A Modified Method for Reconstructing Periodic Jacobi Matrices

By Daniel Boley and Gene H. Golub*

Abstract. In this note, we discuss the reconstruction of periodic Jacobi matrices from spectral data. The method combines ideas and techniques from the algorithms given by Boley and Golub [1], [2], and Ferguson [3], resulting in a numerically stable algorithm applicable to a larger class of problems. The number of initial data items needed for this method equals the number of items in the resulting matrix, namely $2n$.

1. Introduction. In this note, we discuss the reconstruction of a periodic Jacobi matrix of the form

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

from the eigenvalue data using methods that are numerically stable. Here the $a_i$ and $b_i$ are real, and $b_i \neq 0$ for all $i$. Such problems arise, for example, in the study of Toda Lattices [9], [10], and might also arise in the study of Sturm-Liouville and Hill’s equations [3]. In particular, we are given the following initial information: the eigenvalues $\lambda_i$, $i = 1, \ldots, n$ of $J$, the eigenvalues $\mu_i$, $i = 1, \ldots, n - 1$ of the submatrix $\tilde{J}$ obtained from $J$ by deleting the first row and column. We also need the product $\beta = b_1 b_2 \cdots b_n$, or equivalently the eigenvalues $\lambda_i$, $i = 1, \ldots, n$ of the matrix

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

Received October 5, 1982.

1980 Mathematics Subject Classification. Primary 15A18, 65F15.

*We gratefully acknowledge the support of the National Science Foundation under grant ECS-8204468 for the first author and the Department of Energy under grant DE-AT03-76ER 71030 for the second author.

©1984 American Mathematical Society

License or copyright restrictions may apply to redistribution; see http://www.ams.org/journal-terms-of-use
The techniques used are based on the methods described in [1] and [2]. In [3], it was pointed out that the method in [2] would fail on a certain class of matrices, and a method was given extending the collection of solvable problems to cover this class. The class in question can be described in terms of the interlacing property

\[ \lambda_i \leq \mu_i \leq \lambda_{i+1}, \quad i = 1, \ldots, n - 1. \]

If the inequalities are strict, then we say \( J \) satisfies the strict interlacing property. This last property is necessary for the algorithm in [2] but not for the one in [3]. In addition, unlike [2], the method in [3] does not require that the order of the system, \( n \), be even. However the method in [3] starts with substantially different initial data and hence does not directly solve the problem posed in [2].

In this method, we need to assume only that the \( \mu_i \) be simple. This is equivalent to assuming that the off-diagonal elements of \( \bar{J} \) be nonzero [8, p. 300]. In order to ensure that a solution exists in the real numbers, we also need to assume that the \( \lambda_i \) satisfy (2), and that \( \beta \) lies in an interval \((0, \beta_{\max})\), which will be defined below.

2. Preliminaries. We need to define the following two bordered diagonal matrices \( A, A^- \) [4] which appear as intermediate results in the course of the computation, and which are similar to \( J \) and \( J^- \), respectively, by the same transformation:

\[
A = \begin{bmatrix} 1 & 0 \\ P^T \end{bmatrix} J \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix}, \quad A^- = \begin{bmatrix} 1 & 0 \\ 0 & P^T \end{bmatrix} J^- \begin{bmatrix} 1 & 0 \\ 0 & P \end{bmatrix},
\]

where \( P \) is the orthogonal matrix of eigenvalues of the submatrix \( \bar{J} \). Hence \( A \) and \( J \) have the same characteristic polynomial, and so do \( A^- \) and \( J^- \).

The matrices \( A, A^- \), can be written in the form:

\[
A = \begin{bmatrix} a_{11} & c^T \\ c & M \end{bmatrix}, \quad A^- = \begin{bmatrix} a_{11} & (c^-)^T \\ c^- & M \end{bmatrix},
\]

where \( M = \text{diag}(\mu_1, \ldots, \mu_{n-1}) = P^T\bar{J}P \), and the eigenvalues of \( A \) and \( A^- \) are \( \lambda_i \) and \( \lambda_i^- \), respectively. We shall later see how to define \( A^- \) using only \( \beta \) instead of the \( \lambda_i^- \).

In the following, we will denote the \( i \)th row of \( P \) as \( p_i \), that is \( P^T = [p_1, \ldots, p_{n-1}] \).

It is obvious that

\[
a_{11} = \text{trace}(A) - \text{trace}(M) = \sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n-1} \mu_i.
\]

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

\[
\det(\lambda I - A) = (\lambda - a_{11}) \prod_{j=1}^{n-1} (\lambda - \mu_j) - \sum_{k=1}^{n-1} c_k^2 \left( \prod_{j=1}^{n-1} (\lambda - \mu_j) \right).
\]

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

\[
c_i^2 = \frac{\prod_{j=1}^{n-1} (\mu_i - \lambda_j)}{\prod_{j=1, j \neq i}^{n-1} (\mu_i - \mu_j)} \geq 0,
\]

thus completely defining \( A \) up to the choice of signs of the \( c_i \).
If we let
\[
\begin{bmatrix}
a_1 & b_1 & & 0 \\
b_1 & a_2 & & \\
& & \ddots & 0 \\
0 & & & a_{n-1} & b_{n-1} \\
& & & b_{n-1} & a_n
\end{bmatrix}
\]
and define \( p(\lambda) \), \( r(\lambda) \) as the characteristic polynomials of the tridiagonal matrices \( \overline{J} \) and \( \tilde{J} \), respectively, then we may write
\[
\det(J - \lambda I) = p(\lambda) - b_1^2r(\lambda) - 2(-1)^\beta,
\]
\[
\det(J - \lambda I) = p(\lambda) - b_1^2r(\lambda) + 2(-1)^\beta.
\]
Subtracting yields
\[
\det(J - \lambda I) = \det(A - \lambda I) = \det(A - \lambda I) + 4(-1)^\beta.
\]
So we can solve for \( c_i^- \) as in (4c) to obtain
\[
(4d) \quad (c_i^-)^2 = -\frac{\prod_{j=1}^{n} (\mu_i - \lambda_j) + 4(-1)^\beta}{\prod_{j=1}^{n-1} (\mu_i - \mu_j)}.
\]
Hence we can define \( A^- \), either with \( \beta \) using (4d), or else by using (4c) with \( \lambda_i^- \) substituted for \( \lambda_i \). Note that (4d) defines the interval \((0, \beta_{\text{max}})\) in which \( \beta \) must lie by the requirement that \((c_i^-)^2 \geq 0\).

If we equate the first columns from Eq. (3a), we get
\[
(5a) \quad [b_1, 0, \ldots, 0, b_n]^T = Pc, \quad [b_1, 0, \ldots, 0, -b_n]^T = Pc^-,
\]
in which the left-hand sides are \((n - 1)\)-vectors. Subtracting yields
\[
[0, \ldots, 0, 2b_n]^T = 2b_n\mathbf{e}_{n-1} = P(c - c^-),
\]
which, applying \( P^T \) to both sides, yields
\[
(5b) \quad 2b_n\mathbf{p}_{n-1} = c - c^-.
\]
Similarly, if we add the two parts of (5a) together, we get
\[
(5c) \quad 2b_1\mathbf{p}_{1} = c + c^-.
\]
In (5b), (5c) the left-hand sides are determined up to sign by the requirement that \( \|\mathbf{p}_1\| = \|\mathbf{p}_{n-1}\| = 1 \). Note that we have a choice of signs in taking the square roots of the quantities defined in (4c), (4d), to obtain the \( c_i, c_i^- \). A different choice of signs will give rise to different vectors \( \mathbf{p}_1, \mathbf{p}_{n-1} \) and hence a different \( \tilde{J} \). Thus, except in certain degenerate cases, there will be more than one matrix satisfying the initial eigenvalue conditions.

3. The Method. What have we now? We have \( \mathbf{p}_1 \), that is, the first row of the eigenvector matrix \( P \) for a tridiagonal matrix \( \tilde{J} \). We also have the eigenvalues \( \mu_i \) of \( \tilde{J} \), so \( \tilde{J} \) is completely determined (elementary consequence of Theorem 4.2 of Chapter 7
of [6]). We can compute $\tilde{J}$ an element at a time, and $P$ a row at a time, by alternating between the formulas

\begin{align}
(6a) & \quad \tilde{J} = PMPT, \\
(6b) & \quad P^T\tilde{J} = MP^T. 
\end{align}

The method that results from this is the Lanczos Algorithm [7], [8], and proceeds as follows:

**Lanczos Algorithm**

Start with $p$, an $(n - 1)$-vector with 2-norm 1, and $M = \text{diag}(\mu_1, \ldots, \mu_{n-1})$, a diagonal matrix.

Object: compute orthonormal vectors $p_2, \ldots, p_{n-1}$ and an $n - 1 \times n - 1$ tridiagonal matrix $J$ satisfying (6).

1. begin
2. $t_1 \leftarrow p_1^TMp_1$;
3. for $i \leftarrow 2, \ldots, n - 1$ do
4. \hspace{1em} begin
5. \hspace{2em} $z_i \leftarrow (\text{if } i = 2 \text{ then } Mp_1 - p_1t_1$
6. \hspace{3em} $\text{else } Mp_{i+1} - p_{i+1}t_{i+1} - p_{i-1}s_{i-1};$
7. \hspace{2em} $s_{i-1} \leftarrow \|z_i\|_2^2;$
8. \hspace{2em} $p_i \leftarrow z_i/s_{i-1};$
9. \hspace{2em} $t_i \leftarrow p_i^TMp_i;$
10. \hspace{1em} end
11. end.

Step 5 is mathematically equivalent to saying

$z_i \leftarrow Mp_{i-1}$

Orthogonalize $z_i$ against all $p_{i-1}, \ldots, p_1$ by Gram-Schmidt.

What was actually implemented was the original step 5, followed by a complete Gram-Schmidt reorthogonalization. The Lanczos Algorithm is stable only if some reorthogonalization is used [11, Theorems 2.3, 2.4, 2.5], and for simplicity we chose to use complete reorthogonalization.

In step 7, $s_{i-1} = 0$ is guaranteed from the assumption that the eigenvalues $\mu_i$ are distinct [8, p. 300].

It must be noted that the tridiagonal matrix $J$ may be constructed by using Householder Transformations instead of using the Lanczos Algorithm with complete reorthogonalization. The details are given in [4], and this method can be substituted
for the Lanczos algorithm giving theoretically equivalent results. However, it was noted in [4] that the method using Householder Transformations is more expensive, but on badly conditioned problems it may be more stable, depending on the kind of reorthogonalization used [11]. On the other hand, the Lanczos Algorithm is much more suited for large sparse problems. Further comparison between the methods is necessary.

The algorithm to reconstruct $J$ is then as follows:

**Reconstruction Algorithm**

Compute the $n - 1 \times n - 1$ submatrix $\bar{J}$ (defined just below (1a)):
1. Compute $c, c'$ using (4c), (4d).
2. Compute $b_1$ and $p_1$ by (5c) subject to $\|p_1\|_2 = 1$.
3. Apply the Lanczos Algorithm starting with vector $p_1$ and matrix $M = \text{diag}(\mu_1, \ldots, \mu_{n-1})$, obtaining vectors $P$ and tridiagonal matrix $\bar{J}$.

Fill in the first row and column of $J$ defined in (1a):
4. Compute $a_i = \Sigma_1^i \lambda_i - \Sigma_1^{i-1} \mu_i$, (Eq. (4a)).
5. Compute $b_n$ by either (5b) or from the definition of $\beta$:

$$b_n = \frac{\beta}{(b_1 \cdots b_{n-1})}.$$

4. **Numerical Results.** A RATFOR program has been written to test this method. It successfully reconstructed the example suggested in [3], with $n = 4, 5, 6$, namely

$$J = \begin{bmatrix}
2 & 1 & & 1 \\
1 & 2 & & \\
& \ddots & \ddots & 0 \\
& & 0 & \ddots \\
& & \ddots & 2 & 1 \\
1 & & & 1 & 2
\end{bmatrix},$$

for which the $\mu_i$ are simple, but the $\lambda_i$ are not. The initial data used in these test cases, as well as the vectors $c, c'$, are listed below:

$n = 4, \beta = 1$

<table>
<thead>
<tr>
<th>$\lambda_i$</th>
<th>$\mu_i$</th>
<th>$c_i$</th>
<th>$c'_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.58578644</td>
<td>1.00000000</td>
<td>0</td>
</tr>
<tr>
<td>2.00000000</td>
<td>2.00000000</td>
<td>0</td>
<td>1.41421356</td>
</tr>
<tr>
<td>2.00000000</td>
<td>3.41421356</td>
<td>1.00000000</td>
<td>0</td>
</tr>
<tr>
<td>4.00000000</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

$n = 5, \beta = 1$

<table>
<thead>
<tr>
<th>$\lambda_i$</th>
<th>$\mu_i$</th>
<th>$c_i$</th>
<th>$c'_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.38196601</td>
<td>0.38196601</td>
<td>0</td>
<td>0.74349607</td>
</tr>
<tr>
<td>0.38196601</td>
<td>1.38196601</td>
<td>1.20300191</td>
<td>0</td>
</tr>
<tr>
<td>2.61803399</td>
<td>2.61803399</td>
<td>0</td>
<td>1.20300191</td>
</tr>
<tr>
<td>2.61803399</td>
<td>3.61803399</td>
<td>0.74349607</td>
<td>0</td>
</tr>
<tr>
<td>4.00000000</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
$n = 6, \beta = 1$

<table>
<thead>
<tr>
<th>$\lambda_i$</th>
<th>$\mu_i$</th>
<th>$c_i$</th>
<th>$c_i^-$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.26794919</td>
<td>0.57735027</td>
<td>0</td>
</tr>
<tr>
<td>1.00000000</td>
<td>1.00000000</td>
<td>0</td>
<td>1.00000000</td>
</tr>
<tr>
<td>1.00000000</td>
<td>2.00000000</td>
<td>1.15470054</td>
<td>0</td>
</tr>
<tr>
<td>3.00000000</td>
<td>3.00000000</td>
<td>0</td>
<td>1.00000000</td>
</tr>
<tr>
<td>3.00000000</td>
<td>3.73205081</td>
<td>0.57735027</td>
<td>0</td>
</tr>
<tr>
<td>4.00000000</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The following example illustrates the fact that the $b_i$ need not be positive. This example is simply the $J^-$, defined by (1b), corresponding to the $J$ used in the above case $n = 4$.

$n = 4, \beta = -1$

<table>
<thead>
<tr>
<th>$\lambda_i$</th>
<th>$\mu_i$</th>
<th>$c_i$</th>
<th>$c_i^-$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.58578644</td>
<td>0.58578644</td>
<td>0</td>
<td>1.00000000</td>
</tr>
<tr>
<td>0.58578644</td>
<td>2.00000000</td>
<td>1.41421356</td>
<td>0</td>
</tr>
<tr>
<td>3.41421356</td>
<td>3.41421356</td>
<td>0</td>
<td>1.00000000</td>
</tr>
<tr>
<td>3.41421356</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The above cases all have remarkable property that for all $i$, $i = 1, \ldots, n - 1$, either $c_i = 0$ or $c_i^- = 0$, so that the reconstructed matrix $J$ is essentially unique (up to signs of the off-diagonal elements). In the following example, we show a more general case, in which the answer is not unique. The results were obtained by starting with one set of eigenvalue data and cycling through all possible sign combinations for the $c_i$, $c_i^-$. The initial data and the $c$ vectors were:

$n = 4, \beta = .25$

<table>
<thead>
<tr>
<th>$\lambda_i$</th>
<th>$\mu_i$</th>
<th>$c_i$</th>
<th>$c_i^-$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.58578644</td>
<td>1.00000000</td>
<td>0.86602540</td>
</tr>
<tr>
<td>2.00000000</td>
<td>2.00000000</td>
<td>0</td>
<td>0.70710678</td>
</tr>
<tr>
<td>2.00000000</td>
<td>3.41421356</td>
<td>1.00000000</td>
<td>0.86602540</td>
</tr>
<tr>
<td>4.00000000</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The 4 distinct matrices that were computed are given below. For brevity, we show each distinct matrix only once, since each one actually appeared 8 times. Since there were no degeneracies encountered in this example, and by observing that any matrix satisfying the initial conditions (given by $\lambda_i$, $\mu_i$, $\beta$) must satisfy (4a)-(4d) and (5c), it follows that the following are all the possible distinct answers for the given initial data above.

$$
a_i =
\begin{align*}
2.00000000 & \quad 1.36602540 \\
2.00000000 & \quad 1.36602540 \\
2.00000000 & \quad 0.36602540 \\
2.00000000 & \quad 0.36602540 \\
\end{align*}
$$
As a final example, we tried the example 3 of Table 2 on page 1218 of [3], in which the matrices in question were reconstructed only with large errors. The matrices are defined as follows, for $n = 5, 10, 15, 20, 25, 30$:

\[
\begin{align*}
    a_i &= \frac{i}{n} - 2, \quad i = 1, \ldots, n - 1, \\
    b_i &= 1 - \frac{i}{n}, \quad i = 1, \ldots, n - 2, \\
    b_n &= b_{n-1} = 1, \\
    a_n &= 0.
\end{align*}
\]

The algorithm was suitably modified to account for the fact that the submatrix $\tilde{J}$ in this example was obtained by deleting the last row and column rather than the first. Unfortunately, there are too many combinations of signs for the $c_i$, $c_i^*$ to compute all the possible matrices. For $n = 10$ alone there are $262144$ different choices of signs. However, in each case we are able to compare the eigenvalues of the reconstructed matrix with the initial eigenvalues. Since the eigenvalues for symmetric matrices are always well-conditioned, the discrepancy between the two sets of eigenvalues is a good indicator of the error in the reconstructed matrices.

<table>
<thead>
<tr>
<th>$n$</th>
<th>Norm of discrepancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>.364539663e $- 15$</td>
</tr>
<tr>
<td>10</td>
<td>.558570184e $- 15$</td>
</tr>
<tr>
<td>15</td>
<td>.130290552e $- 14$</td>
</tr>
<tr>
<td>20</td>
<td>.191718261e $- 14$</td>
</tr>
<tr>
<td>25</td>
<td>.304003744e $- 14$</td>
</tr>
<tr>
<td>30</td>
<td>.340721065e $- 14$</td>
</tr>
</tbody>
</table>
Here, the norm of discrepancy is the square root of the sum of the squares of the differences between the two sets of eigenvalues.

Computer Science Department
Institute of Technology
University of Minnesota
Minneapolis, Minnesota 55455

Computer Science Department
Stanford University
Stanford, California 94305

5. John Thompson, personal communication.