Numerical Analysis Project
Manuscript NA-86-37

November 1986

A Survey of Matrix Inverse Eigenvalue Problems

by

Daniel Boley and Gene H. Golub

Numerical Analysis Project
Computer Science Department
Stanford University
Stanford, California 94305

A Survey of Matrix Inverse Eigenvalue Problems*

Daniel Boley†

University of Minnesota

Gene H. Golub†

Stanford University

ABSTRACT

In this paper, we present a survey of some recent results regarding direct methods for solving certain symmetric inverse eigenvalue problems. The problems we discuss in this paper are those of generating a symmetric matrix, either Jacobi, banded, or **some variation** thereof, given only some information on the eigenvalues of the matrix itself and some of its principal submatrices. Much of the motivation for the problems discussed in this paper came about from an interest in the inverse Sturm-Liouville problem.

* A preliminary version of this report was issued as a technical report of the Computer Science Department, University of Minnesota, TR **86-20,** May **1986.**

† **The** research of the first author was partlally supported by NSF Grant DCR-8420935 and DCR-8619029, and that of the second author by NSF Grant DCR-8412314.

A Survey of Matrix Inverse Eigenvalue Problems

Daniel Boley and Gene H. Golub

0. Introduction.

In this paper, we present a survey of some recent results regarding direct methods for solving certain symmetric inverse eigenvalue problems. Inverse eigenvalue problems have been of interest in many application areas, including particle physics and geology, as well as numerical integration methods based on Gaussian quadrature rules [14]. Examples of applications can be found in [11], [12], [13], [24], [29]. For example, in [11] the problem is to compute the values of the masses, lengths, and spring constants for a mass-spring system, given only certain spectral data. This leads directly to a problem addressed in this paper of generating a pentadiagonal matrix from spectral data.

Much of the motivation for the problems discussed in this paper came about from an interest in the inverse Sturm-Liouville problem. Frequently, the, matrix approximation to the Sturm-Liouville operator is tridiagonal matrix. There are many similarities between the matrix problems and the continuous problems, but numerical evidence demonstrates that the solution to the matrix problem is not a good approximation to the continuous one (cf [23]).

Much of the basic theory of existence and uniqueness of the solutions for Jacobi matrices was developed in [21] and [20]. These papers also present reconstruction algorithms based on the characteristic polynomials of the matrices and/or moments of functions of these polynomials. The paper [20] also has several illustrative **physical examples.**

The problems we **discuss in** this paper are those of generating a symmetric matrix, either Jacobi, banded, or some variation thereof, given only some information on the eigenvalues of the matrix itself and some of its principal submatrices. Thus, the matrix problems considered here are of a highly structured form, and as a consequence we are able to construct algorithms to solve these problems in a ***finite*** number of steps. General matrix inverse eigenvalue problems have recently been considered in [10], and the algorithms for such problems are of an iterative nature. In this paper, we attempt to bring together several recent methods developed to solve these structured inverse eigenvalue problems, which have been proposed with the **specific** aim of being numerically stable and reasonably **efficient.**

All the methods we will discuss consist of two major parts: In the first part, we start with the given initial eigenvalue data and then compute a certain set of intermediate data consisting of a vector or several vectors. In the second part, an algorithm which has been classically used for reducing a general matrix to a structured form is used here to generate the desired structured matrix. In this paper we emphasize the modularity of the methods in that there is a choice of several different methods to carry out each part.

We begin with several mathematical results which are the foundation of all the methods presented in the paper. We then present methods to solve several inverse eigenvalue problems of various sorts, starting with Jacobi matrices and leading to banded matrices, periodic matrices and other special problems. Note that all the matrices mentioned in this paper are symmetric, with the exception of the transformation matrices which are generally orthogonal. The numerical stability of the methods is assured because we depend exclusively on orthogonal transformations for the generation of the matrix solutions, with the exception of the Lanczos-based schemes whose stability is enhanced under re-orthogonalization (see e.g. [25]). Numerical results are not presented here, but can be found in, for example, [4], [1], [2], [7].

1. Preliminary Theory

In order to present the methods of this paper in a concise way, it is useful to give some preliminary results which play a fundamental role throughout the rest of the paper. These results consist of some preliminary constructions which appear as intermediate steps in the course of the various algorithms discussed.

1.1. Bordered Diagonal Matrices

The first problem we discuss is the construction of a certain matrix having some given eigenvalue properties. Specifically, given the set **of eigenvalues** $\{\lambda_i\}_1^n$ and the set of distinct eigenvalues $\{\mu_i\}_1^{n-1}$ which satisfy an interlacing property

$$\lambda_i \geq \mu_i \geq \lambda_{i+1}, \quad i = 1, \ldots, n\text{-}l, \tag{1.1}$$

we wish to construct a ***bordered diagonal*** matrix A of the form

$$A = \begin{bmatrix} a_{11} & b^T \\ b & A \end{bmatrix} \tag{1.2}$$

where A has eigenvalues $\{\lambda_i\}_1^n$ and $M = \mathrm{diag}(\mu_1, \ldots, \mu_{n-1})$, and $b = (\delta_1, \ldots, \delta_{n-1})^T$ is an $(n-1)$-vector. It is easily shown that

$$a_{11} = \mathrm{trace}(A) - \mathrm{trace}(M) = \sum_1^n \lambda_i - \sum_1^{n-1} \mu_i . \tag{1.3}$$

By expanding the determinant, the characteristic polynomial of A may be written as

$$\det(\lambda I - A) = (\lambda - a_{11}) \prod_{j=1}^{n-1} (\lambda - \mu_j) - \sum_{k=1}^{n-1} \delta_k^2 \Big(\prod_{\substack{j=1 \\ j \neq k}}^{n-1} (\lambda - \mu_j) \Big).$$

Setting $\lambda = \mu_1, \ldots, \mu_{n-1}$ and solving the $(n-1)$ equations for the δ_i^2 yields

$$\delta_i^2 = - \frac{\prod_{j=1}^{n} (\mu_i - \lambda_j)}{\prod_{\substack{j=1 \\ j \neq i}}^{n-1} (\mu_i - \mu_j)} \geq 0, \tag{1.4}$$

thus completely deflning A . The non-negativity condition in (1.4) is guaranteed by the interlacing property (1.1).

Once A (1.2) has been computed, it is easy to show the existence of a tridiagonal matrix J which also satisfles the eigenvalue **conditions,** namely that J have eigenvalues $\{\lambda_i\}_1^n$ and the $(n-1)$ X $(n-1)$ lower principal submatrix $\bar{J}$ have eigenvalues $\{\mu_i\}_1^{n-1}$. We denote the elements of J as

$$J = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & \ddots & & \\ & & & 0 & \\ & & & & \\ & 0 & & & \\ & & & a_{n-1} & b_{n-1} \\ & & & b_{n-1} & a_n \end{bmatrix}. \tag{1.5}$$

Conceptually, the easiest way to generate such a J (though not necessarily the best way from a computational standpoint) is to **first** construct the matrix A above (1.2), and then to use Householder transformations (cf [28] [16]) to reduce it to tridiagonal form in the manner of **TRED2** [8]. The similarity transformations applied in this way have the form

$$\begin{bmatrix} 1 & \mathbf{0} \\ 0 & X_1 \end{bmatrix},$$

where $\mathbf{0}$ denotes a zero vector, and hence the transformations preserve the eigenvalues of both the whole matrix and the first lower principal submatrix. Though this method does produce the desired tridiagonal matrix J, it does not take advantage of any special structure present in the matrix A. In Section 2, we will discuss efficient computational schemes for generating the matrix J. The resulting J will be related to A by

$$\begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \bar{Q} \end{bmatrix} A \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \bar{Q}^T \end{bmatrix} = \begin{bmatrix} a_{11} & b_1 \mathbf{e}_1 \\ b_1 \mathbf{e}_1^T & \bar{J} \end{bmatrix} \equiv J. \tag{1.6}$$

From (1.6), it is evident that the matrix $\bar{Q}$ is the matrix of eigenvectors of $\bar{J}$, and that the first row $\mathbf{x}_1^T$ of $\bar{Q}$ satisfies

$$\hat{\mathbf{b}} = b_1 \mathbf{x}_1; \text{ and } \|\hat{\mathbf{b}}\| = b_1. \tag{1.7}$$

That is, $\hat{\mathbf{b}}$ is the (scaled) first row of the matrix of eigenvectors of $\bar{J}$. We summarize this in

Proposition 1. (a) If we have two sets of eigenvalues $\{\lambda_i\}_1^n$ and $\{\mu_i\}_1^{n-1}$ satisfying (1.1) with the μ_i's simple, then there exists a bordered diagonal matrix A whose eigenvalues are $\{\lambda_i\}_1^n$ and whose first lower principal submatrix M has the form $M = \mathrm{diag}(\mu_1, \ldots, \mu_{n-1})$. One such bordered diagonal A is given by (1.2), whose elements are deflned by (1.3) (1.4).

(b) Under the same conditions, there exists a tridiagonal matrix J whose eigenvalues are $\{\lambda_i\}_1^n$ and whose first lower principal submatrix $\bar{J}$ has eigenvalues $\{\mu_i\}_1^{n-1}$

(c) Given a tridiagonal matrix J having eigenvalues λ_i and whose first lower principal submatrix $\bar{J}$ has distinct eigenvalues μ_i, a scalar multiple of the first row $\mathbf{x}_1^T$ of the matrix of eigenvectors of $\bar{J}$ satisfles the formula (1.4).

Proof: Parts (a) and (b) follow from the previous discussion. Part (c) follows by reversing the construction: starting with J, deflning A by (1.6) and then noticing that the entries of the resulting A must satisfy (1.3) and (1.4). □

Part (c) of this proposition says that we can compute the first row of the matrix of eigenvectors of $\bar{J}$ from the eigenvalue data, without knowing $\bar{J}$ at all! To do this, we compute $\hat{\mathbf{b}}$ by (1.4) and then scale it to have norm 1.

1.2. Leading Entries of the Eigenvectors

In Proposition 1, we gave a formula for the leading entries of the eigenvectors for the $(n-1) \times (n-1)$ submatrix of a tridiagonal matrix. In this section, we derive a similar formula for the leading entries of the eigenvectors for the main $n \times n$ matrix using the same initial data. In this case, however, the formulas does not require that the matrix be tridiagonal. Specifically, we show how to compute the first row $(q_{11} \cdots q_{1n})$ of the matrix Q of eigenvectors of an arbitrary symmetric matrix A, given only eigenvalue data. We assume there exists a symmetric matrix A which is unknown, but which has eigenvalues λ_i and whose $(n-1)\times(n-1)$ lower principal submatrix $\bar{A}$ has eigenvalues μ_i. In addition, we must assume that the $\{\lambda_i\}_1$ and $\{\mu_i\}_1^{n-1}$ satisfy the interlacing property (1.1) and that each set is distinct. In this case, however, we will see that the formulas will not require that the matrix be tridiagonal. With these conditions we construct the first row $(q_{11} \cdots q_{1n})$ of $\mathbf{Q}$ by considering the stationary values of

$$\begin{aligned} &\mathbf{x}^T A \mathbf{x} \\ &\text{subject to } \mathbf{x}^T \mathbf{x} = 1; \mathbf{x}^T \mathbf{c} = 0 \end{aligned} \tag{1.8}$$

where c is some vector with $\mathbf{c}^T \mathbf{c} = 1$. In [15] it was shown that if the stationary values of (1.8) are prescribed to be the μ_i's, then the entries of the vector $\mathbf{d} = \mathbf{Q}^T \mathbf{c}$ satisfy

$$d_i^2 = \frac{\prod_{j=1}^{n-1} (\mu_j - \lambda_i)}{\prod_{j=1}^{n} (\lambda_j - \lambda_i)} \tag{1.9}$$

If we use $\mathbf{c}=\mathbf{e}_1\equiv(1,0,\ldots,0)^T$ then the stationary values of (1.8) are precisely the eigenvalues of $\bar{A}$, and the vector $\mathbf{d}$ is the first column of $\mathbf{Q}^T$. So we have the formula for the first row of $\mathbf{Q}$:

$$q_{1i}^2 = \frac{\prod_{j=1}^{n-1}(\mu_j-\lambda_i)}{\prod_{\substack{j=1\\ j\neq i}}^{n}(\lambda_j-\lambda_i)}. \tag{1.10}$$

We can summarize these results in

Proposition 2. (a) If we are given the distinct eigenvalues $\{\lambda_i\}_1^n$ and the first row $(q_{11}\cdots q_{1n})$ of the matrix $\mathbf{Q}$ of eigenvectors of an arbitrary real symmetric matrix A, partitioned as

$$A = \begin{bmatrix} a_{11} & \mathbf{v}^T \\ \mathbf{v} & \bar{A} \end{bmatrix}$$

where A is $n \times n$ and $\bar{A}$ is $(n-1)\times(n-1)$, then the eigenvalues $\{\mu_i\}_1^{n-1}$ of $\bar{A}$ are uniquely determined by the given eigenvalue/eigenvector information, by formula (1.10). The μ_i's are also independent of the choice of signs for the q_{1i}. Furthermore, if A is also tridiagonal, then A is entirely uniquely determined up to signs of the off-diagonal elements, again independent of the choice of signs of the q_{1i}.

(b) Conversely, if we are given any symmetric matrix A whose eigenvalues are $\{\lambda_i\}_1^n$ and whose first lower principal submatrix has eigenvalues $\{\mu_i\}_1^{n-1}$, where the eigenvalues are distinct and satisfy (1.1), then the first row $(q_{11}\cdots q_{1n})$ of the matrix $\mathbf{Q}$ of eigenvectors of that given matrix A is given by formula (1.10).

Proof: We rewrite (1.10) as

$$q_{1i}^2 \prod_{\substack{j=1\\ j\neq i}}^{n}(\lambda_j-\lambda_i) = \prod_{j=1}^{n-1}(\mu_j-\lambda_i) \equiv p_{n-1}(\lambda_i), \quad i=1,\ldots,n,$$

where $p_{n-1}\equiv\prod_{j=1,\ldots,n-1}(\mu_j-\lambda)$ is a polynomial in λ of degree $n-1$. We have n values of the polynomial p_{n-1} at n different values of λ, so the polynomial p_{n-1}, and hence its roots μ_i, are uniquely determined. The choice of signs of the q_{1i} is irrelevant since the q_{1i} appear only in squared form. The uniqueness of A if A is tridiagonal follows easily from Theorem 4.1 of Chapter 7 of [28]. Again the signs of q_{1i} are irrelevant since we can change the sign of any individual eigenvector of A arbitrarily.

Part (b) follows from the derivation of (1.10) above. □

1.3. Relation to Polynomials

We can show a close relationship between the two formulas (1.4) and (1.9). Consider the Jacobi matrix J (1.5), and its lower principal submatrices $\bar{J}$ $(n-1)\times(n-1)$, and $\bar{\bar{J}}$ $(n-2)\times(n-2)$. Define the characteristic polynomials

$$\begin{aligned} p_n(\lambda) &= \det(J-\lambda I), \\ p_{n-1}(\lambda) &= \det(\bar{J}-\lambda I), \\ p_{n-2}(\lambda) &= \det(\bar{\bar{J}}-\lambda I). \end{aligned}$$

The zeros of the p_k, $k=n, n-1, n-2$, are $\{\lambda_j\}_1^n$, $\{\mu_j\}_1^{n-1}$, and $\{\nu_j\}_1^{n-2}$, respectively. We may expand the determinant in the definition of p_n to obtain the relation

$$p_n(\lambda) = (a_1-\lambda)p_{n-1}(\lambda) - b_1^2 p_{n-2}(\lambda). \tag{1.11}$$

We use this formula to show the close relation between (1.9) and (1.4). Written using the characteristic polynomials, the formula (1.9) can be written as

$$[d_j]^2 = -\frac{p_{n-1}(\lambda_j)}{p'_m(\lambda_j)} \tag{1.12}$$

We can use the same formula to express the $(n$ -l)-vector $\bar{\mathbf{d}}$ for the matrix $\bar{J}$ in terms of the μ_j and ν_j. If we write the result in terms of polynomials we get

$$[\bar{d}_j]^2 = -\frac{p_{n-2}(\mu_j)}{p'_{n-1}(\mu_j)} \tag{1.13}$$

In the same way, we may write the formula (1.4) for the $(n-1)$-vector 6 as

$$[\delta_j]^2 = \frac{p_n(\mu_j)}{p'_{n-1}(\mu_j)} \tag{1.14}$$

Note that the formulas (1.12) and (1.13) hold even if J is not tridiagonal, but that (1.14) holds only for tridiagonal J. The denominators in (1.13) and (1.14) are the same, and the numerators are related by (1.11):

$$p_n(\mu_j) = -b_1^2 p_{n-2}(\mu_j), \; j = 1, \ldots, n-1, \tag{1.15}$$

since $p_{n-1}(\mu_j)$=O. If J is tridiagonal, then the choice of signs in the square root in formulas (1.13), (1.14) is arbitrary; changing signs is equivalent to changing the sign on the entire corresponding eigenvector of $\bar{J}$. So in this case, we can arbitrarily choose all the signs to be non-negative. So we obtain the equivalence in the tridiagonal case:

$$\delta = \pm b_1 \bar{\mathbf{d}}. \tag{1.16}$$

Since (1.14) was derived strictly in terms of determinants, this gives an independent derivation of the formula (1.13) strictly in terms of determinants. Since $\|\delta\| = b_1$ from (1.7), this implies that $\|\bar{\mathbf{d}}\| = 1$. Furthermore, any $(n-1)$ X $(n-1)$ matrix A having eigenvalues μ_j and whose first lower principal submatrix A has eigenvalues ν_j must be related to any other matrix satisfying the same conditions by similarity transformations of the form

$$\begin{bmatrix} 1 & 0 \\ 0 & X \end{bmatrix} \quad ((n-1)\times(n-1)).$$

Applying such transformations leaves unchanged the first components of the eigenvectors of $\bar{A}$, so that the first row of the matrix of eigenvectors is the same (up to signs) for all such matrices. This shows in an independent way that the formula (1.13) holds for all symmetric matrices, not just tridiagonal ones. Conversely, this yields an independent derivation of (1.4) based on the use of problem (1.8) to derive (1.9).

By carrying out a similar development using an $(n+1)$ X$(n+1)$ tridiagonal matrix, one can obtain a similar derivation for the vector $\mathbf{d}$ (1.9). Finally, we note that the polynomials we have deflned are part of a sequence of orthogonal polynomials. We review a few of the close relationships between the polynomials, the eigenvectors of the Jacobi matrix J, and Gaussian Quadrature from [7] [14]. It is well known that if the polynomials $p_0(x), p_1(x), \cdots$ are mutually orthogonal with respect to the inner product

$$\int_\Omega p_i(x) p_j(x) w(x) dx, \tag{1.17}$$

where $w(x)$ is a positive weight function over the interval Ω, then one can determine weights $w_i > 0$ such that the Gauss Quadrature Formula

$$\int_\Omega f(x) w(x) dx = \sum_{i=1}^{n} w_i f(\lambda_i) \tag{1.18}$$

is exact if f is any polynomial of degree up to $2n$ -1, where the $\{\lambda_j\}_1^n$ are the roots of p_n. We assume that the weights are scaled so that

$$\int_\Omega w(x) dx = \sum_{i=1}^{n} w_i = 1.$$

In this case, the $\{ w_i \}_1^n$ are known [14] to be the squares of the first components of the normalized eigenvectors of J, that is (in the notation of (1.10))

$$w_i = q_{1i}^2, \; i = 1, \ldots, n. \tag{1.19}$$

If in (1.18) we let $f(x) = x^k$, $k = 0, \ldots, n-1$, we obtain the first n moments

$$m_k \equiv \int_\Omega x^k \, w(x) dx = \sum_{i=1}^{n} w_i \lambda_i^k. \tag{1.20}$$

All these relationships are fully explored and derived in [14] and [7].

1.4. Lanczos Algorithms

A fundamental procedure that will be needed is the Lanczos Algorithm, used to actually generate the desires matrices. In this paper, we use two flavors of the Lanczos Algorithm, the ordinary scalar version and the "block" version. The Lanczos Algorithm has been used more often to reduce symmetric matrices to tridiagonal form in order to solve for their eigenvalues. In this paper, we use a variation of it to solve the inverse eigenvalue problem.

The "scalar" Lanczos Algorithm is a process that reduces any symmetric matrix to tridiagonal form, starting with the given matrix and a certain starting vector of unit length. It has been discussed extensively elsewhere (cf [16] and references therein), and we present here a summary of the process.

Algorithm 1. - Lanczos Algorithm. Given a symmetric matrix A and x,, with $\mathbf{x}_1^T \mathbf{x}_1 = 1$, compute a Jacobi matrix J (1.5) and an orthogonal matrix $X = [\mathbf{x}_1 \cdots \mathbf{x}_n]$ such that $A = XJX^T$.

1. **begin**
2. set $a_1 := \mathbf{x}_1^T A \mathbf{x}_1$
3. **for** $i = 1, \ldots, n-1$ **do**
   begin
4. **if** $i = 1$ **then** $\mathbf{r}_1 := A\mathbf{x}_1 - \mathbf{x}_1 a_1$
   else $\mathbf{z}_i := A\mathbf{x}_i - \mathbf{x}_i a_i - \mathbf{x}_{i-1} b_{i-1}$
5. **compute** $\mathbf{x}_{i+1}, b_i$ so that $\mathbf{x}_{i+1} b_i = \mathbf{z}_i$, $\mathbf{x}_{i+1}^T \mathbf{x}_{i+1} = 1$
6. **set** $a_{i+1} := \mathbf{x}_{i+1}^T A \mathbf{x}_{i+1}$
   end
   end

This algorithm is based on the idea of alternately using the formulas

$$J = X^T AX \tag{1.21}$$

and $XJ = AX$

to step by step fill in all of J and X (see e.g. [4]) starting with A and the first column $\mathbf{x}_1$ of X. This way to reduce a matrix to tridiagonal form is an alternative to the use of Householder Transformations, but in many situations, numerical stability is enhanced by re-orthogonalizing the vector $\mathbf{z}_i$ computed at each pass through step 4 against all the previously computed x vectors (see e.g. [16]).

If at any stage the vector $\mathbf{z}_i$ is zero, then it must be replaced by an arbitrary unit vector orthogonal to all the previous x vectors, and the corresponding b_i in step 5 must be set to zero. If the eigenvalues $\{\lambda_i\}_1^n$ are all distinct, and the initial vector $\mathbf{x}_1$ is not orthogonal to any eigenvector of A, then this situation cannot occur, and the Jacobi matrix J that results from the algorithm will be unique up to the signs of the off-diagonal elements [25].

The Lanczos Algorithm, as stated, takes $O(n)$ floating point operations for each pass through the loop (steps 4, 5). So the total work is $O(n^2)$. However, the cost of the reorthogonalization step is about $O(ni)$ in each pass through step 4, so that with reorthogonalization the overall cost is about $O(n^3)$.

The Block Lanczos Algorithm is an extension of the scalar Lanczos Algorithm which generates banded matrices. **Specifically,** the Block Lanczos Algorithm starts with an $n \times n$ symmetric matrix A and p orthonormal starting vectors $x_1, \ldots, x_p$, where $n = ps$ for some integer s. The Algorithm then generates a block tridiagonal matrix J with $p \times p$ blocks, as well as computing a complete set of n orthogonal vectors $Q = X^T$ as in the scalar Lanczos Algorithm (Algorithm 1.). We write the matrix of generated orthogonal columns as

$$\mathbf{X} = [X_1, \ldots, X_s], \text{ where } X_1 = [\mathbf{x}_1, \ldots, \mathbf{x}_p], \tag{1.22}$$

and the generated $n \times n$ matrix J as

$$J = \begin{vmatrix} H_1 & B_1^T & & & \\ B_1 & H_2 & \ddots & & \\ & \ddots & \ddots & 0 & \\ & & \ddots & & \\ & 0 & \ddots & \ddots & \\ & & & H_{s-1} & B_{s-1}^T \\ & & & B_{s-1} & H_s \end{vmatrix}. \tag{1.23}$$

The matrices H_i are symmetric, and the B_i **are** upper triangular, so that the entire matrix J has $2p+1$ bands. We say that J has ***half-bandwidth*** p, meaning that J has p bands below the main diagonal, so that $p = 1$ for a tridiagonal matrix.

As in Algorithm 1, this method consists of alternately using the formulas

$$X^T AX = J \quad \text{and} \quad AX = XJ$$

to compute the individual blocks of J and $\mathbf{X}$ one at a time. The **specific** algorithm is as follows:

Algorithm 4. - Block Lanczos Algorithm. Given a symmetric A and the vectors $\mathbf{X}_1$ n Xp, with $\mathbf{X}_1^T\mathbf{X}_1 = I$, compute an $n \times n$ block tridiagonal matrix J (1.23) and an orthogonal matrix $X = [X_1 \cdots \mathbf{X}_s]$ (1.22) such that $A = XJX^T$. Each block H_i and B_i **is** $p \times p$. The H_i **will** be symmetric, and the B_i will be upper triangular.

1. **begin**
2. **set** $H_1 := X_1^T AX_1$
3. **for** $i = 1, \ldots, s-1$ **do**
   begin
4. **if** $i = 1$ **then**
   set $Z_1 := AX_1 - X_1 H_1$
   else
   set $Z_i := AX_i - X_i H_i - X_{i-1} B_{i-1}^T$
5. **compute** X_{i+1}, B_i such that

$$X_{i+1} B_i = Z_i$$
$$X_{i+1}^T X_{i+1} = I_{p \times p}$$
B_i is $p \times p$ upper triangular

6. **set** $H_{i+1} := X_{i+1}^T AX_{i+1}$
   end
   end.

Step 5 can be carried out by using either a modified Gram-Schmidt procedure, or using an QR-decomposition algorithm ([16], [22] routine SQRDC). In case the matrix Z_i is rank deficient, one must choose the columns of X_{i+1} so that $X_{i+1}^T X_{i+1} = I_{p \times p}$ still holds and so that X_{i+1} is still orthogonal to all previous columns $X_1, \ldots, X_i$, absorbing the rank-deficiency into B_i. Beyond

this requirement, one can choose the columns of X_i arbitrarily.

We note that this algorithm reduces almost exactly to Algorithm 1 in the case that $p = 1$.

In the context of the methods in this paper, the Lanczos algorithms are used in a somewhat unusual way. In all cases, we wish to generate matrices J with certain prescribed eigenvalues $\{\lambda_i\}_1^n$. Hence, the "starting matrix" A that we typically use in this situation is one guaranteed to have the given eigenvalues λ_i, namely $A = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. In this special case, the matrix X generated by the Lanczos process is exactly thetranspose of the matrix of eigenvectors of the generated matrix J, and the p starting vectors $\mathbf{x}_1^T, \ldots, \mathbf{x}_p^T$ are exactly the first p rows of this eigenvector matrix. In this context, the scalar Lanczos Algorithm becomes a method that generates a tridiagonal matrix, given its eigenvalues and the first row of its eigenvector matrix, and the Block Lanczos Algorithm becomes a method that generates a banded matrix with half-bandwidth p, given its eigenvalues and the first p rows of its eigenvector matrix.

2. Jacobi Matrices.

In this section we introduce the basic techniques used throughout this paper by discussing methods for the classical problem of reconstructing a Jacobi matrix (1.5):

$$J = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & \cdot & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \\ & & & a_{n-1} & b_{n-1} \\ & & & b_{n-1} & a_n \end{bmatrix}$$

given its-eigenvalues $\{\lambda_i\}_1^n$ and the eigenvalues $\{\mu_i\}_1^{n-1}$ of the $(n-1) \times (n-1)$ lower principal submatrix J of J. In the following, we will refer to this reconstruction problem as **Problem J.** These methods have varying requirements on the Xi, μ_i, but in general we will assume they interlace as in (1.1). In some cases, we will need to assume that every eigenvalue be simple, whereas in others we will also need a strict interlacing property, in which the inequalities in (1.1) are strict.

2.1. Method 1 - Lanczos.

This method is based on [7] [2]. The goal is to construct the flrst row of the eigenvector matrix Q of J from the given eigenvector data, and then use the Lanczos algorithm, suitably modified, to construct J.

From part (b) of Proposition 2, we know that the first row $(q_{11} \cdots q_{1n})$ of $\mathbf{Q}$ is determined by the given eigenvalue data and can be computed by formula (1.10). It remains to show how this information can be used to generate J.

To generate the tridiagonal matrix J, we next apply the Lanczos Algorithm (Algorithm 1), suitably modifled. To see how this is done, recall that the Lanczos Algorithm computes a tridiagonal matrix J orthogonally similar to the original starting matrix (A). The matrix A and the first column of the transformation $\mathbf{X}$ relating J and A (1.21) forms the required input data for the algorithm. For the current problem, we set the starting matrix to be $A = \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, and the starting vector to be the vector $\mathbf{x}_1 = \mathbf{d}$ deflned by (1.9). It then follows from (1.21) that the generated matrix J will have eigehvalues $\lambda_1, \ldots, \lambda_n$ and eigenvectors X^T. Hence by part (a) of Proposition 2, the matrix J must solve Problem J.

When the starting matrix in the Lanczos Algorithm is diagonal, the matrix-vector products takes only $0(n)$ time, instead of $0(n^2)$ time in the general case. Hence the Lanczos Algorithm takes $0(n)$ floating point operations for each pass through the loop (steps 4, 5). So the total work is $0(n^2)$. However, as noted above, to maintain numerical stability, one must reorthogonalize the $\mathbf{z}_i$ produced in step 4, increasing the overall cost to about $0(n^3)$.

12.2. Method 2 - Orthogonal Reduction.

This method is based on ideas of [1] and of [30]. The idea is to first construct a bordered diagonal matrix A_{aug} with the same information that was needed for Method 1: i.e. the eigenvalues λ_i and the flrst row $\mathbf{d}^T$ of the matrix of eigenvectors. Then we reduce this matrix to tridiagonal form using orthogonal (Householder) transformations that do not affect this eigenstruc ture.

In this method, we form the following augmented $(n+1) \times (n+1)$ bordered diagonal matrix

$$A_{aug} = \begin{bmatrix} a_{00} & \\ \mathbf{d}^T & \mathbf{1} \end{bmatrix}, \tag{2.1}$$

where a_{00} is a dummy entry, $\mathbf{d}$ is the n-vector of weights defined by (1.9), and

$\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. The idea of this method is to use a reduction scheme that reduces A_{aug} to tridiagonal form by computing a series of Householder transformations of the form

$$\begin{bmatrix} 1 & 0^T \\ 0 & X \end{bmatrix},$$

where $X = [\mathbf{x}_1 \cdots \mathbf{x}_n]$ is an orthogonal matrix, and 0 represents the zero vector. This method is the same as that used to reduce the n X n bordered diagonal matrix (1.2) to tridiagonal form, mentioned in the discussion leading up to Proposition 1. The resulting tridiagonal form will be related to A_{aug} by

$$\begin{bmatrix} 1 & 0^T \\ 0 & Q \end{bmatrix} \begin{bmatrix} a_{00} & \mathbf{d} \\ \mathbf{d}^T & \Lambda \end{bmatrix} \begin{bmatrix} 1 & 0^T \\ 0 & Q^T \end{bmatrix} = \begin{bmatrix} a_{00} & b_0 \mathbf{e}_1 \\ b_0 \mathbf{e}_1^T & J \end{bmatrix} = J_{aug}, \tag{2.2}$$

where $\mathbf{Q}$ is an orthogonal matrix. It is clear from (2.2) that $\mathbf{Q}$ is exactly the matrix of eigenvectors of the resulting J. If we denote the first row of $\mathbf{Q}$ by $\mathbf{x}_1^T$, it is also clear from (2.2) that

$$\mathbf{d} = Q^T b_0 \mathbf{e}_1 = b_0 \mathbf{x}_1. \tag{2.3}$$

Since $\|\mathbf{x}_1\| = 1$, this formula, with (1.9) (1.10), deflnes the value $b_0 = \mathrm{fl}$. Hence by Proposition 2, the generated matrix J must solve Problem J.

The reduction algorithm used here based on Householder transformations is essentially the same as that described in detail in ([28], pp.334ff). It also appears in [8] under the name TRED2, where it is used during the solution of the ordinary symmetric eigenvalue problem to carry out the initial reduction to tridiagonal form. This method takes about $0(n^3)$ operations (see e.g. [16]).

We conclude this section by noting that during this reduction by orthogonal transformations, we have preserved two sets of eigenvalues: those of A_{aug}, transformed to J_{aug}, and those of A, transformed to J. We have started with exactly the same information as for Method 1 (based on the Lanczos Algorithm) and computed the same intermediate vector of data $\mathbf{d}$.

2.3. Method 3 - Fast Orthogonal Reduction.

The bordered diagonal matrix is a matrix of a very special form, and it is not surprising to discover that one can reduce such a matrix to tridiagonal form using a scheme that is faster than the method based on Householder transformations described above. In this section, we present such a method which takes only $0(n^2)$ operations. This method, noticed by Gragg [17], is based on a reduction scheme of Rutishauser [26]. This method is applied on the same matrix (2.1) as the previous method based on Householder transformations and consists of applying a certain sequence of orthogonal plane rotations in a very particular order.

To explain the process, we need to describe a basic step which is used as the basis for the complete reduction. The basic step consists of reducing the half-bandwidth by 1 band of a k X k matrix of the form

$$H = \begin{bmatrix} a_1 & b_1 & 0^T & d_1 \\ b_1 & a_2 & b_2 \mathbf{e}_1^T & d_2 \\ 0 & 6 & {}_2\mathbf{e}_1 \; T & 0 \\ d_1 & d_2 & 0^T & d_k \end{bmatrix} \tag{2.4}$$

where T is a (possibly empty) $(k-3) \times (k-3)$ tridiagonal matrix, and $\mathbf{e}_1$ is the k-3-vector $[1,0,\ldots,0]^T$. Note the matrix H is tridiagonal, except that the last row and column have a special form. We deflne the orthogonal plane rotation R between rows 2 and k :

$$R = \begin{vmatrix} 1 & 0 & 0^T & 0 \\ 0 & c & 0^T & s \\ 0 & 0 & I & 0 \\ 0 & -s & 0^T & c \end{vmatrix} \tag{2.5}$$

where c ,s are chosen to annihilate the d_1 in position k ,1 of H in (2.4), i.e. choose

$$\sigma = \sqrt{b_1^2 + d_1^2}$$
$$c = 6_1/\sigma$$
$$s = d\ ,/a.$$

If we apply R (2.5) as a similarity transformation to (2.4) we obtain a matrix

$$\tilde{H} = RHR^T = \begin{vmatrix} a_1 & \sigma & 0^T & 0 \\ \sigma & \tilde{a}_2 & \tilde{b}_2 e_1^T & \tilde{d}_2 \\ 0 & \tilde{b}_2 e_1 & T & \tilde{d}_3 e_1 \\ 0 & \tilde{d}_2 & \tilde{d}_3 e_1^T & \tilde{d}_k \end{vmatrix} \qquad (2.6)$$

where the entries have the values:

$$\tilde{a}_2 = \frac{b_1^2 a_2 + 2 b_1 d_1 d_2 + d_1^2 d_k}{\sigma^2},$$
$$\tilde{b}_2 = \frac{b_1 b_2}{\sigma},$$
$$\tilde{d}_2 = d_2 + \frac{b_1 d_1}{\sigma^2}(d_k - a_2),$$
$$\tilde{d}_3 = \frac{b_1^2 d_k - 2 b_1 d_1 d_k + d_1^2 d_2}{\sigma^2},$$

σ has the value deflned above, and the entries a_1, T are unchanged. In this fashion, we have reduced the half-bandwidth by 1 in a constant number of operations independent of k .

In order to further reduce the bandwidth, we re-partition $\tilde{H}$ to obtain the form

$$\tilde{H} = \begin{bmatrix} a_1 & \sigma e_1^T \\ \sigma e_1 & \bar{H} \end{bmatrix} \qquad (2.7)$$

where $\bar{H}$ is a $(k-1) \times (k-1)$ matrix of the same form as H of (2.4):

$$\bar{H} = \begin{bmatrix} \bar{a}_1 & \bar{b}_1 & 0^T & \bar{d}_1 \\ \bar{b}_1 & \bar{a}_2 & \bar{b}_2 e_1^T & \bar{d}_2 \\ 0 & \bar{b}_2 e_1 & \bar{T} & 0 \\ \bar{d}_1 & \bar{d}_2 & 0^T & \bar{d}_{k-1} \end{bmatrix}$$

At this point, $\bar{H}$ is tridiagonal except for the last row and column, which have the same structure as H in (2.4). Since $\bar{H}$ comes from just a re-partitioning (2.7) of $\tilde{H}$, we can write the correspondence between the labels for the individual elements in $\bar{H}$ and the labels for the same elements in $\tilde{H}$:

$\bar{a}_1 = \tilde{a}_2$,
$\bar{a}_2 = a_3$ (part of the unchanged T matrix),
$\bar{b}_1 = \tilde{b}_2$,
$\bar{b}_2 = 6_3$ (part of the unchanged T matrix),
$\bar{d}_1 = \tilde{d}_2$,
$\bar{d}_2 = \tilde{d}_3$,
$\bar{d}_{k-1} = \tilde{d}_k$,
$\bar{T}$ = the $(k-4) \times (k-4)$ lower principal submatrix of T .

We can now "recursively" apply this same operation on the $(k-1) \times (k-1)$ matrix $\bar{H}$ to obtain an orthogonal plane rotation $\bar{R}$ which reduces the half-bandwidth of $\bar{H}$ by 1. That is, the k X k plane rotation

$$P = \begin{bmatrix} 1 & 0^T \\ 0 & \bar{R} \end{bmatrix}$$

transforms the $k \times k$ matrix $\tilde{H}$ to the following matrix of half-bandwidth 2:

$$P\tilde{H}P^T = \tilde{H}^{(2)} = \begin{bmatrix} a_1 & \sigma & 0^T \\ \sigma & \tilde{a}_2 & \tilde{b}_2 e_1^T \\ 0 & \tilde{b}_2 e_1 & \bar{H}^{(2)} \end{bmatrix} = \begin{bmatrix} \hat{H}^{(2)} & b_j e_1^T \\ b_j e_1 & \bar{H}^{(2)} \end{bmatrix}$$

where $\bar{H}^{(2)}$ again has the same structure as H in (2.4), and $\hat{H}^{(2)}$ is a 2×2 matrix. After j steps of this process, we will have a matrix $\bar{H}^{(j)}$ which will have a j X j tridiagonal part in the upper left corner and a $(k-j) \times (k-j)$ part $\bar{H}_{(j)}$ of the form (2.4) in the lower right corner.

After k -2 such steps, the part with the form *(2.4)* in the lower right corner (corresponding to $\bar{H}$ in (2.7)) will be reduced to size 2X 2, and hence will be tridiagonal. In summary, our algorithm is:

Algorithm 2. Given a $k \times k$ matrix H of the form (2.4), orthogonally reduce it to tridiagonal form $J^{(k)} := S^{(k)} H S^{(k)T}$, where the orthogonal transformations are accumulated into $S^{(k)}$.

1. **begin**
2. **if** $k = 2$ **then set** $S^{(k)} := I_{k \times k}$; **set** $J^{(k)} := H$; **return.**
3. **compute** plane-rotation R between planes 2 and k deflned by (2.5).
4. **compute** $\tilde{H}$ deflned by (2.6).
5. **Partition** $\tilde{H}$ as in (2.7), to obtain the $(k-1)$ X $(k$ -1) lower right submatrix $\bar{H}$.
6. **Recursively appiy** this algorithm to $\bar{H}$. The result is a $(A$ -1) X $(k$ -1) tridiagonal matrix $J^{(k-1)}$, and a $(k$ -1) $\times$ $(k$ -1) orthogonal matrix $S^{(k-1)}$ consisting of the accumulated transformations, such that $J^{(k-1)} = S^{(k-1)} \bar{H} S^{(k-1)T}$.
7. **Accumulate** the orthogonal transformations, i.e.

$$\text{set } S^{(k)} := \begin{bmatrix} 1 & 0 \\ 0 & S^{(k-1)} \end{bmatrix} R.$$

8. **Set** $J^{(k)} := S^{(k)} H S^{(k)T}$.

end.

We note that in step **8.**, we have the identity

$$J^{(k)} = S^{(k)} H S^{(k)T} = \begin{bmatrix} 1 & 0^T \\ 0 & S^{(k-1)} \end{bmatrix} \tilde{H} \begin{bmatrix} 1 & 0^T \\ 0 & S^{(k-1)T} \end{bmatrix}.$$

This transformation applied to $\tilde{H}$ has been constructed to have the property that it leaves unchanged the flrst row and column of $\tilde{H}$, so that we may form $J^{(k)}$ by simply replacing the block $\bar{H}$ in $\tilde{H}$ (2.7) with the $(k$ -1) X $(k$ -1) matrix $J^{(k-1)}$, obtaining a tridiagonal matrix.

At each level of recursion in this algorithm, the cost is approximately 30 operations, so the total cost is approximately $30k$ operations, ignoring lower order terms in k . Of course, in an actual implementation, one might not use a recursive definition. The recursive definition is used mainly for clarity in that it eliminates the need to use many iteration indices in the description. One can easily convert this recursive definition to an iterative form (see e.g. [27]).

To see how to apply Algorithm *2* to the problem of reducing an m X m bordered diagonal matrix A of the form (2.1) to tridiagonal form, notice that the upper 3X 3 principal submatrix of A is already of the form (2.4), with the third block column and row of *(2.4)* empty; i.e. the columns and rows occupied by T are not present. Hence we may apply Algorithm 2 to that *3X 3* submatrix, obtaining the matrix $A^{(1)}$ whose upper 3 X3 principal submatrix is tridiagonal. At this stage, the upper 4 X4 principal submatrix of $A^{(1)}$ has the form (2.4), where T is just a scalar, so

we may apply Algorithm 2 to that 4X 4 submatrix, obtaining $A^{(2)}$. We continue in this manner until we reach the bottom row of A after m -2 steps. We summarize the method in

Algorithm 3 - Rutishauser. Given an m X m bordered diagonal matrix $A^{(0)}=A$ of the form (2.1), reduce it to tridiagonal form J by similarity transformations based on orthogonal plane rotations.

begin

1. **Let** S := an m X m identity matrix. (S is used to accumulate the transformations).
2. **For** $i = 3, \ldots,$ m **do begin**
3. **Apply** Algorithm 2 to upper i X i principal submatrix $\bar{A}$ of $A^{(i-3)}$, **obtaining** an orthogonal transformation $S^{(i)}$ and an i X i tridiagonal matrix $J^{(i)}=S^{(i)}\bar{A}S^{(i)T}$
4. **Accumulate** the orthogonal transformations in S , i.e.

$$\textbf{set } S := \begin{bmatrix} S^{(i)} & \Theta \\ \Theta & I \end{bmatrix} S,$$

(where Θ represents the zero matrix).

5. **Form** the next iterate $A^{(i-2)}$ by:

$$A^{(i-2)} := \begin{bmatrix} S^{(i)} & \Theta \\ \Theta & I \end{bmatrix} A^{(i-3)} \begin{bmatrix} S^{(i)T} & \Theta \\ \Theta & I \end{bmatrix} = \begin{bmatrix} J^{(i)} & \mathbf{e}_1\mathbf{v}^T \\ \mathbf{v}\mathbf{e}_1^T & D \end{bmatrix},$$

where $\mathbf{v}$ is some $(m-i)$-vector to be further discussed below, and D is an $(m-i)$ X $(m-i)$ diagonal matrix.

end

6. **Set** $J := A^{(m-2)}$.
(This matrix $A^{(m-2)}$ is tridiagonal, and the orthogonal transformations have been accumulated into S , so that $J=SAS^T$.)

end.

In step 5 at the i-th iteration, we have formed $A^{(i-2)}$, whose upper i X i principal submatrix $J^{(i)}$ is tridiagonal. We also note that all the rotations computed in step 3. in the call to Algorithm 2 involve orthogonal rotations which rotate among the planes 2, . . . , i ; they specifically do not involve plane 1. Hence the transformation $S^{(i)}$, as well as the accumulated transformation S , have the form

$$\begin{bmatrix} 1 & 0 \\ 0 & X \end{bmatrix}, \tag{2.8}$$

where X is an orthogonal matrix. A consequence of this is that the vector v and diagonal matrix D in, step 5 actually consist of the initial values already in those positions in $A^{(0)}$. We can show inductively that those positions are unchanged by the computation in step 5. We write

$$A^{(i-3)} = \begin{bmatrix} \bar{A} & \mathbf{e}_1\mathbf{v}^T \\ \mathbf{v}\mathbf{e}_1^T & D \end{bmatrix}.$$

Then the formula in step 5 yields

$$A^{(i-2)} = \begin{bmatrix} S^{(i)}\bar{A}S^{(i)T} & S^{(i)}\mathbf{e}_1\mathbf{v}^T \\ \mathbf{v}\mathbf{e}_1^T S^{(i)T} & D \end{bmatrix}.$$

Since $S^{(i)}$ has the form (2.8), the 2,1 block of $A^{(i-2)}$ satisfles $S^{(i)}\mathbf{e}_1\mathbf{v}^T=\mathbf{e}_1\mathbf{v}^T$. Similarly the 1,2 block is simply $\mathbf{v}\mathbf{e}_1^T$. So we may conclude that the only block affected by step 5 is the 1,1 block, and furthermore the contents $J^{(i)}$ of that block was already computed in step 3. Hence step 5 is

essentially free.

The cost of each pass through steps 3-5 of Algorithm 3 is entirely in step 3, that is approximately $30i$, so the total cost for i -3, . . . , m is approximately $15m^2$, ignoring lower order terms. We note Anally that the 1,1 element of the matrix $A^{(0)}$ (that is a, in (2.1)) is untouched and ignored by the entire computation.

2.4. Method 4 - Alternative Data.

In this section we describe an alternate way to set up the problem of reconstructing a Jacobi matrix, starting with the same initial data as in Section 2.1, but computing a different vector of intermediate data. This method is from [1] and also follows from a suggestion of [30]. This method does not require that the eigenvalues λ_i ,μ_i be mutually distinct. The property (1.1) is suffkient as long as the μ_i 's are simple. The idea is to first to construct an n X n bordered diagonal matrix A with the desired eigenvalues λ_i and μ_i and then to reduce this matrix to tridiagonal form using orthogonal transformations that do not affect this eigenvalue structure.

Specifically, this method consists of first computing the bordered diagonal matrix A deflned by (1.2), where the elements of A are deflned by (1.3) (1.4). From A we generate the tridiagonal matrix J that solves Problem J. To generate J, one may apply either Method 2 or 3 to A instead of A_{aug}. We see now that one can also compute J with this same alternative intermediate data by a scheme based on Method 1. By Proposition 1, we have the flrst row of the eigenvectors of the submatrix $\bar{J}$, namely $\delta/\|\delta\|$, where 6 is deflned by (1.4). So we may generate $\bar{J}$ by using the Lanczos Algorithm, starting with the matrix A4 (1.2) and the vector $\delta/\|\delta\|$. The remaining entries a_1, b_1 in J can then be computed from (1.3) and (1.7).

2.6. Modularity.

The methods we have described are modular to a certain extent. Each method consists of two parts: in the flrst part we compute a certain vector of data, and in the second we apply some matrix reduction algorithm to generate the tridiagonal matrix J. For each part, we have mentioned several choices of algorithms, and to a certain extent one is free to combine any choice for the flrst part with any choice for the second part. For the flrst part, we have mentioned two choices: either one can compute an n vector of weights d_i , i -1, . . . , n (1.9) which deflnes a bordered diagonal matrix (2.1), or one can compute the n -1 vector δ (1.4) deflning a bordered diagonal matrix A (1.2).

For the second part, we have described three possible algorithms based on, respectively, the Lanczos Algorithm, Householder transformations, and Rutishauser's fast reduction scheme. The Lanczos Algorithm takes $O(n^2)$ operations, $0(n^3)$ if we carry out re-orthogonalization, which is useful for numerical stability. The algorithm based on Householder transformations takes $0(n^3)$ operations, and the fast reduction algorithm (Rutishauser) takes $0(n^2)$ operations. Both of the latter two algorithms consist of applying a series of orthogonal similarity transformations to the matrix, so there is no problem with numerical stability.

In the rest of this paper, we will describe some generalizations and variations of the basic problem discussed in this section. It will be seen that the Lanczos Algorithm and the Householder reduction scheme can easily be applied to banded and periodic problems.

3. Banded Matrices.

In this section we extend the methods for Jacobi matrices to the problem of reconstructing band matrices. Before deflning the problem, we must first deflne some notation. In this section we let J denote a symmetric n X n banded matrix with $2p+1$ bands, p below the diagonal. We call such a matrix a "p -banded matrix", and say it has "half-bandwidth p ". Let $J^{(k)}$ denote the lower $(n-k)\times(n-k)$ principal submatrix of J, so that $J^{(0)}=J$, and $J^{(1)}$ corresponds to $\overline{J}$ of Sections 1 and 2. Assume the following identities:

$$J^{(k)} = \begin{bmatrix} a_{k+1} & \mathbf{b}^{(k)T} \\ \mathbf{b}^{(k)} & J^{(k+1)} \end{bmatrix}, \quad k=0, \ldots, p-1, \tag{3.1}$$

and

$$Q^{(k)T} J^{(k)} Q^{(k)} = \Lambda^{(k)}, k=0, \ldots, p, \tag{3.2}$$

where $\Lambda^{(k)}=\mathrm{diag}(\lambda_1^{(k)}, \ldots, \lambda_{n-k}^{(k)})$ is the diagonal matrix of eigenvalues of $J^{(k)}$, and $Q^{(k)}=[\mathbf{q}_1^{(k)} \cdots \mathbf{q}_{n-k}^{(k)}]$ is the orthogonal matrix of eigenvectors of $J^{(k)}$. We denote the (i, j)-th element of $Q^{(k)}$ by $q_{i,j}^{(k)}$.

The problem is then as follows

Problem B: Given the $p+1$ sets of eigenvalues $\{\lambda_j^{(k)}\}$, k -0, . . . , p , satisfying the interlacing property

$$\lambda_j^{(k)} > \lambda_j^{(k+1)} > \lambda_{j+1}^{(k)}, \; j=1, \ldots, n-k-1, \; k=0, \ldots, p-1; \tag{3.3}$$

construct a J with half-bandwidth p such that each submatrix $J^{(k)}$ has eigenvalues $\{\lambda_j^{(k)}\}$, for $k=0, \ldots, p$.

To compute such a J , we use Proposition 1. For each submatrix $J^{(k)}$, we can deflne a bordered diagonal matrix $A^{(k)}$ corresponding to (1.2):

$$A^{(k)} = \begin{bmatrix} a_{k+1} & \mathbf{b}^{(k)T} \\ \mathbf{b}^{(k)} & \Lambda^{(k+1)} \end{bmatrix}, \; (n-k)\times(n-k), \; k=0, \ldots, p-1, \tag{3.4}$$

where $A^{(k)}$ has eigenvalues $\{\lambda_j^{(k)}\}$. The value a_{k+1} is determined, as in (1.3), by a trace argument:

$$a_{k+1} = \sum_{j=1}^{n-k} \lambda_j^{(k)} - \sum_{j=1}^{n-k-1} \lambda_j^{(k+1)}, \tag{3.5}$$

and hence the first k diagonal entries of A are unique. Using the same development used for (1.4) based on the characteristic polynomial of $A^{(k)}$, we can determine the vectors $\mathbf{b}^{(k)}$, $k=0, \ldots, p-1$, by the formula

$$[b_i^{(k)}]^2 = -\frac{\prod_{j=1}^{n-k} (\lambda_i^{(k+1)} - \lambda_j^{(k)})}{\prod_{\substack{j=1 \\ j \neq i}}^{n-k-1} (\lambda_i^{(k+1)} - \lambda_j^{(k+1)})} \geq 0. \tag{3.6}$$

We Anally note that, as in equation (1.6), $A^{(k)}$ is related to $J^{(k)}$ by

$$A^{(k)} = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & Q^{(k+1)T} \end{bmatrix} J^{(k)} \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & Q^{(k+1)} \end{bmatrix}, \tag{3.7}$$

for k -0, . . . , p -1. We may conclude from this that

$$Q^{(k+1)T} \mathbf{b}^{(k)} = \mathbf{b}^{(k)}, \; k=0, \ldots, p-1. \tag{3.8}$$

In this case, the vector $\mathbf{b}^{(k)}$ is no longer a multiple of the unit vector $\mathbf{e}_1$, as it was in the tridiagonal case. Hence it is no longer true that the vector $\mathbf{b}^{(k)}$ is the same as the first row of the matrix $Q^{(k+1)T}$ of eigenvectors of $J^{(k+1)}$.

We may also use the second part of Proposition 2 to arrive at a formula for the **first** row $\mathbf{d}^{(k)}$ of the matrix of eigenvectors $\mathbf{Q}^{(k)}$, $k=0, \ldots, p-1$:

$$[d_j^{(k)}]^2 = [q_{1j}^{(k)}]^2 = \frac{\prod_{i=1}^{n-k-1} (\lambda_i^{(k+1)} - \lambda_j^{(k)})}{\prod_{\substack{i=1 \\ i \neq j}}^{n-k} (\lambda_i^{(k)} - \lambda_j^{(k)})} . \tag{3.9}$$

The intermediate data consisting of the matrices (3.2) and vectors (3.9) form the input data to the algorithms that actually generate solutions to Problem B. The rest of this section is devoted to alternative algorithms to generate the solutions.

3.1. Block Lancsos.

We have now deflned enough quantities to be able to describe the first method we can use to generate the banded matrix J. This method is based on the use of the Block Lanczos **Algorithm** (Algorithm 4). To carry out this method, we need to compute the first p rows of the **eigenvector** matrix $\mathbf{Q}^{(0)}$.

For each $k=0, \ldots, p$-1, we develop a formula for the flrst $p-k$ rows of $\mathbf{Q}^{(k)}$ in terms of the **first** rows $\mathbf{d}^{(k)\,T}$ which we already know. That is, we compute the flrst few components of each eigenvector $\mathbf{q}_j^{(k)}$ in terms of its flrst component $d_j^{(k)} = q_{1j}^{(k)}$.

Partition the eigenvector $\mathbf{q}_j^{(k)}$ **as**

$$\mathbf{q}_j^{(k)} = \begin{bmatrix} d_j^{(k)} \\ \mathbf{y}_j^{(k)} \end{bmatrix}, \tag{3.10}$$

where $\mathbf{y}_j^{(k)}$ is an $n-k$ -l-vector. From the identity $J^{(k)}\mathbf{q}_j^{(k)} = \lambda_j^{(k)}\mathbf{q}_j^{(k)}$, we use (3.1) to extract all but the first row of (3.10) to get:

$$\mathbf{b}^{(k)} d_j^{(k)} + J^{(k+1)}\mathbf{y}_j^{(k)} = \lambda_j^{(k)}\mathbf{y}_j^{(k)}. \tag{3.11}$$

Multiply by $\mathbf{Q}^{(k+1)\,T}$ to obtain

$$d_j^{(k)} Q^{(k+1)\,T}\mathbf{b}^{(k)} + \Lambda^{(k+1)} Q^{(k+1)\,T}\mathbf{y}_j^{(k)} = \lambda_j^{(k)} Q^{(k+1)\,T}\mathbf{y}_j^{(k)}.$$

Using (3.8) and solving for $\mathbf{y}_j^{(k)}$, we get

$$\mathbf{y}_j^{(k)} = -d_j^{(k)} Q^{(k+1)}(\Lambda^{(k+1)} - \lambda_j^{(k)} I)^{-1}\mathbf{b}^{(k)}.$$

Written in in terms of the individual elements, the above is

$$(i-1)\text{-st entry of } \mathbf{y}_j^{(k)} \equiv q_{ij}^{(k)} = -d_j^{(k)} \sum_{l=1}^{n-k-1} \frac{q_{i-1,l}^{(k+1)} b_l^{(k)}}{\lambda_l^{(k+1)} - \lambda_j^{(k)}}, \tag{3.12}$$

where $i=2, \ldots, n-k$, $j=1, \ldots, n-k$, $k=p-2, p-3, \ldots, 0$. Recall from (3.9) that when i takes on its **first value** 2, the values $q_{1l}^{(k+1)} \equiv d_l^{(k+1)}$ are known. When $k=p$ -2, the right hand side of (3.12) is completely known for i -2, so we can obtain the **first** 2 rows of $Q^{(p-2)}$. Then when $k=p$ -3, we 'know the right hand side for $i=2,3$, so we can obtain the first 3 rows of $Q^{(p-3)}$. We continue in this way until k -0, at which point we will have the **first** p rows of $Q^{(0)}$.

Once we have the flrst p rows of $Q^{(0)}$, we can now carry out the Block Lanczos Algorithm. Let $\mathbf{x}_i^T = [q_{i1}^{(0)}, \ldots, q_{in}^{(0)}]$ denote the i-th row of $\mathbf{Q}^{(0)} \equiv Q$, so $\mathbf{x}_j$ is the j-th column of the the matrix $Q^T \equiv X$. Let $X_1 = [\mathbf{x}_1 \cdots \mathbf{x}_p]$ be an $n \times p$ matrix consisting of the flrst p columns of X, so that $X_1^T X_1 = I_{p \times p}$. The Block Lanczos Algorithm is then carried out with starting matrix A and p starting vectors $X_1 = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$.

The result will be the $n \times n$ orthogonal matrix $X = [X_1, \ldots, X_s]$ and the p -banded matrix J of the form **(1.23)**, where $n = ps$. Because the **first** p rows of the eigenvector matrix of J have been determined from the eigenvalue requirements, in a manner analogous to the Jacobi case, it is a simple matter to show that the banded J produced by algorithm **4** will indeed solve

Problem B.

Note that, we must require that p divide n , and also that the interlacing among the eigenvalues be strict, i.e. that the inequalities in (3.3) be strict.

To summarize: the complete algorithm is as follows:

Algorithm 6. To solve Problem B:

1. **Compute** $\mathfrak{b}^{(0)}, \ldots, \mathfrak{b}^{(p-2)}$ by taking the square root of (3.6).
2. **Compute** $\mathbf{d}^{(0)}, \ldots, \mathbf{d}^{(p-1)}$ by taking the square root of (3.9).
3. **for** $k = p-2, \ldots, 0$ **do**
   for $j = 1, \ldots, n-k$ **do**
   for $i = 2, \ldots, p-k$ **do**
   compute $i-1$-**st** entry of $\mathbf{y}_j^{(k)} = q_{ij}^{(k)}$ by (3.12).
4. **Define** the starting vectors:

$$\mathbf{x}_1 := \mathbf{d}^{(0)}, \text{ and}$$
$$\mathbf{x}_i := [q_{i\,1}^{(0)}, \ldots, q_{in}^{(0)}], \text{ for } i = 2, \ldots, p.$$

5. **Apply** Algorithm 4 (Block Lanczos) to generate the matrix J, starting with `matrix` A and the vectors $X_1 \equiv [\mathbf{x}_1, \ldots, \mathbf{x}_p]$.

Different choices of signs for the square roots in steps 1 and 2 yield different, and possibly non-equivalent, solutions. We can obtain in this fashion a large but flnite set of possible solutions to Problem B. In case of rank-degeneracy during the Block Lanczos process, one may obtain a continuum of solutions. An example is the following 4 X 4 matrix with $p = 2$ [18] [19] :

$$J(\gamma) = \begin{vmatrix} \alpha_1 & \beta_1 & 0 & 0 \\ \beta_1 & \alpha_2 & -\sin\gamma & \cos\gamma \\ 0 & -\sin 7 & 1-\sin 27 & \cos 2\gamma \\ 0 & \cos\gamma & \cos 2\gamma & 1+\sin 2\gamma \end{vmatrix},$$

for which, assuming $\beta_1 \neq 0$, the eigenvalues of J, $J^{(1)}$, $J^{(2)}$ are all constants independent of 7.

We note in passing that if $p = 1$, the problem reduces to the tridiagonal case discussed in Section 2. In this case, steps 1, 3, 4 of Algorithm 5 become empty: only steps 2 and 5 remain to do. The result is the same method as Method 1 (Lanczos) for Problem J described in Section 2.

3.2. Orthogonal Reduction.

We now turn to an alternate method for generating a band matrix solving Problem B. This method, based on the use of Householder transformations, was originally proposed for this problem by [1]. This method does not require that the interlacing among the eigenvalues be strict (3.3), nor does it require that p divide n exactly. On the other hand it is more involved and somewhat more expensive in that it requires the solution of several eigenvalue problems.

This method begins by forming the bordered diagonal matrices $A^{(k)}$ (3.4). We let $P^{(k)}$ denote the orthogonal matrix of eigenvectors of $A^{(k)}$, and $\mathbf{Q}^{(k)}$ denote the matrix of eigenvectors of the submatrix $J^{(k)}$, k -0, . . . , p . By Proposition 2, $P^{(k)}$ and $\mathbf{Q}^{(k)}$ have the same first row, up to signs. We defer for the moment the discussion on how to compute the $P^{(k)}$.

The intermediate goal of this method is to construct a sequence of n X n matrices $H^{(0)}, H^{(1)}, \ldots, H^{(p-1)} \equiv H$, culminating in a matrix H which has the eigenstructure demanded by Problem B with respect to H and its first p principal submatrices, but is not banded. Then this matrix H is reduced to banded form by orthogonal similarity transformations in such a way as not to destroy the eigenstructure desired. In this paper we will only indicate the major steps. The interested reader is referred to the paper of [1].

We begin the sequence of matrices by setting $H^{(0)} := A^{(0)}$ deflned by (3.4). We form $H^{(1)}$ by

$$H^{(1)} = \begin{bmatrix} 1 & 0^T \\ 0 & P^{(1)} \end{bmatrix} H^{(0)} \begin{bmatrix} 1 & 0^T \\ 0 & P^{(1)\,T} \end{bmatrix}.$$

The matrix $H^{(1)}$ will have the form

$$H^{(1)} = \begin{bmatrix} a_1 & * \\ * & A^{(1)} \end{bmatrix} = \begin{bmatrix} *_{2\times 2} & * \\ * & \Lambda^{(2)} \end{bmatrix} \tag{3.13}$$

where $*$ denotes a sub-block of the matrix that we do not specify explicitly. We then form $H^{(2)}$ by

$$H^{(2)} = \begin{bmatrix} I_{2\times 2} & \Theta \\ \Theta & P^{(2)} \end{bmatrix} H^{(1)} \begin{bmatrix} I_{2\times 2} & \Theta \\ \Theta & P^{(2)\,T} \end{bmatrix} = I \begin{bmatrix} *_{2\times 2} & * \\ * & A^{(2)} \end{bmatrix}, \tag{3.14}$$

where Θ is a matrix of zeroes. After p -1 steps we will have the matrix

$$H \equiv H^{(p-1)} = \begin{bmatrix} *_{p-1\times p-1} & \Theta \\ \Theta & A^{(p-1)} \end{bmatrix} = \begin{bmatrix} H_p & B^T \\ B & \Lambda^{(p)} \end{bmatrix} \tag{3.15}$$

where H_p *is* $p \times p$. We examine the submatrices of the sequence $H^{(0)}, H^{(1)}, \ldots, H^{(p-1)} \equiv H$, corresponding to the submatrices $J^{(0)}, \ldots, J^{(k)}$ of J. We see that at each stage, the transformation was constructed just so the first $k+1$ of those submatrices of $H^{(k)}$ respectively have eigenvalues $\{\lambda_j^{(i)}\}$, $i = 0, \ldots, k$. Hence, at the end, $H \equiv H^{(p-1)}$ satisfies the eigenvalue requirements demanded by Problem B, with respect to H itself and its first p principal submatrices.

From (3.15), it is clear that H is not banded, rather it is "block" bordered diagonal. To reduce H to a p -banded matrix (i.e. half-bandwidth p) by orthogonal similarity transformations, we may apply a series of plane rotations in planes $p+1, \ldots, n$ or Householder transformations in a manner very similar to the reduction algorithm TRED2 [8]. The transformations applied in this way will have the form

$$\begin{bmatrix} I_{p\times p} & \Theta \\ \Theta & U_i \end{bmatrix} \quad n \times n, \tag{3.16}$$

where U_i is either an orthogonal plane rotation or Householder transformation. It can be verified that such transformations will indeed not affect the eigenvalue structure demanded by Problem B.

Finally, we give the process by which we compute the matrices $P^{(k)}$, consisting of the eigenvectors of $A^{(k)}$. The simplest conceptually is to use a standard symmetric eigenvalue/vector solver on the $\{A^{(k)}\}$; for example one might use the EISPACK [8] routines **TRED2** followed by **TQL2**. The **TRED2** part is $0(n^3)$, and the **TQL2** part is $0(n^2)$. Since $A^{(k)}$ is a bordered diagonal matrix of the form (1.2), we can reduce the cost of this step to $0(n^2)$ by replacing the TRED2 part with Algorithm 3 (Rutishauser).

In summary, the method is then as follows:

Algorithm 8. Solve Problem B.

1. for $i = 0, \ldots, p-1$ **do**
2. **Compute** $A^{(k)}$ by (3.4) (3.5) (3.6).
3. **Compute** $P^{(k)}$, eigenvectors of $A^{(k)}$ using Algorithm 3 (Rutishauser) and **TQL2** [8].
4. **Compute** the sequence $H^{(k)}$, $k = 0, \ldots, p-1$, ending with $H \equiv H^{(p-1)}$.
5. **Apply** sequence of plane rotations or Householder similarity transformations of the form (3.16) to reduce H to a banded matrix J of half-bandwidth p, without destroying the eigenvalues structure carefully assembled in H.

Though steps 1-4 are relatively fast, it is not obvious to the authors how to reduce step 5 from $0(n^3)$ to $0(n^2)$. This will be reserved for future work.

4. Periodic Jacobi Matrices.

In this section, we apply Proposition 1 to the problem of reconstructing a periodic Jacobi matrix given certain eigenvalue data. We follow our usual notation: let J be a Jacobi matrix, $\bar{J}$ be its lower $(n-1)\times(n-1)$ principal submatrix. We would like to solve the following problem:

Problem P. Compute a periodic Jacobi matrix

$$J_{per} = \begin{bmatrix} a_1 & b_1 & & & b_n \\ b_1 & a_2 & \ddots & & \\ & \ddots & \ddots & 0 & \\ & & \ddots & & \\ & 0 & \ddots & \ddots & \\ & & & a_{n-1} & b_{n-1} \\ b_n & & & b_{n-1} & a_n \end{bmatrix}. \tag{4.1}$$

We call J_{per}^- the matrix (4.1) with b_n replaced with $(-b_n)$. To start with, we are given the eigenvalue data: $\{\lambda_i\}_1^n$ eigenvalues of J_{per}, and $\{\mu_i\}_1^{n-1}$ eigenvalues of $\bar{J}$. In addition, we are given one of two sets of data: either the set $\{\lambda_i^-\}_1^n$ eigenvalues of J_{per}^-, or the single product $\beta = b_1 \ldots b_n$. In the latter case, the number of items of input data $(2n)$ is identical to the number of elements to compute $(2n)$.

4.1. Preliminary Construction.

In all the methods we propose, we start as before by computing the bordered diagonal matrix A of the form (1.2)

$$A \equiv \begin{bmatrix} a_{11} & b^T \\ b & M \end{bmatrix} = \begin{bmatrix} 1 & 0^T \\ 0 & \bar{Q}^T \end{bmatrix} J \begin{bmatrix} 1 & 0^T \\ 0 & \bar{Q} \end{bmatrix} \tag{4.2}$$

where $\bar{Q}$ is the orthogonal matrix of eigenvectors of $\bar{J}$ (1.6), and the a_{11} and b are deflned by (1.3) and (1.4) in terms of the eigenvalues $\{\lambda_i\}_1^n$, and $\{\mu_i\}_1^{n-1}$. Specifically,

$$a_{11} = \text{trace(A)} - \text{trace(M)} = \sum_1^n \lambda_i - \sum_1^{n-1} \mu_i , \tag{4.3}$$

and

$$b_i^2 = -\frac{\prod_{j=1}^{n} (\mu_i - \lambda_j)}{\prod_{\substack{j=1 \\ j \neq i}}^{n-1} (\mu_i - \mu_j)} \geq 0. \tag{4.4}$$

From the $\{\lambda_i^-\}_1^n$, we also compute the bordered diagonal matrix A^- corresponding to (4.2)

$$A^- \equiv \begin{bmatrix} a_{11}^- & b^{-T} \\ b^- & M \end{bmatrix} = \begin{bmatrix} 1 & 0^T \\ 0 & \bar{Q}^T \end{bmatrix} J^- \begin{bmatrix} 1 & 0^T \\ 0 & \bar{Q} \end{bmatrix} \tag{4.5}$$

where $\bar{Q}$ is the same as in (4.2), and the a_{11}^- and b^- are defined as in (4.3) (4.4) using the $\{\lambda_i^-\}_1^n$ in place of the $\{\lambda_i\}_1^n$.

We now show how to compute A^- using β instead of the $\{\lambda_i^-\}_1^n$. We can expand determinants to obtain the formulas:

$$\det(\lambda I - J_{per}) \equiv \prod_{j=1}^{n} (\lambda - \lambda_j) = p_n(\lambda) - b_n^2 p_{n-2}(\lambda) - 2\beta$$

$$\det(\lambda I - J_{per}^-) \equiv \prod_{j=1}^{n} (\text{X-X;}) = p_n(\lambda) - b_n^2 p_{n-2}(\lambda) + 2\beta,$$

where $p_n(\lambda)$ and $p_{n-2}(\lambda)$ are the characteristic polynomials of J and $\tilde{J}$ as deflned in Section 1.3, except that $\tilde{J}$ is the (n -2) $X(n-2)$ principal submatrix consisting of rows and columns 2, . . . , n -1 of J . Subtracting gives

$$\det(\lambda I - J_{\text{per}}^{-}) = \det(\lambda I - J_{\text{per}}^{-}) + 4\beta$$

Using this expression, we can rewrite the formula for the vector $\hat{b}^{-}$:

$$(\hat{b}_i^{-})^2 = -\frac{\prod_{j=1}^{n}(\mu_i - \lambda_j) + 4\beta}{\prod_{\substack{j=1 \\ j \neq i}}^{n-1}(\mu_i - \mu_j)} \geq 0 \tag{4.6}$$

The use of (4.3) (4.4) (4.6) requires only that the interlacing property (1.1) hold and that the $\{\mu_i\}_1^{n-1}$ be simple.

We can write the subblocks of (4.2) (4.5) corresponding to $\hat{b}^T$ as

$$[b_1, 0, \ldots, 0, b_n]\bar{Q} = \hat{b}^T \tag{4.7}$$
$$[b_1, 0, \ldots, 0, -b_n]\bar{Q} = \hat{b}^{-T}$$

Subtracting and transposing yield

$$\bar{Q}^T [0, \ldots, 0, 2b_n]^T = 2b_n \bar{Q}^T \mathbf{e}_{n-1} = (\hat{b} - \hat{b}^{-})$$

This simplifies to

$$2b_n \mathbf{x}_{n-1} = h\text{-}6\text{-} \tag{4.8}$$

where $\mathbf{x}_1^T, \ldots, \mathbf{x}_{n-1}^T$ are the n -1 rows of the matrix $\bar{Q}$. If instead we add the two equations (4.7), we get

$$2b_1 \mathbf{x}_1 = \hat{b} + \hat{b}^{-} \tag{4.9}$$

The last two equations determine b_1 and b_n (up to sign) from the fact that $\|\mathbf{x}_1\| = \|\mathbf{x}_{n-1}\| = 1$.

4.2. Generation of Periodic Matrix.

At this point, we have enough information to actually generate the periodic Jacobi matrix we are seeking to compute. The most straightforward algorithm is a simple variation to the scheme proposed in Section 2.4. This scheme was originally adapted from [9].

Algorithm 7. Given the two sets $\{\lambda_i\}_1^n$ and $\{\mu_i\}_1^{n-1}$ satisfying (1.1) (with the μ_i simple), and the number β, construct a periodic Jacobi matrix J_{per} (4.1) solving Problem P.

1. **Compute** $\hat{b}$ and $\hat{b}^{-}$ from (4.4) (4.6).
2. **Compute** the flrst row $\mathbf{x}_1$ of the matrix $\bar{Q}$ of eigenvectors of $\bar{J}$ using (4.9). Also obtain value for b_1.
3. Starting with the vector $\mathbf{x}_1$ and the set $\{\mu_i\}_1^{n-1}$, **Apply Algorithm 1** (Lanczos) to generate the matrix $\bar{J}$. Or we may use the alternate scheme mentioned below.
4. **Compute** a_1 from (4.3).
5. **Compute** b_n from (4.8), or else from the relation

$$b_n := \frac{\beta}{b_1 \cdots b_{n-1}}$$

In step 3, we may instead form the $n \times n$ bordered diagonal matrix

$$\begin{bmatrix} a_{11} & \mathbf{x}_1^T \\ \mathbf{x}_1 & \text{ii} \end{bmatrix},$$

where $M = \mathbf{diag}(\mu_1, \ldots, \mu_{n-1})$ and a_{11} is a dummy value. In a manner analogous to the methods for Problem J, we may replace the Lanczos Algorithm in step 3 and instead generate $\bar{J}$ by applying Householder transformations or Algorithm 3 (Rutishauser) to this bordered diagonal matrix.

4.3. Alternate Method.

We mention briefly an alternate method, usable if the order n of the matrix is even. This method is derived from the fact that by a suitable permutation, a periodic Jacobi matrix may be permuted into a pentadiagonal band matrix (i.e. *half-bandwidth* $p = 2$). Specifically, if we permute both the rows and columns by the permutation

$$(1, \ldots, n) \rightarrow (1, 3, \ldots, n-1, n, n-2, \ldots, 4, 2),$$

the periodic matrix J_{per} is permuted into a pentadiagonal matrix we will call J_{penta}. Let Q_{per} and Q_{penta} be the orthogonal eigenvector matrices of J_{per} and J_{penta} respectively. From the construction of J_{penta}, it follows that the matrix Q_{penta} can be obtained by simply applying the above permutation to the *rows* of Q_{per}. In applying this permutation to J, the $(n-1) \times (n-1)$ submatrix $\bar{J}$ is mapped onto itself.

In order to generate the matrix J_{penta} according to the methods of Section 3, we need the first 2 rows of Q_{penta}, which are the same as the first and last rows of Q_{per}. To clarify the connection with Section 3, we note that the matrix J_{penta} corresponds to $J^{(0)}$ in the notation of the Section 3, the eigenvector matrix Q_{penta} corresponds to $Q^{(0)}$, and the two sets of eigenvalues $\{\lambda_i\}_1^n$ and $\{\mu_i\}_1^{n-1}$ correspond to the two sets $\{\lambda_i^{(0)}\}_1^n$ $\{\lambda_i^{(1)}\}_1^{n-1}$, respectively. Formula (4.9) gives the first row of $Q^{(1)}$, hence we may use (3.12) to give to first two rows of $Q_{penta} \equiv Q^{(0)}$, which are the same as the flrst and last rows of Q_{per}.

We can now apply Algorithm 4 (block Lanczos) to generate J_{penta}. However, it is easier to re-arrange the algorithm to generate J_{per} and Q_{per} directly. For the remainder of this section, we denote the rows of Q_{per} by $\mathbf{x}_1^T, \ldots, \mathbf{x}_n^T$. From the above discussion, we know the two vectors $\mathbf{x}_1$ and $\mathbf{x}_n$.

Algorithm 10. - Periodic Lanczos Algorithm. Given $\Lambda = \mathbf{diag}(\lambda_1, \ldots, \lambda_n)$ and the two orthonormal $\mathbf{x}_1, \mathbf{x}_n$, compute a periodic Jacobi matrix J_{per} (4.1) and an orthogonal matrix $X = [\mathbf{x}_1 \cdots \mathbf{x}_n]$ such that $\Lambda = XJX^T$.

```
1.  begin
2.  b_0 := b_n := x_n^T Λ x_1
3.  a_1 := x_n^T Λ x_n
4.  for i = 1, ..., n-1 do
        begin
5.      a_i := x_i^T Λ x_i
6.      z_i := Λx_i - x_i a_i - x_{i-1} b_{i-1}
7.      Compute x_{i+1}, b_i so that x_{i+1} b_i = z_i, x_{i+1}^T x_{i+1} = 1
        end
    end
```

If the given data for Problem P is such that a solution J_{per} exists, the solution is unique up to choice of signs for the vectors $\mathbf{d}^{(0)}$ and $\mathbf{b}^{(1)}$, respectively the first rows of $\mathbf{Q}_{per}$ and $\bar{Q}$. Hence Algorithm 10 will yield a matrix solving Problem P. The Appendix of [3] shows in detail that this algorithm does indeed construct the desired matrix.

6. Miscellaneous Problems.

In this section, we briefly mention how the methods of this paper can be used to solve certain miscellaneous inverse problems. We discuss two problems which were studied in [7]:

Problem R (Rank one update). Given two strictly increasing sequences $\{\lambda_i\}_1^n$ and $\{\lambda_i^*\}_1^n$, with $\lambda_i < \lambda_i^*$ for all i, compute an n X n Jacobi matrix J with eigenvalues $\{\lambda_i\}_1^n$, such that if the 1-1 element of J, a_1 is replaced with a_1^*, the resulting matrix J^* will have eigenvalues $\{\lambda_i^*\}_1^n$.

Problem S (perSymmetric problem). Given a strictly increasing sequence $\{\lambda_i\}_1^n$, construct a persymmetric Jacobi matrix J having eigenvalues $\{\lambda_i\}_1^n$.

A matrix is ***persymmetric*** if it is symmetric around its anti-diagonal, i.e.

$$a_{ij} = a_{n-1-i,n+1-j}.$$

A persymmetric Jacobi matrix satisfies

$$a_i = a_{n+1-i}, \quad b_i = b_{n-i}.$$

We also study the following problem:

Problem DD (Double Dimension). Given an n X n Jacobi matrix J_n and a set of ***distinct*** eigenvalues $\{\lambda_i\}_1^{2n}$, construct a $2n \times 2n$ Jacobi matrix J_{2n} whose eigenvalues are the given values $\{\bar{\lambda}_i\}_1^{2n}$, and whose leading n X n principal submatrix is exactly J_n.

Solution to Problem R.

We define the polynomials as in Section 1.3: let $p_n(\lambda)$ and $p_{n-1}(\lambda)$ be the characteristic polynomials of J and $\bar{J}$, respectively. We also let $p_n^*(\lambda)$ be the characteristic polynomial of J^*. We can expand the determinant of J to obtain the relation (1.11):

$$p_n(\lambda) = (a_1 - \lambda)p_{n-1}(\lambda) - b_1^2 p_{n-2}(\lambda). \tag{5.1}$$

Analogously, we obtain a similar formula for J^*:

$$p_n^*(\lambda) = (a_1^* - \lambda)p_{n-1}(\lambda) - b_1^2 p_{n-2}(\lambda). \tag{5.2}$$

Subtracting yields

$$p_n(\lambda) - p_n^*(\lambda) = (a_1 - a_1^*)p_{n-1}(\lambda) = \left[\text{trace}(J) - \text{trace}(J^*)\right] p_{n-1}(\lambda),$$

which we can write directly in terms of the eigenvalues:

$$p_n(\lambda) - p_n^*(\lambda) = p_{n-1}(\lambda) \sum_{j=1}^{n} (\lambda_j - \lambda_j^*). \tag{5.3}$$

The polynomials p_n and p_n^* are known from the given data, and the polynomial p_{n-1} can be computed from (5.3), so we may use formula (1.12) to obtain the first row **d** of the matrix of eigenvectors of J. With this information, we may apply any of Algorithm 1 (Lanczos), TRED2, or Algorithm 3 (Rutishauser) to generate the matrix J. The quantity a_1^* may be computed from

$$a_1^* - a_1 = \text{trace}(J^*) - \text{trace}(J),$$

thus defining J^*.

Solution to Problem S.

It has been shown ([7], Lemma 2) that

$$p_{n-1}(\lambda_j)\bar{p}_{n-1}(\lambda_j) = [b_1 \cdots b_{n-1}]^2 \equiv \gamma^2, \; j = 1, \ldots, n, \tag{5.4}$$

where 7 is a constant, and p_{n-1} and $\bar{p}_{n-1}$ are, the characteristic polynomials of the lower and upper, respectively, $(n-1) \times (n-1)$ principal submatrices of J, i.e. the submatrices obtained by deleting the first row and column or the last row and column, respectively. For persymmetric J, $p_{n-1} = \bar{p}_{n-1}$, so we have

$$p_{n-1}(\lambda_j) = \pm\gamma, j = 1, \ldots, n. \tag{5.5}$$

We use these equalities to compute the vector $\mathbf{d}$ in (1.12). The denominator of (1.12) is determined by the given data $\{\lambda_i\}_1^n$ and can also be written as in (1.9). We pick the signs of 7 to make *(1.12)* positive for each j . In using **(1.12)**, we do not need to know the polynomial p_{n-1}; all we need are the specific values *(5.5)* at the given points λ_j, $j = 1, \ldots, n$. In fact, we do not need to know 7 itself; we can use (1.12) to compute the values d_j^2 up to a scalar multiple and then scale the vector $\mathbf{d}$ to have norm 1: $d_1^2 + \cdots + d_n^2 = 1$. As in the solution for Problem R, we now have sufficient information to apply any of the algorithms of Section 2 to generate J. If such a solution exists, we are guaranteed by the uniqueness result (Proposition 2(a)) that the matrices generated using this intermediate data will indeed solve Problem S.

Solution to Problem DD.

We derive the solution in terms of the relationship to Gaussian Quadrature discussed in Section 1.3. In fact, this problem corresponds exactly to the problem of computing the Gauss Quadrature Formula of order $2n$ (exact for polynomials of degree up to $4n-1$), given the Formula of order n (exact for polynomials of degree up to $2n-1$). Since both formulas are exact for polynomials of degree up to n , the flrst n moments (1.20) from either formula must agree:

$$\sum_{s=1}^{n} w_s \lambda_s^k = m_k = \sum_{i=1}^{2n} \bar{w}_i \bar{\lambda}_i^k, \tag{5.6}$$

where w_s, λ_s are the weights and nodes for the formula of order n , respectively, and $\bar{w}_i$, $\bar{\lambda}_i$ are the same for the formula of order $2n$. Here, the $\bar{\lambda}_i$ are the given values, and the λ_s , w, can be computed easily from the given J_n , where the w, are obtained from the eigenvectors of J_n by (1.19). It remains to-solve for the $\bar{w}_i$. If the $\bar{w}_i$ are positive, then by (1.19) they give the leading components of the eigenvectors of J_{2n} . One can then apply any of the methods discussed in Section 2 to flnally generate the Jacobi matrix J_{2n} . Hence we have solved Problem DD if we can find a positive solution $\bar{w}_i$, $i = 1, \ldots, 2n$, to the system of equations *(5.6)*.

The system of equations *(5.6)* is linear in the $\bar{w}_i$, but is very ill-conditioned. Therefore, instead of solving it by a standard linear system solver, we follow the suggestions of [6] based on the use of Lagrange Interpolation. It is well known (see e.g. [5]) that if one interpolates a function $f(x)$ at the m nodes $z_1, \ldots, z_m$ using the Lagrange Interpolating Polynomials, one obtains the formula

$$f(x) = \sum_{i=1}^{m} \frac{\prod_{j \neq i} (z_j - x)}{\prod_{j \neq i} (z_j - z_i)} f(z_i), \tag{5.7}$$

where equality is exact if f is a polynomial of degree at most m . To solve (5.6), we set $m = 2n$, set the knots to be $z_i = \bar{\lambda}_i$, $i = 1, \ldots, 2n$, and set the function to be $f(x) = x^k$. In this case *(5.7)* becomes

$$x^k = \sum_{i=1}^{2n} \frac{\prod_{j \neq i} (\bar{\lambda}_j - x)}{\prod_{j \neq i} (\bar{\lambda}_j - \bar{\lambda}_i)} \bar{\lambda}_i^k.$$

Setting $x = \lambda_s$, $s = 1, \ldots, n$, the eigenvalues of J_n , we obtain the formula

$$\lambda_s^k = \sum_{i=1}^{2n} \bar{\lambda}_i^k \theta_{si}, \tag{5.8}$$

where

$$\theta_{si} = \frac{\prod_{j \neq i} (\bar{\lambda}_j - \lambda_s)}{\prod_{j \neq i} (\bar{\lambda}_j - \bar{\lambda}_i)}. \tag{5.9}$$

From *(5.8)* we form the weighted sum of the λ_s^k to obtain the formula

$$\sum_{s=1}^{n} w_s \lambda_s^k = \sum_{s=1}^{n} w_s \left[\sum_{i=1}^{2n} \bar{\lambda}_i^k \theta_{si} \right] = \sum_{i=1}^{2n} \bar{\lambda}_i^k \left[\sum_{s=1}^{n} w_s \theta_{si} \right].$$

From this formula, it is easy to see that the solution to the system of equations (5.6) is given by

$$\bar{w}_i \equiv \sum_{s=1}^{n} w_s \theta_{si}, \tag{5.10}$$

where the θ_{si} are defined by (5.9). Thus, if the $\bar{w}_i$ are positive, they define the first components of the eigenvectors of J_{2n} :

$$\bar{q}_{1i}^2 = \bar{w}_i .$$

We then have sufficient information to apply any of the algorithms of Section 2 to obtain J_{2n} : Algorithm 1 (Lanczos), TRED2, or Algorithm 3 (Rutishauser). Note, we have not given conditions under which a solution exists, but our procedure will yield positive values for the weights $\bar{w}_i$ if and only if a solution exists.

ACKNOWLEDGEMENT

We have benefltted greatly by discussions and communications with Victor Barcilon, Carl de Boor, Ole Hald, and John Thompson.

BIBLIOGRAPHY

[1] Biegler-König F W: Construction of band matrices from spectral data; Lin Alg & Appl 40, pp79-84, Oct. 1981.

[2] Boley D L, Golub G H: Inverse eigenvalue problems for band matrices; Lecture notes on Mathematics, Numerical Analysis, Dundee, 1977, Springer Verlag.

[3] Boley D L, Golub G H: The Matrix Inverse Eigenvalue Problem for Periodic Matrices; invited paper at Fourth Conference on Basic Problems of Numerical Analysis (LIBLICE IV), Pilsen, Czechoslovakia, Sept 1978, pp 63-78.

[4] Boley D L, Golub G H: A modifled method for reconstructing periodic Jacobi matrices; Math Comp 42 #165, pp 143-150, 1984.

[5] Conte S D, deBoor C: Elementary Numerical Analysis: An Algorithmic Approach: 3rd ed., McGraw Hill, New York, 1980.

[6] de Boor C: personal communication.

[7] de Boor C, Golub G H: The numerically stable reconstruction of a Jacobi matrix from; spectral data, Lin Alg & Appl 21, pp 245-260, 1978.

[8] Smith B T, Boyle J M, Ikebe Y, Klema V C, Moler C B: Matrix Eigenvalue Routines, EISPACK Guide, 2nd ed.,; Springer Verlag, New York, 1970.

[9] Ferguson W Jr: The construction of Jacobi and periodic Jacobi matrices with prescribed spectra; Math Comp 35, pp 79-84, 1980.

[10] Friedland S, Nocedal J, Overton M L: The Formulation and Analysis of Numerical Methods for Inverse Eigenvalue Problems; Comp. Sci. Dept. Report 179, Courant Institute of Math. Sci., New York Univ., Sept. 1985.

[11] Gladwell G M L: "The inverse problem for the vibrating beam", Proceedings of the Royal Society of London Series A - Mathematical and Physical Sciences, v393, n1805, p277-295, 1984

[12] Gladwell G M L, Gbadeyan J A: "On the inverse problem of the vibrating string or rod", Quarterly Journal of Mechanics and Applied Mathematics , v38, Feb, p169-174, 1985

[13] Gladwell G M L: "The inverse mode problem for lumped-mass systems", Quarterly Journal of Mechanics and Applied Mathematics , v39, May, p297-307, 1986

[14] Golub G H, Welsch J H: Calculation of Gauss quadrature rules; Math Comp 23, pp 221-230, 1969.

[15] Golub G H: Some Modifled Eigenvalue Problems; SIAM Review 15 #2, pp 318-334, 1973.

[16] Golub G H, Van Loan C: Matrix Computations; Johns Hopkins Univ. Press, 1983.

[17] Gragg W B, Harrod W J: The numerically stable reconstruction of Jacobi matrices from spectral data; Numer Math 44, pp 317-335, 1984.

[18] Grunbaum F A: An Inverse Spectral Problem for Band Matrices; Report PAM-10, Ctr for Pure & Appl Math, Univ of Calif - Berkeley, Dec 1980.

[19] Grunbaum F A: Role of Strict Interlacing in Inverse Eigenvalue Problems; Report PAM-8, Ctr for Pure & Appl Math, Univ of Calif - Berkeley, 1980.

[20] Hald 0: Inverse eigenvalue problems for Jacobi matrices; Lin Alg & Appl 14, pp 63-85, 1976.

[21] Hochstadt H: On construction of a Jacobi matrix from spectral data; Lin Alg & Appl 8, pp 435-446, 1974.

[22] Dongarra J, Bunch J R, Moler C B, Stewart G W: LINPACK Users' Guide; SLAM Publications, Philadelphia, 1978.

[23] Paine J: A numerical method for the inverse Sturm--Liouville problem; SLAM J. Sci. Stat. Comput. 5, pp149-156, 1984.

[24] Parker R L, Whaler K A: "Numerical-methods for establishing solutions to the inverse problem of electromagnetic induction", Journal of Geophysical Research , v86, nb10, p9574-9584, 1981

[25] Parlett B N: The Symmetric Eigenvalue Problem; Prentice Hall, 1980.

[26] Rutishauser H: On Jacobi rotation patterns; in Experimental Arithmetic, High Speed Computing and Mathematics, Proc. Symp. Appi. Math 15, Amer. Math. Soc. pp 219-239, 1963.

[27] Horowitz E, Sahni S: Fundamentals of Computer Algorithms; Computer Science Press, Rockville, MD, 1978 and 1984.

[28] Stewart G W: Introduction to Matrix Computations: Academic Press, 1973.

[29] Sussman-Fort S E: "A numerical design procedure for general canonic LC one-ports", IEEE Transactions on Circuits and Systems , v29, n9, p633-638, 1982

[30] Thompson John: personal communication.