AUG 5 1976

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the Latest Date stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN
CAC Document No. 21

A JACOBI ALGORITHM FOR ILLIAC IV

by

Lawrence M. McDaniel

November 8, 1971
(revised June 7, 1972)
A JACOBI ALGORITHM FOR ILLIAC IV

by

Lawrence M. McDaniel

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

November 8, 1971
(revised June 7, 1972)

This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the U. S. Army Research Office-Durham under Contract No. DAHC04-72-C-0001.
ABSTRACT

This revised version of CAC Document #21 supercedes the document dated November 8, 1971.

Several methods have been proposed to enable the computation of eigenvalues and eigenvectors of large, real symmetric or complex Hermitian matrices on ILLIAC IV.

One of the most effective methods in the utilization of parallel computations has proven to be a modified Jacobi algorithm. This document presents yet another modification which exploits the parallelism of ILLIAC IV to a greater extent than has been previously done.

Flow charts and the assembly language (ASK) routine JACOBI are included in the report.
ACKNOWLEDGEMENT

The author would like to thank Dr. Ahmed H. Sameh, who developed the modified Jacobi algorithm and provided guidance in its implementation.
# Table of Contents

1.0 Introduction ............................................. 2  
2.0 Jacobi's Method ........................................... 3  
3.0 Modifications to the Classical Jacobi Method ............... 3  
3.1 Decomposition Into Block-Diagonal Submatrices ............. 3  
3.2 Optimal Construction of Orthogonal Transformations .......... 3  
3.2.1 Elimination Scheme .................................. 4  
4.0 JACOBI - An ILLIAC IV Routine ......................... 6  
4.1 Introduction ............................................. 6  
4.2 Procedure .............................................. 6  
4.3 Main Flowchart ......................................... 7  
4.4 Description of Main and Auxiliary Routines .............. 10  
4.5 Program Utilization ..................................... 14  
5.0 Test Results ............................................ 16  
5.1 System of Even Order .................................. 16  
5.2 System of Odd Order ................................... 20  
5.3 Comparison with Existing Jacobi Algorithm .......... 24  

References ..................................................... 28  

APPENDIX A - Determination of the Rotation Angle .......... 29  
APPENDIX B - Examples of Elimination Scheme .............. 30  
APPENDIX C - A Typical Calling Program .................. 32  
APPENDIX D - JACOBI Listing (ASK) ........................ 35
1.0 Introduction

The computation of eigenvalues and eigenvectors of large, real symmetric matrices is of great practical value in many fields. One of the most effective methods that has been examined to solve this problem on the ILLIAC IV is a parallel version of Jacobi's algorithm.

The main difference between this code and the one previously written [1] is that within each sweep, defined by \((2m-1)\) transformations (where \(m = \lfloor (n+1)/2 \rfloor\), i.e. the greatest integer less than or equal to \((n+1)/2\), in which \(n\) is the order of the matrix), each orthogonal transformation annihilates different \(\lfloor n/2 \rfloor\) off-diagonal elements. This proved to be a substantial improvement that led to a greater speed of convergence.
2.0 Jacobi's Method

In the classical method of Jacobi, a real symmetric matrix is reduced to the diagonal form by a sequence of plane rotations

\[ A_k = R_k A_{k-1} R_k^T \quad (k = 1, 2, \ldots) \]

where \( A_0 = A \) is the original matrix, and the rotation matrix \( R_k \) eliminates the off-diagonal element \( a_{pq}^{(k)} \) (hence \( a_{pq} \)) through an angle \( \alpha_{pq}^{(k)} \)[5]. See Appendix A for the appropriate value of \( \alpha_{pq}^{(k)} \) to annihilate the element \( a_{pq} \).

3.0 Modifications to the Classical Jacobi Method

3.1 Decomposition Into Block Diagonal Submatrices

Rather than searching for the largest off-diagonal element of \( A_{k-1} \) in the \((p,q)\) position and eliminating \( a_{pq} \) and \( a_{qp} \), A. Sameh and L. Han.[4] proposed a modified Jacobi algorithm where all off-diagonal elements of each \(2 \times 2\) submatrix along the diagonal are eliminated through an orthogonal transformation.

In order to bring to the diagonal new submatrices with non-zero off-diagonal elements, the method necessitates a large degree of row and column shuffling which tends to be expensive on a parallel computer such as ILLIAC IV.

3.2 Optimal Construction of Orthogonal Transformations

A. Sameh[3] showed that a judicious choice of the pairs \((p,q)\) can produce a modified Jacobi algorithm that attains maximum efficiency of parallel computation.

For example, for a matrix \( A \) of order 4, if the orthogonal transformation \( R \) is chosen as,

\[
R = \begin{bmatrix}
c_1 & 0 & s_1 & 0 \\
0 & c_2 & 0 & s_2 \\
-s_1 & 0 & c_1 & 0 \\
0 & -s_2 & 0 & c_2
\end{bmatrix}
\]

where \( c_i = \cos \alpha_i (i = 1, 2) \), then \( RAR^T \) would have zero elements in positions (1,3) and (2,4) provided that the angles \( \alpha_1 \) and \( \alpha_2 \) are properly chosen.
Define \( m = \lfloor (n+1)/2 \rfloor \) where \( n \) is the order of the matrix and \( \lfloor \rfloor \) is the greatest integer function. Let each of the \((2m-1)\) orthogonal transformations be denoted by a sweep. Noting that the maximum number of the \((n^2-n)/2\) off-diagonal elements which can be eliminated by an orthogonal transformation of the type (3.1) is \([n/2]\), an optimal algorithm requires that, [3]:

1) Each orthogonal transformation \( R_k \) should eliminate \([n/2]\) off-diagonal elements.

2) Each sweep should eliminate each off-diagonal element once, i.e. each of the \(2m-1\) orthogonal transformations in a sweep should annihilate different \([n/2]\) off-diagonal elements.

3.2.1 Elimination Scheme

In [3] several schemes were proposed to satisfy the requirements discussed in the previous section. The scheme implemented in the JACOBI algorithm is described below:

For a given sweep, each of the \((2m-1)\) orthogonal matrices \( R_k \) consist of the elements,

\[
(3.2) \quad R_{pp}^{(k)} = R_{qq}^{(k)} = -\cos \alpha_{pq}^{(k)}, \quad R_{pq}^{(k)} = -R_{qp}^{(k)} = \sin \alpha_{pq}^{(k)}, \quad p < q,
\]

\[
= -\sin \alpha_{pq}^{(k)} \quad p > q,
\]

where \( p \) and \( q \) are sequences defined by

(a) for \( k = 1,2,\ldots,m-1, \)
\[
q = m - k + 1, m - k + 2, \ldots, n - k.
\]
\[
p = (2m - 2k + 1) - q,
\]

\[
m - k + 1 \leq q \leq 2m - 2k,
\]

\[
= (4m - 2k) - q, \quad 2m - 2k < q \leq 2m - k - 1,
\]

\[
= n, \quad 2m - k - 1 < q,
\]

(b) for \( k = m, m + 1,\ldots, 2m - 1, \)
\[
q = 4m - n - k, 4m - n - k + 1, \ldots, 3m - k - 1
\]
\[
p = n, \quad q < 2m - k + 1,
\]

\[
= (4m - 2k) - q, \quad 2m - k + 1 \leq q \leq 4m - 2k - 1,
\]

\[
= (6m - 2k - 1) - q, \quad 4m - 2k - 1 < q.
\]
The remaining elements of \( R_k \) are zero except for \( n \) odd, then \( R_{2m-k,2m-k}^{(k)} = 1 \).

For a given \( k \), the angles \( \alpha_{pq}^{(k)} \) are determined for all \((p,q)\) such that \( \alpha_{pq}^{(k)} \) eliminates the element \( a_{pq}^{(k)} \); see Appendix A.

For example, in a given sweep, denoting each element in the transformation by the integer \( k \), the patterns of the eliminated elements for matrices of orders 8 and 7 are shown below.

\[
\begin{array}{cccccccc}
\ast & 3 & 6 & 2 & 5 & 1 & 4 & 7 \\
\ast & 2 & 5 & 1 & 4 & 7 & 6 \\
\ast & 1 & 4 & 7 & 3 & 5 \\
\ast & 7 & 3 & 6 & 4 \\
\ast & 6 & 2 & 3 \\
\ast & 5 & 2 \\
\ast & 1 \\
\ast \\
7 \times 7 \\
8 \times 8 \\
\end{array}
\]

Note that since \( a_{qp}^{(k)} \) as well as \( a_{pq}^{(k)} \) is eliminated, if one completes the lower diagonal portion of the matrix above, it is evident that any given \( k \) appears in each row and column once and only once.

For further examples of particular orthogonal transformations constructed by this elimination scheme see Appendix B.
4.0 **JACOBI - An ILLIAC IV Routine**

4.1 **Introduction**

JACOBI is an ILLIAC IV routine written in the assembly language ASK, which implements the modified Jacobi algorithm discussed in Section 3.2.

The program accepts as input a matrix $A$, and $n$, the order of the matrix, and returns as output the matrix $A$ reduced to diagonal form, the matrix of eigenvectors corresponding to the computed eigenvalues and the number of sweeps required to achieve convergence.

A flow chart, a description of JACOBI and auxillary routines, and a short discussion of JACOBI for the potential user are now presented.

4.2 **Procedure**

Each sweep requires $2m-1$ orthogonal transformations as described in Section 3.2. For each transformation the following sequences of events occur:

(i) The pairs $(p,q)$ corresponding to the element $a_{pq}^{(k-1)}$ to be eliminated are determined.

(ii) The orthogonal transformation matrix $R_k$ is constructed in order to eliminate the proper elements of $A$.

(iii) $A_{k-1}$ is pre- and post-multiplied by the transformation matrix $R_k$ to yield $A_k = R_k^t A_{k-1} R_k$ with elements $a_{pq}^{(k)} = 0$.

(iv) $E_{k-1}$, the eigenvector matrix, is pre-multiplied by $R_k^t$ to yield $E_k$, where $E_0$ is the identity matrix. (Note that the rows and columns of $E$ correspond to the left and right eigenvectors respectively).

After $2m-1$ such transformations have been applied, the following convergence criterion are subjected to $A_k$:

(a) if the sum of the squares of the off-diagonal elements is zero, convergence is attained.

(b) if the ratio of the sum of the squares of the off-diagonal elements to the sum of the squares of the diagonal elements at step $k$ is sufficiently small ($10^{-16}$) in comparison to an equivalent ratio at step 1, the method has converged. (to insure numerical stability this criteria is not applied for the first three sweeps).
If the convergence tests fail, a new sweep is initiated. When the method does converge, the bounds on the eigenvalues are computed using Gerschgorin discs, and this information is output for the user.

4.3 Main Flowchart

Notation: $A_k$ - Matrix to be diagonalized at step $k$
$E_k$ - Corresponding eigenvector matrix
$R_k$ - Orthogonal transformation matrix
$B$ - Temporary matrix

\[
\text{Ratio: } \frac{\sum_{i,j} a_{ij}(k)^2}{\sum_{i=1}^{n} a_{ii}(k)^2} \text{ where } a_{ij} \in A_k
\]

\[
\text{Kconv: } \frac{\sum_{i,j} a_{ij}(1)^2}{\sum_{i=1}^{n} a_{ii}(1)^2}
\]

$L$ - Transformation count
$swp$ - Sweep count
$k = swp \times (2m-1) + \ell$ where $m = \left\lceil \frac{n+1}{2} \right\rceil$

$n = \text{order of matrix}$

A detailed discussion of the components of JACOBI, depicted in the flowchart presented in this section may be found in Section 4.4.
Ao input matrix
n - order of matrix

JACOBI

Initialize
Set \( E_0 = 1 \)
\( m = \left[ \frac{n + 1}{2} \right] \)
swp = 0
kconv = 0.0

Set up matrix defining pattern of pairs \((p,q)\) to be eliminated for each sweep.

SWEEP

swp = swp + 1

LOOP

\( \ell = 1, 2, \ldots, 2m - 1 \)

Note \( k = \text{swp} \times (2m - 1) + \ell \)

Retrieve \((p,q)\) pattern in \(q\)th row of matrix computed above.

ANGLE

Compute sines & cosines of \( R_k \)

B is skewed in this routine to prepare for transpose below

MULTPL

Compute
\[ B^t = R_k^t A_{k-1} \]

Align rows of \( B^t \) to produce
\[ B = A_{k-1}^t R_k \]

Note \( A_{k-1} = A_k \)

see (1) p. 10

see (2) p. 10

see (3) p. 11

see (3) p. 12
Determine new $A_k$

Update Eigenvector matrix.

End of sweep test

Ensure numerical stability

Convergence Test

Compute

\[ R_k^t B = A_k \]

MULTPL

Compute

\[ E_k = R_k^t E_{k-1} \]

MULTPL

\[ \lambda = 2^m - 1 \] ?

\[ \text{ADDIT} \]

Find Convergence Ratio

Sum Of Off-Diagonal Elements of \[ A_k = 0.0 \] ?

\[ KCONV = 0.0 \] ?

\[ \text{SWP < 4} \] ?

\[ \text{RATIO < KCONV} \] ?

Computes Bounds on Eigenvalues

\[ \text{RETURN} \]

see (4) p. 12

see (5) p. 13

see (6) p. 13

see (7) p. 13
4.4 Description of Main and Auxiliary Routines

After saving the necessary registers, return addresses, and addresses of calling parameters, setting up constants and counters, JACOBI constructs $E_0 = I$, the identity matrix.

\(1\) Determination of pairs \((p,q)\)

In order to implement this phase of the algorithm on ILLIAC IV, it is desirable to maintain compatibility with PE numbering (i.e. $0 \leq p, q \leq n-1$) and also alter the definition for $p,q$. The following definition will produce identical pairs except that now $a_{ij}$ in Section 3.2.1 corresponds to $a_{i-1,j-1}$ as defined in this section.

Let $q$ = PE number $q = 0,1,...,n-1$

\[ m = \left[ \frac{n+1}{2} \right] \]

\[ n = \text{dimension of matrix} \]

For $k = 1,2,...,2m-1$ let $k_0 = 4m-2$ \quad $k = 1,2,...,m-1$

\[ k_0 = 6m-3 \quad k = m,m+1,...,2m-1 \]

Then $p = (k_0 - 2k - q) \mod (2m-1)$

Thus, \((p,q)\) are defined above except for the following cases:

(a) if $n$ even, set $p = n-1$ in PE($n-1-k$)

set $p = n-1-k$ in PE($n-1$)

(b) if $n$ odd, note that $p = q$ in one PE. This fact will be taken into consideration later on in the program.

As the pattern of pairs is constant for each sweep this calculation is only done once in the program and saved for subsequent usage.

At label SWEEP, all necessary preparations are made for another $2m-1$ transformations.

\(2\) ANGLE - Compute sines and cosines of the transformation matrix.

Input to this routine is the matrix $A_k$ and the pairs $(p,q)$ determined in $(1)$ above.

Element $a_{pq}$ is brought to PE $q$. This is accomplished by a right route of $(q-p)$ for $p < q$ or by a left route of $(p-q)$ for $p > q$. 

-10-
The sines and cosines are computed using the formulas in Appendix A and stored in two rows of PE memory (SIN, COS). If \( n \) is odd, the \( q \)th PE has \( p = q \) and in this PE, cosine and sine are set to 1.0 and 0.0 respectively.

(3) \textbf{MULTPL} – Compute \( B = A_{k-1} R_k \). Let \( n = 4 \), for \( k = 1 \) the ordered pairs are \((0,1), (2,3)\). Let \( R_{ij}^{(k)} \) be as defined in Section 3.2.1 with modifications to that definition as noted in (1) of this section. We wish to compute:

\[
B = A_{k-1} R_k = \begin{bmatrix}
  a_{00}^{(k-1)} & a_{01}^{(k-1)} & a_{02}^{(k-1)} & a_{03}^{(k-1)} \\
  a_{01}^{(k-1)} & a_{11}^{(k-1)} & a_{12}^{(k-1)} & a_{13}^{(k-1)} \\
  a_{02}^{(k-1)} & a_{12}^{(k-1)} & a_{22}^{(k-1)} & a_{23}^{(k-1)} \\
  a_{03}^{(k-1)} & a_{13}^{(k-1)} & a_{23}^{(k-1)} & a_{33}^{(k-1)}
\end{bmatrix}
\begin{bmatrix}
  R_{00} & R_{01} & 0 & 0 \\
  -R_{01} & R_{11} & 0 & 0 \\
  0 & 0 & R_{22} & R_{23} \\
  0 & 0 & -R_{23} & R_{33}
\end{bmatrix}
\]

We note that column 1 of \( B = R_{00} \times \text{col 1 of } A + [-R_{01}] \times \text{col 2 of } A \) column 2 of \( B = R_{01} \times \text{col 1 of } A + [R_{11}] \times \text{col 2 of } A \) etc.

Rather than working with columns it is preferable to work with rows, and without loss of generality the transformation matrix \( R \) above may now be considered as \( R^t \).

Then \( B^t = (AR)^t = R^t A^t = R^t A \) (since \( A \) is symmetric)

\[
B^t = R^t A_{k-1} = \begin{bmatrix}
  R_{00} & R_{01} & 0 & 0 \\
  -R_{01} & R_{11} & 0 & 0 \\
  0 & 0 & R_{22} & R_{23} \\
  0 & 0 & -R_{23} & R_{33}
\end{bmatrix}
\begin{bmatrix}
  a_{00}^{(k-1)} & a_{01}^{(k-1)} & a_{02}^{(k-1)} & a_{03}^{(k-1)} \\
  a_{01}^{(k-1)} & a_{11}^{(k-1)} & a_{12}^{(k-1)} & a_{13}^{(k-1)} \\
  a_{02}^{(k-1)} & a_{12}^{(k-1)} & a_{22}^{(k-1)} & a_{23}^{(k-1)} \\
  a_{03}^{(k-1)} & a_{13}^{(k-1)} & a_{23}^{(k-1)} & a_{33}^{(k-1)}
\end{bmatrix}
\]
In PE memory we have:

<table>
<thead>
<tr>
<th></th>
<th>PE 0</th>
<th>PE 1</th>
<th>PE 2</th>
<th>PE 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>row COS:</td>
<td>R\textsubscript{00}</td>
<td>R\textsubscript{11}</td>
<td>R\textsubscript{22}</td>
<td>R\textsubscript{33}</td>
</tr>
<tr>
<td>row SIN:</td>
<td>R\textsubscript{01}</td>
<td>-R\textsubscript{01}</td>
<td>R\textsubscript{23}</td>
<td>-R\textsubscript{23}</td>
</tr>
<tr>
<td>row PROW(p):</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>row PEN(q):</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Row A (0): \[ A_00 \ A_01 \ A_02 \ A_03 \]
Row A (1): \[ A_01 \ A_{11} \ A_{12} \ A_{13} \]
Row A (2): \[ A_02 \ A_{12} \ A_{22} \ A_{23} \]
Row A (3): \[ A_03 \ A_{13} \ A_{23} \ A_{33} \]

The computation of \( B^T \) is simply done as follows:

Row \( q \) of \( B^T \) = element \( q \) of row "COS" x row \( q \) of A
+ element \( q \) of row "SIN" x row \( p \) of A.

As each row of \( B^T \) is computed it is skewed in preparation for realignment to yield \( B \). The main routine needs only to shift row \( i \) left by \( i \) \((i=0,1,\ldots,n-1)\) to achieve the desired matrix:

\[
\begin{bmatrix}
  b_{00} & b_{01} & b_{02} & b_{03} \\
  b_{13} & b_{10} & b_{11} & b_{12} \\
  b_{22} & b_{23} & b_{20} & b_{21} \\
  b_{31} & b_{32} & b_{33} & b_{30}
\end{bmatrix}
\begin{bmatrix}
  b_{00} & b_{01} & b_{02} & b_{03} \\
  b_{10} & b_{11} & b_{12} & b_{13} \\
  b_{20} & b_{21} & b_{22} & b_{23} \\
  b_{30} & b_{31} & b_{32} & b_{33}
\end{bmatrix}
\]

\( B \) skewed (output from MULTPL)
\( \bar{B} \) re-aligned (in main routine)

(4) Compute \( R_k^T B = A_k \). The routine MULTPL is employed to determine the new \( A_k \). Naturally the skewing logic in MULTPL, described in (3) is bypassed.
(5) Compute $R_k^t E_{k-1} = E_k$. The eigenvector matrix is updated using routine MULTPL once more, and of course bypassing the skewing logic. It is preferable to pre-multiply, as a result the rows and columns of $E_k$ will contain the left and right eigenvectors corresponding to the diagonal elements of $A_k$.

(6) Convergence (see Section 4.2).

(7) GERSH - Finally the bounds of the eigenvalues are determined in routine GERSH by Gerschgorin discs. The eigenvalue is the center of the disc, and the bound on eigenvalue $i$ is the sum of the off-diagonal elements of row $i$ in the diagonal matrix $A_k$. 
4.5 Program Utilization

The usage of JACOBI is enabled by the call statement below:

CALL JACOBI (< the \{A\}_{\text{diagonal}} \text{ matrix }>,
             < temporary matrix 1 >,
             < temporary matrix 2 >,
             < eigenvector matrix >,
             < bounds on eigenvalues >,
             < order of matrix >,
             < sweep count >);

All parameters are passed as addresses whose contents are described as follows:

1. < the \{A\}_{\text{diagonal}} \text{ matrix } > - As input this is the original symmetric matrix to be diagonalized. If the user desires to display the original matrix or retain its contents, he should make necessary preparations before the call to JACOBI. The diagonal matrix replaces the A matrix and is output via this calling parameter.

2. < temporary matrix 1 > - a temporary matrix to enable matrix multiplication for JACOBI which is available to the user after exiting JACOBI.

3. < temporary matrix 2 > - a temporary matrix to contain the pattern describing the elements to be annihilated in a given sweep. This matrix is also available for usage upon exiting the routine.

4. < eigenvector matrix > - the matrix of eigenvectors computed by JACOBI whose rows contain the right eigenvectors and columns the left eigenvectors.

5. < bounds on eigenvalues > - a vector whose value in PE i gives the bound on \( \lambda_i \).

6. < order of matrix > - the order of the matrix, an integer \( \leq 64 \).

7. < sweep count > - the number of sweeps to achieve convergence, an integer value.

In accordance with ILLIAC IV subroutine linkage standards, the contents of ACARS 0 and 1, as well as \$D32-$D63 are saved. The user is advised not to use \$D0-$D31 since they will be overwritten.
<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>STORAGE REQUIRED</th>
<th>STORAGE MODE</th>
<th>RELOCATABLE</th>
<th>CONTENTS DESTROYED</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; the ( A ) diagonal matrix &gt;</td>
<td>( N ) rows</td>
<td>Straight</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt; temporary matrix 1 &gt;</td>
<td>( N ) rows</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt; temporary matrix 2 &gt;</td>
<td>( N ) rows</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt; eigenvector matrix &gt;</td>
<td>( N ) rows</td>
<td>Straight</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt; bounds on eigenvalues &gt;</td>
<td>1 row</td>
<td>Straight</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt; order of matrix &gt;</td>
<td>1 word</td>
<td>-</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>&lt; sweep count &gt;</td>
<td>1 word</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

* where \( N = < \text{order of matrix} > \)
5.0 Test Results

5.1 System of Even Order

See [2] p. 55. The matrix to be diagonalized is

\[
\begin{bmatrix}
5 & 4 & 1 & 1 \\
4 & 5 & 1 & 1 \\
1 & 1 & 4 & 2 \\
1 & 1 & 2 & 4
\end{bmatrix}
\]

\[\lambda_1 = 10.0\]
\[\lambda_2 = 5.0\]
\[\lambda_3 = 2.0\]
\[\lambda_4 = 1.0\]

\[
x_1 = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \end{bmatrix}
x_2 = \begin{bmatrix} -1 \\ -1 \\ 2 \\ 2 \end{bmatrix}
x_3 = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix}
x_4 = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix}
\]

Output from JACOBI (note the eigenvector matrix could be scaled to yield the above result):

* Due to the slow speed of the ILLIAC IV Simulator (about $10^6$ times slower than the ILLIAC IV), we tested only matrices of small order.
THE EIGENVECTOR MATRIX
5.2 System of Odd Order

See [2] pp. 58-59 the matrix to be diagonalized is

\[
\begin{bmatrix}
5 & 4 & 3 & 2 & 1 \\
4 & 6 & 0 & 4 & 3 \\
3 & 0 & 7 & 6 & 5 \\
2 & 4 & 6 & 8 & 7 \\
1 & 3 & 5 & 7 & 9
\end{bmatrix}
\]

\[
\lambda_1 = 22.40687532 \\
\lambda_2 = 7.513724155 \\
\lambda_3 = 4.848950120 \\
\lambda_4 = 1.327045605 \\
\lambda_5 = 1.096595181
\]
THE EIGENVECTOR MATRIX
5.3 Comparison with Existing Jacobi Algorithm

W. Bernhard's ILLIAC IV routine EIGEN [1] is essentially the algorithm discussed briefly in Section 3.1. A comparison run was performed to insure the two algorithms produced compatible results. See [1] pp. 127-129.

Output from JACOBI:
References


The orthogonal matrix $R(p,q,a_{pq}^{(k)})$, differs from the identity matrix by a $2 \times 2$ diagonal submatrix whose elements are

\begin{equation}
R_{pp} = R_{qq} = \cos a_{pq}^{(k)} \quad R_{pq} = -R_{qp} = \sin a_{pq}^{(k)}
\end{equation}

where $p < q$. In order to eliminate the off-diagonal element $a_{pq}^{(k)}$, the angle $a_{pq}^{(k)}$ is chosen such that

\begin{equation}
\tan 2a_{pq}^{(k)} = \frac{2a_{pq}^{(k)}}{a_{pp}^{(k)} - a_{qq}^{(k)}}
\end{equation}

in which $a_{pq}^{(k)}$ is restricted by $|a_{pq}^{(k)}| \leq \pi/4$. Let

\[ t_k = |2a_{pq}^{(k)}|, \quad x_k = |a_{pp}^{(k)} - a_{qq}^{(k)}|, \quad y_k = (t_k^2 + x_k^2)^{1/2} \]

then

\begin{align*}
\cos^2 a_{pq}^{(k)} &= \frac{1}{2} \left(1 + \frac{x_k}{y_k}\right); \\
\sin^2 a_{pq}^{(k)} &= \frac{1}{2} \left(1 - \frac{x_k}{y_k}\right).
\end{align*}

Since $|a_{pq}^{(k)}| \leq \pi/4$, then $\cos a_{pq}^{(k)}$ will always be taken positive and $\sin a_{pq}^{(k)}$ will be of the same sign as $[2a_{pq}^{(k)}/(a_{pp}^{(k)} - a_{qq}^{(k)})]$. 
APPENDIX B  
Examples of Elimination Scheme [3]

Let \( n = 8 \) and \( k = 2 \), then the pairs \((p, q)\) are given by \{(2,3); (1,4); (7,5); (8,6)\} and \( R_2 \) of the form

\[
R_{11}^{(2)} \quad \cdots \quad R_{14}^{(2)}
\]
\[
R_{22}^{(2)} \quad R_{23}^{(2)}
\]
\[
- R_{23}^{(2)} \quad R_{33}^{(2)}
\]
\[
- R_{14}^{(2)} \quad R_{44}^{(2)}
\]

while for \( k = 7 \) the pairs \((p, q)\) are \{(8,1); (7,2); (6,3); (5,4)\} and \( R_7 \) is the form

\[
R_{11}^{(7)} \quad \cdots \quad R_{18}^{(7)}
\]
\[
R_{22}^{(7)} \quad R_{27}^{(7)}
\]
\[
- R_{27}^{(7)} \quad R_{77}^{(7)}
\]
\[
- R_{10}^{(7)} \quad R_{88}^{(7)}
\]
If the order of the matrix is odd, say $n = 7$, then for $k = 3$ the pairs $(p, q)$ are given by $\{(1,2); (7,3); (6,4)\}$ and $R_3$ is of the form

\[
\begin{bmatrix}
R_{11}^{(2)} & R_{12}^{(3)} \\
-R_{12}^{(3)} & R_{22}^{(2)} \\
R_{33}^{(3)} & R_{34}^{(3)} \\
-R_{43}^{(3)} & R_{46}^{(3)} \\
-R_{57}^{(3)} & R_{77}^{(3)}
\end{bmatrix}
\]
APPENDIX C  A Typical Calling Program
?USER=CACIEIGN
?COMPILE MCD/ASKO/JACOB/INRIVER WITH ASK LIBRARY
?BCL CARD

\$BEGIN

\$FILL 12A1

\$DEFINE PRINMTX=

CLC(1)1;
CADD(1) $C31;
CADD(1) $D41;
CADD(1) $C41;
CCCH(1) 151;
CLC(0)1;
CADD(0) $D41;
CADD(0) $D41;
CADD(0) 151;
DISPLAYR $C10=161;
LIT(2) =641;
CADD(1) $C21;
CRTRC(1) 241;
CADD(1) $C21;
CROT(1) 241;
\$THEM(0) =0111;

\$edefines

\$DEFINE NMAX=16#

MATA: BLK NMAX;
MATB: BLK NMAX;
MATP: BLK NMAX;
EIGV: BLK NMAX;
GERSH: BLK 11;
N: WDS 11;
SPW: WDS 11;
DATA 00101;
DATA "###THE ORIGINAL MATRIX###";
DATA "###THE DIAGNOL MATRIX###";
DATA "###THE EIGENVECTOR MATRIX###";
DATA "###NUMBER OF SWEPS REQUIRED###";
DATA "###PE I CONTAINS BOUND ON EVAL(I)###";
SWP: WDS 01;
\$START1

\$SET E,DR=E1 \$SET E,AND,E2
LIT(0) 1 N11 N1; % READ DIMENSION OF INPUT $C0,11; % SYSTEM FROM INPUT

LIT(0) =11;
STL(0) $D401; % FIXED PT 1 FOR PRINMTX DEF
SLIT(0) =N1;
LOAD(0) $C01;
CSUB(0) $D401;
STL(0) $D411; % N-1 FOR PRINMTX DEF

LIT(0) 1 MATA,MATA;
CRTRC(0) 241;
CADD(0) $D411; % SET UP ACAR 0 FOR INPUT INSTRN
CROT(0) 241 % USED TO READ MATA
LIT(0)   0*1:01   % ACAR 1 LOOP INDEX
CADD(1) $041;   % FROM 0 TO N-1
CRLTL(1) 24;   % CLEAR RGA FOR ZERO-FILLING MTX
%
RDINPT1 CLC(2);   % ACAR2 USED FOR PE ROW ADDR
CADD(2) $CO;   % CU ADDR FROM ACARO
CSHR(2) 61;   % PE ADDR TO $C2
STA 0(2);   % ZERO-FILL ROW OF MATRIX
INPUT $CO*1;   % READ MATRIX FROM INPUT
CRLTR(0) 24;   % RUMP ACARO
LIT(2) =64;   % TO PREPARE
CADD(0) $C2;   % FOR NEXT
CRLTL(0) 24;   % ROW OF MATRIX
CADD(0) $C2;   % INPUT
TXLTM(1) RDINPT1
%
LIT(0) 1,MESS1=1,MESS0;
DISPLAY $CO*32;
SLIT(3) =MATB; PRINTMTX; % THE ORIGINAL MTX
%
******
CALL JACOBI(MATA,MATB,MATP,EIGV,GERSH,N,SWP);%
%
******
LIT(0) 1,MESS2=1,MESS1;
DISPLAY $CO*32;
SLIT(3) =MATA; PRINTMTX; % THE EIGENVALUES
LIT(0) 1,MESS5=1,MESS4;
DISPLAY $CO*32;
LIT(0) 1,GERSH,GERSH;
CRLTR(0) 24;
CADD(0) $041;
CRLTL(0) 24;
DISPLAY $CO*16; % BOUNDS ON EIGENVALS
LIT(0) 1,MESS3=1,MESS2;
DISPLAY $CO*32;
SLIT(3) =EIGV; PRINTMTX; % THE EIGENVECTORS
LIT(0) 1,MESS4=1,MESS3;
DISPLAY $CO*32;
LIT(0) 1,MESS0=1,SWP   % ITERATION COUNT
DISPLAY $CO*16;
HALT;
?END
END START.
?USER=CACI\GEN
?COMPILE MCD/ASK/NEW/ACTRJ WITH ASK LIBRARY
?DATA
BEGIN
  FILL 12@1
  ZERD EQU $001
  ONE EQU $011
  TWO EQU $021
  SIXTATIFOU $031
  N EQU $051 % DIMENSION OF SYSTEM
  N1 EQU $061 % N MINUS 1
  M EQU $071 % [(N+1)/2]
  M2 EQU $091 % 2*M
  M2M1 EQU $091 % 2*M1
  M4M2I EQU $0101 % LOOP FROM 0 TO N-1
  ****COUNTERS, INDICATORS AND LIMITS
  ANTI EQU $0121 % ROUTING DISTANCE
  MKRT EQU $0131 % INDICATOR TO ENABLE CONVERGENCE TEST
  EROO绮 EQU $0141 % EVEN/ODD INDICATOR FOR N
  ICETI EQU $0151 % ITERATION COUNTER
  CONVIE EQU $0161 % CONVERGENCE FACTOR
  RATION EQU $0171 % RATIO OF SUM OF SQRS OF OFF-DIAG TO SUM OF
  SQUARES OF DIAG ELEMENTS
  ROUTI EQU $0181 % 64-N
  ****SWITCHES
  ODSMT EQU $0191 % =0 IF OFF-DIAG SUM, =1 IF DIAG SUM
  HSVTHCE EQU $0201 % ROUTING SWITCH (0=RT, 1=LFT)
  ****SAVE PATTERNS
  PDMODIE EQU $0211 % ENABLING PATTERN FOR PFS 0 TO N-1
  SAVI EQU $0221
  SAVEL EQU $0231
  SVMODIFOU $0241 % SAVE REGISTERS OR ADDRESSES
  ADRI EQU $0251 % ADDR OF MTX TO BE MULTIPLIED
  ADR2 EQU $0261 % ADDR OF MTX CONTAINING RESULT OF MULT
  INDEXFOU $0271 % INDEX USED IN (P,O) SETUP
  INLOOPI EQU $0281 % RETURN ADDR TO CALLING PROG
  SAVEO EQU $0291 % SAVE REGISTERS
  SAVE1 EQU $0311
  SAVE2 EQU $0321
  SAVE3 EQU $0331
  ANRAI EQU $0341 % ADDR OF THE INPUT MTX
  ANRAI EQU $0351 % ADDR OF MTX CONTAINING TEMPS RESULTS
  ADRP1 EQU $0361 % ADDR OF MTX CONTAINING PAIRS (P,O)
  ADRE EQU $0371 % ADDR OF EIGENVECTOR MTX
  ARG1 EQU $0381 % ADDR OF ROW TO CONTAIN ROWS ON EVAL
  ARDII EQU $0391 % ADDR OF SWEEP COUNT FOR OUTPUT
  ****NOTE: PROG CURRENTLY EXPECTS COSA, SINA, & PROW TO BE STORFD
  IN ORDER
  COSA RLK 11 % COSINES OF TRANSFORM MTX
  SINA RLK 11 % SINES OF TRANSFORM MTX
  PRLK RLK 11 % ROW INDEX OF ELEMENT TO BE ANNIHILATED
  TEMPI RLK 11 % TEMP STORAGE
  TEPPL RLK 11

-36-
ENI DATA 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63
SUMMI: WDS 01 *FILL 161
DBSAVE WDS 10: % SAVE AREA FOR $D32-$D3$ AND $C0-$C1

RANGE FILL:

ROUTINE ANGLE Computes the sines and cosines of the transformation. The
cosines and sines are stored in two rows of PF mem --COSA & SINA
and are determined by the values $PEN (Q) and $PROW (P) in each pe
i.e., $A(P,P), $A(Q,P), and $A(Q,Q) are determined by $PQ.

STL(3)  *SAVE3* % RETURN ADDR
LDX $PEN;
LDL(0)  *ANRA* % MAIN DIAGONAL TO RGR
LDH $O(O); % PEN1; SBSM $PROW;
CLC(2);% IME;
TAG  *SC2*;
SETC(1)  I; % SET I BITS WHERE P<0
STL(1)  *SAVE1*;
SAP: % RGA = ABS(Q-P)
IME;
SETC(3)  I; % IF N IS EVEN *SAVE=0
STL(3)  *SAVE3*;
SKIP  *BUMPX2*;

BRING ELEMENT A(P,P) TO PE Q

ANG1: IME;
SETC(3)  I; % SEE IF RGA=SC2
ZERT(3)  *BUMPX2*;
LDEE1  *SC2*;
LDA(1)  *SAVF1*;
CAND(1)  *SC2*;
LDEE1  *SC1*;
% TURN ON PE=S WHERE RGA=SC2 AND P<0
RTL 0(2); % P>Q DO RIGHT ROUTE OF $C2 TO GET A(P,P)
STH $TEMPPI;
LDL(1)  *SAVF1*;
COMPC(1);;
CAND(1)  *SC2*;
LDEE1  *SC1*;
% TURN ON PE=S WHERE RGA=SC2 AND P>0
LDA(3)  *STXTA*;
CSUB(3)  *SC2*;
% BRING RGR BACK INTO POSITION AND
CSUB(3)  *SCP*;
% DO LEFT ROUTE OF $C2 (RIGHT 64=$CP)
RTL 0(3); % TO GET A(P,P)
STH $TEMPPI;
LDB 0(0); % BRING BACK MAIN DIAGONAL
LDA(3)  *PEMONE; DLEEI  *SC3*;
% TURN FIRST N-1 PE=S ON
UMPX21ALIT(2) =1; % LOOP FROM 1 THRU N-1
LESST(2) *$5, ANG1;

COMPUTE SINES AND COSINES

-37-
LIT(1) = 2.0;
LDX PROW1;
LDA *O(0)1
MLKN $C11;
STA TEMP1;  % TEMP1 = 2*A(p, q)
LDX PENV1;
LDA *O(0)1
SRN TEMP1;
STA TEMP2;  % TEMP2 = A(q, p) = A(p, p)
EOR TEMP1;
LC1(1)
IAL $C11
SETC(3) J;  % SET I BITS WHERE TEMP1 & 2 DIFFER IN SIGN
LDA TEMP2;
JL;
SETC(1) J;  % CHECK IF ANY TEMP2=0.0
ZERT(1)  % J BITS ON WHERE TEMP2=0.0
LDL(9) $SAVFI;  % SET BITS IN $CO WHERE P<q
CAND(1) $C01;  % P<q AND TEMP2=0.0
CEXOR(3) $C11;  % CHANGE MODF STATUS OF THESE PE=S
OR TEMP1;  % SOME TEMP2=0 SEE IF CORRESPONDING
IL;
SETC(1) J;  % BRING TO ACAR 1
ZERT(1)  % IF NO TEMP1 & 2 BOTH = 0 JUMP
LDL(2) $SAVFI;  % NOTE THESE PE-S IN PATTERN TO SET
COK(2) $C11;  % COS & SIN = 1.0 AND 0.0 RESPECT.
STL(2) $SAVFI;  % WILL BE USED SHORTLY
SETE -$I, AND, EI;  % TURN OFF PE-S WHERE TEMP1=TEMP2=0
SETEl $E, AND, EI;  % TO AVOID ZERO DIVIDING
NOZERO: STL(3) % SAVE PATTERN
LDA TEMP2;
MLKN $SAI;
LDS $SAI;
LDA TEMP1;
MLKN $SAI;  % TEMP1 = TEMP1
AORN $SI;
CALL SORT64();  % SORT(TEMP1*2+TEMP2*2)
LDS $SAI;
LDA TEMP2;
SAP1
DVRN $SI;
LIT(1) = 1.0;
TAG $C11;
SETC(2) J;
ZERT(2)  % SVTEMP;
LDEE1 $C21;  % INSURE ARC(COS) & ARC(SIN) LEQ 1.0
LDA $C11;
LDDL(2) $PEMNDE1;
LDEE1 $C21
SVTEMP: STA TEMP1;  % SAVE TEMP2/(ABOVE ROOT)
ADHN $C11;
LIT(0) = 0.5;
MLKN $C01;
CALL SORT64();  % COSINE
STA COSA1;
LDA $C11

-38-
SBHN  TEMPP;  00017200
MLHN  $C0;  00017300
CALL SORT4A();  % SINF  00017400
STA  SINA;  00017500
LDL(3)  *SVMN;  00017600
LDLE1  *C3;  00017700
CHSA;  00017800
STA  SINA;  00017900
CLC(2);  00018000
LDL(3)  *SAV1;  00018100
LDL(1)  *C3;  00018200
LDA  *C1;  00018300
STA  COSA;  % COS=1.0  00018400
LDA  *C2;  00018500
STA  SINA;  % SIN=0.0  00018600
RESETI  00018700
LDL(3)  *PENONE1;  00018800
LDL(1)  *C3;  00018900
LDL(3)  *SAVE3;  00019000
EXCHL(3)  *ICRI;  00019100
******************************************************

**********MULTIPLY ROUTINE **********************

MULTIPLY TRANSFORMATION MATRIX BY MATRIX IN ADR1=RESULT IN MATRIX IN ADR2

MULTIPLY IS DONE AS FOLLOWS=-
  + ELEMENT = PEN: IF ROW COS * ROW 1PEN: IF MTX IN ADR1
  + ELEMENT = PEN: IF ROW SIN * ROW 1PEN: IF MTX IN ADR1
  RESULT IN ROW 1PEN: IF MTX IN ADR2

MATX IN ADR1  MATA  MATH  EIGV
MATX IN ADR2  MATA  MATH

MULTPL:FULL:
  STL(3)  *SAVE3;  00020200
  LDL(0)  *SVMN;  00020300
  LDX  PEN1;  00020400
  MULTI:
  SLIT(1)  *COSA;
  CADD(1)  *C0;
  LOAD(1)  *C2;  % COS (P,Q)
  LDL(3)  *ADR1;
  CADD(3)  *CO;
  LDA  O(3);  % LOAD ROW (PEN) OF MATX IN ADR1
  MLHN  *C2;
  LDA  *A;
  CADD(1)  *SIXT4;
  LOAD(1)  *C2;  % SIN(P,Q)
  CADD(1)  *SIXT4;
  LOAD(1)  *C3;  % VALUE OF ROW1 IN PFO
  LDL(1)  *ADR1;
  CADD(1)  *C3;
  LDA  O(1);  % LOAD ROW (PEN) OF MTX IN ADR1
  MLHN  *C2;
  ADRH  *S;
  IF THIS IS POST MULTIPLY (CALL1) = SKFW MATRXA
  LDL(1)  *ADR1;

-39-
MULTI

LDL(2)  $ADR2!
EQLXF(1) $D34,MULT1
CLC(3);  % NO NEED TO SKFW FIRST ROW
EQLXT(0) $C3,MULT1
STL(0)  $AMT;  % SKW MATX IN ADR2 TO PEPARF FOR TRANPOSE
SLIT(3) =ROUTE;
FXCHL(3) $ICR;

STA *D(2);  % STORE RGA SKewed IN MATX IN ADR2
CLRA;
LDA $X;
LDL(3) $D1;
STL(3) $AMT;
CLC(3);  % STORE RGA IN MTX IN ADR2
SLIT(3) =ROUTE;  % ROUTE PE INDICES 1 RIGHT
FXCHL(3) $ICR;
SWAP;
LDA $B;
SKIP $CKACO;

MULTI

CADD(2) $CO;
STA *D(2);  % STORE RGA IN MTX IN ADR2
CKACO
TXLTM(0), MULT1
LDL(3) $SAVE3;
FXCHL(3) $ICR;

*******************************************************

ROUTE: FILL;
STL(0)  $SAVE0;
STL(1)  $SAVE1;
STL(3)  $SAVE3;
LDL(0)  $AMT;
RTL $A,0(0);
LDA $R;
LDS PEN;

SEE IF LEFT($D31=1) OR RIGHT ($D31=0) ROUTE

CLC(1);
EQLXT(1) $D20,RIGHT;
LDL(3) $N1;
CADD(0) $N11;
CSUB(0) $STXT4;
ISG $CO;
SKIP $SETIT;

RIGHT:
LDL(3) =ROUTE;  % 64-N
ISL $CO;

SETIT:
SETE $AND_E1;
SETE1 $AND_E1;
CLHA;
RTL $D(3);
LDA $R;
LDL(3) $PBMONE;
LDL(1) $C3;
LDL(0) $SAVE0;
LDL(1) $SAVE1;
LDL(3) $SAVE3;
FXCHL(3) $ICR;

-40-
ADIT: FILL1
ADIT CALCULATES THE SUM OF THE OFF-DIAGONALS SQUARE AND
DIVIDES IT BY THE SUM OF THE DIAGONALS SQUARED

STL(3) SAVE31
CLC(1); STL(1) AINSW1
LDL(0) SVLOOP; SETF E,OR,-E1
SETE1 E,ANN,E1
CLHA3
LDS $A
LDL(3) PFMONE1
LDEF1 $C31
LDL(2) ANRA1
CALC(2) $CO;
LDA PC(2)1
MLKN $A
ADRN $S;
LDS $A
TXEFM(0) ASI1; X END ROW-SUM

ADIT1
SETE E,OR,-E1
ADIT3
LDL(0) $INF;
LDS $R;
ADRN $S;
CADD(0) $C0;
LDL(1) $SIXTA4;
EQLX(0) $R1,ANII1 X END LOG-SUM
RTL $S,0(0); X USED FOR BOTH OFF-DIAGONALS AND DIAGONALS
LDS $R;
ADRN $S;
CALC(0) $CO;
LDL(1) $SIXTA4;
EQLX(0) $R1,ANII1 X END LOG-SUM
LDL(1) AINSW1
FQLXT(1) $O1,ANII2:
S1A TFMP1;
CLHA3
LDS $A;
LDL(1) PFMONE1 X PICK UP THE A11,11-S
LDEF1 $C1;
LDX PVN1:
LDL(2) AANRA1
LDA PC(2);
MLKN $A;
LDL(1) ONE1
STL(1) AINSW1
SKIP AINTO1:
LDA TEMMP1
SHRN $S; X SUBTRACT THE A11,11
CLC(0); THE $CO1
SETC(0) II
ANEST(0) A,NII1
DVHN $S; X CONVERGENCE FACTOR
LOC(0) $A1;
STL(0) *RATIO1

-42-
ADI31    LDL(3)  \*SAVE31
EXCHL(3)  \*SICR1

******
GERSH: FILL1    \* GERSH FINDS A ROUND ON THE EIGENVALUES
******

STL(3)  \*SAVE31
SETI E.0P,-E1
SETI1 E. AND,E1
CLRA1
LDS  \*SA1

******
% THE RADIUS OF THE DISK ACC. TO GERSHGORIN'S TH' CONSISTS
% THE SUM OF THE ABS. VALUE OF THE OFF- DIAG ELEMENTS

% LOCATED IN ROW I FOR WHICH A(i,i)*EIGENVALUES, THE ROWSUM
% OF EACH ROW IS FOUND. THE CENTER OF THE DISK IS THE EIGENVALUE

******
LDL(1)  \*SVL0NP1

******
% FINDING THE SUM OF THE ROW IS EQUIVALENT TO FINDING THE SUM
% OF THE COLUMNS, SINCE THE MATRIX IS SYMMETRIC

******
GERII: LDL(0)  \*PMODE1
CCB(0)  \*O(1);
LDEE1  \*CO1;
LDL(2)  \*ADRA1
CAUD(2)  \*C11;
LDA  \*O(2);
SA11;
ADRI1  \*SA1;
LDS  \*SA1;
XEFM(1)  \*GERH;
LDL(0)  \*PMODE1;
LDEE1  \*CO1;
LDL(0)  \*ADRI1;
STA  \*O(0);  \* STORE BOUNDS ON EIGENVALUES
LDL(3)  \*SAVF31;
EXCHL(3)  \*SICR1

******

******************************************************************************
JACOBIENTRY): FILL1    \* SAVE \*CO, \*C1, ADR \$D32-\$D39
CHIS  641
SETI E.0R,-E1
SETI1 E. AND,E1
STL(3)  \*RETURN1
CLC(3);SLIT(3)  \*ADRSAV+R1
STURE(3)  \*CO1
ALIT(3)  \*I1
STURE(3)  \*C11
CLC(3);SLIT(3)  \*ADRSAV1
LIT(0)  \*17,01
STURE(3)  \$D32(0);
ALIT(3)  \*I1
XEFM(0)  \*SA1
LIT(0)  \*4,01
CLC(3);
LOAD(2) &C3;  \% $C2$ SET IN CALI STMT FROM MAIN PRG
CCHR(3) \#1;
STL(3) \$D34(0);  \% AND OF DIAG*TMP* PAIR AND FTG V MTXS
ALIT(2) 1;
TEXFM(0) $SA1$;
LOAD(2) &C3;
LOAD(3) &C3;
STL(3) \#N;  \% DIMENSION OF SYSTEM
ALIT(2) 1;
LOAD(2) &C3;
STIL(3) \#ADR1;  \% ADDR OF SWEEP COUNT - TO BE OUTPUT

INITIALIZE CONSTANTS AND OTHER PARAMETERS

CL(C0); ZER0;
STL(0) \#RSWITCH;
STL(0) \#C0NT;
STIL(0) \#KCONV;
LIT(0) \#1;
STL(0) \#ONE;
CAUD(O) \#C0;
STL(0) \#TWO;
LIT(0) \#4;
STL(0) \#BOUND;
LIT(0) \#64;
STL(0) \#IXTA;
CSHR(0) \#N;
STL(0) \#RINT;
LDD(0) \#N;
CSUR(0) \$N;
STL(0) \#N1;  \% N MINUS 1
LIT(0) \#0,1,0;
CAUD(0) \#N1;
CRUTL(O) \#24;
STL(0) \#SVLNP;
LDD(0) \#N;
CAUD(0) \#01;
CCHR(0) \#1;
STL(0) \#M1 \& M1^+(N+1)/2 \] 
CSHL(0) \#1;
STL(0) \#M2;  \% 2^+4
CSUR(0) \#N;
STL(0) \#EVENn; \% =0 IF N EVEN, =1 IF ODD
LDD(0) \#M2;
CSUR(0) \#01;
STL(0) \#M2M1;
CSHL(0) \#1;
STL(0) \#M4M2;  \% 4*M=2

TURN ON FIRST N=1 PS=5

LDA \#PE1;
LDD(0) \#N;
IAL \#C0;
SETC(O) \#1;
STL(0) \#PEMN=E1
LDZ1 $C01

* INITIATE EIGENVECTOR MATRIX *

CLRA;
LDL(O)  * SVLONP1 % LOOP FROM 0 TO N-1
LDL(1)  * ANRE1
EVINIT: STA 0(1);
CADD(1)  *MNF1;
TXLM(0)  * EVINIT;
LDX PEN;
LIT(0)  =1,01;
LDA $C01;
LDL(1)  * ANRE1;
STA *0(1);;

SET UP PAIRS (P,Q) FOR ANNihilation

LDL(1)  *PEMONE1;
LDZ1 $C11;  % TURN ON PE-S 0 TO N-1

SET UP LOOP COUNTER TO GO FROM 1 TO 2M-1

LIT(0)  =0,1,01;
CADD(0)  *M2M11;
CROL(0) 24;  % LOOP FROM 1 TO 2M-1
CADD(0)  *D11;
LDL(1)  *M4M21;
STL(1)  *INFX1;

SETUP: EQLXFA(7) $D7*KNOTM1;
CADD(1)  *M21;
CSUB(1)  *ONE1;
STL(1)  *INFX1 % 6*M=3 RESET WHEN K=M
KNOTM1: CSUB(1)  *CO1;
CSUB(1)  *CO1;
LDA $C11;  % INDEX = 2*K
SHM PEN;  % INDEX = 2*K - 0
LDL(2)  *M2M11;
IAL $C21;
SETE I, AND, E1;
SET1 E, AND, E1;
SRM $C21;  % IF P GEQ 2*M-1 SKIP 2*M-1
LDL(2)  *PEMONE1 % TURN FIRST N-1 PE-S ON
LDZ1 $C21;
LDL(2)  *APE1;  % GET AND OF PAIR MATRIX
CADD(2)  $CO1;  % GET I-TH ROW AND STORE P=Q
STA 0(2);;
CLC(3);  % IF N IS ODD SKIP TO PEMONE
EQLXFA(3) $D14,BUMP0;

CSHL(2)  $;  % BACK TO CU ADDR
CADD(2)  *N11 % + (N=1)
CSUR(2)  *CO1 % -K
LDL(3)  *N11;
STORE(2)  *CO1 % STORE N-1 IN PF(N-1-K)
CADD(2)  $CO1 % CU ADDR PRW + (N-1)
BEGIN  ITERATION  LOGIC

CRLF

POST  MULTIPLY  MATRIX  A  BY  THE  TRANSFORMATION  MATRIX
ACTUALLY  PREMULTIPLY  A  BY  TRANSPOSE  OF  TRANSFORM  Mtx
THEN  TAKE  TRANSPOSE  OF  PRODUCT

BEGIN  ITERATION  LOGIC
**MULTIPLY TRANSPOSE OF TRANSFORM MTX BY MATRIX INL**

```
XA YIELDING NEW MATRIXA WITH ALL A(P,Q) ANNihilated (HOPEFULLY)
L0L(2)  *AR
STL(2)  *AR1
L0L(2)  *AR2
STL(2)  *AR3
CLC(3)  *RSWITCH: %RESET ROUTING SWITCH FOR RIGHT ROUTING
SLIT(3) =MULTPL:
EXCHL(3) =CIR
L0L(2)  *PRMONE:
L0L(3)  *SAV1:
COMPC(3):
CAND(3)  *CP:
LDEE1  *C3:
L0L(0)  *AR:
CLC(1):
L0X  *PROW:
LDA  *C1:
STA *O(0):
LDEE1  *C2:
```

**UPDATE EIGENVECTOR MATRIX**

```
L0L(2)  *AR:
STL(2)  *AR1:
L0L(2)  *AR2:
STL(2)  *AR3:
CLC(3):
SLIT(3) =MULTPL:
EXCHL(3) =CIR:
L0L(0)  *AR:
L0L(1)  *AR:
L0L(2)  *SYNLOOP: %LOOP FROM 0 TO N-1
```

```
LDA 0(0):
STA 0(0):
CADD(0)  *ONE:
CADD(1)  *ONE:
TXLTM(2)  *COPY:
L0L(0)  *INLOOP:
TXLTM(0)  *LOOP:
```

```
CLC(3):
SLIT(3) =ADDIT:
EXCHL(3) =CIR:
ONE(0)  *+1:
```

**SKIP:**

```
L0L(0)  *KONV:
EQLXF(0)  *D0 *CKICNT:
IT2  =0000000000000001:
LDA  *C2:
L0L(2)  *RATIN:
```

```
L0S  *C2:
MLKN  *S:
L0C(1)  *RA:
```

```
STL(1)  *KONV: %KONV = RATIO X 1.0E-1K
CKICNT =L0L(0)  *ICNT 1
```
```
```
LSSF(0) $013,+1; % DON'T CHECK RATIO WHILE ICNT < BOUND
JUMP SWEEP;
LDL(1) ,RATIO* % ICNT IS GEQ BOUND
LDL(2) ,KCONV;
LDA $C1;
IAL $C2; % SET I RITS IF RATIO<KCONV
SEI(T(1) I; % IF RATIO NOT LESS THAN KCONV CONTINUE
JUMP SWEEP;

% ****** END OF SWEEP LOGIC **************

EXIT: CLC(3);
SLIT(3) = GERSH;
FXCHL(3) $ICR;
LDL(3) ,IUNIT;
LDL(1) ,AORTI;
STURE(1) $C3;

EXIT: CLC(3);
SLIT(3) ANHSAV1
BIN(3) $032;
CLC(3);
SLIT(3) ANHSAV81
LOAD(3) $C1;
ALIT(3) 11
LOAD(3) $C1
LDL(3) ,RETURN; % RETURN TO MAIN PROGRAM
FXCHL(3) $ICR;
END JACOBI.

-47-
This revised version of CAC Document #21 supercedes the document dated November 8, 1971.

Several methods have been proposed to enable the computation of eigenvalues and eigenvectors of large, real symmetric or complex Hermitian matrices on ILLIAC IV. One of the most effective methods in the utilization of parallel computations has proven to be a modified Jacobi algorithm. This document presents yet another modification which exploits the parallelism of ILLIAC IV to a greater extent than has been previously done. Flow charts and the assembly language (ASK) routine JACOBI are included in the report.
<table>
<thead>
<tr>
<th>KEY WORDS</th>
<th>LINK A</th>
<th>LINK B</th>
<th>LINK C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix Algebra</td>
<td>ROLE</td>
<td>WT</td>
<td>ROLE</td>
</tr>
</tbody>
</table>

UNCLASSIFIED
Security Classification