

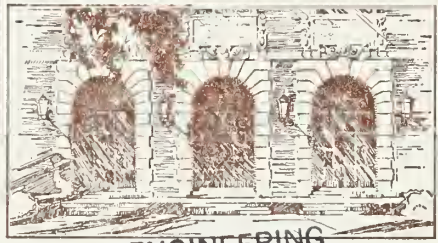


LIBRARY OF THE  
UNIVERSITY OF ILLINOIS  
AT URBANA-CHAMPAIGN

510.84

Il 63c

no. 11-20



ENGINEERING

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The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.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

ENGINEERING

CONFERENCE ROOM

AUG 16 1977

AUG. 23 RES

NOV 16 1984



510.84  
I263c  
no.19

Engin.

CONFERENCE ROOM

ENGINEERING LIBRARY  
UNIVERSITY OF ILLINOIS  
URBANA, ILLINOIS

# Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN  
URBANA, ILLINOIS 61801

CAC Document No. 19


ILLIAC IV CODES FOR JACOBI AND  
JACOBI-LIKE ALGORITHMS

By

Winfried H. Bernhard

November 5, 1971





Digitized by the Internet Archive  
in 2012 with funding from  
University of Illinois Urbana-Champaign

<http://archive.org/details/illiacivcodesfor00bern>

ILLIAC IV CODES FOR JACOBI AND  
JACOBI-LIKE ALGORITHMS

By

Winfried H. Bernhard

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

November 5, 1971

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. DAHCO4 72-C-0001.

Approved for public release; distribution unlimited.





## ABSTRACT

I. Modified JACOBI's Method [1] for finding the eigenvalues and eigenvectors of a Hermitian matrix is a well-suited algorithm for ILLIAC IV. It is based on the idea of subjecting the matrix to a series of orthogonal transformations that eliminate the off-diagonal elements such that the matrix under consideration becomes diagonal. ILLIAC IV with its parallel structure provides a tool for eliminating  $n$  off-diagonal elements in one single sweep, so that the whole process of making the matrix diagonal becomes very rapid.

II. Modified EBERLEIN's Method for real matrices:

While Jacobi's method is applied to Hermitian matrices, Eberlein's method [2] applies a series of similarity transformations to a non-symmetric matrix until it is practically normal. The resultant normal matrix is then reduced to the diagonal form [2], obtaining the eigenvalues and eigenvectors. The results, of course, are best when the matrix can be made diagonal.

This document presents a brief theoretical background and a detailed description of both programs, written in ASK, including the flow-charts.



# TABLE OF CONTENTS

	Page
I. Theoretical Background . . . . .	1
A. Modified JACOBI's Method . . . . .	1
B. Modified EBERLEIN's Method . . . . .	6
II. Implementation . . . . .	9
A. Jacobi: Finding Eigenvalues of a Hermitian Matrix. . .	9
1. Storage Scheme . . . . .	9
2. Computation: a) The calling program . . . . .	13
b) The subroutine. . . . .	15
3. Flowcharts . . . . .	23
B. Eberlein: Normalizing a Matrix. . . . .	44
1. Storage Scheme . . . . .	44
2. Computation . . . . .	45
3. Flowcharts . . . . .	50
Bibliography . . . . .	66
APPENDIX A. The Calling Program . . . . .	67
APPENDIX B. The Subroutine EIGEN, Jacobi's Method . . . . .	70
APPENDIX C. The Subroutine EBERL, Jacobi-Like Method. . . . .	94
APPENDIX D. Results from EIGEN . . . . .	126
a) The Original Matrix . . . . .	127
b) The Diagonalized Matrix . . . . .	128
c) The Eigenvector Matrix . . . . .	129



## LIST OF FLOWCHARTS

Flowchart No.	Page
1. MAINPROGRAM: of Subroutine EIGEN . . . . .	23
2. Flowcharts of the Individual Routines . . . . .	27
a) RWSM . . . . .	27
b) ANYR . . . . .	29
c) ANGLE . . . . .	31
d) MULTPL . . . . .	35
e) SHUFL . . . . .	37
f) SAMUL . . . . .	39
3. Eberlein's Algorithm (MAINPROGRAM) . . . . .	50
4. Procedures to Eberlein's Method . . . . .	55
a) FNDMX (C,N,MAXR,MAXC,G) . . . . .	55
b) FINDC (A,C,N) . . . . .	56
i) MLTRPS . . . . .	56
ii) TRPS (A,B,N) . . . . .	57
c) ANGL (ANMAT,C,N) . . . . .	58
d) HYANG (ANMAT,A,N) . . . . .	60
e) MULSA . . . . .	63



# I. THEORETICAL BACKGROUND

## A. Modified JACOBI's Method:

The classical Jacobi Method reduces a symmetric matrix to a diagonal matrix by a series of orthogonal transformations:

$$A_{r+1} = \phi_r A_r \phi_r^t, \quad \phi_r \phi_r^t = I.$$

Each transformation  $\phi_r A_r \phi_r^t$  eliminates two identical off-diagonal elements. It is, however, possible [1] to eliminate  $n$  off-diagonal elements,  $n$  being the order of the matrix  $A$ , by one orthogonal transformation. This can be achieved if the transformation matrices  $\phi_r$  are of the form

$$\phi_r = \text{diag} (T_0, T_1, T_2, \dots, T_{n/2})$$

assuming that  $n$  is even and

$$T_k = \begin{bmatrix} \cos \alpha_k & \sin \alpha_k \\ -\sin \alpha_k & \cos \alpha_k \end{bmatrix}$$

The matrix  $A_{r+1}$  will therefore consist of  $2 \times 2$  submatrices of the form

$$(A_{pq})_{(r+1)} = (T_p A_r T_q^t)_{pq} \quad p, q = (0, 1, \dots, n/2 - 1).$$

For the diagonal submatrix



$$(A_{kk})_r = \begin{bmatrix} a_{2k,2k}^{(r)} & a_{2k,2k+1}^{(r)} \\ a_{2k,2k+1}^{(r)} & a_{2k+1,2k+1}^{(r)} \end{bmatrix}$$

$\cos \alpha_k^{(r)}$  and  $\sin \alpha_k^{(r)}$  are chosen such that

$$\cos^2 \alpha_k^{(r)} = \frac{1}{2} \left( 1 + \frac{X_k}{Y_k} \right) \quad \text{and} \quad \sin^2 \alpha_k^{(r)} = \frac{1}{2} \left( 1 - \frac{X_k}{Y_k} \right)$$

where

$$X_k = a_{2k,2k}^{(r)} - a_{2k+1,2k+1}^{(r)}, \quad Y_k = \left( t_k^2 + X_k^2 \right)^{1/2}$$

with

$$t_k = 2a_{2k,2k+1}^{(r)}$$

Since  $|\alpha_k^{(r)}| \leq \pi/4$ , then  $\cos \alpha_k^{(r)}$  will always be taken positive and  $\sin \alpha_k^{(r)}$  will be of the same sign as

$$\tan 2\alpha_k^{(r)} = \frac{2a_{2k,2k+1}^{(r)}}{a_{2k,2k}^{(r)} - a_{2k+1,2k+1}^{(r)}}.$$

With this transformation-matrix  $T_k$  the new matrix  $(A_{kk})_{r+1}$  will be of the form

$$(A_{kk})_{r+1} = \begin{bmatrix} a_{2k,2k}^{(r+1)} & 0 \\ 0 & a_{2k+1,2k+1}^{(r+1)} \end{bmatrix}.$$

After eliminating  $n$  off-diagonal elements of  $A$ , the matrix must be prepared for another transformation by applying the orthogonal transformation:

$$A'_{(r+1)} = \psi A_{(r+1)} \psi^t, \quad \psi \psi^t = I,$$

where

$$\psi = \left[ \begin{array}{cc|cccc} 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ \hline 0 & 0 & & & & & \\ \cdot & \cdot & & & & & \\ \cdot & \cdot & & & & & \\ \cdot & \cdot & & & & & \\ 0 & 0 & & & & & \\ \hline 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 \end{array} \right]$$

I  
(n-2) x (n-2)  
identity matrix

This permutation shifts the second row and second column into the place of the last row and last column, respectively. In this way, new elements are brought into the off-diagonal positions, and  $A'_{(r+1)}$  is ready for the transformation.

$$A_{(r+2)} = \varphi_{r+1} A'_{(r+1)} \varphi_{r+1}^t.$$

In order to subject all off-diagonal elements to this orthogonal transformation, the matrix  $A$  is exposed to a further transformation  $\Gamma$

$$\Gamma \Gamma^t = I,$$

after  $(n-2)$  orthogonal transformations  $\psi$  have been performed.  $\Gamma$  is given by

$$\Gamma = \left[ \begin{array}{cc} 0 & I_1 \\ I_2 & 0 \end{array} \right]$$

with  $I_1$  an  $(n-m) \times (n-m)$  identity matrix and  $I_2$  an  $m \times m$  identity matrix, where  $m$  is determined by

$$m = \text{index } i \text{ of } \max_{i \neq j} \sum_{j=1}^n |a_{ij}|,$$

The convergence of  $A$  toward a diagonal matrix  $WAW^t$  is the fastest, since  $\Gamma A \Gamma^t$  rearranges the matrix such that the largest off-diagonal element is eliminated first.

The matrix is sufficiently made diagonal if the ratio

$$\eta = E/D$$

is less than an arbitrarily small number

$$\xi; \quad \xi = 10^{-8} (E_0/D_0)$$

where  $E$  is the sum of the squares of the off-diagonal elements and  $D$  is the sum of the squares of the diagonal elements.

$E_0$  and  $D_0$  are calculated from the original matrix. The above value of  $\xi$  has proven to be sufficient. The almost diagonal matrix  $A_m$  will be of the form

$$A_m = WAW^t$$

where

$$W^t = (\varphi_{m-1} \dots \Gamma \varphi_{n-1} \dots \psi \varphi_2 \psi \varphi_1)^t$$

is the matrix whose columns are the eigenvectors. Since the number of transformations is finite, the resultant matrix  $A_m$  has the form:

$$A_m = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} + \begin{bmatrix} \epsilon \end{bmatrix}$$

To have bounds for these eigenvalues one is referred to Gershgorin's theorem which states that if  $\lambda$  is an eigenvalue of an arbitrary  $n$ -rowed matrix  $A = (a_{jk})$ , then for some  $k$ , ( $1 \leq k \leq n$ ),

$$|a_{kk} - \lambda| \leq |a_{k,1}| + \dots + |a_{k,k-1}| + |a_{k,k+1}| + \dots + |a_{kn}|.$$

For each  $k = 1, 2, 3, \dots, n$  this inequality determines a closed circular disk, whose center is the eigenvalue  $\lambda_k$  and whose radius is given by the sum of the absolute values of the elements in row  $k$  excluding  $a_{kk} = \lambda_k$ .

It may be added that the eigenvalue problem for a complex Hermitian matrix may be reduced to that of real symmetric matrices.

Let the  $n \times n$  complex Hermitian matrix  $A$  be denoted by

$$A = B + i C$$

where  $B$  is real symmetric ( $B = B^t$ ), and  $C$  is skew-symmetric ( $C = -C^t$ ). Then the  $2n \times 2n$  real symmetric matrix  $A^1$  given by

$$A^1 = \begin{bmatrix} B & \vdots & -C \\ \text{---} & \vdots & \text{---} \\ C & \vdots & B \end{bmatrix}$$

has the eigenvalues  $\lambda_1, \lambda_1; \lambda_2, \lambda_2; \dots; \lambda_n, \lambda_n$  and to each  $\lambda_j$  there correspond two orthogonal eigenvectors

$$\begin{bmatrix} u_j \\ v_j \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} v_j \\ -u_j \end{bmatrix}.$$

If  $\lambda_j$  is the eigenvalue of  $A$  then  $(u_j + iv_j)$  is the corresponding eigenvector.

#### B. Modified EBERLEIN's Method:

This method can be stated briefly as follows [2]: "A matrix  $A$ , which can be made diagonal, is normalized by subjecting it to a sequence of similarity transformations  $A_{e+1} = U_e^{-1} A_e U_e$ , such that  $A_{e+1}$  is arbitrarily close to being normal, i.e., the matrix  $C_e = A_e A_e^t - A_e^t A_e$  is arbitrarily small. Once the matrix is normal, it can be subjected to algorithms like Jacobi's method to reduce the matrix to a diagonal form and, thus, obtain the eigenvalues and eigenvectors of  $A$ ."

The transformation matrices  $U_e$  are given by  $U_e = M_e P_e Q_e$ , where

(1)  $M_e$  is a permutation matrix determined as follows:

Let  $A'' = M_e^t A M_e$  and  $C'' = A'' A''^t - A''^t A''$ , then  $M_e$  is chosen such that each  $2 \times 2$  diagonal submatrix  $C''_{kk}$  has an element  $c''_{2k-1,2k}$  of at least average value of all the off-diagonal elements of  $C''$ . For example, in order to bring the off-diagonal element  $c_{uv}$ , ( $u < v$ ), of maximum absolute value, in the position (1,2),  $M_e$  is given by  $I_{1u} I_{2v}$ , where  $I_{ij} = I - (e_i - e_j)(e_i - e_j)^t$ . Essentially  $I_{ij}^t \cdot A \cdot I_{ij}$  has the  $i$ -th and  $j$ -th rows and columns of  $A$  exchanged.

(2)  $P_e = \text{diag} (T_2^{(e)}, \dots, T_{n/2}^{(e)})$

with

$$T_k^{(e)} = \begin{bmatrix} \cos y_k & \sin y_k \\ -\sin y_k & \cos y_k \end{bmatrix} (e).$$

If  $y_k$  is determined by

$$\tan 2y_k = \left( \frac{c_{2k-1,2k-1} - c_{2k,2k}}{2c_{2k-1,2k}} \right)_{(e)},$$

where  $c_{ij}$  are the elements of the matrix  $C = AA^t - A^t A$ , and if  $\cos 2y_k$  is of the same sign as  $c_{2k-1,2k}$ , then  $(c_{2k-1,2k})_{(e+1)}$  attains its maximum value.

$$(3) \quad Q_e = \text{diag} \left[ S_1^{(e)}, S_2^{(e)}, \dots, S_{n/2}^{(e)} \right]$$

with

$$S_1^{(e)} = S_2^{(e)} = \dots = S_{n/2}^{(e)} = \begin{bmatrix} \cos hx_e & \sin hx_e \\ \sin hx_e & \cos hx_e \end{bmatrix}$$

$x_e$  is derived from:

$$\tanh 4x_e = -2 \cdot K_2(A'_e)/K_1(A'_e),$$

where

$$A'_e = (M_e P_e)^t A_e (M_e P_e),$$

$$K_2 = \sum_{k,m} D_{km} E_{km} \text{ and } K_1 = \sum_{k,m} (D_{km}^2 + E_{km}^2)$$

with

$$D_{km} = (a_{2k-1,2m-1} - a_{2k,2m})$$

and

$$E_{km} = (a_{2k-1,2m} - a_{2k,2m-1}).$$

It can be proved [2] that  $A_e$  approaches a normal matrix as  $e \rightarrow \infty$ .

Considering only real matrices then for the practically normal matrix A, any diagonal submatrix

$$\tilde{A}_{pq} \begin{bmatrix} \tilde{a}_{pp} & \tilde{a}_{pq} \\ \tilde{a}_{qp} & \tilde{a}_{qq} \end{bmatrix}$$

is also normal, where either

$$\text{a) } \tilde{a}_{pq} = \tilde{a}_{qp} \quad \text{or}$$

$$\text{b) } \tilde{a}_{pq} = -\tilde{a}_{qp} \quad \text{and} \quad \tilde{a}_{pp} = \tilde{a}_{qq}$$

A generalized Jacobi method is then used to reduce A to the diagonal form hence obtaining both the eigenvalues and eigenvectors [2].



## II. IMPLEMENTATION

### A. Jacobi: Finding Eigenvalues of a Hermitian Matrix

#### 1. Storage Scheme

To demonstrate the storage scheme let us look at an example. Let ILLIAC IV, for the sake of demonstration, be a 6 (six) PE machine.

Assuming all preliminary tests (see section 2.b.1-3, p. 14) are executed, then BASE, the  $N \times N$  matrix for which the eigenvalues are sought will be of the form:

BASE: in PE memory

$$\begin{array}{r}
 \text{PE:} \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \\
 \left( \begin{array}{cc} a_{00} & a_{01} \end{array} \right) \quad a_{02} \quad a_{03} \quad a_{04} \quad a_{05} \\
 \left( \begin{array}{cc} a_{01} & a_{11} \end{array} \right) \quad a_{12} \quad a_{13} \quad a_{14} \quad a_{15} \\
 a_{02} \quad a_{12} \quad \left( \begin{array}{cc} a_{22} & a_{23} \end{array} \right) \quad a_{24} \quad a_{25} \\
 a_{03} \quad a_{13} \quad \left( \begin{array}{cc} a_{23} & a_{33} \end{array} \right) \quad a_{34} \quad a_{35} \\
 a_{04} \quad a_{14} \quad a_{24} \quad a_{34} \quad \left( \begin{array}{cc} a_{44} & a_{45} \end{array} \right) \\
 a_{05} \quad a_{15} \quad a_{25} \quad a_{35} \quad \left( \begin{array}{cc} a_{45} & a_{55} \end{array} \right)
 \end{array}$$

To calculate  $\cos \alpha_k$ ,  $\sin \alpha_k$  (see section 2.b.4, p. 14) the matrix is partitioned into  $2 \times 2$  submatrices; i.e., the procedure ANGLE considers only the elements

$$a_{2k,2k}, a_{2k+1,2k+1}, a_{2k,2k+1}, \quad (k = 0, 1, 2, \dots, \frac{n}{2} - 1).$$

To achieve greatest efficiency, the algorithm calculates  $\cos \alpha_k$ ,  $\sin \alpha_k$  in pairs of PEs, i.e.,

PE	0	1	2	3	4	5
	$\cos \alpha_1$	$\cos \alpha_1$	$\cos \alpha_2$	$\cos \alpha_2$	$\cos \alpha_3$	$\cos \alpha_3$
	$-\sin \alpha_1$	$\sin \alpha_1$	$-\sin \alpha_2$	$\sin \alpha_2$	$-\sin \alpha_3$	$\sin \alpha_3$

In doing so, the anglematrix ANMAT is formed and unnecessary routing is avoided. ANMAT will be of the form:

ANMAT: in PE memory

PE	0	1	2	3	4	5
	$\cos \alpha_1$	$\sin \alpha_1$	0	0	0	0
	$-\sin \alpha_1$	$\cos \alpha_1$	0	0	0	0
	0	0	$\cos \alpha_2$	$\sin \alpha_2$	0	0
	0	0	$-\sin \alpha_2$	$\cos \alpha_2$	0	0
	0	0	0	0	$\cos \alpha_3$	$\sin \alpha_3$
	0	0	0	0	$-\sin \alpha_3$	$\cos \alpha_3$

Since ANMAT is of this tridiagonal nature, special treatment is required to perform the transformation:

$$(\text{ANMAT})^t \cdot \text{BASE} \cdot (\text{ANMAT})$$

$\text{BASE} \cdot (\text{ANMAT})$  is multiplied using Knapp's method [3] with the provision that diagonals with all zeroes are skipped. In doing so, one row of  $(\text{BASE}') =$

BASE · (ANMAT) is computed using only three multiplications and additions. Thus, to multiply BASE · (ANMAT), 3N multiplications are required.

In calculating  $BASE = (ANMAT)^t \cdot (BASE')$  the algorithm which performs this computation takes advantage of the fact that BASE is symmetric; i.e., only  $N/2 + 1$  diagonals of (BASE') participate in the multiplication. The order of simultaneous multiplications then becomes:

Example: Let (BASE') be an 8 x 8 matrix.

1	2	3	4	5			
	1	2	3	4	5		
		1	2	3	4	5	
			1	2	3	4	5
5				1	2	3	4
4	5				1	2	3
3	4	5				1	2
2	3	4	5				1

Equal numbers represent simultaneous computations. Each diagonal again needs only 3 multiplications so that BASE, the final matrix, has all elements computed after  $3((N/2) + 1) + 3N$  multiplications =  $3 \cdot (3(N/2) + 1)$ .

Further explanations are found in Section 2.b.5-7, p. 17. The matrix (BASE') is now of the form

BASE: in PE memory

PE	0	1	2	3	4	5
	$a'_{00}$	0	$a'_{02}$	$a'_{03}$	$a'_{04}$	$a'_{05}$
	0	$a'_{11}$	$a'_{12}$	$a'_{13}$	$a'_{14}$	$a'_{15}$
	$a'_{02}$	$a'_{12}$	$a'_{22}$	0	$a'_{24}$	$a'_{25}$
	$a'_{03}$	$a'_{13}$	0	$a'_{33}$	$a'_{34}$	$a'_{35}$
	$a'_{04}$	$a'_{14}$	$a'_{24}$	$a'_{34}$	$a'_{44}$	0
	$a'_{05}$	$a'_{15}$	$a'_{25}$	$a'_{35}$	0	$a'_{55}$

Now the matrix is rearranged to bring new elements into the  $(2k, 2k+1)$ ,  $(2k+1, 2k)$  positions (see Section 2.b.9, p. 21). Then BASE will be of the form:

(BASE'): in PE memory

PE	0	1	2	3	4	5
	$\begin{pmatrix} a'_{00} & a'_{02} \\ a'_{02} & a'_{22} \end{pmatrix}$	$a'_{03}$	$a'_{04}$	$a'_{05}$	0	
	$a'_{03}$	0	$\begin{pmatrix} a'_{33} & a'_{34} \\ a'_{34} & a'_{44} \end{pmatrix}$	$a'_{35}$	$a'_{13}$	
	$a'_{04}$	$a'_{24}$		0	$a'_{14}$	
	$a'_{05}$	$a'_{25}$	$a'_{35}$	0	$\begin{pmatrix} a'_{55} & a'_{15} \\ a'_{15} & a'_{11} \end{pmatrix}$	
	0	$a'_{12}$	$a'_{13}$	$a'_{14}$		

The matrix is ready for another elimination and transformation process as described above.

## 2. Computation

With no loss in generality, let

BASE: be the matrix from which the eigenvalues are  
being calculated

EIGV: be the eigenvector-matrix

ANMAT: be the angle-matrix

TBASE: a matrix for temporary storage

EPS: an error-matrix

The program is subdivided into two parts:

- the part that calls the subroutine (Appendix A)
- the subroutine itself (Appendix B)

a) The calling program: has to contain

- (1) the "DEFINE CALL" statement (standard form)
- (2) the matrix containing the data from the eigenvalues  
are to be found
- (3) the definition of:
  - i. the eigenvector-matrix: EIGV BLK N;
  - ii. the angle-matrix: ANMAT: BLK N;
  - iii. the temporary storage-matrix: TBASE: BLK N;

Note: The two blocks of storage for ANMAT and TBASE are purposely left outside of the subroutine, so that the space becomes available for the user after leaving the subroutine.

- iv. the error-matrix: EPS: BLK N;

The address of EPS is needed if this routine is used in connection with EBERL, a routine which normalizes a matrix. If EIGEN routine is used alone the statement under (iv) is left out.

- v. the order of the matrix: DEFINE N = n##;

n has to be an even integer.

Thus, the actual call-statement becomes

```
CALL EIGEN(BASE,EIGV,ANMAT,TBASE,0|EPS,N);
```

The word "EIGEN" is the name of the subroutine, and the user is required to use that word.

The terms in parentheses are optional, but not their order; that is, to make the call-statement more general, it must read

```
CALL EIGEN (<original matrix>,  
           <eigenvector-matrix>,  
           <temporary storage matrix>,  
           <temporary storage matrix>,  
           <error-matrix>|0,  
           <order of matrix>);
```

The printout is up to the discretion of the user. If he, however, wants the original matrix BASE printed, the sequence of instructions accomplishing that task must appear before he calls the subroutine, since BASE is changed during the computation and will contain the eigenvalues on the main diagonal after leaving the subroutine.

The subroutine itself contains one print statement originating from the internal procedure "GERSH". The values printed are the radii of the Gershgorin disks, representing the bounds on the eigenvalues.

b) The subroutine EIGEN and the Jacobi algorithm: The entry point of the subroutine is at card image 111000 and is named EIGEN.

EIGEN makes a matrix diagonal and returns the eigenvalues, their corresponding eigenvectors and an upper bound for those eigenvalues.

At the beginning of the program the registers S, R, X and D, the ACARs 0 and 1 are saved as well as a block of 8 local memory registers, namely \$D32-\$D39. The user is advised not to use \$D0-\$D31, since they will be overwritten. For him, \$D32-\$D63 are available, and are restored to their original state before leaving EIGEN.

Upon entering the subroutine, \$C3 will contain the return-address which is saved in .RETUR.

\$C2 contains the address of LIST, which in turn contains the addresses of the parameters. These are stored as follows:

.ADRA	contains address of BASE
.ADRB	contains address of EIGV
.ADRC	contains address of ANMAT
.ADRD	contains address of TBASE
.ADRE	contains address of EPS
.N	contains n as given by DEFINE N = n###;

After executing a sequence of instructions which set up a series of constants, the actual Jacobi algorithm is entered, beginning with a sequence of tests:



1. RWSM: serves to find the row index  $i$  for

$$\max_{i \neq j} \sum_{j=0}^{n-1} |A_{ij}|, \quad i = 0, 1, 2, \dots, n-1$$

and to store the result in .MAX.

2. ANYR: The value for .MAX is passed into ANYR for an any-row, any-column shuffle, with any = .MAX. This procedure rearranges BASE such that the row with the maximal rowsum is shuffled into the place of the first row.

The reason for 1. and 2. is to bring the largest elements of BASE into such a position that they can be eliminated first. Thus, one achieves a faster convergence toward a diagonal matrix BASE.

Note: All calls of a procedure are done through \$C3 by

```
SLIT(3) = <name of procedure>;  
EXCHL(3) $ICR;
```

When entering a procedure the return address in \$C3 is always stored in .SAV1. If the procedure calls another procedure the return address is stored in .SAV3.

3. CONV: The next test is a so-called threshold check. All elements of BASE [2I-1,2I] are compared to a value BD.  $BD = 10^{-k}$  for  $k = 1, 2, 4, 8$ . If the above test is satisfied, i.e.,  $BASE [2I-1,2I] < BD$ , all other computations are skipped (see Flowchart 1: MAINPROGRAM), and BASE is rearranged by a 2nd row, 2nd-column shuffle (for SHUFL see 8. below). Thus, new elements are brought into the (2I-1,2I) positions, and BASE is retested. BD changes only after SHUFL has been executed (n-2) times. As long as  $BD > 10^{-8}$ , BASE is not tested for diagonalization. The purpose of CONV is to eliminate larger elements of BASE first, thus further speeding up the convergence of the algorithm.

4. ANGLE: If the test in CONV is not satisfied, i.e.,  $BASE [2I-1,2I] > BD$ , then the procedure ANGLE is entered, and the rotation-angles are calculated; i.e.,  $\cos \alpha_k$  and  $\sin \alpha_k$  are formed from BASE and

placed as  $2 \times 2$  matrices into ANMAT. The program is straightforward and can be surveyed easily, since many subdivisions and comments contribute to its better readability.

Note: However, the reader should keep in mind that a parallel machine is being used, and that one test can be satisfied for one register in one PE but not necessarily for another PE.

5. MULTPL: Having found ANMAT, the off-diagonal elements of BASE are eliminated by a series of multiplications; i.e.,  $(ANMAT) \times (BASE) \times (ANMAT)^T$  is executed. The same procedure is used to find the eigenvectors; i.e.,  $EIGV = EIGV \times (ANMAT)^T \cdot EIGV = I$  initially. The method used to multiply two matrices stored in straight format is known as Knapp's Method [3]. Its advantage over the log-sum method is that only  $N^2$  multiplications and additions are needed to achieve the result, while in the log-sum method  $N^2$  multiplications, and  $(k+1) \times N^2$  additions are needed, where  $2^k < N \leq 2^{k+1}$ . As  $k$  increases, i.e.,  $N$  increases, the log-sum method becomes more inefficient. Because of the tridiagonal nature of ANMAT further efficiency in multiplying  $BASE \cdot ANMAT$  is achieved by skipping the multiplication when all elements of ANMAT are zero. The elements of each row of  $(BASE')$  are then found, after 3 multiplications, so that  $(BASE') = BASE \cdot ANMAT$  is found after  $3 \cdot N$  multiplications.
6. TRASPOS: finds the transpose of ANMAT by changing the sign of  $\sin \alpha_k$ .
7. SAMUL: This procedure multiplies  $(ANMAT)^t$  by  $(BASE')$ . Taking advantage of the symmetry of the resulting matrix only the first  $(N/2) + 1$  diagonals of  $(BASE')$  are considered. The multiplication process is done by multiplying diagonals with diagonals.

Example:

$(ANMAT)^t$ : in PE memory

PE	0	1	2	3	4	5
	$a_{00}$	$a_{01}$				
	$a_{10}$	$a_{11}$				
			$a_{22}$	$a_{23}$		
			$a_{32}$	$a_{33}$		
					$a_{44}$	$a_{45}$
					$a_{54}$	$a_{55}$

$(BASE')$ : in PE memory

$b_{00}$	$b_{01}$	$b_{02}$	$b_{03}$	$b_{04}$	$b_{05}$
$b_{10}$	$b_{11}$	$b_{12}$	$b_{13}$	$b_{14}$	$b_{15}$
$b_{20}$	$b_{21}$	$b_{22}$	$b_{23}$	$b_{24}$	$b_{25}$
$b_{30}$	$b_{31}$	$b_{32}$	$b_{33}$	$b_{34}$	$b_{35}$
$b_{40}$	$b_{41}$	$b_{42}$	$b_{43}$	$b_{44}$	$b_{45}$
$b_{50}$	$b_{51}$	$b_{52}$	$b_{53}$	$b_{54}$	$b_{55}$

First step: Multiply

$a_{00}b_{00}$      $a_{11}b_{11}$      $a_{22}b_{22}$      $a_{33}b_{33}$      $a_{44}b_{44}$      $a_{55}b_{55}$

Second step: Multiply after routing  $d = 1$  left ( $d = \text{distance}$ )

$$a_{01}b_{10} \quad 0 b_{21} \quad a_{23}b_{32} \quad 0 b_{43} \quad a_{45}b_{54} \quad 0 b_{05}$$

Third step: Multiply after routing  $d = 1$  right

PE:	0	1	2	3	4	5
	$0 b_{50}$	$a_{10}b_{01}$	$0 b_{12}$	$a_{32}b_{23}$	$0 b_{34}$	$a_{54}b_{45}$

Fourth step: Add result from steps 1-3.

$a_{00}b_{00}^+$	$a_{11}b_{11}^+$	$a_{22}b_{22}^+$	$a_{33}b_{33}^+$	$a_{44}b_{44}^+$	$a_{55}b_{55}^+$
$a_{01}b_{10}$	$a_{10}b_{01}$	$a_{23}b_{32}$	$a_{32}b_{23}$	$a_{45}b_{54}$	$a_{54}b_{45}$

and store in the main diagonal of BASE.

$$\text{BASE} = (\text{ANMAT})^t \cdot (\text{BASE}')$$

PE:	0	1	2	3	4	5
-----	---	---	---	---	---	---

$$a_{00}b_{00}^+ + a_{01}b_{10}$$

$$a_{11}b_{11}^+ + a_{10}b_{01}$$

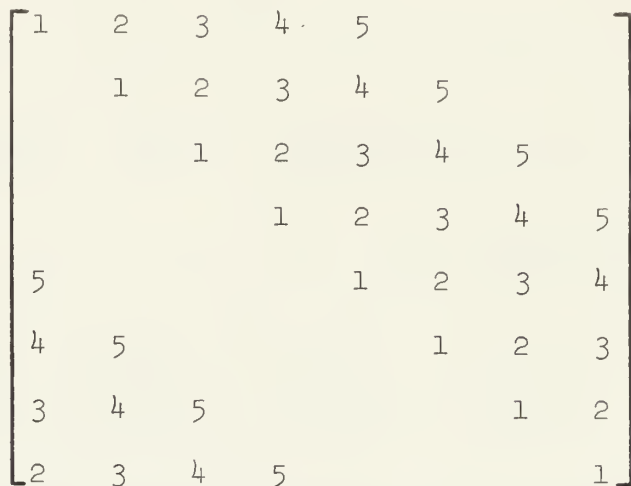
$$a_{22}b_{22}^+ + a_{23}b_{32}$$

$$a_{33}b_{33}^+ + a_{32}b_{23}$$

$$a_{44}b_{44}^+ + a_{45}b_{54}$$

$$a_{55}b_{55}^+ + a_{54}b_{45}$$

After shifting the matrix  $(ANMAT)^t$   $d = 1$  to the right (this is not actually done but think of it that way for better understanding), repeat steps 1-4 starting with element  $b_{01}$  down the diagonal in step 1,  $b_{00}$  down the diagonal in step 2 and  $b_{02}$  down the diagonal in step 3. Repeating steps 1-4  $((N/2) + 1)$  times, all elements of the symmetric matrix BASE are known and the following pattern of execution has developed:



where equal numbers represent simultaneous operations. The diagonal numbered  $5 = ((N/2) + 1)$  forms an exception. When filling the rest of the matrix only the elements in PE  $(N/2)$  to PE  $(N - 1)$  participate, overwriting the elements numbered  $((N/2) + 1)$  in PE 0 to PE  $((N/2) - 1)$ .

Since each diagonal is formed after 3 multiplications, all elements are found after  $3 \cdot ((N/2) + 1)$  multiplications. The total transformation

$$BASE = (ANMAT)^t \cdot BASE \cdot ANMAT$$

is executed with  $3((N/2) + 1) + 3N = 3((3N/2) + 1)$  multiplications instead of  $2 \cdot (N * 2)$  multiplications using the conventional ways in multiplying 3 matrices. Looking at a  $64 \times 64$  matrix the transformation is executed after  $3((3N/2) + 1) = 291$  multiplications instead of  $2 \cdot (N * 2) = 8192$  multiplications using conventional ways.

8. SHUFL: To bring new elements of BASE into the  $(2I-1, 2I)$  positions for another elimination, BASE enters the procedure SHUFL in which the 2nd-row and 2nd-column are brought into the place of the last row

and last column, respectively. This process corresponds in theory to the transformation  $\phi \times \text{BASE}_{(r)} \times \phi^t$ .

Now BASE is ready for another transformation; i.e., steps 3-8 are re-executed. After  $(n - 2)$  repetitions of these steps, BD changes from  $10^{-k}$  to  $10^{-2k}$ , and the algorithm is started from 1. After BD has reached the value  $10^{-8}$ , the matrix BASE is tested for diagonalization. The convergence factor is found in

9. ADDIT: Calculating first

$$E = \sum_{i \neq j} a_{ij}^2 \text{ for all } i \text{ and } j,$$

then

$$D = \sum_{i=0}^{n-1} a_{ii}^2,$$

one has S as  $S = E/D$ . S is then checked against  $\text{KSI} = 10^{-8} \times E_0/D_0$ , where  $E_0$  and  $D_0$  are taken from the original matrix. If  $S < \text{KSI}$  is satisfied, the matrix BASE is sufficiently diagonalized, and  $\text{BASE}[I,I]$  are the eigenvalues. If  $S \geq \text{KSI}$ , the algorithm is repeated going back to 3. or 1.

10. GERSH: Having calculated the eigenvalues, GERSH finds their upper bounds [4] by

$$\text{radii} = \sum_{j=0}^{n-1} a_{ij} \quad i = 0, 1, 2, 3, \dots, n - 1, i \neq j$$

according to Gershgorin's theorem for bounds on eigenvalues. The procedure prints out the result.

Final Comment: The routines ROUTE, ROTAL and ROTAR are standard parts of any major program. They adjust routing of registers and rotating of patterns in ACARs to the left and right, respectively. They make it possible to handle matrices of sizes  $N \leq 64$  and, therefore, make every program more general in nature.

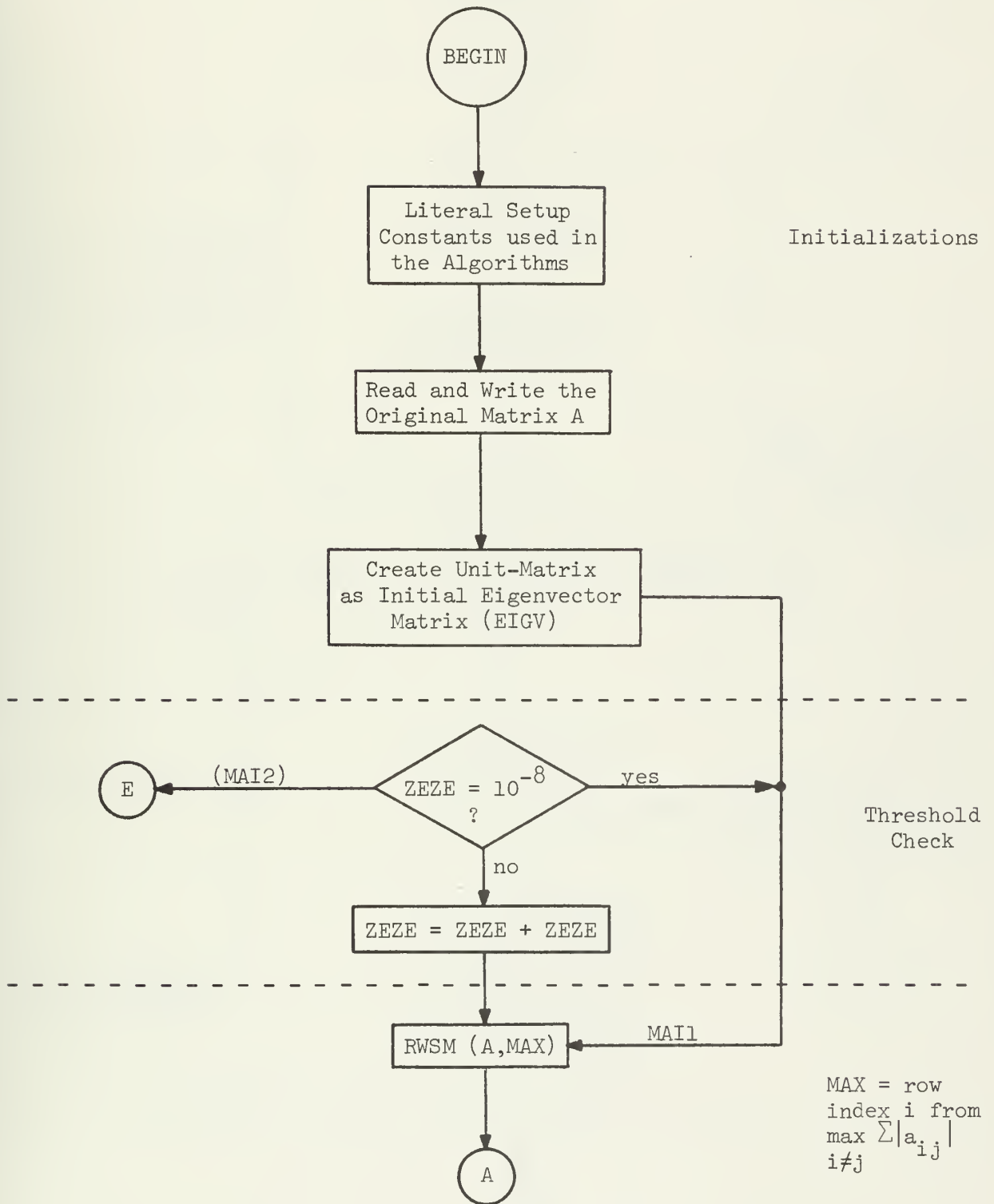
The flow-charts are purposely made lengthy to aid the reader and help him gain a better understanding of the program.

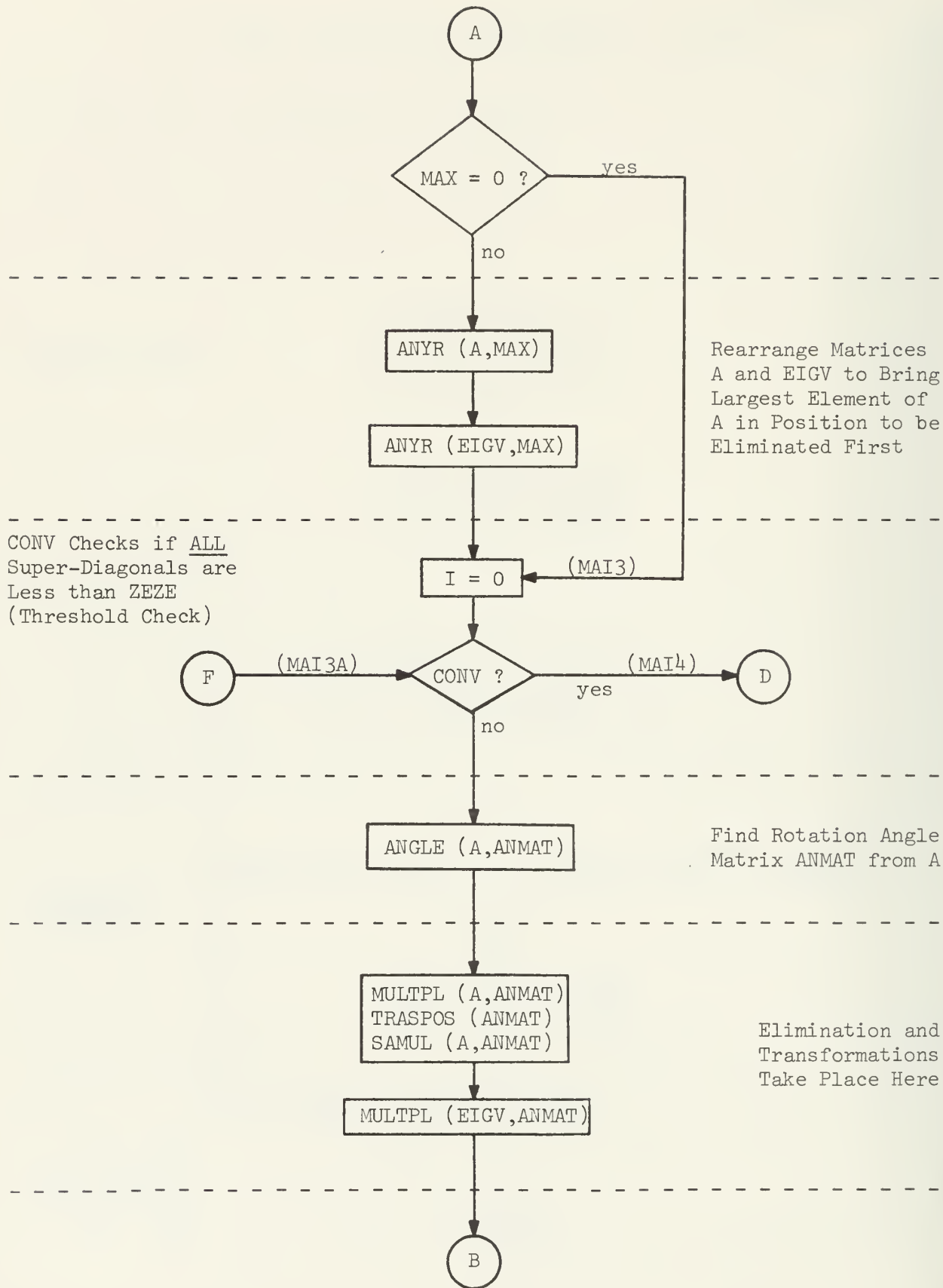
Comments within the program are made wherever the author deemed it necessary. For the most part these comments pertain to groups of instructions. Since the assembly language ASK is highly mnemonic, comments in abundance would hamper rather than facilitate readability.

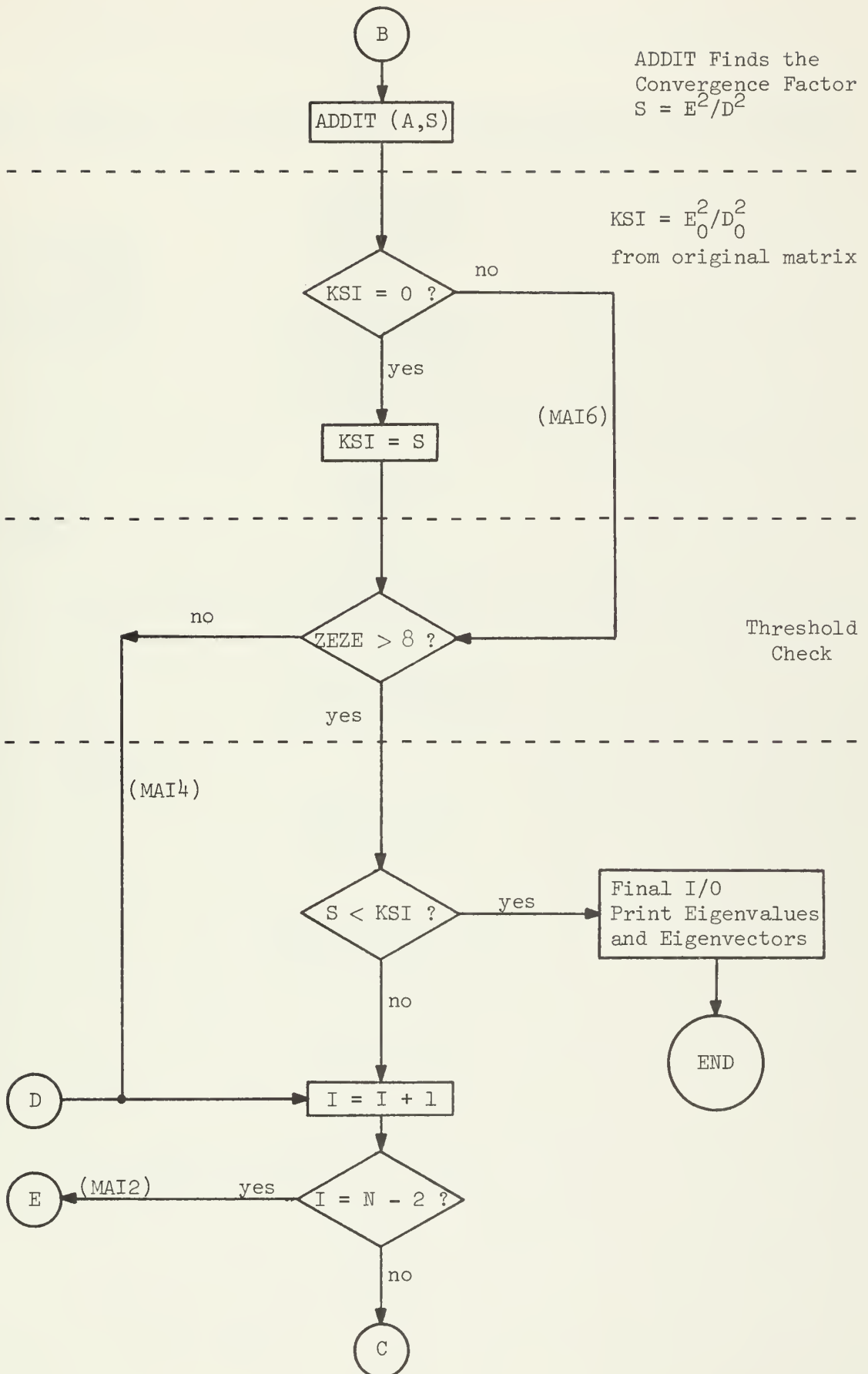


3. Flowcharts

Flowchart 1. MAINPROGRAM: of Subroutine EIGEN







C

SHUFFLE (A)

SHUFFLE (EIGV)

(MAI3A)

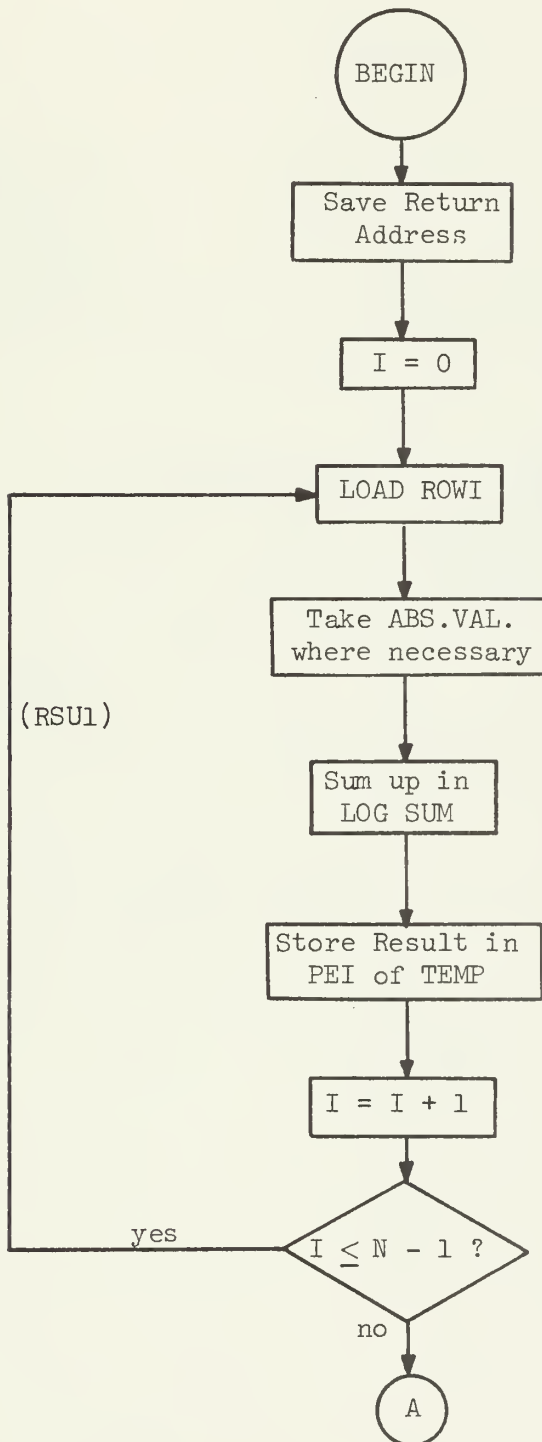
F

2nd Row-2nd Column  
Shuffle to Bring New  
Elements into the  
 $A[2I-1, 2I]$  Position  
for Elimination

Go Back and  
Start a New  
Transformation

Flowchart 2. Flowcharts of the Individual Routines

- a) RWSM: Summing up the Individual Rows of A and Find the INDE of the Row with Maximal Sum ( $A[I,I]$  does not participate).



A

Find Maximal  
Value of TEMP

Compare Register \$A  
Containing the  
Individual Sum against  
Register \$S and  
Overwrite \$A where  $\$A < \$S$

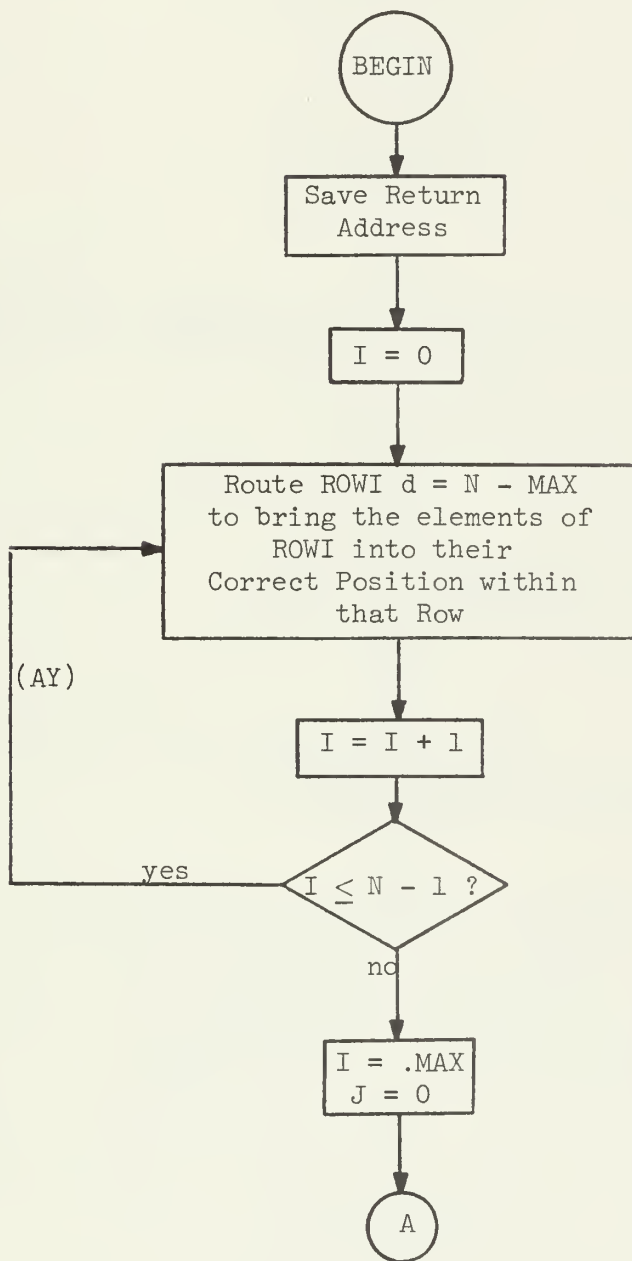
This comparison is done  
in the LOOP "RSM". \$S  
is shuffled in powers  
of two. Six compari-  
sons will fill \$S with  
the maximal value.

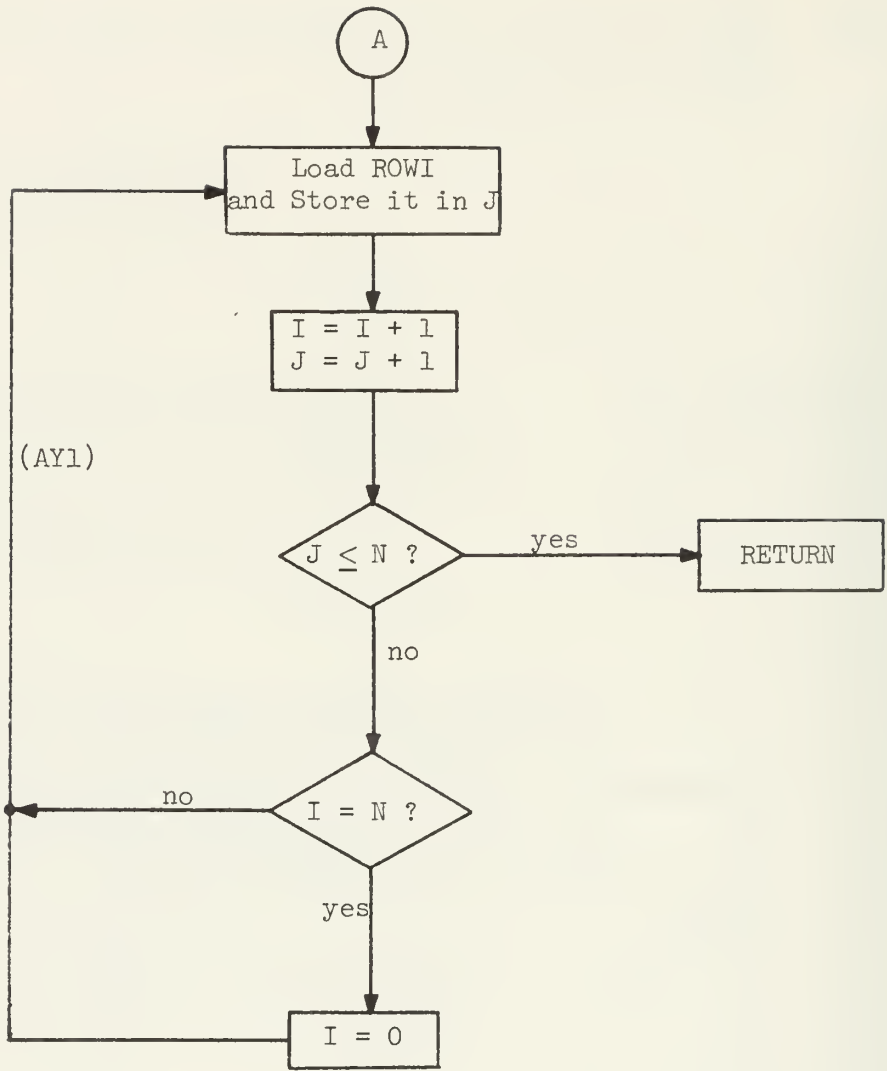
Compare the Largest  
Value in \$S against  
TEMP Containing Individual  
Sum to Find Row Index I

Store Index I  
in .MAX

Return to  
MAINPROGRAM

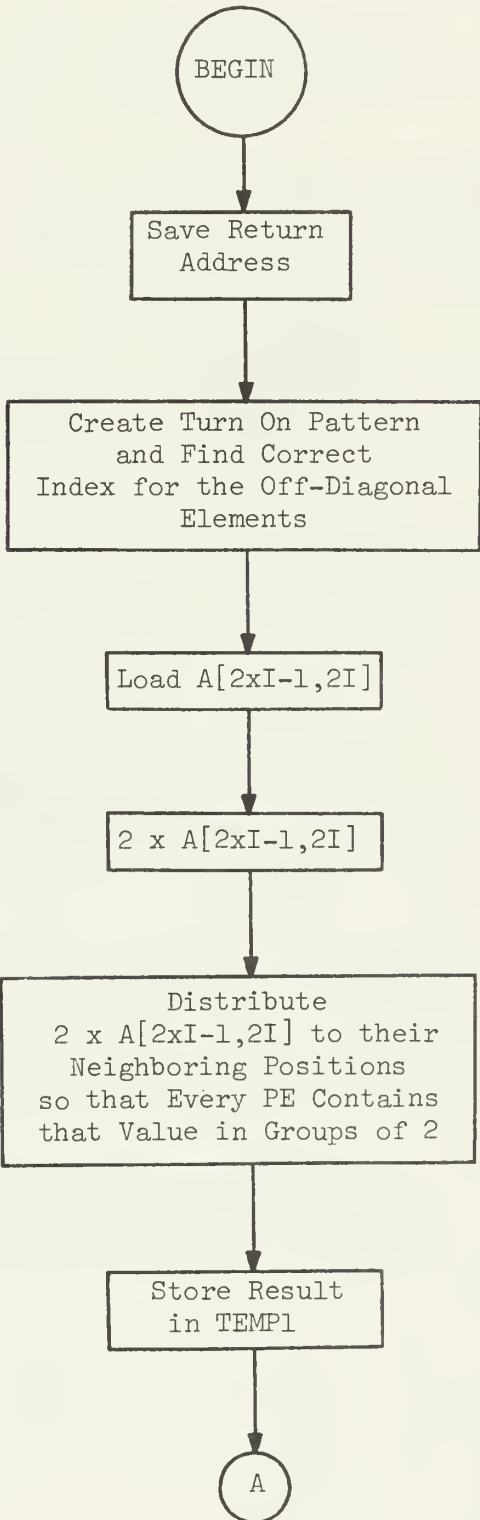
- b) ANYR: Stands for ANY-row, ANY-column Shuffle, where ANY = .MAX. It brings the Row with the Largest Elements into a Position, where they can be Eliminated First.

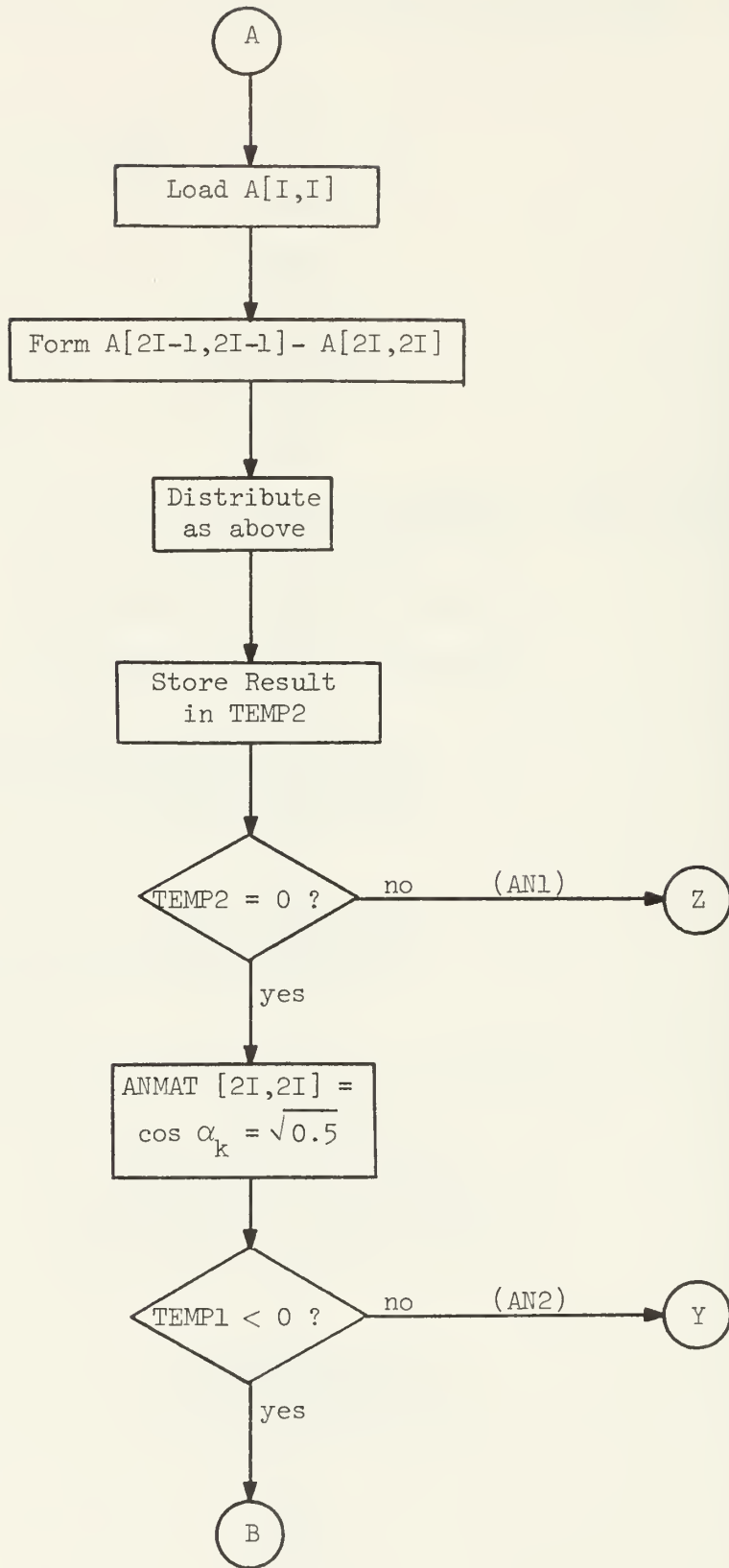


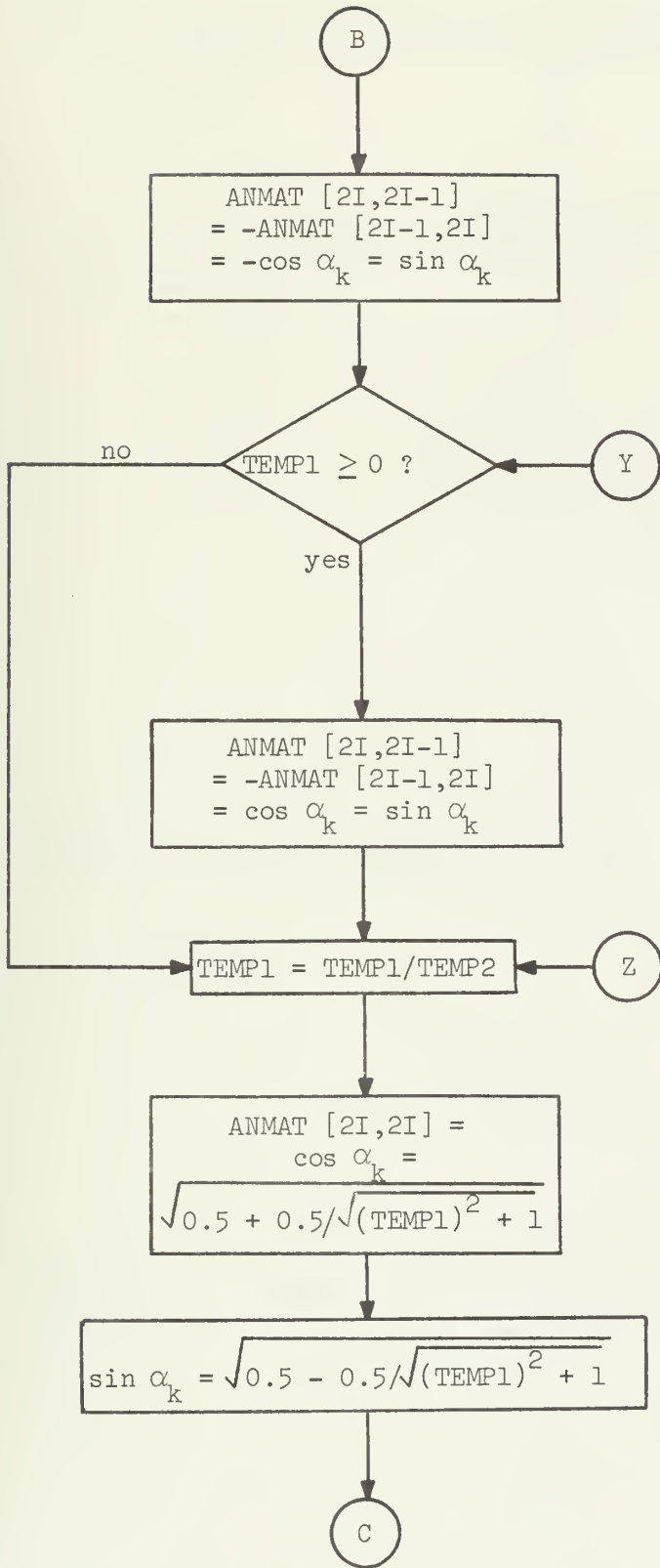




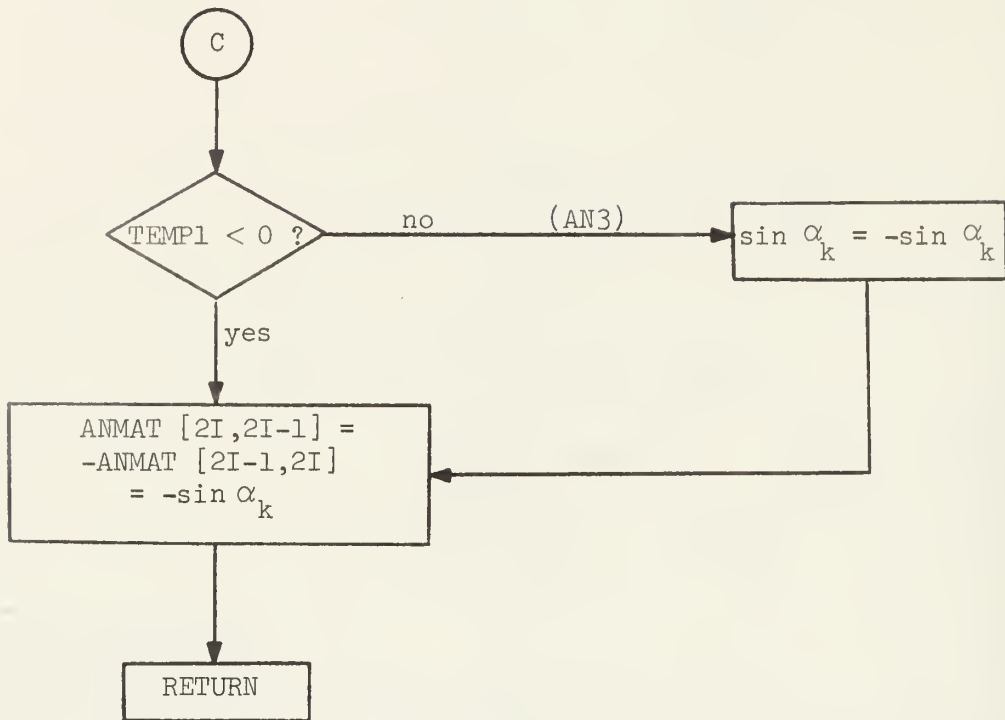
c) ANGLE: ANGLE Calculates the Transformation-Angles and Creates the Transformation Matrix.



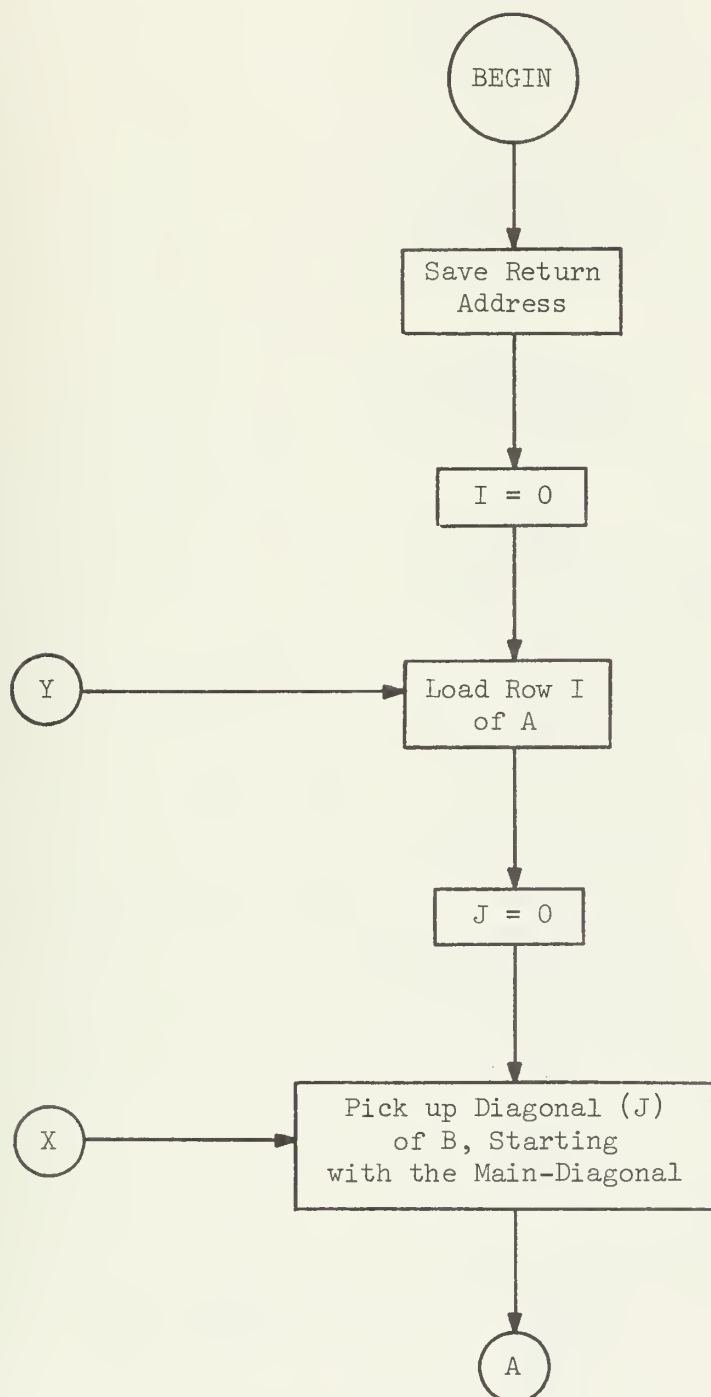


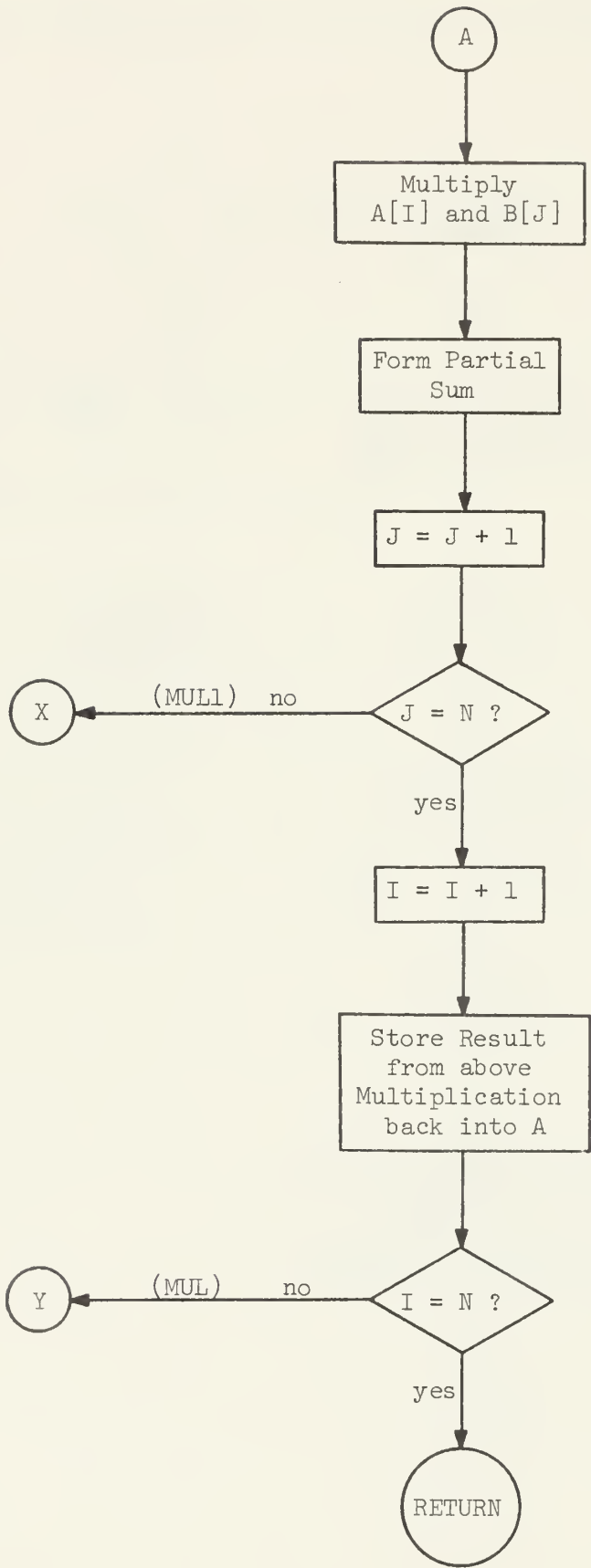


We have to realize that these tests are based on the parallel structure of ILLIAC IV, i.e. the elements ANMAT [2I, 2I-1] are never the same for the different steps and that all branches are satisfied for the set of PEs under consideration.

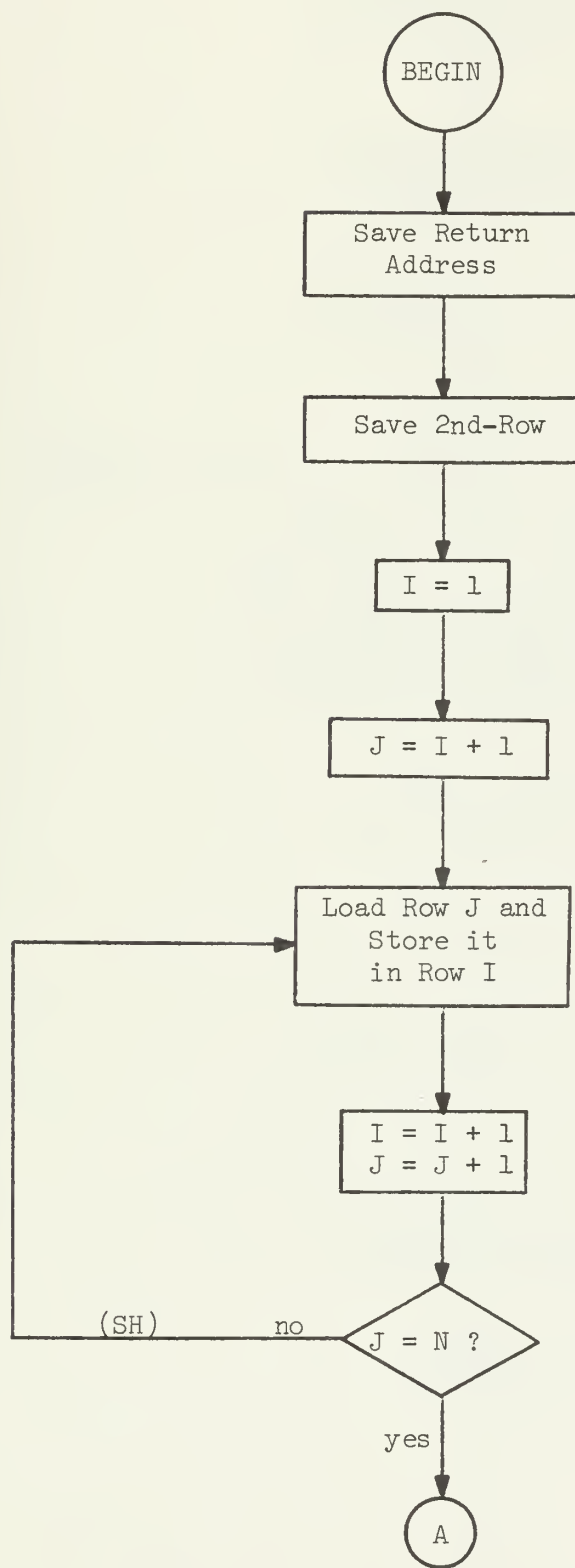


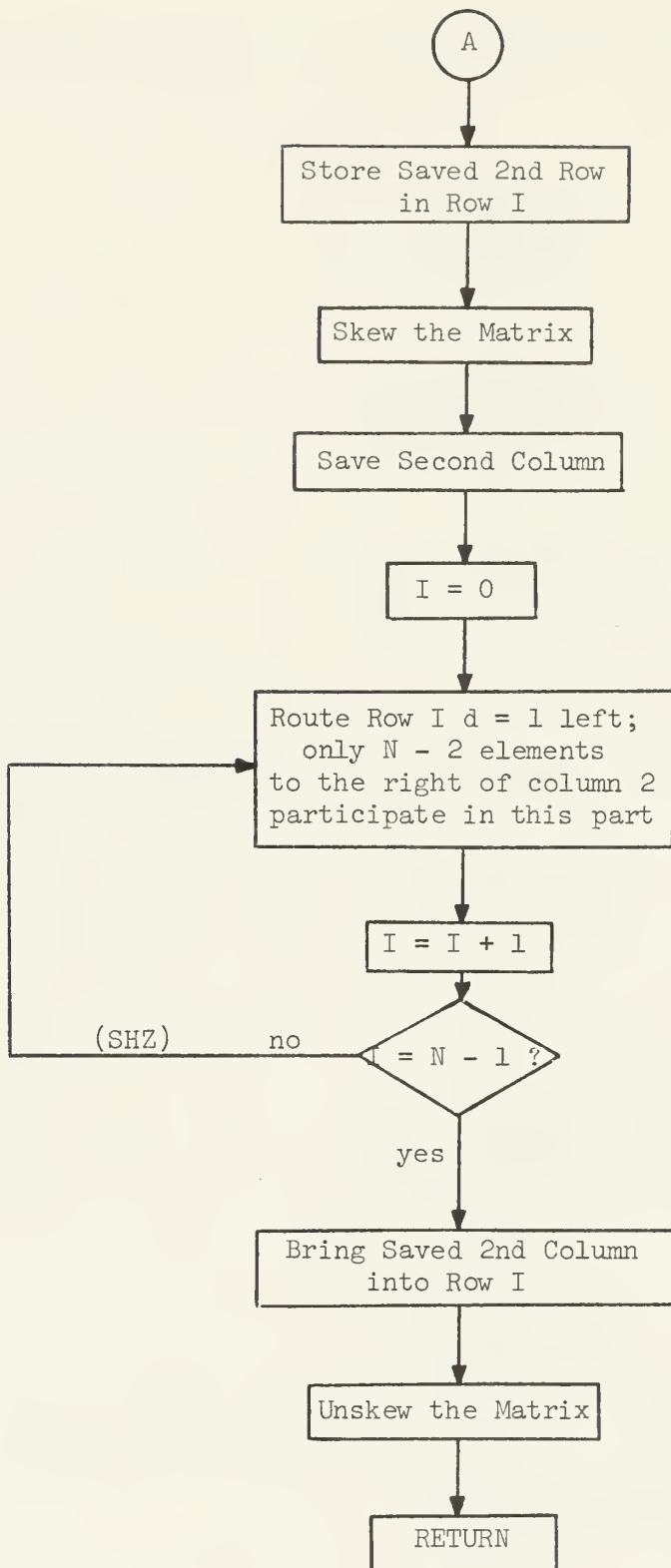
d) MULTPL: Multiplication-Routine A x B for Two Matrices. Addressing is done with reference to the location of BASE. Knapp's method is applied.





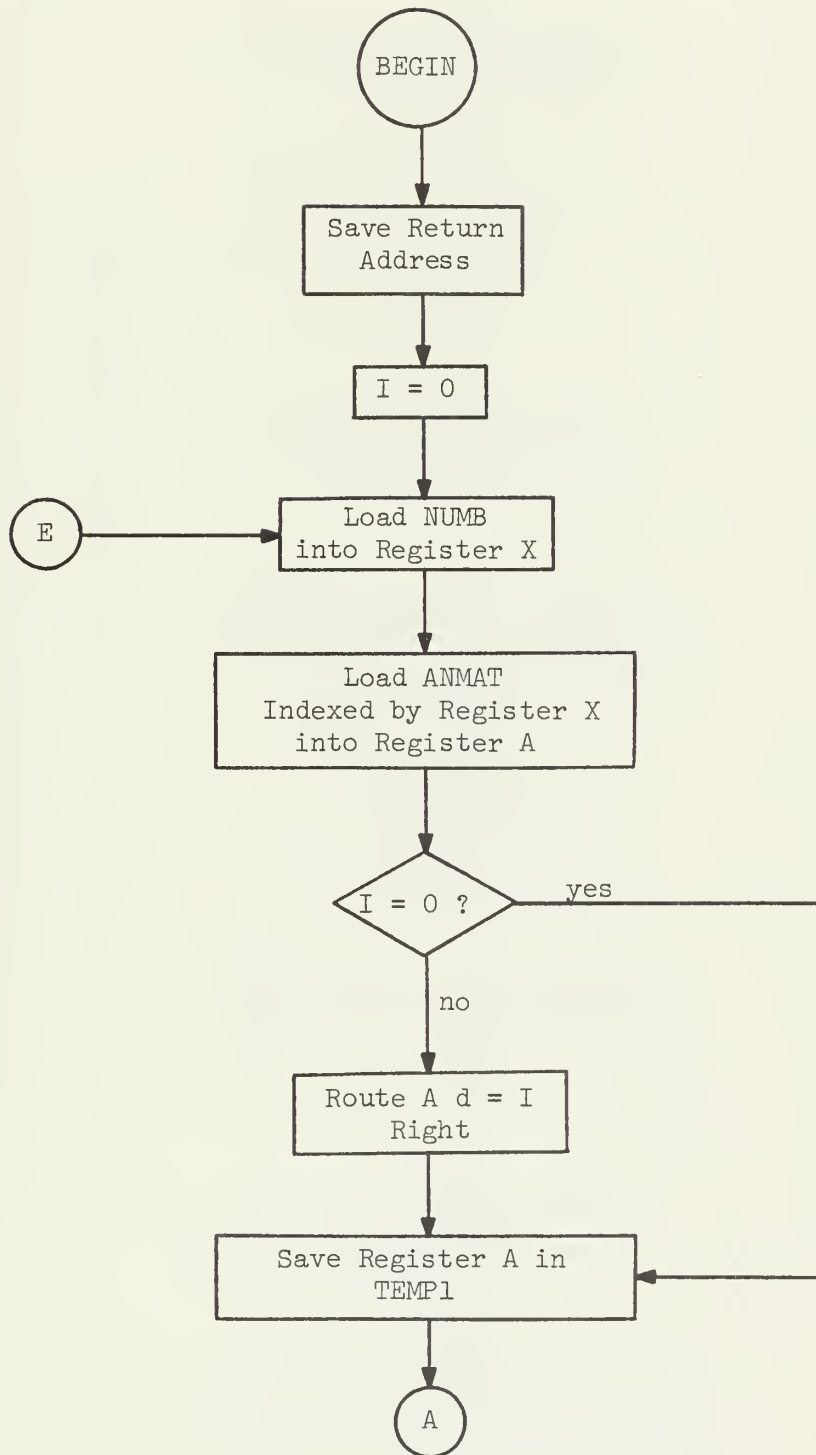
e) SHUFL: This Part brings the 2nd-Row to the Bottom of the Matrix and the 2nd-Column to the Far Right and Adjusts the Rest of the Matrix

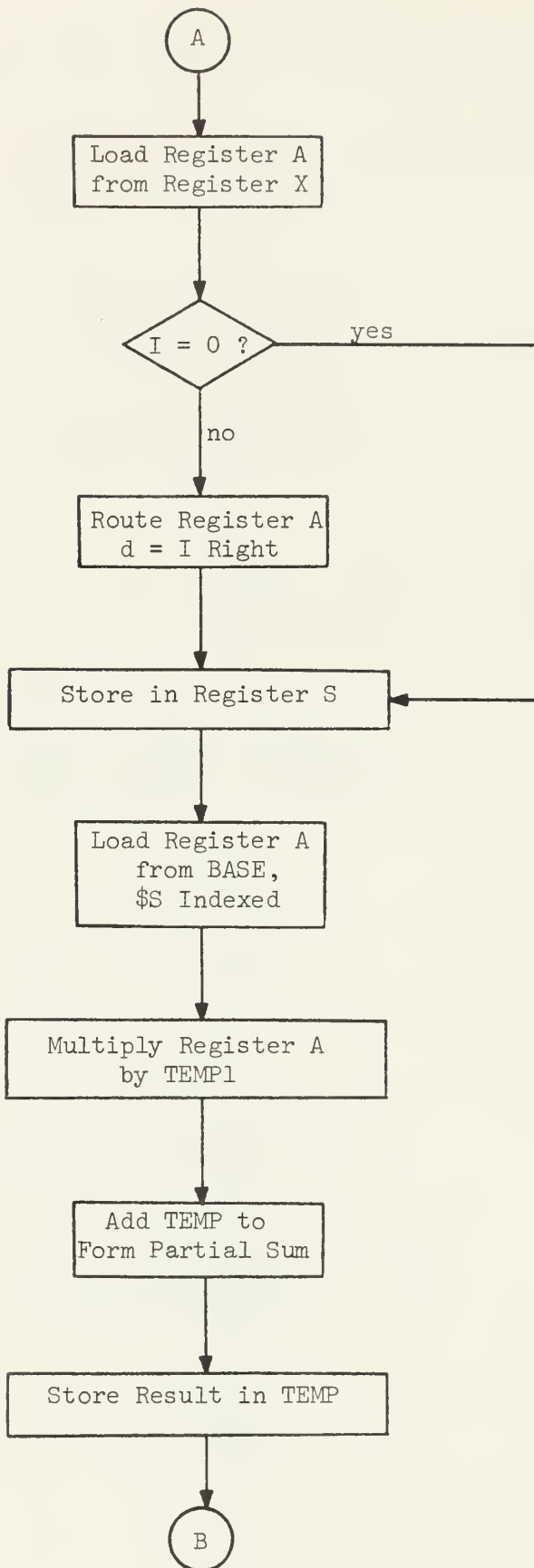


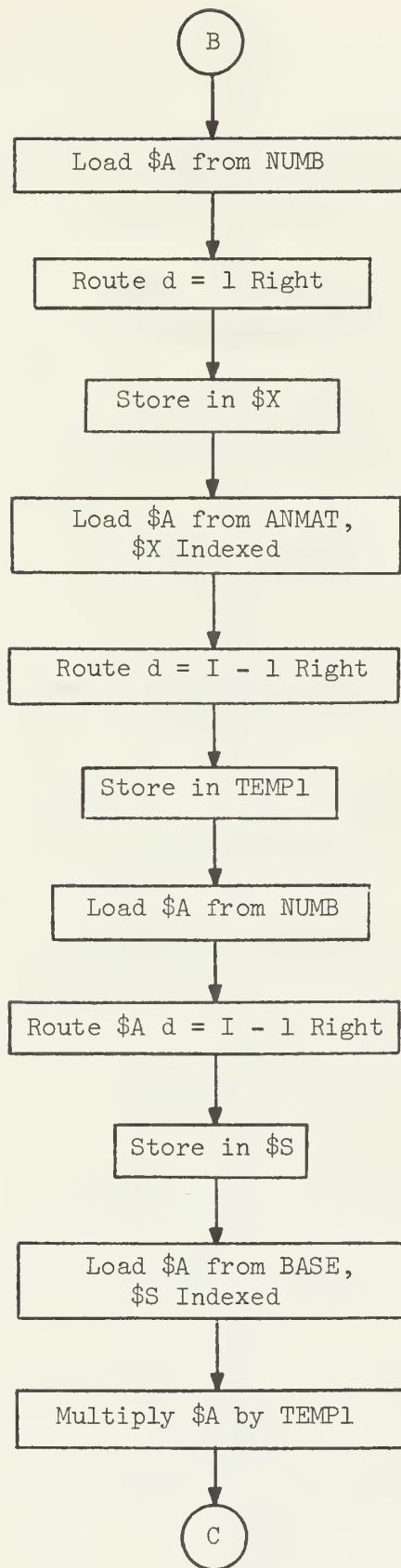


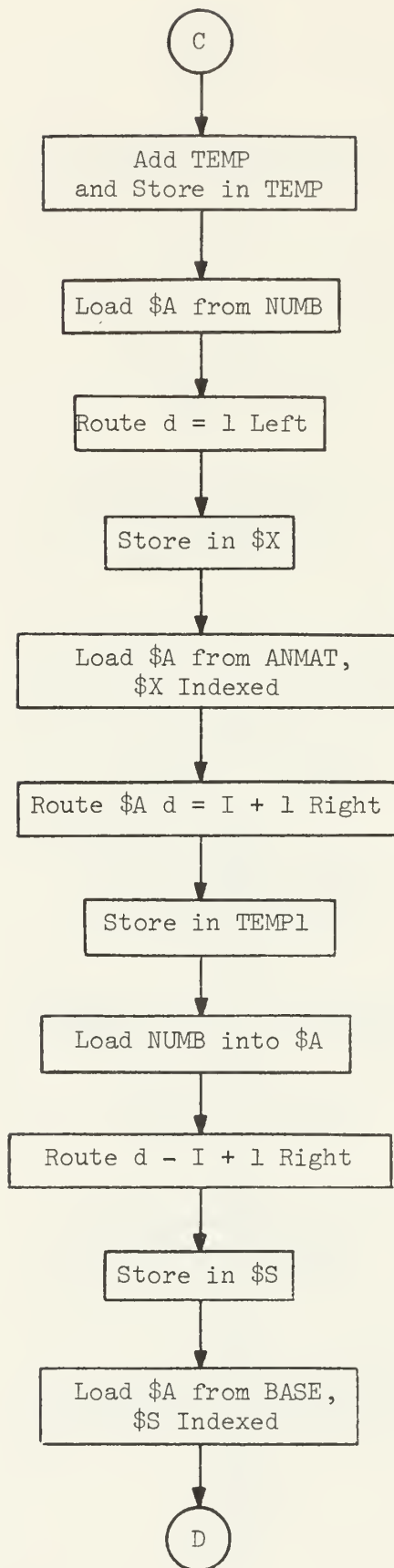


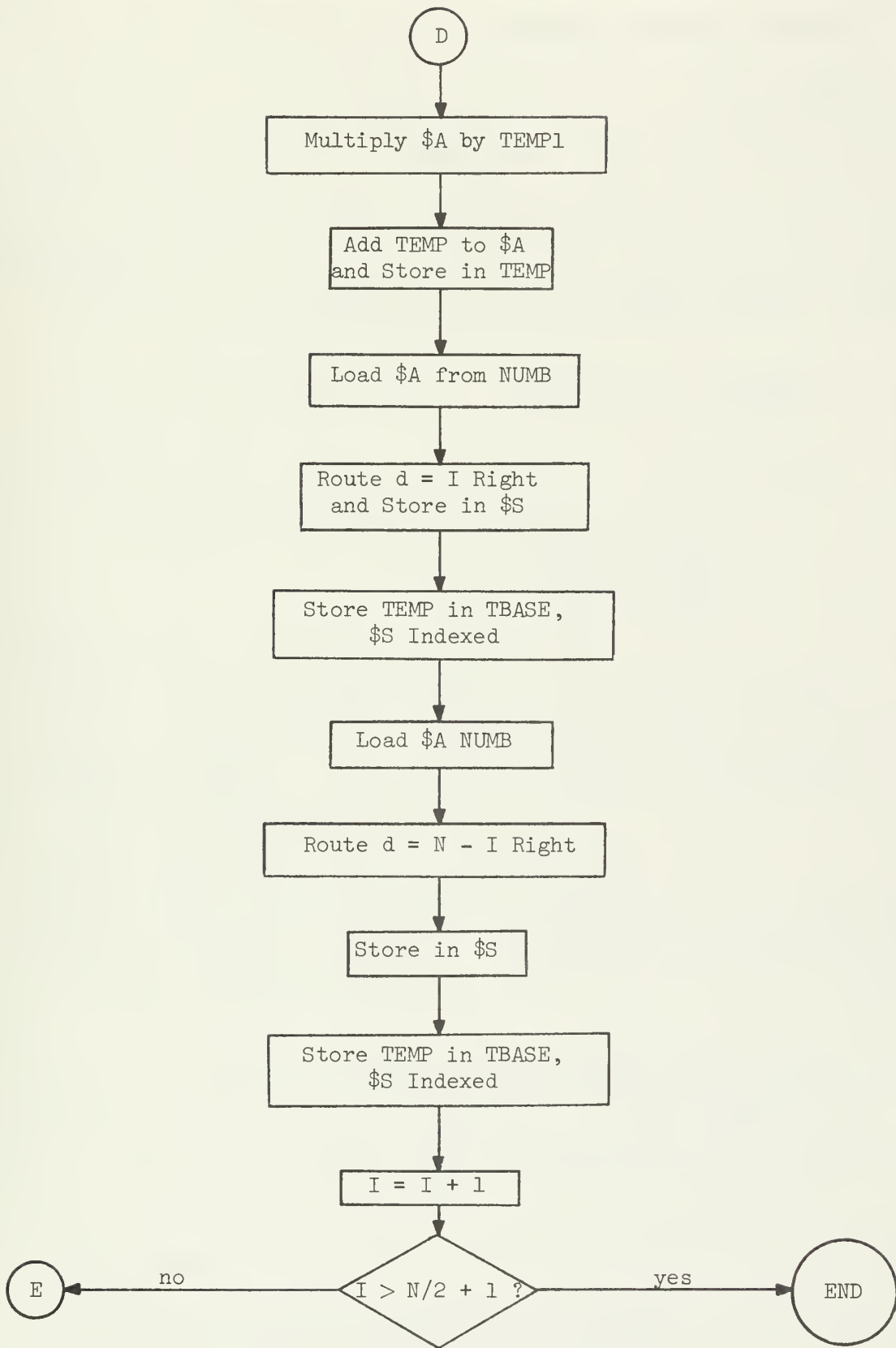
) SAMUL: Multiplying Two Matrices  $BASE = (ANMAT)^t \cdot (BASE)^1$











## B. Eberlein: Normalizing a Matrix

### 1. Storage Scheme

The storage scheme is the same as described under Jacobi (see Section II.A.1). The transformations here are somewhat modified since we are dealing with non-symmetric matrices. The transformation  $(\text{BASE}') = \text{BASE} \cdot \text{ANMAT}$  follows the same pattern as under Jacobi, however the transformation  $(\text{ANMAT})^t \cdot (\text{BASE}') = \text{BASE}$  is done in the following way:

$(\text{ANMAT})^t$ : in PE memory

PE	0	1	2	3	4	5
	$\cosh\psi_k$	$\sinh\psi_k$				
	$\sinh\psi_k$	$\cosh\psi_k$				
			$\cosh\psi_k$	$\sinh\psi_k$		
			$\sinh\psi_k$	$\cosh\psi_k$		
					$\cosh\psi_k$	$\sinh\psi_k$
					$\sinh\psi_k$	$\cosh\psi_k$

We notice that all  $2 \times 2$  submatrices are equal to each other. Bringing  $\cosh\psi_k$  and  $\sinh\psi_k$  up into CU memory and then broadcasting, the multiplication becomes:

Step 1: Multiply simultaneously  $\cosh\psi_k$  by row I,  $I = 0, 2, 4, \dots, N - 2$  of  $(\text{BASE}')$

Step 2: Multiply simultaneously  $\sinh\psi_k$  by row  $I + 1$ ,  $I = 0, 2, 4, \dots, N - 2$  of  $(\text{BASE}')$

Step 3: Add results from steps 1-2 and store in row I of BASE

Then switch  $\cosh\psi_k$  and  $\sinh\psi_k$  and repeat steps 1-3, but storing the result in row  $I + 1$ , now, of BASE. Each row, thus, is found after 2 multiplications,

achieving a  $2 \cdot N$  multiplication for the transformation  $BASE = (ANMAT)^t \cdot (BASE')$ . The total amount of multiplications for  $BASE = (ANMAT)^t \cdot BASE \cdot ANMAT$  is  $2N + 3N = 5N$ . For a  $64 \times 64$  matrix this means 320 instead of the 3192 multiplications necessary for conventionally multiplying three matrices by each other.

## 2. Computation

The program is subdivided into two separate parts:

- a. the part that calls the subroutine
- b. the subroutine itself

The "call" statement reads

```
CALL EBERL(BASE,CMAT,ANMAT,TBASE,EBEIG,EPS,N);
```

where

EBERL is the name of the subroutine

BASE: the original matrix, which will be returned normalized

CMAT: a temporary matrix used for storing  $A \cdot A^T - A^T \cdot A$  ( $A = BASE$ )

ANMAT: a temporary matrix used for storing the transformation matrices  $P_e$  and  $Q_e$ .

TBASE: a temporary matrix used for scratch

EBEIG: the matrix which returns the product

$$\prod_{e=1}^m Q_e^{-1} P_e^{-1} M_e,$$

needed, if EBERL is used in connection with Jacobi, to produce correct eigenvectors

EPS: the matrix which returns the error made when making BASE symmetric after BASE is normalized

N: the order of the matrix

ANMAT, TBASE and CMAT are available to the user after leaving the subroutine.

The Subroutine EBERL and the Eberlein Algorithms. The entry point of the subroutine is at card image 155300 and is named EBERL.

EBERL normalizes a matrix and returns a symmetric matrix if all  $a_{pq} = a_{qp}$  or an asymmetric matrix if some/all  $a_{pq} = -a_{qp}$  and  $a_{pp} = a_{qq}$ . It also returns the product of all transformation matrices and the error-matrix found when making the practically normal matrix symmetric.

At the beginning of the program the registers S, R, X and D, and the ACARs 0 and 1 are saved as well as two blocks of 8 local memory registers, namely \$D32-\$D47. The user is advised not to use \$D0-\$D31, since they will be overwritten, unless he saves the content of those registers himself.

Upon entering the subroutine, \$C3 will contain the return-address which is saved in .RETUR.

\$C2 contains the address of LIST, which in turn contains the of the parameters. These are stored as follows:

- .ADRA contains address of BASE
- .ADRCM contains address of CMAT
- .ADRC contains address of ANMAT
- .ADRD contains address of TBASE
- .ADRE contains address of EBEIG
- .ADRF contains address of EPS
- .N contains n, the order of the matrix.

After these initializations and after setting up the constants pertaining to the algorithms, the following procedures are executed in this order:

1. MLTRPS: This procedure accomplishes the multiplication of  $A \cdot A^t$  ( $A = \text{BASE}$ ) without actually transposing the matrix A.
2. TRPS: creates  $A^t$ , to "ready" A for the multiplication of  $A^t \cdot A$ . 1. and 2. together enable us to calculate  $\text{CMAT} = A \cdot A^t - A^t \cdot A$ . Having calculated CMAT we next find the largest off-diagonal element in
3. FNDMX: which returns this element to the local memory with address .G. The location of this element within CMAT is also returned in .MAXR = rowindex and MAXC = columnindex.
4. AVERG: then tests if the largest off-diagonal is within the range of the averaged off-diagonal elements. If so we enter



5. NULCHK: which determines whether or not any of the  $c_{2k-1,2k-1} - c_{2k,2k}$  are zero. If this test is satisfied,
6. SHFT: is executed. This procedure exchanges the second row for row I, where I = row index for which  $\text{sign } c_{00} = \text{sign } (-c_{II})$ .

Procedures 5. and 6. cause the difference,  $c_{2k-1,2k-1} - c_{2k,2k}$ , to be unequal to zero and, thus:

$$h = 4 c_{2k-1,2k}^2 + (c_{2k-1,2k-1} - c_{2k,2k})^2^{1/2}$$

becomes a maximum which in turn causes  $c_{2k-1,2k} = (1/2) h$  to be maximal.

Tests have shown that including these procedures avoids oscillation of the elements of CMAT when BASE is close to normal but not yet normal enough to consider the results final. If the checks under 4. or 5. are not satisfied, enter:

7. SHUF: which rearranges CMAT according to MAXR and MAXC found in 3.; i.e., it brings the largest off-diagonal element into the (2k-1,2k) position. This rearrangement corresponds to the transformation  $M^t \cdot A \cdot M$ .
8. ANGL: This procedure represents the algorithm used to find the transfer matrix ANMAT, a tridiagonal matrix consisting of 2 x 2 submatrices of the form  $P_e$  (described in the discussion of the mathematical background of the Eberlein method). The actual transformation of  $\text{BASE} = (\text{ANMAT})^t \cdot \text{BASE} \cdot \text{ANMAT}$  takes place in the following sequence of procedures:
  9. MULTPL:  $(\text{BASE}') = \text{BASE} \cdot \text{ANMAT}$  is found. The scheme is the same as that under Jacobi.
  10. TRAPOS: finds the transpose of NNMAT by changing the sign of the  $\sin k$ .
  11. MUISA:  $\text{BASE} = (\text{ANMAT})^t \cdot (\text{BASE}')$ . (For a detailed description, the reader is referred to the discussion of "storage scheme" in Section II.B.1 and Flowchart 4.b.e.) Having the intermediate form of the matrix BASE, we calculate from it in:

12. HYANG:  $\cosh_k$  and  $\sinh_k$  and again form a tridiagonal matrix using the core space of ANMAT. HYANG also finds the convergence factor  $N^2(A) = \text{FINVAL}$ . A check to determine if BASE can be made diagonal, i.e., whether  $\tanh^4$  is not equal to one, is included. If it is equal to 1, we leave HYANG immediately and determine if BASE has reached its normal form. If so, we leave the subroutine altogether. If not, we return to step one and repeat the sequence of procedures without undergoing a transformation on BASE using ANMAT just found. If  $\tanh^4$  is equal to 1, we form  $\text{BASE} = (\text{ANMAT})^t \cdot \text{BASE} \cdot \text{ANMAT}$  according to the procedures described under steps 9-11.

Now a sequence of tests is performed to check the state of normalization. First  $\text{FINVAL} \leq 10^{-9}$  is tested. If this test is not satisfied, we return to 1. On the other hand, if it is satisfied, we determine whether or not the difference between the previous FINVAL and the present FINVAL is less than  $10^{-12}$ . If not, we again return to step one and repeat all steps. If it is satisfied, the matrix BASE is sufficiently normal and we enter:

13. SATE: a procedure for checking that all  $a_{pq} = a_{qp}$ ; i.e., we test sign  $(a_{qp})$  since an accumulation of errors will never bring the ideal result. If  $\text{sign}(a_{pq}) = \text{sign}(a_{qp})$  for all p and q,

14. SYM: is executed. There we make BASE truly symmetric by calculating  $(a_{ij} + a_{ji})/2$ , which is returned to BASE into both (i,j) and (j,i) locations, after finding the error by computing:

$$(a_{ij})_{\text{old}} - (a_{ij} + a_{ij})/2 \quad \text{and} \quad (a_{ji})_{\text{old}} - (a_{ij} + a_{ji})/2$$

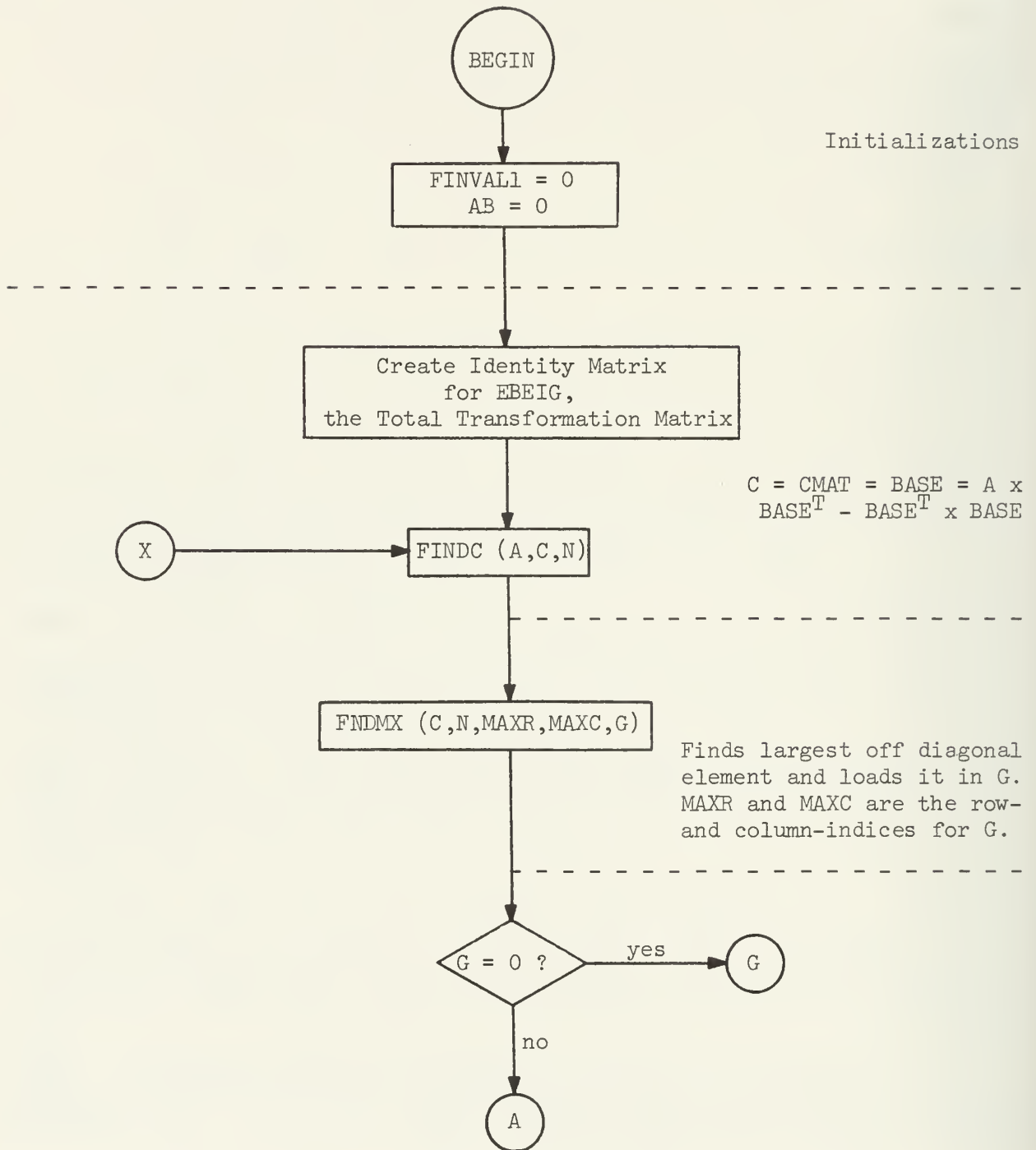
The last results are returned to the error-matrix EPS. If, however,  $\text{sign}(a_{pq}) = -\text{sign}(a_{qp})$ , i.e., if a difference in sign occurs for some  $a_{pq}$ ,

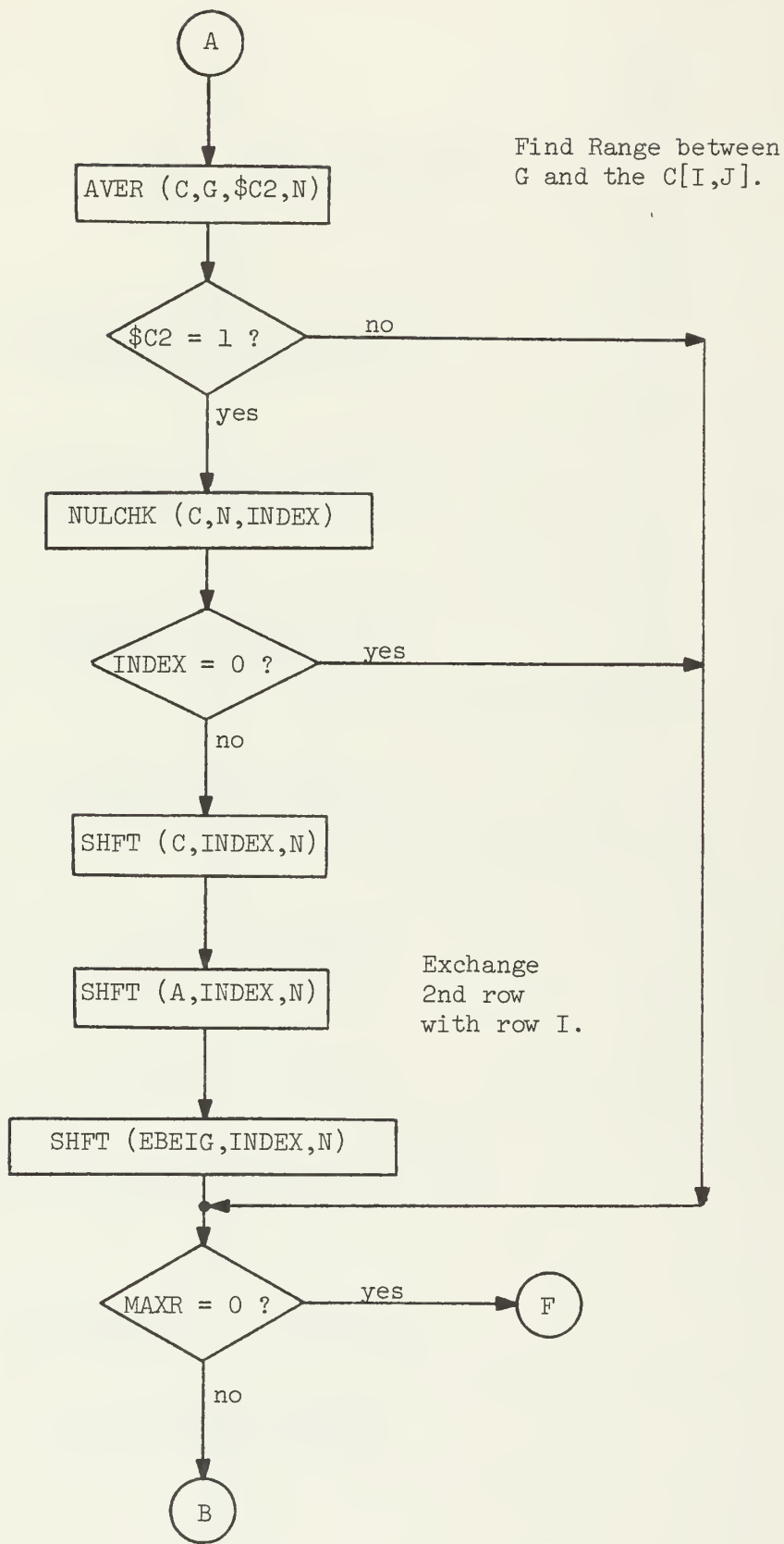
15. ASYM: is entered, which also averages the off-diagonal elements by computing  $(a_{ij} + a_{ji})/2$ . Here, too, an error-matrix is created as under 13.

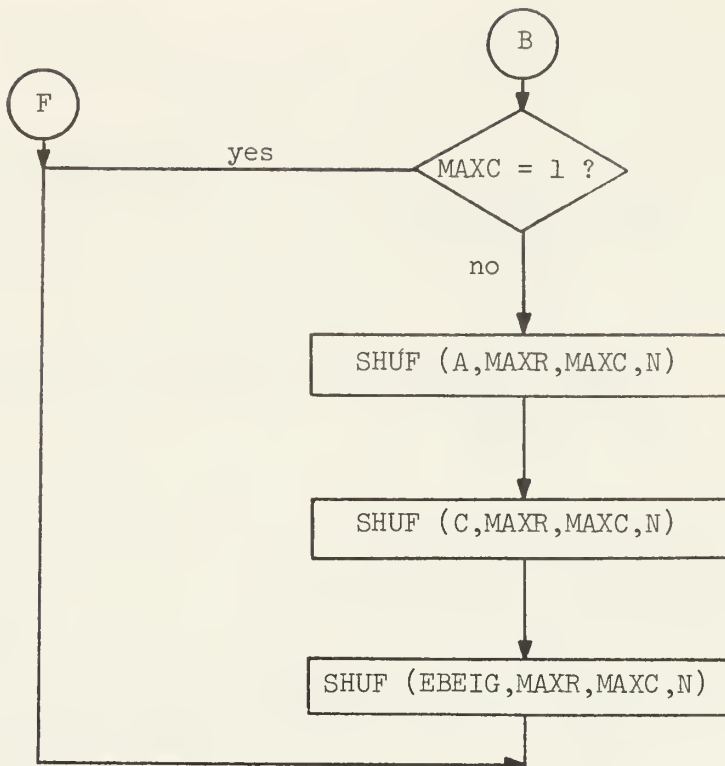
The subroutine is then left with a flag in  $\$C2$ , to signal the symmetric case ( $a_{pq} = a_{qp}$ ) or the asymmetric case ( $a_{pq} = -a_{qp}$ ). From the content of  $\$C2$ ,  $\$C2 = 0$  or  $\$C2 = 1$  respectively, a decision can be made to enter EIGEN, the subroutine for finding eigenvalues and eigenvectors of a real symmetric matrix, or perform the algorithm of finding complex eigenvalues and complex eigenvectors through the subroutine CEIGEN.

3. Flowcharts

Flowchart 3. Eberlein's Algorithm (MAINPROGRAM)







Bring largest element of C into the (2k-1, 2k) portion and rearrange A and EBEIG accordingly.

ANMAT is the transformation matrix



MULTPL (A, ANMAT, N)

TRASPOS (ANMAT, N)

MULSA (A, ANMAT, N)

MULTPL (EBEIG, ANMAT, N)

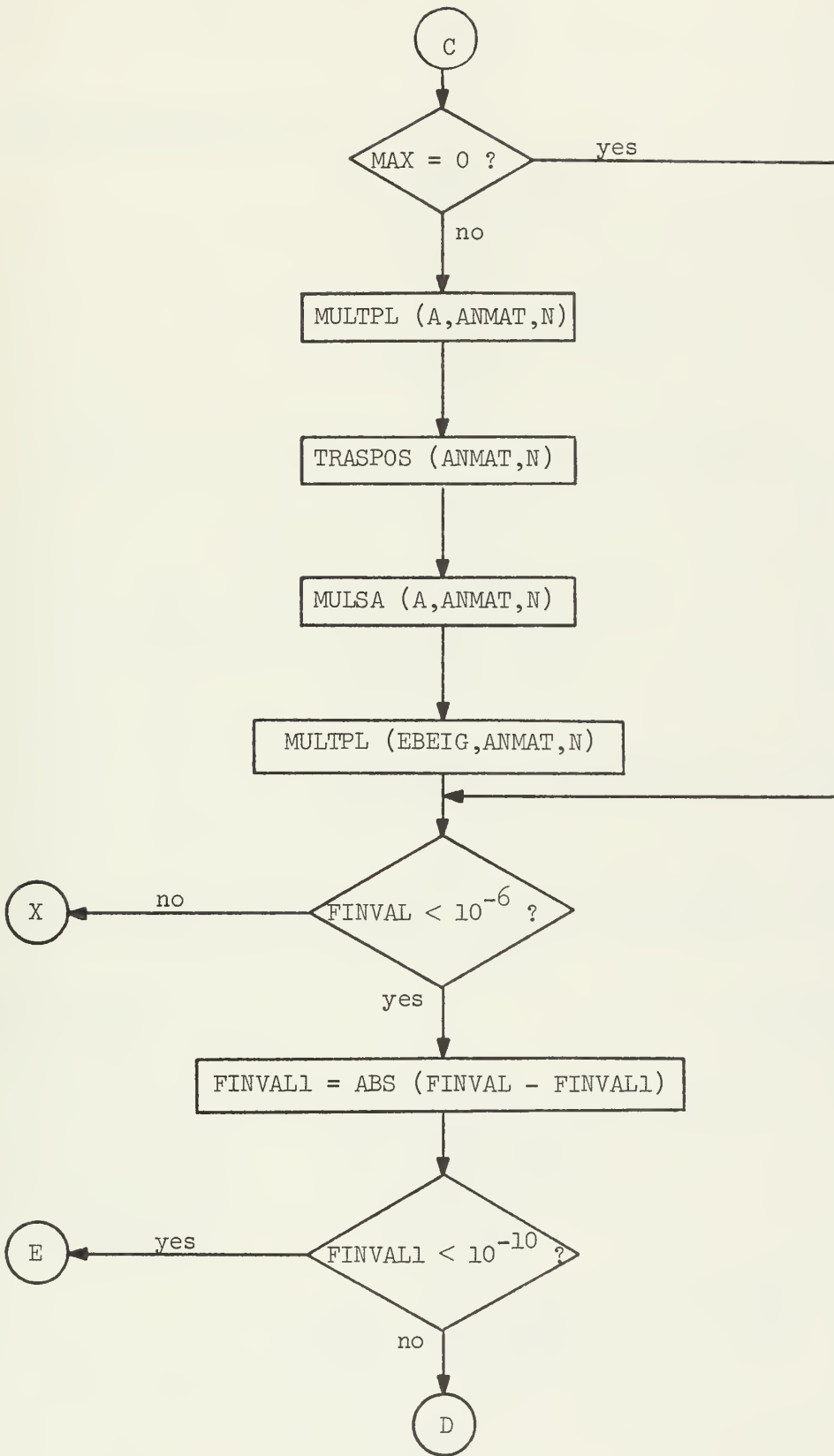
The transformation  $A = ANMAT^T \textcircled{x} A \textcircled{x}$   
 $ANMAT^T$  takes place here.

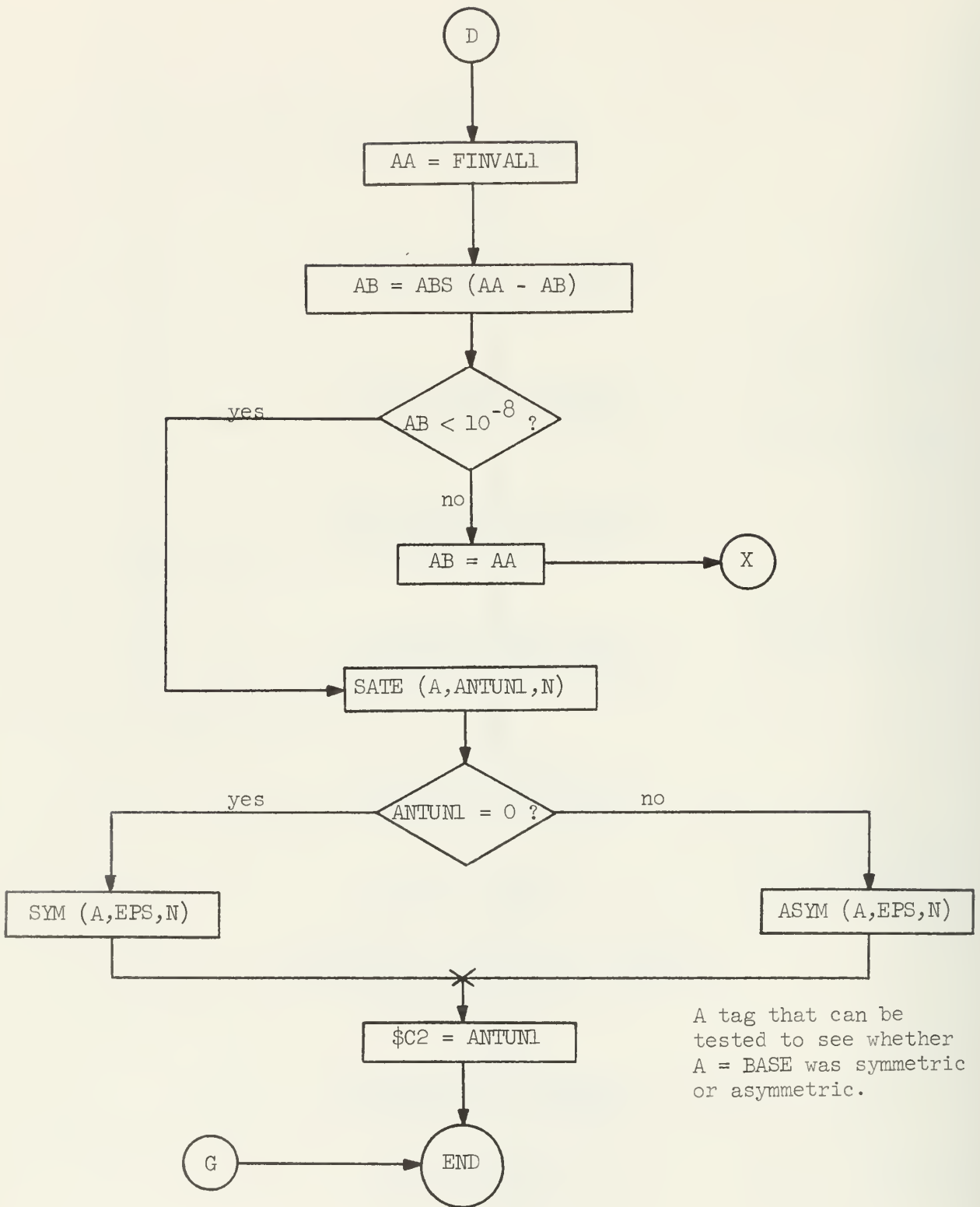


HYANG (ANMAT, A, N)

Second transformation matrix



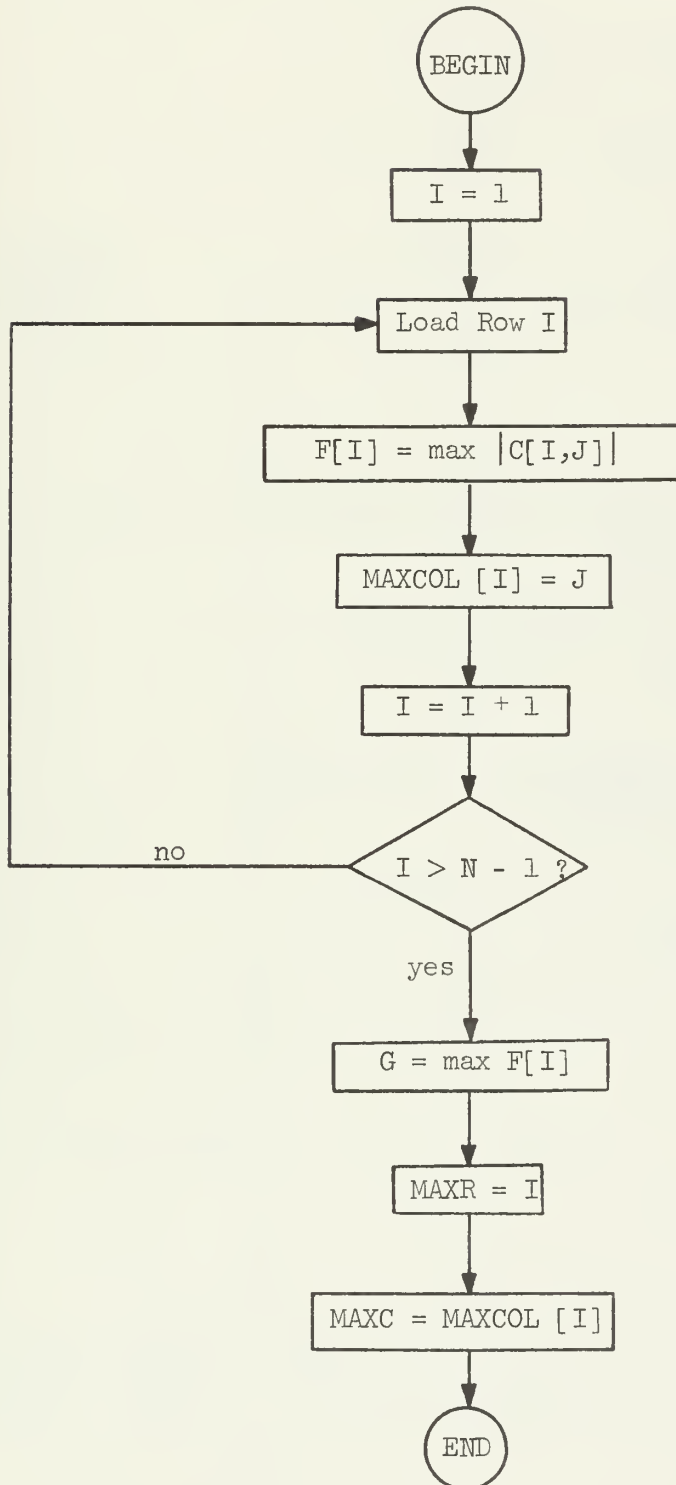






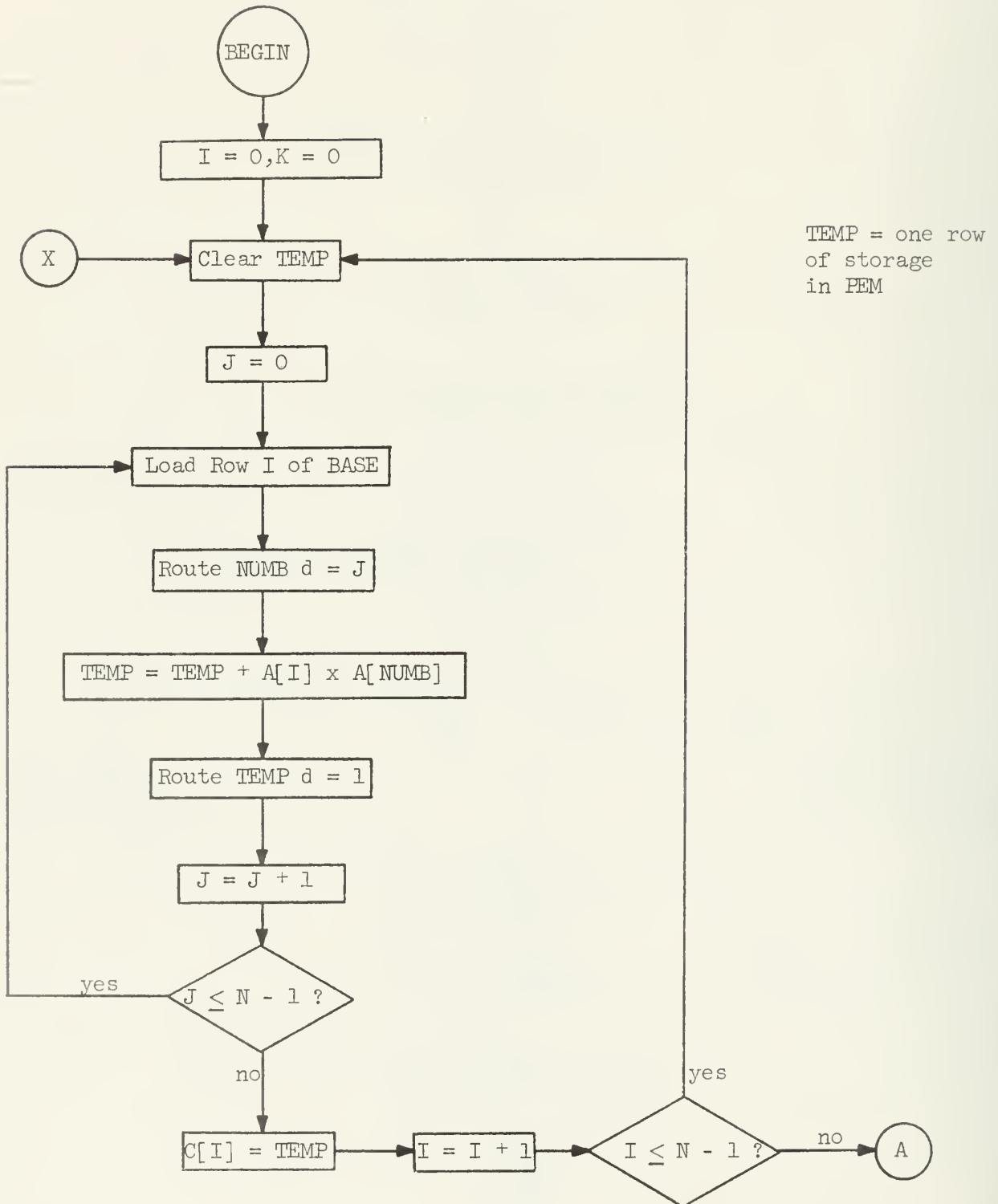
Flowchart 4. Procedures to Eberlein's Method

a) FINMX (C,N,MAXR,MAXC,G)

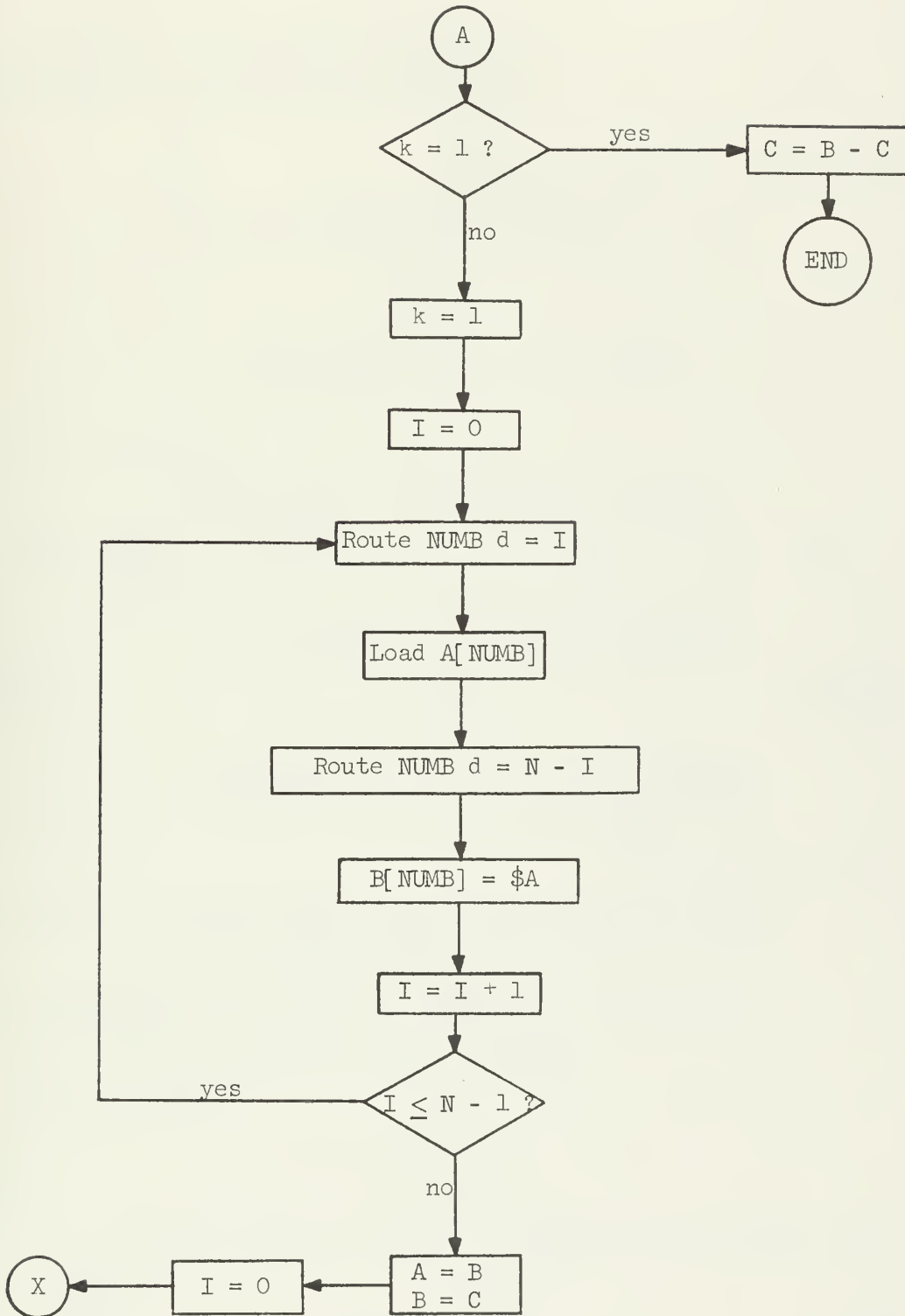


b) FINDC (A,C,N)

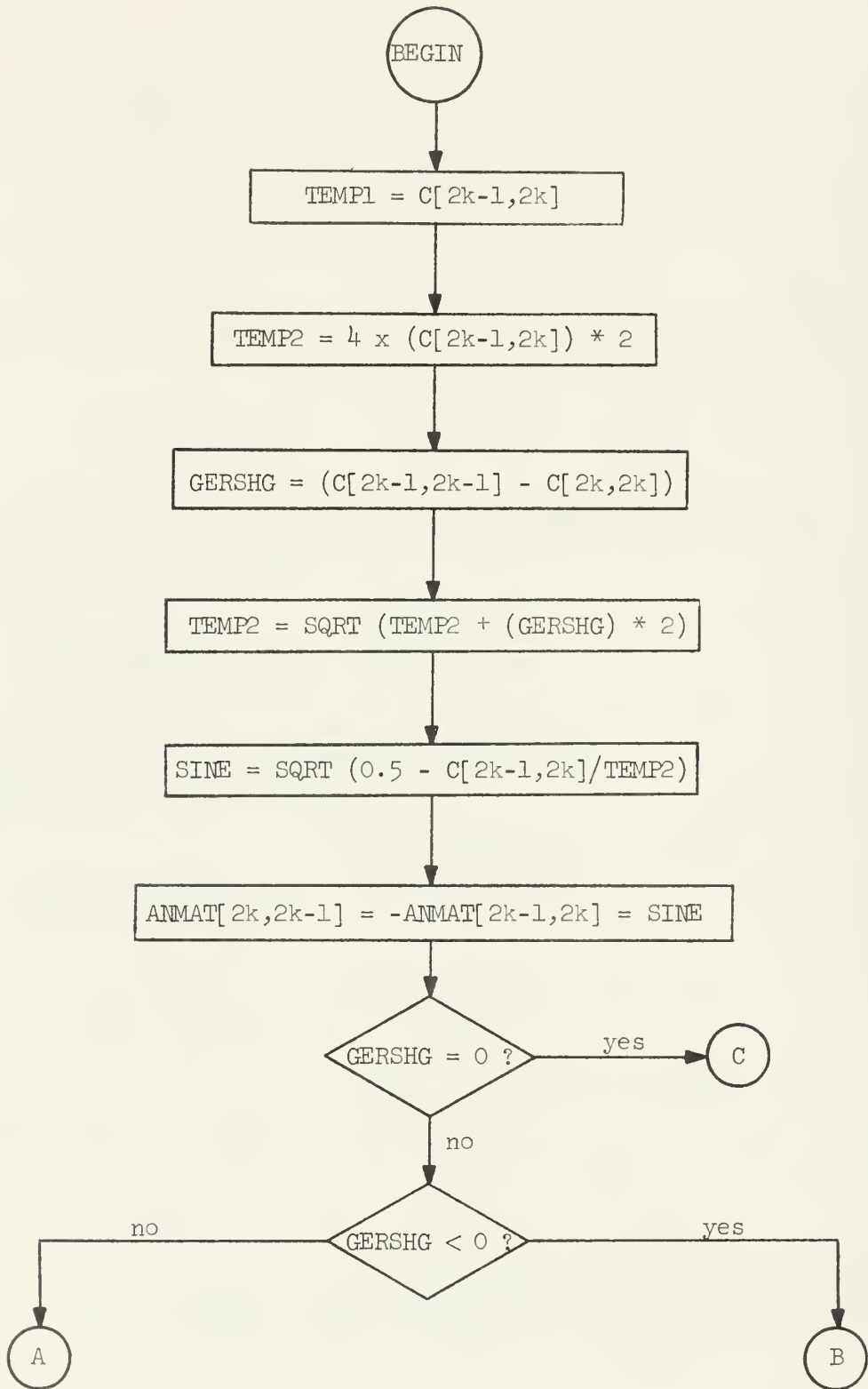
i) MLTRPS



ii) TRPS (A,B,N)



c) ANGL (ANMAT, C, N)



A

B

COSINE = SQRT (0.5 + C[2k-1,2k]/TEMP2)

COSINE = -SQRT (0.5 + C[2k-1,2k]/TEMP2)

D

C

TEMP1 > 0 ?

TEMP1 < 0 ?

COSINE = 1

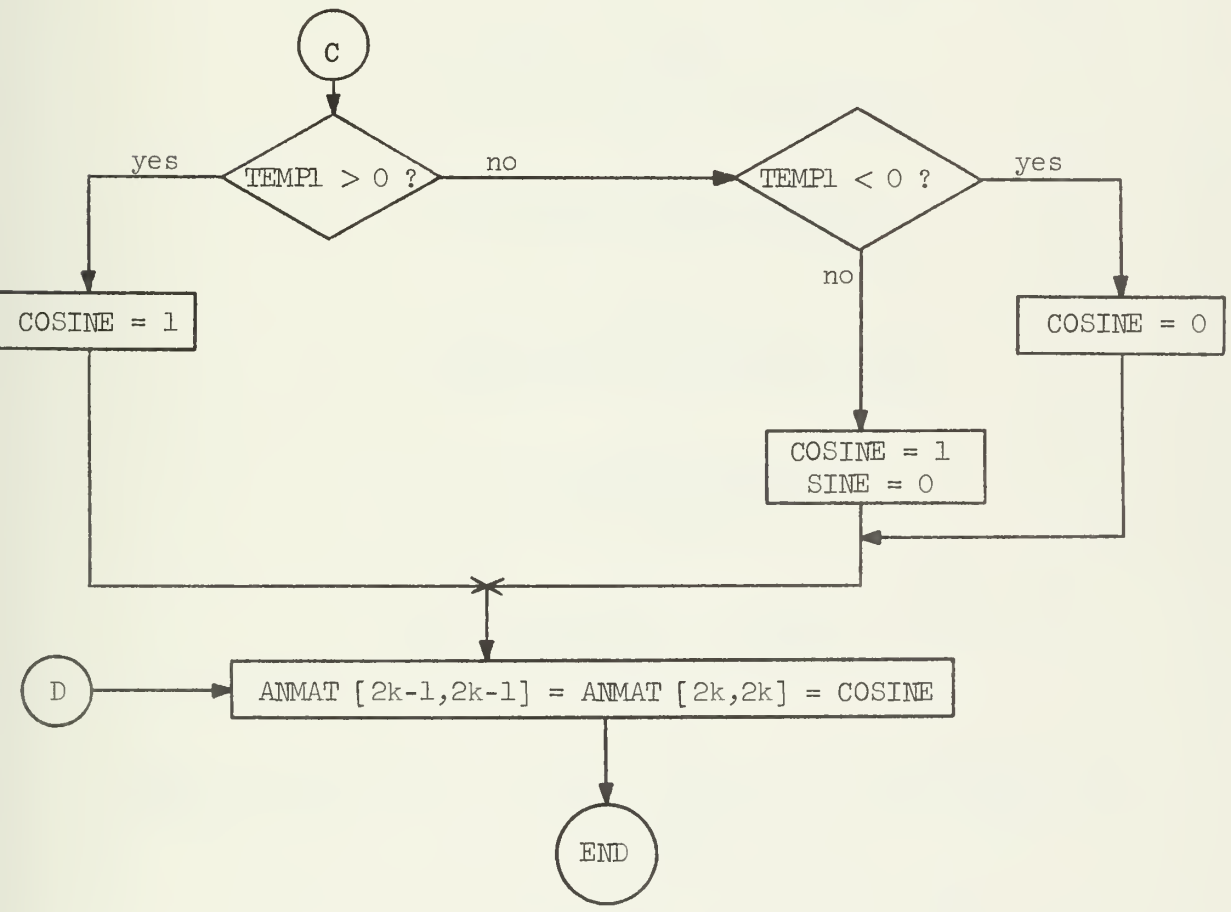
COSINE = 0

COSINE = 1  
SINE = 0

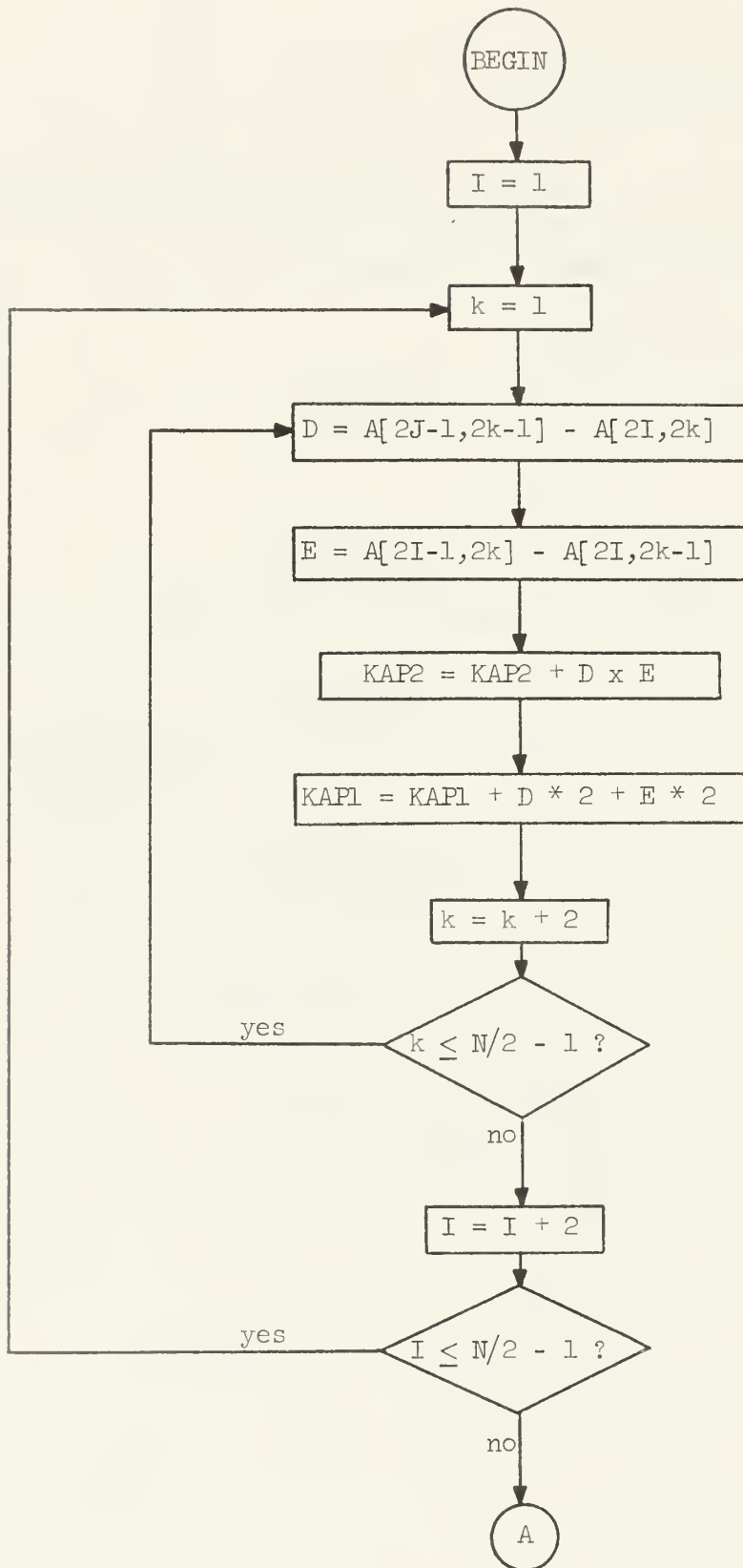
D

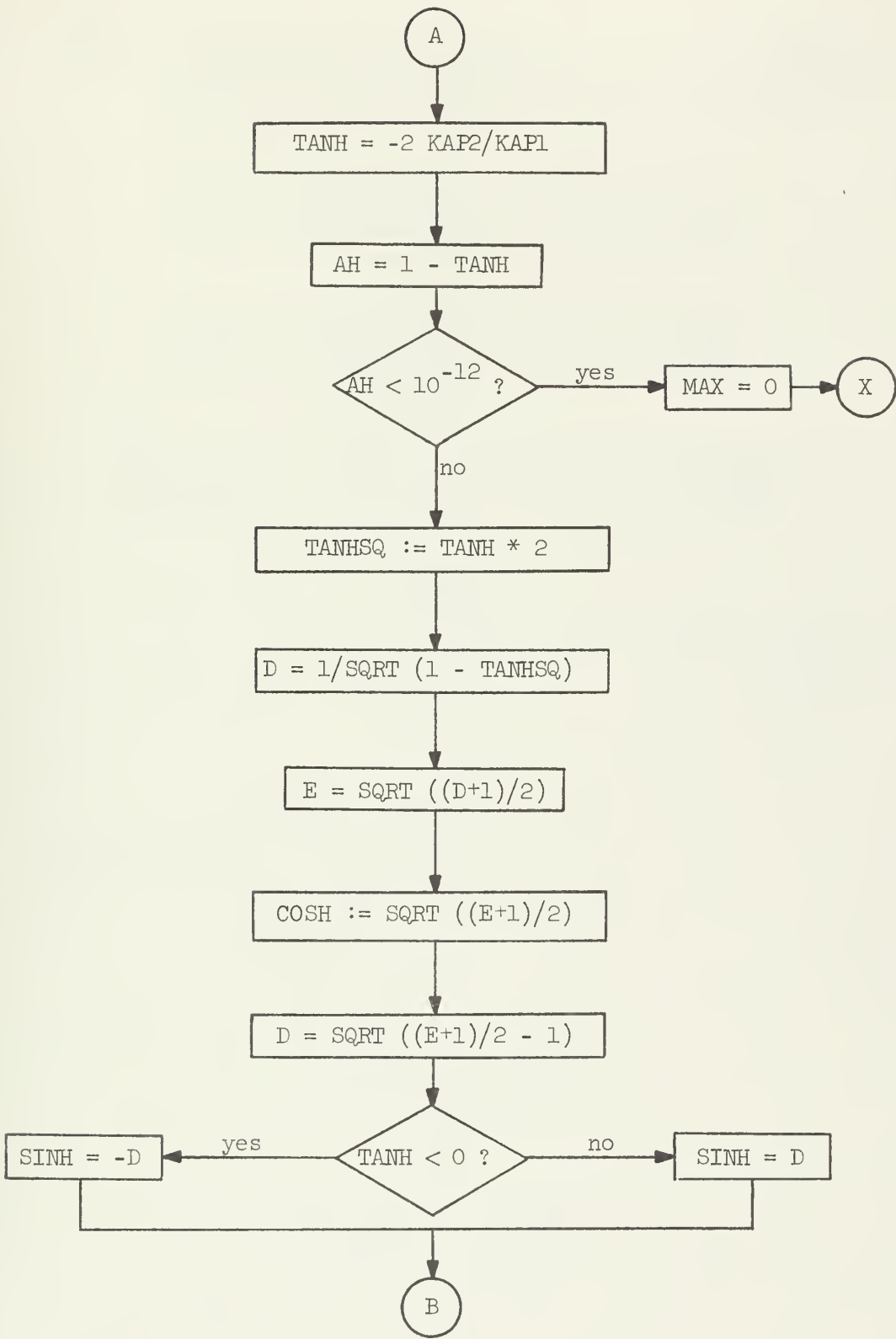
ANMAT [2k-1,2k-1] = ANMAT [2k,2k] = COSINE

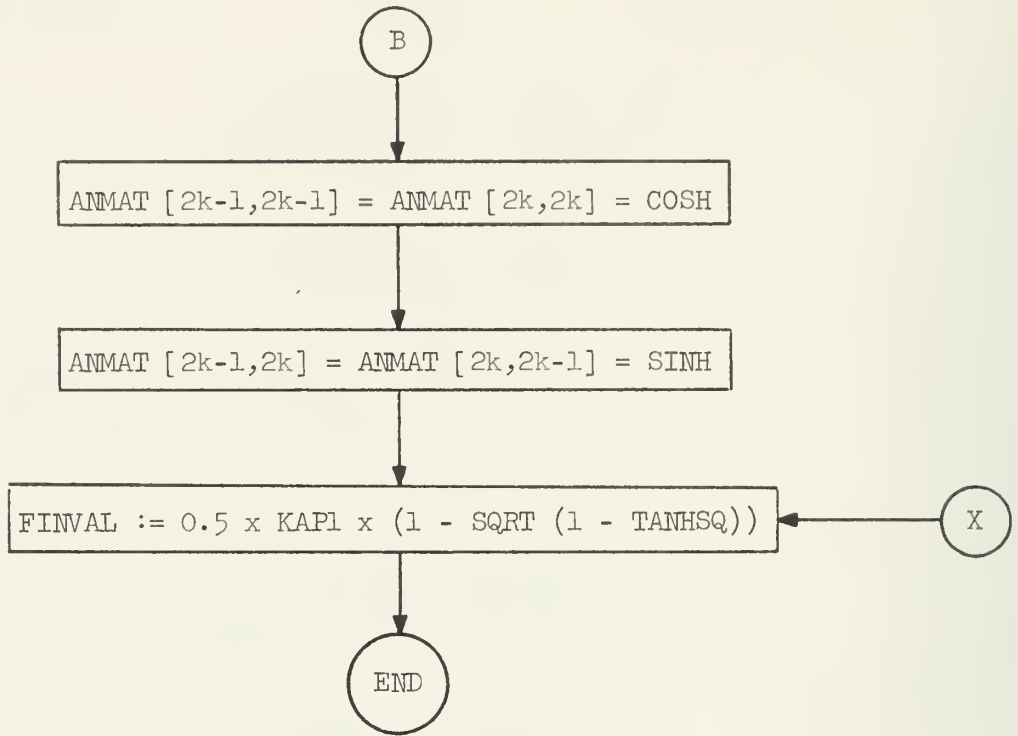
END



d) HYANG (ANMAT, A, N)

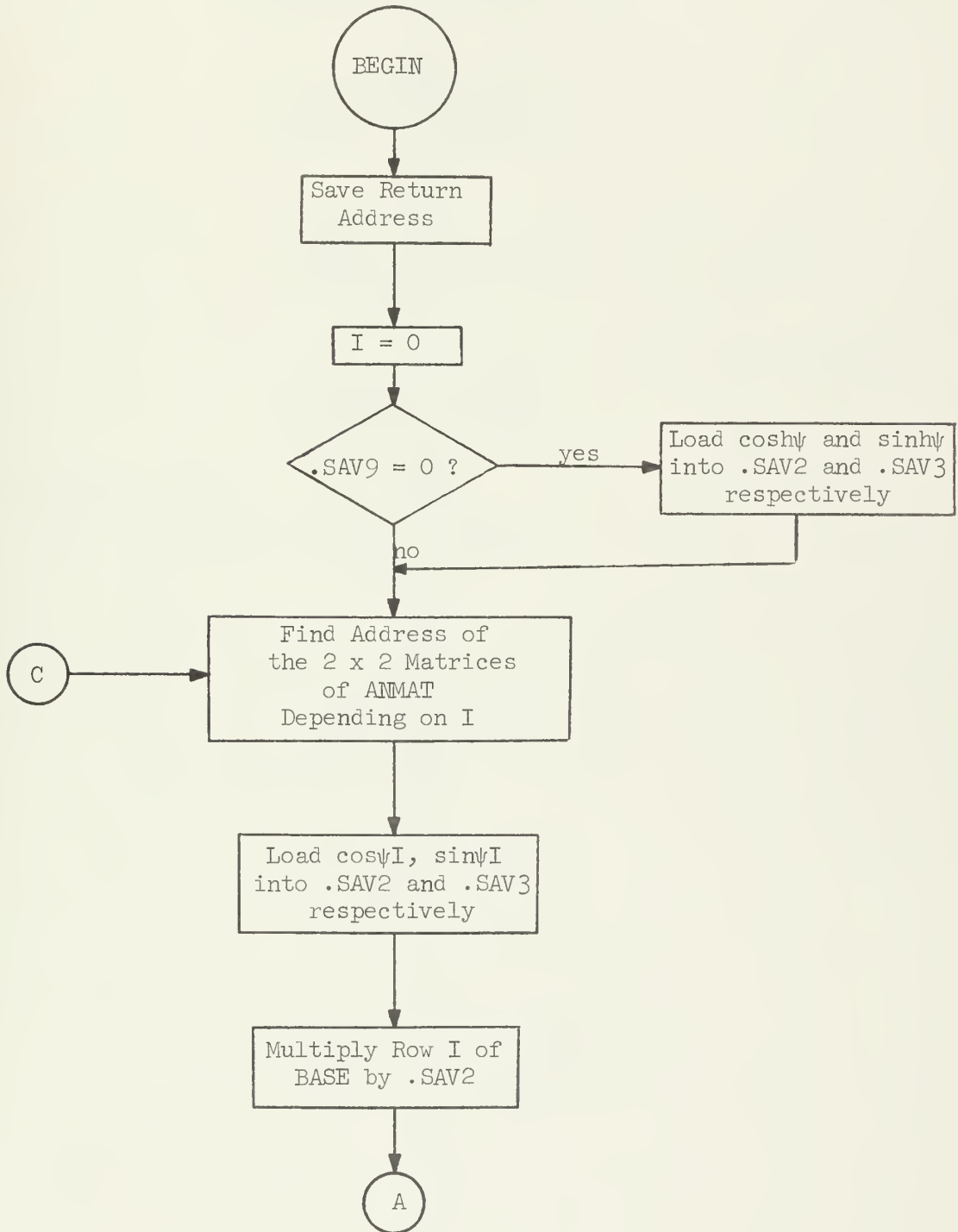


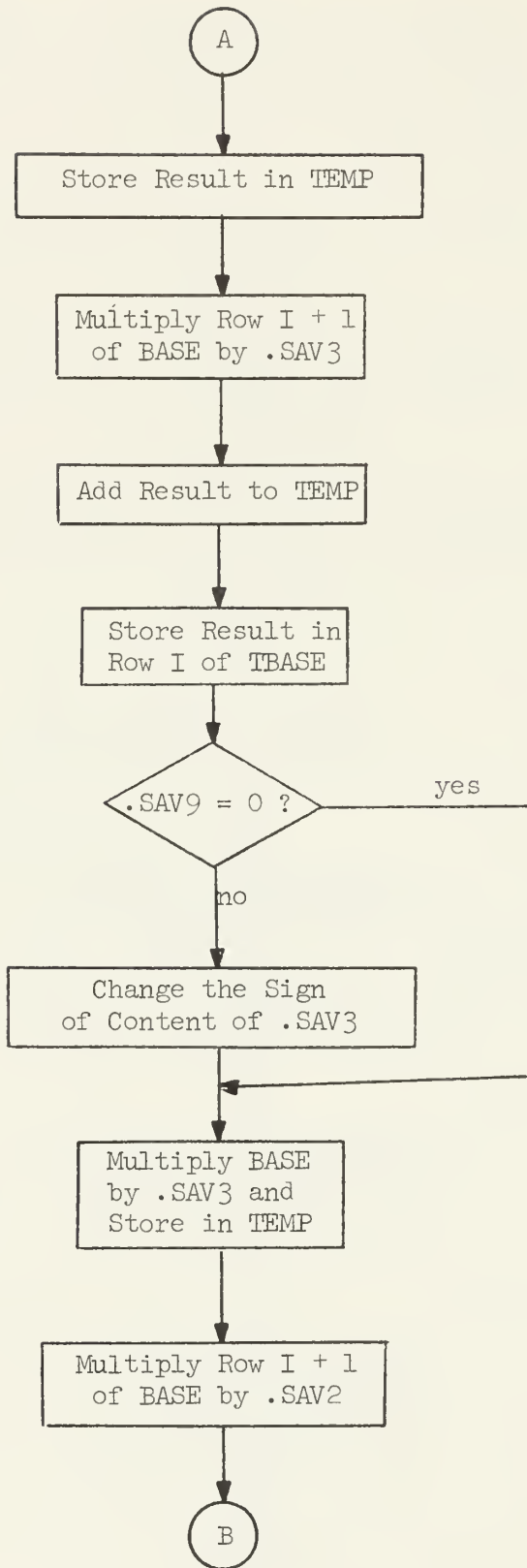


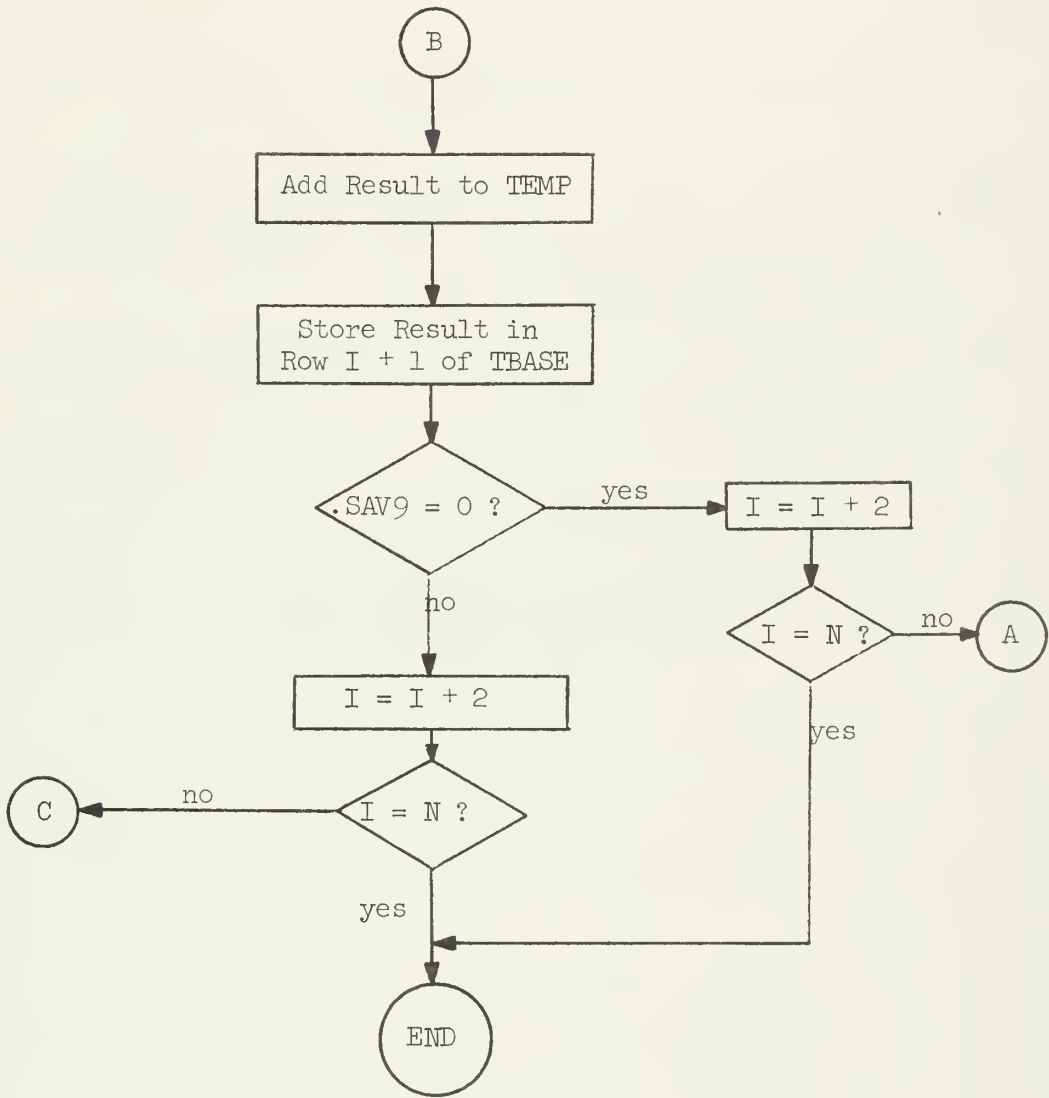




- e) MULSA: Multiplying Two Matrices  $BASE = (ANMAT)^t \cdot (BASE')$ .  
 If ANMAT was created in ANGL, the 2 x 2 matrices going down the diagonal are different; if it was created in HYANG, they are equal. A tag in .SAV9 notifies us of this fact.







## Bibliography

- [1] Sameh, A., et al., "Eigenvalue Problems," ILLIAC IV Document No. 127. Urbana, Illinois: ILLIAC IV Project, University of Illinois at Urbana-Champaign, (April 4, 1968).
- [2] Sameh, A., "On Jacobi and Jacobi-like Algorithms for a Parallel Computer," J. Math. Comp., July 1971.
- [3] Stevens, J., "Matrix Multiplications," a summary of unpublished ideas and private communications.
- [4] Kreyszig, E., Advanced Engineering Mathematics, 2nd ed. New York: John Wiley and Sons, Inc., 1967. Pp. 422-23.

APPENDIX A  
The Calling Program

```

BEGIN
FILL                128;

DEFINE CALL  $\beta$ NAME(BPARAMETERS)=
 $\beta$ IF  $\beta$ SIGN( $\beta$ MFIELD( $\beta$ NAME))  $\beta$  THEN
EXTERNAL  $\beta$ NAME;  $\beta$ FI
 $\beta$ IF  $\beta$ EMPTY( $\beta$ PARAMETERS)  $\beta$ THEN  $\beta$ ELSE
BEGIN BLOCK
BEGIN USE (63)
LIST:  DATA  $\beta$ PARAMETERS
END;
CLC(2);
SLIT(2) LIST;
END;  $\beta$ FI
CLC(3);
SLIT(3)  $\beta$ NAME;
EXCHL(3) $ICR;###;
DEFINE  MM=4###;
BASE:  DATA  3.9999970707,-3.05879213585 -4;
        DATA  -2.40792235573 -3,3.28938777238 -5,(0)60;

        DATA  -3.05879213585 -4,1.000000040880;
        DATA  -3.94565748157 -6,-5.94569958816 -5,(0)60;

        DATA  -2.40792235573 -3,-3.9456574157 -6;
        DATA  2.00000292089,-1.72122395238 -4,(0)60;

        DATA  3.28938777238 -5,-5.9456996816 -5;
        DATA  -1.72122395238 -4,2.99999997304,(0)60;

EIGV:  BLK    MM;%    the eigenvector matrix

ANMAT: BLK    MM;%    THE TRANSFORMATION MATRIX

TBASE: BLK    MM;%    TEMPORARY STORAGE

```

START: FILL;

#####

LIT(0) =1,3,0;% print out the original matrix

LIT(1) =1,BASE+3,BASE;

MAI: DISPLAYR \$C1,16;

LIT(2) =64;

CADD(1) \$C2;

CROTR(1) 24;

CADD(1) \$C2;

CROTL(1) 24;

TXEFM(0) ,MAI;

#####

CALL EIGEN(BASE,EIGV,ANMAT,TBASE,O,MM);

#####

LIT(0) =1,3,0;% PRINT THE DIAGONAL MATRIX

LIT(1) =1,BASE+3,BASE,

MAIE: DISPLAYR \$C1,16;

LIT(2) =64;

CADD(1) \$C2;

CROTR(1) 24;

CADD(1) \$C2;

CROTL(1) 24;

TXEFM(0) ,MAIE;

#####

LIT(0) =1,3,0;

LIT(1) =1,EIGV+3,EIGV;% PRINT THE EIGENVECTOR MATRIX

MAIE1: DISPLAYR \$C1,16;

LIT(2) =64;

CADD(1) \$C2;

CROTR(1) 24;

CADD(1) \$C2;

CROTL(1) 24;

TXEFM(0) ,MAIE1;

#####

END START.

APPENDIX B

The Subroutine EIGEN, Jacobi's Method





```

.ZEZE1  EQU  $n15;x  THRESHOLD FACTOR TO BE CHECKED AGAINST B0400006200
.BD1    EQU  $n16;x  THRESHOLD FOR SUPER-DIAGONALS. 00006300
.BD11   EQU  $n17;x
.BD21   EQU  $n18;x 00006500
.BD31   EQU  $n19;x 00006600
.INDEX1 EQU  $n20;x  INNER LOOP COUNT IN MAIN PROGRAM. 00006700
.S1     EQU  $n21;x  CONVERGENCE-FACTOR FOUND IN ADDIT-ROUTINE. 00006800
.CONVE1 EQU  $n22;x  CHECK FOR SUPERDIAGS EQL 0. 00006900
.SAV11  EQU  $n24;x  SAVE REGISTER. 00007000
.SAV21  EQU  $n25;x 00007100
.SAV31  EQU  $n26;x 00007200
.SAV41  EQU  $n27;x 00007300
.SAV81  EQU  $n28;x 00007400
.SAV51  EQU  $n29;x 00007500
.SAV91  EQU  $n30;x 00007600
.RETUR1 EQU  $n31;x  RETUR CONTAINS THE RETURN ADR. 00007700
%
% TO LINK TO THE OUTSIDE. 00007800
.ADR11  EQU  $n32;x  ADDRESS OF ORIGINAL MATRIX 00007900
.ADR81  EQU  $n33;x  ADDRESS OF EIGENVECTORMATR. 00008000
.ADRC1  EQU  $n34;x  ANGLE MATRIX 00008100
.ADRD1  EQU  $n35;x  TEMP. STORAGE MATRIX 00008200
.ADRE1  EQU  $n36;x  ADDRESS OF THE ERROR MATRIX 00008300
.SAV61  EQU  $n37;x 00008400
TEMP1   HLK  11x  ONE ROW OF SAVE-STORAGE IN PE-MEMORY 00008500
TEMP11  HLK  1; 00008600
TEMP21  HLK  1; 00008700
MANT1   HLK  1; 00008800
GERSH1  HLK  1; 00008900
NUMB1   DATA 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16; 00009000
          DATA 17,18,19,20,21,22,23,24,25,26,27,28,29,30; 00009100
          DATA 31,32,33,34,35,36,37,38,39,40,41,42,43,44; 00009200
          DATA 45,46,47,48,49,50,51,52,53,54,55,56,57,58; 00009300
          DATA 59,60,61,62,63;x PE NUMBERING 00009400
MESSO1  DATA "### PE I CONTAINS BOUND ON EIGENVAL. I ###"; 00009500
TACK1   WDS  0; 00009600
%MESS11 DATA "### BASE AFTER ANY ROW/COL. SHUFFLE ###"; 00009700
%MESS21 DATA "### THE TRANSFORMATION MATRIX - ANMAT ###"; 00009800
%MESS31 DATA "### BASE AFTER THE TRANSFORMATION ###"; 00009900
%MESS41 DATA "### BASE AFTER 2-ND ROW/COL. SHUFFLE ###"; 00100000
ADBSAV1 WDS  10; 00101000
% ***** 00102000
ROUTE1  FILL;x  ROUTE-ROUTINE FOR N=64 00103000
          STL(3)  .SAV3;x  SAVE RETURN ADDRESS. 00104000
          LDL(3)  .SAV21;x  LOAD ROUTING DISTANCE. 00105000
          RTL     $A,0(3);x $A IS ASSUMED TO CONTAIN THE ELEMENTS 00106000
          % TO BE ROUTED. 00107000
          LDL(3)  .SPEC;x 00108000
          LOFE1   $n3;x 00109000
          LDA     $R1 00110000
%***** 00111000
          LDG     NUMB1;x THIS PART CLEANS THE FIRST 0 PE'S 00112000
          LDL(3)  .SAV2;x OF RGA, IN CASE ELEMENTS HAVE BEEN 00113000
          ISL     $n3;x ROUTED INTO THOSE PE'S. FROM ACROSS 00114000
          SETE    I,AND,E;x THE BOUNDARY N. 00115000
          SETE1   E,AND,E; 00116000
          CLRA1  00117000
          LDL(3)  .ROUT;x BRING THOSE ELEMENTS WHICH WERE ROUTED 00118000
          RTL     0(3);x ACROSS BOUNDARY INTO THE FIRST 0 PE'S 00119000
          LDA     $R1 00120000
          LDL(3)  .SPEC;x 00121000
          COMPC(3); 00122000

```

```

LDEE1      $C3;                                00012300
CLRA;X     CLEAR RGA PAST THE BOUNDARY N      00012400
COMPC(3);  00012500
LDFE1      $C3;                                00012600
*****
LDL(3)     .SAV3;X   AND THE RESULT WILL BE IN $A PROPERLY 00012800
EXCHL(3)   $TCR;X   RUTED.THEN RETURN.          00012900
% *****
ROTAR:     FILL;                                00013100
% ADJUST ACAR1 FOR AN END-AROUND SHIFT,RIGHT FOR D LSS 64 00013200
STL(3)     .SAV3;X   SAVE RETURN ADDRESS          00013300
LDL(1)     .SAV3;X   PATTERN THAT NEEDS ADJUSTMENT 00013400
CRNTR(1)   0(0);X   ACARO CONTAINS D             00013500
LDL(2)     .SPEC;   00013600
CSHR(2)    0(0);    00013700
CAND(2)    $C1;    00013800
CAND(2)    $D5;    00013900
CEXOR(1)   $C2;    00014000
LDL(3)     .N;     00014100
CRNTR(1)   0(3);    00014200
CEXOR(1)   $C2;    00014300
STL(1)     .SAV3;  00014400
LDL(3)     .SAV3;  00014500
EXCHL(3)   $ICR;  00014600
*****
ROTAL:     FILL;                                00014800
% ADJUST ACAR1 FOR AN END-AROUND SHIFT,LEFT FOR D LSS 64 00014900
STL(3)     .SAV3;  00015000
LDL(1)     .SAV3;  00015100
CRNTR(1)   0(0);  00015200
LDL(3)     .SPEC;  00015300
CSHL(3)    0(0);  00015400
CAND(3)    $C1;  00015500
CAND(3)    $D5;  00015600
CEXOR(1)   $C3;  00015700
LDL(2)     .RUIT;  00015800
CSHL(1)    0(2);  00015900
CEXOR(1)   $C3;  00016000
STL(1)     .SAV3;  00016100
LDL(3)     .SAV3;  00016200
EXCHL(3)   $ICR;  00016300
*****
RWSM:     FILL;                                00016500
%
% SUM UP IN ABS. VALUE EVERY ROW OF THE 00016600
% MATRIX BASE AND FIND FIRST THE MAXIMUM 00016700
% OF THE INDIVIDUAL SUMS AND THEN THE 00016800
% INDEX OF THE ROW WHERE THE MAXIMUM CAME 00016900
% FROM. 00017000
% SAVE RETURN ADDRESS. 00017100
STL(3)     .SAV1;X  00017200
SETE      E.OR.-E;  00017300
SETE1     E.AND.-E; 00017400
CLRA;     00017500
LDS       $A;      00017600
LIT(0)    =0,1,0;X LOOP FOR PICKING UP THE ROWS OF THE MATRI 00017700
CADD(0)    $D3;    00017800
CRNTR(0)   24;    00017900
RSL1:     LDL(1)    .SPEC;X   THE ELEMENT A[I,I] DOES NOT PARTICIPATE 00018000
% IN THE SUMMING.THEREFORE WE TURN OFF 00018100
% PE[I] BY THIS CONSTRUCT 00018200
CCR(1)    0(0);X   00018300
LDL(2)    .ADRA;  00018400
CADD(2)    $C0;  00018500

```

```

LDEE1    %C1;          00018400
LDA      0(2);        00018500
CLC(3);  00018600
LDEE1    %C3;          00018700
LIT(1)   =0.0;        00018800
LDA      %C1;          00018900
LDL(1)   .SPEC;        00019000
LDEE1    %C1;          00019100
ADRN     %S;#         FORM PARTIAL SUM. WE KNOW THE
%                                MATRIX IS SYMMETRIC          00019200
%                                00019300
LDS      %A;          00019400
TXETM(0) .+1;         00019500
JUMP     RSU1;        00019600
STS      TEMP;        00019700
SETE     E.OR.=E;#    FIND MAXIMAL VALUE IN TEMP.          00019800
SETE1    E.AND.E;     00019900
LIT(2)   =0.0;        00020000
LDS      %C2;          00020100
LDA      %S;          00020200
LDL(2)   .SPEC;        00020300
LDEE1    %C2;          00020400
LDA      TEMP;        00020500
SETE     E.OR.=E;     00020600
SETE1    E.AND.E;     00020700
LOS      %A;          00020800
LIT(1)   =1;          00020900
RSM#     RTL    %S,0(1);# ROUTE %S IN POWERS OF TWO          00021000
LDS      %R;          00021100
IAL      %S;#         ELIMINATE SMALLER VALUES OF %A WHEN
SETE     I.AND.E;#    COMPARED WITH %S.                    00021200
SETE1    E.AND.E;     00021300
LDA      %S;          00021400
SETE     E.OR.=E;     00021500
SETE1    E.AND.E;     00021600
LDS      %A;          00021700
CADD(1)  %C1;          00021800
LIT(2)   =64;         00021900
EQLXF(1) %C2,RSM;#    MAXIMAL VALUE IS FOUND, WHEN TEST FAILS.
%                                WHERE DID THE LARGEST VALUE COME FROM
%                                FIND ROW INDEX.              00022000
%                                00022100
%                                00022200
%                                00022300
LDL(2)   .SPEC;        00022400
LDEE1    %C2;          00022500
LDA      TEMP;#         AT THIS POINT EVERY PE CONTAINS THE
%                                MAXIMAL VALUE IN %S, WHILE THE FIRST N
%                                PEs CONTAIN THE INDIVIDUAL SUMS IN %A.
%                                00022600
%                                00022700
%                                00022800
IAL      %S;          00022900
SETC(1)  I;           00023000
COMPC(1); 00023100
CAND(1)  %C2;          00023200
LEADD(1); 00023300
LIT(0)   =7718;       00023400
CAND(1)  %C0;          00023500
STL(1)   .MAX;#       .MAX CONTAINS THE ROW INDEX WHERE LARGEST
%                                VALUE CAME FROM.              00023600
%                                00023700
LDL(3)   .SAV1;#      RETURN TO THE MAIN PROGRAM.          00023800
EXCHL(3) %ICR;        00023900
%*****
ANYR#    FILL;        00024000
%                                00024100
%                                ANYR BRINGS THE ROW WITH THE LARGEST SUM
%                                INTO THE TOP ROW AND PULLS ALL OTHER WITH
%                                IT IN A CIRCULAR MOTION.      00024200
%                                00024300
%                                00024400

```

```

STL(3)      .SAV1;X      SAVE RETURN ADDRESS.      00024500
LDL(2)      .N;          00024600
CSUB(2)     $D7;X      N=,MAX, MAX IS THE ROW=INDEX FOR THE 00024700
%           LARGEST ROW=SUM, N=MAX IS THE ROUTING 00024800
%           DISTANCE. 00024900

STL(2)      .SAV2;      00025000
LDL(3)      .SPEC;      00025100
LDEF1      $C3;          00025200
LIT(0)      =0,1,0;X    BRING THE ELEMENTS INTO THEIR RESPECTIVE 00025300
CRQTL(0)    24;X      POSITIONS WITHIN THEIR ROWS. 00025400
CADD(0)     $D3;          00025500
AY1: LDL(2)      .ADRES; 00025600
CADD(2)     $C0;          00025700
LDA         0(2);        00025800
CLC(3);     00025900
SLIT(3)    =ROUTE;EXCHL(3) $ICR;X GO TO ROUTE ROUTINE 00026000
STA         0(2);        00026100
TXEFM(0)    .AY;X      THE ELEMENTS ARE IN THEIR NEW POSITIONS 00026200
LIT(0)      =0; X      NOW WE BRING THE ELEMENTS INTO THEIR 00026300
%           FINAL LOCATION BY CIRCULAR MOTION 00026400

LIT(1)      =0,1,0;    00026500
CADD(1)     $D3;          00026600
CRQTL(1)    24;          00026700
CADD(1)     $D7;          00026800
AY1: LDL(3)      .ADRES; 00026900
CADD(3)     $C1;          00027000
LDA         0(3);        00027100
LDL(2)      .ADRD;      00027200
CADD(2)     $C0;          00027300
STA         0(2);        00027400
ALIT(0)     =1;          00027500
TXEFM(1)    .AY1;       00027600
LIT(1)      =1,0,0;    00027700
CRQTR(1)    24;          00027800
CADD(1)     $D7;          00027900
CSUB(1)     $D1;          00028000
AY2: CRQTL(1)    24;          00028100
LDL(3)      .ADRES;    00028200
CADD(3)     $C1;          00028300
LDA         0(3);        00028400
LDL(2)      .ADRD;      00028500
CADD(2)     $C0;          00028600
STA         0(2);        00028700
ALIT(0)     =1;          00028800
TXEFM(1)    .AY2;       00028900
LIT(1)      =1,0,0;X    BRING THE ELEMENTS FROM TRASE INTO 00029000
CRQTR(1)    24;X      THEIR RESPECTIVE MATRIX. 00029100
CADD(1)     $D3;          00029200
CRQTL(1)    24;          00029300
AY3: LDL(3)      .ADRES; 00029400
CADD(3)     $C1;          00029500
LDL(2)      .ADRD;      00029600
CADD(2)     $C1;          00029700
LDA         0(2);        00029800
STA         0(3);        00029900
TXEFM(1)    .AY3;       00030000
LDL(3)      .SAV1;      00030100
EXCHL(3)    $ICR;X      GO BACK INTO MAIN=PROGRAM 00030200
% ***** 00030400
SHUFL: FILL; 00030500

```

			SHUFL BRINGS 2ND ROW TO THE BOTTOM AND 2ND COLUMN TO THE RIGHT HAND END.	00030600
X				00030700
	STL(3)	.SAV1;		00030800
	LDL(3)	.ADRES;*	PICK UP THE SECOND ROW OF THE MATRIX	00030900
	CADD(3)	%D1;*	IN USE AND STORE IT TEMPORARILY IN TEMP	00031000
	LDL(0)	.SPEC;		00031100
	LDEE1	%C0;		00031200
	LDA	0(3);		00031300
	STA	TEMP;		00031400
	LDL(2)	%C3;		00031500
	LDL(0)	.N;		00031600
	CADD(0)	%D4;		00031700
SH1	CADD(2)	%D1;		00031800
	EQLXT(2)	%C0,SHH;		00031900
	LDA	0(2);		00032000
	STA	0(3);		00032100
	CADD(3)	%D1;		00032200
	SKYP	.SH;		00032300
SHH1	LDA	TEMP;		00032400
	STA	0(3);*	END 2ND ROW SHUFFLE	00032500
	LIT(0)	=0,1,0;*	SKEW MATRIX IN USE FOR COLUMN EXCESS	00032600
	CADD(0)	%D3;		00032700
	CRNTL(0)	24;		00032800
SH11	LDL(2)	.ADRES;		00032900
	CADD(2)	%C0;		00033000
	LDA	0(2);		00033100
	STL(0)	.SAV2;		00033200
	CLC(3);			00033300
	SLIT(3)	=R01TE;EXCHL(3) %ICR;		00033400
	STA	0(2);		00033500
	TXEFM(0)	.SH1;*	SKEW=END	00033600
	LDA	NUMB;*	ADJUST INDEX TO PICK UP 2ND COLUMN.	00033700
	LDL(0)	.ONE;		00033800
	STL(0)	.SAV2;		00033900
	CLC(3);			00034000
	SLIT(3)	=R01TE;EXCHL(3) %ICR;		00034100
	LDS	%A;		00034200
	LDX	%S;*	END ADJUST	00034300
	LDL(1)	.ADRES;*	PICK UP 2ND COLUMN AND STORE IT IN TEMP.	00034400
	LDA	*0(1);		00034500
	STA	TEMP;*	END PICK UP	00034600
	LIT(0)	=0,1,0;*	REARRANGE THE REST OF THE MATRIX	00034700
	CADD(0)	%D3;		00034800
	CRNTL(0)	24;		00034900
SH21	LDL(2)	.SPEC;		00035000
	LIT(3)	=140000000000000000000000000018;		00035100
	CXOR(2)	%C3;		00035200
	STL(2)	.SAV8;		00035300
	CLC(3);			00035400
	SLIT(3)	=R01AR;EXCHL(3) %ICR;		00035500
	LDL(2)	.SAV8;		00035600
	LDEE1	%C2;		00035700
	LDL(1)	.ADRES;		00035800
	CADD(1)	%C0;		00035900
	STL(1)	.SAV4;		00036000
	LDA	0(1);		00036100
	LDL(3)	.NUM0;		00036200
	STL(3)	.SAV2;		00036300
	CLC(3);			00036400
	SLIT(3)	=R01TE;EXCHL(3) %ICR;		00036500
	STL(0)	.SAV5;		00036600



```

LDL(0)      .ONE;                00036700
STL(2)      .SAV8;                00036800
CLC(3);     00036900
SLIT(3)     =ROTAL;EXCHL(3) SICR; 00037000
LDL(2)      .SAV8;                00037100
LDEE1       $C2;                  00037200
LDL(0)      .SAV5;                00037300
LDL(1)      .SAV4;                00037400
STA         0(1);                  00037500
TXEFM(0)    ,SH2;%                00037600      END REST=ARRANGE
LDL(0)      .SPEC;%                00037700      BRING 2ND COL. INTO LAST COL.
LDEE1       $C0;                  00037800
LDA         %X;%                   00037900      FIRST; ADJUST INDEX
LDL(1)      .NM0;                  00038000
CSUB(1)     $D1;                  00038100
STL(1)      .SAV2;                00038200
CLC(3);     00038300
SLIT(3)     =R0ITE;EXCHL(3) SICR; 00038400
LDS         $A;                    00038500
LDX         $S;%                   00038600      END INDEX=ADJUST
LDA         TEMP;                  00038700
LDL(2)      .NM0;                  00038800
CSUB(2)     $D1;                  00038900
STL(2)      .SAV2;                00039000
CLC(3);     00039100
SLIT(3)     =R0ITE;EXCHL(3) SICR; 00039200
LDL(1)      .ADRES;                00039300
STA         *0(1);%                00039400      END COL.=SWITCH
LDL(1)      .ADRES;%               00039500      UNSKEW MATRIX IN USE
LDL(2)      .N;                    00039600
SH3:        CADD(1) $D1;            00039700
           CSUB(2) $D1;            00039800
           STL(2)  .SAV2;          00039900
           LDA     0(1);            00040000
           CLC(3);                  00040100
           SLIT(3) =R0ITE;EXCHL(3) SICR; 00040200
           STA     0(1);            00040300
           EQLXF(2) $D1,SH3;%      00040400      END UNSKEW
           LDL(3)  .SAV1;%         00040500      RETURN TO MAIN=PROGRAM
           EXCHL(3) SICR;          00040600
% *****
ANGLE:     FILL;                    00040700
%
%          ANGLE FINDS THE TRANSFORMATION=MATRIX
%          CONSISTING OF SINES & COSINES AS 212
%          DIAGONAL MATRICES.      00040800
%
%          STL(3)  .SAV1;            00041200
%          LIT(0)  =0.2,0;%         00041300      CREATE THE TURN-ON PATTERN FOR THE
%                                     SUPER-DIAGONALS
%
%          CADD(0) $D3;              00041500
%          CR0TL(0) 24;              00041600
%          CADD(0) $D1;              00041700
%          LDL(1)  .SPEC;            00041800
AN:        CCR(1)  0(0);              00041900
           TXEFM(0) ,AN;%           00042000      END TURN-ON PATTERN
           STL(1)  .ANTUN;          00042100
           LDEE1   $C1;              00042200
           LDA     NUMB;%            00042300      FIND CORRECT INDEX
           LDS     =1;                00042400
           ADM     $S;                00042500
           STA     TEMP;              00042600
           LDA     NUMB;              00042700

```

```

RIL          SA,1;                                00042800
CSHR(1)     1;                                    00042900
LOEE1       SC1;                                    00043000
LDA         SR;                                    00043100
STA         TEMP1;  TEMP CONTAINS THE INDEX OF THE FORM 00043200
%           2L+1*2L FOR L=0,1,2,...,N-1/2          00043300
LDL(0)      .ANTUN;                                00043400
LOEE1       SC0;                                    00043500
LDX         TEMP;                                    00043600
LOL(3)      .ADRA;                                    00043700
LDA         +0(3);%  LOAD A[2I,2I-1] FOR ALL I LESS THAN NREQN/2 00043800
ADRN        SA;%  2[A[2I,2I-1]].                    00043900
STA         TEMP1;                                    00044000
RTL         SA,1;                                    00044100
CSHR(0)     1;                                    00044200
LOEE1       SC0;                                    00044300
LDA         SR;                                    00044400
STA         TEMP1;%  ALL PE S CONTAIN THE VALUE 2[A[2I,2I-1] 00044500
%           IN GROUPS OF 2.                          00044600
LDL(2)      .SPEC;                                    00044700
LOEE1       SC2;                                    00044800
LDX         NUMB;                                    00044900
LDL(0)      .ADRA;                                    00045000
LDS         +0(0);%  LOAD A[I,I] FOR ALL I              00045100
LOL(3)      .ANTUN;                                    00045200
LOEE1       SC3;                                    00045300
LDA         SA;%  SA CONTAINS A[2I-1,2I-1] IN EVERY OTHER PE 00045400
RTL         SA,-1;                                    00045500
LDS         SR;                                    00045600
SRN         SA;%  FORM A[2I-1,2I-1]-A[2I,2I]          00045700
STA         TEMP2;                                    00045800
RTL         SA,1;                                    00045900
CSHR(3)     1;                                    00046000
LOEE1       SC3;                                    00046100
LDA         SR;                                    00046200
STA         TEMP2;%  ALL PE S OF TEMP2 CONTAIN A[2I-1,2I-1]- 00046300
%           A[2I,2I] IN GROUPS OF 2.                  00046400
%           NOW WE HAVE TO DIFFERENTIATE BETWEEN CASES 00046500
% *****                                           00046600
%CASE 1: TEMP2=0                                     00046700
LOEE1       SC2;                                    00046800
LDA         TEMP2;                                    00046900
LIT(0)      =0.0;                                    00047000
IME         SC0;%  SET I WHERE MANTISSA IS ZERO,        00047100
% AND MASK OUT THE PART BEYOND THE BOUNDARY N      00047200
SETI       I,AND.E;                                    00047300
SETC(0)     E;                                       00047400
STL(0)      .ANTUN1;%                                00047500
ZERF(0)     +1;%  IF NONE OF THE TEMP2 WERE ZERO, THEN GO 00047600
JUMP        AN1;%  TO AN1; OTHERWISE CONTINUE.         00047700
LOEE1       SC0;                                    00047800
LIT(3)      =0.5;                                    00047900
LDA         SC3;                                    00048000
CALL SQRT64();                                       00048100
LDX         NUMB;                                    00048200
LOL(1)      .ADRC;                                    00048300
STA         +0(1);%  COSINES = SQRT(0.5) WHERE TEMP2=0. 00048400
LDA         TEMP1;                                    00048500
% *****                                           00048600
%CASE1A: FIND OUT WHERE TEMP1 LESS 0 WHERE TEMP2=0. 00048700
LOL(1)      .ZER0;                                    00048800

```



	IAL	SC1;		00048900
	SETC(1)	1;		00049000
	CAND(1)	SC0;		00049100
%	ZERT(1)	AN2;*	IF TEMP1 IS NOWHERE LSS 0 WHERE TEMP2=0 THEN GO TO AN2. OTHERWISE	00049200
	LDL(2)	ANTUN;*	PLACE THE SINES WITH THEIR CORRECT SIGNS	00049300
	CSHR(2)	1;		00049400
	CAND(2)	SC1;		00049500
	LDEE1	SC2;		00049600
	LDS	TEMP;*	TEMP CONTAINS INDEX FOR SUPER-DIAGONALS	00049700
	LDL(3)	ADRC;		00049800
	LDA	*n(j);		00049900
	CHSA;			00050000
	STA	#n(3);		00050100
	CSHL(2)	1;		00050200
	LDEE1	SC2;		00050300
	LDS	TEMP;		00050400
	LDA	*n(3);		00050500
	STA	*n(j);		00050600
	CUMPC(1);			00050700
	CAND(0)	SC1;*		00050800
%	*****			00050900
%CASE1B;	TEMP2=0 AND TEMP1 GTR UR EQL 0.			00051000
	ZERT(0)	AN1;		00051100
AN2;	LDEE1	SC0;		00051200
	LDL(3)	ADRC;		00051300
	LDA	*n(3);		00051400
	LDS	TEMP;		00051500
	LDL(1)	ANTUN;		00051600
	CAND(1)	SC0;		00051700
	LDEE1	SC1;		00051800
	CHSA;			00051900
	STA	*n(3);		00052000
	CSHR(1)	1;		00052100
	LDEE1	SC1;		00052200
	STA	*n(j);		00052300
	LOL(0)	ANTUN;*		00052400
%	*****			00052500
%CASE2;	FIND COSINE & SINE FOR TEMP2=0			00052600
AN1;	LDL(1)	SPEC;		00052700
	CEXDR(0)	SC1;		00052800
	LDEE1	SC0;		00052900
	LDA	TEMP1;		00053000
	DVRN	TEMP2;		00053100
	STA	TEMP1;		00053200
	MLRN	SA;		00053300
	LIT(3)	=1.;		00053400
	ADRN	SC3;		00053500
	CALL SQRT64();			00053600
	LDS	SA;		00053700
	LIT(3)	=0.5;		00053800
	LDA	SC3;		00053900
	DVRN	TS;		00054000
	STA	TEMP2;		00054100
	LIT(3)	=0.5;		00054200
	LDA	SC3;		00054300
	ADRN	TEMP2;		00054400
	CALL SQRT64();			00054500
	LDS	NUMB;		00054600
	LDL(3)	ADRC;		00054700
	STA	*n(j);*	COSINES STORED FOR TEMP2=0	00054800
				00054900

```

LIT(3)      =0,5;                                00055000
LOA         $C3;                                  00055100
SBRN       TFMP2;                                00055200
CALL SQRT64();                                    00055300
% *****
%CASE2A;    TEMP1 LSS 0                            00055400
           STA     TFMP2;                          00055600
           LDA     TFMP1;                          00055700
           ISN;                                       00055800
           SETC(1)  I;                               00055900
           CAND(1)  $C0;                             00056000
           ZERT(1)  ,AN3; * NONE OF THE TEMP1 WERE LSS 0, SO GO TO AN3 00056100
           LDEE1   $C1;                               00056200
           LDA     TFMP2;                          00056300
           LDS     TFMP;                            00056400
           LDL(2)  ,ANTUN;                          00056500
           CAND(2)  $C1;                             00056600
           LDEE1   $C2;                               00056700
           LDL(3)  ,ADRC;                          00056800
           CHSA;                                       00056900
           STA     #0(3);                          00057000
           CSHR(2)  1;                               00057100
           LDEE1   $C2;                             00057200
           STA     #0(3);                          00057300
           COMPC(1);                                00057400
           CAND(0)  $C1; ZERT(0) ,AN4;             00057500
% *****
%CASE2B;    TEMP1 GEQ 0                            00057600
AN3;        LDEE1   $C0;                             00057700
           LDS     TEMP;                            00057800
           LDA     TFMP2;                          00057900
           LDL(1)  ,ANTUN;                          0005A100
           CAND(1)  $C0;                             00058200
           LDEE1   $C1;                               00058300
           LDL(3)  ,ADRC;                          00058400
           STA     #0(3);                          0005A500
           CSHR(1)  1;                               00058600
           LDEE1   $C1;                             00058700
           CHSA;                                       00058800
           STA     #0(3);                          00058900
AN4;        LDL(3)  ,SAV1;                          00059000
           EXCHL(3) $ICR;                          00059100
% *****
MULTPL;    FILL;                                    00059200
%
%          MULTIPLICATION OF TWO MATRICES. THE
%          SET-UP IS SUCH THAT THE ADDRESS OF
%          THE MATRIX IS TREATED AS A VARIABLE
%          SAVE RETURN ADDRESS
           STL(3)   ,SAV1; *                          00059700
           LIT(1)  =0,1,0;                          00059800
           CAND(1)  $C3;                              00059900
           CRNTL(1) 24;                               00060000
           LDL(3)   ,SPEC;                          00060100
           LDEE1   $C3;                              00060200
MULT;      LDY     NUMB; * PE=NUMBERS                00060300
           LDL(3)  ,ADRES; * =AMATRIX=BASE          00060400
           CAND(3)  $C1;                              00060500
           LDA     0(3); * LOAD ROW OF AMATRIX       00060600
           LIT(0)  =0,1,0;                          00060700
           CAND(0)  $C3;                              00060800
           CRNTL(0) 24;                              00060900
           LOS     =0;                                00061000

```

MUL1:;	LDL(3)	.ADRRES1;X	BMATRIX=BASE	00061100
	MLRN	*n(3);X	MULTIPLY BMATRIX	00061200
	ADRN	%S;X	FORM PARTIAL SUM	00061300
	STA	MAINT;		00061400
	SETE	E.OR.=E;		00061500
	SETE1	E.AND.E;		00061600
	CLRA;			00061700
	LDS	%A;		00061800
	LDL(3)	.SPEC;		00061900
	LDEE1	%C3;		00062000
	LDA	%R;		00062100
	EQLXF(0)	%D1*MUL3;X	IF %C0=1, CHECK IF .SAV9=1	00062200
	LDL(3)	.SAV9;		00062300
	EQLXF(3)	%D1*MUL3;		00062400
	CADD(0)	%D3;		00062500
	CSUB(0)	%D1;		00062600
	LDL(2)	.N=0;		00062700
	CSUB(2)	%D1;		00062800
	JUMP	MUL4;		00062900
MUL3;	LDL(2)	.ONE;		00063000
MUL4:;	STL(2)	.SAV2;		00063100
	CLC(3);			00063200
	SLIT(3)	=RQITE;EXCHL(3) %ICR;SKIP ,0;		00063300
	STA	TEMP;		00063400
	SETE	E.OR.=E;		00063500
	SETE1	E.AND.E;		00063600
	CLRA;			00063700
	LDS	%A;		00063800
	LDL(3)	.SPEC;		00063900
	LDEE1	%C3;		00064000
	LDA	%X;		00064100
	CLC(3);			00064200
	SLIT(3)	=RQITE;EXCHL(3) %ICR;		00064300
	LDS	%A;		00064400
	LDX	%S;		00064500
	SETE	E.OR.=E;		00064600
	SETE1	E.AND.E;		00064700
	CLRA;			00064800
	LDS	%A;		00064900
	LDL(3)	.SPEC;		00065000
	LDEE1	%C3;		00065100
	LDS	MAINT;		00065200
	LDA	TEMP;		00065300
	TXLFM(0)	,+1;		00065400
	JUMP	MUL1;		00065500
	LDL(2)	.ADR0;		00065600
	CADD(2)	%C1;		00065700
	STS	O(2);		00065800
	TXLFM(1)	,+1;		00065900
	JUMP	MUL;		00066000
	LIT(0)	=0,1,0;		00066100
	CADD(0)	%C3;		00066200
	CRCTL(0)	24;		00066300
MUL2;	LDL(3)	.ADRRES2;		00066400
	CADD(3)	%C0;		00066500
	LDL(2)	.ADR0;		00066600
	CADD(2)	%C0;		00066700
	LDA	O(2);		00066800
	STA	O(3);		00066900
	TXFFM(0)	.MUL2;		00067000
	LDL(3)	.SAV1;		00067100



```

LUX      $S;                                00073300
LDL(1)  .ADRES1;                             00073400
LDA     +0(1);                                00073500
LDL(1)  .NM0;                                 00073600
STL(1)  .SAV2; % BRING THESE ELEMENTS INTO  00073700
% THE LOCATION OF THE ELEMENTS OF BASE.      00073800
      CLC(3);                                00073900
      SLIT(3) =ROUTE;EXCHL(3) $ICR;          00074000
      EWLXT(0)  $D0,SAL2;                    00074100
      STL(0)    .SAV2;                        00074200
      CLC(3);                                00074300
      SLIT(3) =ROUTE;EXCHL(3) $ICR;          00074400
      SKIP     #0;                            00074500
SAL2:  STA     TEMP1;                         00074600
      EWLXT(0)  $D0,SAL3;                    00074700
      LDL(1)    $C0;                          00074800
      CSUB(1)   $D1;                          00074900
      JUMP     SAL4;% FIND THE ROUTING DISTANCE FOR THE IN- 00075000
% DEX FR DIAGONAL PICK-UP OF THE ELEMENTS OF BASE. 00075100
SAL3:  LDL(1)  .NM0;                          00075200
SAL4:  STL(1)  .SAV2;                         00075300
      LDA     NUMB;                           00075400
      CLC(3);                                00075500
      SLIT(3) =ROUTE;EXCHL(3) $ICR;          00075600
      LDS     $A;                             00075700
      LDL(1)  .ADRES;                         00075800
      LDA     #0(1);                          00075900
      MLRN   TEMP1;                          00076000
      ADRN   TEMP;                           00076100
      STA   TEMP;                            00076200
      LDA   NUMB;                            00076300
      LDL(1)  .NM0;                          00076400
      STL(1)  .SAV2;                         00076500
      CLC(3);                                00076600
      SLIT(3) =ROUTE;EXCHL(3) $ICR;          00076700
% PICK UP THE ELEMENTS BELOW THE MAIN-DIAGONAL OF ANMAT(TR.) 00076800
% AD MULTIPLY THEM BY THE ELEMENTS ABOVE THE DIAGONAL 00076900
% OF BASE                                     00077000
      LDS     $A;                             00077100
      LDX     $S;                             00077200
      LDL(1)  .ADRES1;                       00077300
      LDA     +0(1);                          00077400
      LDL(1)  $C0;                            00077500
      CADD(1)  $D1;                          00077600
      STL(1)  .SAV2;                         00077700
      CLC(3); % MATCH UP THE ELEMENTS WITH THE 00077800
% EEMENTS OF BASE.                          00077900
      SLIT(3) =ROUTE;EXCHL(3) $ICR;          00078000
      STA     TEMP1;                         00078100
      LDA     NUMB;                          00078200
      CLC(3); % ADJUST THE INDEX FOR BASE PICK-UP 00078300
      SLIT(3) =ROUTE;EXCHL(3) $ICR;          00078400
      LDS     $A;                             00078500
      LDL(1)  .ADRES;                         00078600
      LDA     #0(1);                          00078700
      MLRN   TEMP1;                          00078800
      ADRN   TEMP;                           00078900
% NOW WE HAVE TO STORE THE ELEMENTS PROPERLY. 00079000
      STA     TEMP;                          00079100
      LDA     NUMB;                          00079200
      EWLXT(0)  $D0,SAL5;                    00079300

```

	STL(0)	.SAV2;	00079400
	CLC(3);		00079500
	SLIT(3)	=R00TE;EXCHL(3) &ICR;	00079600
SAL5:	LDS	&A;	00079700
	LDL(1)	.ADRES2;% ADRES2=ADDRESS OF TRASE A TEMP. MATRIX	00079800
	LDA	TEMP;	00079900
	STA	#0(1);	00080000
	EQLXF(0)	&00+1;	00080100
	JUMP	SAL6;	00080200
	LDA	NUMB;	00080300
	LDL(3)	.N;	00080400
	CSUR(3)	&00;	00080500
	STL(3)	.SAV2;	00080600
	CLC(3);		00080700
	SLIT(3)	=R00TE;EXCHL(3) &ICR;	00080800
	STA	TEMP1;	00080900
	LDA	TEMP;	00081000
	CLC(3);		00081100
	SLIT(3)	=R00TE;EXCHL(3) &ICR;	00081200
	TXET(0)	&00+SAL7;	00081300
	LDS	TEMP1;	00081400
	STA	#0(1);	00081500
	JUMP	SAL6;	00081600
SAL7:	CSHR(2)	0(0);	00081700
	CAND(2)	&05;	00081800
	CSHL(2)	0(0);	00081900
	LDEE1	&02;	00082000
	LDS	TEMP1;	00082100
	STA	#0(1);	00082200
SAL8:	SETE	F.NR.=E;	00082300
	SETE1	F.AND.=E;	00082400
	CLRA;		00082500
	STA	TEMP;	00082600
	TXETH(0)	.+1;	00082700
	JUMP	SAL9;	00082800
% REPLACE THE 2K-1,2K AND THE 2K,2K-1 POSITIONS OF TEMP-			00082900
% ORARY STORAGE MATRIX BY FOUR ZERUES.			00083000
	LIT(0)	=2,0,0,0;	00083100
	CRNTR(0)	24;	00083200
	CAND(0)	&02;	00083300
	CRNTL(0)	24;	00083400
	CLC(2);		00083500
SAL10:	LDL(1)	.FNB;	00083600
	CSHR(1)	0(0);	00083700
	CXOR(2)	&01;	00083800
	TXEFM(0)	.SAL10;	00083900
	LDEE1	&02;	00084000
	LDA	NUMB;	00084100
	LIT(0)	=1;	00084200
	ADM	&00;	00084300
	LDS	&A;	00084400
	CIMPC(2);		00084500
	CAND(2)	&05;	00084600
	LDEE1	&02;	00084700
	LDA	NUMB;	00084800
	SBM	&00;	00084900
	LDS	&A;	00085000
% REGISTER &S CONTAINS THE PATTERN 1,0,3,2,5,4,.....			00085100
	LDL(2)	.SPEC;	00085200
	LDEE1	&02;	00085300
	LIT(0)	=0,0;	00085400

```

LDA      $C0;                00085500
LDL(1)  .ADR$2;             00085600
STA     #0(1);              00085700
% THE MATRIX HAS NOW ITS FINAL FORM. WE NOW MAP THE
% MATRIX STORED IN TRASE INTO BASE. 00085800
LIT(0)  =1,0,0;            00085900
CRCTR(0) 24;                00086000
CADD(0)  $D3;               00086100
CRCTL(0) 24;                00086200
SAL11;  LDL(1) .ADR$;        00086300
LDL(3)  .ADR$2;             00086400
CADD(3)  $C0;               00086500
LDA     0(3);                00086600
CADD(1)  $C0;               00086700
STA     0(1);                00086800
TXEFM(0) .SAL11;           00086900
LDL(3)  .SAV1;              00087000
EXCHL(3) $TCR;             00087100
#####
ADDIT;  FILL;               00087200
% ADDIT CALCULATES THE SUM OF THE OFF- 00087300
% DIAGONALS SQUARE AND DIVIDES IT BY THE 00087400
% SUM OF THE DIAGONALS SQUARE.         00087500
STL(3)  .SAV1;              00087600
LIT(1)  =0;X                00087700
STL(1)  .SAV5;              00087800
LIT(0)  =0,1,0;X           00087900
CADD(0)  $D3;               00088000
CRCTL(0) 24;                00088100
LDL(3)  .SPEC;              00088200
SETE    E,OR,=E;           00088300
SETE1   E,AND,=E;          00088400
CLRA;  00088500
LDS     $A;                 00088600
LDDEE1  $C3;                00088700
ADI;    LDL(2) .ADRA;        00088800
CADD(2)  $C0;               00088900
LDA     0(2);                00090000
MLRN    $A;                 00090100
ADRN    $S;                 00090200
LDS     $A;                 00090300
TXEFM(0) .ADI;X            00090400
SETE    E,OR,=E;           00090500
SETE1   E,AND,=E;          00090600
LIT(0)  =1;                 00090700
ADI1;   LDS     $A;X        00090800
RTL     $S,0(0);X          00090900
LDS     $R;                 00091000
ADRN    $S;                 00091100
CADD(0)  $C0;               00091200
LIT(1)  =64;                00091300
EQLXF(0) $C1,ADI1;X       00091400
LDL(1)  .SAV5;              00091500
EQLXT(1) $D1,ADI2;        00091600
STA     TEMP;               00091700
CLRA;  00091800
LDS     $A;                 00091900
LDL(1)  .SPEC;X           00092000
LDDEE1  $C1;                00092100
LDX     NIMB;               00092200
LDL(2)  .ADRA;             00092300

```



```

LDA      #0(2);          00091600
MLRN     #A;            00091700
SETE     E.OR.-E;      00091800
SETE1    E.AND.-E;     00091900
LIT(0)   =1;           00092000
LDL(1)   .SAV5;        00092100
ALIT(1)  =1;           00092200
STL(1)   .SAV5;        00092300
JUMP     ADI1;         00092400
ADI2:    LDS           #A;            00092500
LDA      TEMP;         00092600
SBRN     #S;          SUBTRACT THE [I+1]. 00092700
LIT(0)   =0.0;        00092800
IMF      #0;          00092900
SETC(0)  I;           00093000
ONEST(0) .ADI3;       00093100
DVRN     #S;          CONVERGENCE FACTUR KSI FOUND. 00093200
STA      TEMP;        00093300
SLIT(0)  =TEMP;       00093400
LOAD(0)  #0;          00093500
STL(0)   .S;          00093600
ADI3:    LDL(3)       .SAV1;         00093700
EXCHL(3) #ICK;        00093800
*****
% M*****
TRASPUS; FILL;        00093900
%
%                   TRASPOS FINDS THE TRANSPUSE OF THE ANGLE- 00094000
%                   MATRIX BY JUST CHANGING THE SIGNS OF THE 00094200
%                   SUPER-DIAGONAL ELEMENTS. 00094300
%
STL(3)    .SAV1;      00094400
LDL(0)    .ANTON;    FIND THE INDEX FOR ABOVE ELEMENTS 00094500
LDFE1     #0;        00094600
LDA       NUMB;      00094700
LDS       =1;        00094800
ADM       #S;        00094900
STA       TEMP;LDA NUMB; 00095000
CSHR(0)   1;         00095100
RTL       #A.1;      00095200
LDEE1     #0;        00095300
LDA       #R;        00095400
STA       TEMP;      INDEX IS FOUND 00095500
LDL(0)    .SPEC;     00095600
LDEE1     #0;        00095700
LDS       TEMP;      00095800
LDL(3)    .ANRC;     00095900
LDA       #0(3);     00096000
CHSA;     00096100
STA       #0(3);     00096200
LDL(3)    .SAV1;EXCHL(3) #ICK; 00096300
*****
%
CONV;     FILL;      00096400
%
%                   CONV CHECKS FOR THE ABS.-VALUE OF THE OFF- 00096600
%                   DIAGONALS ELEMENTS LSS .BD(INDEX). 00096700
%
STL(3)    .SAV1;      00096800
LIT(0)    =0.2*0;    00096900
CAND(0)   #0;        00097000
CSUP(0)   #0;        00097100
CRRTL(0)  24;        00097200
CLC(1);   00097300
CON:      LDL(2)     .FNB;      00097400
CSHR(2)   0(0);     00097500
CEXOR(1)  #C;       00097600

```



```

TXEFM(0)  .CON;                                00097700
LDEE1     %C1;                                  00097800
LDA       NIJMB;                                00097900
RTL       %A,1;                                  00098000
CSHR(1)   1;                                    00098100
LDEE1     %C1;                                  00098200
LDS       %B;                                    00098300
CSHL(1)   1;                                    00098400
LDEE1     %C1;                                  00098500
LDL(2)    .ONE;                                  00098600
ADM       %C2;                                  00098700
LDS       %A;                                    00098800
LDL(1)    .SPEC;                                 00098900
LDEE1     %C1;                                  00099000
LDL(2)    .ADRA;                                  00099100
LDA       %D(2);                                 00099200
LDL(0)    .ZERO;                                  00099300
IAL       %C0;                                  00099400
SETE      I.AND.E;                               00099500
SETE1     E.AND.E;                               00099600
CHSA;                                           00099700
COMPCC(1);                                     00099800
LDEE1     %C1;                                  00099900
CLRA;                                           00100000
COMPCC(1);                                     00100100
LDEE1     %C1;                                  00100200
LDL(0)    .SAV6;                                  00100300
LDL(2)    .RD(0);                                 00100400
IAL       %C2;                                  00100500
SETC(2)   1;                                    00100600
STL(2)    .CONVE;                                00100700
LDL(3)    .SAV1;                                  00100800
EXCHL(3)  %ICR;                                  00100900
% *****00101000
TRPS:    FILL;                                  00101100
%                THIS PROCEDURE TRANSPOSES THE 00101200
% ORIGINAL MATRIX.                               00101300
        STL(3)  .SAV1;%  SAVE RETURN ADDRESS 00101400
%*****00101500
        LIT(0)  =0,1,0;%  SET UP LOOP-INDEX I 00101600
        CAD0(0)  %D3;                                00101700
        CRTL(0)  24;                                  00101800
        LDL(3)   .SPEC;                                00101900
        LDEE1    %C3;                                  00102000
TS3:;    LDA     NIJMB;                                00102100
        EQLXT(0) %D0,TS;                               00102200
        STL(0)   .SAV2;%  ROUTE D = I              00102300
        CLC(3);                                       00102400
        SLIT(3)  =ROUTE;EXCHL(3) %ICR;              00102500
TS:;    LDS     %A;                                    00102600
        LDX     %S;                                    00102700
        LDL(3)  .ADRB;%  ADDRESS OF EIGV          00102800
        LDA     %D(3);                                 00102900
        EQLXT(0) %D0,TS1;                             00103000
        LDL(3)  .N;                                    00103100
        CSJIB(3) %C0;                                  00103200
        STL(3)  .SAV2;%  ROUTE D= N-1             00103300
        CLC(3);                                       00103400
        SLIT(3)  =ROUTE;EXCHL(3) %ICR;              00103500
TS1:;   STA     TEMP;                                  00103600
        LDA     NIJMB;                                  00103700

```

```

EQLXT(0)  $00*TS2;                                00103800
CLC(3);                                         00103900
SLIT(3)  =RQJTL;EXCHL(3) $ICR;                   00104000
TS2:
LDS      $A;                                       00104100
LDA      TFMP;                                       00104200
LDL(3)   .ADRC;# ADDRESS OF ANMAT                 00104300
STA      #0(3);                                       00104400
TXETM(0) .+1;                                       00104500
JUMP     TS3;                                       00104600
LDL(3)   .SAV1;# RETURN TO THE OUTSIDE           00104700
EXCHL(3) $ICR;                                       00104800
*****
GERSH:  FILL;# GERSH FINDS A ROUND ON THE EIGEN-   00105000
%      VALUES USING THE GERSHGORING DISK         00105100
      STL(3)   .SAV1;                                       00105200
      SETE    E.ON.E;                                       00105300
      SETE1   E.AND.E;                                       00105400
      CLRA;                                       00105500
      LDS     $A;                                       00105600
*****
% THE RADIUS OF THE DISK ACC. TO GERSHGORIS THEOR4 CON- 00105700
% SISTS OF THE SUM OF THE ABS. VALUE OF THE OFF-DIAGONAL 00105800
% ELEMENTS LOCATED IN ROW I FOR WHICH A(I,I)=EIGVAL. I    00105900
% I.E. THE ROWSUM OF EACH INDIVIDUAL ROW HAS TO BE      00106000
% FOUND. THE CENTER OF THE DISK IS THE EIGENVALUE ITSELF 00106100
*****
      LIT(1)   =0.1,0;                                       00106200
      CAND(1)  $03;                                       00106300
      CRQTL(1) 24;                                       00106400
*****
% FINDING THE SUM OF THE ROW EQUALS FINDING THE SUM OF THE 00106500
% COLUMNS, SINCE THE MATRIX IS SYMMETRIC.              00106600
*****
GERS:  LDL(0)   .SPEC;                                       00106700
      CCR(0)   0(1);                                       00106800
      LOEE1   $00;                                       00106900
      LDL(2)   .ADRA;                                       00107000
      CAND(2)  $01;                                       00107100
      LDA     0(2);                                       00107200
      SAP;# TAKES ABS. VALUE, WHERE NECESSARY          00107300
      ADRN    $S;                                       00107400
      LDS     $A;                                       00107500
      TXEFM(1) .GERS;                                       00107600
      LDL(0)   .SPEC;                                       00107700
      LOEE1   $00;                                       00107800
      STA     GERSHG;# CONTAINS RADII. THE CENTER HAS   00107900
%      HAS TO BE READ FROM OUTPUT OF                 00108000
%      THE FINAL MATRIX, WHICH COMES                 00108100
%      OUT OF THE PROCEDURE EIGEN                     00108200
      LIT(0)   =1,GERSHG,GERSHG;                          00108300
      CRQTR(0) 24;                                       00108400
      CAND(0)  $03;                                       00108500
      CRQTL(0) 24;                                       00108600
      LIT(3)   =1,TACK=1,MESS0;                          00108700
      DISPLAY  $03,32;                                    00108800
      DISPLAYR $00,=16;                                    00108900
      LDL(3)   .SAV1;                                       00109000
      EXCHL(3) $ICR;                                       00109100
*****
EIGEN[ENTRY]: FILL;# SAVE THE CONTENT OF ACAR0,ACAR1.   00109200
%      AND AOB $0J2 THRU $0J9                        00109300
%                                                     00109400
%                                                     00109500
%                                                     00109600
%                                                     00109700
%                                                     00109800

```



```

LDL(0)      .ADRE;
EQLXT(0)    $D0+1; * CHECK WHETHER AN IDENTITY MATRIX HAS
JUMP        MAI1; * TO BE CREATED OR NOT
LDL(0)      .SPEC; * CREATE IDENTITY MATRIX.
LDEE1      $C0;
LDX        NIUMB;
LIT(1)      =1.;
LDA        $C1;
LDL(2)      .ADRE;
STA        *0(2); * EIGV=EIGENVECTOR=MATRIX INITIALLY THE
%           IDENTITY=MATRIX.
% *****
%           SKIP      .MAI1;
MAI2:      LDL(0)      .SAVE;
%           ALIT(0)    =1;
%           STL(0)     .SAVE;
% *****
%           LDL(0)     .7F4E;
%           EQLXT(0)   $D14+MAI1;
%           CADD(0)    $=0;
%           STL(0)     .7F4E;
% *****
MAI1:      CLC(3); * FIND MAXIMAL ROW=SUM AND ROW=INDEX .MAX.
%           SLIT(3)   =RHS; EXCHL(3) SICR;
%           LDL(0)    .MAX;
%           SETE      E.OR.=E;
%           SETE1     E.AND.E;
%           CLRA;
%           LDS       $A;
%           EULXF(0)  $D0+1; * IS ANY=ROW,COLUMN SHUFFLE NECESSARY.
%           JUMP      MAI3;
% *****
%           LDL(3)    .ADRA; * FIRST WE SHUFFLE BASE;
%           STL(3)    .ADRES;
%           CLC(3);
%           SLIT(3)   =ANYR; EXCHL(3) SICR;
%           LIT(3)    =1,MESS2=1,MESS1;
%           DISPLAY   $C3,C2;
%           LDL(3)    .ADRA; CSHL(3) $; RTPEM;
%           SETE      E.OR.=E;
%           SETE1     E.AND.E;
%           CLRA;
%           LDS       $A;
%           LDL(3)    .ADRE; * NOW WE SHUFFLE EIGV.
%           STL(3)    .ADRES;
%           CLC(3);
%           SLIT(3)   =ANYR; EXCHL(3) SICR;
% *****
MAI3:      LIT(0)     =0,1,0; * INNER LOOP, IN WHICH ALL TRANSFORMATIONS
%           TAKE PLACE.
%           CADD(0)   $D3;
%           CSUR(0)   $D1;
%           CRTL(0)   24;
%           STL(0)    .INDEX;
% *****
MAI3A:     CLC(3); * FIND THE ABSOLUTE VALUE OF THE SUPER-DIAG.
%           TO COMPARE THEM AGAINST A THRESHOLD FACTO=
%           SLIT(3)   =CONV; EXCHL(3) SICR;
%           SETE      E.OR.=E;
%           SETE1     E.AND.E;
%           CLRA;

```

```

LDS      $A;                                00122100
LDL(1)  .CONVE;                             00122200
ONESF(1) .+1;                                00122300
JUMP    MAI4;                                00122400
MAI5:   CLC(3);%                               FIND THE ANGLE-MATRIX. 00122500
        SLIT(3) =ANGLE;EXCHL(3) $ICR;        00122600
%       LIT(3) =1,MESS3=1,MESS2;            00122700
%       DISPLAY $C3,-32;                     00122800
%       LDL(3)  .ADRRC;CSHL(3) 6;WRTPM;      00122900
% *****
LDL(3)  .ADRRC;%                               GO THROUGH THE TRANSFORMATION BY MEANS 00123000
STL(3)  .ADRRES1;%                             OF MULTIPLICATION 00123100
LDL(3)  .ADRRA;                                00123300
STL(3)  .ADRRES;                                00123400
STL(3)  .ADRRES2;                                00123500
LIT(3)  =0;                                    00123600
STL(3)  .SAV9;                                00123700
SETE    E.DR.*E;                                00123800
SETE1   E.AN).E;                                00123900
CLRA;                                       00124000
LDS     $A;                                    00124100
CLC(3);                                       00124200
SLIT(3) =MULTPL;EXCHL(3) $ICR;              00124300
LDL(3)  .ADRRA;                                00124400
STL(3)  .ADRRES;                                00124500
STL(3)  .ADRRES2;                                00124600
LDL(3)  .ADRRC;                                00124700
STL(3)  .ADRRES1;                                00124800
SETE    E.DR.*E;                                00124900
SETE1   E.AN).E;                                00125000
CLRA;                                       00125100
LDS     $A;                                    00125200
CLC(3);                                       00125300
SLIT(3) =MULTPL;                                00125400
EXCHL(3) $ICR;                                00125500
CLC(3);%                               FIND TRANSPOSE OF ANGLE-MATRIX 00125600
SLIT(3) =TRASPDS;EXCHL(3) $ICR;            00125700
LDL(3)  .ADRRA;                                00125800
STL(3)  .ADRRES;                                00125900
LDL(3)  .ADRRA;                                00126000
STL(3)  .ADRRES2;                                00126100
LDL(3)  .ADRRC;                                00126200
STL(3)  .ADRRES1;                                00126300
SETE    E.DR.*E;                                00126400
SETE1   E.AN).E;                                00126500
CLRA;                                       00126600
LDS     $A;                                    00126700
CLC(3);                                       00126800
SLIT(3) =SAMUL;EXCHL(3) $ICR;              00126900
LIT(3)  =1,MESS4=1,MESS3;                    00127000
%       DISPLAY $C3,-32;                     00127100
%       LDL(3)  .ADRRA;CSHL(3) 6;WRTPM;      00127200
% *****
CLC(3);%                               FIND CONVERGENCE-FACTOR. 00127300
SLIT(3) =ADDIT;EXCHL(3) $ICR;              00127400
ONESF(0) .+1;                                00127500
JUMP    MAIEND;                                00127600
LDL(0)  .KSI;                                  00127700
LDA     $C0;                                    00127800
ILZ;                                       00127820
SETC(0) I;                                    00127840

```

	DNFSF(0)	.MAI6;		001278A0
	LIT(2)	=0.0000000000000001;		00127900
	LDA	%C2;		00128200
	LDL(2)	.S;		00128300
	LDS	%C2;		00128400
	MLRN	%S;		00128500
	STA	TFMP;		00128600
	SLIT(1)	=TEMP;		00128700
	LOAD(1)	%C1;		00128800
	STL(1)	.KS1;		00128900
MAI6:	LDL(0)	.ZFZE;		00129000
	LESSF(0)	%D14,+1;		00129100
	JUMP	MAI4;		00129200
	LDL(0)	.S;		00129300
	LDL(1)	.KS1;		00129400
	LDA	%C0;		00129420
	IAG	%C1;		00129440
	SETC(0)	T;		00129460
	DNFSF(0)	.+1;		00129480
	JUMP	MAIEND;		00129500
%	*****			00129600
MAI4:	LDL(0)	.INDEX;		00129700
	TXFFM(0)	.+1;		00129800
	JUMP	MAI2;		00129900
	STL(0)	.INDEX;		00130000
	LDL(3)	.ADRA;%	DO A 2ND-ROW*2ND-COLUMN SHUFFLE ON BASE	00130100
%			AND EIGENVECTOR=MATRIX;	00130200
%			BASE=SHUFLE	00130300
	STL(3)	.ADRES;		00130400
	SETE	E.OR.E;		00130500
	SETE1	E.AND.E;		00130600
	CLRA;			00130700
	LDS	%A;		00130800
	CLC(3);			00130900
	SLIT(3)	=SHUFL;EXCHL(3) %ICR;		00131000
%	LIT(3)	=1,ADRSAV=1,MESS4;		00131100
%	DISPLAY	%C3,%32;		00131200
%	LDL(3)	.ADRA;CSHL(3) 6;WRTPEM;		00131300
	LDL(3)	.ADRB;%	EIGV=SHUFLE	00131400
	STL(3)	.ADRES;%		00131500
	SETE	E.OR.E;%		00131600
	SETE1	E.AND.E;%		00131700
	CLRA;			00131800
	LDS	%A;		00131900
	CLC(3);SLIT(3)	=SHUFL;EXCHL(3) %ICR;		00132000
	JUMP	MAI3A;		00132100
MAIEND:	LDL(0)	.ADRE;		00132200
	EQLXF(0)	%D0,+1;		00132300
	JUMP	MAIENDE;		00132400
	LDL(3)	.ADRB;%	ADDRESS OF EIGV	00132500
	STL(3)	.ADRES;		00132600
	LDL(3)	.ADRE;		00132700
	STL(3)	.ADRES1;		00132800
	STL(3)	.ADRES2;		00132900
	LIT(3)	=0;		00133000
	STL(3)	.SAV9;		00133100
	CLC(3);			00133200
	SLIT(3)	=MULTPL;EXCHL(3) %ICR;		00133300
	CLC(3);			00133400
	SLIT(3)	=TRPS;EXCHL(3) %ICR;		00133500
				00133600

```

      LDL(3)      .ADRE;
      STL(3)      .ADRES;
      STL(3)      .ADRES2;
      LDL(3)      .ADRC;
      STL(3)      .ADRES1;
      CLC(3);
      SLIT(3) =MULTPL;EXCHL(3) $ICR;
      LIT(0)      =0,1,0;
%   ADD BASE AND ERROR MATRIX TO FIND THE CORRECT ROUND ON
%   THE EIGENVALUES
      CADD(0)     $D3;
      CRNTL(0)    24;
ATI:  LDL(3)      .ADKA;
      CADD(3)     $C0;
      LDL(2)      .ADRE;
      CADD(2)     $C0;
      LDA         0(3);
      ADRN        0(2);
      STA         0(3);
      TXFFM(0)    .AT;
MATIENDE:; CLC(3);
      SLIT(3)     =CFRSH;
      EXCHL(3)    $ICR;
      CLC(3);
      SLIT(3)     ADRSAY;
      BIN(3)      $D32;
      CLC(3);
      SLIT(3)     ADRSAY+8;
      LOAD(3)     $C0;
      ALIT(3)     1;
      LOAD(3)     $C1;
      LDL(3)      .RETR;%   TURN BACK TO THE OUTSIDE
      EXCHL(3)    $ICR;
#####
%#
%#
%#           END JACOBI/EIGEN
%#
%#
#####
END          EIGEN.

```

```

00133700;
00133800;
00133900;
00134000;
00134100;
00134200;
00134300;
00134400;
00134500;
00134600;
00134700;
00134800;
00134900;
00135000;
00135100;
00135200;
00135300;
00135400;
00135500;
00135600;
00135700;
00135800;
00135900;
00136000;
00136100;
00136200;
00136300;
00136400;
00136500;
00136600;
00136700;
00136800;
00136900;
00137000;
00137100;
00137200;
00137300;
00137400;
00137500;
00137600;
00137700;

```

APPENDIX C

The Subroutine EBERL, Jacobi-Like Method



```

REGTN                                00000100
FILL 128;                             00000200
*****                               00000300
% CALL = DEFINITION                    00000400
  DEFINE CALL &NAME(&PARAMETERS)=      00000500
    &IF &SIGN(&MFIELD(&NAME)) &THEN    00000600
      EXTERNAL &NAME; &FI             00000700
    &IF &EMPTY(&PARAMETERS) &THEN &ELSE 00000800
      REGTN BLOCK                      00000900
        BEGIN USE (63)                 00001000
          LIST: DATA &PARAMETERS     00001100
        END;                            00001200
        CLC(2);                         00001300
        SLIT(2) LIST;                  00001400
      END; &FI                          00001500
    CLC(3);                             00001600
    SLIT(3) &NAME;                     00001700
    EXCHL(3) $ICR;###;                 00001800
*****                               00001900
%DEFINE WRTPEM =                       00002000
%   LIT(1) =1,0,0;                     00002100
%   CADD(1) $C3;                       00002200
%   CRNTR(1) 24;                        00002300
%   CADD(1) $C3;                       00002400
%   CADD(1) $D3;                       00002500
%   CRNTR(1) 24;                       00002600
%   LIT(0) =0,1,0;                     00002700
%   CADD(0) $D3;                       00002800
%   CRNTR(0) 24;                       00002900
%   DISPLAYR $C1,-16;                  00003000
%   LIT(2) =64;                        00003100
%   CADD(1) $C2;                       00003200
%   CRNTR(1) 24;                       00003300
%   CADD(1) $C2;                       00003400
%   CRNTR(1) 24;                       00003500
%   TYFFM(0) ,-9;###;                  00003600
*****                               00003700
%#                                     00003800
%#          EBERL NORMALIZES A MTRIX   00003900
%#          AND, THUS, PREPARES IT FOR EIGEN- 00004000
%#          VALUE COMPUTATION SUCH AS JACOBI 00004100
*****                               00004200
.ZERO:  EQII  $D0;#  FIXED POINT ZERO.    00004200
.ONE:   EQII  $D1;#  FIXED POINT ONE.     00004300
.N:    EQII  $D2;#  ORDER OF MATRIX       00004400
.NMD:  EQII  $D3;#  N-1;                  00004500
.ENB:  EQII  $D4;#  10000000000000000000;#ENABLING ONE PE 00004600
.SPEC: EQII  $D5;#  ENABLING PATTERN FOR THE FIRST N PE S. 00004700
.ROUT: EQII  $D6;#  64-N, CONSTANT USED IN END AROUND ROUTING 00004800
.MAX:  EQII  $D7;#  ROW INDEX FOR MAX. VAL. FOUND IN RWSM. 00004900
.ADRS: EQII  $D8;#  ADDRESS SAVED HERE    00005000
.ADRS1:EQII  $D9;#  00005100
.ADRS2:EQII  $D10;#  00005200
.FINVAL1:EQII $D11;#  CONVERGENCE CHECK 00005300
.ANTIN: EQII  $D12;#  TURN-ON PATTERN FOR THE ANGLE ROUTINE 00005400
.ANTIN1:EQII  $D13;#  00005500
.G:    EQII  $D14;#  THE MAXIMAL OFF-DIAGONAL ELEMENT 00005600
.MAXR: EQII  $D15;#  THE ROWINDEX FOR ABOVE ELEMENT 00005700
.MAXC: EQII  $D16;#  THE COLUMNINDEX FOR ABOVE ELEM. 00005800
.AA:   EQII  $D17;#  OSCILLATION CONSTANTS 00005900
.AP:   EQII  $D18;#  00006000
.AC:   EQII  $D19;#  00006100
.INDEX:EQII  $D20;#  INNER LOOP COUNT IN MAIN PROGRAM. 00006200

```

```

.FINVAL: FQI      $D21;X  CONVERGENCE-FACTOR FOUND IN HYANG-ROUTINE. 00006300
,CONVE:  EQI      $D22;X  CHECK FOR SUPERDIAGS EQI 0. 00006400
.SAV1:   EQI      $D24;X  SAVE REGISTER. 00006500
.SAV2:   EQI      $D25;  00006600
.SAV3:   FQI      $D26;  00006700
.SAV4:   EQI      $D27;  00006800
.SAV8:   FQI      $D28;  00006900
.SAV5:   FQI      $D29;  00007000
.SAV9:   FQI      $D30;  00007100
.RFTIR:  EQI      $D31;X  RFTIR CONTAINS THE RETURN ADR. 00007200
%
%      ADDRESS TO LINK TO THE OUTSIDE 00007300
%      ADDRESS OF ORIGINAL MATRIX 00007400
%      ADDRESS OF EIGENVECTORMATR. 00007500
%      ANGLE MATRIX 00007600
%      TEMP. STORAGE MATRIX 00007700
%      ADDRESS OF THE ERROR MATRIX 00007800
%      ADDRESS OF MATRIX CARRIED OVER 00007900
%      TO THE JACOBI-SUBROUTINE FOR 00008000
%      PROPER COMPUTATION OF THE EIGEN- 00008100
%      VECTORS. 00008200
%      FACTOR FOR CHECK ON C(2I-1,2I-1) 00008300
%      ONE ROW OF SAVE-STORAGE IN PE-MEMORY 00008400
.TNMTW:  FQI      $D38;X
TEMP:    BLK      1;X
TEMP1:   BLK      1;
TEMP2:   BLK      1;
MANT:    BLK      1;
GERSHG:  BLK      1;
KAP2:    BLK      1;
KAP1:    BLK      1;
NUMB:    DATA    0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16; 00009100
          DATA    17,18,19,20,21,22,23,24,25,26,27,28,29,30; 00009200
          DATA    31,32,33,34,35,36,37,38,39,40,41,42,43,44; 00009300
          DATA    45,46,47,48,49,50,51,52,53,54,55,56,57,58; 00009400
          DATA    59,60,61,62,63;X  PE NUMBERING 00009500
%MFSS1:  DATA    "##### THE C - MATRIX#####"; 00009600
%MFSS2:  DATA    "##### THE MATRIY BASE AFTER SHUFFLE####"; 00009700
%MFSS3:  DATA    "#### AFTER THE 2-ND TRANSFORMATION ####"; 00009800
ADRSAV:  WDS      10; 00009900
% *****
ROUTE:   FILL;X  ROUTE-ROUTINE FOR N<64 00010100
          STI(3)  .SAV3;X  SAVE RETURN ADDRESS. 00010200
          LDI(3)  .SAV2;X  LOAD ROUTING DISTANCE. 00010300
          RTI     $A,0(3);X $A IS ASSUMED TO CONTAIN THE ELEMENTS 00010400
          %      TO BE ROUTED. 00010500
          LDI(3)  .SPEC;X 00010600
          LDEF1   $C3; 00010700
          LDA     $R; 00010800
% ***** 00010900
%      THIS PART IS THE HEART OF THE MATTER. IT ELIMINATES 00011000
%      THE ELEMENTS WHICH MIGHT 00011100
%      HAVE COME INTO THE FIRST 00011200
%      D PFS BY ROUTING A DISTANCE 00011300
%      D, BY CLEARING RGA IN THESE PFS 00011400
          ISI     $C3;X 00011500
          SFTF   I.AND.F;X 00011600
          SETF1  E.AND.E; 00011700
          CLRA; 00011800
% ***** 00011900
          LDI(3)  .ROUT; 00012000
          RTI     0(3);X  ROUTE THE ELEMENTS WHICH MOVED PAST THE 00012100
%      BOUNDARY N. 64-N INTO THE FIRST D PFS OF RGR AND THEN 00012200
%      LOAD RGA FROM RGR IN THOSE FIRST D PFS. 00012300

```

```

LDA      $R;                                00012400
LDI(3)  .SPFC;                              00012500
COMPC(3);                                  00012600
LDFF1   $C3;                                00012700
CLRA;                                        00012800
COMPC(3);                                  00012900
LDFF1   $C3;                                00013000
LDI(3)  .SAV3; AND THE RESULT WILL BE IN $A PROPERLY 00013100
FXCHL(3) $ICR; ROUTED, THEN RETURN.                00013200
*****
ROTAR:   FILL;                               00013300
*****
* ADJUST ACAR1 FOR AN END-AROUND SHIFT, RIGHT FOR D LSS 64 00013400
STI(3)  .SAV3; SAVE RETURN ADDRESS                00013500
LDI(1)  .SAV8; PATTERN THAT NEEDS ADJUSTMENT     00013600
CROTL(1) 0(0); ACARO CONTAINS D                  00013700
LDI(2)  .SPFC;                                    00013800
CSHR(2)  0(0);                                    00013900
CAND(2)  $C1;                                      00014000
CAND(2)  $D5;                                      00014100
CEXOR(1) $C2;                                      00014200
LDI(3)  .N;                                        00014300
CROTL(1) 0(3);                                    00014400
CEXOR(1) $C2;                                      00014500
STI(1)  .SAV8;                                      00014600
LDI(3)  .SAV3;                                      00014700
FXCHL(3) $ICR;                                      00014800
*****
* C*****
ROTAR:   FILL;                               00014900
*****
* ADJUST ACAR1 FOR AN END-AROUND SHIFT, LEFT FOR D LSS 64 00015000
STI(3)  .SAV3;                                    00015100
LDI(1)  .SAV8;                                    00015200
CROTL(1) 0(0);                                    00015300
LDI(3)  .SPFC;                                    00015400
CSHL(3)  0(0);                                    00015500
CAND(3)  $C1;                                      00015600
CAND(3)  $D5;                                      00015700
CEXOR(1) $C3;                                      00015800
LDI(2)  .ROUT;                                     00015900
CSHL(1)  0(2);                                     00016000
CEXOR(1) $C3;                                      00016100
STI(1)  .SAV8;                                      00016200
LDI(3)  .SAV3;                                      00016300
FXCHL(3) $ICR;                                      00016400
*****
*****
FNDMX:   FILL;                               00016500
*
* FIND THE LARGEST OFF-DIAGONAL 00016600
* ELEMENT AND RETURN IT AS WELL 00016700
* AS THE ROW- AND COLUMN-INDEX. 00016800
* SAVE RETURN ADDRESS 00016900
STL(3)  .SAV1;
*****
* PART 1: FIND THE LARGEST ELEMENT WITHIN ONE ROW 00017000
* IIT(0) =0,1,0; SET UP THE LOOP FOR THE ROW 00017100
* PICK-UP 00017200
CADD(0)  $D3;                                      00017300
CSUR(0)  $D1;                                      00017400
CROTL(0)  24;                                       00017500
IDL(1)  .SPFC;                                       00017600
STL(1)  .SAV9;                                       00017700
FND1: SETF F.OR.-F; CLEAN OUT $S & $A 00017800
SFTF1   E.AND.E; 00017900
CLRA; 00018000

```

	LDS	\$A;		00018500
	LDL(1)	.SAV9;		00018600
	CCP(1)	0(0);X	COMPL. THE T = \$C0 -TH RTT	00018700
*	STL(1)	.SAV9;X	ONLY THE UPPER TRIANGULAR	00018800
			OFF-DIAG. ELEM. PARTICIPATE	00018900
	LDL(2)	.ADRCM;X	FFETCH ADDRESS OF CMAT	00019000
	CADD(2)	\$C0;		00019100
	LDF1	\$C1;		00019200
	LDA	0(2);X	LOAD ROW OF CMAT	00019300
	SAP;X		TAKE APS. VAL. WHERE NCESS.	00019400
	SFTF	F.OR.=F;		00019500
	SFTF1	E.AND.F;		00019600
	LDS	\$A;X	FIND LARGEST ELEMENT IN THE	00019700
	STA	TEMP2;X	ROW LOADED	00019800
	IIT(1)	=1;		00019900
FND:	RTL	\$S,0(1);		00020000
	LDS	\$R;		00020100
	IAL	\$S;		00020200
	SFTF	I.AND.F;		00020300
	SFTF1	F.AND.F;		00020400
	LDA	\$S;		00020500
	SFTF	F.OR.=F;		00020600
	SFTF1	F.AND.E;		00020700
	LDS	\$A;		00020800
	CADD(1)	\$C1;		00020900
	IIT(3)	=64;		00021000
	FQLXF(1)	\$C3,FND;X	FND FIND MAXIMAL ELEMENT	00021100
	LDL(1)	.ENB;		00021200
	CSHR(1)	0(0);		00021300
	LDF1	\$C1;		00021400
	STS	TEMP;X	STORE VALUE JUST FOUND	00021500
	LDL(2)	.SAV9;		00021600
	LDF1	\$C2;X	FIND OUT WHICH COLUMN THE	00021700
	LDA	TEMP2;X	ELEMENT CAME FROM	00021800
	IAL	\$S;		00021900
	SFTF	=J.AND.E;		00022000
	SFTC(2)	F;		00022100
	LEADD(2);			00022200
	IIT(3)	=77;8;		00022300
	LDF1	\$C1;		00022400
	CAND(2)	\$C3;		00022500
	LDA	\$C2;		00022600
	STA	TEMP1;X	COLUMNINDEX STORED	00022700
	TXFTM(0)	,+1;		00022800
*			TEMP CONTAINS THE MAXIMAL	00022900
	JUMP	FND1;		00023000
*			VALUES OF EACH ROW. TEMP1 CONTAINS THE COLUMNINDEXES.	00023100
*			NOW WE HAVE TO FIND THE MAXIMAL VALUE OF TEMP=MAXIMAL	00023200
*			OFF-DIAGONAL ELEMENT.	00023300
	SFTF	F.OR.=F;		00023400
	SFTF1	E.AND.F;		00023500
	CLRA;			00023600
	LDS	\$A;		00023700
	LDL(1)	.SPFC;		00023800
	LDI(2)	.NMD;		00023900
	CCP(1)	0(2);X	TURN OFF THE (N-1)TH PE	00024000
	LDF1	\$C1;		00024100
	LDA	TEMP;		00024200
	SFTF	F.OR.=F;		00024300
	SFTF1	E.AND.E;		00024400
	LDS	\$A;		00024500

```

FND2:  LIT(2)      =1;                                00024600
      RTL        $$,0(2);                             00024700
      LDS        $R;                                   00024800
      IAL        $$;                                   00024900
      SFTF      I.AND.F;                               00025000
      SFTF1     E.AND.F;                               00025100
      LDA       $$;                                    00025200
      SFTF      F.OR.-F;                               00025300
      SFTF1     F.AND.F;                               00025400
      LDS       $A;                                    00025500
      CADD(2)   $C2;                                   00025600
      LIT(3)    =64;                                   00025700
      FOLYF(2)  $C3,FND2;X   THE LARGEST IS FOUND    00025800
X  $$ AND $A CONTAIN IT.                                00025900
      LIT(0)    =0;                                    00026000
      IMF       $C0;X   CHECK FOR THE LARGEST FLEM. TO BE ZERO 00026100
      SFTC(0)   I;                                     00026200
      ONE$F(0)  ,+1;                                   00026300
      JUMP      FND3;                                   00026400
      LDFF1     $C1;                                   00026500
      LDA       TEMP;X   FIND ROW- AND COLUMN-INDEX BY      00026600
X                                     COMPARING TEMP AGAINST $$ 00026700
      IAL       $$;                                    00026800
      SFTF      -I.AND.F;                               00026900
      SFTC(0)   E;                                     00027000
      IFAND(0);  00027100
      LIT(3)    =77;8;                                  00027200
      CADD(0)   $C3;                                   00027300
      STL(0)    ,MAXR;X   ROWINDEX FOUND                00027400
      SLIT(1)   =TEMP;                                  00027500
      CADD(1)   $C0;                                   00027600
      LDAN(1)   $C3;                                   00027700
      STL(3)    ,G;X   STORE THE LARGEST ELEMENT        00027800
      CLC(1);   00027900
      SLIT(1)   =TEMP1;                                 00028000
      CADD(1)   $C0;                                   00028100
      LDAN(1)   $C3;                                   00028200
      STL(3)    ,MAXC;X   THE COLUMNINDEX FOR ABOVE FLEM. 00028300
FND3:  LDL(3)   ,SAV1;                                  00028400
      FXCH(3)  $ICR;                                   00028500
X#####
MLTRPS:  FILL;                                        00028600
X                                     MULTIPLY TWO MATRICES, ONE OF 00028700
X                                     WHICH IS THE TRANSPOSE OF THE 00028800
X                                     OTHER, WITHOUT ACTUALLY TRANS- 00028900
X                                     POSING THE MATRIX.          00029000
X                                     00029100
      STI(3)    ,SAV1;X   SAVE RETURN-ADDRESS           00029200
      LIT(0)    =0,1,0;X   SET UP LOOP FOR ROW-FETCH OF BASE 00029300
      CADD(0)   $D3;                                   00029400
      CROTL(0)  24;                                       00029500
MT1:    SFTF      F.OR.-F;                               00029600
      SFTF1     F.AND.F;                               00029700
      CLRAX;X   CLEAN UP $A & $$                       00029800
      LDS       $A;                                    00029900
      STA       TEMP;                                  00030000
      LDI(2)    ,SPEC;                                  00030100
      LDFF1     $C2;                                   00030200
      LIT(2)    =0,1,0;                                  00030300
      CADD(2)   $D3;X   INNER LOOP FOR PICK-UP OF TRANS- 00030400
      CROTL(2)  24;X   POSE ELEMENTS OF BASE          00030500
MT:     LDA       NUMR;X   MAKE PROPER INDEXING        00030600

```



```

STI(2)      .SAV2;                                00030700
CLC(3);    00030800
SLIT(3) =ROUTE;FXCHL(3) $ICR;                    00030900
LDS        $A;                                    00031000
LDY        $S;                                    00031100
LDI(1)     .ADRES;                                00031200
CAND(1)    $C0;X   PIXK UP ROW I=$C0 OF BASE     00031300
LDA        0(1);                                  00031400
LDI(1)     .ADRES;                                00031500
MLRN       *0(1);X  MULTIPLY BY TRANSP. FIEM. OF BASE 00031600
ADR        TEMP;X  ADD PARTIAL PRODUCT           00031700
CLC(3);    00031800
LDI(1)     .DNF;X  00031900
STI(1)     .SAV2;                                00032000
SLIT(3) =ROUTE;FXCHL(3) $ICR;                    00032100
STA        TEMP;                                  00032200
TXFFM(2)   ,MT;                                   00032300
LDA        TEMP;                                  00032400
LDI(3)     .ADRES1;                                00032500
CAND(3)    $C0;                                   00032600
STA        0(3);                                  00032700
TXFTM(0)   ,+1;                                   00032800
JUMP       MT1;                                    00032900
LDI(3)     .SAV1;                                00033000
FXCHL(3)   $ICR;                                  00033100
#####
TRPS:      FILL;                                  00033300
%
%           THIS PROCEDURE TRANSPPOSES
%           THE ORIGINAL MATRIX, TO PREPARE
%           IT FOR THE MULTIPLICATION
%           (A)TXA=A(TRANSP.)X(A(TRP.)TRP.)
%           A(TRP.) IS TEMPLY STORED IN ANMAT
%
STI(3)     .SAV1;X  SAVE RETURN ADDRESS           00033900
#####
IIT(0)     =0,1,0;                                00034100
%           SET UP LOOP=INDFY I
%
CAND(0)    $D3;                                    00034300
CRTL(0)    24;                                     00034400
LDI(3)     .SPFC;                                  00034500
LDF1       $C3;                                    00034600
TS3:      LDA        NUMR;                          00034700
EQIXT(0)   $D0,TS;                                 00034800
STI(0)     .SAV2;X  ROUTE D=I;                    00034900
CLC(3);    00035000
SLIT(3) =ROUTE;FXCHL(3) $ICR;                    00035100
TS:      LDS        $A;                              00035200
LDY        $S;                                      00035300
LDI(3)     .ADRA;X  ADDRESS OF BASE               00035400
LDA        *0(3);                                  00035500
EQIXT(0)   $D0,TS1;                                00035600
LDI(3)     .N;                                       00035700
CSHP(3)    $C0;                                       00035800
STI(3)     .SAV2;X  ROUTE D=N-I                   00035900
CLC(3);    00036000
SLIT(3) =ROUTE;FXCHL(3) $ICR;                    00036100
TS1:     STA        TEMP;                            00036200
LDA        NUMR;                                    00036300
EQIXT(0)   $D0,TS2;                                00036400
CLC(3);    00036500
SLIT(3) =ROUTE;FXCHL(3) $ICR;                    00036600
SKIP      ,0;                                       00036700

```

```

TS2:   LDS          SA;                                00036800
       LDA          TEMP;                             00036900
       LDI(3)      .ADR0;X  ADDRESS OF TBASE         00037000
       STA          #0(3);                            00037100
       TXFFM(0)    ,TS3;                              00037200
       LDI(3)      .SAV1;X  RETURN                  00037300
       FXCHL(3)    $ICR;                              00037400
#####
SHUF:  FILL;                                          00037500
#####
%
%           THIS PROCEDURE FALLS INTO TWO PARTS     00037600
%           PART ONE BRINGS ROW MAXR INTO THE PLACE  00037700
%           OF THE FIRST ROW. THE SECOND PART      00037800
%           BRINGS THE COLUMN MAXC=MAXR INTO THE   00037900
%           PLACE OF THE SECOND COLUMN. THIS      00038000
%           PROCEDURE IS ENTERED ONLY IF MAXR+MAXC  00038100
%           NEQ 1.                                  00038200
%
%           STI(3)  .SAV1;X  SAVE RETURN ADDRESS    00038300
#####
% PART ONE:
%
%           LDI(3)  .SPEC;                                00038400
%           LDFF1   $C3;                                  00038500
%           LDI(0)  .MAXR;X  IF MAXR=0, THEN THE LARGEST  00038600
%           ELEMENT IS ALREADY IN ROW 0.            00038700
%           FOLLOWS WE JUST NEED TO ADJUST THE     00038800
%           JUMP    PAR2;                                  00038900
%           LDI(1)  .N;X      COLUMN                 00039000
%           CSHP(1)  $C0;                                  00039100
%           STI(1)  .SAV2;X  ROUTE DISTANCE D=N-MAXR  00039200
%           IIT(0)  =0,1,0;                              00039300
%           CADD(0) $D3;                                  00039400
%           CRTL(0) 24;                                  00039500
%
%           LDI(1)  .ADRES;X  CONTAINS THE ARESS OF THE  00039600
%           MATRIX UNDER CONSIDERATION             00039700
%           CADD(1) $C0;X  ROUTE ALL ROWS OF MATRIX    00039800
%           LDA     0(1);                                00039900
%           CLC(3);X  GO TO THE ROUTE PROCEDURE      00040000
%           SLIT(3) =ROUTE;FXCHL(3) $ICR;           00040100
%           STA     0(1);                                00040200
%           TXFFM(0) ,PAR1;                            00040300
%           NOW REARRANGE THE MATRIX SO THAT THE LARGEST  00040400
%           ELEMENT IS IN THE FIRST ROW.           00040500
%           IIT(0)  =0,1,0;                              00040600
%           CADD(0) $D15;X  LIMIT IS MAXR-1, STORED IN $D15  00040700
%           CSHP(0) $D1;                                  00040800
%           CRTL(0) 24;                                  00040900
%
%           LDI(1)  .ADRES;                              00041000
%           CADD(1) $C0;                                  00041100
%           LDA     0(1);                                00041200
%           LDI(1)  .ADR0;X  ADDRESS OF TBASE         00041300
%           CADD(1) $C0;                                  00041400
%           STA     0(1);X  STORE ROWS TEMPORARILY IN TBASE  00041500
%           TXFFM(0) ,PARA1;                            00041600
%           IIT(0)  =0,1,0;X  LOOP TO FETCH ROWS MAXR TO N-1  00041700
%           CADD(0) $D3;                                  00041800
%           CRTL(0) 24;                                  00041900
%           CADD(0) $D15;                                00042000
%
%           LDI(1)  .ADRES;                              00042100
%           CADD(1) $C0;                                  00042200
%           LDA     0(1);                                00042300
%           CSHP(1) $D15;                                00042400
%           STA     0(1);                                00042500
%           STA     0(1);                                00042600
%           STA     0(1);                                00042700
%           STA     0(1);                                00042800

```

```

TXFFM(0) ,PARA2; 00042900
LIT(0) =0,1,0; 00043000
CADD(0) $D15; 00043100
CSHR(0) $D1; 00043200
CRTL(0) 24;X THIS LOOP FETCHES THE ROWS OF 00043300
X TRACE AND STORES THEM BACK INTO 00043400
X THE MATRIX UNDER CONSID. 00043500

LDI(3) ,N;X 00043600
CSHR(3) $D15;X $C3 CONTAINS ROW INDEX FOR BASE 00043700
PARA3: LDI(1) ,ADDR; 00043800
CADD(1) $C0; 00043900
LDA 0(1); 00044000
LDI(1) ,ADDR; 00044100
CADD(1) $C3; 00044200
CADD(1) $C0; 00044300
STA 0(1); 00044400
TXFFM(0) ,PARA3; 00044500
***** 00044600
X PART TWO: 00044700
PAR2: LDI(0) ,MAXC;X FIND OUT WHERE THE COLUMN, 00044800
CSHR(0) $D15;X WENT TO AFTER ABOVE REARR. 00044900
FOIYF(0) $D1,+1; 00045000
JUMP OVER1; 00045100
X IF THE TEST IS SATISFED, THEN THE COLUMN IS IN THE, 00045200
X POSITION OF THE SECOND COLUMN, BECAUSE OF THE REARR. 00045300
X IN PART ONE. 00045400
STI(0) ,SAV9;X SAVE THE NEW COLUMN INDEX 00045500
LDI(1) ,ADDR;X FETCH ROW MAXC=MAXR 00045600
CADD(1) $C0; 00045700
LDA 0(1); 00045800
STA TEMP; 00045900
LIT(2) =-1,1,0; 00046000
CADD(2) $C0; 00046100
CSHR(2) $D1; 00046200
PART2: LDI(3) ,ADDR;X PULL ROWS ABOVE ROW MAXC= 00046300
CADD(3) $C2;X MAXR DOWN BY ONE 00046400
LDA 0(3); 00046500
STA 0(1); 00046600
CSHR(1) $D1; 00046700
TXFFM(2) ,PART2; 00046800
LDA TEMP; 00046900
STA 0(1);X ROW MAXC=MAXR IS NOW IN PLACE 00047000
X OF THE SECOND ROW. 00047100
***** 00047200
X REARRANGE THE COLUMNS NEXT. BUT SKEW FIRST FOR BETTER ACCESS 00047300
LIT(0) =0,1,0; 00047400
CADD(0) $D3; 00047500
CRTL(0) 24; 00047600
SETF E.OR.-E; 00047700
SETF1 E.AND.E; 00047800
CLRAB; 00047900
LDS $A; 00048000
LDI(3) ,SPFC; 00048100
IDFF1 $C3; 00048200
PART3: LDI(1) ,ADDR; 00048300
CADD(1) $C0; 00048400
LDA 0(1); 00048500
STI(0) ,SAV2; 00048600
CLC(3); 00048700
SLIT(3) =ROUTE;FXCHI(3) $ICR; 00048800
STA 0(1); 00048900

```



	TXFFM(0)	,PART3;	00049000
	LDI(0)	.SAV9;X MAXC-MAXR	00049100
	STI(0)	.SAV2;	00049200
	LIT(0)	=-1,1,0;	00049300
	CADD(0)	\$D30;	00049400
	CSUR(0)	\$D1;	00049500
	IDA	NUMB;X ADJUST INDEX FOR COL.-FETCH	00049600
	CLC(3);		00049700
	SLIT(3)	=ROUTF;EXCHL(3) \$ICR;	00049800
	LDS	\$A;	00049900
	IDX	\$S;	00050000
	LDI(1)	.ADRES;	00050100
	LDA	*0(1);X FETCH COLUMN #(MAXC-MAXR)	00050200
PART4:	STA	TEMP;	00050300
	LDI(2)	.NM0;	00050400
	STI(2)	.SAV2;	00050500
	LDA	\$X;X ADJUST PICK-UP INDEX	00050600
	CLC(3);X	FOR ROWS .SAV9-1 TO 1	00050700
	SLIT(3)	=ROUTF;FXCHL(3) \$ICR;	00050800
	LDS	\$A;	00050900
	STS	TEMP;	00051000
	LDA	*0(1);	00051100
	LDI(2)	.ONE;	00051200
	STI(2)	.SAV2;	00051300
	CLC(3);		00051400
	SLIT(3)	=ROUTF;FXCHL(3) \$ICR;	00051500
	STA	*0(1);	00051600
	LDX	TEMP;	00051700
	TXFFM(0)	,PART4;	00051800
	LDA	TEMP;X FOR MAXC-MAXR IN TO THE SECOND	00051900
	LDI(2)	.N;	00052000
	CSUR(2)	\$D30;	00052100
	CADD(2)	\$D1;	00052200
	STI(2)	.SAV2;	00052300
	CLC(3);		00052400
	SLIT(3)	=ROUTF;FXCHI(3) \$ICR;	00052500
	STA	*0(1);	00052600
	LIT(0)	=-1,1,0;	00052700
	CADD(0)	\$D2;	00052800
	LDI(1)	.ADRES;	00052900
	EQUXT(0)	\$D2,PART5;	00053000
PART4:	STI(0)	.SAV2;X UNSKEW MATRIX	00053100
	LDA	0(1);	00053200
	CLC(3);		00053300
	SLIT(3)	=ROUTF;FXCHL(3) \$ICR;	00053400
	STA	0(1);	00053500
PART5:	ALIT(1)	=1;	00053600
	TXFFM(0)	,PART6;	00053700
OVER1:	LDI(3)	.SAV1;X RETURN TO MAIN PROGRAM	00053800
	FXCHL(3)	\$ICR;	00053900
*****			
MULTPL:	FI1;		00054000
X		MULTIPLICATION OF TWO MATRICES. THE	00054200
X		SET-UP IS SUCH THAT THE ADDRESS OF	00054300
X		THE MATRIX IS TREATED AS A VARIABLE	00054400
	STI(3)	.SAV1;X	00054500
	LIT(1)	=0,1,0;	00054600
	CADD(1)	\$D3;	00054700
	CRNTL(1)	24;	00054800
	LDI(3)	.SPFC;	00054900
	LDFI	\$C3;	00055000

MUL1:	LDS	NUMR1X	PE-NUMBERS	00055100
	LDI(3)	.ADRF51X	=AMATRIX-BASE	00055200
	CADD(3)	\$C1		00055300
	LDA	0(3)1X	LOAD ROW OF AMATRIX	00055400
	LIT(0)	=0,1,0		00055500
	CADD(0)	\$D3		00055600
	CRCTL(0)	24		00055700
	LDS	=0		00055800
MUL11:	LDI(3)	.ADRF511X	RMATRIX-BASE	00055900
	MLRN	*0(3)1X	MULTIPLY RMATRIX	00056000
	ADRN	\$S1X	FORM PARTIAL SUM	00056100
	STA	MANT		00056200
	SETF	F.OR.=F		00056300
	SETF1	F.AND.F		00056400
	CLRA			00056500
	LDS	\$A		00056600
	LDI(3)	.SPFC		00056700
	LDF1	\$C3		00056800
	LDA	\$R1		00056900
	FOIXF(0)	\$D1,MUL31X	IF \$C0=1 CHECK IF .SAV9=1	00057000
	LDI(3)	.SAV9		00057100
	FOIXF(3)	\$D1,MUL3		00057200
	CADD(0)	\$D3		00057300
	CSHR(0)	\$D1		00057400
	CSHR(0)	\$D1		00057500
	LDI(2)	.NMD		00057600
	CSHR(2)	\$D1		00057700
	SKIP	,MUL4		00057800
MUL3:	LDI(2)	.DNF		00057900
MUL4:	STI(2)	.SAV2		00058000
	CLC(3)			00058100
	SLIT(3)	=ROUTF1FXCHI(3) \$ICR		00058200
	STA	TEMP		00058300
	SETF	E.OR.=E		00058400
	SETF1	E.AND.F		00058500
	CLRA			00058600
	LDS	\$A		00058700
	LDI(3)	.SPFC		00058800
	LDF1	\$C3		00058900
	LDA	\$X		00059000
	CLC(3)			00059100
	SLIT(3)	=ROUTF1FYCHI(3) \$ICR		00059200
	LDS	\$A		00059300
	LDS	\$S		00059400
	SETF	F.OR.=F		00059500
	SETF1	F.AND.F		00059600
	CLRA			00059700
	LDS	\$A		00059800
	LDI(3)	.SPFC		00059900
	LDF1	\$C3		00060000
	LDS	MANT		00060100
	LDA	TEMP		00060200
	TXIFM(0)	,+1		00060300
	JUMP	MUL1		00060400
	LDI(2)	.ADRN		00060500
	CADD(2)	\$C1		00060600
	STC	0(2)		00060700
	TXIFM(1)	,+1		00060800
	JUMP	MUL		00060900
	LIT(0)	=1,0,0		00061000
	CRCTR(0)	24		00061100

```

CADD(0)    $D3;                                00061200
CRTL(0)    24;                                00061300
MUL2:     LDI(3)    .ADRES2;                   00061400
          CADD(3)    $C0;                       00061500
          LDI(2)    .ADR0;                       00061600
          CADD(2)    $C0;                       00061700
          LDA      0(2);                          00061800
          STA      0(3);                          00061900
          TXFRM(0)  ,MUL2;                       00062000
          LDI(3)    .SAV1;                       00062100
          FXCHL(3)  $ICR;                       00062200
% *****00062300
MULSA:    FILL;                                00062400
%
%           MULTIPLY ANMAT(TR.) X (BASE X ANMAT); 00062500
% IF ANMAT(TR.) CONSISTS OF COSH & SINH, WE USE THESE TWO 00062600
% VALUES ONLY TO MULTIPLY ROW I AND ROW I+1, WHERE 00062700
% I:=0,2,4,...,N-2. THEN WE ADD ROW I & ROW I+1 TO 00062800
% FORM THE NEW ROW I OF BASE. 00062900
%           IF ANMAT(TR.) CONSISTS OF COS & SIN, THEN THE 00063000
% 2X2 MATRICES, GOING DOWN THE DIAGONAL, WILL BE DIFFERENT 00063100
% SO THAT THE COS & SIN HAVE TO BE PULLED AS I INCREASES. 00063200
% FURTHER EXPLANATIONS IN THE PROGRAM. 00063300
%#####00063400
          STI(3)    .SAV1; SAVE THE RETURN ADDRESS 00063500
          IIT(0)    =0,2,0; 00063600
          CADD(0)    $D3; 00063700
          CSUP(0)    $D1; 00063800
          CRTL(0)    24; 00063900
% SETTING UP THE LOOP FOR ROW INDEX I; 00064000
          LDI(1)    .ADRES1; ADRS1=ADDRESS OF ANMAT(TR.) 00064100
          CSHL(1)    6; 00064200
MLS:      FRIXT(0)  $D0,MIS2; 00064300
% CHECK TO SEE WHICH COS & SIN HAS TO 00064400
% BE PICKED IF $C0=I IS NOT EQUAL TO ZERO 00064500
MLS4A:   CADD(1)    $D1; THE CONTENT OF $C1=ADDRESS 00064600
% OF ANMAT(TR.) IS NOT DESTROYED WITHIN THIS LOOP. 00064700
% IT IS CHANGED AIRIGHT TO FIND THE PROPER ELEMENTS 00064800
% OF ANMAT(TR.). 00064900
          IIT(2)    =128; 00065000
% GO TWO ROWS DOWN IN ANMAT(TR.) 00065100
          CADD(1)    $C2; FIND THE NEW ADDRESS FOR ANMAT 00065200
MLS2:    LOAD(1)    $C2; SAV9#0 AND $C0=0 00065300
          STI(2)    .SAV2; SAV2=COSH OR COS 00065400
          CADD(1)    $D1; 00065500
          LOAD(1)    $C2; 00065600
          STI(2)    .SAV3; SAV3=SIN OR SINH 00065700
MLS1:    LDI(2)    .ADRES; ADRS=ADDRESS OF BASE 00065800
          CADD(2)    $C0; $C2=ADR. OF ROW I OF BAE 00065900
          LDI(3)    .SPFC; 00066000
          LDEF1     $C3; 00066100
          LDA      0(2); 00066200
          LDI(3)    .SAV2; 00066300
          MLRN     $C3; ROW I OF BASE TIMES COSH 00066400
% OR COS DEPENDING ON CONTENT OF .SAV2 00066500
          STA      TEMP; 00066600
          CADD(2)    $D1; $C2=ADR. OF ROW I+1 OF BASE 00066700
          LDA      0(2); 00066800
          LDI(3)    .SAV3; 00066900
          MLRN     $C3; ROW I+1 TIMES SINH OR SIN 00067000
          ADRN     TEMP; 00067100
          LDI(2)    .ADRES2; ADRS2=ADR. OF TRASF 00067200

```

```

      CADD(2)   $C0;
      STA      0(2);
      LDI(2)   .SAV9;
      FOIXT(2) $D0,MLS3;
      LDI(2)   .SAV3;
      CCR(2)   0;
      STI(2)   .SAV3;
% WE JUST CHANGED THE SIGN OF SIN
MLS3:  LDI(2)   .ADRES;
      CADD(2)   $C0;
      LDA      0(2);% WE NOW CALCULATING THE ELEMENTS
% OF THE SECOND ROW OF BASE
      LDI(3)   .SAV3;% $C3=SINH OR -SINH.
      MLRN    $C3;% ROW I TIMES $C3;
      STA      TEMP;
      CADD(2)   $D1;
      LDA      0(2);
      LDI(3)   .SAV2;% $C3=COSH OR COS
      MLRN    $C3;
      ADRN    TEMP;
      LDI(2)   .ADRES2;
% .ADRES2 IS THE ADDRESS OF THE TEMP.SOR.MATRY TRASE
      CADD(2)   $C0;
      CADD(2)   $D1;% FIND ROW I+1;
      STA      0(2);
      LDI(2)   .SAV9;% CHECK .SAV9:=0 OR :=1.
% IF .SAV9:=0 WE ARE WORKING WITH COSH & SINH WHICH IS THE
% SAME FOR THE TOTAL MATRIX ANMAT(TR.), SO WE NEED NOT
% CHANGE THE ADDRESS IN $C1 TO PICK UP A DIFFERENT ELEMENT
% OF ANMAT(TR.) AND JUMP TO MLS1.
% IF .SAV9:=1 WE ARE WORKING WITH COS & SIN AND HAVE TO PICK
% UP DIFFERENT ELEMENTS FROM ANMAT(TR.), SO WE SKIP TO MLS4A.
      FOIXF(2) $D0,MLS6;
      TXFFM(0) ,MLS1;
      SKIP    ,MLS7;
MLS6:  TXFFM(0) ,MLS4A;
% REESTABLISH BASE BY LOADING TRASE INTO BASE
MLS7:  IIT(0)   =0,1,0;
      CADD(0)   $D3;
      CRTL(0)   24;
MLS5:  LDI(1)   .ADRES2;
      LDI(2)   .ADRES;
      CADD(1)   $C0;
      LDA      0(1);
      CADD(2)   $C0;
      STA      0(2);
      TXFFM(0) ,MLS5;
      LDI(3)   .SAV1;
      FXCHL(3) $ICR;
%#####
TRASPOS: FILL;
%
% PROCEDURE TRASPOS FINDS THE TRANSPOSE
% OF THE ANGLE-MATRIY BY CHANGING THE
% SIGN OF THE OFF-DIAGONAL ELEMENTS
%
      STI(3)   .SAV1;% SAVE RETURN ADDRESS
      LDI(0)   .ANTIUN;
      IDEF1   $C0;
      LDA      NUMR;% FIND THE INDEX FOR THE ELEMENTS
% THAT NEED THE SIGN CHANGE
      LDS     =1;
      ADM     $$;

```

```

00067300
00067400
00067500
00067600
00067700
00067800
00067900
00068000
00068100
00068200
00068300
00068400
00068500
00068600
00068700
00068800
00068900
00069000
00069100
00069200
00069300
00069400
00069500
00069600
00069700
00069800
00069900
00070000
00070100
00070200
00070300
00070400
00070500
00070600
00070700
00070800
00070900
00071000
00071100
00071200
00071300
00071400
00071500
00071600
00071700
00071800
00071900
00072000
00072100
00072200
00072300
00072400
00072500
00072600
00072700
00072800
00072900
00073000
00073100
00073200
00073300

```

```

STA      TEMP;                                00073400
LDA      NUMB;                                00073500
CSHR(0)  1;                                    00073600
RTL      $A,1;                                00073700
LDF1     $C0;                                  00073800
LDA      $R;                                    00073900
STA      TFMP;X      THE INDEX IS FOUND      00074000
LDL(0)   .SPEC;                                00074100
LDF1     $C0;                                  00074200
LDS      TEMP;                                  00074300
LDL(3)   .ADRC;                                00074400
LDA      #0(3);                                00074500
CHSA;                                          00074600
STA      #0(3);                                00074700
LDL(3)   .SAV1;X      RETURN                  00074800
EXCH(3)  $ICR;                                00074900
X *****00075000
ANGL:    FJ(1);                                00075100
X      PROCEDURE ANGL FINDS COSINE AND SINE, TO FORM THE
X TRANSFORMATION MATRIX ANMAT.                00075200
X      SINE=SQRT(0.5-C[(2K-1,2K)/H]).          00075300
X      COSINE=SQRT(0.5+C[(2K-1,2K)/H]).        00075400
X      H=SQRT(4X[C[(2K-1,2K)+2+(C[(2K-1,2K-1)-C[(2K,2K)])+2]. 00075500
X THERE ARE DIFFERENT CASES TO BE CONSIDERED FOR
Y THE CALCULATION OF COSINE. THEY WILL BE DISCUSSED AS THEY
X COME UP IN THE PROGRAM.                    00075600
Y#####00075700
      STI(3)   .SAV1;X      SAVE RETURN ADDRESS 00075800
      LIT(0)   =0,1,0;      00075900
      ADD(0)   $D3;          00076000
      CRTL(0)  24;           00076100
      LD(1)    .ADRC;        00076200
      LIT(3)   =0.;          00076300
      LD(2)    .SPEC;        00076400
      LDF1     $C2;          00076500
      LDA      $C3;          00076600
AG:      STA      0(1);      00076700
      ADD(1)   $D1;          00076800
      TXFFM(0) ,AG;X      MAKE SURE THAT THE OFF-DIAGN
X                                     NAL ELEMENTS ARE ZERO. 00076900
      LIT(0)   =0,2,0;      00077000
      ADD(0)   $D3;          00077100
      CSUP(0)  $D1;          00077200
      CRTL(0)  24;           00077300
      CLC(1);   00077400
AG1:    CCR(1)  0(0);X      CREATE THE TURN ON PATTERN
X                                     FOR THE OFF-DIAG. PICK=UP 00077500
      TXFFM(0) ,AG1;        00077600
      STI(1)   .ANTUN;      00077700
      LDF1     $C1;          00077800
      LDA      NUMB;        00077900
      LIT(2)   =1;          00078000
      ADM      $C2;          00078100
      STA      TEMP;        00078200
      LDA      NUMB;        00078300
      RTI      $A,1;        00078400
      CSHR(1)  1;           00078500
      LDF1     $C1;          00078600
      LDA      $R;          00078700
      STA      TFMP;X      INDEX IS FOUND AND IT IS OF THE
X FORM 2K,2K-1;2K-1,2K. I.E. 1,0,3,2,5,4,..... 00078800

```



```

* NOW WE CALCULATE
* A) HSQR=4XC[2K-1,2K]+2+(C[2K-1,2K]-C[2K,2K])*2
* B) H =SQRT(HSQR)
* C) SINE=SQRT(0.5-C[2K-1,2K]/H)
* TO A):
LDC      TFMP1;      INDEX IN $$ FOR OFF-DIAGS.
LDI(2)   .ADPCM;    ADDRESS OF MATRIX IN USE
LDA      #0(2);     C[2K-1,2K] LOADED AND
STA      TFMP1;     SAVED HERE FOR LATER CASE
                                DIFFERENTIATION FOR COSINE
*
LDI(3)   .NMN;
STI(3)   .SAV2;
CLC(3);
SLIT(3)  =ROUTF;EXCH(3) $ICR;
CSHI(1)  1;
LDFF1    $C1;
STA      TEMP1;     TEMP1 HAS IN EVERY PF C[2K-1,2K]
                                IN PAIRS OF TWO.
*
LDI(1)   .SPFC;
LDFF1    $C1;
LDA      TFMP1;
MLRN     TFMP1;     $A=(C[2K-1,2K])*2
LIT(3)   =4.;
LDC      $C3;
MLRN     $S;       $A=4X(C[2K-1,2K])*2
STA      TFMP2;
LDY      NUMP;     INDEX FOR THE DIAG. ELEMENTS
LDA      #0(2);   C[I,I] LOADED
LDI(1)   .ANTIIN;
LDFF1    $C1;
STA      GERSHG;   SAVE C[2K-1,2K-1]
CLC(3);
SLIT(3)  =ROUTF;EXCH(3) $ICR;
LDC      $A;
LDFF1    $C1;
LDA      GERSHG;
SBRN     $S;
STA      GERSHG;   C[2K-1,2K-1]-C[2K,2K]
RTI      $A,1;
CSHR(1)  1;
LDFF1    $C1;
LDA      $R;
STA      GERSHG;
* ABOVE VALUES IN ALL PFS OF GERSHG IN PAIRS OF TWO
LDI(1)   .SPFC;
LDFF1    $C1;
LDA      GERSHG;
MLRN     GERSHG;   $A=(C[2K-1,2K-1]-C[2K,2K])*2
ADRN     TFMP2;     $A=HSQR
* TO B):
CALL SQRT64();
STA      TFMP2;     H STORED HERE
LIT(0)   =0.;
IAI      $C0;
SETF     I.AND.F;
IAC      $C0;
SETF     J.OR.F;
SETF1    F.AND.F;
* TO C):
LDA      TFMP1;
DVRN     TEMP2;

```

```

00079500
00079600
00079700
00079800
00079900
00080000
00080100
00080200
00080300
00080400
00080500
00080600
00080700
00080800
00080900
00081000
00081100
00081200
00081300
00081400
00081500
00081600
00081700
00081800
00081900
00082000
00082100
00082200
00082300
00082400
00082500
00082600
00082700
00082800
00082900
00083000
00083100
00083200
00083300
00083400
00083500
00083600
00083700
00083800
00083900
00084000
00084100
00084200
00084300
00084400
00084500
00084600
00084700
00084800
00084900
00085000
00085100
00085200
00085300
00085400
00085500

```

```

STA      TFMPI;X      $A=C[2K-1,2K]/H.      00085600
LDI(1)   ,SPFC;      00085700
LDF1     $C1;        00085800
IIT(0)   =0.5;      00085900
LDA      $C0;        00086000
SRRN     TFMPI;X      $A=SINE=0.5-C[2K-1,2K]/H 00086100
CALL     SORT64();    00086200
LDC      TEMP1;X     $S=INDEX FOR C[2K,2K-1] & C 00086300
X        C[2K-1,2K]  00086400
LDI(2)   ,ANTUN;    00086500
LDF1     $C2;        00086600
CH<A;    00086700
LDF1     $C1;        00086800
LDI(3)   ,ADRC;     00086900
STA      #0(3);X     SINE INTO ANMAT 00087000
X#C#####
X NOW HEAR THIS: CASE DIFFERENTIATION FOR COSINE 00087100
X CASE ONE: C[2K-1,2K-1] - C[2K,2K] LSS 0 OR GTR 0 00087200
LDA      GFRSHG;    00087300
IIT(0)   =0.;      00087400
IAI      $C0;X      $A LSS 0 00087500
JAG      $C0;X      $A GRT 0 00087600
SETF     I.AND.E;   00087700
SETC(0)  F;        00087800
STI(0)   ,SAV9;    00087900
SETF     J.OP.F;   00088000
SETC(0)  E;        00088100
CAND(0)  $D5;X     CLEAN OUT WHERE $A EQUAL TO ZERO 00088200
ZERT(0)  ,AG2;X     NO C[2K-1,2K-1]-C[2K,2K] LSS OR 00088300
X        GRT ZERO, SO SKIP TO AG2 00088400
SETF1    E.AND.E;   00088500
IIT(1)   =0.5;    00088600
LDA      $C1;        00088700
ADRN     TEMP2;    00088800
CALL     SORT64();X  $A=SQRT(0.5+C[2K-1,2K]/H) 00088900
LDI(1)   ,SAV9;    00089000
LDF1     $C1;        00089100
CH<A;X    $A LSS 0 WHERE 1 BITS WPF SFT 00089200
LDF1     $C0;        00089300
LDX      NUMR;      00089400
LDI(2)   ,ADRC;    00089500
STA      +0(2);X    COSINE STORED DIAGONALLY WHERE 00089600
X C[2K-1,2K-1] - C[2K,2K] LSS 0 OR GRT 0 00089700
X#C#####
X CASE TWO: C[2K-1,2K-1] - C[2K,2K] EQL 0 00089800
AG2:     COMPC(0);  00089900
LDI(1)   ,SPFC;    00090000
CAND(1)  $C0;      00090100
ZERT(1)  ,AG5;X    IF NONE ARE ZERO FINISHED 00090200
LDF1     $C1;X     PES TURNED ON WHERE CASE TWO = 0 00090300
X CASE 2A: C[2K-1,2K] GRT 0 00090400
IIT(0)   =0.;      00090500
LDA      TEMP1;    00090600
TAG      $C0;      00090700
SETF     I.AND.E;   00090800
SETC(3)  F;        00090900
ZERT(3)  ,AG3;X    NONE ARE GRT 0 00091000
SETF1    F.AND.F;   00091100
IIT(0)   =1.;      00091200
LDA      $C0;      00091300
LDX      NUMR;      00091400
00091500
00091600

```

```

LDI(2)      .ADRC;                                00091700
STA         *0(2);* COSINE = 1 WHERE C[2K-1,2K] GRT 0  00091800
% CASE 2B; C[2K-1,2K] LSS 0                          00091900
AG3:        COMPC(3);                               00092000
           CAND(3)  $C1;                            00092100
           STI(3)   .SAV9;                          00092200
           LDF      $C3;                            00092300
           IIT(0)   =0.;                             00092400
           IAI      $C0;                             00092500
           SFTF     I.AND.E;                         00092600
           SFTC(0)  E;                               00092700
           7FRT(0)  ,AG4;* NONE ARE LSS 0           00092800
           SFTF1    F.AND.E;                        00092900
           IIT(1)   =0.;                             00093000
           LDA      $C1;                             00093100
           LDY      NUMB;                             00093200
           LDI(2)   .ADRC;                            00093300
           STA      *0(2);* COSINE=0., WHERE C[2K-1,2K] LSS 0  00093400
% CASE 2C; C[2K-1,2K] = 0                             00093500
AG4:        COMPC(0);                               00093600
           LDI(3)   .SAV9;                          00093700
           CAND(3)  $C0;                             00093800
           7FRT(3)  ,AG5;* NONE ARE EQ 0           00093900
           LDFF1    $C3;                             00094000
           IIT(1)   =1.;                             00094100
           LDA      $C1;                             00094200
           LDY      NUMB;                             00094300
           LDI(2)   .ADRC;                            00094400
           STA      *0(2);* COSINE=1., WHERE C[2K-1,2K]=0  00094500
           IIT(1)   =0.;                             00094600
           LDA      $C1;                             00094700
           LDS      TEMP;                             00094800
           STA      *0(2);                            00094900
AG5:        LDI(3)   .SAV1;* RETURN TO THE OUTSIDE WORLD  00095000
           FXCHL(3) $ICR;                             00095100
%*****                                              00095200
HYANG:     FI1;                                       00095300
           PROCEDURE HYANG CALCULATES:
%           D=A[2I-1,2K-1]-A[2I,2K].
%           F=A[2I-1,2K]-A[2I,2K-1].
%           K2=SUM(D X F).
%           K1=SUM(D+2+F*2).
%           TANH=-2K2/K1.
%           TANSQ=TANH*2.
%           D1=1./SQRT(1.-TANSQ).
%           F1=SQRT((D+1.)/2.).
%           COSH=SQRT((F+1.)/2).
%           D2=SQRT((F+1.)/2)-11.
%           IF TANH LSS 0 THEN SINH=-D ELSE SINH=D.
%           HYANG[2I-1,2I-1]=HYANG[2I,2I]=COSH
%           HYANG[2I,2I-1]=HYANG[2I-1,2I]=SINH.
% AT THE END OF THE TOTAL COMPUTATION THE CONVERGENCE
% FACTOR IS FOUND BY
%           FINVAL:= 0.5XK1X(1.-SQRT(1.-TANSQ)).
%*****                                              00097100
           STI(3)   .SAV1;* SAVE RETURN ADDRESS      00097200
           SFTF     F.OR.=F;                          00097300
           SFTF1    E.AND.E;                          00097400
           CLRA;                                       00097500
           STA      KAP1;* CLEAR KAP1 AND KAP2       00097600
           STA      KAP2;                              00097700

```



	LIT(0)	=0,2,0; LOOP FOR FETCHING THE PROPER ROWS OF THE MATRIX UNDER CONSIDERATION	00097800 00097900
	CAND(0)	\$D3;	00098000
	CSHP(0)	\$D1;	00098100
	CRNTL(0)	24;	00098200
HYP1:	LDI(1)	.ANTIIN; EVERY OTHER PF TURNED ON	00098300
	LDI(2)	.SPFC;	00098400
	IDFF1	\$C2;	00098500
	LDX	\$C0; INDEX FOR THE A[2I-1,2K-1]	00098600
	CSHP(1)	1;	00098700
	IDFF1	\$C1;	00098800
	ITT(2)	=1;	00098900
	LDA	\$X;	00099000
	ADN	\$C2;	00099100
	LDR	\$A;	00099200
	LDX	\$R1; INDEX FOR A[2I,2K]	00099300
	LDI(2)	.SPFC;	00099400
	LDF1	\$C2;	00099500
	LDI(2)	.ADRA; ADDRESS OF BASE.	00099600
	LDA	*0(2);	00099700
	CHSA;		00099800
	LDI(1)	.ANTIIN;	00099900
	LDF1	\$C1;	00100000
	STA	TEMP; -A[2I-1,2K-1] STORED HERE	00100100
	LDI(3)	.NMD;	00100200
	STI(3)	.SAV2;	00100300
	CLC(3);		00100400
	SLIT(3)	=ROUTE;FXCHI(3) \$ICR;	00100500
	LDF1	\$C1;	00100600
	SBRN	TEMP; -A[2I,2K]-(-A[2I-1,2K-1]).	00100700
	STA	TEMP;	00100800
	RTI	\$A,1;	00100900
	CSHP(1)	1;	00101000
	LDF1	\$C1;	00101100
	STR	TEMP; TEMP:=D	00101200
	LDI(1)	.SPFC;	00101300
	LDF1	\$C1;	00101400
	LDA	TEMP;	00101500
	MLRN	TEMP;	00101600
	STA	TEMP; TEMP:=D*2	00101700
	LDI(3)	.DNF;	00101800
	STI(3)	.SAV2;	00101900
	LDA	\$X;	00102000
	CLC(3);		00102100
	SLIT(3)	=ROUTE;FXCHI(3) \$ICR;	00102200
	LDS	\$A;	00102300
	LDX	\$S; INDEX FOR A[2I-1,2K],A[2I,2K-1]	00102400
	LDA	*0(2); A[2I-1,2K],A[2I,2K-1] LOADED	00102500
	CHSA;		00102600
	LDI(1)	.ANTIIN;	00102700
	CSHP(1)	1;	00102800
	LDF1	\$C1;	00102900
	STA	TEMP2;	00103000
	CLC(3);		00103100
	SLIT(3)	=ROUTE;FXCHI(3) \$ICR;	00103200
	LDF1	\$C1;	00103300
	SBRN	TEMP2; -A[2I,2K-1]-(-A[2I-1,2K])	00103400
	STA	TEMP2;	00103500
	LDI(3)	.NMD;	00103600
	STI(3)	.SAV2;	00103700
	CLC(3);		00103800

```

SLIT(3) =POUTFIFXCH(3) $ICR)
CSHI(1) 1)
LDFF1 $C1)
STA TFMP2); TFMP2:=F.
LDI(1) .SPFC)
LDFF1 $C1)
LDA TEMP2)
MLRN TEMP2)
STA GFRSHG); GFRSHG:=E+2.
LDA TEMP)
MLRN TFMP2); D X F.
ADRN KAP2)
STA KAP2); KAP2:=K2.
LDA TFMP1)
ADRN GFRSHG); D+2+E+2.
ADRN KAP1)
STA KAP1); KAP1:=K1.
TXFTM(0) ,+1)
JUMP HYP)
HYP2: IIT(0) =0)
SETF E.OR.-F)
SETF1 E.AND.F)
CIRA)
LDI(1) .ANTIUM)
LDFF1 $C1)
LDA KAP2(0))
IIT(2) =1)
SETF F.OR.-F)
SETF1 F.AND.F)
HYP1: LDS $A); LOG-SUM FOR KAP2 AND KAP1
RTI $S,0(2))
LDS $R)
ADRN $S)
CAND(2) $C2)
IIT(3) =64)
FOIXF(2) $C3,HYP1); END SUMING
STA KAP2(0))
AIIT(0) =1)
GRTP(0) $D1,+1)
JUMP HYP2)
LDI(1) .SPFC)
LDFF1 $C1)
IIT(0) =-2.)
LDA $C0)
MLRN KAP2)
DVRN KAP1)
STA KAP2); KAP2:=TANH
SAP)
LIT(0) =1.); CHECK IF ABS(TANH-1) LSS 1.E-12.
% IF SO GO OUT AND TRY A NEW MATRIX BASE, FORMED BY THE TRANS-
% FORMATION FOUND UNDER ANGI.
SPRN $C0)
SAP)
IIT(0) =0.000000000001)
IAI $C0)
SETF I.AND.F)
SFTC(0) E)
CMPCC(0))
CAND(0) $D5)
STI(0) .MAX)
ZFRF(0) ,+1)

```

```

00103900
00104000
00104100
00104200
00104300
00104400
00104500
00104600
00104700
00104800
00104900
00105000
00105100
00105200
00105300
00105400
00105500
00105600
00105700
00105800
00105900
00106000
00106100
00106200
00106300
00106400
00106500
00106600
00106700
00106800
00106900
00107000
00107100
00107200
00107300
00107400
00107500
00107600
00107700
00107800
00107900
00108000
00108100
00108200
00108300
00108400
00108500
00108600
00108700
00108800
00108900
00109000
00109100
00109200
00109300
00109400
00109500
00109600
00109700
00109800
00109900

```

	JUMP	HYP3A;	00110000
	LDF1	%C1;	00110100
	LDA	KAP2;	00110200
	MLRN	KAP2; TANSH:=TANH*2	00110300
	STA	GFRSHG; STORE TANSH FOR CALC. OF FINVAL	00110400
	CHSA;		00110500
	LIT(0)	=-1.;	00110600
	SFRN	%C0; SA:=-TANSH-(-1.)	00110700
	CALL	SQRT64();	00110800
*		SA=SQRT(1.-TANSH)	00110900
	LDS	SA;	00111000
	CCP(0)	0;	00111100
	LDA	%C0;	00111200
	DVRN	%S; SA:=1./SQRT(1.-TANSH)	00111300
	ADRN	%C0;	00111400
	LIT(0)	=2.;	00111500
	DVRN	%C0;	00111600
	CALL	SQRT64();	00111700
*		F1 FOUND	00111800
	LIT(0)	=1.;	00111900
	ADRN	%C0;	00112000
	LIT(2)	=2.;	00112100
	DVRN	%C2;	00112200
	CALL	SQRT64();	00112300
	IDY	NUMR;	00112400
	LDI(2)	.ADRC; COSH STORED IN ANMAT	00112500
	STA	#0(2);	00112600
	MLRN	%A; SA:=(COSH)*2	00112700
	SFRN	%C0;	00112800
	CALL	SQRT64();	00112900
	LDS	SA;	00113000
	LDA	KAP2;	00113100
	LIT(0)	=0.;	00113200
	IAI	%C0;	00113300
	SETF	I.AND.F;	00113400
	SETF1	E.AND.F;	00113500
	LDA	%S;	00113600
	CHSA;		00113700
	LDS	SA;	00113800
	LDF1	%C1;	00113900
	STS	KAP2; SINH FOUND	00114000
	LDI(0)	.ANTUN;	00114100
	LDF1	%C0;	00114200
	LDA	NUMR;	00114300
	LIT(3)	=1;	00114400
	ADM	%C3;	00114500
	LDS	SA;	00114600
	LDA	NUMR;	00114700
	RTI	SA,1;	00114800
	CSHR(0)	1;	00114900
	LDF1	%C0;	00115000
	LDS	%R;	00115100
	LDF1	%C1;	00115200
	LDA	KAP2;	00115300
	LDI(2)	.ADRC;	00115400
	STA	#0(2);	00115500
*	SINH	AND COSH ARE STORED PROPERLY. FIND FINVAL	00115600
	LIT(0)	=1.;	00115700
	LDA	%C0;	00115800
	SFRN	GFRSHG; SA:=1.-GFRSHG; GFRSHG:=TANSH	00115900
	CALL	SQRT64();	00116000

	CHSA)			00116100
	CCR(0)	0)X	\$C0:=-1.	00116200
	SRRN	\$C0;		00116300
	MLRN	KAP1;		00116400
	LIT(0)	=0.5)		00116500
	MLRN	\$C0;		00116600
	STA	KAP2;		00116700
	CLC(3);			00116800
	SLIT(3)	=KAP2;		00116900
	LDD(3)	\$C0;		00117000
	STI(0)	.FINVAL;		00117100
HYP3A:;	LDI(3)	.SAV1;		00117200
	FYCHL(3)	\$ICR;		00117300
*****				00117400
NULCHK: FILI:				00117500
THIS PROCEDURE CHECKS:				00117600
% APS(C[2K-1,2K-1]-C[2K,2K]) LEQ 1.0-12.				00117700
% IF SO, IT THEN FINDS THE SIGN OF C[I,I] OPPOSITE OF THAT				00117800
% OF C[0,0] AND THE ROWINDEX I=.MAX. IT THEN LEADS INTO				00117900
% THE PROCEDURE SHEET WHICH EXCHANGES THE SECOND POW				00118000
% WITH ROW I. THE PROCEDURE SHEET IS ALSO USED FOR THE				00118100
% MATRIX BASE.				00118200
*****				00118300
	STI(3)	.SAV1;X	SAVE RETURN ADDRESS	00118400
	LDI(0)	.SPEC;		00118500
	LDEF1	\$C0;		00118600
	LDY	NUMP;		00118700
	LDI(1)	.ADRCM;		00118800
	LDA	*0(1);X	LOAD DIAGONAL ELEMENTS	00118900
	CHSA)			00119000
	CLC(2);			00119100
	LIT(3)	=0,2,0)X	CREATE PATTERN TO TURN ON	00119200
			THE EVEN PFS.	00119300
	CADD(3)	\$D3;		00119400
	CSHR(3)	\$D1;		00119500
	CRDTL(3)	24;		00119600
NUL:	CCP(2)	0(3);		00119700
	TXFFM(3)	,NUL;		00119800
	STI(2)	.ANTUN;		00119900
	LDEF1	\$C2;		00120000
	STA	TEMP;X	-C[2K-1,2K-1] SAVED	00120100
	LDI(3)	.NMN;		00120200
	STI(3)	.SAV2;		00120300
	CLC(3);			00120400
	SLIT(3)	=ROUTE;FYCHL(3) \$ICR;		00120500
	SRRN	TEMP;X	DIFFERENCE OF OFF DTAGS IN EVEN	00120600
	LDEF1	\$C2;X	PFS	00120700
	STA	TEMP;X	-C[2K,2K]-(-C[2K-1,2K-1])	00120800
	LDI(3)	.DNF;		00120900
	STI(3)	.SAV2;		00121000
	CLC(3);			00121100
	SLIT(3)	=ROUTE;FYCHL(3) \$ICR;		00121200
	CSHR(2)	1;		00121300
	LDEF1	\$C2;		00121400
	STA	TEMP;		00121500
	LDEF1	\$C0;		00121600
	LDA	TEMP;		00121700
	SAPIX		TAKE ARS. VAL. WHERE NECESS.	00121800
	LDI(1)	.TNMTW;		00121900
	LDEF1	\$C0;		00122000
	IAC	\$C1;		00122100

```

      SETF      =I.AND.E;
      SFTC(3)  F;
      STI(3)   .INDEX;
      ZFRF(3)  ,+1; NO EXCHANGE NECESSARY IE TEST IS
      JUMP     NUL5; SATISFIED.
*
      LDFF1    $C0; FIND THE SIGN OF C(0,0)
      LDY      NUMR;
      LDI(2)   .ADRCM;
      STI(2)   .ADRES;
      LDA      +0(2); LOAD DIAGS. OF CMAT
      ISN;
      SFTF      J.AND.E;
      SFTC(1)  F;
      LDI(2)   .FNR;
      CAMD(2)  $C1;
      ZFRF(2)  ,NUL2; IF TEST IS SATISFIED, C(0,0)
*                               IS NEGATIVE, ELSE POS.
      LEADD(1);
      SKIP     ,+1;
NUL2: LEADZ(1);
      IIT(2)   =77;8;
      CAMD(2)  $C1; THE INDEX OF C[I,I] DIFFERENT
*                               IN SIGN FROM C(0,0) IS FOUND
      STI(2)   .MAX;
      EQUXF(2) $D1,+1;
      JUMP     NUL5;
      SKIP     ,+2;
SHT:  FIIL;
      STI(3)   .SAV1;
      LDI(0)   .SPEC;
      LDFF1    $C0;
      LDI(2)   .ADRES; EXCHANGE ROW 1 WITH ROW I
      CAMD(2)  $D1;
      LDA      0(2);
      STA      TEMP;
      LDI(3)   $C2;
      CAMD(3)  $D7;
      CSJIP(3) $D1;
      LDA      0(3);
      STA      0(2);
      LDA      TFMP;
      STA      0(3); END EXCHANGE ROWS. TO EXCHANGE
* COLUMNS SKFW MATRIX FIRST.
      IIT(0)   =0,1,0;
      CAMD(0)  $D3;
      CRTL(0)  24;
NUL3: LDI(1)   .ADRES;
      CAMD(1)  $C0;
      LDA      0(1);
      STI(0)   .SAV2;
      CLC(3);
      SLIT(3)  =ROUTE;FXCHI(3) $ICR;
      STA      0(1);
      TXFFM(0) ,NUL3;
* NOW EXCHANGE SECOND COLUMN WITH COLUMN I
      LDA      NUMR;
      LDI(1)   .DNF;
      STI(1)   .SAV2;
      CLC(3);
      SLIT(3)  =ROUTE;FXCHI(3) $ICR;

```

```

00122200
00122300
00122400
00122500
00122600
00122700
00122800
00122900
00123000
00123100
00123200
00123300
00123400
00123500
00123600
00123700
00123800
00123900
00124000
00124100
00124200
00124300
00124400
00124500
00124600
00124700
00124800
00124900
00125000
00125100
00125200
00125300
00125400
00125500
00125600
00125700
00125800
00125900
00126000
00126100
00126200
00126300
00126400
00126500
00126600
00126700
00126800
00126900
00127000
00127100
00127200
00127300
00127400
00127500
00127600
00127700
00127800
00127900
00128000
00128100
00128200

```

```

LDS      $A;                                00128300
LDY      $S;X   COLUMN INDEX FOR COLUMN 1  00128400
LDI(1)   .MAX;                                00128500
CSHR(1)  $D1;                                00128600
STI(1)   .SAV2;                              00128700
CLC(3);                                     00128800
SLYT(3)  =ROUTE;FXCHI(3) $ICR;             00128900
STA      TEMP1;X COLUMN INDEX FOR ROW I=.MAX 00129000
LDI(1)   .ADRES;                              00129100
LDA      *0(1);X   SECOND ROW LOADED        00129200
CLC(3);                                     00129300
SLYT(3)  =ROUTE;EXCHL(3) $ICR;X   D=.MAX-1  00129400
STA      TEMP; .                               00129500
LDS      TFMP1;                                00129600
LDA      #0(1);X   COLUMN I=MAX LOADED      00129700
LDI(2)   .N;                                  00129800
CSHR(2)  $D7;                                00129900
CADD(2)  $D1;                                00130000
STI(2)   .SAV2;                              00130100
CLC(3);                                     00130200
SLYT(3)  =ROUTE;FXCHI(3) $ICR;X   D=N-(MAX-1) 00130300
STA      *0(1);X   COLMN. I STORED IN COL. 1 00130400
LDA      TFMP; .                               00130500
LDS      TFMP1;                                00130600
STA      #0(1);X   COL. 1 STORED INTO COL. I. 00130700
X COLUMNS ARE EXCHANGED , NOW UNSKEW THE MATRIX 00130800
LIT(0)   =-1,1,0;                             00130900
CADD(0)  $D3;                                00131000
LDI(2)   .ADRES;                              00131100
CADD(2)  $D1;                                00131200
NUL4:    LDA      0(2);                         00131300
STI(0)   .SAV2;                              00131400
CLC(3);                                     00131500
SLYT(3)  =ROUTE;FXCHL(3) $ICR;             00131600
STA      0(2);                                00131700
ALYT(2)  =1;                                  00131800
TXFFM(0) ,NUL4;                              00131900
NUL5:    I DI(3)   .SAV1;                      00132000
FXCHL(3) $ICR;                                00132100
X*****
SYASY:   FILL;                                00132300
X THIS PROCEDURE MAKES THE MATRIX TRULY SYMMETRIC 00132400
X BY STORING (ABS(A[I,J]) + ABS(A[J,I]))/2 INTO 00132500
X THE A[I,J] & A[J,I] POSITIONS OF BASE. AT THE 00132600
X SAME TIME AN ERROR MATRIX EPS IS CREATED BY FIND- 00132700
X ING THE DEVIATION FROM A[I,J](OLD) & A[J,I](OLD) 00132800
X AND THE ABOVE AVERAGE.                        00132900
STI(3)   .SAV1;                              00133000
CLC(3);                                     00133100
STI(3)   .ANTUNI;                             00133200
X ANTUNI WILL CONTAIN A TAG. IF !=0, THEN A[I,J]=A[J,I] 00133300
X AND BASE WILL HAVE REAL EIGENVALUES. IF !=1 THEN 00133400
X AT LEAST ONE A[I,J]= -A[J,I] AND BASE WILL HAVE 00133500
X COMPLEX EIGENVALUES, WHICH MEANS A PROCEDURE HAS TO BE 00133600
X CHOSEN THAT CAN HANDLE COMPLEX EIGENVALUES.    00133700
LIT(0)   =0,1,0;X   SET UP MAIN LOOP        00133800
X THE DIAGONALS FROM 1 TO N/2 PARTICIPATE 00133900
X IN THE FOLLOWING ALGORITHM.                    00134000
LDI(1)   .N;X   $C1:=N                       00134100
CSHR(1)  1;X   $C1:=N/2                      00134200
CADD(0)  $C1;                                00134300

```



```

CRCTL(0) 24; 00134400
CADD(0) $D1; 00134500
ASY6: LDI(1) .SPFC; 00134600
LDEF1 $C1; 00134700
LDA NUMR; 00134800
STI(0) .SAV2; 00134900
CLC(3); 00135000
SLIT(3) =ROUTE;FXCHI(3) $ICR; 00135100
LDS $A; 00135200
STS TFMP2; 00135300
LDX $S;X INDEX FOR THE ELEMENTS IN THE 00135400
X DIAGONALS TO THE RIGHT OF THE MAIN DIAGONAL 00135500
LDI(2) .ADRA; 00135600
LDA *0(2); 00135700
TSM; 00135800
SETH I.AND.F;X H HAS THE SIGN PATTERN OF 00135900
X THE ELEMENTS UNDER CONSIDERATION 00136000
SETC(2) H;X $C2 IS NEEDED TO CHECK THE 00136100
X DIFFERENCE IN SIGN PERFORMED LATER 00136200
SAP; 00136300
STA TFMP;X :=A[I,J](OLD) 00136400
LDI(3) .N; 00136500
CSIP(3) $C0; 00136600
STI(3) .SAV2; 00136700
CLC(3); 00136800
SLIT(3) =ROUTE;FXCHI(3) $ICR; 00136900
LDS $A;X INDEX FOR THE ELEMENTS IN THE 00137000
X DIAGONALS TO THE LEFT OF THE MAIN DIAGONAL 00137100
STS GERSH; 00137200
LDI(3) .ADRA; 00137300
LDA *0(3); 00137400
STI(0) .SAV2; 00137500
CLC(3); 00137600
SLIT(3) =ROUTE;FXCHI(3) $ICR; 00137700
JSN; 00137800
SETJ J.AND.F;X SAME SIGN CHECK AS BEFORE 00137900
SETC(1) J; 00138000
CFYDR(2) $C1; 00138100
ZFPT(2) .ASY2;X CHECK FOR DIFFERENCE IN SIGN 00138200
LDI(2) .N; 00138300
STI(2) .ANTUN1; 00138400
ASY2: SAP; 00138500
STA TFMP1; 00138600
ADRN TEMP; 00138700
IIT(2) =2.0; 00138800
DVN $C2;X FIND AVERAGE 00138900
LDS $A;X A[I,J](NEW) & A[I,I](NEW) 00139000
LDA TFMP; 00139100
X FIND DEVIATION FROM OLD AND NEW VALUES OF BASE 00139200
SRN $S;X :=A[I,J](OLD) - A[I,I](NEW) 00139300
SETF H.AND.F; 00139400
SETF1 E.AND.E; 00139500
CHCA; 00139600
LDI(1) .SPFC; 00139700
LDEF1 $C1; 00139800
LDX TEMP2; 00139900
LDI(2) .ADRF;X $C2:=ADDRESS OF ERROR MATRIX 00140000
STA *0(2); 00140100
LDA $S;X $A=AVERAGE 00140200
SETF H.AND.F; 00140300
SETF1 E.AND.F; 00140400

```

```

CHSA;
LDF1 $C1;
LDI(2) .ADRA; STORE A[I,J](NEW)
STA #0(2);
LDA TEMP1;
SRBN $S; A[J,I](OLD) = A[I,J](NEW)
SETF J.AND.F;
SETF1 F.AND.F;
CHSA;
LDF1 $C1;
LDI(2) .N;
CSHP(2) $C0;
STC TEMP; SAVE A[I,J](NEW) TEMPORARILY
STI(2) .SAV;
SLIT(3) =ROUTE;FXCH(3) $ICR;
LDI(2) .ADRA;
LDC GFRSHG; STORE A[J,I](OLD) = A[I,J](NEW)
STA #0(2);
LDA TEMP1;
SETF J.AND.F;
SETF1 F.AND.E;
CHSA;
LDF1 $C1;
CIC(3);
SLIT(3) =ROUTE;FXCH(3) $ICR;
LDI(2) .ADRA;
LDC GFRSHG; STORE A[J,I](NEW)
STA #0(2);
TXSTM(0) ,+1;
JUMP ASY6;
LDI(3) .SAV1;
FXCHL(3) $ICR;
#####
AVFR: FTIL;
%
% AVER TAKES THE AVERAGE OF THE OFF-
% DIAGONAL ELEMENTS EXCEPT MAX/C[I,J]/ AND COMPARES THE RE-
% SULT WITH MAX/C[I,J]/. WE WANT TO CHECK THIS WAY THE RAN-
% GE OF THE OFF-DIAGONALS WITH RESPECT TO MAX/C[I,J]/
% SO THAT WHEN EXCHANGING ROW I WITH ROW J IN SHFT, THE ERR-
% OR MADE BY BRINGING A DIFFERENT ELEMENT C[I,J] INTO THE
% 2K-1,2K POSITION, WILL NOT BE OF SIGNIFICANCE. THIS EXCHAN-
% GE IS ALLOWABLE IF C[I,J]/MAX(C[I,J]) IS 1.
%
% STI(3) .SAV; SAVE THE RETURN ADDRESS
%
% IIT(0) =0,1,0; LOOP FOR PICKING UP THE
% OFF - DIAGONAL ELEMENTS OF CMAT. CMAT IS SYMMETRIC.
%
% LDI(1) .N; N:=ORDER OF THE MATRIX
% CSHP(1) 1; DIVIDE BY TWO(2)
% CSUP(1) $D1; $C1:=(N/2)-1
% CADD(0) $C1;
% CRTL(0) 24; THIS LOOP ENABLES US TO PICK
% CADD(0) $D1;
% UP THE NECESSARY ELEMENTS FROM THE UPPER AND THE LOWER HALF
% OF CMAT, WHICH TOGETHER MAKE UP THE NUMBER OF OFF-DIAGS.
% OF THE UPPER HALF OF CMAT, EXCEPT THE N/2 ELEMENTS
% OF CMAT IF GOING DOWN THE DIAGONAL STARTING IN COL. N/2+1
%
% SETF F.OP.F;
% SETF1 F.AND.F;
% CIRA;
% STA TEMP; CLEAN UP TEMP;
%
% AV:
% LDI(3) .SPEC;
% LDF1 $C3;

```



```

LDA      NUMR;          00146600
STL(0)  .SAV2;        00146700
CIC(3);          00146800
SI TT(3) =R01TF;FXCHL(3) $ICR; 00146900
LDS     $A; $S:= INDEX FOR OFF-DIAGS. 00147000
LDI(2)  .ADRCM; $ ADDRESS OF CMAT 00147100
LDA     #0(2);      00147200
SAP;          00147300
ADRN     TEMP; $ FORM PARTIAL SUM 00147400
STA     TEMP;      00147500
TXFFM(0) ,AV;      00147600
LDI(3)  .SPEC;      00147700
IDFF1   $C3;        00147800
LDA     NUMR; $ INDEX FOR THE REST N/2 00147900
* ELEMENTS OF CMAT NOT YET CONSIDERED 00148000
RTI     $A,0(0);    00148100
CASHR(3) 0(0);      00148200
CAMP(3)  $D5;       00148300
IDFF1   $C3;        00148400
LDS     $R;         00148500
LDA     #0(2);      00148600
SAP;          00148700
ADRN     TEMP;      00148800
STA     TEMP;      00148900
* ALL OFF-DIAGONAL ELEMENTS HAVE BEEN LOOKED AT (I.F. UPPER 00149000
* HALF OF CMAT) AND THEY ARE PARTIALLY SUMMED. 00149100
* NOW WE LOG-SUM,I.F., ACROSS PLS 00149200
LIT(2)  =1;         00149300
SETF    F,OR.-F;    00149400
SETF1   F,AND.-F;   00149500
LDA     TEMP;       00149600
AV1:    LDS     $A; 00149700
RTI     $S,0(2);    00149800
LDS     $R;         00149900
ADRN     $S;        00150000
CADD(2)  $C2;       00150100
LIT(3)  =64;        00150200
FOI XF(2) $C3,AV1; $ END SUMMING 00150300
LDL(2)  .G;         00150400
SPRN     $C2;       00150500
* LET G1 BE THE SUM OF THE OFF-DIAGS. AND G BE THE LARGEST 00150600
* OFF-DIAGONAL ELEMENT; THEN $A CONTAINS G1 - G. NOW WE 00150700
* AVERAGE. THE NUMBER OF ELEMENTS INVOLVED IS ((N+2-N)/2)-1 00150800
LDL(0)  .N;         00150900
LEADD(0);          00151000
LIT(1)  =77;8;      00151100
CAND(0)  $C1;       00151200
LIT(1)  =16;        00151300
CSUP(0)  $C1;       00151400
LIT(1)  =47;        00151500
CSUP(1)  $C0;       00151600
LDL(0)  .N;         00151700
CSHL(0)  0(1);      00151800
CSHR(0)  $D2; $ $C0:=N+2-N 00151900
CSHR(0)  1; $ $C0:=(N+2-N)/2 00152000
CSHR(0)  $D1; $ $C0:=((N+2-N)/2)-1 00152100
LDI(3)  $C0; $ SAVE $C0 TEMPORARILY 00152200
* CONVERT FIXED $C0 INTO FLOAT $C0 00152300
LEADD(0);          00152400
LIT(1)  =77;8; $ 00152500
CAND(0)  $C1;       00152600

```

```

LIT(1)      =16;
CSHR(0)     %C1;
CSSL(3)     0(0);% MANTISSA-PART ADJUSTED
LIT(1)      =48;
CSHR(1)     %C0;% CALCULATE THE EXPONENT-PART
LIT(0)      =48;
CSSL(1)     0(0);
CFXOR(3)    %C1;% EXPONENT & MANTISSA JOINED
LNL(1)      .FNR;
CSHR(1)     1;
CFXOR(3)    %C1;% SIGN OF EXPONENT JOINED
DVRN        %C3;
LNL(0)      .G;
DVRN        %C0;
SAP;%
LIT(1)      =0.4;
TAL         %C1;
SFTC(0)     1;
CIC(2);
DNFST(0)    ,AV2;% IF ALL ONE'S THEN
% THE OFF-DIAGONALS ARE NOT IN RANGE
LIT(2)      =1;
AV2:        LNL(3)    .SAV1;
FXCHL(3)    %ICR;% RETURN TO MAINPROGRAM WITH
% TEST RESULT IN %C2.
%*****
%BFRIENTRY1: FILL;% SAVE THE CONTENT OF ACAR0,ACAR1,
% AND ADP-LOCATION $D32 - $D39
STI(3)      .RTHR;% SAVE RETURN ADDRESS
CLC(3);
SLIT(3)     =ADBSAV+R;
STORE(3)    %C0;
ALIT(3)     =1;
STORE(3)    %C1;
CLC(3);
SLIT(3)     =ADBSAV;
LIT(0)      =1,7,0;
SA:         STORE(3)  %D32(0);
TXFFM(0)    ,SA;
LIT(0)      =1,5,0;
SA1:        CLC(3);
LOAD(2)     %C3;% %C2 CONTAINS ADDRESS OF LIST WHICH
% IN TURN CONTAINS THE ADDRESSES OF
% THE PARAMETERS
CSHR(3)     6;
STI(3)      %D32(0);% THE CONTENT OF LIST=PARAMETERS
% PASSED TO THE SUBROUTINE ARE STORED IN $D32 TO $D37
ALIT(2)     =1;%
TXFFM(0)    ,SA1;
CLC(3);
LOAD(2)     %C3;
STI(3)      .N;
LIT(0)      =1;
STI(0)      .DNF;
LIT(0)      =0;
STI(0)      .ZERO;
%*****
LNL(0)      .N;
CSHR(0)     %D1;
STI(0)      .NM0;
LIT(0)      =64;

```

```

00152700
00152800
00152900
00153000
00153100
00153200
00153300
00153400
00153500
00153600
00153700
00153800
00153900
00154000
00154100
00154200
00154300
00154400
00154500
00154600
00154700
00154800
00154900
00155000
00155100
00155200
00155300
00155400
00155500
00155600
00155700
00155800
00155900
00156000
00156100
00156200
00156300
00156400
00156500
00156600
00156700
00156800
00156900
00157000
00157100
00157200
00157300
00157400
00157500
00157600
00157700
00157800
00157900
00158000
00158100
00158200
00158300
00158400
00158500
00158600
00158700

```

```

CSIR(0)   $D2;                                00158800
STI(0)    .ROUT;                               00158900
LIT(0)    =8000000000000000;16;              00159000
STI(0)    .FNR;                                00159100
*****                                         00159200
*
SETF      E.OR.=F;  CREATE THE TURN ON PATTERN  00159300
*                                     FOR THE FIRST N PFS  00159400

SETF1     E.AND.E;                               00159500
LDA       NUMB;                                   00159600
LDI(0)    .N;                                     00159700
TAX       $C0;                                   00159800
SFTC(0)   I;                                     00159900
CLRA;                                           00160000
STI(0)    .SPFC;                                  00160100
*****                                         00160200
IDFF1     $C0;X  CREATE FREIG INITIALLY AS AN  00160300
LIT(1)    =1.;X  IDENTITY= MATRIX              00160400
LDA       $C1;                                   00160500
LDY       NUMB;                                   00160600
LDI(0)    .ADRF;                                  00160700
STA       *0(0);                                  00160800
CLRA;                                           00160900
*****                                         00161000
LIT(0)    =0.000001;                             00161100
STI(0)    .TNMTW;                                00161200
CLC(3);                                         00161300
STI(3)    .AB;                                    00161400
STI(3)    .FINVAL;                                00161500
COMP6:    LDI(3) .ADRA;                            00161600
* NOW WE PERFORM CMAT:=BASE X BASE(TR.) - BASE(TR.) X BASE  00161700
* BUT FIRST: BASE X BASE(TR.) X BASE(TR.) - BASE(TR.) X B)  00161800
STI(3)    .ADRF;                                  00161900
LDI(3)    .ADRCM;                                  00162000
STI(3)    .ADRES;X  ADRES1 GIVES THE ADDRESS OF THE  00162100
*                                     MATRIX WHICH WILL CONTAIN  00162200
*                                     BASE X BASE(TR.). HERE ADRES1:=  00162300
*                                     CMAT.  00162400
*
CLC(3);                                         00162500
SLIT(3) =MLTRPS;EXCH(3) $ICR;                   00162600
* NOW WE TRANSPOSE BASE. THE RESULT WILL BE IN ADRC.  00162700
CLC(3);                                         00162800
SLIT(3) =TRPS;EXCH(3) $ICR;                       00162900
* WE FORM BASE(TR) X BASE. RESULT IS IN ADDR.  00163000
LDI(3)    .ADDR;                                  00163100
STI(3)    .ADRF;                                  00163200
LDI(3)    .ADRC;                                  00163300
STI(3)    .ADRES;                                  00163400
CLC(3);                                         00163500
SLIT(3) =MLTRPS;EXCH(3) $ICR;                       00163600
* NOW THE COMPUTATION OF CMAT=CMAT-ADRC  00163700
LIT(0)    =0.1,0;X  SET UP THE LOOP FOR I  00163800
CAND(0)   $D3;                                    00163900
CRNTL(0)  24;                                      00164000
COMP:     LDI(1) .ADRCM;                            00164100
          CAND(1) $C0;X  00164200
          LDI(2) .ADRC;                              00164300
          CAND(2) $C0;                                00164400
          LDI(3) .SPFC;                              00164500
          LDFF1  $C3;                                00164600
          LDA   0(1);X  $A1= ROW I OF CMAT  00164700
          SBRN  0(2);X  SUBTRACT ROW I OF ANMAT  00164800

```

```

          STA      0(1)X STORE RESULT IN ROW I OF CMAT          00164900
          TXFEM(0) ,COMP;                                       00165000
%      IIT(3)    =1,MESS2=1,MESS1;                               00165100
%      DISPLAY  $C3,32;                                         00165200
%      LDI(3)    .ADRCM;CSHL(3) 6;WRTPEM;                       00165300
%*****CMAT:=BASE X BASE(TP) - BASE(TR) X BASE *****
% LES FIND THE LARGEST OFF DIAGONAL ELEMENT                    00165400
          CLC(3);                                             00165500
          SLIT(3) =FNDMX;FXCH(3) $ICR;                         00165600
%      .G WILL CONTAIN THE LARGEST OFF DIAGONAL ELEMENT        00165700
%      .MAXR & .MAXC WILL CONTAIN THE ROW- & COL.-INDEX      00165800
          ONESE(0) ,+1;X THE MAX. OFF-DIAGONAL FLEM.          00165900
          JUMP    COMP11;                                       00166000
%      WAS ZERO.                                             00166100
          CLC(3);                                             00166200
          SLIT(3) =AVFR;FXCH(3) $ICR;                         00166300
          FOIXF(2) $D0,COMP2A;                                 00166400
          JUMP    COMP2;                                       00166500
%*****CHECK FOR C[2I-1,2I-1]=C[2I,2I]*****
COMP2A:  CLC(3);                                             00166600
          SLIT(3) =NULCHK;FXCH(3) $ICR;                       00166700
          LDI(0)  .INDEX;                                       00166800
          ZFRF(0) ,+1;                                         00166900
          JUMP    COMP2;                                       00167000
          LDI(0)  .MAX;                                         00167100
          FOIXF(0) $D1,+1;                                       00167200
          JUMP    COMP3;                                       00167300
          LDI(3)  .ADRA;                                       00167400
          STI(3)  .ADRES;                                       00167500
          CLC(3);                                             00167600
          SLIT(3) =SHFT;FXCH(3) $ICR;                         00167700
%***** WE JUST EXCHANGED ROW I & ROW J OF BASE *****
%***** NOW WE DO THE SAME TO ERETG*****
          LDI(3)  .ADRF;                                       00167800
          STI(3)  .ADRES;                                       00167900
          CLC(3);                                             00168000
          SLIT(3) =SHFT;FXCH(3) $ICR;                         00168100
          JUMP    COMP3;                                       00168200
COMP2B:  LDI(0)  .MAXR;                                       00168300
          FOIXF(0) $D0,COMP4;                                 00168400
          LDI(0)  .MAXC;                                       00168500
          FOIXF(0) $D1,COMP4;                                 00168600
          JUMP    COMP3;                                       00168700
%***** WE BRING THE LARGEST OFF-DIAGONAL ELEMENT *****
%***** INTO THE 2I,2I-1 POSITION. *****
COMP4:  LDI(3)  .ADRA;                                       00168800
          STI(3)  .ADRES;X SHUFFLE BASE                       00168900
          CLC(3);                                             00169000
          SLIT(3) =SHUF;FXCH(3) $ICR;                         00169100
%      IIT(3)    =1,MESS3=1,MESS2;                               00169200
%      DISPLAY  $C3,32;                                         00169300
%      LDI(3)    .ADRA;CSHI(3) 6;WRTPEM;                       00169400
          LDI(3)  .ADRCM;                                       00169500
          STI(3)  .ADRES;X SHUFFLE CMAT                       00169600
          CLC(3);                                             00169700
          SLIT(3) =SHUF;FXCH(3) $ICR;                         00169800
          LDI(3)  .ADRF;                                       00169900
          STI(3)  .ADRES;X SHUFFLE ERETG                      00170000
          CLC(3);                                             00170100
          SLIT(3) =SHUF;FXCH(3) $ICR;                         00170200
%***** NOW WE ARE READY (OR NOT) FOR THE *****
          LDI(3)  .ADRA;                                       00170300
          STI(3)  .ADRES;X SHUFFLE ERETG                      00170400
          CLC(3);                                             00170500
          SLIT(3) =SHUF;FXCH(3) $ICR;                         00170600
          LDI(3)  .ADRF;                                       00170700
          STI(3)  .ADRES;X SHUFFLE ERETG                      00170800
          CLC(3);                                             00170900

```

```

***** TRANSFORMATION *****
COMP3:  CLC(3);
        SLIT(3) = ANGL;FXCHL(3) $ICR;
***** DO THE TRANSFORMATION BASE X ANMAT *****
        LDI(3)  .ADRA;
        STI(3)  .ADRES;
        STI(3)  .ADRES2;
        LDI(3)  .ADRC;
        STI(3)  .ADRES1;
        IIT(3)  =1;
        STI(3)  .SAV9;
        CLC(3);
        SLIT(3) =MULTPL;FXCHL(3) $ICR;
***** TRANSPOSE ANMAT *****
        CLC(3);
        SLIT(3) =TRASPOS;EXCHL(3) $ICR;
***** NOW BASE:=ANMAT(TR) X BASE *****
        LDI(3)  .ADRA;
        STI(3)  .ADRES;
        LDI(3)  .ADRC;
        STI(3)  .ADRES1;
        LDI(3)  .ADRD;
        STI(3)  .ADRES2;
        IIT(3)  =1;
        STI(3)  .SAV9;
        SLIT(3) =MULSA;FXCHL(3) $ICR;
***** SAVE THE TRANSFORMATION MATRIX FOR *****
***** THE EIGENVECTOR COMPUTATION UNDER JACOBI *****
        LDI(3)  .ADRF;
        STI(3)  .ADRES;
        STI(3)  .ADRES2;
        LDI(3)  .ADRC;
        STI(3)  .ADRES1;
        IIT(3)  =1;
        STI(3)  .SAV9;
        SLIT(3) =MULTPL;FXCHL(3) $ICR;
***** NOW WE FIND THE SECOND TRANSFORM- *****
***** MATION MATRIX CORRESP. TO 0[1] *****
        CLC(3);
        SLIT(3) =HYANG;FXCHL(3) $ICR;
        LDI(0)  .MAX;
        ZFPT(0) .COMP5;
        LDI(3)  .ADRA;          BASE X ANMAT
        STI(3)  .ADRES;
        STI(3)  .ADRES2;
        LDI(3)  .ADRC;
        STI(3)  .ADRES1;
        IIT(3)  =1;
        STI(3)  .SAV9;
        CLC(3);
        SLIT(3) =MULTPL;FXCHL(3) $ICR;
        CLC(3);
        SLIT(3) =TRASPOS;EXCHL(3) $ICR;
        LDI(3)  .ADRA;
        STI(3)  .ADRES;
        LDI(3)  .ADPC;
        STI(3)  .ADRES1;
        LDI(3)  .ADRD;
        STI(3)  .ADRES2;
        IIT(3)  =0;
        STI(3)  .SAV9;

```

```

00171000
00171100
00171200
00171300
00171400
00171500
00171600
00171700
00171800
00171900
00172000
00172100
00172200
00172300
00172400
00172500
00172600
00172700
00172800
00172900
00173000
00173100
00173200
00173300
00173400
00173500
00173600
00173700
00173800
00173900
00174000
00174100
00174200
00174300
00174400
00174500
00174600
00174700
00174800
00174900
00175000
00175100
00175200
00175300
00175400
00175500
00175600
00175700
00175800
00175900
00176000
00176100
00176200
00176300
00176400
00176500
00176600
00176700
00176800
00176900
00177000

```



```

SLIT(3) =MULSA;FYCHI(3) $ICR; 00177100
* LIT(3) =1,ADRS AV-1,MFESS3 00177200
* DISPLAY $C3,32; 00177300
* LDI(3) .ADRA;CSHI(3) 6;WRTPEM; 00177400
LDI(3) .ADRE;X FRFIG X ANMAT(TR.) 00177500
STI(3) .ADRES; 00177600
STI(3) .ADRES2; 00177700
LDI(3) .ADRC; 00177800
STI(3) .ADRES1; 00177900
LIT(3) =1; 00178000
STI(3) .SAV9; 00178100
COMP5: SLIT(3) =MULTPL;FXCHL(3) $ICR; 00178200
LDI(0) .FINVAL; 00178300
LIT(1) =0.000001; 00178400
SFTF F.OR.-F; 00178500
SFTF1 F.AND.-F; 00178600
LDA $C0; 00178700
TAI $C1; 00178800
SFTC(0) I; 00178900
CLRA; 00179000
DNFST(0) ,+1; 00179100
JUMP COMP6; 00179200
LDI(1) .FINVAL; 00179300
LDA $C1; 00179400
SRRN $C0; 00179500
SAP; 00179600
LIT(1) =0.0000000001; 00179700
TAI $C1; 00179800
SFTC(1) I; 00179900
DNFST(1) ,COMP9; 00180000
LDI(1) $A; 00180100
STI(0) .FINVAL; 00180200
LDI(0) .AB; 00180300
SRRN $C0; 00180400
SAP; 00180500
LIT(0) =0.00000001; 00180600
TAI $C0; 00180700
SFTC(0) I; 00180800
DNFSE(0) ,+1; 00180900
JUMP COMP9; 00181000
STI(1) .AB; 00181100
JUMP COMP6; 00181200
COMP9: CLC(3); 00181300
***** MAKE THE RESULTANT MATRIX SYMMETRIC AND ***** 00181400
***** KEEP A TAG IN CASE OF ASYMMETRY ***** 00181500
SLIT(3) =SYASY;FYCHI(3) $ICR; 00181600
* LDI(3) .ADRA;CSHI(3) 6;WRTPEM; 00181700
* LDI(3) .ADRE;CSHI(3) 6;WRTPEM; 00181800
COMP11: CLC(3); 00181900
SLIT(3) =ADPSAV; 00182000
PIN(3) $D32; 00182100
CLC(3); 00182200
SLIT(3) =ADRS AV+8; 00182300
LOAD(3) $C0; 00182400
ALIT(3) =1; 00182500
LOAD(3) $C1; 00182600
LDI(2) .ANTUNI;X TAKE THE PATTERN OF ANTUNI TO THE 00182700
* OUTSIDE TO CHECK FOR THE NEXT PATH OF ACTION 00182800
* CONCERNING THE CALCULATION OF EIGENVALUES. I.E., IF 00182900
* ANTUNI CONTAINS AT LEAST ONE ONE, THEN THE EIGEN- 00183000
* VALUE WILL BE A COMPLEX ONE RESULTING FROM THE JACOB 00183100

```

```

* SUBROUTINE. SO WE TAKE THE PATH TO THE MODIFIED JACOBI
* WHICH CAN HANDLE THE CASE OF THE COMPLEX EIGENVALUE.
  LDL(3)      .RETURN*   TURN BACK TO THE OUTSIDE
  EXCH(3)    STCR)
*****
**
**
**          END FRFR/FRERL
**
**
*****
FND      EBFRL.

```

COLUMN 10,20,30

```

00183200
00183300
00183400
00183500
00183600
00183700
00183800
00183900
00184000
00184100
00184200
00184300
00184400
00184500

```

APPENDIX D  
Results from EIGEN



a) The Original Matrix

```

XXXXXXXXXX DISPLAY # 1 ICR= 002443 1116 TIME=09127144158 FLAPSEC PROCESSOR TIME=00100105111XXXXXXXXXXXXXXXXXXXXX
MEMORY:
-----
LOCATION#8 C(LOCATION) C(LOCATION +0000 0118) C(LOCATION +0000 0218) C(LOCATION +0000 0318)
0200 001+.39999970707000058+0001 -.30587921358500068-0003 -.24079223557299948-0002 +.32893877723800098-0004
XXXXXXXXXX DISPLAY # 2 ICR= 002443 1116 TIME=09127144158 FLAPSEC PROCESSOR TIME=00100105124XXXXXXXXXXXXXXXXXXXXX
MEMORY:
-----
LOCATION#8 C(LOCATION) C(LOCATION +0000 0118) C(LOCATION +0000 0218) C(LOCATION +0000 0318)
0201 001+.30587921358500068+0003 +.10000000408800038+0001 -.3945657461569928-0005 -.59456996816000028-0004
XXXXXXXXXX DISPLAY # 3 ICR= 002443 1116 TIME=09127145106 FLAPSEC PROCESSOR TIME=00100105131XXXXXXXXXXXXXXXXXXXXX
MEMORY:
-----
LOCATION#8 C(LOCATION) C(LOCATION +0000 0118) C(LOCATION +0000 0218) C(LOCATION +0000 0318)
0202 001+.24079223557299948-0002 -.3945657461569928-0005 -.2000002920889978+0001 +.17212239523799978-0003
XXXXXXXXXX DISPLAY # 4 ICR= 002443 1116 TIME=09127145115 FLAPSEC PROCESSOR TIME=00100105138XXXXXXXXXXXXXXXXXXXXX
MEMORY:
-----
LOCATION#8 C(LOCATION) C(LOCATION +0000 0118) C(LOCATION +0000 0218) C(LOCATION +0000 0318)
0203 001+.39999970707000058+0001 -.59456996816000028-0004 -.17212239523799978-0003 +.29999999730400058+0001

```

b) The Diagonalized Matrix

```

##### NTSPLAY # 12 ICR# 002454 1116 TIME=09:44:30:27 FLAPSD PROCESSOR TIME=00:07:08:07#####
MEMORY:
-----
LOCATION:R C(LOCATION)
0200 001+.400000002031611E+0001 C(LOCATION +000 0118) C(LOCATION +000 0318)
      +.2305007226754974E+0010 -.1608705970702920E+0010 +.6136249238732424E+0008
#####

##### NTSPLAY # 13 ICR# 002454 1116 TIME=09:44:31:05 FLAPSD PROCESSOR TIME=00:07:08:25#####
MEMORY:
-----
LOCATION:R C(LOCATION)
0201 001+.2305007226754974E+0010 +.3000000003323763E+0001 C(LOCATION +000 0118) C(LOCATION +000 0318)
      +.412033759593724E+0008 +.412033759593724E+0008 +.260395393283002E+0011
#####

##### NTSPLAY # 14 ICR# 002454 1116 TIME=09:44:31:12 FLAPSD PROCESSOR TIME=00:07:08:32#####
MEMORY:
-----
LOCATION:R C(LOCATION)
0202 001-.1608705797237314E+0010 +.4120337595790831E+0008 C(LOCATION +000 0218) C(LOCATION +000 0318)
      +.1000000007907644E+0001 +.1000000007907644E+0001 -.2699380817071557E+0009
#####

##### NTSPLAY # 15 ICR# 002454 1116 TIME=09:44:31:19 ELAPSED PROCESSOR TIME=00:07:08:39#####
MEMORY:
-----
LOCATION:R C(LOCATION)
0203 001+.6136249224854583E+0008 +.2603955668466920E+0011 C(LOCATION +000 0218) C(LOCATION +000 0318)
      +.2699380816529456E+0009 +.1999999992247012E+0001

```

c) The Eigenvector Matrix

```

XXXXXXXXXX DISPLAY #      A      ICR#  002440 0116  TIME=09:44:12:55  FLAPSEI PROCESSOR  TIME=00:07:05:12XXXXXXXXXXXXXXXXXXXX
MEMORY:
-----
LOCATION:18 C(LOCATION)
0205 001+.999999269847109E+0000      -.333173088480774E-0004      C(LOCATION +0000 01:8)      +.101962835169459E-0003      C(LOCATION +0000 02:8)      +.1203961622744242E-0002      C(LOCATION +0000 03:8)
XXXXXXXXXX DISPLAY #      0      ICR#  002440 0116  TIME=09:44:15:48  FLAPSEI PROCESSOR  TIME=00:07:05:13XXXXXXXXXXXXXXXXXXXX
MEMORY:
-----
LOCATION:18 C(LOCATION)
0206 001+.3310712060560698E-0004      +.999999984203818E+0000      C(LOCATION +0000 01:8)      +.297292404016334E-0004      C(LOCATION +0000 02:8)      +.1720824139929307E-0003      C(LOCATION +0000 03:8)
XXXXXXXXXX DISPLAY #      10     ICR#  002440 0116  TIME=09:44:15:56  FLAPSEI PROCESSOR  TIME=00:07:05:13XXXXXXXXXXXXXXXXXXXX
MEMORY:
-----
LOCATION:18 C(LOCATION)
0207 001-.1019587319959067E-0003      -.297251231758357E-0004      C(LOCATION +0000 01:8)      +.999999943510716E+0000      C(LOCATION +0000 02:8)      -.4323882547824215E-0005      C(LOCATION +0000 03:8)
XXXXXXXXXX DISPLAY #      11     ICR#  002440 0116  TIME=09:44:16:03  FLAPSEI PROCESSOR  TIME=00:07:05:14XXXXXXXXXXXXXXXXXXXX
MEMORY:
-----
LOCATION:18 C(LOCATION)
0210 001-.1203967771670256E-0002      -.1720425542802309E-0003      C(LOCATION +0000 01:8)      +.419601036016502E-0005      C(LOCATION +0000 02:8)      +.9999992604224204E+0000      C(LOCATION +0000 03:8)

```



## DOCUMENT CONTROL DATA - R &amp; D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
3. REPORT TITLE  ILLIAC IV CODES FOR JACOBI AND JACOBI-LIKE ALGORITHMS		2b. GROUP	
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Research Report			
5. AUTHOR(S) (First name, middle initial, last name)  Winfried H. Bernhard			
6. REPORT DATE November 5, 1971	7a. TOTAL NO. OF PAGES 35	7b. NO. OF REFS 4	
8a. CONTRACT OR GRANT NO. DAHCO4 72-C-0001	8a. ORIGINATOR'S REPORT NUMBER(S) CAC Document No. 19		
b. PROJECT NO. ARPA Order 1899	8b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)		
c.			
d.			
10. DISTRIBUTION STATEMENT Copies may be obtained from the address given in (1) above. Approved for public release; distribution unlimited			
11. SUPPLEMENTARY NOTES  None		12. SPONSORING MILITARY ACTIVITY U.S. Army Research Office - Durham Duke Station Durham, North Carolina	
13. ABSTRACT I. Modified JACOBI's Method [1] for finding the eigenvalues and eigenvectors of a Hermitian matrix is a well-suited algorithm for ILLIAC IV. It is based on the idea of subjecting the matrix to a series of orthogonal transformations that eliminate the off-diagonal elements such that the matrix under consideration becomes diagonal. ILLIAC IV with its parallel structure provides a tool for eliminating n off-diagonal elements in one single sweep, so that the whole process of making the matrix diagonal becomes very rapid. II. Modified EBERLEIN's method for real matrices: While Jacobi's method is applied to Hermitian matrices, Eberlein's method [2] applies a series of similarity transformations to a non-symmetric matrix until it is practically normal. The resultant normal matrix is then reduced to the diagonal form [2], obtaining the eigenvalues and eigenvectors. The results, of course, are best when the matrix can be made diagonal.  This document presents a brief theoretical background and a detailed description of both programs, written in ASK, including the flow-charts.			

14

KEY WORDS

LINK A

LINK B

LINK C

ROLE

WT

ROLE

WT

ROLE

WT

Numerical Analysis

ASK























UNIVERSITY OF ILLINOIS-URBANA

510.841L63C C001  
CAC DOCUMENTS\$URBANA  
11-20 1971



3 0112 007263806