquad (4.4.23)$$

com

$$\mu = \frac{1}{\left[2s\left(s - a_{n-2\,n-1}^{(n-2)}\right)\right]^{1/2}}, \qquad (4.4.24)$$

$$s = \pm\left[\sum_{k=1}^{n-2}\left(a_{k\,n-1}^{(n-2)}\right)^2\right]^{1/2}. \qquad (4.4.25)$$

Desse modo as r últimas colunas e linhas de $A^{(n-r-1)}$ são da forma tridiagonal. Então

$$A^{(n-r-2)} = P^{(n-r-1)} A^{(n-r-1)} P^{(n-r-1)\,T}, \qquad (4.4.26)$$

onde a matriz transformação $P^{(n-r-1)}$ é construída a partir do vetor unitário w, dado por

$$w^T = \mu\left[a_{1\,n-r}^{(n-r-1)} \quad a_{2\,n-r}^{(n-r-1)} \quad \cdots \quad a_{n-r-1\,n-r}^{(n-r-1)} - s \quad 0 \ \ldots \ 0\right] \qquad (4.4.27)$$

com

$$\mu = \frac{1}{\left[2s(s-a_{n-r-1\,n-r}^{(n-r-1)})\right]^{1/2}}, \qquad (4.4.28)$$

$$s = \pm\left[\sum_{k=1}^{n-r-1} (a_{k\,n-r}^{(n-r-1)})^2\right]^{1/2}. \qquad (4.4.29)$$

Essa transformação não muda os autovalores das duas maiores submatrizes principais de $A^{(n-r-1)}$ e a forma das suas últimas r colunas e linhas. Em consequência, as duas maiores submatrizes principais de $A^{(1)} = A$ são da forma tridiagonal e semelhantes as correspondentes submatrizes de $A^{(n-1)} = B$.

CAPITULO 5

RECONSTRUÇÃO DE UMA MATRIZ DE JACOBI A PARTIR DE DADOS ESPECTRAIS

5.1 - INTRODUÇÃO

Neste capítulo, como nos anteriores, continuaremos estudando a reconstrução de uma matriz de Jacobi, com a ressalva que este se diferencia dos demais pelos valores espectrais dados.

Esta reconstrução foi desenvolvida por Gladwell [8] onde ele faz aplicações a um modelo físico.

Consideremos a matriz de Jacobi de ordem n a ser reconstruída, denotada por (1.2.1), sob a forma

$$A = \left[\begin{array}{c|c|c} B & -b_m & 0 \\ \hline -b_m & a_{m+1} & -b_{m+1} \\ \hline 0 & -b_{m+1} & C \end{array}\right], \qquad (5.1.1)$$

onde $B \in R^{m\times m}$, $C \in R^{p\times p}$ e $p = n-(m+1)$.

Dadas as sequências $(\lambda_i)_1^n$, $(\mu_i)_1^m$ e $(\gamma_i)_1^p$ satisfazendo as seguintes desiguldades

$$\begin{aligned} &\lambda_1 < \lambda_2 < \ldots < \lambda_n \\ &\mu_1 < \mu_2 < \ldots < \mu_m \\ &\gamma_1 < \gamma_2 < \ldots < \gamma_p, \end{aligned} \qquad (5.1.2)$$

desejamos construir A de forma que λ_1, λ_2, ..., λ_n sejam os seus autovalores, μ_1, μ_2, ..., μ_m corresponda aos autovalores de B, que é a submatriz principal esquerda de A de ordem m e γ_1, γ_2, ..., γ_p sejam os autovalores de C, que corresponde a submatriz principal direita de A de ordem p.

Observemos que B e C, por sua vez, são matrizes de Jacobi de ordem m e p, respectivamente.

Há dois casos a se considerar:

1º - Quando os dois conjuntos de autovalores $(\mu_i)_1^m$ e $(\gamma_i)_1^p$ são disjuntos. Se isso ocorre temos garantido a unicidade de construção a menos de um fator escalar.

2º - Quando existe um ou mais pares (μ_j, γ_k) idênticos, acarretando a coincidência também com os autovalores de $(\lambda_i)_1^n$. Neste caso a reconstrução não será única e a matriz resultante dependerá das escolhas feitas no processo.

Estes dois casos serão vistos com detalhes nas seções subsequentes, sendo que a reconstrução de A recairá nos métodos vistos nos capítulos 2 e 3, ou seja, poderá ser

feita através da sequência de polinômios ortogonais ou do algoritmo de Lanczos.

Trabalharemos com a seguinte notação: $\{P_i(\lambda)\}_1^n$ corresponderá aos menores principais esquerdos de $(\lambda I-A)$ e $\{Q_i(\lambda)\}_1^n$ aos menores principais direitos de $(\lambda I-A)$.

5.2 - RECONSTRUÇÃO DA MATRIZ DE JACOBI ATRAVÉS DE POLINÔMIOS ORTOGONAIS

Dadas as sequências $(\mu_i)_1^m$ e $(\gamma_i)_1^p$, suponhamos que os seus elementos sejam distintos entre si. Podemos, dessa forma, arrumar todos os elementos em ordem crescente e os denotamos por $(\xi_i)_1^{n-1}$.

Através das propriedades de matrizes de Jacobi temos que a desigualdade

$$\lambda_i < \xi_i < \lambda_{i+1}, \quad \forall\, i=1,2,\ldots,n-1 \qquad (5.2.1)$$

é satisfeita.

Considerando as expansões de Laplace[1] para a matriz $(\lambda I-A)$ as quais usando as m primeiras linhas ou as $(m+1)$ primeiras obtemos

$$P_n(\lambda) = P_m(\lambda)Q_{p+1}(\lambda) - b_m^2 P_{m-1}(\lambda)Q_p(\lambda), \qquad (5.2.2)$$

---

(1) - Ver referência [17].

$$P_n(\lambda) = P_{m+1}(\lambda)Q_p(\lambda) - b_{m+1}^2 P_m(\lambda)Q_{p-1}(\lambda). \qquad (5.2.3)$$

Como

$$P_n(\lambda) = \prod_{i=1}^{n} (\lambda-\lambda_i), \qquad (5.2.4)$$

$$P_m(\lambda) = \prod_{i=1}^{m} (\lambda-\mu_i), \qquad (5.2.5)$$

$$Q_p(\lambda) = \prod_{i=1}^{p} (\lambda-\gamma_i), \qquad (5.2.6)$$

são os polinômios característicos de A, B e C, respectivamente, temos para $\lambda = \mu_i$ que a equação (5.2.2) é dada por

$$P_n(\mu_i) = - b_m^2 P_{m-1}(\mu_i)Q_p(\mu_i), \quad i=1,2,\ldots,m, \qquad (5.2.7)$$

e para $\lambda = \gamma_i$ a equação (5.2.3) é dada por

$$P_n(\gamma_i) = - b_{m+1}^2 P_m(\gamma_i)Q_{p-1}(\gamma_i), \quad i=1,2,\ldots,p. \qquad (5.2.8)$$

Do capítulo 1 sabemos que

$$\begin{aligned} &P_{m-1}(\mu_i) \neq 0, \ \forall\ i=1,2,\ldots,m, \\ &Q_{p-1}(\gamma_j) \neq 0, \ \forall\ j=1,2,\ldots,p, \end{aligned} \qquad (5.2.9)$$

pois B e C são matrizes de Jacobi.

Como γ_i e μ_j são diferentes $\forall$ i,j, temos que $Q_p(\mu_i)$ e $P_m(\gamma_i)$ são não nulos de modo que, a menos dos fatores b_m^2 e b_{m+1}^2, as equações (5.2.7) e (5.2.8) fornecem $P_{m-1}(\mu_i)$ e $Q_{p-1}(\mu_i)$, respectivamente.

Necessitaremos desses valores para reconstruir B e C através de polinômios ortogonais conforme explicitado no capítulo 1. Para realizar tais reconstruções nos falta apenas os pesos $(\omega_i)_B$ e $(\omega_i)_C$ usados no produto interno para gerar a sequência de polinômios ortogonais.

Descreveremos a seguir a obtenção dos pesos $(\omega_i)_B$ para a matriz B.

De (2.3.22) temos que

$$b_m^2(\omega_i)_B = b_m^2 \frac{P_{m-1}(\mu_i)}{P'_m(\mu_i)}. \qquad (5.2.10)$$

Mas (5.2.7) podemos escrevê-la como

$$b_m^2(\omega_i)_B = - \frac{P_n(\mu_i)}{P'_m(\mu_i)Q_p(\mu_i)}. \qquad (5.2.11)$$

Para verificarmos que os pesos são positivos suponhamos que μ_i tem s γ's com valor menor e p-s com valores maiores, então por (5.2.1) temos $\lambda_{i+s} < \mu_i < \lambda_{i+s+1}$, desse modo

$$\text{sinal}\,(P_n(\mu_i)) = (-1)^{n-i-s},$$
$$\text{sinal}\,(Q_p(\mu_i)) = (-1)^{p-s}, \qquad (5.2.12)$$
$$\text{sinal}\,(P'_m(\mu_i)) = (-1)^{m-i}.$$

Logo

$$\text{sinal}\,[(\omega_i)_B] = (-1)^{2(n-i-s)} = (+). \qquad (5.2.13)$$

Assim B pode ser construída unicamente.

No final do processo os valores de $P_{m-1}(\mu_i)$ serão conhecidos $\forall$ i=1,2,...,m, através da equação (2.3.21), sendo que qualquer um destes pode ser substituído na equação (5.2.7) a fim de obter-se o valor de b_m^2.

A matriz C e o escalar b_{m+1}^2 podem ser reconstruídos de maneira similar.

Consideremos agora o segundo caso, onde existem valores idênticos entre os conjuntos $(\mu_j)_1^m$ e $(\gamma_k)_1^p$.

Suponhamos que $\mu_j = \gamma_k$. Mas por (5.1.2) podemos reordenar $\mu_1, \mu_2, \ldots, \mu_{j-1}, \gamma_1, \gamma_2, \ldots, \gamma_{k-1}$, denotando-os por $(\xi_i)_1^{j+k-2}$. Pelas propriedades de matrizes de Jacobi temos

$$\lambda_1 < \xi_1 < \lambda_2 < \ldots < \xi_{j+k-2} < \lambda_{j+k-1} < \mu_j < \lambda_{j+k} < \gamma_k < \lambda_{j+k+1} < \ldots \qquad (5.2.14)$$

Dessa forma

$$\mu_j = \gamma_k = \lambda_l, \qquad (5.2.15)$$

com $l = j+k$.

Através das relações de recorrência

$$P_{m+1}(\lambda) = (\lambda - a_{m+1})P_m(\lambda) - b_m^2 P_{m-1}(\lambda), \qquad (5.2.16)$$

$$Q_{p+1}(\lambda) = (\lambda - a_{m+1})Q_p(\lambda) - b_{m+1}^2 Q_{p-1}(\lambda), \qquad (5.2.17)$$

as quais os polinômios característicos de uma matriz de Jacobi e de suas submatrizes satisfazem, temos que

$$P_{m+1}(\mu_j) = - b_m^2 P_{m-1}(\mu_j), \qquad (5.2.18)$$

$$Q_{p+1}(\gamma_k) = - b_{m+1}^2 Q_{p-1}(\gamma_k). \qquad (5.2.19)$$

Derivando-se a equação (5.2.2) (ou a equação (5.2.3)) e fazendo $\lambda = \lambda_l = \mu_j = \gamma_k$ temos

$$P_n'(\lambda_l) = P_m'(\mu_j)Q_{p+1}(\gamma_k) - b_m^2 P_{m-1}(\mu_j)Q_p'(\gamma_k). \qquad (5.2.20)$$

Substituindo (5.2.18) e (5.2.19) em (5.2.20) obtemos

$$P_n'(\lambda_l) = - b_m^2 P_{m-1}(\mu_j)Q_p'(\gamma_k) - b_{m+1}^2 P_m'(\mu_j)Q_{p-1}(\gamma_k). \qquad (5.2.21)$$

Esta equacão ocupa o lugar das equações (5.2.7) e (5.2.8) para o autovalor comum.

Dividindo (5.2.21) por $P_m'(\mu_j)Q_p'(\gamma_k)$ obtemos

$$W = - \frac{P_n'(\lambda_l)}{P_m'(\mu_j)Q_p'(\gamma_k)} = b_m^2 \frac{P_{m-1}(\mu_j)}{P_m'(\mu_j)} + b_{m+1}^2 \frac{Q_{p-1}(\gamma_k)}{Q_p'(\gamma_k)} .$$

(5.2.22)

Mas de acordo com a equação (5.2.10)

$$(\omega_j)_B = \frac{P_{m-1}(\mu_j)}{P_m'(\mu_j)}, \qquad (5.2.23)$$

$$(\omega_k)_C = \frac{Q_{p-1}(\gamma_k)}{Q_p'(\gamma_k)}. \qquad (5.2.24)$$

Então (5.2.22) torna-se

$$W = - \frac{P_n'(\lambda_l)}{P_m'(\mu_j)Q_p'(\gamma_k)} = b_m^2(\omega_j)_B + b_{m+1}^2(\omega_k)_C. \qquad (5.2.25)$$

Observemos que W é uma quantidade positiva.

Desejamos obter os valores para $(\omega_j)_B$ e $(\omega_k)_C$.

Para este caso podemos encontrar uma família de matrizes tridiagonais tendo os autovalores especificados.

Escolhamos $\alpha \in (0, \pi/2)$ e façamos

$$\begin{aligned} b_m^2(\omega_j)_B &= W(\cos\alpha)^2, \\ b_{m+1}^2(\omega_k)_C &= W(\mathrm{sen}\alpha)^2. \end{aligned} \qquad (5.2.26)$$

Obtemos, assim, as quantidades $b_m^2(\omega_j)_B$ e $b_{m+1}^2(\omega_k)_C$ que faltavam para realizar a construção de B e C,

respectivamente, desde que os outros pesos necessários para a reconstrução são obtidos conforme o primeiro caso, pois não há coincidência de autovalores.

Se houver mais de um par comum de autovalores esse procedimento deve ser seguido para cada um.

Na etapa final do algoritmo de reconstrução, os polinômios $P_{m-1}(\mu_j)$ e $Q_{p-1}(\gamma_k)$ serão conhecidos de modo que b_m^2 e b_{m+1}^2 podem ser encontrados pelas equações (5.2.7) e (5.2.8), respectivamente, aplicadas a um dos autovalores distintos.

5.3 - RECONSTRUÇÃO DA MATRIZ DE JACOBI ATRAVÉS DO ALGORITMO DE LANCZOS

Seja o problema de autovalor de A dado por $(A-\lambda I)x = 0$.

Expressaremos tal problema em termos dos autovetores normalizados $(y^{(i)})_1^m$ e $(z^{(i)})_1^p$ de B e C, respectivamente.

Dessa forma temos

$$x = \sum_{i=1}^{m} p_i \begin{bmatrix} y^{(i)} \\ 0 \\ \vdots \\ 0 \end{bmatrix} + x_{m+1} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \sum_{j=1}^{p} q_j \begin{bmatrix} 0 \\ \vdots \\ 0 \\ z^{(j)} \end{bmatrix}, \qquad (5.3.1)$$

onde

$$[x] = \begin{bmatrix} p_1 \\ \vdots \\ p_m \\ x_{m+1} \\ q_1 \\ \vdots \\ q_p \end{bmatrix} \qquad (5.3.2)$$

corresponde as coordenadas do autovetor x na base formada pelos vetores

$$\begin{bmatrix} y^{(i)} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \; e^{m+1}, \; \begin{bmatrix} 0 \\ \vdots \\ 0 \\ z^{(j)} \end{bmatrix}, \; \text{com} \quad \begin{matrix} i=1,2,\ldots,m \\ j=1,2,\ldots,p \end{matrix} . \qquad (5.3.3)$$

Aplicando a transformação linear representada por $(\lambda I - A)$ nos elementos da base, observamos que

$$(\lambda I - A) \begin{bmatrix} y^{(i)} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} (\lambda - \mu_i) y^{(i)} \\ b_m y_m^{(i)} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \; \forall \; i=1,2,\ldots,m; \qquad (5.3.4)$$

$$(\lambda I - A) e^{m+1} = \begin{bmatrix} 0 \\ \vdots \\ b_m \\ \lambda - a_{m+1} \\ b_{m+1} \\ \vdots \\ 0 \end{bmatrix}; \qquad (5.3.5)$$

$$(\lambda I - A)\begin{bmatrix} 0 \\ \vdots \\ 0 \\ z^{(j)} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ b_{m+1} z_1{}^{(i)} \\ (\lambda-\gamma_i)\ z^{(j)} \end{bmatrix}. \qquad (5.3.6)$$

Agora, escrevendo-os como combinação linear da nova base, obtemos que a matriz que representa a transformação linear na nova base é dada por

$$G = \left[\begin{array}{ccc|c|ccc} \lambda-\mu_1 & & 0 & s_1 & & & \\ & \ddots & & \vdots & & & \\ 0 & & \lambda-\mu_m & s_m & & & \\ \hline s_1 & \cdots & s_m & \lambda-a_{m+1} & t_1 & \cdots & t_p \\ \hline & & & t_1 & \lambda-\gamma_1 & & 0 \\ & & & \vdots & & \ddots & \\ & & & t_p & 0 & & \lambda-\gamma_p \end{array}\right], \qquad (5.3.7)$$

onde

$$\begin{aligned} s_i &= b_m y_m{}^{(i)}, \\ t_i &= b_{m+1} z_1{}^{(i)}. \end{aligned} \qquad (5.3.8)$$

Logo o problema de autovalor torna-se

$$\left[\begin{array}{ccc|c|ccc} \lambda-\mu_1 & & 0 & s_1 & & & \\ & \ddots & & \vdots & & & \\ 0 & & \lambda-\mu_m & s_m & & & \\ \hline s_1 & \cdots & s_m & \lambda-a_{m+1} & t_1 & \cdots & t_p \\ \hline & & & t_1 & \lambda-\gamma_1 & & 0 \\ & & & \vdots & & \ddots & \\ & & & t_p & 0 & & \lambda-\gamma_p \end{array}\right]\begin{bmatrix} p_1 \\ \vdots \\ p_m \\ x_{m+1} \\ q_1 \\ \vdots \\ q_p \end{bmatrix} = 0. \qquad (5.3.9)$$

Desta equação obtemos

$$\begin{aligned}(\lambda-\mu_i)p_i + s_i x_{m+1} &= 0 \ , \ i=1,2,\ldots,m \\ (\lambda-\gamma_j)q_j + t_j x_{m+1} &= 0 \ , \ j=1,2,\ldots,p.\end{aligned} \qquad (5.3.10)$$

Donde vem que

$$p_i = - \frac{s_i x_{m+1}}{(\lambda-\mu_i)} \ , \ \forall \ i=1,2,\ldots,m \ , \qquad (5.3.11)$$

$$q_j = - \frac{t_j x_{m+1}}{(\lambda-\gamma_j)} \ , \ \forall \ j=1,2,\ldots,p. \qquad (5.3.12)$$

A (m+1)-ésima equação do problema de autovalor (5.3.9) é dada por

$$\sum_{i=1}^{m} s_i p_i + (\lambda-a_{m+1})x_{m+1} + \sum_{j=1}^{p} q_j t_j = 0. \qquad (5.3.13)$$

Substituindo (5.3.11) e (5.3.12) em (5.3.13) obtemos

$$\left[(a_{m+1}-\lambda) + \sum_{i=1}^{m} \frac{s_i^2}{(\lambda-\mu_i)} + \sum_{j=1}^{p} \frac{t_j^2}{(\lambda-\gamma_j)}\right] x_{m+1} = 0. \qquad (5.3.14)$$

Fazendo $\lambda = \lambda_k$, $\forall \ k=1,2,\ldots,n$, tem-se

$$\left[(a_{m+1}-\lambda_k) + \sum_{i=1}^{m} \frac{s_i^2}{(\lambda_k-\mu_i)} + \sum_{j=1}^{p} \frac{t_j^2}{(\lambda_k-\gamma_j)}\right] x_{m+1}^{(k)} = 0. \quad (5.3.15)$$

Para o caso em que $(\mu_i)_1^m$ e $(\gamma_i)_1^p$ são diferentes, temos que $x_{m+1}^{(k)} \neq 0$, $\forall$ k=1,2,...,n, pois se $x_{m+1}^{(k)} = 0$ teríamos que

$$(\lambda_k I - A) \begin{bmatrix} x_1^{(k)} \\ \vdots \\ x_m^{(k)} \\ 0 \\ x_1'^{(k)} \\ \vdots \\ x_p'^{(k)} \end{bmatrix} = 0, \quad (5.3.16)$$

e isso acarretaria que

$$(\lambda_k I - B) \begin{bmatrix} x_1^{(k)} \\ \vdots \\ x_m^{(k)} \end{bmatrix} = 0, \quad (5.3.17)$$

$$(\lambda_k I - C) \begin{bmatrix} x_1'^{(k)} \\ \vdots \\ x_p'^{(k)} \end{bmatrix} = 0, \quad (5.3.18)$$

ou seja, λ_k também seria autovalor de B e C, o que não é o caso.

Assim, de (5.3.14) temos para o primeiro caso que

$$(a_{m+1}-\lambda) + \sum_{i=1}^{m} \frac{s_i^2}{(\lambda-\mu_i)} + \sum_{j=1}^{p} \frac{t_j^2}{(\lambda-\gamma_j)} \equiv 0, \qquad (5.3.19)$$

quando $\lambda = \lambda_k$, $\forall\ k=1,2,\ldots,n$. Escrevendo-a na forma racional temos

$$\frac{(a_{m+1}-\lambda)\prod_{i=1}^{m}(\lambda-\mu_i)\prod_{j=1}^{p}(\lambda-\gamma_j) + \sum_{i=1}^{m} s_i^2 \prod_{\substack{k=1\\k\neq i}}^{m}(\lambda-\mu_k)\prod_{j=1}^{p}(\lambda-\gamma_j)}{\prod_{i=1}^{m}(\lambda-\mu_i)\prod_{j=1}^{p}(\lambda-\gamma_j)} +$$

$$+ \frac{\sum_{i=1}^{p} t_j^2 \prod_{\substack{k=1\\k\neq j}}^{p}(\lambda-\gamma_k)\prod_{i=1}^{m}(\lambda-\mu_i)}{\prod_{i=1}^{m}(\lambda-\mu_i)\prod_{j=1}^{p}(\lambda-\gamma_j)} = -\frac{P_n(\lambda)}{P_m(\lambda)Q_p(\lambda)}, \qquad (5.3.20)$$

ou seja,

$$(a_{m+1}-\lambda)\frac{P_m(\lambda)P_n(\lambda)}{P_m(\lambda)Q_p(\lambda)} + \frac{\sum_{i=1}^{m} s_i^2 \prod_{\substack{k=1\\k\neq i}}^{m}(\lambda-\mu_k)Q_p(\lambda)}{P_m(\lambda)Q_p(\lambda)} +$$

$$+ \frac{\sum_{i=1}^{p} t_j^2 \prod_{\substack{k=1\\k\neq j}}^{p}(\lambda-\gamma_k)P_m(\lambda)}{P_m(\lambda)Q_p(\lambda)} = -\frac{P_n(\lambda)}{P_m(\lambda)Q_p(\lambda)} \qquad (5.3.21)$$

Logo temos

$$(a_{m+1}-\lambda)P_m(\lambda)Q_p(\lambda)+\sum_{i=1}^{m} s_i^2 \prod_{\substack{k=1\\k\neq i}}^{m}(\lambda-\mu_k)Q_p(\lambda)+\sum_{i=1}^{p} t_j^2 \prod_{\substack{k=1\\k\neq j}}^{p}(\lambda-\gamma_k)P_m(\lambda)=$$

$$= P_n(\lambda). \qquad (5.3.22)$$

Fazendo $\lambda = \mu_i$ em (5.3.22) obtemos que

$$s_i^2 = -\frac{P_n(\mu_i)}{P_m'(\mu_i)Q_p(\mu_i)}, \quad i=1,2,\ldots,m. \qquad (5.3.23)$$

Analogamente para $\lambda = \gamma_j$ temos

$$t_j^2 = -\frac{P_n(\gamma_j)}{P_m(\gamma_j)Q_p'(\gamma_j)}, \quad j=1,2,\ldots,p. \qquad (5.3.24)$$

De acordo com (5.2.11)

$$s_i^2 = b_m^2(\omega_i)_B, \qquad (5.3.25)$$
$$t_j^2 = b_{m+1}^2(\omega_j)_C.$$

Dessa forma b_m^2 e b_{m+1}^2 podem ser calculados, pois

$$\sum_{i=1}^{m} s_i^2 = \sum_{i=1}^{m} b_m^2(\omega_i)_B = b_m^2, \qquad (5.3.26)$$

onde $\sum_{i=1}^{m}(\omega_i)_B = 1$.

Da mesma forma

$$\sum_{j=1}^{p} t_j^2 = \sum_{j=1}^{p} b_{m+1}^2(\omega_j)_C = b_{m+1}^2. \qquad (5.3.27)$$

Feito isto, pelas equações (5.3.8) podemos obter $(y_m^{(i)})_1^m$ e $(z_1^{(i)})_1^p$.

De modo que B e C podem ser reconstruídos usando o algoritmo de Lanczos, conforme o capítulo 3 descreve.

Consideremos agora que existem r pares (j_q, k_q), com q=1,2,...,r, tal que

$$\mu_{jq} = \gamma_{kq} = \lambda_{lq} . \qquad (5.3.28)$$

Então

$$f(\lambda) \equiv (a_{m+1} - \lambda) + \sum_{i=1}^{m^*} \frac{s_i^2}{(\lambda - \mu_i)} + \sum_{i=1}^{p^*} \frac{t_i^2}{(\lambda - \gamma_i)} + \sum_{q=1}^{r} \frac{s_{jq}^2 + t_{kq}^2}{(\lambda - \mu_{jq})} \qquad (5.3.29)$$

tem como raízes os (n-r) autovalores λ_i não degenerados. Sendo que o * significa que foi omitido do somatório os r termos: $i = j_q$ (q = 1,2,...,r).

Assim, s_i e t_i são calculados como no caso anterior e os valores $s_{jq}^2 + t_{kq}^2$, dos modos degenerados, são calculados como na seção anterior, isto é,

$$W_q = - \frac{P_n'(\lambda_{l\,q})}{P_m'(\mu_{j\,q})Q_p'(\gamma_{k\,q})} = b_m^{\,2}(\omega_{j\,q})_B + b_{m+1}^{\,2}(\omega_{k\,q})_C = s_{j\,q}^{\,2} + t_{k\,q}^{\,2}. \qquad (5.3.30)$$

Escolhemos $\alpha_q \in (0, \pi/2)$ de forma que

$$\begin{aligned} s_{j\,q}^{\,2} &= W_q(\cos\alpha_q)^2, \\ t_{k\,q}^{\,2} &= W_q(\mathrm{sen}\,\alpha_q)^2. \end{aligned} \qquad (5.3.31)$$

Ficando assim $s_{j\,q}^{\,2}$ e $t_{k\,q}^{\,2}$ determinados $\forall\ q=1,2,\ldots,r$.

Das equações (5.3.26) e (5.3.27) obtemos $b_m^{\,2}$ e $b_{m+1}^{\,2}$ como funções dos parâmetros (α_q), mas observemos que $b_m^{\,2} + b_{m+1}^{\,2}$ é invariante.

Assim, novamente por (5.3.8) obtemos $(y_m^{(i)})_1^{\,m}$ e $(z_1^{(i)})_1^{\,p}$. Podemos então calcular B e C usando o algoritmo de Lanczos.

CAPITULO 6

APLICAÇÃO DOS MÉTODOS DE RECONSTRUÇÃO DE UMA MATRIZ DE JACOBI A UM SISTEMA VIBRACIONAL

6.1 - INTRODUÇÃO

Este capítulo trata de uma aplicação às reconstruções feitas até aqui. Descreveremos um modelo físico vibracional bastante simples que consiste de um conjunto de massas conectadas por molas.

Nas seções 6.1 e 6.2 serão feitas com detalhes as descrições da equação do movimento desse modelo e a relação linear existente entre a função entrada (função força) e a função saída (função resposta), respectivamente. Estas seções basear-se-ão nos autores [2], [3], [19] e [21].

A seção 6.3 tratará da reconstrução do sistema vibracional através dos pólos e zeros da função frequência resposta, sendo que esta será única quando a força, que consideraremos do tipo senoidal, for realizada em dos extremos do sistema. Neste caso os pólos produzirão os autovalores da matriz de Jacobi A a ser reconstruída,

enquanto os zeros corresponderão aos autovalores da submatriz principal direita ou esquerda de A de ordem n-1.

Os métodos numéricos que abordam este caso foram descritos nos capítulos 2, 3 e 4 , onde todos eles consideraram a submatriz principal de A como sendo a do lado esquerdo, isso corresponde ao caso em que a força é aplicada na última componente do sistema. Mas a reconstrução é análoga quando da aplicação da força na primeira componente.

Porém, quando for realizada uma força em um ponto interior do sistema, a reconstrução da matriz de Jacobi será única somente se o ponto de aplicação não é um nodo de algum automodo, isto é, se a aplicação não se dá numa posição em que a componente de um autovetor associado a autovalores de A for nula.

Nessas duas situações a reconstrução do sistema é feita usando os métodos descritos no capítulo 2 e 3, ou seja, através de polinômios ortogonais ou do algoritmo de Lanczos, com modificações adequadas.

A partir da matriz A procedemos a obtenção das matrizes de inércia e de rigidez do sistema vibracional em questão.

6.2 - OBTENÇÃO DA EQUAÇÃO DE MOVIMENTO DE UM SISTEMA MASSA-MOLA

Consideremos um sistema de vibração massa-mola bastante simples com n graus de liberdade, como mostra a figura 6.2.1. As n massas estão conectadas por molas lineares de rigidez $(k_r)_1^n$ e o conjunto se encontra em uma linha reta sobre uma superfície horizontal lisa. Denotemos por $(q_r)_1^n$ os deslocamentos realizados pelas n partículas e $(m_r)_1^n$ as massas das mesmas.

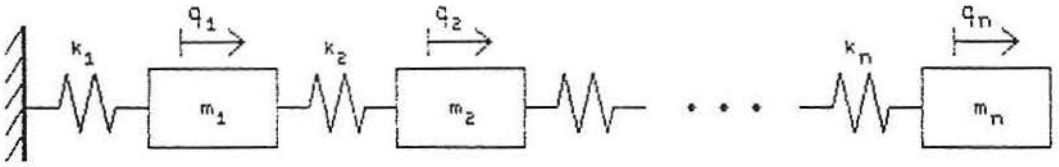

FIGURA 6.2.1 - n massas conectadas por molas

Um sistema mecânico está sujeito a dois tipos básicos de vibração: - *Vibração livre* que é o movimento periódico que se observa quando o sistema é deslocado da sua posição de equilíbrio estático. As forças atuantes são: força da mola, força de atrito e peso da partícula. Consideremos que no sistema acima não existe força de atrito e o peso das partículas não interfere. Portanto, a única força atuante nele, em termos de vibração livre, é a força da mola.

- *Vibração forçada* ocorre quando forças externas atuam sobre as partículas do sistema durante seu movimento de vibração. Denotemos as forças externas por

$[F_r(t)]_1^n$.

As equações de movimento para o sistema podem ser obtidas a partir das leis de movimento de Newton. Aplicando-as ao sistema temos

$$m_r\ddot{q}_r = F_r + \theta_{r+1} - \theta_r \qquad r=1,2,\ldots,n-1 \qquad (6.2.1)$$

$$m_n\ddot{q}_n = F_n - \theta_n \ , \qquad (6.2.2)$$

onde $\cdot$ denota diferenciação em relação ao tempo.

A lei de Hooke afirma que as forças das molas são dadas por

$$\theta_r = k_r(q_r - q_{r-1}) \ , \quad r=1,2,\ldots,n \ . \qquad (6.2.3)$$

O extremo esquerdo do sistema está fixo, então

$$q_0 = 0 \ . \qquad (6.2.4)$$

O sistema vibracional considerado tem bastante importância na Engenharia, embora seja o modelo discreto mais simples possível de uma haste vibrando longitudinalmente.

As equações (6.2.1) - (6.2.4) também podem descrever vibrações de torção de um sistema, como mostra a figura 6.2.2, onde q_r, k_r e m_r são interpretadas como rotação torsional, rigidez torsional e momento de inércia,

respectivamente. Tal sistema discreto proporciona um modelo simples para as vibrações de torção de uma haste com distribuição contínua de inércia e rigidez.

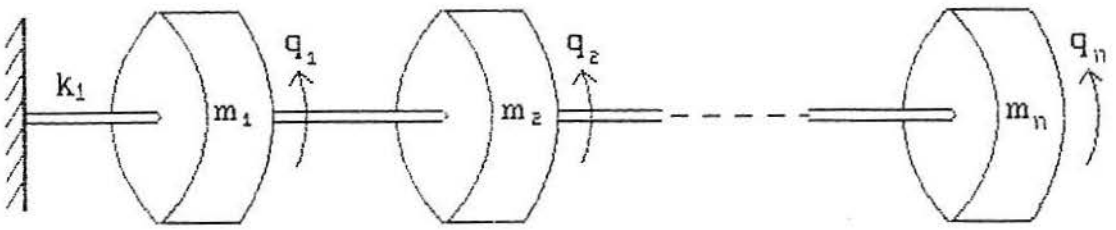

FIGURA 6.2.2 - Um sistema vibrando torsionalmente

Um terceiro sistema, o qual é matematicamente equivalente as equações (6.2.1) - (6.2.4), é o movimento transversal de um fio, como mostra a figura 6.2.3, que está esticado por uma tensão T e carregado por massas $(m_r)_1^n$. De acordo com a suposição de vibração infinitesimal, o fio parte com um afastamento muito pequeno da posição de equilíbrio, então a equação governante do movimento da massa m_r pode ser obtida pela consideração da figura 6.2.4 . A equação do movimento de Newton produz:

$$m_r \ddot{q}_r = F_r + T\,\mathrm{sen}\,\alpha_{r+1} - T\,\mathrm{sen}\,\alpha_r \ , \qquad (6.2.5)$$

$$m_r \ddot{q}_r = F_r + \theta_{r+1} - \theta_r \ , \qquad (6.2.6)$$

onde:

$$\theta_r = T\,\mathrm{sen}\,\alpha_r = k_r(q_r - q_{r-1}) \ , \ k_r = T/l_r \ . \qquad (6.2.7)$$

A analogia do extremo livre para o fio é que o último segmento deste está preso a um anel de massa desprezível, o qual pode deslisar sobre uma haste vertical lisa.

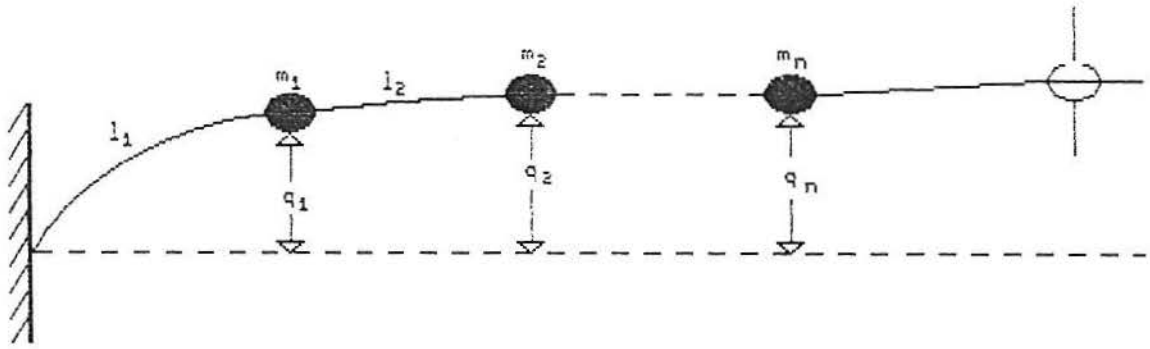

FIGURA 6.2.3 - n massas sobre um fio esticado

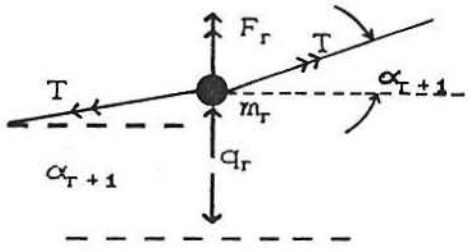

FIGURA 6.2.4 - As forças agindo sobre a massa m_r

Para expressar as equações (6.2.1) e (6.2.2) na forma matricial usamos (6.2.3) e obtemos

$$m_r\ddot{q}_r = F_r + k_{r+1}q_{r+1} - (k_{r+1}-k_r)q_r + k_r q_{r-1} \qquad (6.2.8)$$

$$m_n\ddot{q}_n = F_n - k_nq_n + k_nq_{n-1} \ , \qquad (6.2.9)$$

as quais produzem

$$\begin{bmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & & m_n \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \vdots \\ \ddot{q}_n \end{bmatrix} + \begin{bmatrix} k_1+k_2 & -k_2 & \cdots & 0 \\ -k_2 & k_2+k_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & -k_n \\ 0 & \cdots & -k_n & k_n \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix} . \qquad (6.2.10)$$

Podemos escrevê-la como

$$M\ddot{q} + Kq = F(t) \ , \qquad (6.2.11)$$

onde q é chamado vetor função resposta de deslocamentos e as matrizes M e K são denominadas matriz inércia e matriz rigidez do sistema, respectivamente. Observemos que ambas, M e K, são simétricas (esta é uma propriedade das matrizes correspondentes a qualquer sistema conservativo). Nesse sistema, em particular, a matriz M é diagonal enquanto K é tridiagonal e ambas são positivas-definidas(1).

6.3 - RELAÇÃO LINEAR ENTRE A FUNÇÃO FORÇA E A FUNÇÃO RESPOSTA

(1) - Ver referência [22].

Suponhamos que seja aplicada uma força senoidal $F(t) = F\,\mathrm{sen}\,\omega t$ na $(m+1)$-ésima massa do sistema $(m=0,1,\ldots,n-1)$.

Temos então

$$M\ddot{q} + Kq = F\,\mathrm{sen}\,\omega t\ , \tag{6.3.1}$$

com

$$F = \begin{bmatrix} 0 \\ \vdots \\ f_{m+1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{6.3.2}$$

Analisemos agora, a forma da solução de (6.3.1). A equação diferencial matricial dada em (6.3.1) é semelhante a equação diferencial ordinária

$$m\frac{d^2y}{dt^2} + ky = f\,\mathrm{sen}\,\omega t\ , \tag{6.3.3}$$

na função $y(t)$. Pode assim, ser demonstrado que a solução geral de (6.3.1) é formada pela soma de uma função complementar e de uma função particular. A função complementar é a solução geral da equação homogênea $M\ddot{q} + Kq = 0$, representando a vibração livre do sistema. Por sua vez, a função particular é uma solução qualquer da

equação (6.3.1), sendo que esta representa o movimento forçado do sistema. Procuramos então uma solução tendo a forma

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \text{sen } \omega t = X \text{sen } \omega t \quad , \qquad (6.3.4)$$

onde os x_r's são constantes e representam as amplitudes dos deslocamentos no movimento harmônico e a freqüência ω é a mesma da força externa.

Diferenciando (6.3.4), temos $\ddot{q} = -\omega^2 X \text{sen } \omega t$ e substituindo-a juntamente com (6.3.4) em (6.3.1), obtemos

$$(K-\omega^2 M)X = F \ . \qquad (6.3.5)$$

Sendo $(K-\omega^2 M)$ uma matriz não singular podemos escrevê-la como

$$X = (K-\omega^2 M)^{-1} F. \qquad (6.3.6)$$

Portanto, o vetor q é dado por

$$q(t) = (K-\omega^2 M)^{-1} F(t) \ . \qquad (6.3.7)$$

A equação acima expressa a relação linear

existente entre a entrada F(t) e a saída q(t). Seja

$$H(\omega) = (K-\omega^2 M)^{-1} \ , \qquad (6.3.8)$$

esta matriz, na literatura vibracional, é chamada matriz das funções freqüências respostas, onde cada $H_{i\,j}(\omega)$ corresponde a relação existente entre a entrada na posição j e saída na posição i .

De álgebra linear, temos

$$H(\omega) = \frac{adj(K-\omega^2 M)}{det(K-\omega^2 M)} \ . \qquad (6.3.9)$$

6.4 - RECONSTRUÇÃO DO SISTEMA VIBRACIONAL ATRAVÉS DOS PÓLOS E ZEROS DA FUNÇÃO FREQÜÊNCIA RESPOSTA

Como a força externa é aplicada apenas na (m+1)-ésima posição do sistema, analisaremos os pólos e zeros da função freqüência resposta $H_{m+1\,m+1}(\omega)$:

- Os pólos de $H_{m+1\,m+1}(\omega)$ são obtidos pela solução de $det(K-\omega^2 M) = 0$, cujos valores são exatamente os autovalores do problema generalizado de autovalor dado por

$$(K-\omega^2 M)q = 0 \ , \quad \text{com } q \neq 0 \ . \qquad (6.4.1)$$

- Os zeros, por sua vez, são obtidos a partir da

resolução de det $C_{m+1} = 0$, onde C_{m+1} é a matriz de ordem n-1 obtida apartir de $(K-\omega^2 M)$, tirando-se a (m+1)-linha e (m+1)-coluna. Mas tal solução é exatamente a mesma do problema generalizado de autovalor dado por

$$(K-\omega^2 M)q = 0 \quad , \text{ com } q_{m+1} = 0 \; . \tag{6.4.2}$$

Assim, as matrizes K e M podem ser reconstruídas a partir dos pólos e zeros da função freqüência resposta $H_{m+1\,m+1}(\omega)$ e, como vimos, estes são os autovalores dos problemas generalizados (6.4.1) e (6.4.2), respectivamente.

Reduziremos (6.4.1) a forma usual, escrevendo

$$\begin{aligned} M &= M^{1/2}M^{1/2} \\ u &= M^{1/2}q \\ \lambda &= \omega^2 \; , \end{aligned} \tag{6.4.3}$$

o problema de autovalor torna-se

$$(A-\lambda I)u = 0 \; , \text{ com } u \neq 0 \; , \tag{6.4.4}$$

onde

$$A = M^{-1/2}KM^{-1/2} \; . \tag{6.4.5}$$

Devido a forma particular das matrizes de inércia e rigidez, temos que A corresponderá a uma matriz de

Jacobi, conforme definida no capítulo 1. Além disso, será uma matriz positiva definida.

Quando for aplicada uma força nos extremos do sistema os pólos da função frequência resposta corresponderão aos autovalores da matriz A e seus zeros, por sua vez, serão os autovalores da submatriz principal de A de ordem n-1. Se a força for aplicada em q_1, teremos que estes serão os autovalores da submatriz principal esquerda, ou se for no outro extremo, q_n, serão autovalores da submatriz principal direita.

Com esses dois conjuntos de autovalores podemos reconstruir a matriz A unicamente, a menos de um fator escalar, através dos métodos vistos nos capítulos 2, 3 e 4, ou seja, utilizando polinômios ortogonais, o algoritmo de Lanczos ou transformações de Householder, respectivamente.

Se a força for aplicada na (m+1)-ésima coordenada, onde m=1,2,...,n-2, isto é, em um ponto interior, temos como anteriormente que os pólos da função frequência resposta corresponderão aos autovalores de A. Quanto aos zeros, sabemos que estes serão os autovalores do problema $(A-\lambda I)u = 0$, com $u_{m+1} = 0$.

Consideremos A sob a forma

$$A = \left[\begin{array}{c|c|c} B & -b_m & 0 \\ \hline -b_m & a_{m+1} & -b_{m+1} \\ \hline 0 & -b_{m+1} & C \end{array}\right], \qquad (6.4.6)$$

onde $B \in R^{m \times m}$ $C \in R^{n-m-1 \times n-m-1}$.

Substituindo-a na equação de autovalor obtemos

$$\left[\begin{array}{c|c|c} B - \lambda I & -b_m & 0 \\ \hline -b_m & a_{m+1} - \lambda & -b_{m+1} \\ \hline 0 & -b_{m+1} & C - \lambda I \end{array}\right]\left[\begin{array}{c} u' \\ 0 \\ u'' \end{array}\right] = 0, \qquad (6.4.7)$$

onde $u' \in R^{m \times 1}$ e $u'' \in R^{n-m-1 \times 1}$.

De acordo com (6.4.7) obtemos que

$$\begin{aligned} &(B-\lambda I)u' = 0, \\ &-b_m u'_m - b_{m+1} u''_1 = 0, \qquad (6.4.8) \\ &(C-\lambda I)u'' = 0. \end{aligned}$$

Assim os zeros da função frequência resposta produzem os autovalores do sistema forçado, isto é, eles corresponderão aos autovalores de um ou outro sistema (B ou C).

Diferentes maneiras de fixar os autovalores do sistema forçado B ou C conduzirão a diferentes soluções. Após terem sido escolhidos, temos como dados os autovalores

$(\lambda_i)_1^n$, $(\mu_i)_1^m$ e $(\gamma_i)_1^{n-m-1}$ de A, B e C, respectivamente. Sendo que tais conjuntos satisfazem as desigualdades

$$\begin{aligned} &\lambda_i \neq \lambda_j \ , \ \forall\ i \neq j, \\ &\mu_i \neq \mu_j \ , \ \forall\ i \neq j, \\ &\gamma_i \neq \gamma_j \ , \ \forall\ i \neq j, \end{aligned} \qquad (6.4.9)$$

pois A, B e C são matrizes de Jacobi.

Há dois casos possíveis:

a - O sistema forçado não tem nenhuma frequência natural dupla, isto é, os $(\mu_i)_1^m$ e $(\gamma_i)_1^{n-m-1}$ são distintos. Se eles forem arrumados em ordem crescente e reclassificados como $(\sigma_i)_1^{n-1}$, satisfazem

$$\lambda_i < \sigma_i < \lambda_{i+1} \ , \ \forall\ i=1,2,\ldots,n-1. \qquad (6.4.10)$$

Isso é equivalente a afirmar que nenhum autovetor $u^{(i)}$ de A tem um nodo em u_{m+1}, isto é,

$$u_{m+1}^{(i)} \neq 0 \ , \ \forall\ i=1,2,\ldots,n. \qquad (6.4.11)$$

b - Um ou mais pares são idênticos: suponhamos que μ_j e γ_k sejam idênticos. Assim $\mu_j = \gamma_k = \lambda_l$ e $u_{m+1}^{(l)} = 0$, onde $l = j + k$.

Neste caso, a reconstrução de A quando da aplicação de uma força no interior do sistema não é única e foi descrita no capítulo 5.

A seguir descreveremos a obtenção das matrizes K e M a partir da matriz A.

Consideremos o comportamento estático do sistema vibracional da figura 6.2.1 . Se uma força concentrada $f_1=k_1$ é aplicada ma massa m_1 então somente a mola k_1 será esticada e, por unidade. Como o extremo direito está livre todos os deslocamentos das massas serão unitários, isto é,

$$q = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} . \qquad (6.4.12)$$

Pela equação (6.2.10) temos que os deslocamentos são dados por

$$Kq = \begin{bmatrix} k_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} . \qquad (6.4.13)$$

Essa equação é equivalente a

$$Au = \begin{bmatrix} m_1^{-1/2}k_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} , \qquad (6.4.14)$$

e de acordo com (6.3.6) sua solução deve ser

$$u = M^{1/2}q = (m_1^{1/2}\ m_2^{1/2}\ \ldots\ m_n^{1/2})^T\ . \qquad (6.4.15)$$

Começando a partir de qualquer valor considerado para $u_n = m_n^{1/2}$, podemos resolver o conjunto tridiagonal de equações (6.3.8) sem conhecer o valor de $m_1^{-1/2}k_1$ e, assim, deduziremos os termos restantes $m_r^{1/2}$ (r=1,2,...,n-1). Se conhecida a massa total m podemos calcular todos os m_r.

Se os termos fora da diagonal principal de A forem negativos, os $m_i^{1/2}$ encontrados serão positivos e sendo estes conhecidos, a matriz de rigidez também o será, pois

$$K = M^{1/2}AM^{1/2}\ . \qquad (6.4.16)$$

CAPITULO 7

RESULTADOS NUMÉRICOS

7.1 - INTRODUÇÃO

Neste capítulo descreveremos alguns exemplos numéricos a fim de ilustrar os métodos descritos nos capítulos 2, 3, 4 e 5, e complementar a aplicação destes a um problema vibracional de molas, conforme apresentado no capítulo 6.

Nos exemplos que veremos a seguir usaremos os dados de um sistema conhecido como dados iniciais para o problema inverso, de modo que o sistema reconstruído possa ser comparado com os resultados corretos.

Consideraremos, por simplicidade, nos exemplos das seções 7.3 e 7.4, que os sistemas massa-mola possuem as massas e os coeficientes de rigidez com magnitude um.

Todas as implementações utilizadas foram realizadas num microcomputador IBM-PC AT 286, onde o algoritmo que faz a reconstrução de uma matriz de Jacobi através de polinômios ortogonais foi implementada na

linguagem FORTRAN e no software MATLAB. Os outros dois métodos que fazem esta mesma reconstrução foram realizados somente no MATLAB.

Na seção 7.4 os dois exemplos ilustram o capítulo 5.

7.2 - COMPARAÇÃO DOS RESULTADOS NUMÉRICOS DOS DIFERENTES MÉTODOS

Mostraremos nesta seção a reconstrução de uma matriz de Jacobi da forma

$$A = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & -1 & 2 \end{bmatrix} \qquad (7.2.1)$$

através dos três métodos estudados nos capítulos 2, 3, e 4. Analisaremos os resultados para matrizes de ordem 25, 50 e 75.

Os autovalores de uma matriz desse tipo são dados por

$$\lambda_i = 2(1 + \cos\theta_i), \quad i=1,2,\ldots,n \qquad (7.2.2)$$

onde

$$\theta_i = \frac{i\pi}{n+1} . \qquad (7.2.3)$$

E os autovalores da submatriz principal de ordem n-1 de A são dados também pela fórmula (7.2.2) mas com

$$\theta_i = \frac{i\pi}{n} , \text{ com } i = 1,2,\ldots,n-1. \qquad (7.2.4)$$

Realizaremos a reconstrução a partir dessas duas sequências de autovalores utilizando as implementações A1, A2 e A3 do apêndice A, que executam a reconstrução através de polinômios ortogonais, algoritmo de Lanczos e transformações de Householder, respectivamente. Tais implementações foram realizadas através do software PC-MATLAB, com quatorze casas decimais.

Os algoritmos fornecem aproximações para os valores das diagonais principais e os valores fora dela, cujo erro máximo e erro médio são mostrados nas tabelas 7.2.2, 7.2.3 e 7.2.4.

Para a reconstrução da matriz de Jacobi (7.2.1) através de transformações de Householder tivemos que fazer uma escolha adequada dos sinais das componentes da última coluna da matriz simétrica B, da forma (4.2.3), para as diferentes ordens. Na tabela 7.2.1 constam as posições na última coluna de B que tiveram de assumir valor negativo.

Estas são as modificações necessárias na implementação A3, pois caso contrário após a execução teremos apenas reconstruído uma matriz tridiagonal com os mesmos valores da matriz de Jacobi desejada, em valor absoluto, mas com sinais diferentes nas posições fora da diagonal principal.

TABELA 7.2.1

n	Linhas da última coluna de B com componentes negativas
25	2,3,4,7,9,10,14,16,17,18,20,21,23
50	1,2,3,4,6,7,9,10,12,13,18,19,20,21,29,31,32,33,37,40,
	41,43,44,46,48
75	1,4,5,8,11,12,15,18,21,22,23,24,25,27,28,29,30,33,45,
	46,47,48,52,53,54,56,57,58,60,61,64,66,69,71,73

TABELA 7.2.2

ERRO MAXIMO E ERRO MÉDIO NOS VALORES DA DIAGONAL PRINCIPAL E FORA DELA QUANDO A RECONSTRUÇÃO É FEITA ATRAVÉS DE POLINOMIOS ORTOGONAIS

n	Diagonal		Fora Diagonal		Flops
	Erro Máx.	Erro Méd.	Erro Máx.	Erro Méd.	
25	1.0×10^{-14}	1.0×10^{-14}	———	———	11130
50	2.0×10^{-14}	1.125×10^{-14}	1.0×10^{-14}	1.0×10^{-14}	54943
75	2.0×10^{-14}	1.348×10^{-14}	1.0×10^{-14}	1.0×10^{-14}	97180

TABELA 7.2.3

ERRO MAXIMO E ERRO MÉDIO NOS VALORES DA DIAGONAL PRINCIPAL E FORA DELA QUANDO A RECONSTRUÇÃO É FEITA ATRAVÉS DO ALGORITMO DE LANCZOS

n	Diagonal		Fora Diagonal		Flops
	Erro Máx.	Erro Méd.	Erro Máx.	Erro Méd.	
25	1.0×10^{-14}	1.0×10^{-14}	———	———	72442
50	1.0×10^{-14}	1.0×10^{-14}	———	———	610024
75	2.0×10^{-14}	1.185×10^{-14}	1.0×10^{-14}	1.0×10^{-14}	2380256

TABELA 7.2.4

ERRO MAXIMO E ERRO MÉDIO NOS VALORES DA DIAGONAL PRINCIPAL E FORA DELA QUANDO A RECONSTRUÇÃO É FEITA ATRAVÉS DE TRANSFORMAÇÕES HOUSEHOLDER

n	Diagonal		Fora Diagonal		Flops
	Erro Máx.	Erro Méd.	Erro Máx.	Erro Méd.	
25	2.0×10^{-14}	1.125×10^{-14}	1.0×10^{-14}	1.0×10^{-14}	1486988
50	3.0×10^{-14}	1.125×10^{-14}	1.0×10^{-14}	1.0×10^{-14}	24387742
75	3.0×10^{-14}	1.444×10^{-14}	3.0×10^{-14}	1.129×10^{-14}	124474603

O que se observa é que trabalhando com o PC-MATLAB, com quinze dígitos significativos, o erro de

execução fica na décima quarta casa decimal, portanto muito pequeno. Para matrizes de pequeno porte, até ordem 20, praticamente o erro não existe nas diagonais principais nem fora delas.

Quando a reconstrução é feita através das transformações de Householder percebe-se que o erro máximo e o erro médio são maiores do que os outros métodos. Fazendo a reconstrução através dessa rotina temos a desvantagem do tempo de execução que é bem maior do que os outros dois métodos levam para reconstruir a matriz. Isso pode-se observar pelo número de operações realizadas em cada um conforme as tabelas anteriores especificam.

Nas três implementações constam o cálculo do erro máximo e do erro médio nas diagonais e fora delas. Mas os resultados fornecidos pelo programa não puderam ser considerados aqui, pois o PC-MATLAB nos dá esses erros considerando os valores antes da saída para quatorze casas decimais, onde ocorre arredondamento. Dessa forma as três tabelas foram montadas com base nos resultados apresentados pelo computador considerando quatorze casas decimais.

A rotina PC-MATLAB trabalha com o cálculo de matrizes que possuam no máximo 8088 elementos. Assim as implementações feitas no PC-MATLAB podem realizar cálculos no máximo com matrizes quadradas de ordem 89.

7.3 - EXEMPLO NUMÉRICO UTILIZANDO A LINGUAGEM FORTRAN

Implementamos na linguagem FORTRAN o método de reconstrução de uma matriz de Jacobi da forma (7.2.1) através de polinômios ortogonais, que corresponde ao programa A4 do apêndice A. Trabalhamos neste exemplo com precisão simples e com sete casas decimais.

A tabela abaixo mostra o cálculo do erro máximo e erro médio para matrizes de ordem 25, 50, 75, 100 e 125.

TABELA 7.3.1

ERRO MÁXIMO E ERRO MÉDIO NOS VALORES DA DIAGONAL PRINCIPAL E FORA DELA QUANDO A RECONSTRUÇÃO É FEITA ATRAVÉS DE POLINÔMIOS ORTOGONAIS UTILIZANDO A LINGUAGEM FORTRAN

n	Diagonal		Fora Diagonal	
	Erro Máx.	Erro Méd.	Erro Máx.	Erro Méd.
25	4.0×10^{-6}	1.88×10^{-6}	1.0×10^{-6}	3.68×10^{-7}
50	7.0×10^{-6}	2.5×10^{-6}	3.0×10^{-6}	1.052×10^{-6}
75	9.0×10^{-6}	2.4×10^{-6}	4.0×10^{-6}	1.54×10^{-6}
100	9.0×10^{-6}	2.97×10^{-6}	5.8×10^{-6}	2.138×10^{-6}
125	2.2×10^{-5}	8.792×10^{-6}	1.0×10^{-5}	3.804×10^{-6}

Num primeiro momento tentamos implementar essa reconstrução utilizando dupla precisão e trabalhando com 14

casas decimais, o mesmo número de casas usadas no PC-MATLAB do exemplo da seção 7.2. Verificamos que a saída dos autovalores da matriz a ser reconstruiída e dos autovalores de sua submatriz principal esquerda eram iguais aos autovalores do exemplo anterior somente até a sexta casa decimal na maioria dos dados, fazendo com que a matriz obtida no final da reconstrução tivesse erros a partir da quinta casa decimal. Observamos que esta diferença ocorria devido a forma interna do cálculo da função cosseno na linguagem fortran em relação ao software PC-MATLAB. Um senão nesta dificuldade é que como estamos usando dados iniciais de um problema inverso já conhecido, de forma a comparar os resultados obtidos com este, o cálculo dos autovalores é feito pela fórmula (7.2.2) mas na prática as duas sequências de autovalores são realmente dados iniciais do problema.

Como podemos ver na tabela 7.3.1, quando trabalhamos com sete casas decimais o erro se reduz consideravelmente.

Esta implementação na linguagem FORTRAN tem a vantagem do fator tempo, pois mesmo para matrizes de ordem 100 ou maiores o cálculo é praticamente imediato. Além disso, temos a possibilidade de escolha do número de casas decimais. Nesta linguagem também podemos realizar a reconstrução de matrizes de ordem maiores do que o PC-MATLAB pode realizar.

No exemplo nos limitamos ao cálculo de matrizes de ordem 125, pois para matrizes de ordem maiores

do que 130 ocorre overflow. Isso devido ao fato de que com sete casas decimais o cálculo dos autovalores através da fórmula (7.2.1) produz valores repetidos, o que não pode ocorrer.

7.4 - EXEMPLOS NUMERICOS APLICADOS A UM MODELO VIBRACIONAL

Consideremos como primeiro exemplo um sistema vibracional massa-mola, conforme descrito no capítulo 6, de ordem 16 cujas massas tem magnitude 1 e as molas possuem rigidez unitária. Suponhamos que seja aplicada uma força na sétima posição provocando um constrangimento para que nesta posição não ocorra deslocamento.

Como novamente usaremos dados de um sistema conhecido como dados iniciais para o problema inverso a fim de que o sistema reconstruído possa ser comparado com os resultados corretos. Descreveremos a seguir a obtenção dos autovalores de A e de suas submatrizes principais esquerda e direita, B e C respectivamente, cujas ordens são m=6 e p=9.

Sabemos do capítulo 6 que os autovalores de A correspondem as frequências naturais do sistema vibracional. Assim a equação (6.2.10) torna-se

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & & 1 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \vdots \\ \ddot{q}_n \end{bmatrix} + \begin{bmatrix} 2 & -1 & & 0 \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 1 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \equiv 0. \tag{7.4.1}$$

Neste caso a equação de autovalor (6.4.4) pode ser escrita como

$$\begin{bmatrix} 2-\lambda & -1 & & 0 \\ -1 & 2-\lambda & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 1-\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \equiv 0, \tag{7.4.2}$$

onde $\lambda = \omega^2$, pois $m \equiv k \equiv 1$.

Observemos que as linhas de (7.4.2) satisfazem a recorrência

$$-x_{r-1} + (2-\lambda)x_r - x_{r+1} = 0 \ , \ r=1,2,\ldots n. \tag{7.4.3}$$

A primeira das equações de (7.4.2) pode ser escrita nesta forma desde que $x_0 = 0$. Da mesma maneira, a última das equações de (7.4.2) pode ser escrita na forma (7.4.3), desde que $x_{n+1} = x_n$.

Assim, as condições finais para a recorrência são

$$x_0 = x_{n+1} - x_n. \tag{7.4.4}$$

A fórmula de recorrência tem como solução geral

$$x_r = A\cos r\theta + B\operatorname{sen} r\theta. \qquad (7.4.5)$$

Substituindo-a em (7.4.3) encontramos que θ deve satisfazer

$$\begin{aligned} \cos(r-1)\theta + \cos(r+1)\theta &= 2\cos\theta\cos r\theta = (2-\lambda)\cos r\theta \\ \operatorname{sen}(r-1)\theta + \operatorname{sen}(r+1)\theta &= 2\cos\theta\operatorname{sen} r\theta = (2-\lambda)\operatorname{sen} r\theta, \end{aligned} \qquad (7.4.6)$$

isto é,

$$2\cos\theta = 2 - \lambda. \qquad (7.4.7)$$

As condições finais serão satisfeitas se e somente se,

$$A = 0 = \operatorname{sen}((n+1)\theta) - \operatorname{sen}(n\theta) = 2\cos((n+1/2)\theta)\operatorname{sen}(\theta/2). \qquad (7.4.8)$$

De modo que os valores possíveis de θ são

$$\theta = \theta_k = \frac{(2k-1)\pi}{2n+1}, \quad k=1,2,\ldots,n \qquad (7.4.9)$$

cujos valores correspondentes de λ são

$$\lambda_k = 2 - 2\cos\theta_k \ . \qquad (7.4.10)$$

Os autovalores de B são obtidos a partir da mesma análise para os autovalores de A, mas com a restrição de que $x_{m+1} = 0$ (m+1 corresponde a posição em que a força foi aplicada). Assim podemos considerar que a m+1-ésima massa está fixa em vez de livre. Desse modo,

$$x_{m+1} = A\cos((m+1)\theta) + B\mathrm{sen}((m+1)\theta). \qquad (7.4.11)$$

Como $x_0 = 0$ implica que A = 0 temos que

$$\theta_k = \frac{k\pi}{m+1}, \quad k = 1,2,\ldots,m. \qquad (7.4.12)$$

Os autovalores de C são obtidos pela mesma fórmula dos autovalores de A, pois as matrizes A e C possuem a mesma forma, sua diferença esta na ordem. Logo

$$\theta_k = \frac{(2k-1)\pi}{2p+1}, \quad k=1,2,\ldots,p. \qquad (7.4.13)$$

As implementações dos dois métodos que resolvem este caso encontram-se no apêndice A e são A5 e A6. Para esta reconstrução usaremos A6.

Assim para quatorze casas decimais os dados desejados são dados por

$\{\lambda_i\}_1^n$	$\{\mu_k\}_1^m$	$\{\gamma_i\}_1^p$
0.009056154853 83	0.19806226419516	0.02727739319456
0.081014052771 01	0.753020396282 53	0.241052497587 02
0.222329102690 15	1.554958132087 37	0.645436856748 52
0.427893810514 43	2.445041867912 63	1.196609150694 06
0.690278532109 43	3.246979603717 47	1.834841309055 34
1.000000000000 00	3.801937735804 84	2.490970974281 60
1.345864073365 16		3.093896316244 85
1.715370323453 43		3.578281018792 79
2.095163831647 49		3.891634483401 27
2.471517871018 86		
2.830830026003 77		
3.160113819142 40		
3.447468076210 14		
3.682507065662 36		
3.856735866032 15		
3.963857394525 41		

Como os autovalores dos três espectros são todos distintos a reconstrução do sistema é única.

A matriz obtida é dada por

$$A = \begin{bmatrix} 2 & -1 & & & & & & & & & & & \\ -1 & 2 & -1 & & & & & & & & & & \\ & -1 & 2 & -1 & & & & & & & & & \\ & & -1 & 2 & -1 & & & & & & & & \\ & & & -1 & 2 & -1 & & & & & & & \\ & & & & -1 & 2 & -1 & & & & & & \\ & & & & & -1 & 2+\varepsilon & -1 & & & & & \\ & & & & & & -1 & 2 & -1 & & & & \\ & & & & & & & -1 & 2 & -1 & & & \\ & & & & & & & & -1 & 2-\varepsilon & -1 & & \\ & & & & & & & & & -1 & 2+\varepsilon & -1 & \\ & & & & & & & & & & -1 & 2-\varepsilon & -1 \\ & & & & & & & & & & & -1 & 2+\varepsilon & -1 \\ & & & & & & & & & & & & -1 & 1-\varepsilon \end{bmatrix}, \tag{7.4.14}$$

onde $\varepsilon = 1.0 \times 10^{-14}$.

Através do PC-MATLAB verificamos que os autovalores de (7.4.14) são os mesmos dos dados iniciais.

Utilizando a implementação A7 que calcula as matrizes de inércia e de rigidez do sistema a partir da matriz de Jacobi A.

Obtemos os seguintes resultados:

$\langle m_i \rangle_1^n$	$1+15\varepsilon$	$1+13\varepsilon$	$1+11\varepsilon$	$1+9\varepsilon$	$1+7\varepsilon$	$1+5\varepsilon$	$1+2\varepsilon$	$1+2\varepsilon$
	$1+2\varepsilon$	$1+2\varepsilon$	1	1	$1-2\varepsilon$	$1-2\varepsilon$	$1-2\varepsilon$	1
$\langle k_i \rangle_1^n$	$1+16\varepsilon$	$1+14\varepsilon$	$1+12\varepsilon$	$1+10\varepsilon$	$1+8\varepsilon$	$1+6\varepsilon$	$1+4\varepsilon$	$1+2\varepsilon$
	$1+2\varepsilon$	$1+2\varepsilon$	$1+1\varepsilon$	1	$1-\varepsilon$	$1-2\varepsilon$	$1-2\varepsilon$	$1-\varepsilon$

Quando consideramos o cálculo das matrizes inércia e rigidez tomando os valores de A diretamente da implementação obtemos os seguintes valores para as componentes da matriz de inércia e de rigidez:

$\langle m_i \rangle_1^n$	$1-\varepsilon$	$1-2\varepsilon$	$1-2\varepsilon$	$1-3\varepsilon$	$1-4\varepsilon$	$1-5\varepsilon$	$1-5\varepsilon$	$1-5\varepsilon$
	$1-4\varepsilon$	$1-3\varepsilon$	$1-3\varepsilon$	$1-2\varepsilon$	$1-3\varepsilon$	$1-3\varepsilon$	$1-3\varepsilon$	1
$\langle k_i \rangle_1^n$	$1-\varepsilon$	$1-\varepsilon$	$1-2\varepsilon$	$1-3\varepsilon$	$1-3\varepsilon$	$1-4\varepsilon$	$1-5\varepsilon$	$1-5\varepsilon$
	$1-4\varepsilon$	$1-4\varepsilon$	$1-4\varepsilon$	$1-3\varepsilon$	$1-3\varepsilon$	$1-2\varepsilon$	$1-\varepsilon$	$1-\varepsilon$

Isto ocorre porque o software realizará os cálculos internamente considerando mais casas decimais, fazendo com que estes sejam diferentes dos valores esperados a partir da matriz (7.4.14).

Como um segundo exemplo consideremos um sistema de ordem 10 com massas unitárias unidas por 10 molas com rigidez também unitária. Se a sexta massa é constrangida para ter deslocamento zero, então m = 5 e p = 4.

Os valores espectrais com quatorze casas decimais são dados por

$\{\lambda_i\}_1^n$	$\{\mu_k\}_1^m$	$\{\gamma_i\}_1^p$
0.02233834754974	0.26794919243112	0.12061475842818
0.19806226419516	1.00000000000000	1.00000000000000
0.53389625634035	2.00000000000000	2.34729635533386
1.00000000000000	3.00000000000000	3.53208888623796
1.55495813208737	3.73205080756888	
2.14946018717285		
2.73068204873279		
3.24697960371747		
3.65247754863199		
3.91114561157228		

Como podemos observar existe um par de autovalores comuns nos espectros de B e C, acarretando assim a igualdade com um autovalor de A, ou seja, $\mu_2 = \gamma_2 = \lambda_4$.

Neste caso a reconstrução não é única. Existe uma família de soluções.

Como sabemos que b_m e b_{m+1} deverão ser um, o ângulo α deverá assumir um valor de forma que $(\cos\alpha)^2 = 3/7$.

A matriz de Jacobi encontrada usando a implementação A5 produz

$$A = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & & 0 \\ 0 & & \ddots & & \vdots \\ \vdots & & & 2 & -1 & 0 \\ 0 & & & -1 & 2 & -1 \\ 0 & & \cdots & 0 & -1 & 1 \end{bmatrix}. \qquad (7.4.15)$$

Observamos que a implementação produz exatamente a matriz desejada, com saída de 14 casas decimais. Consequentemente as matrizes inércia e de rigidez também serão as esperadas.

Fazendo o cálculo dessas duas matrizes considerando os valores internos no MATLAB obtemos

$\langle m_i \rangle_1^n$	$1+4\varepsilon \quad 1+3\varepsilon \quad 1+2\varepsilon \quad 1+2\varepsilon \quad 1+\varepsilon \quad 1 \quad 1 \quad 1 \quad 1 \quad 1$
$\langle k_i \rangle_1^n$	$1+4\varepsilon \quad 1+4\varepsilon \quad 1+3\varepsilon \quad 1+2\varepsilon \quad 1+\varepsilon \quad 1 \quad 1 \quad 1 \quad 1 \quad 1$

No exemplo acima valores diferentes para o parâmetro α produzem sistemas de soluções diferentes, as quais entretanto, possuem os valores espectrais desejados.

Como uma ilustração final consideremos $(\cos\alpha)^2 = 19/20$ e o sistema reconstruído a partir deste valor para α tem a forma:

Valores da diagonal principal	Valores da codiagonal
2.159041394335 51	-0.98727191537479
1.84095860566449	-1.09046740940959
2.00000000000000	-0.90048921649043
2.23322683706070	-0.97242235807012
1.76677316293930	-1.14200116754173
2.00000000000000	-0.83416625041615
2.43712574850299	-0.89940040026436
1.56287425149700	-1.33288999721689
2.00000000000000	-0.47265659343666
1.00000000000000	

Para este sistema verificamos que os autovalores da matriz reconstruída realmente coincidem com os valores dados.

Neste caso as massas e os coeficientes de rigidez encontrados usando os valores internos da matriz reconstruida são dados por

$\langle mi \rangle_1^{10}$	$\langle k_i \rangle_1^{10}$
3.86197636949516	3.86197636949515
5.32272390821612	4.47619047619047
4.47619047619046	5.32272390821611
3.62965704416479	3.62965704416480
5.83769841269837	4.47619047619044
4.47619047619043	5.83769841269836
3.11468253968252	3.11468253968251
7.95238095238093	4.47619047619046
4.47619047619046	7.95238095238092
1.00000000000000	1.00000000000000

APÊNDICE A

IMPLEMENTACAO A1

```
echo on
clear
clc

%        DISSERTACAO DE MESTRADO DE INES FERREIRA MORAES
%                        UFRGS  -  FEV 93
%
%        Este programa faz  a  reconstrucao  de  uma  Matriz
%    Jacobi atraves de polinomios  ortogonais, sendo  que  a
%    Matriz original e formada por 2 nas diagonais e -1  nas
%    outras duas diagonais adjacentes.

pause %  Pressione qualquer tecla para continuar
clc

%        Obtencao  dos  autovalores  da  matriz   A   a   ser
%    reconstruida  e  dos  autovalores  de  sua   submatriz
%    principal esquerda de ordem N-1.

echo off
N = input('ordem da Matriz = ');
format long;
V = zeros(N,1);
U = zeros(N-1,1);
LB = zeros(N,1);
MU = zeros(N-1,1);
```

```
for i = 1:N-1;
V(i) = 2 + 2*cos(i*pi/(N+1));
U(i) = 2 + 2*cos(i*pi/N);
end;
V(N) = 2 + 2*cos(N*pi/(N+1));
LB = sort(V);
MU = sort(U);

echo on
LB    %  Autovalores de A

pause %  Pressione qualquer tecla para continuar
clc

MU    %  Autovalores da submatriz principal de A

pause %  Pressione qualquer tecla para continuar
clc

%        Fazendo a reconstrucao da Matriz de Jacobi.
%        Aguarde...

echo off
%        Calculo dos pesos Wi a partir de LB e MU

W = zeros(N,1);
C = ones(N,N);
D = ones(N,N);
NUM = ones(1,N-1);
DEN = ones(1,N-1);
for k = 1:N;
  for i = 1:N-1;
  C(i,k) = LB(k) - MU(i);
    if abs(i-k) > 0;
    D(i,k) = LB(k) - LB(i);
    end;
```

```
  end;
  if N-k > 0;
  DC N,k) = LBCk) - LBCN);
  end;
end;
for k = 1:N;
NUMC:,k) = prodCCC:,k));
DENC:,k) = prodCDC:,k));
WCk,1) = NUMC:,k)/DENC:,k);
end;

%        Os valores dos dois primeiros polinomios sao
%    gerados a partir de LB (PK e P) e BC1,1) e obtido

AX = zerosCN,1);
P = zerosCN,1);
PK = onesCN,1);
B = zerosCN,N);
for k = 1:N;
AXCk) = WCk)*LBCk);
end;
BC1,1) = sumCAX)/sumCW);
for j = 1:N;
PCj) = LBCj) - BC1,1);
end;

%        Calculo dos outros polinomios e dos elementos nao
%    nulos da matriz B (reflexao de A sobre a segunda
%    diagonal)

PL = zerosCN,1);
PM = zerosCN,1);
Q = zerosCN-1,1);
A = zerosCN,N);
S = sumCW);
for i = 2:N;
```

```
  for k = 1:N;
  PL(k) = W(k)*P(k)^2;
  PM(k) = LB(k)*PL(k);
  end;
 Q(i-1) = sum(PL)/S;
 B(i,i) = sum(PM)/sum(PL);
 S = sum(PL);
  for j = 1:N;
  PL(j) = P(j);
  P(j) = (LB(j)-B(i,i))*PL(j) - Q(i-1)*PK(j);
  PK(j) = PL(j);
  end;
end;
for k = 2:N;
B(k-1,k) = - sqrt(Q(k-1));
end;

%         Obtencao da matriz A a partir de B

A(N,N) = B(1,1);
for k = 2:N;
A(N+1-k,N+1-k) = B(k,k);
A(N+1-k,N-k+2) = B(k-1,k);
A(N-k+2,N+1-k) = A(N+1-k,N-k+2);
end;

echo on
flops
echo off

%         A matriz C(3,N) representa os elementos nao  nulos
%    da matriz de Jacobi

C = zeros(3,N);
C(1,1) = A(1,1);
C(2,1) = A(2,1);
```

```
for k = 2:N-1;
  for i = 1:3;
  C(i,k) = A(i+k-2,k);
  end;
end;
C(2,N) = A(N-1,N);
C(3,N) = A(N,N);
echo on
%        Reconstrucao da Matriz de Jacobi concluida.
pause %  Pressione qualquer tecla para continuar
clc
A

pause % Pressione qualquer tecla para continuar
clc
C

pause %  Pressione qualquer tecla para continuar
clc

%        Calculo do erro maximo e do erro medio cometidos na
%    diagonal principal. Aguarde...

echo off
E = zeros(N,1);
F = zeros(N,1);
for i = 1:N;
  if A(i,i) ~= 2;
  E(i) = abs(A(i,i) - 2);
  end;
end;
F = sort(E);
g = length(find(F));
ERROMAXIDIAG = max(F);
ERROMEDIODIAG = mean(F(N-g+1:N,1));
echo on
```

```
%         Calculo concluido.

pause %  Pressione qualquer tecla para continuar

clc
ERROMAXIDIAG
ERROMEDIODIAG
pause %  Pressione qualquer tecla para continuar
clc

%         Calculo do erro maximo e do  erro  medio  cometidos
%     nas duas diagonais adjacentes a diagonal principal.
%         Aguarde...

echo off
L = zeros(N-1,1);
M = zeros(N-1,1);
for i = 1:N-1;
  if A(i,i+1) ~= -1;
  L(i) = abs(A(i,i+1) + 1);
  end;
end;
M = sort(L);
h = length(find(M));
ERROMAXFORADIAG = max(M);
ERROMEDIOFORADIAG = mean(M(N-h:N-1,1));

echo on
%         Calculo concluido.

pause %  Pressione qualquer tecla para continuar
clc

ERROMAXFORADIAG
ERROMEDIOFORADIAG
```

IMPLEMENTACAO A2

```
echo on
clear
clc

%    DISSERTACAO DE MESTRADO DE INES FERREIRA MORAES
%                  UFRGS  -  FEV 93
%
%    Este programa faz  a  reconstrucao  de  uma  Matriz  de
%    Jacobi  de  ordem  n,  cujos  elementos  da  diagonal
%    principal  devem  ter  valor  2  e  os  das  diagonais
%    adjacentes valor -1. A reconstrucao e feita  a  partir
%    de dados espectrais utilizando o Algoritmo de Lanczos

pause % Pressione qualquer tecla para continuar
clc

%     Obtencao dos Autovalores da matriz a ser  reconstruida
%     e dos autovalores da sua submatriz principal esquerda

echo off
N = input('Ordem da Matriz = ');
format long;
V = zeros(N,1);
U = zeros(N-1,1);
LB = zeros(N,N);
MU = zeros(N-1,N-1);
AX = zeros(N);
BX = zeros(N);
for i= 1:N-1;
```

```
V(i,1) = 2 + 2*cos(i*pi/(N+1));
U(i,1) = 2 + 2*cos(i*pi/N);
end;
V(N) = 2 + 2*cos(N*pi/(N+1));
AX = sort(V);
BX = sort(U);
LB = diag(AX);
MU = diag(BX);

echo on
AX     % Autovalores de A (LB)

pause % Pressione qualquer tecla para continuar
clc
BX     % Autovalores da submatriz principal de A (MU)

pause % Pressione qualquer tecla para continuar
clc

%        Fazendo a renconstrucao da Matriz de Jacobi.
%        Aguarde...

echo off
clear U;
clear V;
X = zeros(N,1);
C = ones(N,N);
D = ones(N,N);
NUM = ones(1,N-1);
DEN = ones(1,N-1);
for k = 1:N;
  for i = 1:N-1;
  C(i,k) = MU(i,i) - LB(k,k);
    if abs(i-k) > 0;
    D(i,k) = LB(i,i) - LB(k,k);
    end;
```

```
  end;
  if N-k > 0;
  DCN,k) = LBCN,N) - LBCk,k);
  end;
end;
for k = 1:N;
NUMC:,k) = prodCCC:,k));
DENC:,k) = prodCDC:,k));
UDCk,1) = NUMC:,k)/DENC:,k);
XCk,1) = sqrtCUDCk,1));
end;
clear C;
clear D;
A = zerosCN,N);
R = zerosCN,N);
XNC:,N) = XC:,:);
for i = 1:N-1;
ACN-i+1,N-i+1) = XNC:,N-i+1)'*LB*XNC:,N-i+1);
RC:,N-i+1) = ACN-i+1,N-i+1)*XNC:,N-i+1) - LB*XNC:,N-i+1);
  if i > 1
  RC:,N-i+1) = RC:,N-i+1) - ACN-i+1,N-i+2)*XNC:,N-i+2);
  end;
ACN-i,N-i+1) = normCRC:,N-i+1));
XNC:,N-i) = RC:,N-i+1)/ACN-i,N-i+1);
end;
AC1,1) = XNC:,1)'*LB*XNC:,1);
for i = 1:N-1;
ACN-i,N-i+1) = - ACN-i,N-i+1);
ACN-i+1,N-i) = ACN-i,N-i+1);
end;

echo on
flops
echo off

%       A matriz CC3,N) representa os elementos nao nulos
```

```
%   da matriz de Jacobi
C = zeros(3,N);
C(1,1) = A(1,1);
C(2,1) = A(2,1);
for k = 2:N-1;
  for i = 1:3;
  C(i,k) = A(i+k-2,k);
  end;
end;
C(2,N) = A(N-1,N);
C(3,N) = A(N,N);

echo on
%       Reconstrucao da Matriz de Jacobi concluida.

pause % Pressione qualquer tecla para continuar
clc
A

pause %  Pressione qualquer tecla para continuar
clc
C

pause %  Pressione qualquer tecla para continuar
clc

%       Calculo do erro maximo e do erro medio cometidos na
%    diagonal principal. Aguarde...

echo off
E = zeros(N,1);
F = zeros(N,1);
for i = 1:N;
  if A(i,i) ~= 2;
  E(i) = abs(A(i,i) - 2);
  end;
```

```
end;
F = sort(E);
g = length(find(F));
ERROMAXIDIAG = max(F);
ERROMEDIODIAG = mean(F(N-g+1:N,1));

echo on
%       Calculo concluido.

pause %  Pressione qualquer tecla para continuar
clc
ERROMAXIDIAG
ERROMEDIODIAG

pause %  Pressione qualquer tecla para continuar
clc

%         Calculo do erro maximo e do  erro  medio  cometidos
%     nas duas diagonais adjacentes a diagonal principal.
%         Aguarde...

echo off
L = zeros(N-1,1);
M = zeros(N-1,1);
for i = 1:N-1;
  if A(i,i+1) ~= -1;
  L(i) = abs(A(i,i+1) + 1);
  end;
end;
M = sort(L);
h = length(find(M));
ERROMAXFORADIAG = max(M);
ERROMEDIOFORADIAG = mean(M(N-h:N-1,1));

echo on
%         Calculo concluido.
```

```
pause %  Pressione qualquer tecla para continuar
clc
ERROMAXFORADIAG
ERROMEDIOFORADIAG
```

IMPLEMENTACAO A3

```
echo on
clear
clc

%     DISSERTACAO DE MESTRADO DE INES FERREIRA MORAES
%                    UFRGS  -  FEV 93
%
%          Este programa faz  a  reconstrucao  de  uma  Matriz
%     tridiagonal atraves de  transformacoes  de  Householder
%     sendo que com  modificacoes  adequadas  nos  sinais  da
%     ultima coluna de  B  podemos  reconstruir  unicamente a
%     matriz de Jacobi cujos elementos da diagonal  principal
%     possuem valor 2  e  os  elementos  das  duas  diagonais
%     adjacentes e -1

pause % Pressione qualquer tecla para continuar
clc

%     Obtencao dos autovalores da Matriz A a ser reconstruida
%     e dos autovalores de sua submatriz principal esquerda

echo off
N = input('Ordem da Matriz = ');
format long ;
LB = zeros(N,1);
MU = zeros(N-1,1);
for i = 1:N-1;
V(i,1) = 2 + 2*cos(i*pi/(N+1));
U(i,1) = 2 + 2*cos(i*pi/N);
end;
```

```
V(N,1) = 2 + 2*cos(N*pi/(N+1));
LB = sort(V);
MU = sort(U);
clear U;
clear V;

echo on
LB    % Autovalores de A

pause % Pressione qualquer tecla para continuar
clc

MU    % Autovalores da submatriz principal de A

pause % Pressione qualquer tecla para continuar
clc

%     Calculo da Matriz B simetrica. Aguarde...

echo off
B = zeros(N,N);
C = ones(N,N);
D = ones(N-1,N-1);
A = zeros(N,N);
NUM = ones(1,N-1);
DEN = ones(1,N-1);
DELTA = sum(LB) - sum(MU);
for i = 1:N-1;
B(i,i) = MU(i,:);
end;
B(N,N) = DELTA;
for k = 1:N-1;
  for i = 1:N;
  C(i,k) = LB(i,:) - MU(k,:);
  end;
end;
```

```
for k = 1:N-1;
  for i = 1:N-1;
     if abs(i-k) > 0;
     D(i,k) = MU(i,:) - MU(k,:);
     end;
  end;
end;
T = zeros(N-1,1);
for k = 1:N-1;
NUM(:,k) = - prod(C(:,k));
DEN(:,k) = prod(D(:,k));
T(k,1) = NUM(:,k)/DEN(:,k);
B(k,N) = sqrt(T(k,1));
B(N,k) = B(k,N);
end;
clear C;
clear D;
clear T;
clear NUM;
clear DEN;

echo on
%     Calculo concluido.

pause % Pressione qualquer tecla para continuar
clc
B

pause % Pressione qualquer tecla para continuar
clc

%     Transformando a Matriz B  em  uma  Matriz  Tridiagonal
%     atraves de transformacoes de Householder.
%     Aguarde ...

echo off
```

```
for j = 1:N-2;
W = zeros(N,1);
P = zeros(N,N);
  if B(N-j,N-j+1) > 0;
  s = - norm(B(1:N-j,N-j+1));
  else;
  s = norm(B(1:N-j,N-j+1));
  end;
t = sqrt(2*s*(s-B(N-j,N-j+1)));
GAMA = 1/t;
  for i = 1:N-1-j;
  W(i,1) = GAMA*B(i,N-j+1);
  end;
W(N-j,1) = (B(N-j,N-j+1) - s) * GAMA;
P = eye(N,N) - 2*W*W';
A = P*B*P';
B = A;
end;

%        A matriz C(3,N) representa os elementos  nao  nulos
%    da matriz tridiagonal reconstruida

C = zeros(3,N);
C(1,1) = A(1,1);
C(2,1) = A(2,1);
for k = 2:N-1;
  for i = 1:3;
  C(i,k) = A(i+k-2,k);
  end;
end;
C(2,N) = A(N-1,N);
C(3,N) = A(N,N);

echo on
clc
%     Transformacao concluida.
```

```
pause % Pressione qualquer tecla para continuar
clc
A

pause %  Pressione qualquer tecla para continuar
clc
C

pause %  Pressione qualquer tecla para continuar
clc

%         Calculo do erro maximo e do erro medio cometidos na
%    diagonal principal. Aguarde...

echo off
E = zeros(N,1);
F = zeros(N,1);
for i = 1:N;
  if A(i,i) ~= 2;
  E(i) = abs(A(i,i) - 2);
  end;
F = sort(E);
g = length(find(F));
ERROMAXIDIAG = max(F);
ERROMEDIODIAG = mean(F(N-g+1:N,1));
end;

echo on
%         Calculo concluido.

pause %  Pressione qualquer tecla para continuar
clc

ERROMAXIDIAG
ERROMEDIODIAG
pause %  Pressione qualquer tecla para continuar
```

```
clc

%         Calculo do erro maximo e do  erro  medio  cometidos
%    nas duas diagonais adjacentes a diagonal principal.
%         Aguarde...

echo off
L = zeros(N-1,1);
M = zeros(N-1,1);
for i = 1:N-1;
  if A(i,i+1) ~= -1;
  L(i) = abs(A(i,i+1) + 1);
  end;
M = sort(L);
h = length(find(M));
ERROMAXFORADIAG = max(M);
ERROMEDIOFORADIAG = mean(M(N-h:N-1,1));
end;

echo on
%         Calculo concluido.

pause %  Pressione qualquer tecla para continuar
clc

ERROMAXFORADIAG
ERROMEDIOFORADIAG
```

IMPLEMENTACAO A4

```
C            DISSERTACAO DE MESTRADO DE INES FERREIRA MORAES
C                        FEV-93 UFRGS
C
C            Reconstrucao de uma matriz de Jacobi de ordem n,
C            cujos elementos da diagonal principal devem ter
C            valor 2 e os das codiagonais adjacentes valor -1
C            A reconstrucao e feita a partir de dados
C            espectrais utilizando o metodo de polinomios
C            ortogonais
C
C                LINGUAGEM: FORTRAN-77 para IBM-PC
C

      REAL A(200,200),C(3,200)
      REAL AUT(200),LB(200),MU(200),WI(200),FNUM(200)
      REAL FDE(200),PKML(200),PK(200),CFFDQ(200),CFFD(200)
      REAL CFP(200),EP(200),EFD(200)
      REAL AUX, PI
      INTEGER N

      PI = 3.14159265
      WRITE(*,5)
    5 FORMAT(1X,26HFORNECA A ORDEM DA MATRIZ:)
      READ(*,10) N
   10 FORMAT(I4)

C ** OBTENCAO DOS AUTOVALORES DE A DA SUBMATRIZ PRINCIPAL **

      DO 20 , I = 1 , (N - 1)
```

```
      LB(I) = 2 + 2 * COS(I * PI / (N+1))
      MU(I) = 2 + 2 * COS(I * PI / N )
   20 CONTINUE
      LB(N) = 2 + 2 * COS(N * PI / (N+1))

C ************** ORDENACAO DOS AUTOVALORES DE A **************

      DO 30 , K = 1 , (N - 1)
        DO 40 , I = (K+1) , N
          IF (LB(K) .LT. LB(I)) GOTO 40
            AUX = LB(K)
            LB(K) = LB(I)
           LB(I) = AUX
   40   CONTINUE
   30 CONTINUE

C ******* ORDENACAO DOS AUTOVALORES DA SUBMATRIZ DE  A *******

      DO 50 , K = 1 , (N - 2)
        DO 60 , I = (K + 1) , (N - 1)
          IF (MU(K) .LT. MU(I)) GOTO 60
            AUX = MU(K)
            MU(K) = MU(I)
            MU(I) = AUX
   60   CONTINUE
   50 CONTINUE

C ************** IMPRESSAO DOS AUTOVALORES DE A **************

      WRITE(*,15)
   15 FORMAT(/,1X,16HAUTOVALORES DE A//)
      DO 70, I = 1 , N
        WRITE(*,25) LB(I)
   25   FORMAT(1X,200F10.7)
   70 CONTINUE
```

```
C ***** IMPRESSAO DOS AUTOVALORES DA SUBMATRIZ DE  A ******

      WRITE(*,35)
   35 FORMAT(/,1X,29HAUTOVALORES DA SUBMATRIZ DE A//)
      DO 80, J = 1 , (N - 1)
        WRITE(*,45) MU(J)
   45   FORMAT(1X,200F10.7)
   80 CONTINUE

C ********** RECONSTRUCAO DA MATRIZ DE JACOBI A **********

      DO 90 , J = 2 , N
       AUT(J-1) = LB(J)
   90 CONTINUE
      SMWI = 0.0
      SMWILB = 0.0
      DO 100 , I = 1, N
        AXNUM = 1.0
        AXDE  = 1.0
        IF (I .GT. 1) AUT(I - 1) = LB(I - 1)
        DO 110 , J = 1 , (N - 1)
          FNUM(J) = LB(I) - MU(J)
          AXNUM   = FNUM(J) * AXNUM
          FDE(J)  = LB(I) - AUT(J)
          AXDE    = FDE(J) * AXDE
  110   CONTINUE
        WI(I) = AXNUM / AXDE
        SMWI = SMWI + WI(I)
        SMWILB = SMWILB + WI(I) * LB(I)
  100 CONTINUE
      CFD1 = SMWILB / SMWI
      DO 120 , I = 1 , N
        PKML(I) = 1.0
        PK(I) = LB(I) - CFD1
  120 CONTINUE
      DO 130 , K = 2 , N
```

```
      SML = SMWI
      SMWI = 0.0
      TWI  = 0.0
      DO 140 , I = 1 , N
        PARC = WI(I) * (PK(I)**2)
        SMWI = SMWI + PARC
        TWI  = TWI + LB(I) * PARC
140   CONTINUE
      CFFDQ(K-1) = SMWI / SML
      CFP(K) = TWI / SMWI
      DO 150 , I = 1 , N
        PARC = PK(I)
        PK(I) = ((LB(I)-CFP(K))*PARC)-(CFFDQ(K-1)*PKML(I))
        PKML(I) = PARC
150   CONTINUE
130 CONTINUE
    DO 160, K = 2 , N
      CFFD(K-1) = SQRT(CFFDQ(K-1))
160 CONTINUE
    DO 170 , K = 2 , N
      EP(N+1-K) = CFP(K)
      EFD(N+1-K) = CFFD(K-1)
170 CONTINUE
    EP(N) = CFD1
    DO 180, I = 1 , N
      DO 190, J = 1, N
        IF (J .EQ. I) A(I,J) = EP(J)
        IF (ABS(I-J) .GT. 1) A(I,J) = 0.
        IF ((I-J) .EQ. -1) A(I,J) = EFD(I) * (-1)
        IF ((I-J) .EQ. 1)  A(I,J) = EFD(J) * (-1)
190   CONTINUE
180 CONTINUE

C ** CALCULO DOS ELEMENTOS NAO NULOS DA MATRIZ DE JACOBI ***

    C(1,1) = A(1,1)
```

```
      C(2,1) = A(2,1)
      DO 200 , K = 2 , (N-1)
        DO 210 , I = 1 , 3
          C(I,K) = A(I+K-2,K)
  210   CONTINUE
  200 CONTINUE
      C(2,N) = A(N-1,N)
      C(3,N) = A(N,N)

C ************** IMPRESSAO DA MATRIZ DE JACOBI **************

      WRITE(*,55)
   55 FORMAT(/,1X,16HMATRIZ DE JACOBI/)
      DO 220 , J = 1 , N
        WRITE(*,65) J
   65   FORMAT(1X,/,I3,7H COLUNA/)
      DO 230 , I = 1 , N
        WRITE(*,75)A(I,J)
   75   FORMAT(1X,200F11.7)
  230   CONTINUE
  220 CONTINUE

C ** IMPRESSAO DOS ELEMENTOS NAO NULOS DA MATRIZ DE JACOBI *

       WRITE(*,85)
   85 FORMAT(/,1X,38HELEMENTOS DAS TRES DIAGONAIS NAO NULAS/)
      DO 240 , J = 1 , N
        WRITE(*,95) J
   95   FORMAT(1X,/,1X,I3,7H COLUNA/)
       DO 250 , I = 1 , 3
        WRITE(*,105) C(I,J)
  105   FORMAT(1X,200F11.7)
  250  CONTINUE
  240 CONTINUE

C ********************** FIM DA EXECUCAO **********************
```

```
      STOP
      END
```

IMPLEMENTACAO A5

```
echo on
clear
clc

%        DISSERTACAO DE MESTRADO DE INES FERREIRA MORAES
%                        UFRGS  -  FEV 93
%
%        Este programa faz a reconstrucao de uma  matriz  de
%    Jacobi de ordem n para o caso  tratado  no  capitulo  5
%    utilizando polinomios ortogonais. Sendo que a matriz  a
%    ser reconstruida devera ter valor 2 para  os  elementos
%    da diagonal principal, exceto  na  ultima  posicao  que
%    sera 1 e os elementos  das  duas  diagonais  adjacentes
%    deverao ser -1.  E valido para os dois casos mencionados
%    no capitulo 5.

pause %  Pressione qualquer tecla para continuar
clc

n = input('ORDEM DA MATRIZ = ');
m = input('ORDEM DA SUBMATRIZ B + 1 = ');

%        Obtencao  dos  autovalores  da  matriz  a  ser
%    reconstruida  e  dos  autovalores  das  submatrizes
%    principais esquerda (B) e direita (C) de A

echo off
format long;
p = n-m;
```

```
V = zeros(n,1);
U = zeros(m-1,1);
X = zeros(p,1);
LB = zeros(n,1);
MU = zeros(m-1,1);
GA = zeros(p,1);
for k = 1:n;
V(k,1) = 2 - 2*cos((2*k - 1)/(2*n + 1)*pi);
end;
for k = 1:m-1;
U(k,1) = 2 - 2*cos(k*pi/m);
end;
for k = 1:p;
X(k,1) = 2 - 2*cos((2*k - 1)/(2*p + 1)*pi);
end;
LB = sort(V);
MU = sort(U);
GA = sort(X);

echo on
LB    %   Autovalores de A

pause %  Pressione qualquer tecla para continuar
clc

MU    %   Autovalores de B

pause %  Pressione qualquer tecla para continuar
clc

GA    %   Autovalores de C

pause %  Pressione qualquer tecla para continuar
clc

%         Fazendo a reconstrucao da matriz B. Aguarde ...
```

```
echo off
%         Obtencao dos pesos Wi

AUXP = ones(n,m-1);
P = ones(1,m-1);
for k = 1:m-1;
  for i = 1:n;
    if MU(k,1) == LB(i,1);
    l = i;
    f = k;
     for j = 1:n;
       if j ~=i;
       AUXP(j,k) = LB(i,1) - LB(j,1);
       end;
     end;
    else;
    AUXP(i,k) = MU(k,1) - LB(i,1);
    end;
  end;
end;
for k = 1:m-1;
P(:,k) = prod(AUXP(:,k));
end;
AUXPP = ones(m-1,m-1);
PP = ones(1,m-1);
for k = 1:m-1;
  for i = 1:m-1;
    if i ~= k;
    AUXPP(i,k) = MU(k,1) - MU(i,1);
    end;
  end;
end;
for k = 1:m-1;
PP(:,k) = prod(AUXPP(:,k));
end;
AUXQ = ones(p,m-1);
```

```
Q = ones(1,m-1);
for k = 1:m-1;
  for i = 1:p;
    if MU(k,1) == GA(i,1);
    g = i;
      for j = 1:p;
        if j ~= i;
        AUXQ(j,k) = GA(g,1) - GA(j,1);
        end;
      end;
    else;
    AUXQ(i,k) = MU(k,1) - GA(i,1);
    end;
  end;
end;
for k = 1:m-1;
Q(:,k) = prod(AUXQ(:,k));
end;
W = ones(m-1,1);
for k = 1:m-1;
 if k == f;
 W(k,:) = - P(:,k)/(PP(:,k)*Q(:,k));
 ALFA = 3/7;
 W(k,:) = W(k,:)*ALFA;
 else;
 W(k,:) = - P(:,k)/(PP(:,k)*Q(:,k));
 end;
end;

%        Os valores dos dois primeiros polinomios sao
%    gerados a partir de MU (PK e PY) e, R(1,1) e obtido

AX = zeros(m-1,1);
PY = zeros(m-1,1);
PK = ones(m-1,1);
R = zeros(m-1,m-1);
```

```
for k = 1:m-1;
AX(k) = W(k)*MU(k);
end;
R(1,1) = sum(AX)/sum(W);
for j = 1:m-1;
PY(j) = MU(j) - R(1,1);
end;

%        Calculo dos outros polinomios e dos elementos nao
%    nulos da matriz R (reflexao de B sobre a segunda
%    diagonal)

PL = zeros(m-1,1);
PM = zeros(m-1,1);
D = zeros(m-2,1);
QY = zeros(m-2,1);
B = zeros(m-1,m-1);
S = sum(W);
for i = 2:m-1;
  for k = 1:m-1;
  PL(k) = W(k)*PY(k)^2;
  PM(k) = MU(k)*PL(k);
  end;
 QY(i-1) = sum(PL)/S;
 R(i,i) = sum(PM)/sum(PL);
 S = sum(PL);
  for j = 1:m-1;
  PL(j) = PY(j);
  PY(j) = (MU(j)-R(i,i))*PL(j) - QY(i-1)*PK(j);
  PK(j) = PL(j);
  end;
  if i == m-2;
  DD = PY(1);
  end;
end;
for k = 2:m-1;
```

```
R(k-1,k) = - sqrt(QY(k-1));
D(k-1,1) = QY(k-1);
end;

%        Obtencao da matriz B a partir de R

B(m-1,m-1) = R(1,1);
for k = 2:m-1;
B(m-k,m-k) = R(k,k);
B(m-k,m-k+1) = R(k-1,k);
B(m-k+1,m-k) = B(m-k,m-k+1);
end;

echo on
%        Reconstrucao concluida.

pause %  Pressione qualquer tecla para continuar
clc
B

pause %  Pressione qualquer tecla para continuar
clc

%        Reconstrucao da matriz C. Aguarde ...

echo off
%        Obtencao dos pesos WWi

AUXPX = ones(n,p);
PX = ones(1,p);
for k = 1:p;
  for i = 1:n;
    if GA(k,1) == LB(i,1);
      for j = 1:n;
        if j ~= i;
        AUXPX(j,k) = LB(i,1) - LB(j,1);
```

```
        end;
      end;
    else;
    AUXPX(i,k) = GA(k,1) - LB(i,1);
    end;
   end;
end;
for k = 1:p;
PX(:,k) = prod(AUXPX(:,k));
end;
AUXPPX = ones(m-1,p);
PPX = ones(1,p);
for k = 1:p;
  for i = 1:m-1;
    if  GA(k,1) == MU(i,1);
      for j = 1: m-1;
        if j ~= i;
        AUXPPX(j,k) = MU(i,1) - MU(j,1);
        end;
      end;
    else;
    AUXPPX(i,k) = GA(k,1) - MU(i,1);
    end;
  end;
end;
for k = 1:p;
PPX(:,k) = prod(AUXPPX(:,k));
end;
AUXQX = ones(p,p);
QX = ones(1,p);
for k = 1:p;
  for i = 1:p;
    if i ~= k;
    AUXQX(i,k) = GA(k,1) - GA(i,1);
    end;
  end;
```

```
end;
for k = 1:p;
QX(:,k) = prod(AUXQX(:,k));
end;
WW = ones(p,1);
for k = 1:p;
  if k == g;
  WW(k,:) = - PX(:,k)/(PPX(:,k)*QX(:,k));
  BETA = 1 - (3/7);
  WW(k,:) = WW(k,:)*BETA;
  else;
  WW(k,:) = - PX(:,k)/(PPX(:,k)*QX(:,k));
  end;
end;

%       Os valores dos dois primeiros polinomios sao
%    gerados a partir de GA (PKX e PZ) e H(p,p) e obtido

AXX = zeros(p,1);
PZ = zeros(p,1);
PKX = ones(p,1);
RX = zeros(p,p);
H = zeros(p,p);
for k = 1:p;
AXX(k) = WW(k)*GA(k);
end;
H(p,p) = sum(AXX)/sum(WW);
for j = 1:p;
PZ(j) = GA(j) - H(p,p);
end;

%       Calculo dos outros polinomios e dos elementos nao
%    nulos da matriz H (reflexao de C sobre a segunda
%    diagonal)

PLX = zeros(p,1);
```

```
PMX = zeros(p,1);
QX = zeros(p-1,1);
C = zeros(p,p);
SX = sum(WW);
for i = 2:p;
  for k = 1:p;
  PLX(k) = WW(k)*PZ(k)^2;
  PMX(k) = GA(k)*PLX(k);
  end;
 QX(p-i+1) = sum(PLX)/SX;
 H(p-i+1,p-i+1) = sum(PMX)/sum(PLX);
 SX = sum(PLX);
  for j = 1:p;
  PLX(j) = PZ(j);
  PZ(j) = (GA(j)-H(p-i+1,p-i+1))*PLX(j) - QX(p-i+1)*PKX(j);
  PKX(j) = PLX(j);
  end;
  if i == p - 1;
  EE = PZ(1);
  end;
end;
E = zeros(p-1,1);
for k = 2:p;
H(p-k+1,p-k+2) = - sqrt(QX(p-k+1));
E(p-k+1,1) = QX(p-k+1);
end;

%        Obtencao da matriz C a partir de H

C(1,1) = H(p,p);
for k = 2:p;
C(k,k) = H(p+1-k,p+1-k);
C(k-1,k) = H(p+1-k,p-k+2);
C(k,k-1) = C(k-1,k);
end;
echo on
```

```
%        Reconstrucao concluida.

pause %  Pressione qualquer tecla para continuar
clc
C

pause %  Pressione qualquer tecla para continuar
clc

%        Reconstrucao da matriz de Jacobi A a partir de B  e
%    C e obtencao dos valores bm e bm+1. Aguarde ...

echo off
A = zeros(n,n);
A(m-1,m) = - sqrt(-P(1)*DD/(QC:,1)*prod(D)));
A(m,m-1) = A(m-1,m);
A(m,m+1) = - sqrt(-PX(1)*EE/(prod(E)*PPX(1)));
A(m+1,m) = A(m,m+1);
A(1:m-1,1:m-1) = B(:,:);
A(m+1:n,m+1:n) = C(:,:);
A(m,m) = sum(LB) - sum(diag(B)) - sum(diag(C));

echo on
%        Reconstrucao da matriz de Jacobi concluida.

pause %  Pressione qualquer tecla pra continuar
clc
A
```

IMPLEMENTACAO A6

```
echo on
clear
clc

%        DISSERTACAO DE MESTRADO DE INES FERREIRA MORAES
%                        UFRGS  -  FEV 93
%
%        Este programa faz a reconstrucao de uma  matriz  de
%    Jacobi de ordem N para o caso  tratado  no  capitulo  5
%    utilizando o algoritmo de Lanczos. Sendo que a matriz a
%    ser reconstruida devera ter valor 2  nos  elementos  da
%    diagonal principal, exceto na ultima posicao que sera 1
%    e os elementos das duas  diagonais  adjacentes  deveram
%    ser -1. E valido para  os  dois  casos  mencionados  no
%    capitulo 5.
%

pause %  Pressione qualquer tecla para continuar
clc

n = input('ORDEM DA MATRIZ = ');
m = input('ORDEM DA SUBMATRIZ B + 1 = ');

%        Obtencao  dos  autovalores  da  matriz  a  ser
%    reconstruida  e  dos  autovalores  das  submatrizes
%    principais esquerda (B) e direita (C) de A

echo off
format long;
```

```
p = n-m;
V = zeros(n,1);
U = zeros(m-1,1);
X = zeros(p,1);
LB = zeros(n,1);
MU = zeros(m-1,1);
GA = zeros(p,1);
for k = 1:n;
V(k,1) = 2 - 2*cos((2*k - 1)/(2*n + 1)*pi);
end;
for k = 1:m-1;
U(k,1) = 2 - 2*cos(k*pi/m);
end;
for k = 1:p;
X(k,1) = 2 - 2*cos((2*k - 1)/(2*p + 1)*pi);
end;
LB = sort(V);
MU = sort(U);
GA = sort(X);

echo on
LB    %  Autovalores de A

pause %  Pressione qualquer tecla para continuar
clc

MU    %  Autovalores de B

pause %  Pressione qualquer tecla para continuar
clc

GA    %  Autovalores de C

pause %  Pressione qualquer tecla para continuar
clc
%        Fazendo a reconstrucao das matrizes B e C.
```

```
%          Aguarde ...

echo off
%          Obtencao da ultima componente dos autovetores de  B

AUXP = ones(n,m-1);
P = ones(1,m-1);
for k = 1:m-1;
  for i = 1:n;
    if MU(k,1) == LB(i,1);
    l = i;
    f = k;
      for j = 1:n;
         if j ~= i;
         AUXP(j,k) = LB(i,1) - LB(j,1);
         end;
      end;
    else;
    AUXP(i,k) = MU(k,1) - LB(i,1);
    end;
  end;
end;
for k = 1:m-1;
P(:,k) = prod(AUXP(:,k));
end;
AUXPP = ones(m-1,m-1);
PP = ones(1,m-1);
for k = 1:m-1;
  for i = 1:m-1;
    if i ~= k;
    AUXPP(i,k) = MU(k,1) - MU(i,1);
    end;
  end;
end;
for k = 1:m-1;
PP(:,k) = prod(AUXPP(:,k));
```

```
end;
AUXQ = ones(p,m-1);
Q = ones(1,m-1);
for k = 1:m-1;
  for i = 1:p;
    if MU(k,1) == GA(i,1);
    g = i;
       for j = 1:p;
         if j ~= i;
         AUXQ(j,k) = GA(g,1) - GA(j,1);
         end;
       end;
    else;
    AUXQ(i,k) = MU(k,1) - GA(i,1);
    end;
  end;
end;
for k = 1:m-1;
Q(:,k) = prod(AUXQ(:,k));
end;
S = ones(1,m-1);
for k = 1:m-1;
   if k == f;
   S(:,k) = - P(:,k)/(PP(:,k)*Q(:,k));
   ALFA = 3/7;
   q = S(:,k);
   S(:,k) = S(:,k)*ALFA;
   else;
   S(:,k) = - P(:,k)/(PP(:,k)*Q(:,k));
   end;
end;
BMQ = sum(S);
Y = ones(m-1,1);
for k = 1:m-1;
Y(k,1) = sqrt(S(:,k))/sqrt(BMQ);
end;
```

```
%       Obtencao da primeira componente dos autovetores de C

AUXPX = ones(n,p);
PX = ones(1,p);
for k = 1:p;
  for i = 1:n;
   if GA(k,1) == LB(i,1);
     for j = 1:n;
       if j ~= i;
       AUXPX(j,k) = LB(i,1) - LB(j,1);
       end;
     end;
   else;
   AUXPX(i,k) = GA(k,1) - LB(i,1);
   end;
  end;
end;
for k = 1:p;
PX(:,k) = prod(AUXPX(:,k));
end;
AUXPPX = ones(m-1,p);
PPX = ones(1,p);
for k = 1:p;
  for i = 1:m-1;
    if GA(k,1) == MU(i,1);
      for j = 1:m-1;
        if j ~= i;
        AUXPPX(j,k) = MU(i,1) - MU(j,1);
        end;
      end;
    else;
    AUXPPX(i,k) = GA(k,1) - MU(i,1);
    end;
  end;
end;
for k = 1:p;
```

```
PPX(:,k) = prod(AUXPPX(:,k));
end;
AUXQX = ones(p,p);
QX = ones(1,p);
for k = 1:p;
  for i = 1:p;
    if i ~= k;
    AUXQX(i,k) = GA(k,1) - GA(i,1);
    end;
  end;
end;
for k = 1:p;
QX(:,k) = prod(AUXQX(:,k));
end;
T = ones(1,p);
for k = 1:p;
 if k == g;
 T(:,k) = - PX(:,k)/(PPX(:,k)*QX(:,k));
 r = T(:,k);
 BETA = 1 - (3/7);
 T(:,k) = T(:,k)*BETA;
 else;
 T(:,k) = - PX(:,k)/(PPX(:,k)*QX(:,k));
 end;
end;
BMUQ = sum(T);
Z = ones(p,1);
for k = 1:p;
Z(k,1) = sqrt(T(:,k))/sqrt(BMUQ);
end;

%         Reconstrucao de B

XN = zeros(m-1,m-1);
B = zeros(m-1,m-1);
R = zeros(m-1,m-1);
```

```
XNC:,m-1) = YC:,:);
for i = 1:m-2;
BCm-i,m-i) = XNC:,m-i)'*diagCMU)*XNC:,m-i);
RC:,m-i) = BCm-i,m-i)*XNC:,m-i) - diagCMU)*XNC:,m-i);
  if i > 1;
  RC:,m-i) = RC:,m-i) - BCm-i,m-i+1)*XNC:,m-i+1);
  end;
BCm-i-1,m-i) = normCRC:,m-i));
XNC:,m-i-1) = RC:,m-i)/BCm-i-1,m-i);
end;
BC1,1) = XNC:,1)'*diagCMU)*XNC:,1);
for i = 1:m-2;
BCm-i-1,m-i) = -BCm-i-1,m-i);
BCm-i,m-i-1) = BCm-i-1,m-i);
end;

%         Reconstrucao de C

C = zerosCp,p);
RX = zerosCp,p);
NX = zerosCp,p);
NXC:,1) = ZC:,:);
for i =2:p;
CCi-1,i-1) = NXC:,i-1)'*diagCGA)*NXC:,i-1);
RXC:,i-1) = CCi-1,i-1)*NXC:,i-1) - diagCGA)*NXC:,i-1);
  if i > 2;
  RXC:,i-1) = RXC:,i-1) - CCi-2,i-1)*NXC:,i-2);
  end;
CCi-1,i) = normCRXC:,i-1));
NXC:,i) = RXC:,i-1)/CCi-1,i);
end;
CCp,p) = NXC:,p)'*diagCGA)*NXC:,p);
for i = 1:p-1;
CCi,i+1) = - CCi,i+1);
CCi+1,i) = CCi,i+1);
end;
```

```
echo on
%         Reconstrucao concluida.

pause %  Pressione qualquer tecla para continuar
clc
B

pause %  Pressione qualquer tecla para continuar
clc
C

pause %  Pressione qualquer tecla para continuar
clc

%         Reconstrucao da matriz de Jacobi A a partir de B  e
%    C e obtencao dos valores bm e bm+1.  Aguarde ...

echo off
A = zeros(n,n);
A(m-1,m) = - sqrt(BMQ);
A(m,m+1) = - sqrt(BMUQ);
A(m,m-1) = A(m-1,m);
A(m+1,m) = A(m,m+1);
A(1:m-1,1:m-1) = B(:,:);
A(m+1:n,m+1:n) = C(:,:);
A(m,m) = sum(LB) - sum(diag(B)) - sum(diag(C));

echo on
%         Reconstrucao da matriz de Jacobi concluida.

pause %  Pressione qualquer tecla para continuar
clc
A
```

```
%    DISSERTACAO DE MESTRADO DE INES FERREIRA MORAES
%                        UFRGS  -  FEV 93
%         Este programa calcula a matriz inercia e  a  matriz
%    rigidez de um sistema vibracional de molas a partir  da
%    matriz de Jacobi reconstruida.

echo off
format long;
n = n;
O = zeros(n+1,1);
O(n,1) = 1;

%    Os valores da matriz A sao a entrada.

A = zeros(n,n);
for i = 1:n-1;
  if i == 1;
  O(n-i,1) = - (A(n-i+1,n-i+1) * O(n-i+1,1))/A(n-i+1,n-i);
  elseif i ~= 1;
  O(n-i,1) = - A(n-i+1,n-i+1) * O(n-i+1,1);
  O(n-i,1) = O(n-i,1) - A(n-i+1,n-i+2) * O(n-i+2,1);
  O(n-i,1) = O(n-i,1)/A(n-i+1,n-i);
  end;
end;
VETOR = O(1:n,1);
RAIZMATRIZINERCIA = diag(VETOR);
MATRIZINERCIA = RAIZMATRIZINERCIA ^ 2;
MATRIZRIGIDEZ = RAIZMATRIZINERCIA * A * RAIZMATRIZINERCIA;
```

BIBLIOGRAFIA

[1] Biegler-König, F.W., Construction of band matrices from spectral data, Linear Algebra Appl., 40 (1981) 79-87.

[2] Bishop, R.E.D., Gladwell, G.M.L. e Michaelson, S., The Matrix Analysis of Vibration, Cambridge, Cambridge University Press, (1965).

[3] Bishop, R.E.D. e Johnson, D.C., The Mechanics of Vibration, Cambridge, Cambridge University Press, (1960).

[4] De Boor, C. e Golub G.H., The numerically stable reconstruction of a Jacobi matrix from spectral data, Linear Algebra Appl., 21, (1978) 245-260.

[5] Forsythe, G.E., Generation and use of orthogonal polynomials for data-fitting with a digital computer, Journal SIAM, 5, (1957) 74-88.

[6] Franklin, J. N., Matrix theory, Prentice

Hall Inc., New Jersey, (1968).

[7] Gladwell, G.M.L., Inverse Problems in Vibration, Martinus Nijhoff Publishers, Dordrechdt, (1986).

[8] Gladwell, G.M.L. e Willms, N.B., The reconstruction of a tridiagonal system from its frequency response at an interior point, Inverse problems, 4, (1988) 1013-1024.

[9] Golub, G.H. e Boley, D.L., Inverse eigenvalue problems for band matrices, Numerical Analysis, G.A. Watson, Berlin (1977).

[10] Golub, G.H., Some modified matrix eigenvalue problems, SIAM Review, 15, (1973) 318-334.

[11] Golub, G.H. e Underwood, R., Stationary values of ratio of quadratic forms subject to linear constraints , Z. Angew. Math. Phys., 21, (1970) 318 - 326.

[12] Golub, G.H. e Van Loan C.F., Matrix Computations, Johns Hopkins University Press, Baltimore, (1983).

[13] Hald, O.H., Inverse eigenvalue problems for Jacobi matrices, Linear Algebra Appl., 14, (1976) 63-89.

[14] Hill, D. R., Experiments in Computacional Matrix Algebra, The Random House, New York, (1988).

[15] Hochstadt, H., On some inverse problems in matrix theory, Archiv der Math., 18 (1967), 201-207.

[16] Hochstadt, H., On the construction of a Jacobi matrix from spectral data, Linear Algebra Appl., 8 (1974), 435-446.

[17] Lancaster, P. e Tismenetsky, M., The Theory of Matrices, Academic Press, New York, (1985).

[18] Lanczos C., An interation method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. NBS, 45 (1950) 255-282.

[19] Newland, D. E., Mechanical vibration analysis and computation, John Willey, New York, (1989).

[20] Ortega, J. M. e Poole, W. G., An Introduction to Numerical Methods for Differential Equations, Pitman Publishing Inc., Marshfield, (1981).

[21] Seto, W. W., Mechanical Vibration, New York, McGraw-Hill, (1964).

[22] Strang, G., Introduction to Apllied Mathematics, Wellesley-Cambridge Press, Wellesley, (1986).

[23] Strang, G., Linear Algebra and its Applications, Academic Press, New York, (1976).

[24] Wilkinson, J. H., The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, (1965).

[25] Willms, N. B., Some matrix inverse eigenvalue problems, Master's thesis, University of Waterloo, Ontario, (1988).