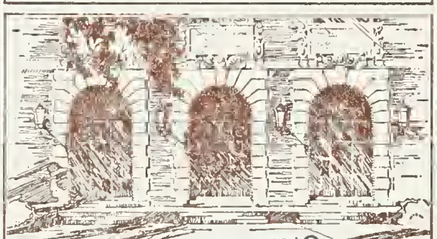


Center for Advanced Computation

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

CAC Document No. 19

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.

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:

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

Each transformation $\varphi_r A_r \varphi_r^t$ 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 <u>one</u> orthogonal transformation. This can be achieved if the transformation matrices φ_r are of the form

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

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 x 2 submatrices of the form

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

For the diagonal submatrix

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

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

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

where

$$X_k = a^{(r)}_{2k,2k} - a^{(r)}_{2k+1,2k+1}, \quad Y_k = \left(t^2_k + X^2_k\right)^{1/2}$$

with

$$t_k = 2a^{(r)}_{2k,2k+1}.$$

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

$$\tan 2\alpha^{(r)}_k = \frac{2a^{(r)}_{2k,2k+1}}{a^{(r)}_{2k,2k} - a^{(r)}_{2k+1,\,2k+1}} .$$

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

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

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

$$\psi = \left[\begin{array}{cc|ccccc} 1 & 0 & 0 & . & . & . & 0 \\ \hline 0 & 0 & & & & & \\ \cdot & \cdot & & & & & \\ \cdot & \cdot & & & I & & \\ \cdot & \cdot & & & & & \\ & & & \multicolumn{3}{c}{(n-2) \times (n-2)} & \\ 0 & 0 & & \multicolumn{3}{c}{\text{identity matrix}} & \\ \hline 0 & 1 & 0 & . & . & . & 0 \end{array}\right]$$

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.

$$A_{(r+2)} = \varphi_{r+1} A'_{(r+1)} \varphi^t_{r+1}.$$

In order to subject <u>all</u> off-diagonal elements to this orthogonal transformation, the matrix A is exposed to a further transformation Γ

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

after (n-2) orthogonal transformations ψ have been performed. Γ is given by

$$\Gamma = \begin{bmatrix} 0 & I_1 \\ & \\ I_2 & 0 \end{bmatrix}$$

with I_1 an (n-m) x (n-m) identity matrix and I_2 an m x 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 ξ has proven to be sufficient. The almost diagonal matrix A_m will be of the form

$$A_m = WAW^t$$

where

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

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 λ is an eigenvalue of an arbitrary n-rowed matrix $A = (a_{jk})$, then for some k, $(1 \leq k \leq n)$,

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

For each k = 1, 2, 3, ..., n this inequality determines a closed circular disk, whose center is the eigenvalue λ_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 x 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 x 2n real symmetric matrix A^1 given by

$$A^1 = \left[\begin{array}{c|c} B & -C \\ \hline C & B \end{array}\right]$$

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

$$\left[\begin{array}{c} u_j \\ \hline v_j \end{array}\right] \text{ and } \left[\begin{array}{c} v_j \\ \hline -u_j \end{array}\right].$$

If λ_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 x 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) $P_e = \text{diag}\,(T_2^{(e)}, \ldots, T_{n/2}^{(e)}$

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.

$$(3) \qquad Q_e = \operatorname{diag}\left[S_1^{(e)}, S_2^{(e)}, \ldots, S_{n/2}^{(e)}\right]$$

with

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

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 $e \to \infty$.

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

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

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

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 x N matrix for which the eigenvalues are sought will be of the form:

BASE: in PE memory

PE: 0 1 2 3 4 5

$$\begin{matrix} \left(\begin{matrix} a_{00} & a_{01} \\ a_{01} & a_{11} \end{matrix}\right) & \begin{matrix} a_{02} & a_{03} \\ a_{12} & a_{13} \end{matrix} & \begin{matrix} a_{04} & a_{05} \\ a_{14} & a_{15} \end{matrix} \\ \begin{matrix} a_{02} & a_{12} \\ a_{03} & a_{13} \end{matrix} & \left(\begin{matrix} a_{22} & a_{23} \\ a_{23} & a_{33} \end{matrix}\right) & \begin{matrix} a_{24} & a_{25} \\ a_{34} & a_{35} \end{matrix} \\ \begin{matrix} a_{04} & a_{14} \\ a_{05} & a_{15} \end{matrix} & \begin{matrix} a_{24} & a_{34} \\ a_{25} & a_{35} \end{matrix} & \left(\begin{matrix} a_{44} & a_{45} \\ a_{45} & a_{55} \end{matrix}\right) \end{matrix}$$

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

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

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

PE	0	1	2	3	4	5
	cos α_1	cos α_1	cos α_2	cos α_2	cos α_3	cos α_3
	-sin α_1	sin α_1	-sin α_2	sin α_2	-sin α_3	sin α_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 α_1	sin α_1	0	0	0	0
	-sin α_1	cos α_1	0	0	0	0
	0	0	cos α_2	sin α_2	0	0
	0	0	-sin α_2	cos α_2	0	0
	0	0	0	0	cos α_3	sin α_3
	0	0	0	0	-sin α_3	cos α_3

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

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

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

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

In calculating BASE = $(ANMAT)^t$ · (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 · (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
	a'_{00}	a'_{02}	a'_{03}	a'_{04}	a'_{05}	0
	a'_{02}	a'_{22}	0	a'_{24}	a'_{25}	a'_{12}
	a'_{03}	0	a'_{33}	a'_{34}	a'_{35}	a'_{13}
	a'_{04}	a'_{24}	a'_{34}	a'_{44}	0	a'_{14}
	a'_{05}	a'_{25}	a'_{35}	0	a'_{55}	a'_{15}
	0	a'_{12}	a'_{13}	a'_{14}	a'_{15}	a'_{11}

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

2. <u>Computation</u>

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:

a. the part that calls the subroutine (Appendix A)

b. 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) <u>The subroutine EIGEN and the Jacobi algorithm</u>: 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

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

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 α_k and sin α_k are formed from BASE and

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

<u>Note</u>: 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. <u>MULTPL</u>: Having found ANMAT, the off-diagonal elements of BASE are eliminated by a series of multiplications; i.e., (ANMAT) x (BASE) x $(\text{ANMAT})^T$ is executed. The same procedure is used to find the eigenvectors; i.e., EIGV = EIGV x $(\text{ANMAT})^T$ · EIGV = I initially. The method used to multiply two matrices stored in straight format is known as <u>Knapp's Method</u> [3]. Its advantage over the <u>log-sum method</u> is that only N^2 multiplications and additions are needed to achieve the result, while in the log-sum method N^2 multiplications, and $(\underline{k+1}) \times N^2$ 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 · 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 · ANMAT is found after 3 · N multiplications.

6. <u>TRASPOS</u>: finds the transpose of ANMAT by changing the sign of sin α_k.

7. <u>SAMUL</u>: This procedure multiplies $(\text{ANMAT})^t$ 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 = 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:

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & & & \\ & 1 & 2 & 3 & 4 & 5 & & \\ & & 1 & 2 & 3 & 4 & 5 & \\ & & & 1 & 2 & 3 & 4 & 5 \\ 5 & & & & 1 & 2 & 3 & 4 \\ 4 & 5 & & & & 1 & 2 & 3 \\ 3 & 4 & 5 & & & & 1 & 2 \\ 2 & 3 & 4 & 5 & & & & 1 \end{bmatrix}$$

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 · ((N/2) + 1) multiplications. The total transformation

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

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

8. <u>SHUFL</u>: 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 $\varphi \times BASE_{(r)} \times \varphi^t$.

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

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

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

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

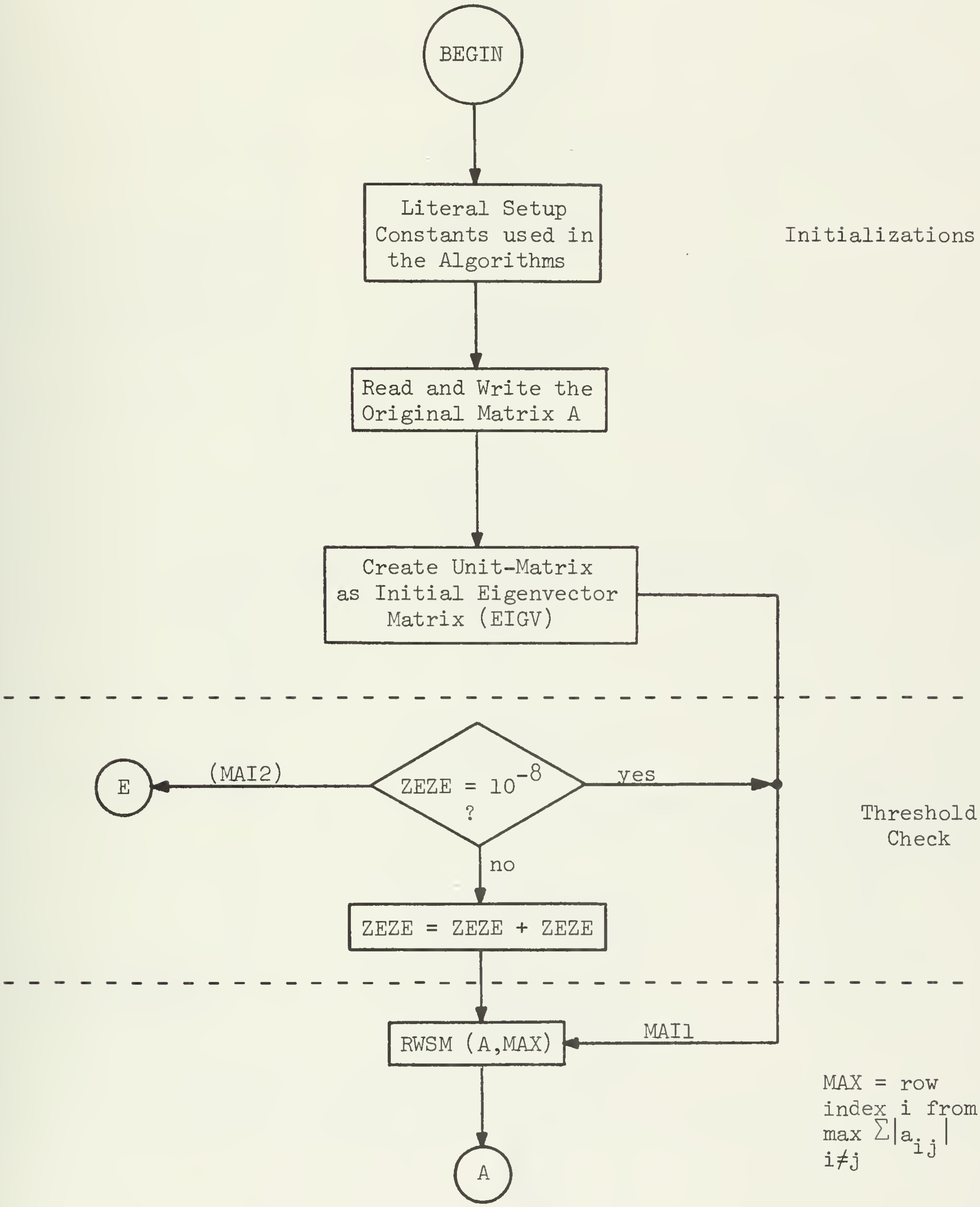
Final Comment: The routines ROUTE, ROTAL and ROTAR are standard parts of any major program. They adjust routing of registers and rotating of patterns in ACARs to the left and right, respectively. They make it possible to handle matrices of sizes $N \leq 64$ and, therefore, make every program more general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

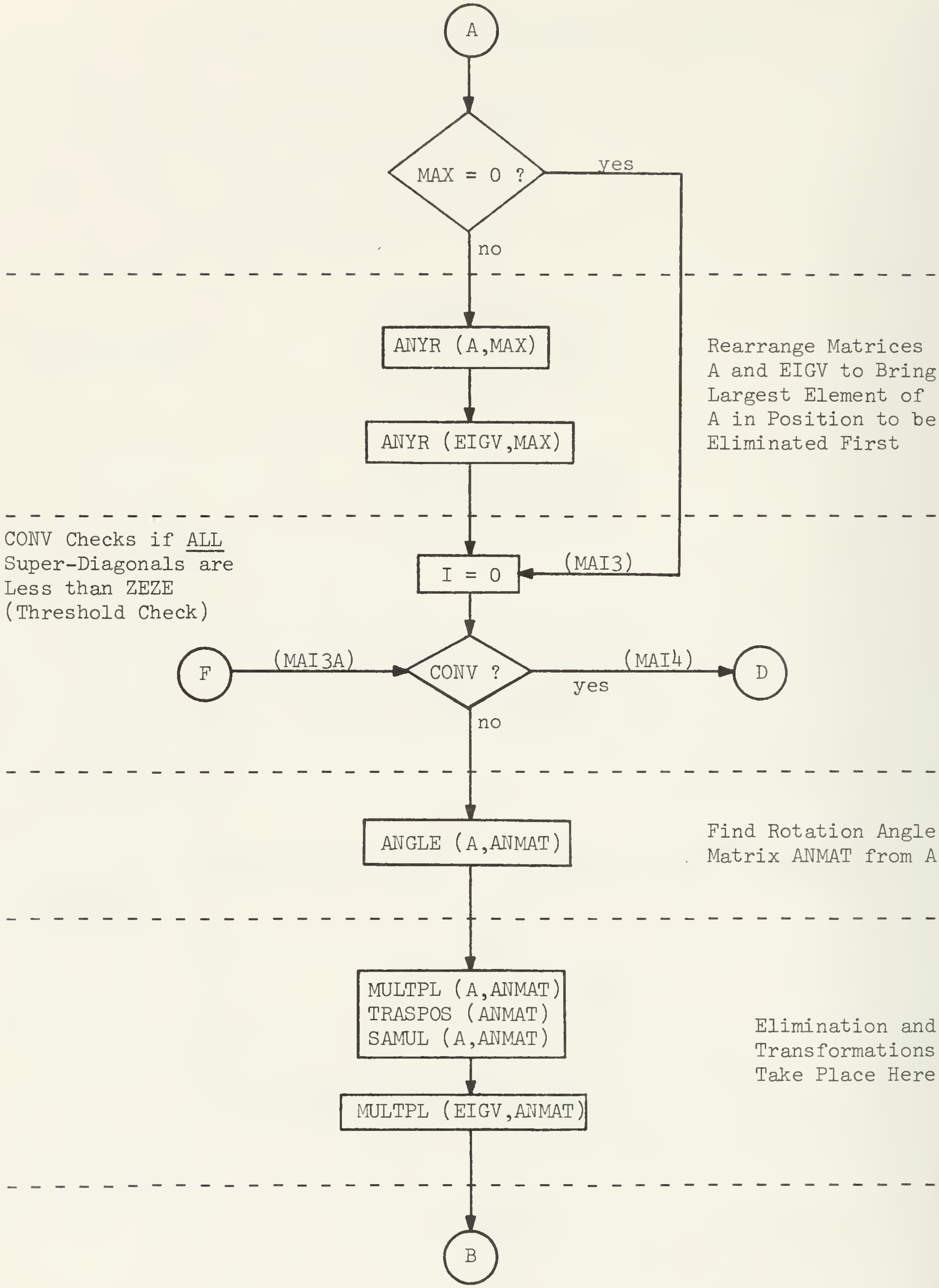
The flow-charts are purposely made lengthy to aid the reader and help him gain a better understanding of the program.

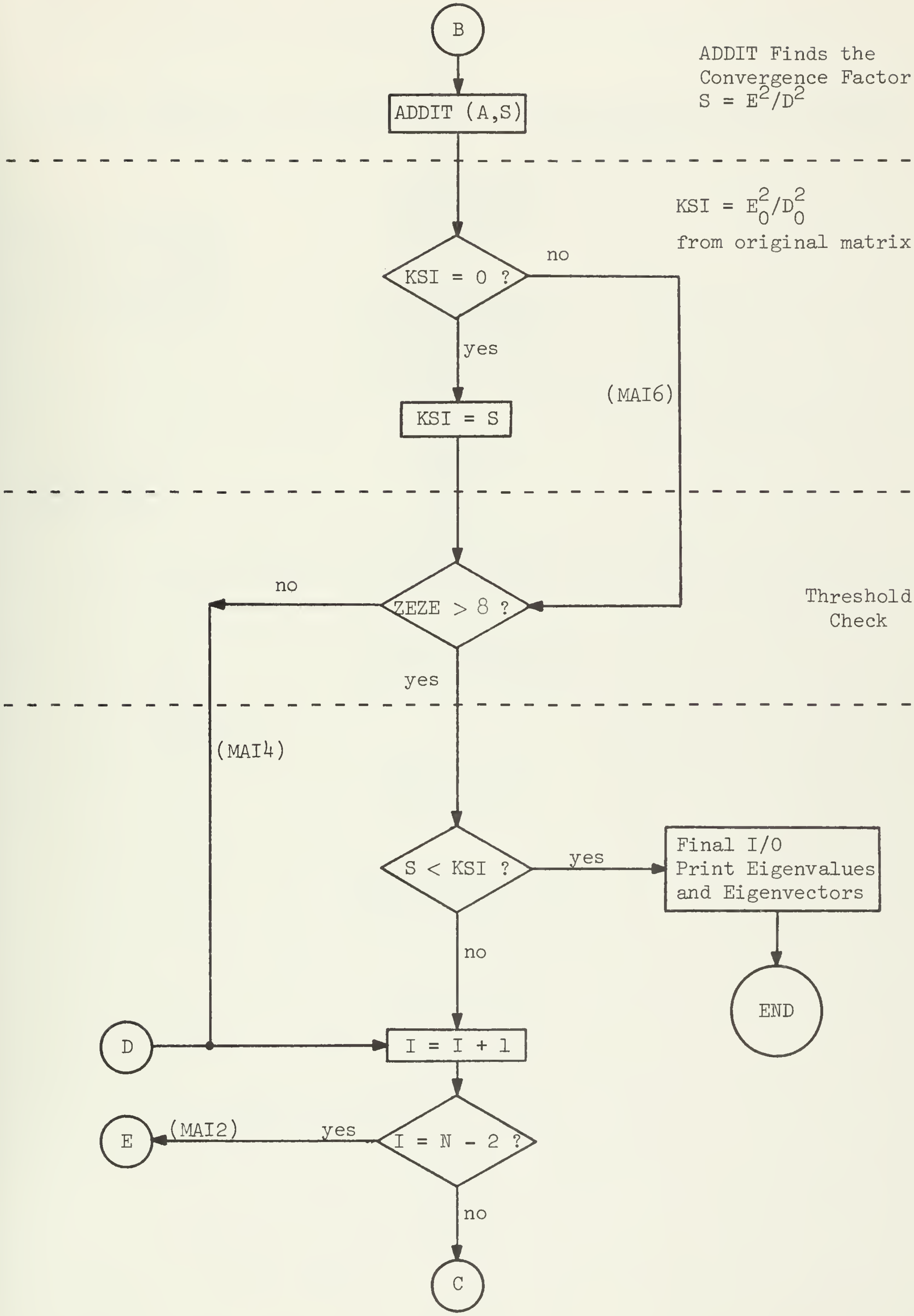
Comments within the program are made wherever the author deemed it necessary. For the most part these comments pertain to groups of instructions. Since the assembly language ASK is highly mnemonic, comments in abundance would hamper rather than facilitate readability.

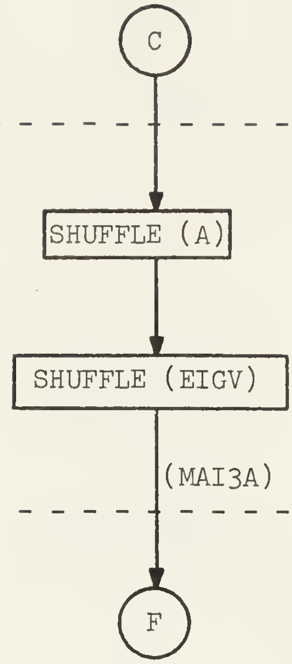
3. Flowcharts

Flowchart 1. MAINPROGRAM: of Subroutine EIGEN




A
MAX = 0 ?
yes
no
ANYR (A,MAX)
ANYR (EIGV,MAX)
Rearrange Matrices A and EIGV to Bring Largest Element of A in Position to be Eliminated First
CONV Checks if ALL Super-Diagonals are Less than ZEZE (Threshold Check)
I = 0
(MAI3)
F
(MAI3A)
CONV ?
(MAI4)
yes
D
no
ANGLE (A,ANMAT)
Find Rotation Angle Matrix ANMAT from A
MULTPL (A,ANMAT)
TRASPOS (ANMAT)
SAMUL (A,ANMAT)
MULTPL (EIGV,ANMAT)
Elimination and Transformations Take Place Here
B


B
ADDIT (A,S)
ADDIT Finds the Convergence Factor $S = E^2/D^2$
$KSI = E_0^2/D_0^2$ from original matrix
KSI = 0 ?
no
yes
(MAI6)
KSI = S
no
ZEZE > 8 ?
Threshold Check
yes
(MAI4)
S < KSI ?
yes
Final I/O Print Eigenvalues and Eigenvectors
END
no
D
I = I + 1
E
(MAI2)
yes
I = N - 2 ?
no
C

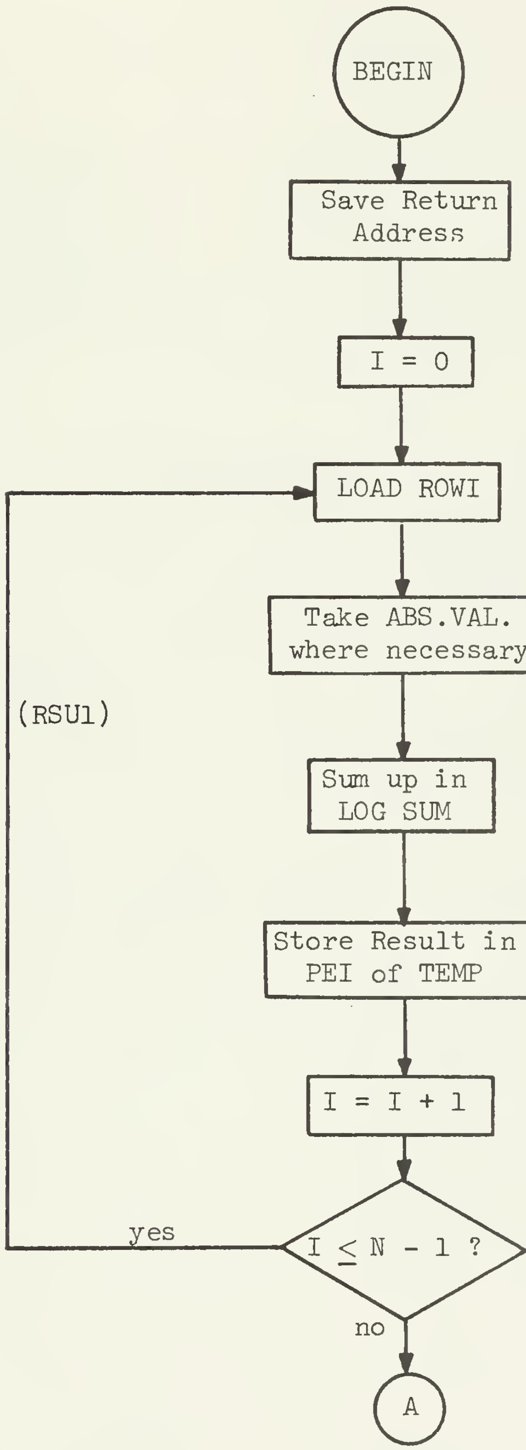


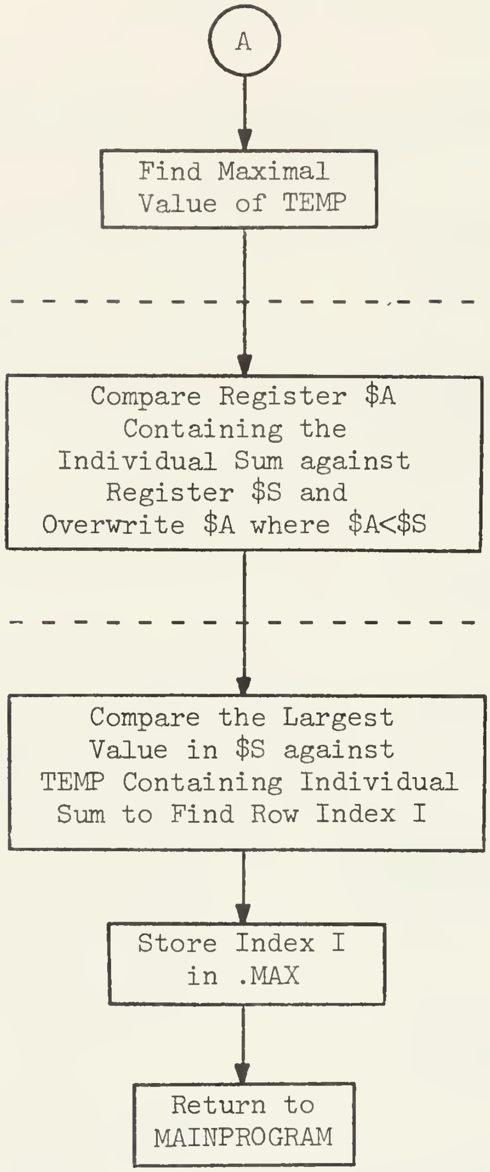
2nd Row-2nd Column Shuffle to Bring New Elements into the A[2I-1,2I] Position for Elimination

Go Back and Start a New Transformation

Flowchart 2. Flowcharts of the Individual Routines

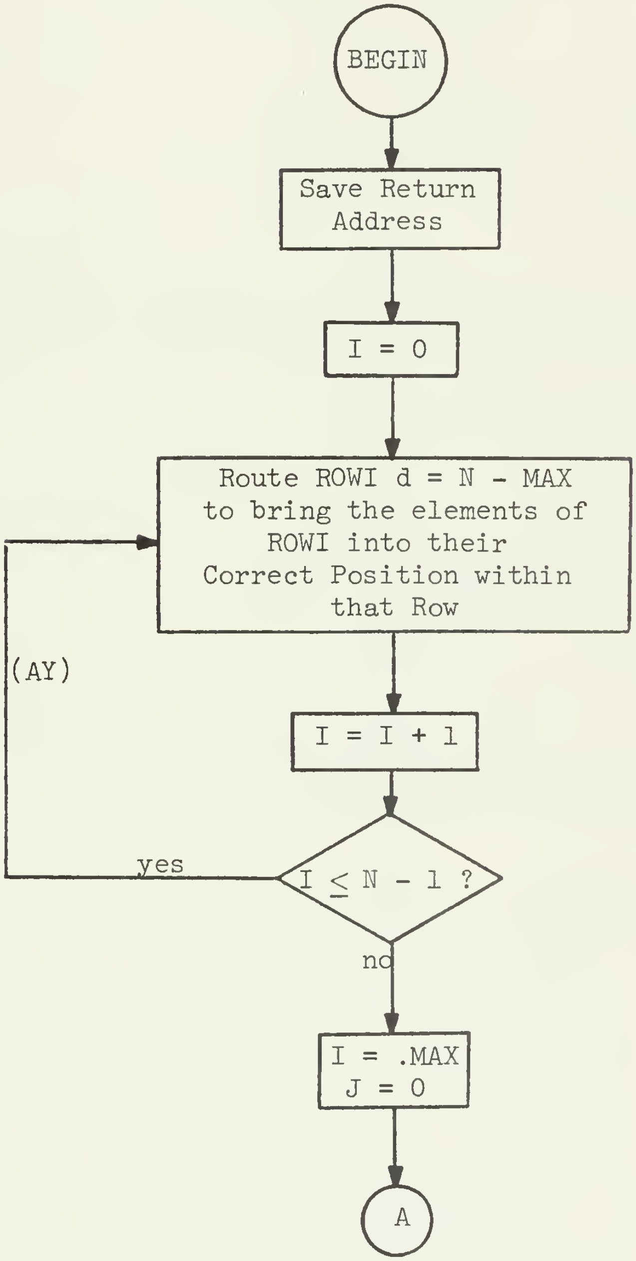
a) RWSM: Summing up the Individual Rows of A and Find the INDE of the Row with Maximal Sum (A[I,I] does not participate).

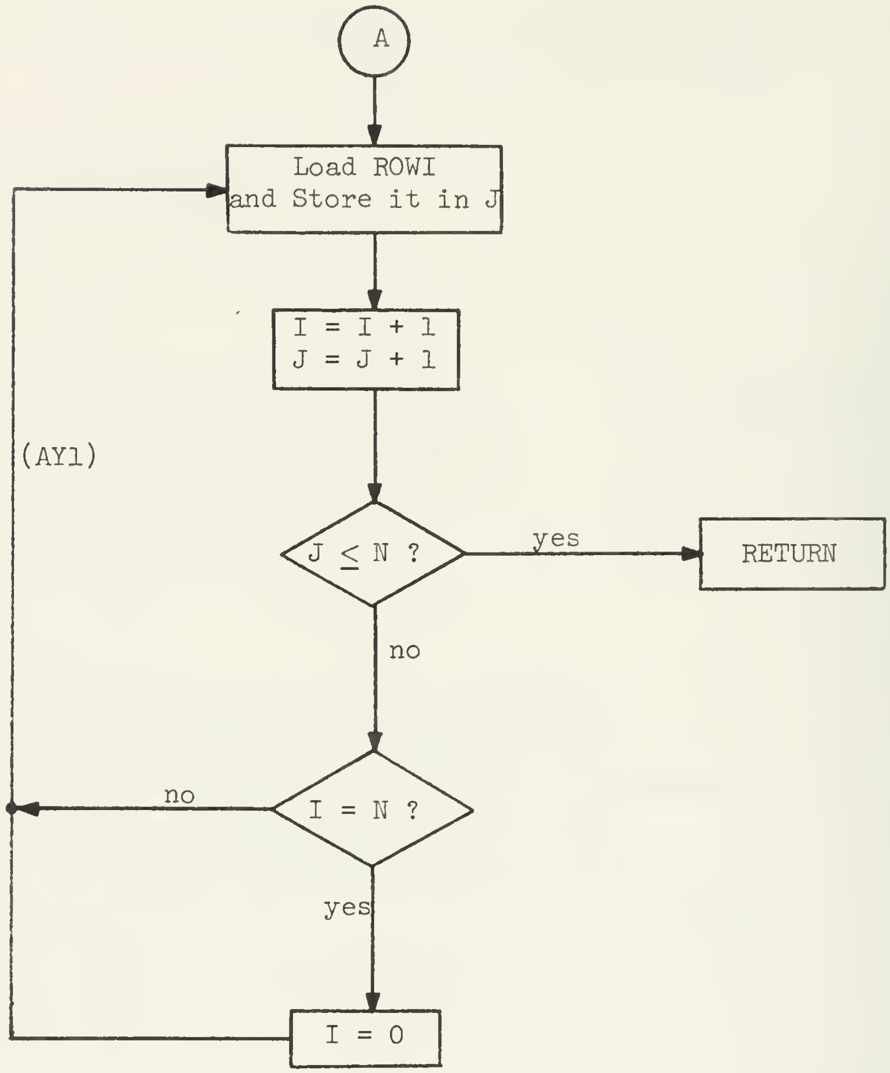




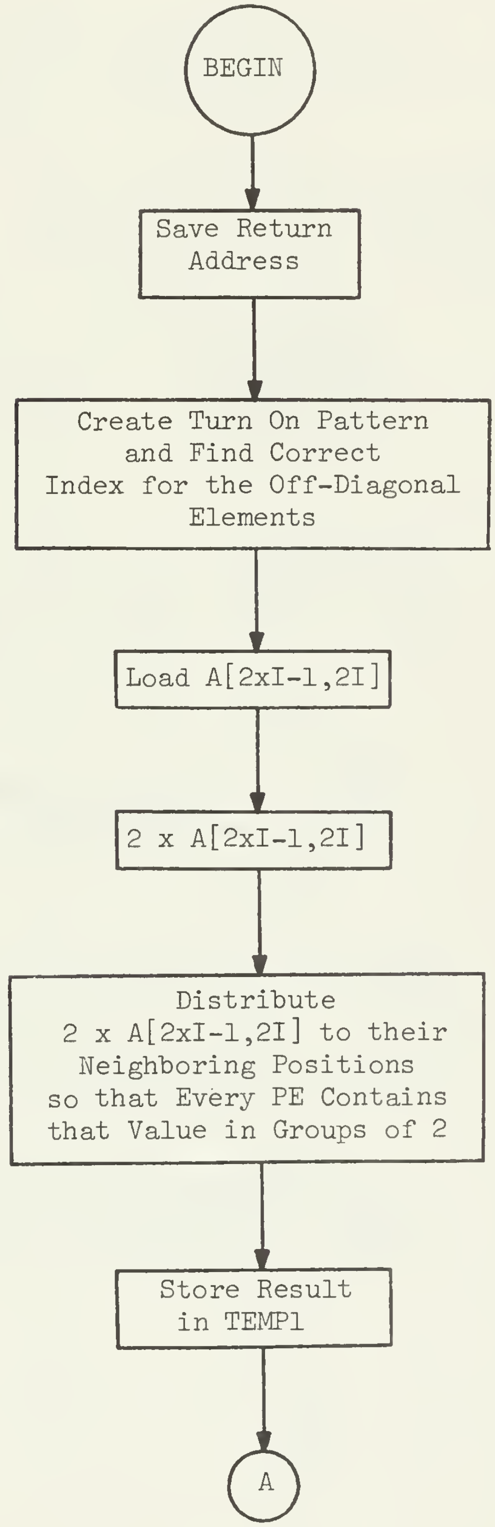
This comparison is done in the LOOP "RSM". $S is shuffled in powers of two. Six comparisons will fill $S with the maximal value.

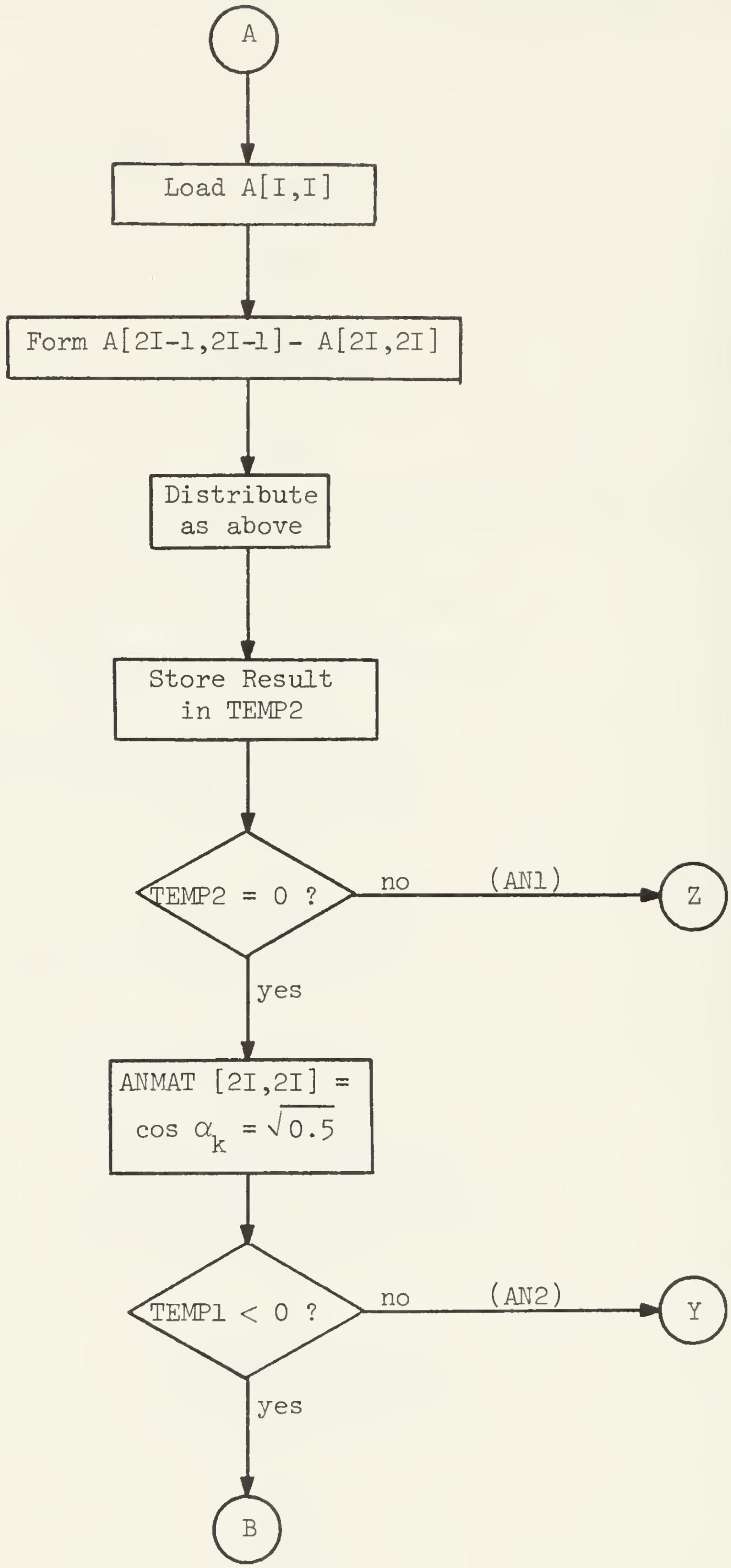
b) <u>ANYR</u>: Stands for ANY-row, ANY-column Shuffle, where ANY = .MAX. It brings the Row with the Largest Elements into a Position, where they can be Eliminated First.

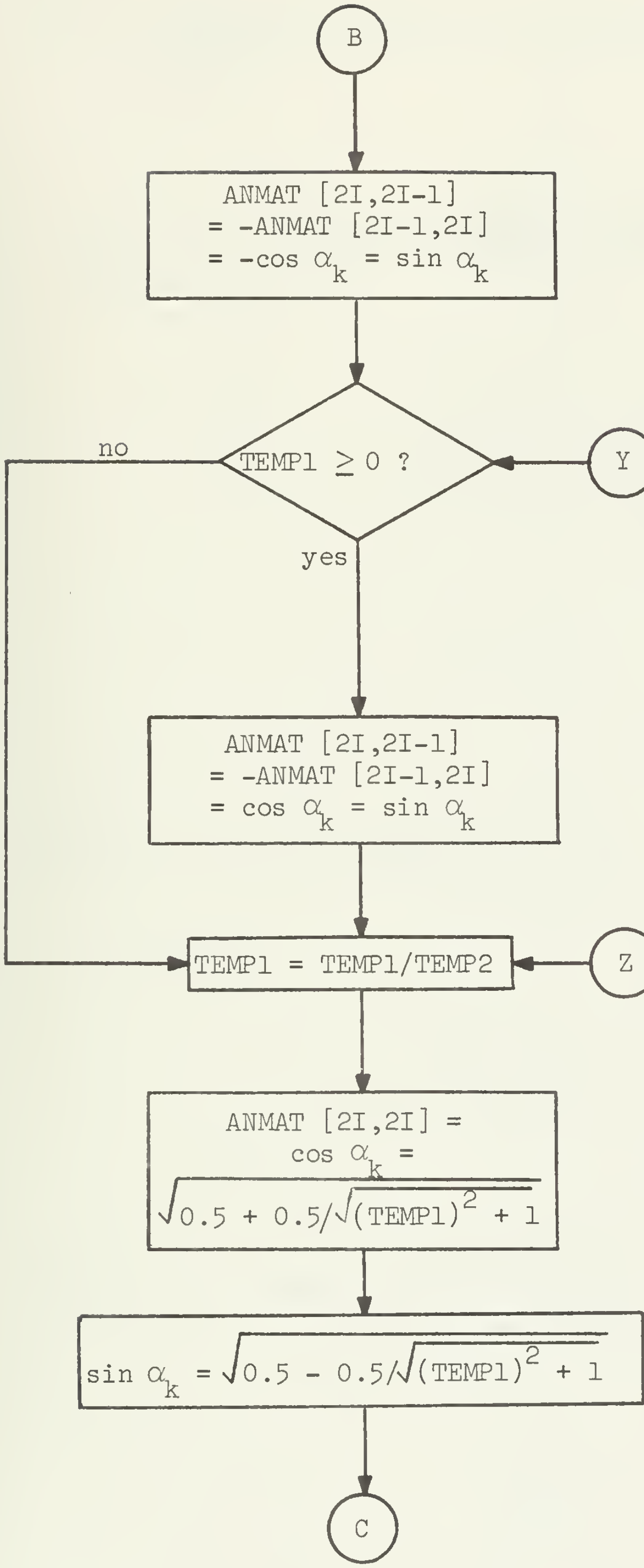



A
Load ROWI
and Store it in J
I = I + 1
J = J + 1
(AY1)
J ≤ N ?
yes
RETURN
no
no
I = N ?
yes
I = 0

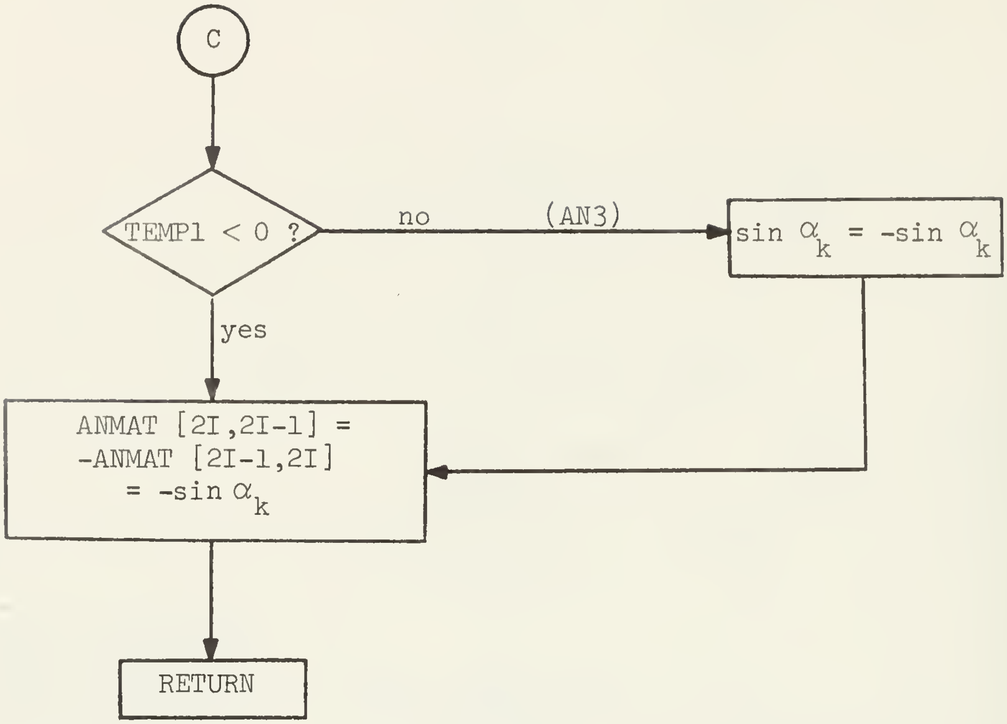
c) <u>ANGLE</u>: ANGLE Calculates the Transformation-Angles and Creates the Transformation Matrix.



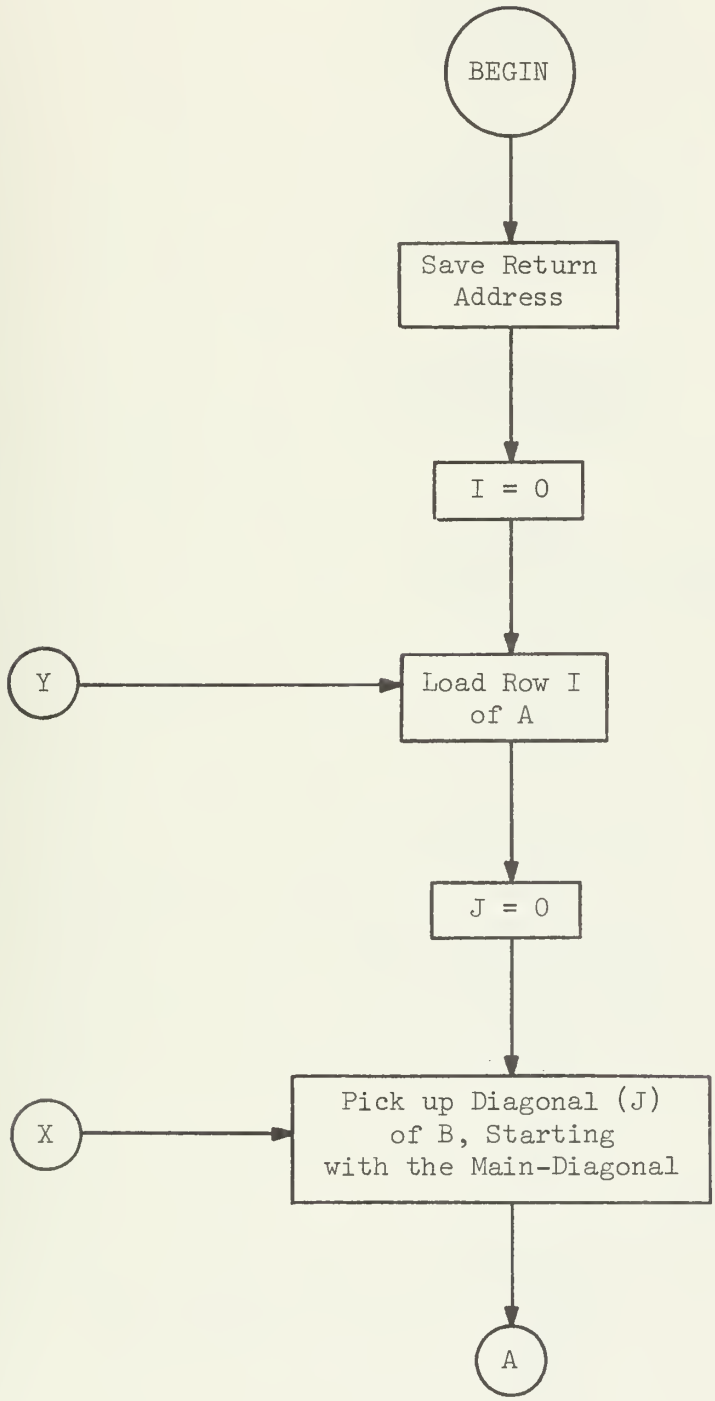

A
Load A[I,I]
Form A[2I-1,2I-1]- A[2I,2I]
Distribute as above
Store Result in TEMP2
TEMP2 = 0 ?
no
(AN1)
Z
yes
ANMAT [2I,2I] = cos α_k = $\sqrt{0.5}$
TEMP1 < 0 ?
no
(AN2)
Y
yes
B

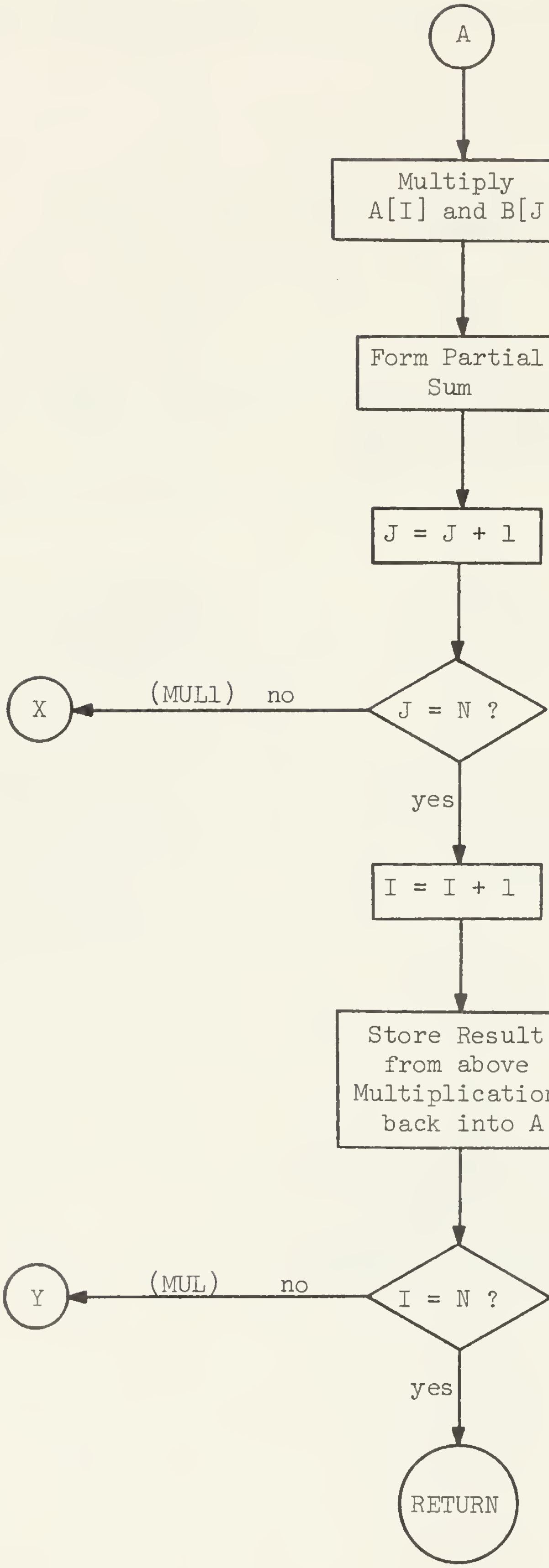


We have to realize that these tests are based on the parallel structure of ILLIAC IV, i.e. the elements ANMAT [2I,2I-1] are never the same for the different steps and that all branches are satisfied for the set of PEs under consideration.

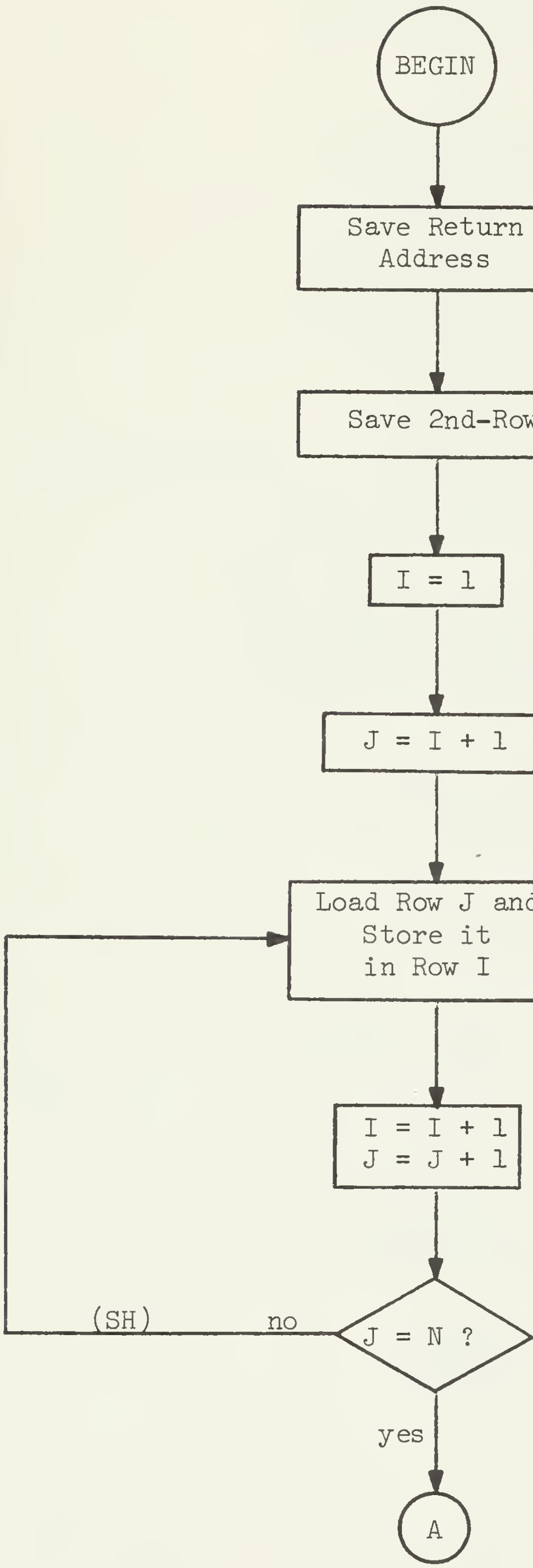

C
TEMP1 < 0 ?
no
(AN3)
sin α_k = -sin α_k
yes
ANMAT [2I,2I-1] =
-ANMAT [2I-1,2I]
= -sin α_k
RETURN

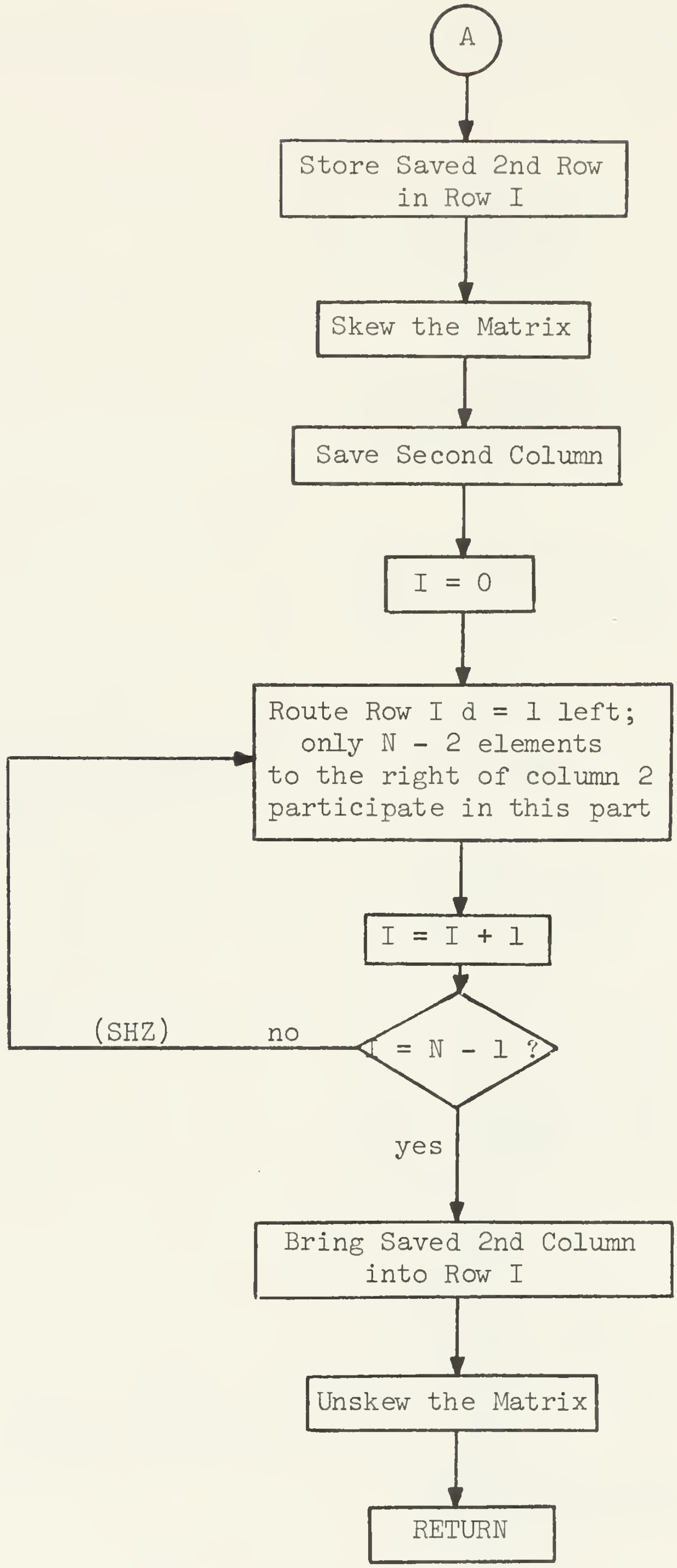
d) MULTPL: Multiplication-Routine A x B for Two Matrices. Addressing is done with reference to the location of BASE. Knapp's method is applied.



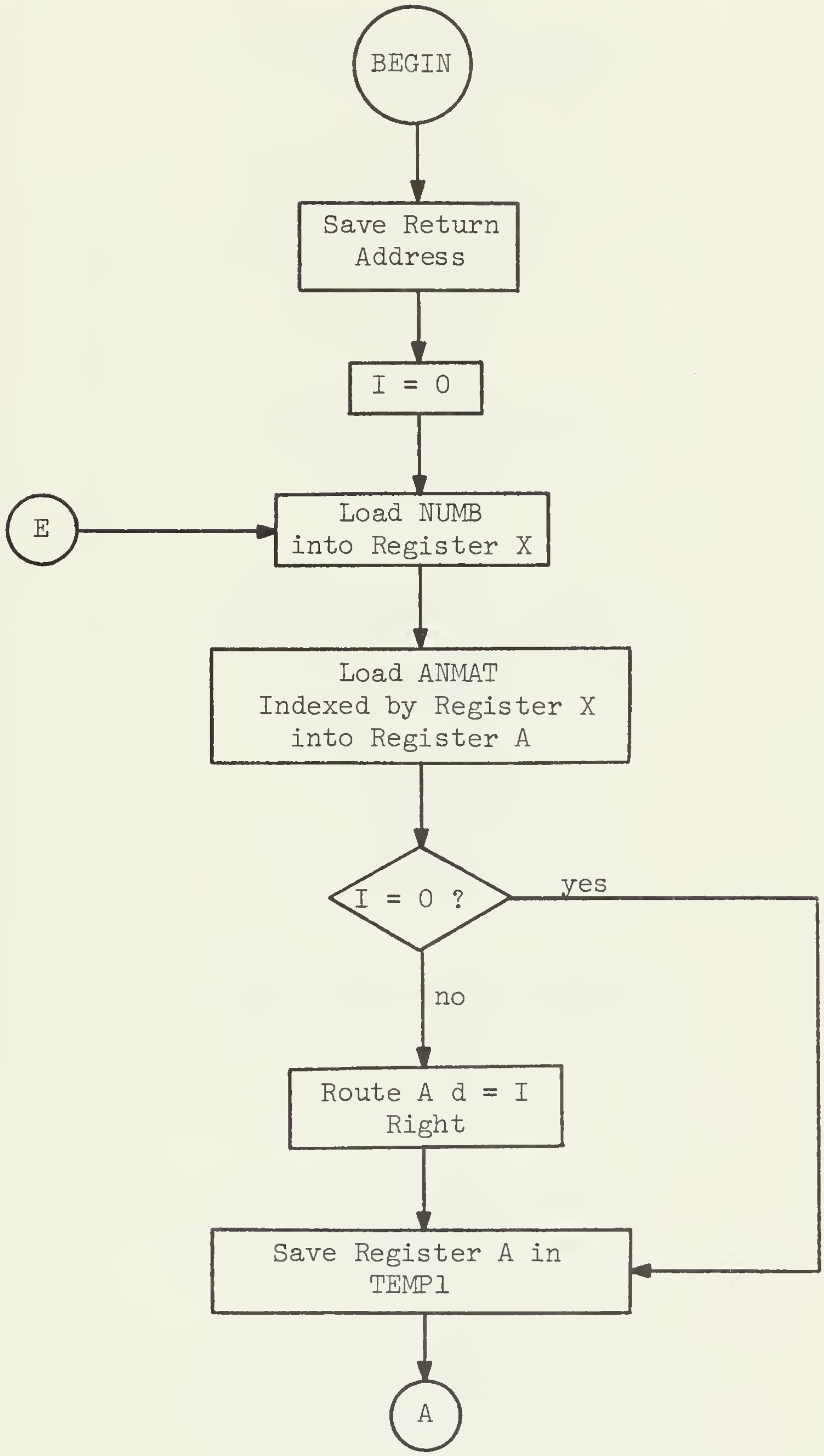

A
Multiply
A[I] and B[J]
Form Partial
Sum
J = J + 1
J = N ?
(MUL1) no
X
yes
I = I + 1
Store Result
from above
Multiplication
back into A
I = N ?
(MUL) no
Y
yes
RETURN

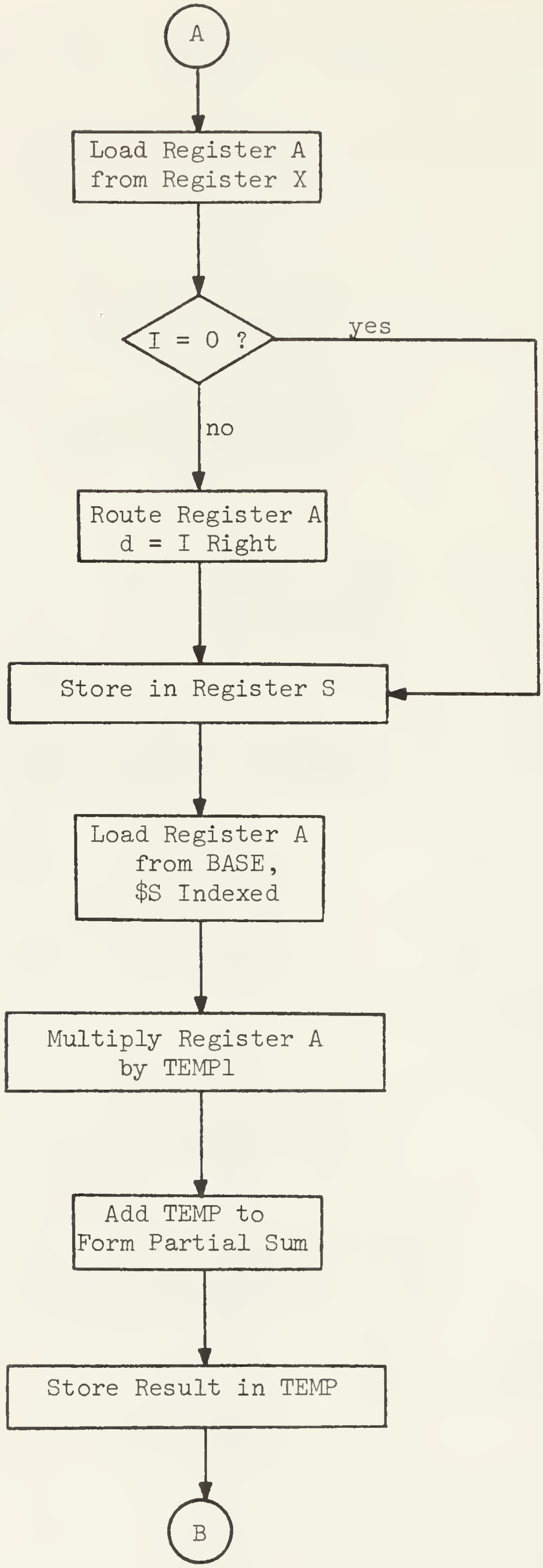
e) <u>SHUFL</u>: This Part brings the 2nd-Row to the Bottom of the Matrix and the 2nd-Column to the Far Right and Adjusts the Rest of the Matrix

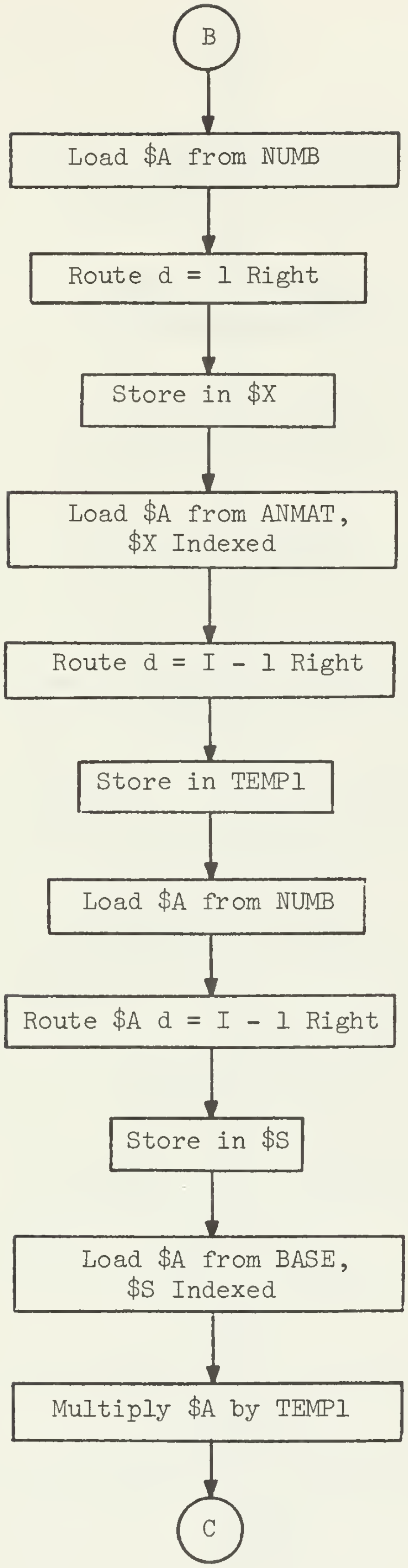


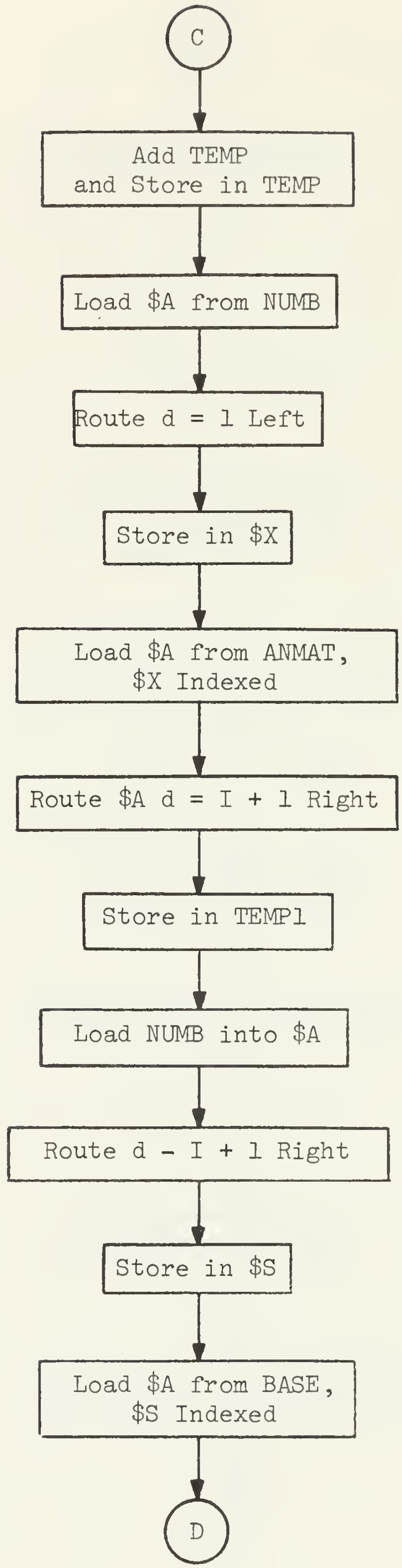

A
Store Saved 2nd Row in Row I
Skew the Matrix
Save Second Column
I = 0
Route Row I d = 1 left; only N - 2 elements to the right of column 2 participate in this part
I = I + 1
I = N - 1 ?
(SHZ)
no
yes
Bring Saved 2nd Column into Row I
Unskew the Matrix
RETURN

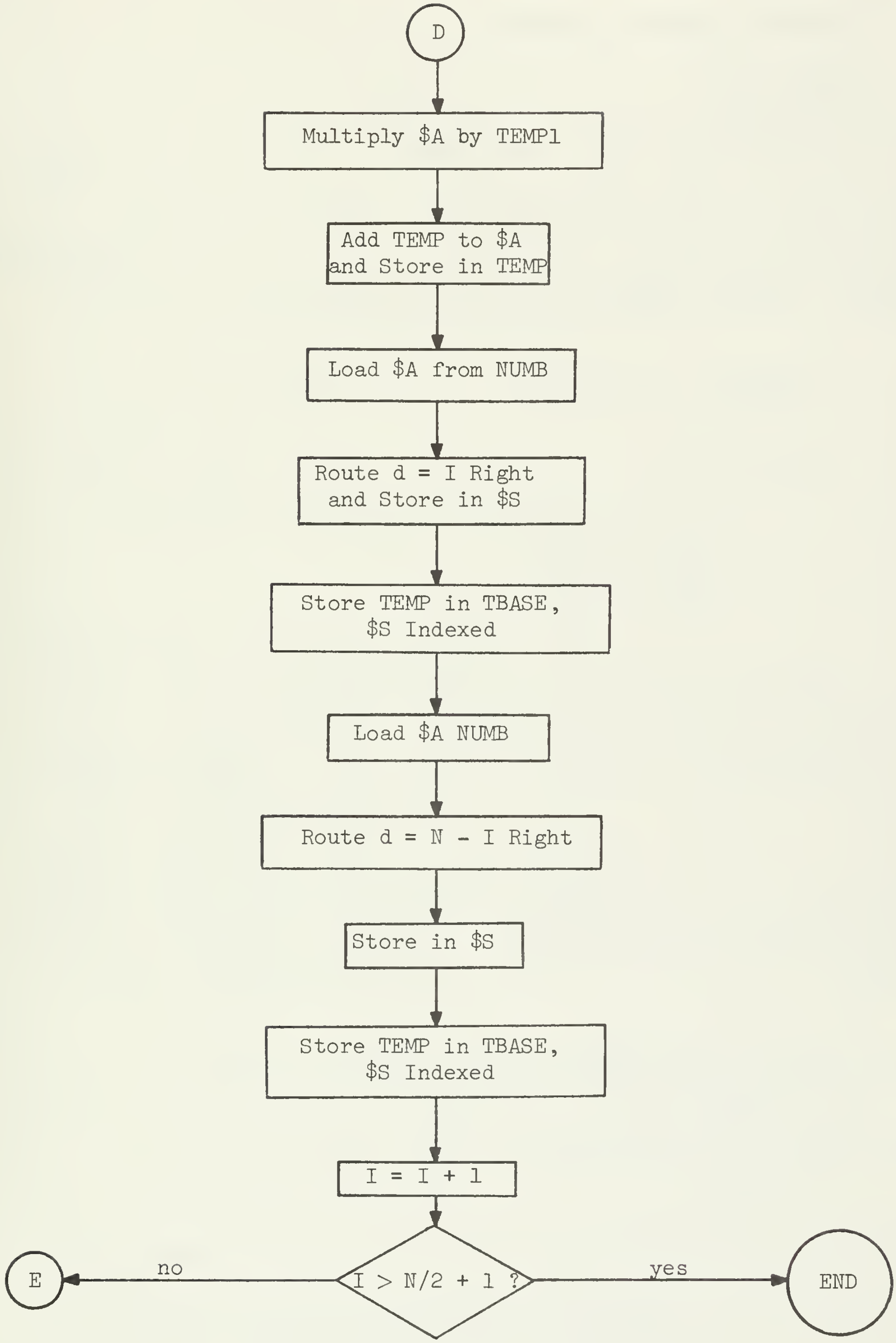
) <u>SAMUL</u>: Multiplying Two Matrices BASE = $(ANMAT)^t \cdot (BASE)^1$




A
Load Register A from Register X
I = 0 ?
yes
no
Route Register A d = I Right
Store in Register S
Load Register A from BASE, $S Indexed
Multiply Register A by TEMP1
Add TEMP to Form Partial Sum
Store Result in TEMP
B


B
Load $A from NUMB
Route d = 1 Right
Store in $X
Load $A from ANMAT,
$X Indexed
Route d = I - 1 Right
Store in TEMP1
Load $A from NUMB
Route $A d = I - 1 Right
Store in $S
Load $A from BASE,
$S Indexed
Multiply $A by TEMP1
C


C
Add TEMP
and Store in TEMP
Load $A from NUMB
Route d = 1 Left
Store in $X
Load $A from ANMAT,
$X Indexed
Route $A d = I + 1 Right
Store in TEMP1
Load NUMB into $A
Route d - I + 1 Right
Store in $S
Load $A from BASE,
$S Indexed
D


D
Multiply $A by TEMP1
Add TEMP to $A
and Store in TEMP
Load $A from NUMB
Route d = I Right
and Store in $S
Store TEMP in TBASE,
$S Indexed
Load $A NUMB
Route d = N - I Right
Store in $S
Store TEMP in TBASE,
$S Indexed
I = I + 1
I > N/2 + 1 ?
no
E
yes
END

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 (BASE') = BASE · ANMAT follows the same pattern as under Jacobi, however the transformation $(\text{ANMAT})^t \cdot (\text{BASE}')$ = 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 x 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, ..., N - 2 of (BASE')

Step 2: Multiply simultaneously $\sinh\psi_k$ by row I + 1, I = 0, 2, 4, ..., N - 2 of (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 · N multiplication for the transformation BASE = $(\text{ANMAT})^t$ · (BASE'). The total amount of multiplications for BASE = $(\text{ANMAT})^t$ · BASE · ANMAT is 2N + 3N = 5N. For a 64 x 64 matrix this means 320 instead of the 8192 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-$47. 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 = 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 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 sign c_{00} = 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:

$$h = \left[4\, c^2_{2k-1,2k} + (c_{2k-1,2k-1} - c_{2k,2k})^2\right]^{1/2}$$

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 larges 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 BASE = $(\text{ANMAT})^t \cdot$ BASE $\cdot$ ANMAT takes place in the following sequence of procedures:

9. MULTPL: (BASE') = BASE · 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. MULSA: BASE = $(\text{ANMAT})^t \cdot$ (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 $N^2(A)$ = FINVAL. A check to determine if BASE can be made diagonal, i.e., whether tanh4 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 tanh4 is equal to 1, we form 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 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 sign (a_{pq}) = 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:

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

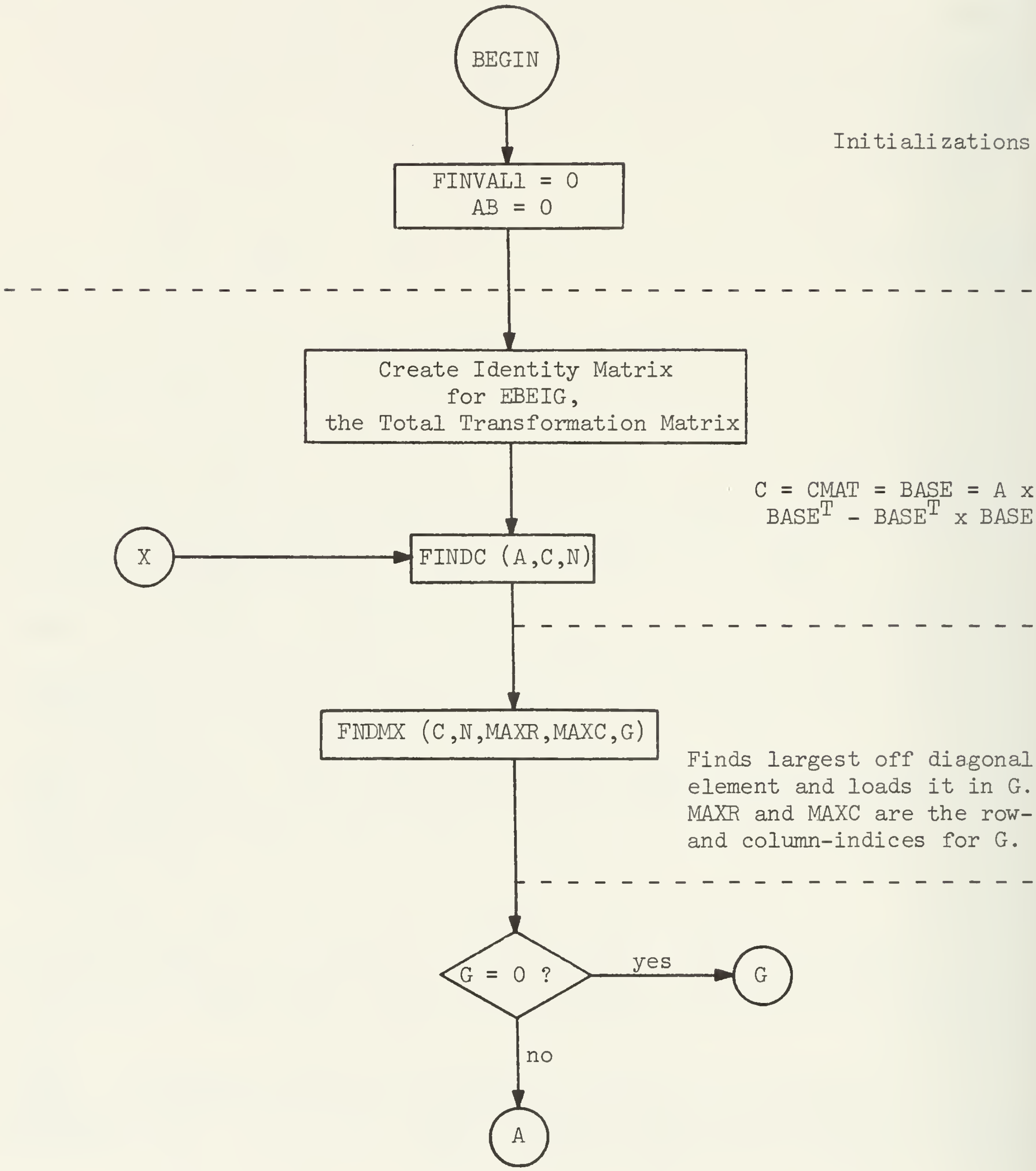
The last results are returned to the error-matrix EPS. If, however, sign (a_{pq}) = -sign (a_{qp}), i.e., if a difference in sign occurs for some a_{pq},

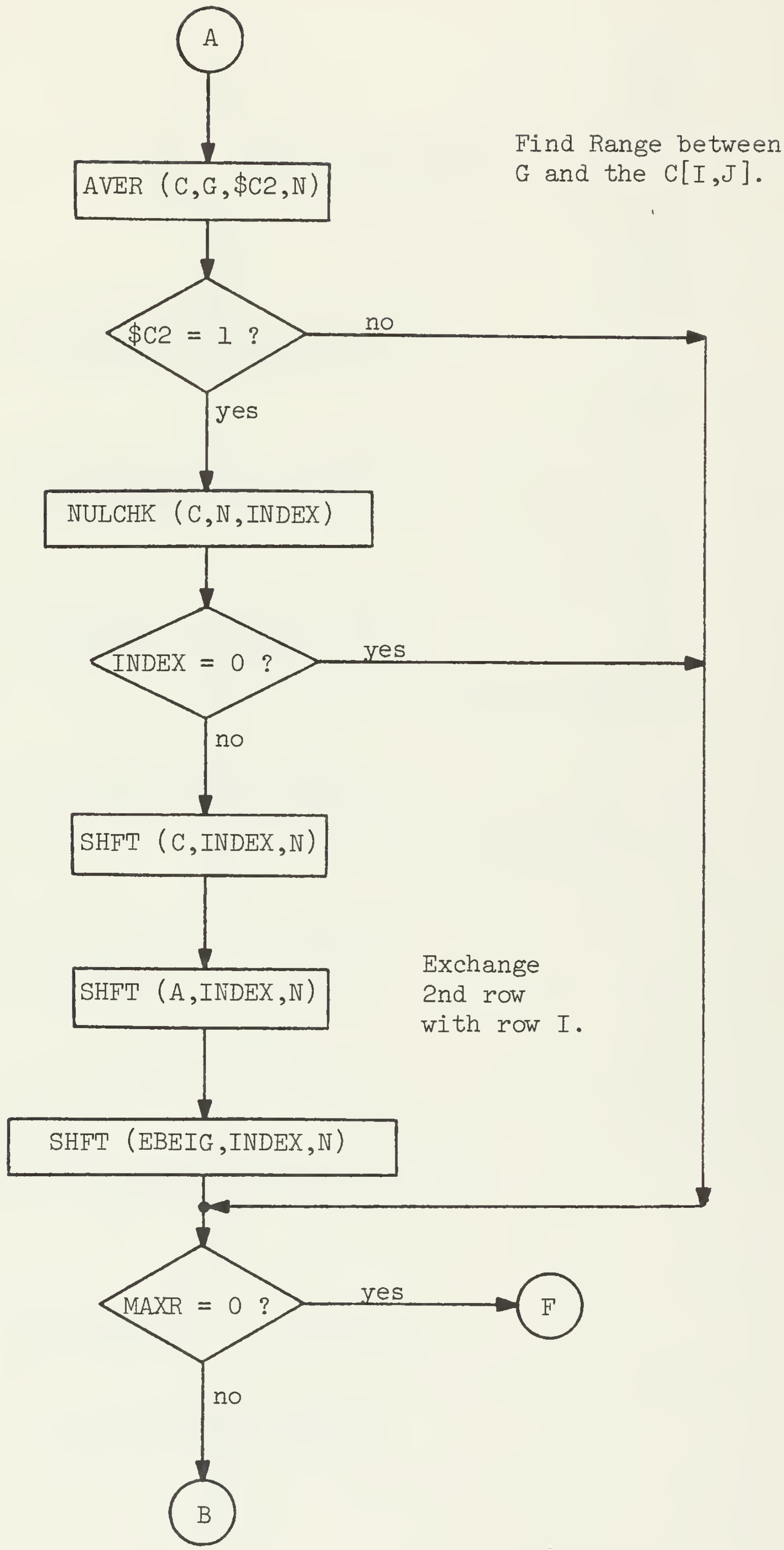
15. ASYM: is entered, which also averages the off-diagonal elements by computing $(a_{ij} + a_{ji})/2$. Here, too, an error-matrix is created as under 13.

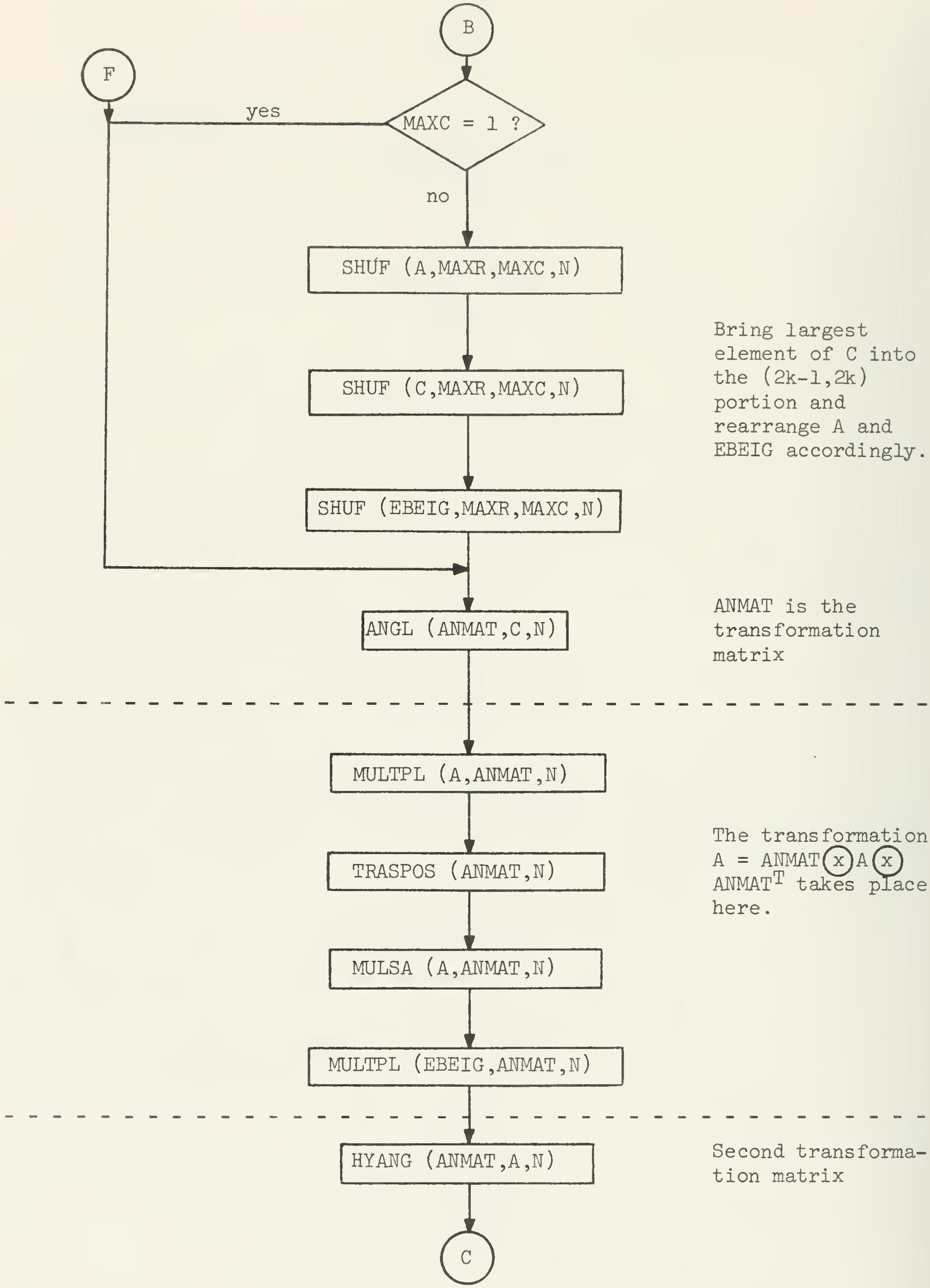
The subroutine is then left with a flag in $C2, to signal the symmetric case ($a_{pq} = a_{qp}$) or the asymmetric case ($a_{pq} = -a_{qp}$). From the content of $C2, $C2 = 0 or $C2 = 1 respectively, a decision can be made to enter EIGEN, the subroutine for finding eigenvalues and eigenvectors of a real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

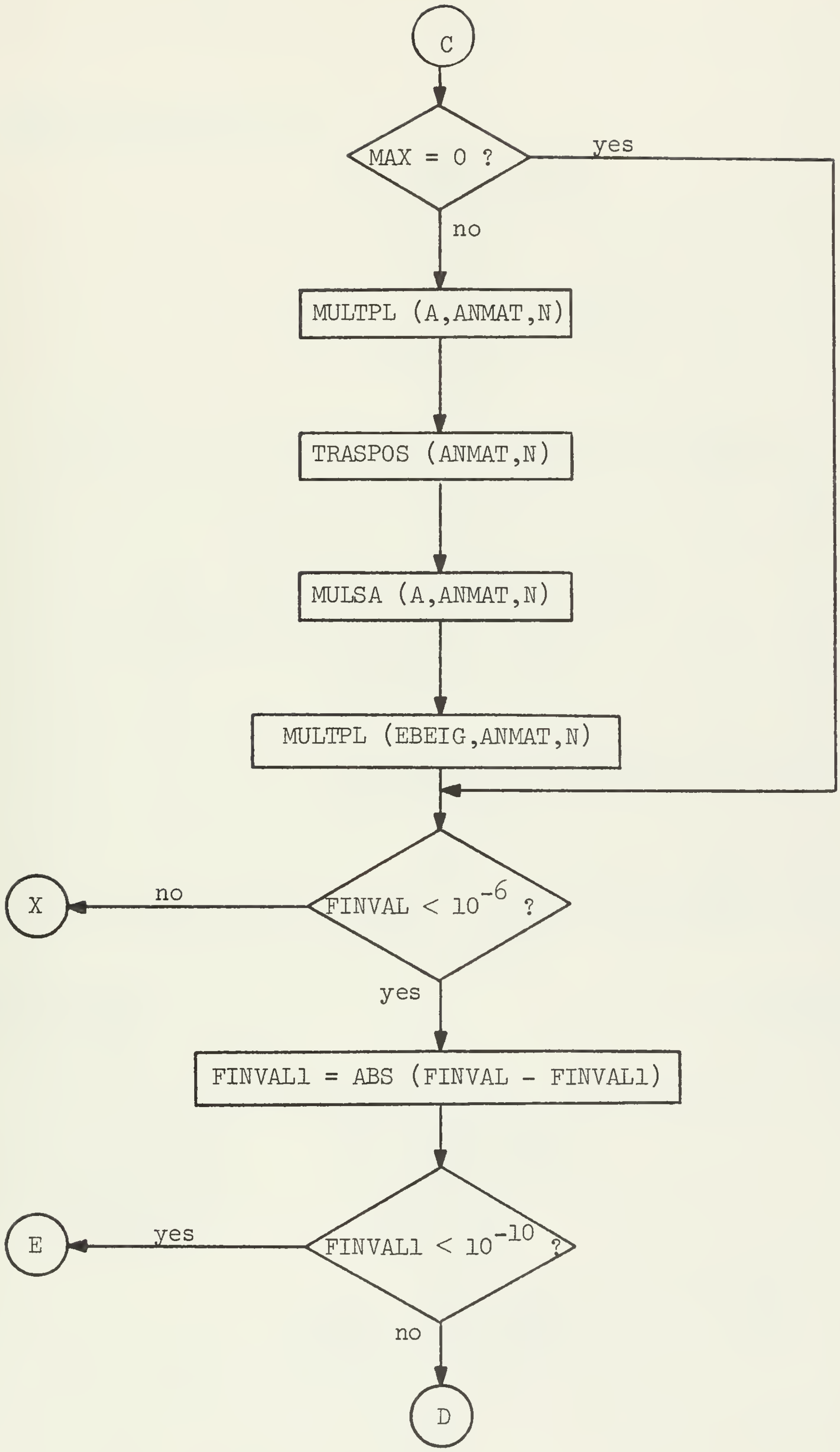
3. Flowcharts

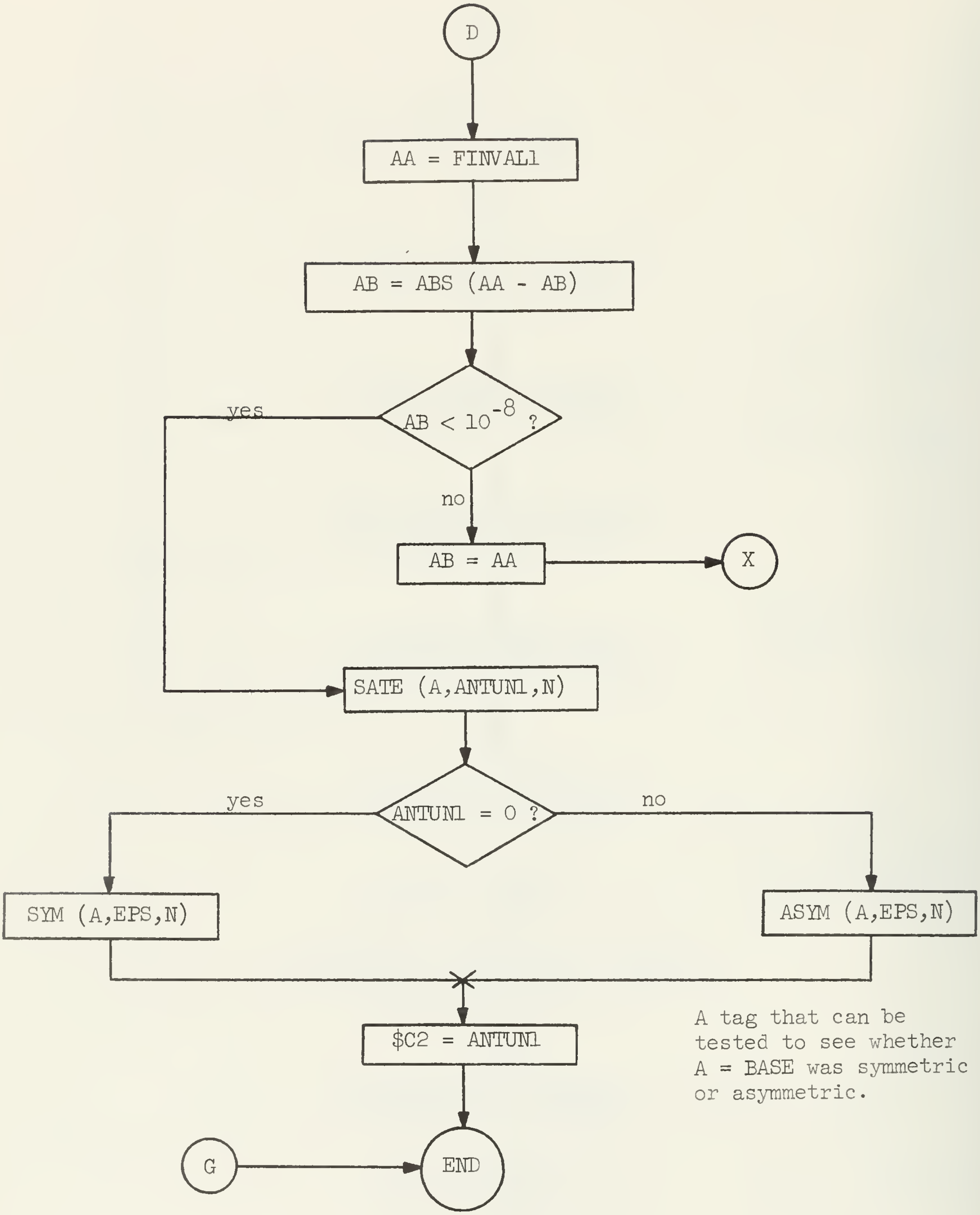
Flowchart 3. Eberlein's Algorithm (MAINPROGRAM)




A
Find Range between
G and the C[I,J].
AVER (C,G,$C2,N)
$C2 = 1 ?
no
yes
NULCHK (C,N,INDEX)
INDEX = 0 ?
yes
no
SHFT (C,INDEX,N)
SHFT (A,INDEX,N)
Exchange
2nd row
with row I.
SHFT (EBEIG,INDEX,N)
MAXR = 0 ?
yes
F
no
B


B
F
yes
MAXC = 1 ?
no
SHUF (A,MAXR,MAXC,N)
SHUF (C,MAXR,MAXC,N)
SHUF (EBEIG,MAXR,MAXC,N)
Bring largest element of C into the (2k-1,2k) portion and rearrange A and EBEIG accordingly.
ANGL (ANMAT,C,N)
ANMAT is the transformation matrix
MULTPL (A,ANMAT,N)
TRASPOS (ANMAT,N)
MULSA (A,ANMAT,N)
MULTPL (EBEIG,ANMAT,N)
The transformation A = ANMAT (x) A (x) ANMAT^T takes place here.
HYANG (ANMAT,A,N)
Second transformation matrix
C

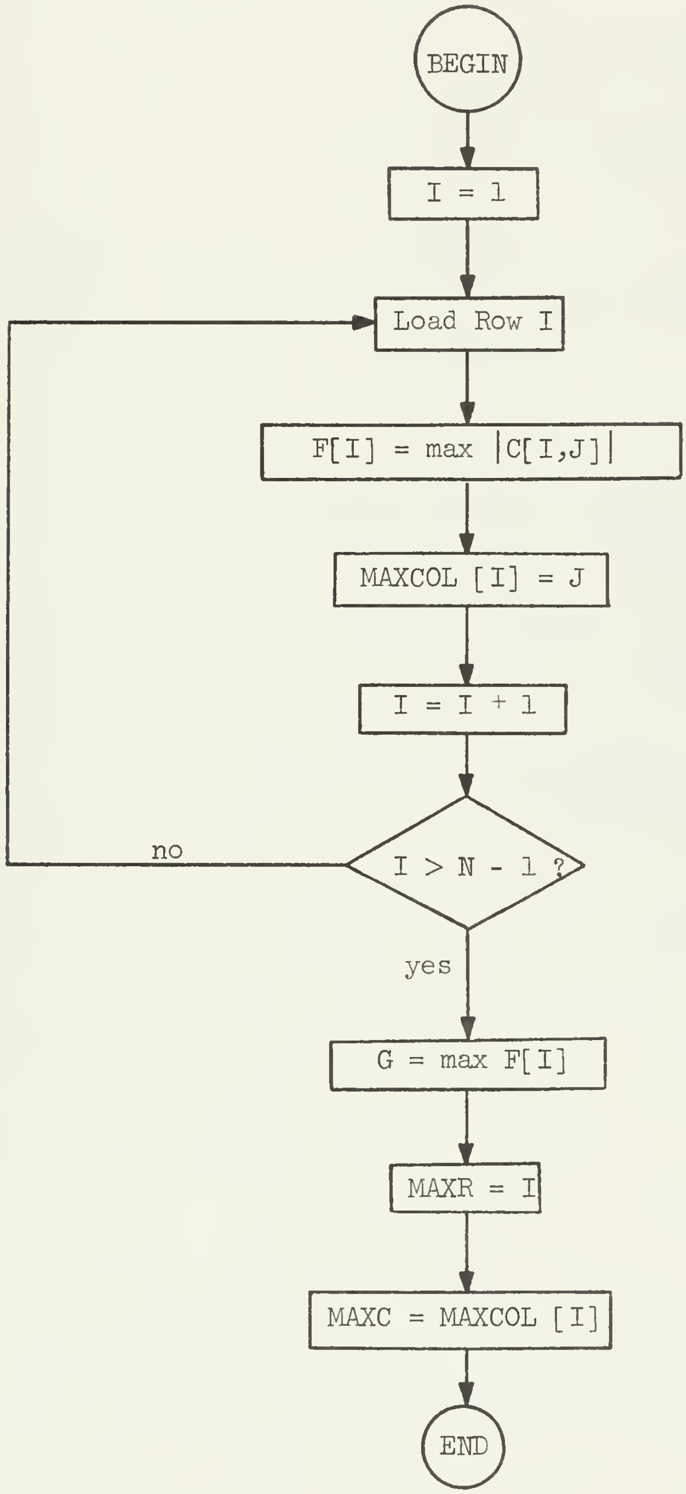

C
MAX = 0 ?
yes
no
MULTPL (A,ANMAT,N)
TRASPOS (ANMAT,N)
MULSA (A,ANMAT,N)
MULTPL (EBEIG,ANMAT,N)
FINVAL < 10^-6 ?
no
X
yes
FINVAL1 = ABS (FINVAL - FINVAL1)
FINVAL1 < 10^-10 ?
yes
E
no
D



A tag that can be tested to see whether A = BASE was symmetric or asymmetric.

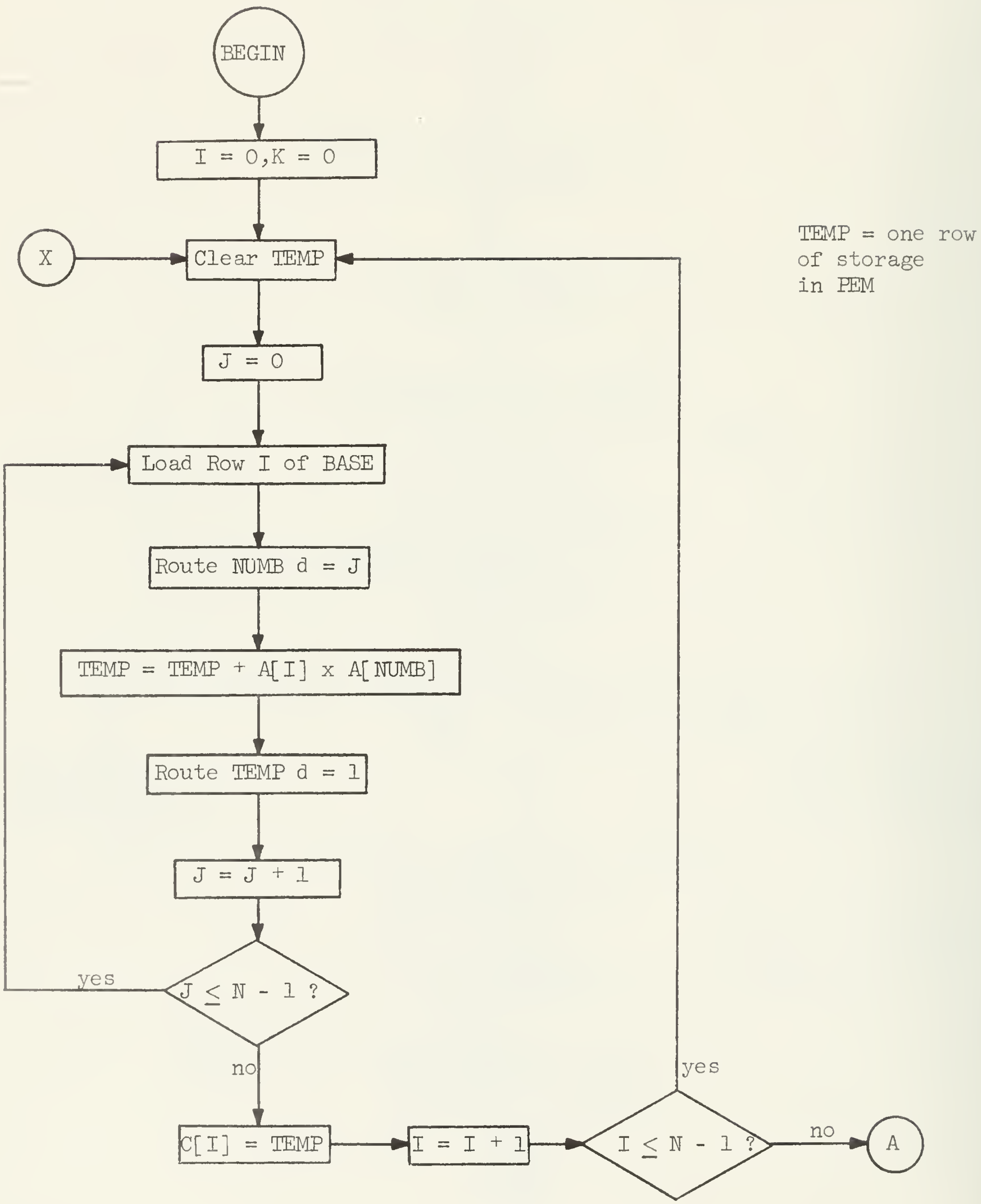
Flowchart 4. Procedures to Eberlein's Method

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

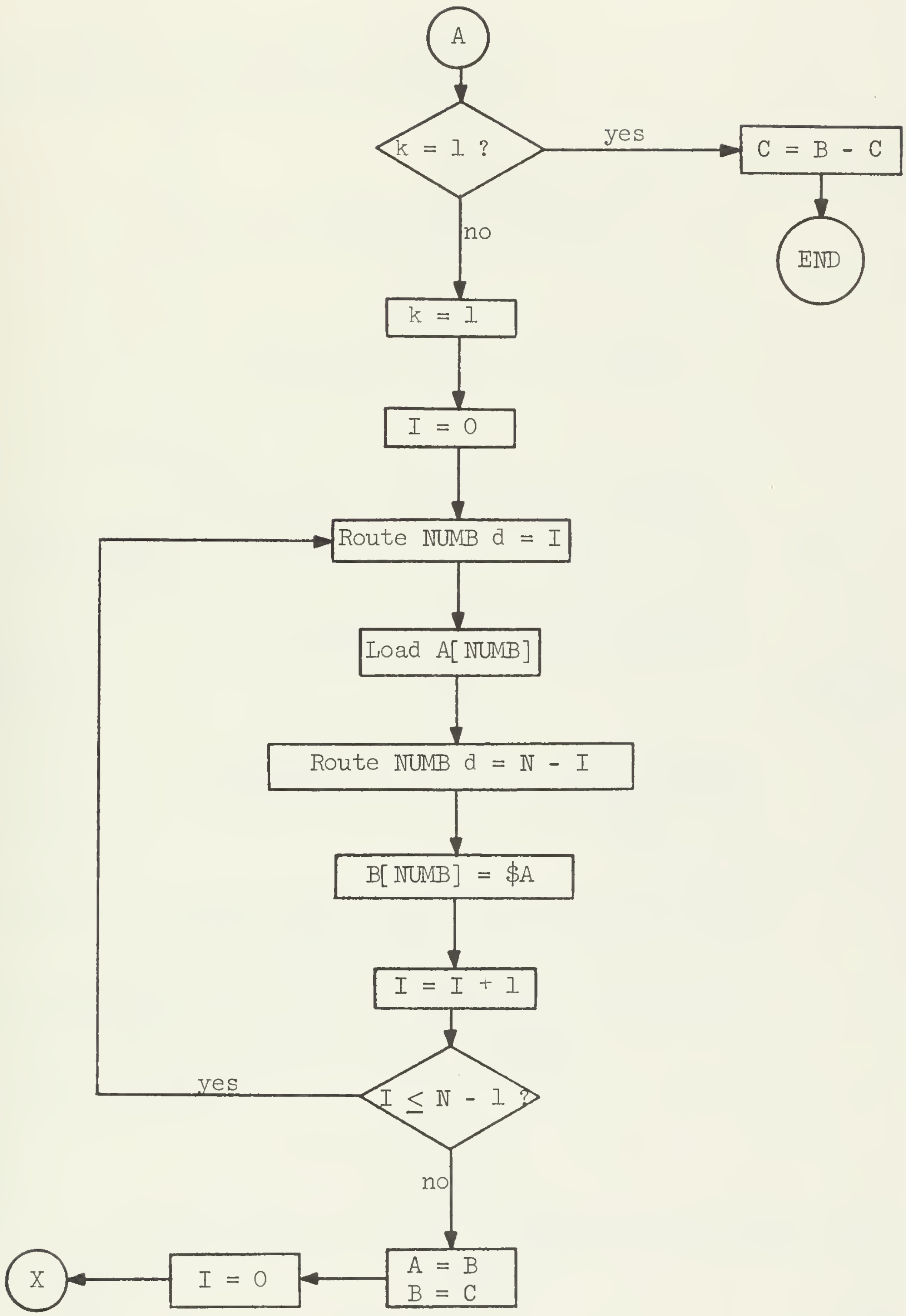


b) FINDC (A,C,N)

i) MLTRPS

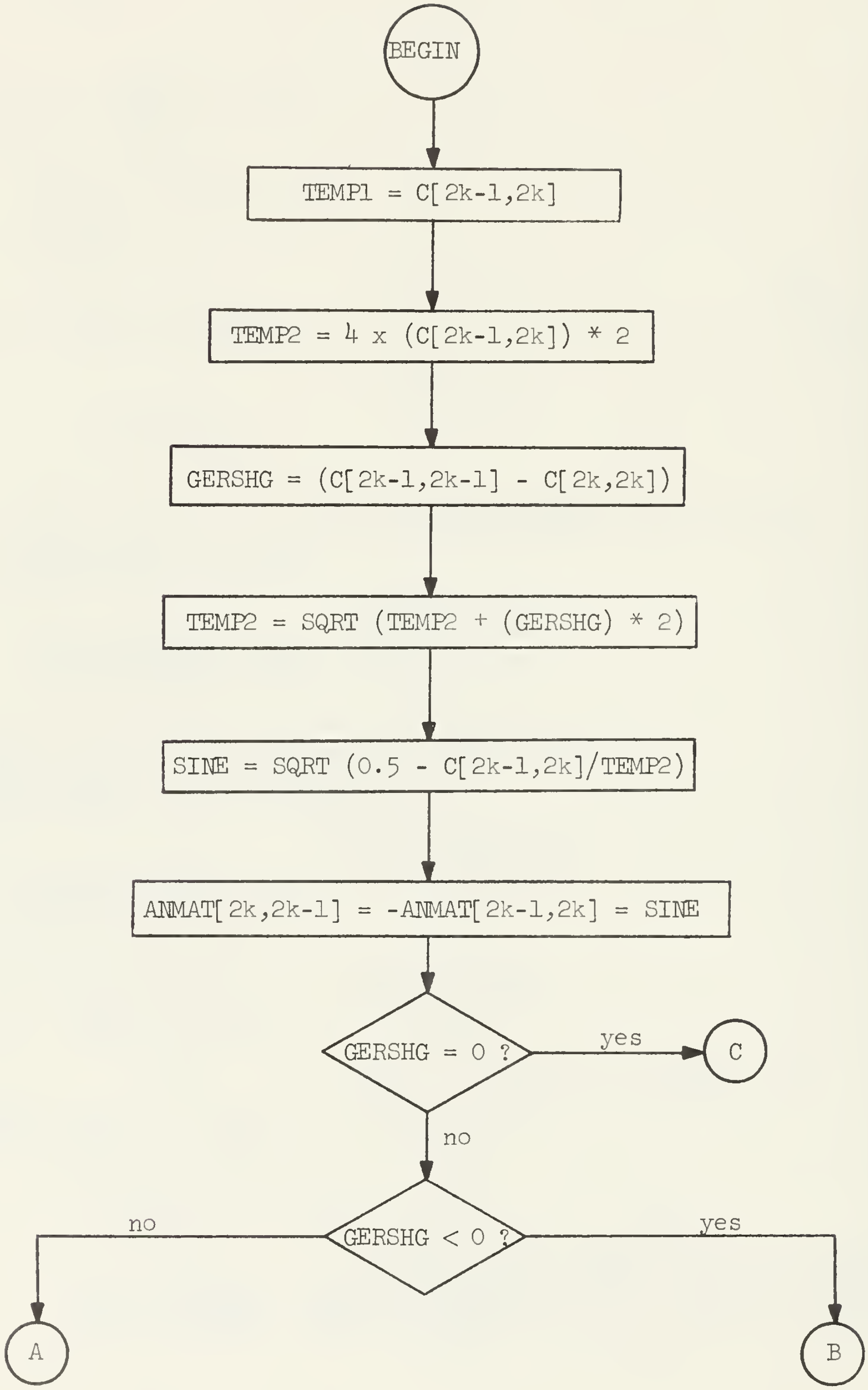


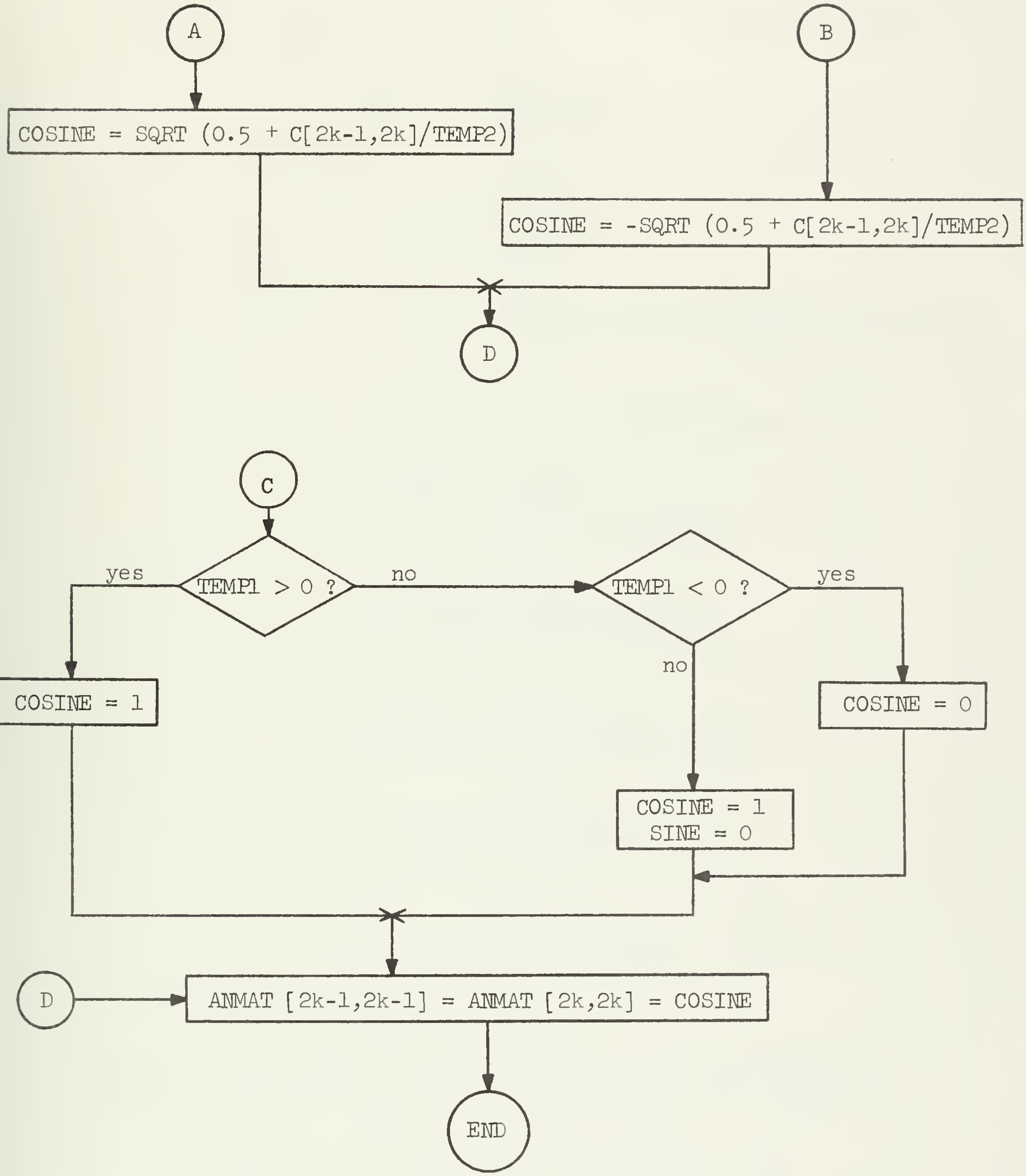
ii) TRPS (A,B,N)



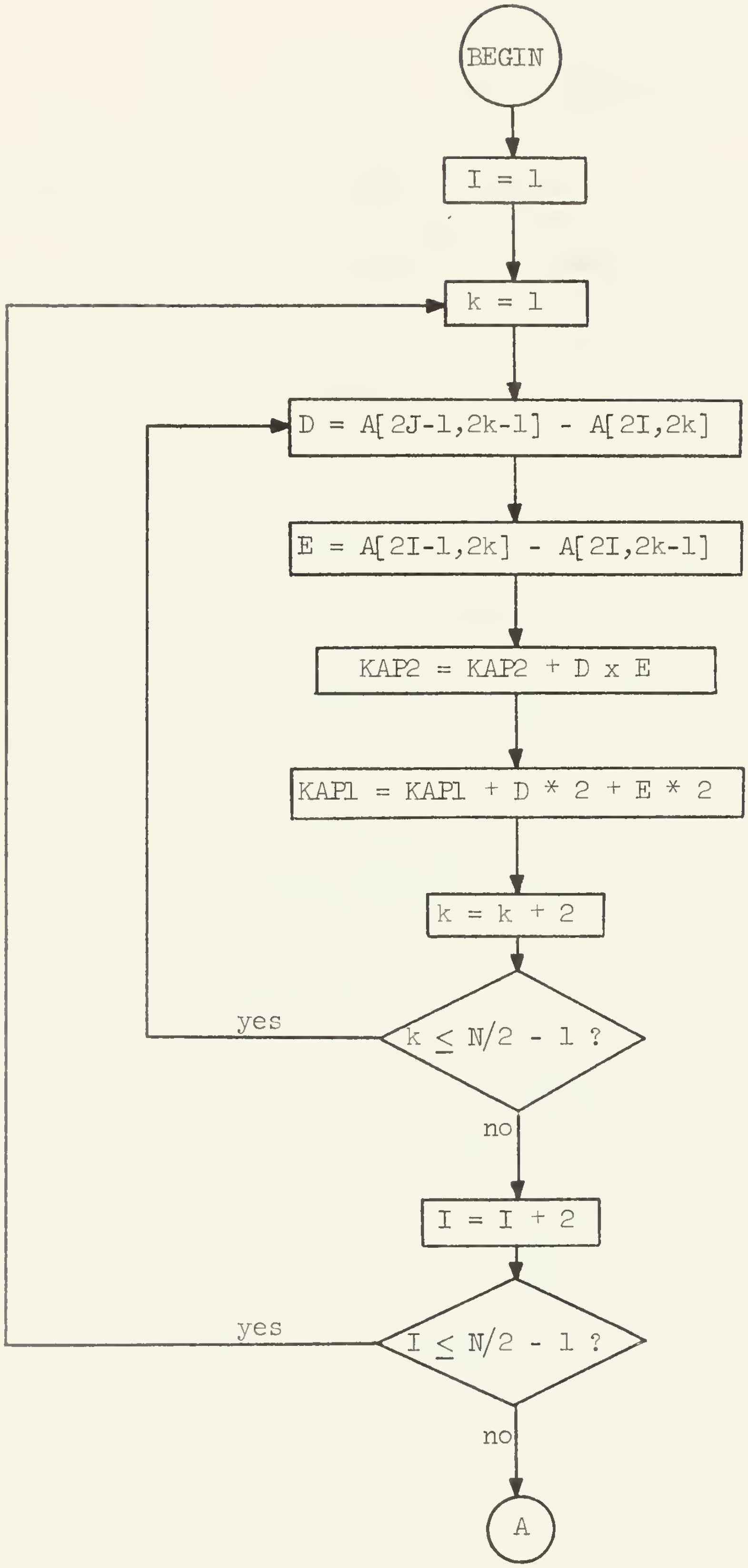
A
k = 1 ?
yes
C = B - C
END
no
k = 1
I = 0
Route NUMB d = I
Load A[NUMB]
Route NUMB d = N - I
B[NUMB] = $A
I = I + 1
I ≤ N - 1 ?
yes
no
A = B
B = C
I = 0
X

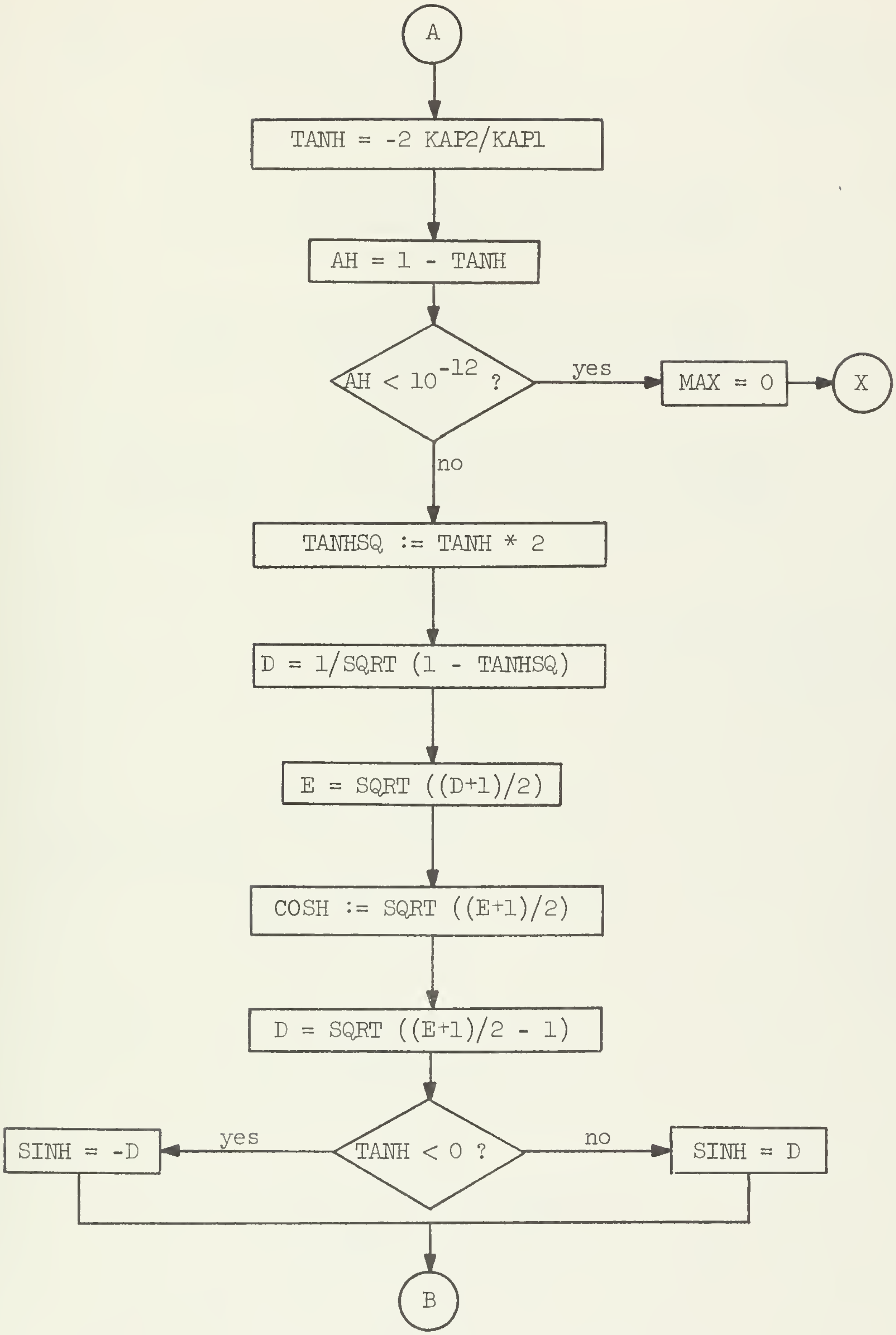
c) ANGL (ANMAT,C,N)

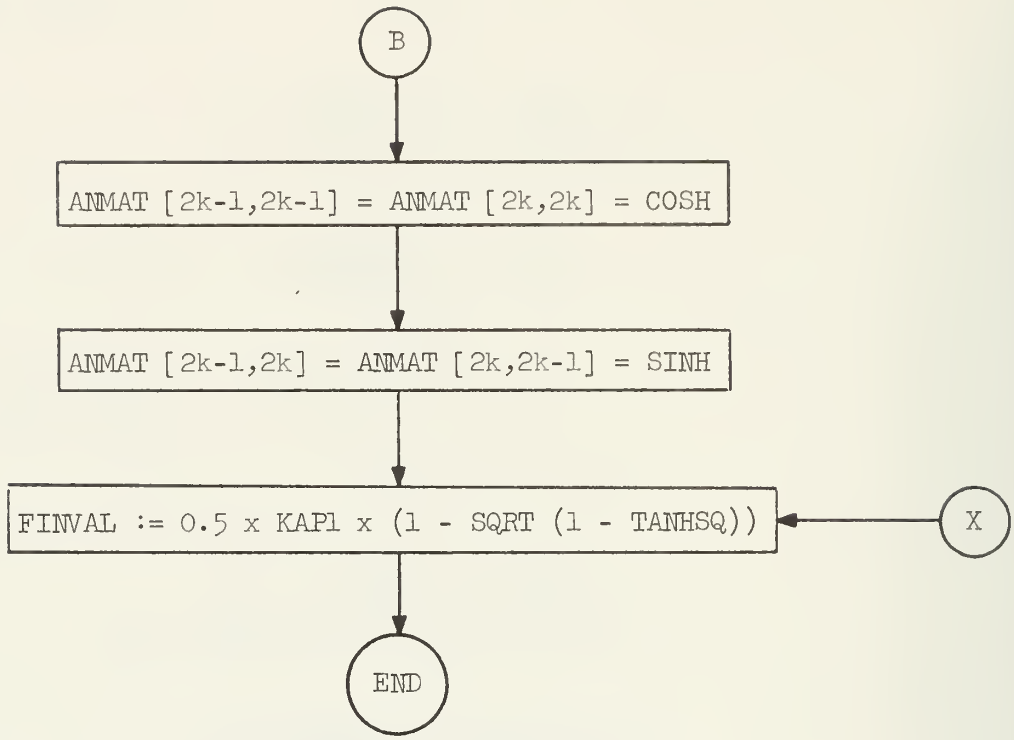



A
B
COSINE = SQRT (0.5 + C[2k-1,2k]/TEMP2)
COSINE = -SQRT (0.5 + C[2k-1,2k]/TEMP2)
D
C
yes
TEMP1 > 0 ?
no
TEMP1 < 0 ?
yes
no
COSINE = 1
COSINE = 0
COSINE = 1
SINE = 0
D
ANMAT [2k-1,2k-1] = ANMAT [2k,2k] = COSINE
END

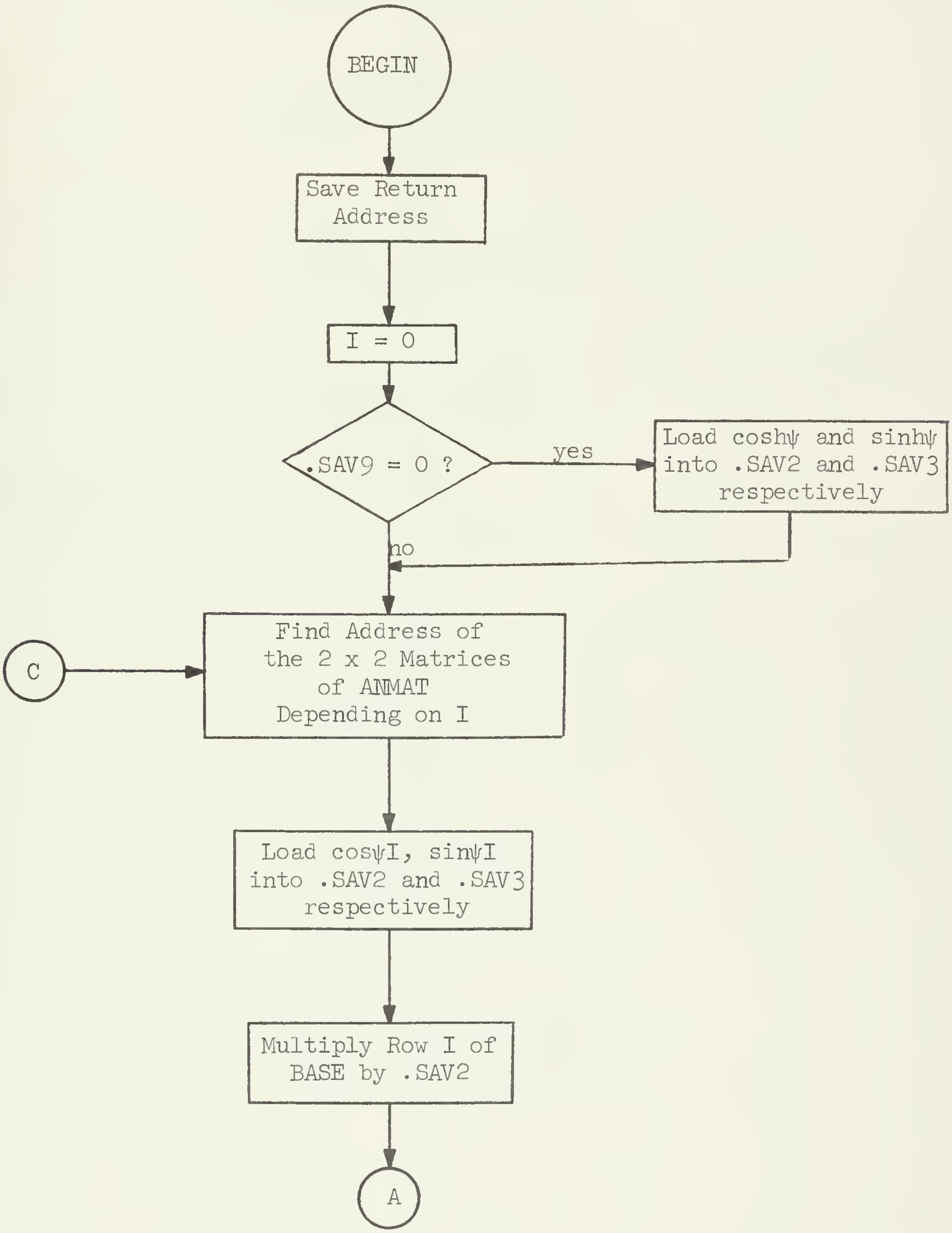
d) HYANG (ANMAT, A, N)

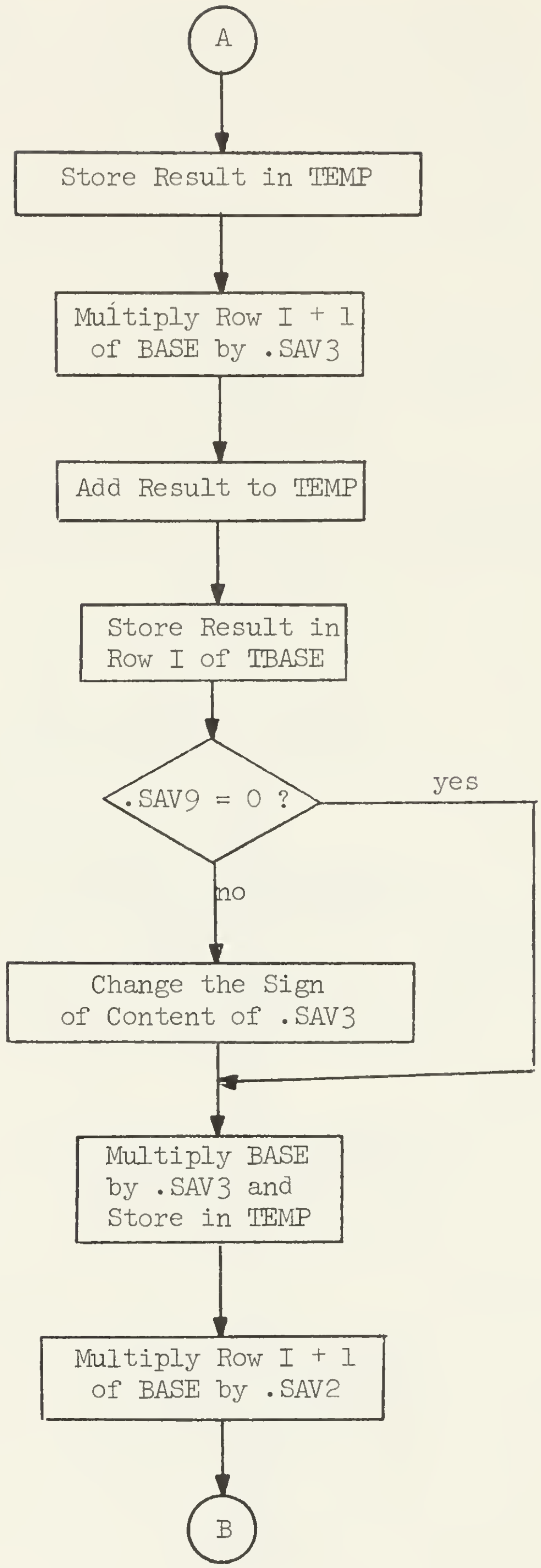


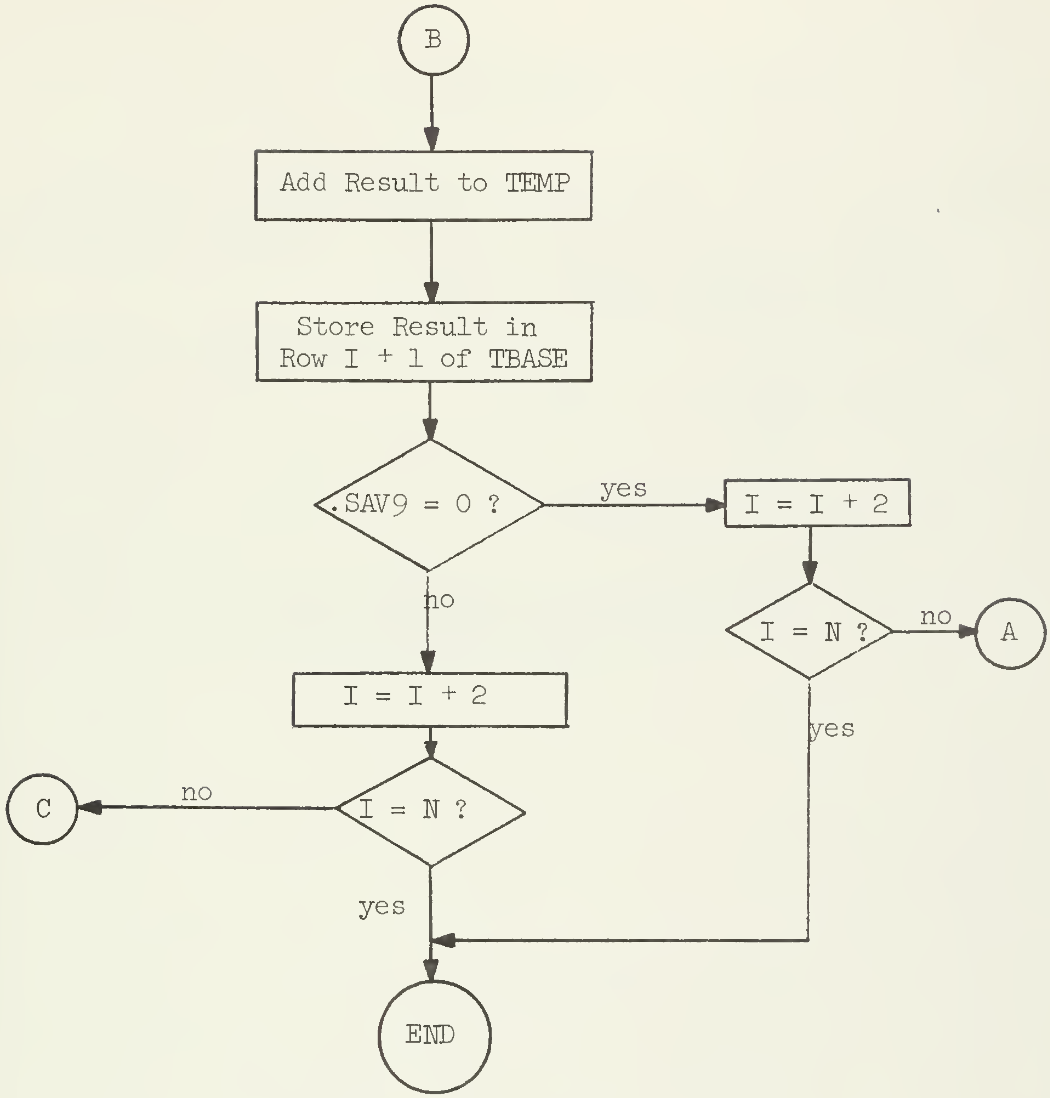

A
TANH = -2 KAP2/KAP1
AH = 1 - TANH
AH < 10^{-12} ?
yes
MAX = 0
X
no
TANHSQ := TANH * 2
D = 1/SQRT (1 - TANHSQ)
E = SQRT ((D+1)/2)
COSH := SQRT ((E+1)/2)
D = SQRT ((E+1)/2 - 1)
TANH < 0 ?
yes
SINH = -D
no
SINH = D
B


B
ANMAT [2k-1,2k-1] = ANMAT [2k,2k] = COSH
ANMAT [2k-1,2k] = ANMAT [2k,2k-1] = SINH
FINVAL := 0.5 x KAP1 x (1 - SQRT (1 - TANHSQ))
X
END

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.




A
Store Result in TEMP
Multiply Row I + 1
of BASE by .SAV3
Add Result to TEMP
Store Result in
Row I of TBASE
.SAV9 = 0 ?
yes
no
Change the Sign
of Content of .SAV3
Multiply BASE
by .SAV3 and
Store in TEMP
Multiply Row I + 1
of BASE by .SAV2
B


B
Add Result to TEMP
Store Result in Row I + 1 of TBASE
.SAV9 = 0 ?
yes
I = I + 2
I = N ?
no
A
yes
no
I = I + 2
I = N ?
no
C
yes
END

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 βNAME(BPARAMETERS)=
βIF βSIGN(βMFIELD(βNAME)) β THEN
EXTERNAL βNAME; βFI
βIF βEMPTY(βPARAMETERS) βTHEN βELSE
BEGIN BLOCK
BEGIN USE (63)
LIST:  DATA βPARAMETERS
END;
CLC(2);
SLIT(2) LIST;
END; βFI
CLC(3);
SLIT(3) βNAME;
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,0,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

```
          BEGIN
          FILL      128;
%%%%%%%%%%%%%%%%%%% CALL - DEFINITION %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  DEFINE CALL &NAME(&PARAMETERS)=
          &IF &SIGN(&MFIELD(&NAME)) &THEN
          EXTERNAL  &NAME; &FI
          &IF &EMPTY(&PARAMETERS) &THEN &ELSE
          BEGIN BLOCK
            BEGIN USE (63)
              LIST:  DATA  &PARAMETERS
            END;
            CLC(2);
            SLIT(2) LIST;
          END; &FI
          CLC(3);
          SLIT(3)   &NAME;
          EXCHL(3)  $ICR;##;
%%%%%%%%%%%%%%%% END DEFINE %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEFINE   WRTPEM=
%         CLC(1);
%         CADD(1)    $C3;
%         CADD(1)    $D3;
%         CSHL(1)    24;
%         CADD(1)    $C3;
% .       CCB(1)     15;
%         CLC(0);
%         CADD(0)    $D3;
%         CSHL(0)    24;
%         CCB(0)     15;
%         DISPLAYR   $C1,-16;
%         LIT(2)     =64;
%         CADD(1)    $C2;
%         CROTR(1)   24;
%         CADD(1)    $C2;
%         CROTL(1)   24;
%         TXEEM(0)   ,-9;##;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%    SINCE INDIRECT ADRESSING IS USED LET
%%%%    BASE:    THE MATRIX FROM WHICH THE EIGENVALUES
%%%%             ARE BEING FOUND.
%%%%    EIGV:    THE EIGENVECTORMATRIX
%%%%    ANMAT:   THE ANGLE MATRIX
%%%%    TBASE:  A TEMPORARY STORAGE MATRIX
%%%%    ALL THESE MATRICES NEED TO BE BY ADRESS PASSED
%%%%    AS PARAMETERS FROM THE OUTSIDE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
.ZERO:    EQU        $D0;%        FIXED POINT ZERO.
.ONE:     EQU        $D1;%        FIXED POINT ONE.
.N:       EQU        $D2;%        ORDER OF MATRIX
.NMO:     EQU        $D3;%        N-1;
.ENB:     EQU        $D4;%        1000000000000000000000018,ENABLING ONE PE
.SPEC:    EQU        $D5;%        ENABLING PATTERN FOR THE FIRT N PE S.
.ROUT:    EQU        $D6;%        64-N, CONSTANT USED IN END AROUND ROUTING
.MAX:     EQU        $D7;%        ROW INDEX FOR MAX. VAL. FOUND IN RWSM.
.ADRES:   EQU        $D8;%        ADDRESS SAVED HERE
.ADRES1:  EQU        $D9;%
.ADRES2:  EQU        $D10;
.KSI:     EQU        $D11;%       CONVERGENCE FACTOR
.ANTUN:   EQU        $D12;%       TURN-ON PATTERN FOR THE ANGLE ROUTINE
.ANTUN1:  EQU        $D13;
.BOUND:   EQU        $D14;%       BOUND OF CONVERGENCE FOR INTERMED. CASE.
```

```
.ZEZE:   EQU      $D15;%      THRESHOLD FACTOR TO BE CHECKED AGAINST BD
.BD:     EQU      $D16;%      THRESHOLD FOR SUPER-DIAGONALS.
.BD1:    EQU      $D17;
.BD2:    EQU      $D18;
.BD3:    EQU      $D19;
.INDEX:  EQU      $D20;%      INNER LOOP COUNT IN MAIN PROGRAM.
.S:      EQU      $D21;%      CONVERGENCE-FACTOR FOUND IN ADDIT-ROUTINE.
.CONVE:  EQU      $D22;%      CHECK FOR SUPERDIAGS EQL 0.
.SAV1:   EQU      $D24;%      SAVE REGISTER.
.SAV2:   EQU      $D25;
.SAV3:   EQU      $D26;
.SAV4:   EQU      $D27;
.SAV8:   EQU      $D28;
.SAV5:   EQU      $D29;
.SAV9:   EQU      $D30;
.RETUR:  EQU      $D31;%      RETUR CONTAINS THE RETURN ADR.
%                             TO LINK TO THE OUTSIDE.
.ADRA:   EQU      $D32;%      ADRESS OF ORIGINAL MATRIX
.ADRB:   EQU      $D33;%      ADRESS OF EIGENVECTORMATR.
.ADRC:   EQU      $D34;%      ANGLE MATRIX
.ADRD:   EQU      $D35;%      TEMP. STORAGE MATRIX
.ADRE:   EQU      $D36;%      ADRESS OF THE ERROR MATRIX
.SAV6:   EQU      $D37;
TEMP:    BLK      1;%         ONE ROW OF SAVE-STORAGE IN PE-MEMORY
TEMP1:   BLK      1;
TEMP2:   BLK      1;
MANT:    BLK      1;
GERSHG:  BLK      1;
NUMB:    DATA     0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16;
         DATA     17,18,19,20,21,22,23,24,25,26,27,28,29,30;
         DATA     31,32,33,34,35,36,37,38,39,40,41,42,43,44;
         DATA     45,46,47,48,49,50,51,52,53,54,55,56,57,58;
         DATA     59,60,61,62,63;%     PE NUMBERING
MESSO:   DATA  "### PE I CONTAINS BOUND ON EIGENVAL. I ###";
TACK:    WDS      0;
%MESS1:  DATA  "### BASE AFTER ANY ROW/COL. SHUFFLE ###";
%MESS2:  DATA  "### THE TRANSFORMATIONMATRIX - ANMAT ###";
%MESS3:  DATA  "### BASE AFTER THE TRANSFORMATION ###";
%MESS4:  DATA  "### BASE AFTER 2-ND ROW/COL. SHUFFLE ###";
ADBSAV:  WDS      10;
% ***********************************************************************
ROUTE:   FILL;%                ROUTE-ROUTINE FOR N=64
         STL(3)   .SAV3;%      SAVE RETURN ADDRESS.
         LDL(3)   .SAV2;%      LOAD ROUTING DISTANCE.
         RTL      $A,0(3);%    $A IS ASSUMED TO CONTAIN THE ELEMENTS
%                             TO BE ROUTED.
         LDL(3)   .SPEC;%
         LDEE1    $C3;
         LDA      $R;
%##########################################################
         LDS      NUMB;%     THIS PART CLEANS THE FIRST D PE"S
         LDL(3)   .SAV2;%    OF RGA, IN CASE ELEMENTS HAVE BEEN
         ISL      $C3;%      ROUTED INTO THOSE PE"S. FROM ACCROSS
         SETE     I.AND.E;%  THE BOUNDARY N.
         SETE1    E.AND.E;
         CLRA;
         LDL(3)   .ROUT;%    BRING THOSE ELEMENTS WHICH WERE ROUTED
         RTL      0(3);%     ACROSS BOUNDARY INTO THE FIRST D PE"S
         LDA      $R;
         LDL(3)   .SPEC;
         COMPC(3);
```

```
        LDEE1       $C3;
        CLRA;%                  CLEAR RGA PAST THE BOUNDARY N
        COMPC(3);
        LDEE1       $C3;
%#############################################################
        LDL(3)      .SAV3;%     AND THE RESULT WILL BE IN $A PROPERLY
        EXCHL(3)    $ICR;%      ROUTED.THEN RETURN.
%  ******************************************************************
ROTAR:  FILL;
%  ADJUST ACAR1 FOR AN END-AROUND SHIFT,RIGHT FOR D LSS 64
        STL(3)      .SAV3;%  SAVE RETURN ADRESS
        LDL(1)      .SAV5;%     PATTERN THAT NEEDS ADJUSTMENT
        CROTR(1)     0(0);%     ACARO CONTAINS D
        LDL(2)      .SPEC;
        CSHR(2)     0(0);
        CAND(2)     $C1;
        CAND(2)     $D5;
        CEXOR(1)    $C2;
        LDL(3)      .N;
        CROTL(1)    0(3);
        CEXOR(1)    $C2;
        STL(1)      .SAV8;
        LDL(3)      .SAV3;
        EXCHL(3)    $ICR:
%#########################################################
ROTAL:  FILL;
%  ADJUST ACAR1 FOR AN END-AROUND SHIFT,LEFT FOR D LSS 64
        STL(3)      .SAV3;
        LDL(1)      .SAV5;
        CROTL(1)     0(0);
        LDL(3)      .SPEC;
        CSHL(3)     0(0);
        CAND(3)     $C1;
        CAND(3)     $D5;
        CEXOR(1)    $C3;
        LDL(2)      .ROUT;
        CSHL(1)     0(2);
        CEXOR(1)    $C3;
        STL(1)      .SAV8;
        LDL(3)      .SAV3:
        EXCHL(3)    $ICR;
%#0#######################################################
RWSM:   FILL;
%                               SUM UP IN ABS. VALUE EVERY ROW OF THE
%                               MATRIX BASE AND FIND FIRST THE MAXIMUM
%                               OF THE INDIVIDUAL SUMS AND THEN THE
%                               INDEX OF THE ROW WHERE THE MAXIMUM CAME
%                               FROM.
        STL(3)      .SAV1;%     SAVE RETURN ADDRESS.
        SETE        E.OR.-E;
        SETE1       E.AND.E;
        CLRA;
        LDS         $A;
        LIT(0)      =0,1,0;%    LOOP FOR PICKING UP THE ROWS OF THE MATRIX
        CADD(0)     $D3;
        CROTL(0)    24;
RSU1:   LDL(1)      .SPEC;%     THE ELEMENT A[I,I] DOES NOT PARTICIPATE
%                               IN THE SUMMING.THEREFORE WE TURN OFF
        CCB(1)      0(0);%     PE[I] BY THIS CONSTRUCT
        LDL(2)      .ADRA;
        CADD(2)     $C0;
```

```
        LDEE1     $C1;
        LDA       0(2);
        CLC(3);
        LDEE1     $C3;
        LIT(1)    =0.0;
        LDA       $C1;
        LDL(1)    .SPEC;
        LDEE1     $C1;
        ADRN      $S;%          FORM PARTIAL SUM. WE KNOW THE
%                               MATRIX IS SYMMETRIC
        LDS       $A;
        TXETM(0)  .+1;
        JUMP      RSU1;
        STS       TEMP;
        SETE      E.OR.-E;%     FIND MAXIMAL VALUE IN TEMP.
        SETE1     E.AND.E;
        LIT(2)    =0.0;
        LDS       $C2;
        LDA       $S;
        LDL(2)    .SPEC;
        LDEE1     $C2;
        LDA       TEMP;
        SETE      E.OR.-E;
        SETE1     E.AND.E;
        LDS       $A;
        LIT(1)    =1;
RSM:    RTL       $S,0(1);%     ROUTE $S IN POWERS OF TWO
        LDS       $R;
        IAL       $S;%          ELIMINATE SMALLER VALUES OF $A WHEN
        SETE      I.AND.E;%     COMPARED WITH $S.
        SETE1     E.AND.E;
        LDA       $S;
        SETE      E.OR.-E;
        SETE1     E.AND.E;
        LDS       $A;
        CADD(1)   $C1;
        LIT(2)    =64;
        EQLXF(1)  $C2,RSM;%     MAXIMAL VALUE IS FOUND, WHEN TEST FAILS.
%                               WHERE DID THE LARGEST VALUE COME FROM
%                               FIND ROW INDEX.
        LDL(2)    .SPEC;
        LDEE1     $C2;
        LDA       TEMP;%        AT THIS POINT EVERY PE CONTAINS THE
%                               MAXIMAL VALUE IN $S, WHILE THE FIRST N
%                               PE S CONTAIN THE INDIVIDUAL SUMS IN $A.
        IAL       $S;
        SETC(1)   I;
        COMPC(1);
        CAND(1)   $C2;
        LEADO(1);
        LIT(0)    =77;8;
        CAND(1)   $C0;
        STL(1)    .MAX;%        .MAX CONTAINS THE ROW INDEX WHERE LARGEST
%                               VALUE CAME FROM.
        LDL(3)    .SAV1;%       RETURN TO THE MAIN PROGRAM.
        EXCHL(3)  $ICR;
% ************************************************************************
ANYR:   FILL;
%                               ANYR BRINGS THE ROW WITH THE LARGEST SUM
%                               INTO THE TOP ROW AND PULLS ALL OTHER WITH
%                               IT IN A CIRCULAR MOTION.
```

```
        STL(3)     .SAV1;%     SAVE RETURN ADDRESS.
        LDL(2)     .N;
        CSUB(2)    $D7;%       N-.MAX. MAX IS THE ROW-INDEX FOR THE
%                              LARGEST ROW-SUM. N-MAX IS THE ROUTING
%                              DISTANCE.
        STL(2)     .SAV2;
        LDL(3)     .SPEC;
        LDEE1      $C3;
        LIT(0)     =0.1.0;%    BRING THE ELEMENTS INTO THEIR RESPECTIVE
        CROTL(0)   24;%        POSITIONS WITHIN THEIR ROWS.
        CADD(0)    $D3;
AY:     LDL(2)     .ADRES;
        CADD(2)    $C0;
        LDA        0(2);
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;% GO TO ROUTE ROUTINE
        STA        0(2);
        TXEFM(0)   .AY;%       THE ELEMENTS ARE IN THEIR NEW POSITIONS
        LIT(0)     =0;     %   NOW WE BRING THE ELEMENTS INTO THEIR
%                              FINAL LOCATION BY CIRCULAR MOTION
        LIT(1)     =0.1.0;
        CADD(1)    $D3;
        CROTL(1)   24;
        CADD(1)    $D7;
AY1:    LDL(3)     .ADRES;
        CADD(3)    $C1;
        LDA        0(3);
        LDL(2)     .ADRD;
        CADD(2)    $C0;
        STA        0(2);
        ALIT(0)    =1;
        TXEFM(1)   .AY1;
        LIT(1)     =1.0.0;
        CROTR(1)   24;
        CADD(1)    $C7;
        CSUB(1)    $D1;
        CROTL(1)   24;
AY2:    LDL(3)     .ADRES;
        CADD(3)    $C1;
        LDA        0(3);

        LDL(2)     .ADRD;
        CADD(2)    $C0;
        STA        0(2);
        ALIT(0)    =1;
        TXEFM(1)   .AY2;
        LIT(1)     =1.0.0;%    BRING THE ELEMENTS FROM TRASE INTO
        CROTR(1)   24;%        THEIR RESPECTIVE MATRIX.
        CADD(1)    $D3;
        CROTL(1)   24;
AY3:    LDL(3)     .ADRES;
        CADD(3)    $C1;
        LDL(2)     .ADRD;
        CADD(2)    $C1;
        LDA        0(2);
        STA        0(3);
        TXEFM(1)   .AY3;
        LDL(3)     .SAV1;
        EXCHL(3)   $ICR;%      GO BACK INTO MAIN-PROGRAM
%  ***********************************************************************
SHUFL:  FILL;
```

```
*                                  SHUFL BRINGS 2ND ROW TO THE BOTTOM
*                                  AND 2ND COLUMN TO THE RIGHT HAND END.
         STL(3)     .SAV1;
         LDL(3)     .ADRES;%       PICK UP THE SECOND ROW OF THE MATRIX
         CADD(3)    $01;%          IN USE AND STORE IT TEMPORARILY IN TEMP
         LDL(0)     .SPEC;
         LDEE1      $C0;
         LDA        0(3);
         STA        TEMP;
         LDL(2)     $C3;
         LDL(0)     .N;
         CADD(0)    $08;
SH;      CADD(2)    $01;
         EQLXT(2)   $C0,SHH;
         LDA        0(2);
         STA        0(3);
         CADD(3)    $01;
         SKIP       .SH;
SHH;     LDA        TEMP;
         STA        0(3);%         END 2ND ROW SHUFFLE
         LIT(0)     =0,1,0;%       SKEW MATRIX IN USE FOR COLUMN EXCESS
         CADD(0)    $03;
         CROTL(0)   24;
SH1;     LDL(2)     .ADRES;
         CADD(2)    $C0;
         LDA        0(2);
         STL(0)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         STA        0(2);
         TXEEM(0)   .SH1;%         SKEW-END
         LDA        NUMB;%         ADJUST INDEX TO PICK UP 2ND COLUMN.
         LDL(0)     .ONE;
         STL(0)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         LDS        $A;
         LDX        $S;%           END ADJUST
         LDL(1)     .ADRES;%       PICK UP 2ND COLUMN AND STORE IT IN TEMP.
         LDA        *0(1);
         STA        TEMP;%         END PICK UP
         LIT(0)     =0,1,0;%       REARRANGE THE REST OF THE MATRIX
         CADD(0)    $03;
         CROTL(0)   24;
SH2;     LDL(2)     .SPEC;
         LIT(3)     =1400000000000000000000;8;
         CEXOR(2)   $C3;
         STL(2)     .SAV8;
         CLC(3);
         SLIT(3) =ROTAR;EXCHL(3) $ICR;
         LDL(2)     .SAV8;
         LDEE1      $C2;
         LDL(1)     .ADRES;
         CADD(1)    $C0;
         STL(1)     .SAV4;
         LDA        0(1);
         LDL(3)     .NNO;
         STL(3)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         STL(0)     .SAV5;
```

```
        LDL(0)    .ONE;
        STL(2)    .SAV8;
        CLC(3);
        SLIT(3) =ROTAL;EXCHL(3) $ICR;
        LDL(2)    .SAV8;
        LDEE1     $C2;
        LDL(0)    .SAV5;
        LDL(1)    .SAV4;
        STA       0(1);
        TXEFM(0)  ,SH2;%     END REST-ARRANGE
        LDL(0)    .SPEC;%    BRING 2ND COL. INTO LAST COL.
        LDEE1     $C0;
        LDA       $X;%       FIRST; ADJUST INDEX
        LDL(1)    .NMO;
        CSUB(1)   $D1;
        STL(1)    .SAV2;
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        LDS       $A;
        LDX       $S;%       END INDEX-ADJUST
        LDA       TEMP;
        LDL(2)    .NMO;
        CSUB(2)   $D1;
        STL(2)    .SAV2;
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        LDL(1)    .ADRES;
        STA       *0(1);%    END COL.-SWITCH
        LDL(1)    .ADRES;%   UNSKEW MATRIX IN USE
        LDL(2)    .N;
SH3;    CADD(1)   $D1;
        CSUB(2)   $D1;
        STL(2)    .SAV2;
        LDA       0(1);
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        STA       0(1);
        EQLXF(2)  $D1,SH3;%  END UNSKEW
        LDL(3)    .SAV1;%    RETURN TO MAIN-PROGRAM
        EXCHL(3)  $ICR;
% ******************************************************************
ANGLE;  FILL;
%                            ANGLE FINDS THE TRANSFORMATION-MATRIX
%                            CONSISTING OF SINES & COSINES AS 2|2
%                            DIAGONAL MATRICES.
        STL(3)    .SAV1;
        LIT(0)    =0.2,0;%   CREATE THE TURN-ON PATTERN FOR THE
%                            SUPER-DIAGONALS
        CADD(0)   $D3;
        CROTL(0)  24;
        CADD(0)   $D1;
        LDL(1)    .SPEC;
AN;     CCB(1)    0(0);
        TXEFM(0)  ,AN;%      END TURN-ON PATTERN
        STL(1)    .ANTUN;
        LDEE1     $C1;
        LDA       NUMB;%     FIND CORRECT INDEX
        LDS       =1;
        ADM       $S;
        STA       TEMP;
        LDA       NUMB;
```

```
          RTL       $A,1;
          CSHR(1)   1;
          LDEE1     $C1;
          LDA       $R;
          STA       TEMP;%      TEMP CONTAINS THE INDEX OF THE FORM
%                               2L+1,2L FOR L=0,1,2,...,N-1/2
          LDL(0)    .ANTUN;
          LDEE1     $C0;
          LDX       TEMP;
          LDL(3)    .ADRA;
          LDA       *0(3);%   LOAD A[2I,2I-1] FOR ALL I LSS THAN OREQLN/2
          ADRN      $A;%        2!A[2I,2I-1].
          STA       TEMP1;
          RTL       $A,1;
          CSHR(0)   1;
          LDEE1     $C0;
          LDA       $R;
          STA       TEMP1;%     ALL PE S CONTAIN THE VALUE 2!A[2I,2I-1]
%                               IN GROUPS OF 2.
          LDL(2)    .SPEC;
          LDEE1     $C2;
          LDX       NUMB;
          LDL(0)    .ADRA;
          LDS       *0(0);%     LOAD A[I,I] FOR ALL I
          LDL(3)    .ANTUN;
          LDEE1     $C3;
          LDA       $S;%        $A CONTAINS A[2I-1,2I-1] IN EVERY OTHER PE
          RTL       $S,-1;
          LDS       $R;
          SBRN      $S;%        FORM A[2I-1,2I-1]-A[2I,2I]
          STA       TEMP2;
          RTL       $A,1;
          CSHR(3)   1;
          LDEE1     $C3;
          LDA       $R;
          STA       TEMP2;%     ALL PE S OF TEMP2 CONTAIN A[2I-1,2I-1]-
%                               A[2I,2I] IN GROUPS OF 2.
%                               NOW WE HAVE TO DIFFERENTIATE BETWEEN CASES
%      ******************************
%CASE 1: TEMP2=0
          LDEE1     $C2;
          LDA       TEMP2;
          LIT(0)    =0.0;
          IME       $C0;%     SET I WHERE MANTISSA IS ZERO,
% AND MASK OUT THE PART BEYOND THE BOUNDARY N
          SETE      I.AND.E;
          SETC(0)   E;
          STL(0)    .ANTUN1;%
          ZERF(0)   .+1;%     IF  NONE OF THE TEMP2 WERE ZERO, THEN GO
          JUMP      AN1;%     TO AN1; OTHERWISE CONTINUE.
          LDEE1     $C0;
          LIT(3)    =0.5;
          LDA       $C3;
          CALL SQRT64();
          LDX       NUMB;
          LDL(1)    .ADRC;
          STA       *0(1);%     COSINES =  SQRT(0.5) WHERE TEMP2=0.
          LDA       TEMP1;
%      ******************************
%CASE1A: FIND OUT WHERE TEMP1 LSS 0 WHERE TEMP2=0.
          LDL(1)    .ZERO;
```

```
          IAL          $C1;
          SETC(1)      I;
          CAND(1)      $C0;
          ZERT(1)      .AN2;%      IF TEMP1 IS NOWHERE LSS 0 WHERE TEMP2=0
%                                  THEN GO TO AN2.OTHERWISE
          LDL(2)       .ANTUN;%    PLACE THE SINES WITH THEIR CORRECT SIGNS
          CSHR(2)      1;
          CAND(2)      $C1;
          LDEE1        $C2;
          LDS          TEMP;%      TEMP CONTAINS INDEX FOR SUPER-DIAGONALS
          LDL(3)       .ADRC;
          LDA          *0(3);
          CHSA;
          STA          *0(3);
          CSHL(2)      1;
          LDEE1        $C2;
          LDS          TEMP;
          LDA          *0(3);
          STA          *0(3);
          COMPC(1);
          CAND(0)      $C1;%
%         ************************
%CASE1B; TEMP2=0 AND TEMP1 GTR OR EQL 0.
          ZERT(0)      .AN1;
AN2;      LDEE1        $C0;
          LDL(3)       .ADRC;
          LDA          *0(3);
          LDS          TEMP;
          LDL(1)       .ANTUN;
          CAND(1)      $C0;
          LDEE1        $C1;
          CHSA;
          STA          *0(3);
          CSHR(1)      1;
          LDEE1        $C1;
          STA          *0(3);
          LDL(0)       .ANTUN1;
%         ************************
%CASE2; FIND COSINE & SINE FOR TEMP2_0
AN1;;     LDL(1)       .SPEC;
          CEXOR(0)     $C1;
          LDEE1        $C0;
          LDA          TEMP1;
          DVRN         TEMP2;
          STA          TEMP1;
          MLRN         $A;
          LIT(3)       =1.;
          ADRN         $C3;
          CALL SQRT64();
          LDS          $A;
          LIT(3)       =0.5;
          LDA          $C3;
          DVRN         *S;
          STA          TEMP2;
          LIT(3)       =0.5;
          LDA          $C3;
          ADRN         TEMP2;
          CALL SQRT64();
          LDX          NUMB;
          LDL(3)       .ADRC;
          STA          *0(3);%     COSINES STORED FOR TEMP2_0
```

```
          LIT(3)    =0.5;
          LDA       $C3;
          SBRN      TEMP2;
          CALL SQRT64();
%       ************************
%CASE2A; TEMP1 LSS 0
          STA       TEMP2;
          LDA       TEMP1;
          ISN;
          SETC(1)   I;
          CAND(1)   $C0;
          ZERT(1)   .AN3;%      NONE OF THE TEMP1 WERE LSS 0, SO GO TO AN3
          LDEE1     $C1;
          LDA       TEMP2;
          LDS       TEMP;
          LDL(2)    .ANTUN;
          CAND(2)   $C1;
          LDEE1     $C2;
          LDL(3)    .ADRC;
          CHSA;
          STA       #0(3);
          CSHR(2)   1;
          LDEE1     $C2;
          STA       #0(3);
          COMPC(1);
          CAND(0)   $C1;ZERT(0) .AN4;
%       ************************
%CASE2B; TEMP1 GEQ 0
AN3;      LDEE1     $C0;
          LDS       TEMP;
          LDA       TEMP2;
          LDL(1)    .ANTUN;
          CAND(1)   $C0;
          LDEE1     $C1;
          LDL(3)    .ADRC;
          STA       #0(3);
          CSHR(1)   1;
          LDEE1     $C1;
          CHSA;
          STA       #0(3);
AN4;      LDL(3)    .SAV1;
          EXCHL(3)  $ICR;
% ***********************************************************************
MULTPL;   FILL;
%                               MULTIPLICATION OF TWO MATRICES. THE
%                               SET-UP IS SUCH THAT THE ADDRESS OF
%                             THE MATRIX IS TREATED AS A  VARIABLE
          STL(3)    .SAV1;%     SAVE RETURN ADDRESS
          LIT(1)    =0,1,0;
          CADD(1)   $D3;
          CROTL(1)  24;
          LDL(3)    .SPEC;
          LDEE1     $C3;
MUL1;     LDX       NUMB;%      PE-NUMBERS
          LDL(3)    .ADRES;%    =AMATRIX-BASE
          CADD(3)   $C1;
          LDA       0(3);%      LOAD ROW OF AMATRIX
          LIT(0)    =0,1,0;
          CADD(0)   $D3;
          CROTL(0)  24;
          LDS       =0;
```

```
MUL1::  LDL(3)     .ADRES1:%  BMATRIX-BASE
        MLRN       *0(3):%    MULTIPLY BMATRIX
        ADRN       $S:3       FORM PARTIAL SUM
        STA        MANT;
        SETE       E.OR.-E;
        SETE1      E.AND.E;
        CLRA;
        LDS        $A;
        LDL(3)     .SPEC;
        LDEE1      $C3;
        LDA        $R;
        EQLXF(0)   $01,MUL3;%  IF $C0=1, CHECK IF .SAV9=1
        LDL(3)     .SAV9;
        EQLXF(3)   $01,MUL3;
        CADD(0)    $03;
        CSUB(0)    $01;
        LDL(2)     .NMO;
        CSUB(2)    $01;
        JUMP       MUL4;
MUL3:   LDL(2)     .ONE;
MUL4::  STL(2)     .SAV2;
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;SKIP ,0;
        STA        TEMP;
        SETE       E.OR.-E;
        SETE1      E.AND.E;
        CLRA;
        LDS        $A;
        LDL(3)     .SPEC;
        LDEE1      $C3;
        LDA        $X;
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        LDS        $A;
        LDX        $S;
        SETE       E.OR.-E;
        SETE1      E.AND.E;
        CLRA;
        LDS        $A;
        LDL(3)     .SPEC;
        LDEE1      $C3;
        LDS        MANT;
        LDA        TEMP;
        TXLFM(0)   ,+1;
        JUMP       MUL1;
        LDL(2)     .ADRO;
        CADD(2)    $C1;
        STS        0(2);
        TXLFM(1)   ,+1;
        JUMP       MUL;
        LIT(0)     =0,1,0;
        CADD(0)    $03;
        CROTL(0)   24;
MUL2:   LDL(3)     .ADRES2;
        CADD(3)    $C0;
        LDL(2)     .ADRO;
        CADD(2)    $C0;
        LDA        0(2);
        STA        0(3);
        TXFFM(0)   .MUL2;
        LDL(3)     .SAV1;
```

```
         EXCHL(3)  $ICR;
%   ***************************%*********************************************************
SAMUL:  FILL;
%     THIS PROCEDURE MULTIPLIES ANMAT(TR.), THE TRANSPOSE
% OF THE TRANSFORMATION-MATRIX BY BASE=BASE X ANMAT COM-
% PUTED IN THE PROCEDURE MULTPL. SAMUL TAKES ADVANTAGE
% OF THE SYMMETRY OF THE RESULTING MATRIX AND, THUS, COM-
% PUTES ONLY THE ELEMENTS OF THE FIRS N/2+1 DIAGONALS
% AND FILLS THE REST WITH THE JUST CALCULATED ELEMENTS.
% THIS MEANS FULL EFFICIENCY , SNCE THE NUMBER OF MULTIPL.
% HAS , THUS, BEEN REDUCED FROM N X N TO 3(N/2+1). FURTHER
% EXPLANATIONS FOLLOW AS THE ALGORITHM GOES ALONG.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          SETE        E.OR.-E;
          SETE1       E.AND.E;
          CLRA;
          STA         TEMP;
          STL(3)      .SAV1;%  SAVE THE RETURN ADDRESS
          LIT(0)      =0,1,0;%  SET UP THE LOOP COUNT
          LDL(1)      .N;
          CSHR(1)     1;%        $C1 = N/2
          CADD(0)     $C1;
          CROTL(0)      24;
SAL9::     LDL(2)     .SPEC;%    $C2 = TURN ON PATTERN
          LDEE1       $C2;
          LDX         NUMB;%     $X = INDEX TO PICK UP THE
%  DIAGONAL EEMENTS OF ANMAT(TR.)
          LDL(1)      .ADRES1;%  ADRES1=ADDRESS OF ANMAT(TR,)
          LDA         *0(1);%    THE DIAGONAL ELEMENTS ARE
%  LOADED
          EQLXT(0)     $D0,SAL;
          STL(0)      .SAV2;
          CLC(3);%               MATCH THESE ELEMENTS WITH
% THE LOCATION OF THE ELEMENTS OF BASE, THEY NEED TO BE
% MULTIPLIED BY.
          SLIT(3) =ROUTE;EXCHL(3) $ICR;
SAL:      STA         TEMP1;%    SAVE THE ELEMENTS OF ANMAT(TR.)
% TEMPORARILY IN TEMP1;
       LDA         $X;
       EQLXT(0)   $D0.SAL1;
       STL(0)      .SAV2;
       CLC(3);
       SLIT(3) =ROUTE;EXCHL(3) $ICR;
% ADJUST THE INDEX FOR THE PROPER PICK UP OF THE ELEMENTS
% OF THE MATRIX BASE
SAL1:  LDS          $A;
       LDL(1)     .ADRES;%  ,ADRES=ADDRESS OF BASE
        LDA        *0(1);
       MLRN        TEMP1;
       ADRN        TEMP;
% FORM THE PARTIAL PRODUCT.
       STA         TEMP;
%     TAKE THE SUPERDIAGONAL OF ANMAT ABOVE THE DIA-
% GONAL AD MULTIPLY IT BY THE SUPERDIAGONAL OF BASE BELOW
% TE DIAGONAL
       LDA          NUMA;
       LDL(1)      .ONE;
       STL(1)       .SAV2;
       CLC(3);
       SLIT(3) =ROUTE;EXCHL(3) $ICR;
       LDS         $A;            %ADJUST THE INDEX
```

```
        LDX        $S;
        LDL(1)      .ADRES1;
        LDA        *0(1);
         LDL(1)    .NMD;
        STL(1)     .SAV2;%    BRING THESE ELEMENTS INTO
% THE LOCTION OF THE ELEMENTS OF BASE.
         CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        EQLXT(0)    $DO,SAL2;
        STL(0)     .SAV2;
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        SKIP       ,0;
SAL2:   STA        TEMP1;
        EQLXT(0)   $DO,SAL3;
        LDL(1)     $CO;
       CSUB(1)    $D1;
          JUMP        SAL4;% FIND THE ROUTING DISTANCE FOR THE IN-
% DEX FR DIAGONAL PICK-UP OF THE ELEMENTS OF BASE.
SAL3: LDL(1)     .NMD;
SAL4::    STL(1)     .SAV2;
       LDA        NUMB;
        CLC(3);
       SLIT(3) =ROUTE;EXCHL(3) $ICR;
       LDS        $A;
       LDL(1)    .ADRES;
       LDA        #0(1);
        MLRN      TEMP1;
       ADRN       TEMP;
        STA       TEMP;
       LDA        NUMB;
       LDL(1)      .NMD;
       STL(1)     .SAV2;
       CLC(3);
       SLIT(3) =ROUTE;EXCHL(3) $ICR;
% PICK UP THE ELEMENTS BELOW THE MAIN-DIAGONAL OF ANMAT(TR.)
% AD MULTIPLY THEM BY THE ELEMENTS ABOVE THE DIAGONAL
% OF BASE
       LDS        $A;
       LDX        $S;
        LDL(1)    .ADRES1;
       LDA        *0(1);
       LDL(1)     $CO;
       CADD(1)    $D1;
       STL(1)     .SAV2;
       CLC(3);%            MATCH UP THE ELEMENTS WITH THE
%  EEMENTS OF BASE.
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
       STA        TEMP1;
        LDA       NUMB;
       CLC(3);%            ADJUST THE INDEX FOR BASE PICK-UP
       SLIT(3) =ROUTE;EXCHL(3) $ICR;
       LDS        $A;
          LDL(1)     .ADRES;
       LDA        #0(1);
          MLRN       TEMP1;
          ADRN       TEMP;
% NOW WE HAVE TO STORE THE ELEMENTS PROPERLY.
          STA        TEMP;
          LDA        NUMB;
          EQLXT(0)   $DO,SAL5;
```

```
         STL(0)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
SAL5:    LDS        $A;
         LDL(1)     .ADRES2;% ADRES2=ADRESS OF TBASE A TEMP. MATRIX
         LDA        TEMP;
         STA        #0(1);
         EQLXF(0)   $00,+1;
         JUMP       SAL6;
         LDA        NUMB;
         LDL(3)     .N;
         CSUB(3)    $C0;
         STL(3)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         STA        TEMP1;
         LDA        TEMP;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         TXET(0)    $C0,SAL7;
         LDS        TEMP1;
         STA        #0(1);
         JUMP       SAL8;;
SAL7:    CSHR(2)    0(0);
         CAND(2)    $D5;
         CSHL(2)    0(0);
         LDEE1      $C2;
         LDS        TEMP1;
         STA        #0(1);
SAL8::   SETE       E.OR.-E;
         SETE1      E.AND.E;
         CLRA;
         STA        TEMP;
         TXETM(0)   ,+1;
         JUMP       SAL9;
% REPLACE THE 2K-1,2K AND THE 2K,2K-1 POSITIONS OF TEMP-
% ORARY SORAGE MATRIX BY FOAT ZEROES.
         LIT(0)     =2,0,0;
         CROTR(0)   24;
         CADD(0)    $D2;
         CROTL(0)   24;
         CLC(2);
SAL10:   LDL(1)     .END;
         CSHR(1)    0(0);
         CEXOR(2)   $C1;
         TXEFM(0)   .SAL10;
         LDEE1      $C2;
         LDA        NUMB;
         LIT(0)     =1;
         ADM        $C0;
         LDS        $A;
         COMPC(2);
         CAND(2)    $D5;
         LDEE1      $C2;
         LDA        NUMB;
         SBM        $C0;
         LDS        $A;
% REGISTER $S CONTAINS THE PATTERN 1,0,3,2,5,4,.....
         LDL(2)     .SPEC;
         LDEE1      $C2;
         LIT(0)     =0,0;
```

```
         LDA        $C0;
         LDL(1)     .ADRES2;
         STA        0(1);
% THE MATRIX HAS NOW ITS FINAL FORM. WE NOW MAP THE
% MATRIX STORED IN TBASE INTO BASE.
         LIT(0)     =1,0,0;
         CROTR(0)   24;
         CADD(0)    $D3;
         CROTL(0)   24;
SAL11:   LDL(1)     .ADRES;
         LDL(3)     .ADRES2;
         CADD(3)    $C0;
         LDA        0(3);
         CADD(1)    $C0;
         STA        0(1);
         TXEFM(0)   .SAL11;
         LDL(3)     .SAV1;
         EXCHL(3)   $ICR;
%###############################################################
ADDIT:   FILL;
%                                 ADDIT CALCULATES THE SUM OF THE OFF-
%                                 DIAGONALS SQUARE AND DIVIDES IT BY THE
%                                 SUM OF THE DIAGONALS SQUARE.
         STL(3)     .SAV1;
         LIT(1)     =0;%          CHECK FOR OFF-DIAGONAL OR DIAGONAL SUM
         STL(1)     .SAV5;
         LIT(0)     =0,1,0;%      ADDING THE ROWS INCLUDING A[I,I].
         CADD(0)    $D3;
         CROTL(0)   24;
         LDL(3)     .SPEC;
         SETE       E.OR.-E;
         SETE1      E.AND.E;
         CLRA;
         LDS        $A;
         LDEE1      $C3;
ADI:     LDL(2)     .ADRA;
         CADD(2)    $C0;
         LDA        0(2);
         MLRN       $A;
         ADRN       $S;
         LDS        $A;
         TXEFM(0)   .ADI;%        END ROW-SUM
         SETE       E.OR.-E;
         SETE1      E.AND.E;
         LIT(0)     =1;
ADI1:    LDS        $A;%          SUM UP THE ELEMENTS IN EACH PE.
         RTL        $S,0(0);%     USED FOR BOTH OFF-DIAGONALS& DIAGONALS
         LDS        $R;
         ADRN       $S;
         CADD(0)    $C0;
         LIT(1)     =64;
         EQLXF(0)   $C1,ADI1;%    END LOG-SUM
         LDL(1)     .SAV5;
         EQLXT(1)   $D1,ADI2;
         STA        TEMP;
         CLRA;
         LDS        $A;
         LDL(1)     .SPEC;%       PICK UP THE A[I,I] S.
         LDEE1      $C1;
         LDX        NUMB;
         LDL(2)     .ADRA;
```

```
        LDA       *0(2);
        MLRN      $A;
        SETE      E.OR.-E;
        SETE1     E.AND.E;
        LIT(0)    =1;
        LDL(1)    .SAV5;
        ALIT(1)   =1;
        STL(1)    .SAV5;
        JUMP      ADI1;
ADI2;   LDS       $A;
        LDA       TEMP;
        SBRN      $S;%         SUBTRACT THE A[I,I].
        LIT(0)    =0,0;
        IMF       $C0;
        SETC(0)   I;
        ONEST(0)  ,ADI3;
        DVRN      $S;%         CONVERGENCE FACTOR KSI FOUND.
        STA       TEMP;
        SLIT(0)   =TEMP;
        LOAD(0)   $C0;
        STL(0)    .S;
ADI3;   LDL(3)    .SAV1;
        EXCHL(3)  $ICR;
%  M*****************************************************************
TRASPUS; FILL;
%                              TRASPOS FINDS THE TRANSPOSE OF THE ANGLE-
%                              MATRIX BY JUST CHANGING THE SIGNS OF THE
%                              SUPER-DIAGONAL ELEMENTS.
        STL(3)    .SAV1;
        LDL(0)    .ANTUN;%     FIND THE INDEX FOR ABOVE ELEMENTS
        LDEE1     $C0;
        LDA       NUMB;
        LDS       =1;
        ADM       $S;
        STA       TEMP;LDA NUMB;
        CSHR(0)   1;
        RTL       $A,1;
        LDEE1     $C0;
        LDA       $R;
        STA       TEMP;%       INDEX IS FOUND
        LDL(0)    .SPEC;
        LDEE1     $C0;
        LDS       TEMP;
        LDL(3)    .ADRC;
        LDA       *0(3);
        CHSA;
        STA       *0(3);
        LDL(3) .SAV1;EXCHL(3) $ICR;
%  ********************************************************************
CONV;   FILL;
%                              CONV CHECKS FOR THE ABS.-VALUE OF THE OFF-
%                              DIAGONALS ELEMENTS LSS .BD(INDEX).
        STL(3)    .SAV1;
        LIT(0)    =0,2,0;
        CADD(0)   $D3;
        CSUP(0)   $D1;
        CROTL(0)  24;
        CLC(1);
CON;    LDL(2)    .FNB;
        CSHR(2)   0(0);
        CEXOR(1)  $C2;
```

```
        TXEFM(0)   .CON;
        LDEE1      $C1;
        LDA        NUMB;
        RTL        $A,1;
        CSHR(1)    1;
        LDEE1      $C1;
        LDS        $R;
        CSHL(1)    1;
        LDEE1      $C1;
        LDL(2)     .ONE;
        ADM        $C2;
        LDS        $A;
        LDL(1)     .SPEC;
        LDEE1      $C1;
        LDL(2)     .ADRA;
        LDA        *0(2);
        LDL(0)     .ZERO;
        IAL        $C0;
        SETE       I.AND.E;
        SETE1      E.AND.E;
        CHSA;
        COMPC(1);
        LDEE1      $C1;
        CLRA;
        COMPC(1);
        LDEE1      $C1;
        LDL(0)     .SAV6;
        LDL(2)     .AD(0);
        IAL        $C2;
        SETC(2)    I;
        STL(2)     .CONVE;
        LDL(3)     .SAV1;
        EXCHL(3)   $ICR;
%  *********************************************************************
TRPS:   FILL;
%                         THIS PROCEDURE TRANSPOSES THE
%  ORIGINAL MATRIX.
        STL(3)      .SAV1;%  SAVE RETURN ADRESS
%#########################
        LIT(0)     =0,1,0;%  SET UP LOOP-INDEX I
        CADD(0)    $03;
        CROTL(0)   24;
        LDL(3)     .SPEC;
        LDEE1      $C3;
TS3::   LDA        NUMB;
        EQLXT(0)   $00,TS;
        STL(0)     .SAV2;% ROUTE D = I
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
TS:     LDS        $A;
        LDX        $S;
        LDL(3)     .ADRB;%  ADRESS OF EIGV
        LDA        *0(3);
        EQLXT(0)   $00,TS1;
        LDL(3)     .N;
        CSUB(3)    $C0;
        STL(3)     .SAV2;%   ROUTE D= N-1
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
TS1:    STA        TEMP;
        LDA        NUMB;
```

```
         EQLXT(0)  $00,TS2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
TS2:     LDS       $A;
         LDA       TEMP;
         LDL(3)    .ADRC;%    ADRESS OF ANMAT
         STA       #0(3);
         TXETM(0)  .+1;
         JUMP      TS3;
         LDL(3)    .SAV1;%    RETURN TO THE OUTSIDE
         EXCHL(3)  $ICR;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GERSH:   FILL;%           GERSH FINDS A BOUND ON THE EIGEN-
%                         VALUES USING THE GERSHGORING DISK
         STL(3)    .SAV1;
         SETE      E.OR.-E;
         SETE1     E.AND.E;
         CLRA;
         LDS       $A;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  THE RADIUS OF THE DISK ACC. TO GERSHGORIS THEORM CON-
%  SISTS OF THE SUM OF THE ABS. VALUE OF THE OFF-DIAGONAL
%  ELEMENTS LOCATED IN ROW I FOR WHICH A[I,I]=EIGVAL. I
%  I.E. THE ROWSUM OF EACH INDIVIDUAL ROW HAS TO BE
%  FOUND. THE CENTER OF THE DISK IS THE EIGENVALUE ITSSELF
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         LIT(1)    =0,1,0;
         CADD(1)   $03;
         CROTL(1)  24;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  FINDING THE SUM OF THE RO  EQUALS FINDING THE SUM OF THE
%  COLUMNS, SINCE THE MATRIX IS SYMMETRIC.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GERS:    LDL(0)    .SPEC;
         CCB(0)    0(1);
         LDEE1     $C0;
         LDL(2)    .ADRA;
         CADD(2)   $C1;
         LDA       0(2);
         SAP;%              TAKES ABS. VALUE, WHERE NECESSARY
         ADRN      $S;
         LDS       $A;
         TXEFM(1)  .GERS;
         LDL(0)    .SPEC;
         LDEE1     $C0;
         STA       GERSHG;%    CONTAINS RADII. THE CENTER HAS
%                              HAS TO BE READ FROM OUTPUT OF
%                              THE FINAL MATRIX, WHICH COMES
%                              OUT OF THE PROCEDURE EIGEN
         LIT(0)    =1,GERSHG,GERSHG;
         CROTR(0)  24;
         CADD(0)   $03;
         CROTL(0)  24;
         LIT(3)    =1,TACK-1,MESS0;
         DISPLAY   $C3,32;
         DISPLAYR  $C0,-16;
         LDL(3)    .SAV1;
         EXCHL(3)  $ICR;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
EIGEN[ENTRY]:  FILL;%    SAVE THE CONTENT OF ACAR0,ACAR1.
                    %  AND ADB $D32 THRU $D39
```

```
        STL(3)    .RETUR;
        CLC(3);
        SLIT(3)   ADBSAV+8;
        STORE(3)  $C0;
        ALIT(3)   1;
        STORE(3)  $C1;
        CLC(3);
        SLIT(3)   ADBSAV;
        LIT(0)    1,7,0;
SA:     STORE(3)  $D32(0);
        ALIT(3)   1;
        TXEFM(0)  .SA;
        LIT(0)    1,4,0;
        CLC(3);
SA1:    LOAD(2)   $C3;
        CSHR(3)   6;
        STL(3)    $D32(0);
        ALIT(2)   1;
        TXEFM(0)  .SA1;
        LOAD(2)   $C3;
        STL(3)    .N;
        LIT(0)    =1;
        STL(0)    .ONE;
        LIT(0)    =0;
        STL(0)    .ZERO;
% ************************************
        LDL(0)    .N;
        CSUB(0)   $D1;
        STL(0)    .NMU;
        LIT(0)    =64;
        CSUB(0)   $D2;
        STL(0)    .POUT;
        LIT(0)    =1000000000000000000000:8;
        STL(0)    .ENB;
        LIT(0)    =0;%        INITIALIZATION OF CONVERGENCE AND
        STL(0)    .KSI;%      THRESHOLD FACTORS.
        LIT(0)    =8;
        STL(0)    .BOUND;
        LIT(0)    =1;
        STL(0)    .ZEZE;
        LIT(0)    =0.1;
        STL(0)    .BD;
        LIT(0)    =0.01;
        STL(0)    .BD1;
        LIT(0)    =0.0001;
        STL(0)    .BD2;
        LIT(0)    =0.00000001;
        STL(0)    .BD3;
        LIT(0)    =0;%        INDEX TO FETCH ONE OF .BD &F.F.
        STL(0)    .SAV6;
% $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
        SETE      E.OR.-E;%  CREATE THE TURN ON PATTERN
        SETE1     E.AND.E;
        LDA       NUMB;
        LDL(0)    .N;
        IAL       $C0;
        SETC(0)   I;
        CLRA;
        STL(0)    .SPEC;
% ****************************************************************
%   ***************************
```

```
        LDL(0)     .ADRE;
        EQLXT(0)   $D0,+1;%   CHECK WETHER AN IDENTITY MATRIX HAS
        JUMP       MAI1;&     TO BE CREATED OR NOT
        LDL(0)     .SPEC;%     CREATE IDENTITY MATRIX.
        LDEE1      $C0;
        LDX        NUMB;
        LIT(1)     =1.;
        LDA        $C1;
        LDL(2)     .ADRB;
        STA        *0(2);%     EIGV=EIGENVECTOR-MATRIX INITIALLY THE
%                              IDENITITY-MATRIX.
%    ****************************
        SKIP       .MAI1;
MAI2::  LDL(0)     .SAV6;
        ALIT(0)    =1;
        STL(0)     .SAV6;
%    ****************************
        LDL(0)     .ZEZE;
        EQLXT(0)   $D14,MAI1;
        CADD(0)    $^0;
        STL(0)     .ZEZE;
%    ****************************
MAI1::  CLC(3);%               FIND MAXIMAL ROW-SUM AND ROW-INDEX .MAX.
        SLIT(3) =RWSM;EXCHL(3) $ICR;
        LDL(0)     .MAX;
        SETE       E.OR.-E;
        SETE1      E.AND.E;
        CLRA;
        LDS        $A;
        EQLXF(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) $ICR;
%       LIT(3)     =1,MESS2-1,MESS1;
%       DISPLAY    $C3,32;
%       LDL(3)     .ADRA;CSHL(3) 6;WRTPEM;
        SETE       E.OR.-E;
        SETE1      E.AND.E;
        CLRA;
        LDS        $A;
        LDL(3)     .ADRB;%     NOW WE SHUFFLE EIGV.
        STL(3)     .ADRES;
        CLC(3);
        SLIT(3) =ANYR;EXCHL(3) $ICR;
%    ****************************
MAI3::  LIT(0)     =0,1,0;&   INNER LOOP, IN WHICH ALL TRANSFORMATIONS
%                              TAKE PLACE.
        CADD(0)    $D3;
        CSUB(0)    $D1;
        CROTL(0)   24;
        STL(0)     .INDEX;
%    ****************************
MAI3A:: CLC(3);%               FIND THE ABSOLUTE VALUE OF THE SUPER-DIAG
%                              TO COMPARE THEN AGAINST A THRESHOLD FACTO
        SLIT(3) =CONV;EXCHL(3) $ICR;
        SETE       E.OR.-E;
        SETE1      E.AND.E;
        CLRA;
```

```
        LDS        $A;
        LDL(1)     .CONVE;
        ONESF(1)   ,+1;%       DON T DO ANY TRANSFORMATION, BUT SHUFFLE
        JUMP       MAI4;
MAI5;   CLC(3);%              FIND THE ANGLE-MATRIX.
        SLIT(3)    =ANGLE;EXCHL(3) $ICR;
 %      LIT(3)     =1,MESS3-1,MESS2;
%       DISPLAY    $C3,32;
%       LDL(3)     .ADRC;CSHL(3) 6;WRTPEM;
%   ***************************
        LDL(3)    .ADRC;%       GO THROUGH THE TRANSFORMATION BY MEANS
        STL(3)     .ADRES1;%       OF MULTIPLICATION
        LDL(3)     .ADRA;
        STL(3)     .ADRES;
        STL(3)     .ADRES2;
        LIT(3)     =0;
        STL(3)     .SAV9;
        SETE       E.OR.-E;
        SETE1      E.AND.E;
        CLRA;
        LDS        $A;
        CLC(3);
        SLIT(3) =MULTPL;EXCHL(3) $ICR;
        LDL(3)     .ADRB;
        STL(3)     .ADRES;
        STL(3)     .ADRES2;
        LDL(3)     .ADRC;
        STL(3)     .ADRES1;
        SETE       E.OR.-E;
        SETE1      E.AND.E;
        CLRA;
        LDS        $A;
        CLC(3);
        SLIT(3)    =MULTPL;
        EXCHL(3)   $ICR;
        CLC(3);%              FIND TRANSPOSE OF ANGLE-MATRIX
        SLIT(3) =TRASPOS;EXCHL(3) $ICR;
        LDL(3)     .ADRA;
        STL(3)     .ADRES;
        LDL(3)     .ADRD;
        STL(3)     .ADRES2;
        LDL(3)     .ADRC;
        STL(3)     .ADRES1;
        SETE       E.OR.-E;
        SETE1      E.AND.E;
        CLRA;
        LDS        $A;
        CLC(3);
        SLIT(3) =SAMUL;EXCHL(3) $ICR;
%       LIT(3)     =1,MESS4-1,MESS3;
%       DISPLAY    $C3,-32;
%       LDL(3)     .ADRA;CSHL(3) 6;WRTPEM;
%   ***************************
        CLC(3);%              FIND  CONVERGENCE-FACTOR.
        SLIT(3) =ADDIT;EXCHL(3) $ICR;
        ONESF(0)   ,+1;
        JUMP       MAIEND;
        LDL(0)     .KSI;
        LDA        $C0;
        ILZ;
        SETC(0)    I;
```

```
          ONESF(0)   .MAI6;
          LIT(2)     =0.000000000000001;
          LDA        $C2;
          LDL(2)     .S;
          LDS        $C2;
          MLRN       $S;
          STA        TEMP;
          SLIT(1)    =TEMP;
          LOAD(1)    $C1;
          STL(1)     .KSI;
MAI6:     LDL(0)     .ZEZE;
          LESSF(0)   $D14,+1;
          JUMP       MAI4;
          LDL(0)     .S;
          LDL(1)     .KSI;
          LDA        $C0;
          IAG        $C1;
          SETC(0)    I;
          ONEST(0)   .+1;
          JUMP       MAIEND;
%    ****************************
MAI4::    LDL(0)     .INDEX;
          TXFEM(0)   .+1;
          JUMP       MAI2;
          STL(0)     .INDEX;
          LDL(3)      .ADRA;%    DO A 2ND-ROW,2ND-COLUMN SHUFFLE ON BASE
%                                AND EIGENVECTOR-MATRIX;
%                                BASE-SHUFLE
          STL(3)     .ADRES;
          SETE       E.OR.-E;
          SETE1      E.AND.E;
          CLRA;
          LDS        $A;
          CLC(3);
          SLIT(3) =SHUFL;EXCHL(3) $ICR;
%         LIT(3)      =1,ADRSAV-1,MESS4;
%         DISPLAY    $C3,32;
%         LDL(3)      .ADRA;CSHL(3) 6;WRTPEM;
          LDL(3)     .ADRB;%      EIGV-SHUFLE
           STL(3)    .ADRES;%
          SETE       E.OR.-E;%
           SETE1     E.AND.E;%
          CLRA;
          LDS        $A;
          CLC(3);SLIT(3) =SHUFL;EXCHL(3) $ICR;
          JUMP       MAI3A;
MAIEND::   LDL(0)    .ADRE;
          EQLXF(0)   $D0,+1;
          JUMP       MAIENDE;
          LDL(3)     .ADRB;%    ADRESS OF EIGV
          STL(3)     .ADRES;
          LDL(3)     .ADRE;
          STL(3)     .ADRES1;
          STL(3)     .ADRES2;
          LIT(3)     =0;
          STL(3)     .SAV9;
          CLC(3);
          SLIT(3) =MULTPL;EXCHL(3) $ICR;
          CLC(3);
          SLIT(3) =TRPS;EXCHL(3) $ICR;
```

```
        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 BOUND ON
%   THE EIGENVALUES
        CADD(0)    $D3;
        CRDTL(0)   24;
AT:     LDL(3)     .ADRA;
        CADD(3)    $C0;
        LDL(2)     .ADRE;
        CADD(2)    $C0;
        LDA        0(3);
        ADRN       0(2);
        STA        0(3);
        TXEEM(0)   .AT;
MATENDE:: CLC(3);
        SLIT(3)    =GERSH;
        EXCHL(3)   $ICR;
        CLC(3);
        SLIT(3)    ADRSAV;
        BIN(3)     $D32;
        CLC(3);
        SLIT(3)    ADRSAV+8;
        LOAD(3)    $C0;
        ALIT(3)    1;
        LOAD(3)    $C1;
       LDL(3)      .RETUR;%   TURN BACK TO THE OUTSIDE
       EXCHL(3)    $ICR;
%###################################################
%#                                                 #
%#                                                 #
%#               END JACOBI/EIGEN                  #
%#                                                 #
%#                                                 #
%###################################################
END       EIGEN.
```

APPENDIX C

The Subroutine EBERL, Jacobi-Like Method

```
          BEGIN
          FILL        128;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CALL - DEFINITION
   DEFINE CALL &NAME(&PARAMETERS)=
          &IF &SIGN(&METFLD(&NAME)) &THEN
          EXTERNAL   &NAME; &FI
          &IF &EMPTY(&PARAMETERS) &THEN &ELSE
          BEGIN BLOCK
            BEGIN USE (63)
              LIST:   DATA   &PARAMETERS
            END;
            CLC(2);
            SLIT(2) LIST;
          END; &FI
          CLC(3);
          SLIT(3)     &NAME;
          EXCHL(3)    $ICR;##;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEFINE   WRTPEM =
%         LIT(1)      =1,0,0;
%         CADD(1)     $C3;
%         CROTR(1)    24;
%         CADD(1)     $C3;
%         CADD(1)     $D3;
%         CROTL(1)    24;
%         LIT(0)      =0,1,0;
%         CADD(0)     $D3;
%         CROTL(0)    24;
%         DISPLAYR    $C1,-16;
%         LIT(2)      =64;
%         CADD(1)     $C2;
%         CROTR(1)    24;
%         CADD(1)     $C2;
%         CROTL(1)    24;
%         TXFEM(0)    ,-9;##;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                  EBERL NORMALIZES A MTRIX
%%               AND, THUS, PREPARES IT FOR EIGEN-
%%                 VALUE COMPUTATION SUCH AS JACOBI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
.ZERO:    EQU         $D0;%       FIXED POINT ZERO.
.ONE:     EQU         $D1;%       FIXED POINT ONE.
.N:       EQU         $D2;%       ORDER OF MATRIX
.NMO:     EQU         $D3;%       N-1;
.ENB:     EQU         $D4;%       1000000000000000000000:8,ENABLING ONE PE
.SPEC:    EQU         $D5;%       ENABLING PATTERN FOR THE FIRT N PE S.
.ROUT:    EQU         $D6;%       64-N, CONSTANT USED IN END AROUND ROUTING
.MAX:     EQU         $D7;%       ROW INDEX FOR MAX. VAL. FOUND IN RWSM.
.ADRES:   EQU         $D8;%       ADDRESS SAVED HERE
.ADRES1:  EQU         $D9;%
.ADRES2:  EQU         $D10;
.FINVAL1:EQU          $D11;%      CONVERGENCE CHECK
.ANTUN:   EQU         $D12;%      TURN-ON PATTERN FOR THE ANGLE ROUTINE
.ANTUN1:  EQU         $D13;
.G:       EQU         $D14;%      THE MAXIMAL OFF-DIAGONAL ELEMENT
.MAXR:    EQU         $D15;%      THE ROWINDEX FOR ABOVE ELEMENT
.MAXC:    EQU         $D16;%      THE COLUMNINDEX FOR ABOVE ELEM.
.AA:      EQU         $D17;%      OSCILLATION CONSTANTS
.AB:      EQU         $D18;
.AC:      EQU         $D19;
.INDEX:   EQU         $D20;%      INNER LOOP COUNT IN MAIN PROGRAM.
```

```
.FINVAL:  EQU       $D21;%      CONVERGENCE-FACTOR FOUND IN HYANG-ROUTINE.
,CONVE:   EQU       $D22;%      CHECK FOR SUPERDIAGS EQL 0.
.SAV1:    EQU       $D24;%      SAVE REGISTER.
.SAV2:    EQU       $D25;
.SAV3:    EQU       $D26;
.SAV4:    EQU       $D27;
.SAV8:    EQU       $D28;
.SAV5:    EQU       $D29;
,SAV9:    EQU       $D30;
.RETUR:   EQU       $D31;%      RETUR CONTAINS THE RETURN ADR.
%                               ADRESS TO LINK TO THE OUTSIDE
.ADRA:    EQU       $D32;%      ADRESS OF ORIGINAL MATRIX
.ADRCM:   EQU       $D33;%      ADRESS OF EIGENVECTORMATR.
,ADRC:    EQU       $D34;%      ANGLE MATRIX
.ADRD:    EQU       $D35;%      TEMP. STORAGE MATRIX
,ADRE:    EQU       $D36;%      ADRESS OF THE ERROR MATRIX
.ADRF:    EQU       $D37;%      ADRESS OF MATRIX CARRIED OVER
%                               TO THE JACOBI-SUBROUTINE FOR
%                               PROPER COMPUTATION OF THE EIGEN-
%                               VECTORS.
.TNMTW:   EQU       $D38;%   FACTOR FOR CHECK ON C[2I-1,2I-1]
TEMP:     BLK       1;%         ONE ROW OF SAVE-STORAGE IN PE-MEMORY
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;
          DATA      17,18,19,20,21,22,23,24,25,26,27,28,29,30;
          DATA      31,32,33,34,35,36,37,38,39,40,41,42,43,44;
          DATA      45,46,47,48,49,50,51,52,53,54,55,56,57,58;
          DATA      59,60,61,62,63;%     PE NUMBERING
%MESS1:   DATA      "########## THE C - MATRIX##########";
%MESS2:   DATA      "###### THE MATRIX BASE AFTER SHUFFLE####";
%MESS3:   DATA      "#### AFTER THE 2-ND TRANSFORMATION ####";
ADRSAV:   WDS       10;
% ********************************************************************
ROUTE:    FILL;%                ROUTE-ROUTINE FOR N<64
          STL(3)    .SAV3;%     SAVE RETURN ADDRESS.
          LDL(3)    .SAV2;%     LOAD ROUTING DISTANCE.
          RTL       $A,0(3);%   $A IS ASSUMED TO CONTAIN THE ELEMENTS
%                               TO BE ROUTED.
          LDL(3)    .SPEC;%
          LDEE1     $C3;
          LDA       $R;
%#####################################################
%         THIS PART IS THE HEART OF THE MATTER. IT ELIMINATES
          LDS       NUMB;%         THE ELEMENTS WHICH MIGHT
          LDL(3)    .SAV2;%        HAVE COME INTO THE FIRST
%                                  D PES BY ROUTING A DISTANCE
%                                  D, BY CLEARING RGA IN THESE PES
          ISL       $C3;%
          SETE      I.AND.E;%
          SETE1     E.AND.E;
          CLRA;
%########################################
          LDL(3)    .ROUT;
          RTL       0(3);%      ROUTE THE ELEMENTS WHICH MOVED PAST THE
%    BOUNDARY N, 64-N INTO THE FIRST D PE"S OF RGR AND THEN
%    LOAD RGA FROM RGR IN THOSE FIRST D PE"S.
```

```
        LDA       $R;
        LDI(3)    .SPFC;
        COMPC(3);
        LDFF1     $C3;
        CLRA;
        COMPC(3);
        LDFF1     $C3;
        LDI(3)    .SAV3;%    AND THE RESULT WILL BE IN $A PROPERLY
        EXCHL(3)  $ICR;%     ROUTED.THEN RETURN.
% ***********************************************************************
ROTAR:    FILL;
%  ADJUST ACAR1 FOR AN END-AROUND SHIFT,RIGHT FOR D LSS 64
        STI(3)    .SAV3;%  SAVE RETURN ADRESS
        LDI(1)    .SAV8;%  PATTERN THAT NEEDS ADJUSTMENT
        CROTR(1)  0(0);%   ACARO CONTAINS D
        LDI(2)    .SPFC;
        CSHR(2)   0(0);
        CAND(2)   $C1;
        CAND(2)   $D5;
        CEXOR(1)  $C2;
        LDI(3)    .N;
        CROTL(1)  0(3);
        CEXOR(1)  $C2;
        STI(1)    .SAV8;
        LDI(3)    .SAV3;
        EXCHL(3)  $ICR;
%%C###################################################
ROTAL:    FILL;
%  ADJSUT ACAR1 FOR AN END-AROUND SHIFT,LEFT FOR D LSS64
        STI(3)    .SAV3;
        LDI(1)    .SAV8;
        CROTL(1)  0(0);
        LDI(3)    .SPFC;
        CSHL(3)   0(0);
        CAND(3)   $C1;
        CAND(3)   $D5;
        CEXOR(1)  $C3;
        LDI(2)    .ROUT;
        CSHL(1)   0(2);
        CEXOR(1)  $C3;
        STI(1)    .SAV8;
        LDI(3)    .SAV3;
        EXCHL(3)  $ICR;
%###################################################
FNDMX:    FILL;
%                            FIND THE LARGEST OFF-DIAGONAL
%                            ELEMENT AND RETURN IT AS WELL
%                            AS THE ROW- AND COLUMN-INDEX.
       STL(3)     .SAV1;%    SAVE RETURN ADRESS
%%########################
% PART 1: FIND THE LARGEST ELEMENT WITHIN ONE ROW
       LIT(0)     =0,1,0;%   SET UP THE LOOP FOR THE ROW
%                            PICK-UP
       CADD(0)    $D3;
       CSUB(0)    $D1;
       CROTL(0)   24;
       LDL(1)     .SPFC;
       STL(1)     .SAV9;
FND1::  SETE      E.OR.-E;%  CLEAN OUT $S & $A
       SETE1      E.AND.E;
       CLRA;
```

```
        LDS         $A;
        LDL(1)      .SAV9;
         CCR(1)     0(0);%    COMPL. THE I = $C0 -TH BIT
        STL(1)      .SAV9;%     ONLY THE UPPER TRIANGULAR
%                               OFF-DIAG. ELEM. PARTICIPATE
        LDL(2)      .ADRCM;%    FETCH ADRESS OF CMAT
        CADD(2)     $C0;
        LDEE1       $C1;
        LDA         0(2);%      LOAD ROW OF CMAT
        SAP;%                   TAKE ABS. VAL. WHERE NECESS.
        SETE        E.OR.-E;
        SETE1       E.AND.E;
        LDS         $A;%        FIND LARGEST ELEMENT IN THE
        STA         TEMP2;%     ROW LOADED
        LIT(1)      =1;
END;    RTL         $S,0(1);
        LDS         $R;
        IAL         $S;
        SETE        I.AND.E;
        SETE1       E.AND.E;
        LDA         $S;
        SETE        E.OR.-E;
        SETE1       E.AND.E;
        LDS         $A;
        CADD(1)     $C1;
        LIT(3)      =64;
        EQLXE(1)    $C3,END;%   END FIND MAXIMAL ELEMENT
        LDL(1)      .ENB;
        CSHR(1)     0(0);
        LDEE1       $C1;
        STS         TEMP;%      STORE VALUE JUST FOUND
        LDL(2)      .SAV9;
        LDEE1       $C2;%       FIND OUT WHICH COLUMN THE
        LDA         TEMP2;%     ELEMENT CAME FROM
        IAL         $S;
        SETE        -I.AND.E;
        SETC(2)     E;
        LEADD(2);
        LIT(3)      =77;8;
        LDEE1       $C1;
        CAND(2)     $C3;
        LDA         $C2;
        STA         TEMP1;%     COLUMNINDEX STORED
         TXETM(0)   ,+1;
%                               TEMP CONTAINS THE MAXIMAL
         JUMP       END1;
% VALUES OF EACH ROW. TEMP1 CONTAINS THE COLUMNINDECES.
% NOW WE HAVE TO FIND THE MAXIMAL VALUE OF TEMP=MAXIMAL
% OFF-DIAGONAL ELEMENT.
        SETE        E.OR.-E;
        SETE1       E.AND.E;
        CLRA;
        LDS         $A;
        LDL(1)      .SPEC;
         LDL(2)     .NMD;
         CCR(1)     0(2);%    TURN OFF THE (N-1)TH PE
        LDEE1       $C1;
        LDA         TEMP;
        SETE        E.OR.-E;
        SETE1       E.AND.E;
        LDS         $A;
```

```
        LIT(2)      =1;
END2:   RTL         $S,0(2);
        LDS         $R;
        IAL         $S;
        SETE        I.AND.E;
        SETE1       E.AND.E;
        LDA         $S;
        SETE        E.OR.-E;
        SETE1       E.AND.E;
        LDS         $A;
        CADD(2)     $C2;
        LIT(3)      =64;
        EQLXE(2)    $C3,END2;%     THE LARGEST IS FOUND
% $S AND $A CONTAIN IT.
        LIT(0)      =0.;
        INE         $C0;%    CHECK FOR THE LARGEST ELEM. TO BE ZERO
        SETC(0)     I;
         ONESE(0)   ,+1;
         JUMP       END3;
        LDEE1       $C1;
        LDA         TEMP;%    FIND ROW- AND COLUMN-INDEX BY
%                             COMPARING TEMP AGAINST $S.
        IAL         $S;
        SETE        -I.AND.E;
        SETC(0)     E;
        LEADD(0);
        LIT(3)      =77:8;
        CAND(0)     $C3;
        STL(0)      .MAXR;%        ROWINDEX FOUND
        SLIT(1)     =TEMP;
        CADD(1)     $C0;
        LOAD(1)     $C3;
        STL(3)      .G;%           STORE THE LARGEST ELEMENT
        CLC(1);
        SLIT(1)     =TEMP1;
        CADD(1)     $C0;
        LOAD(1)     $C3;
        STL(3)      .MAXC;%        THE COLUMNINDE FOR ABOVE ELEM.
END3:   LDL(3)      .SAV1;
        EXCHL(3)    $ICR;
###########################################################
MLTRPS:  FILL;
%                            MULTIPLY TWO MATRICES, ONE OF
%                            WHICH IS THE TRANSPOSE OF THE
%                            OTHER, WITHOUT ACTUALLY TRANS-
%                            POSING THE MATRIX.
         STL(3)      .SAV1;%  SAVE RETURN-ADRESS
         LIT(0)      =0,1,0;% SET UP LOOP FOR ROW-FETCH OF BASE
         CADD(0)     $D3;
         CROTL(0)    24;
MT1::    SETE        E.OR.-E;
         SETE1       E.AND.E;
         CLRA;%                CLEAN UP $A & $S
         LDS         $A;
         STA         TEMP;
         LDL(2)      .SPEC;
         LDEE1       $C2;
         LIT(2)      =0,1,0;
         CADD(2)     $D3;%    INNER LOOP FOR PICK-UP OF TRANS-
         CROTL(2)    24;%     POSE ELEMENTS OF BASE
MT:      LDA         NUMB;%   MAKE PROPER INDEXING
```

```
        STI(2)     .SAV2;
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        LDS        $A;
        LDX        $S;
        LDI(1)     .ADRES;
        CADD(1)    $CO;%    PIXK UP ROW I=$CO OF BASE
        LDA        0(1);
        LDI(1)     .ADRES;
        MLRN       *0(1);%  MULTIPLY BY TRANSP. ELEM. OF BASE
        ADRN       TEMP;%  ADD PARTIAL PRODUCT
        CLC(3);
        LDI(1)     .ONE;
        STI(1)     .SAV2;
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        STA        TEMP;
        TXEEM(2)   ,MT;
        LDA        TEMP;
        LDI(3)     .ADRES1;
        CADD(3)    $CO;
        STA        0(3);
        TXETM(0)   ,+1;
        JUMP       MT1;
        LDI(3)     .SAV1;
        EXCHL(3)   $ICR;
%##########################################################
TRPS:   FILL;
%                                THIS PROCEDURE TRANSPOSES
%                             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;% SAVE RETURN ADRESS
%%C#########################
        LIT(0)     =0,1,0;
%                               SET UP LOOP-INDEX I
        CADD(0)    $D3;
        CRDTL(0)   24;
        LDI(3)     .SPEC;
        LDEE1      $C3;
TS3:    LDA        NUMB;
        EQLXT(0)   $DO,TS;
        STI(0)     .SAV2;% ROUTE D=I;
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
TS:     LDS        $A;
        LDX        $S;
        LDI(3)     .ADRA;% ADRESS OF BASE
        LDA        *0(3);
        EQLXT(0)   $DO,TS1;
        LDI(3)     .N;
        CSUB(3)    $CO;
        STI(3)     .SAV2;% ROUTE D=N-I
        CLC(3);
        SLIT(3)  =ROUTE;EXCHL(3) $ICR;
TS1:    STA        TEMP;
        LDA        NUMB;
        EQLXT(0)   $DO,TS2;
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        SKIP       ,0;
```

```
TS2:        LDS         $A;
            LDA         TEMP;
            LDI(3)      .ADRD;%  ADRESS OF TBASE
            STA         #0(3);
            TXFEM(0)    ,TS3;
            LDI(3)      .SAV1;%  RETURN
            EXCHL(3)    $ICR;
%####################################################################
SHUF:       FILL;
%                           THIS PROCEDURE FALLS INTO TWO PARTS
%                       PART ONE BRINGS ROW MAXR INTO THE PLACE
%                       OF THE FIRST ROW. THE SECOND PART
%                       BRINGS THE COLUMN MAXC=MAXR INTO THE
%                       PLACE OF THE SECOND COLUMN. THIS
%                       PROCEDURE IS ENTERED ONLY IF MAXR+MAXC
%                       NEQL 1.
            STI(3)      .SAV1;%   SAVE RETURN ADRESS
%#########################################################
% PART ONE:
            LDI(3)      .SPEC;
            LDFF1       $C3;
            LDI(0)      .MAXR;%  IF MAXR=0, THEN THE LARGEST
%                            ELEMENT IS ALREADY IN ROW 0.
            EQLXF(0)    $D0,+1;%   WE JUST NEED TO ADJUST THE
            JUMP        PAR2;
            LDI(1)      .N;%       COLUMNS
            CSUP(1)     $C0;
            STI(1)      .SAV2;%    ROUTE DISTANCE D=N-MAXR
            LIT(0)      =0,1,0;
            CADD(0)     $D3;
            CROTL(0)    24;
PAR1:       LDI(1)      .ADRES;%   CONTAINS THE ARESS OF THE
% @                             MATRIX UNDER CONSIDERATION
            CADD(1)     $C0;%      ROUTE ALL ROWS OF MATRIX
            LDA         0(1);
            CLC(3);%            GO TO THE ROUTE PROCEDURE
            SLIT(3) =ROUTE;EXCHL(3) $ICR;
            STA         0(1);
            TXFEM(0)    ,PAR1;
%         NOW REARRANGE THE MATRIX SO THAT THE LARGEST
%       ELEMENT IS IN THE FIRST ROW.
            LIT(0)      =0,1,0;
            CADD(0)     $D15;%   LIMIT IS MAXR-1, STORED IN $D15
            CSUP(0)     $D1;
            CROTL(0)     24;
PARA1:      LDI(1)     .ADRES;
            CADD(1)     $C0;
            LDA         0(1);
            LDI(1)      .ADRD;%  ADRESS OF TBASE
            CADD(1)     $C0;
            STA         0(1);%   STORE ROWS TEMPORARILY IN TBASE
            TXFEM(0)    ,PARA1;
            LIT(0)      =0,1,0;%  LOOP TO FETCH ROWS MAXR TO N-1
            CADD(0)     $D3;
            CROTL(0)    24;
            CADD(0)     $D15;
PARA2:      LDI(1)      .ADRES;
            CADD(1)     $C0;
            LDA         0(1);
            CSUP(1)     $D15;
            STA         0(1);
```

```
         TXFEM(0)   ,PARA2;
         LIT(0)     =0,1,0;
         CADD(0)    $D15;
         CSUB(0)    $D1;
         CRDTL(0)   24;%       THIS LOOP FETCHES THE ROWS OF
%                            TRASE AND STORES THEM BACK INTO
%                           THE MATRIX UNDER CONSID.
         LDL(3)     .N;%
         CSUB(3)     $D15;%  $C3 CONTAINS ROW INDEX FOR BASE
PARA3:   LDL(1)     .ADRD;
         CADD(1)    $C0;
         LDA        0(1);
         LDL(1)     .ADRES;
         CADD(1)    $C3;
         CADD(1)    $C0;
         STA        0(1);
         TXFEM(0)   ,PARA3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PART TWO:
PAR2::   LDL(0)     .MAXC;%    FIND OUT WHERE THE COLUMN
         CSUB(0)    $D15;%     WENT TO AFTER ABOVE REARR.
         EQLXF(0)   $D1,+1;
         JUMP       OVER1;
%  IF THE TEST IS SATISFIED, THEN THE COLUMN IS IN THE
%  POSITION OF THE SECOND COLUMN, BECAUSE OF THE REARR.
%  IN PART ONE.
         STL(0)     .SAV9;%    SAVE THE NEW COLUMN INDEX
         LDL(1)     .ADRES;%  FETCH ROW MAXC-MAXR
         CADD(1)    $C0;
         LDA        0(1);
         STA        TEMP;
         LIT(2)     =-1,1,0;
         CADD(2)    $C0;
         CSUB(2)    $D1;
PART2:   LDL(3)     .ADRES;%    PULL ROWS ABOVE ROW MAXC-
         CADD(3)    $C2;%     MAXR DOWN BY ONE
         LDA        0(3);
         STA        0(1);
         CSUB(1)    $D1;
         TXFEM(2)   ,PART2;
         LDA        TEMP;
         STA        0(1);%    ROW MAXC-MAXR IS NOW IN PLACE
%                             OF THE SECOND ROW.
%%%%%%%%%%%%%%%%%%%%%%%%
% REARRANGE THE COLUMNS NEXT.BUT SKEW FIRST FOR BETTER ACCESS
         LIT(0)      =0,1,0;
         CADD(0)     $D3;
         CRDTL(0)    24;
         SETE       E.OR.-E;
         SETE1      E.AND.E;
         CLRA;
         LDS        $A;
         LDL(3)     .SPEC;
         LDEE1      $C3;
PART3:  LDL(1)       .ADRES;
         CADD(1)     $C0;
         LDA         0(1);
         STL(0)      .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         STA         0(1);
```

```
         TXFEM(0)    ,PART3;
         LDI(0)      .SAV9;%      MAXC-MAXR
         STI(0)      .SAV2;
         LIT(0)      =-1,1,0;
         CADD(0)     $D30;
         CSUB(0)     $D1;
         LDA         NUMB;%  ADJUST INDEX FOR COL.-FETCH
         CLC(3);
         SLIT(3)    =ROUTE;EXCHL(3) $ICR;
         LDS         $A;
         LDX         $S;
         LDI(1)      .ADRES;
         LDA         *0(1);%   FETCH COLUMN #(MAXC-MAXR)
         STA         TEMP;
PART4:   LDI(2)      .NM0;
         STI(2)      .SAV2;
         LDA         $X;%     ADJUST PICK-UP INDEX
         CLC(3);%            FOR ROWS .SAV9-1 TO 1
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         LDS         $A;
         STS         TEMP1;
         LDA         *0(1);
         LDI(2)      .ONE;
         STI(2)      .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         STA         *0(1);
         LDX         TEMP1;
         TXFEM(0)    ,PART4;
         LDA         TEMP;%     ROW MAXC-MAXR IN TO THE SECOND
         LDI(2)      .N;
         CSUB(2)     $D30;
         CADD(2)     $D1;
         STI(2)      .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         STA         *0(1);
         LIT(0)      =-1,1,0;
         CADD(0)     $D2;
         LDI(1)      .ADRES;
         EGLXT(0)    $D2,PART5;
PART6:   STI(0)      .SAV2;%      UNSKEW MATRIX
         LDA         0(1);
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         STA         0(1);
PART5:   ALIT(1)     =1;
         TXFEM(0)    ,PART6;
OVER1::  LDI(3)      .SAV1;%   RETURN TO MAIN PROGRAM
         EXCHL(3)    $ICR;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MULTPL:  FILL;
%                            MULTIPLICATION OF TWO MATRICES. THE
%                            SET-UP IS SUCH THAT THE ADDRESS OF
%                         THE MATRIX IS TREATED AS A  VARIABLE
         STI(3)      .SAV1;%   SAVE RETURN ADDRESS
         LIT(1)      =0,1,0;
         CADD(1)     $D3;
         CROTL(1)    24;
         LDI(3)      .SPEC;
         LDEF1       $C3;
```

```
MUL:    LDX         NUMB;%        PE-NUMBERS
        LDI(3)      .ADRES;%      =AMATRIX-BASE
        CADD(3)     $C1;
        LDA         0(3);%        LOAD ROW OF AMATRIX
        LIT(0)      =0,1,0;
        CADD(0)     $D3;
        CROTL(0)    24;
        LDS         =0;
MUL1:   LDI(3)      .ADRES1;%     BMATRIX-BASE
        MLRN        *0(3);%       MULTIPLY BMATRIX
        ADRN        $S;%          FORM PARTIAL SUM
        STA         MANT;
        SETE        E.OR.-E;
        SETE1       E.AND.E;
        CLRA;
        LDS         $A;
        LDI(3)      .SPEC;
        LDEE1       $C3;
        LDA         $R;
        EQLXE(0)    $D1,MUL3;%  IF $C0=1 CHECK IF .SAV9=1
        LDI(3)      .SAV9;
        EQLXE(3)    $D1,MUL3;
        CADD(0)     $D3;
        CSUB(0)     $D1;
        CSUB(0)     $D1;
        LDI(2)      .NMD;
        CSUB(2)     $D1;
        SKIP        ,MUL4;
MUL3:   LDI(2)      .ONE;
MUL4:   STI(2)      .SAV2;
        CLC(3);
        SLIT(3) =ROUTE;EXCHI(3) $ICR;
        STA         TEMP;
        SETE        E.OR.-E;
        SETE1       E.AND.E;
        CLRA;
        LDS         $A;
        LDI(3)      .SPEC;
        LDEE1       $C3;
        LDA         $X;
        CLC(3);
        SLIT(3) =ROUTE;EXCHI(3) $ICR;
        LDS         $A;
        LDX         $S;
        SETE        E.OR.-E;
        SETE1       E.AND.E;
        CLRA;
        LDS         $A;
        LDI(3)      .SPEC;
        LDEE1       $C3;
        LDS         MANT;
        LDA         TEMP;
        TXLEM(0)    ,+1;
        JUMP        MUL1;
        LDI(2)      .ADRD;
        CADD(2)     $C1;
        STS         0(2);
        TXLEM(1)    ,+1;
        JUMP        MUL;
        LIT(0)      =1,0,0;
        CROTR(0)    24;
```

```
        CADD(0)    $D3;
        CROTL(0)   24;
MUL2:   LDI(3)     .ADRES2;
        CADD(3)    $C0;
        LDI(2)     .ADRD;
        CADD(2)    $C0;
        LDA        0(2);
        STA        0(3);
        TXEEN(0)   ,MUL2;
        LDI(3)     .SAV1;
        EXCHL(3)   $ICR;
%   ************************7*************************************************
MULSA:  FILL;
%                           MULTIPLY ANMAT(TR.) X (BASE X ANMAT);
% IF ANMAT(TR.) CONSISTS OF COSH & SINH, WE USE THESE TWO
% VALUES ONLY TO MULTIPLY ROW I AND ROW I+1, WHERE
% I:=0,2,4,...,N-2. THEN WE ADD ROW I & ROW I+1 TO
% FORM THE NEW ROW I OF BASE.
%        IF ANMAT(TR.) CONSISTS OF COS & SIN, THEN THE
% 2X2 MATRICES, GOING DOWN THE DIAGONAL, WILL BE DIFFERENT
% SO THAT THE COS & SIN HAVE TO BE PULLED AS I INCREASES.
% FURTHER EXPLANATIONS IN THE PROGRAM.
%#######################################################
        STI(3)     .SAV1;%   SAVE THE RETURN ADDRESS
        LIT(0)     =0,2,0;
        CADD(0)    $D3;
        CSUB(0)    $D1;
        CROTL(0)   24;
% SETTING UP THEE LOOP FOR ROW INDEX I;
        LDI(1)     .ADRES1;% ADRES1=ADDRESS OF ANMAT(TR.)
        CSHL(1)    6;
MLS:    EQLXT(0)   $D0,MLS2;
%       CHECK TO SEE WHICH COS & SIN HAS TO
%       BE PICKED IF $C0=I IS NOT EQUAL TO ZERO
 MLS4A: CADD(1)    $D1;%     THE CONTENT OF $C1=ADDRESS
%       OF ANMAT(TR.) IS NOT DISTROYED WITHIN THIS LOOP.
%       IT IS CHANGED ALRIGHT TO FIND THE PROPER ELEMENTS
%       OF ANMAT(TR.).
        LIT(2)     =128;
% GO TWO ROWS DOWN IN ANMAT(TR.)
        CADD(1)    $C2;%     FIND THE NEW ADDRESS FOR ANMAT
MLS2:   LOAD(1)    $C2;%     SAV9#0 AND $C0=0
        STI(2)     .SAV2;%   SAV2=COSH OR COS
        CADD(1)    $D1;
        LOAD(1)    $C2;
        STI(2)     .SAV3;%   SAV3=SIN OR SINH
MLS1::  LDI(2)     .ADRES;%  ADRES=ADDRESS OF BASE
        CADD(2)    $C0;%     $C2=ADR. OF ROW I OF BAE
        LDI(3)     .SPEC;
        LDEE1      $C3;
        LDA        0(2);
        LDI(3)     .SAV2;
        MLRN       $C3;%     ROW I OF BASE TIMES COSH
% OR COS DEPENDING ON CONTENT OF .SAV2
        STA        TEMP;
        CADD(2)    $D1;%     $C2=ADR. OF ROW I+1 OF BASE
        LDA        0(2);
        LDI(3)     .SAV3;
        MLRN       $C3;%     ROW I+1 TIMES SINH OR SIN
        ADRN       TEMP;
        LDI(2)     .ADRES2;% ADRES2=ADR. OF TBASE
```

```
          CADD(2)    $C0;
          STA        0(2));
          LDI(2)     .SAV9;
          EQLXT(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 -SIN.
          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.SQR.MATRX TBASE
          CADD(2)    $C0;
          CADD(2)    $D1;%      FIND ROW I+1;
          STA        0(2));
          LDL(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.
          EQLXF(2)   $D0,MLS6;
          TXEEM(0)   ,MLS1;
          SKIP       ,MLS7;
MLS6:     TXEEM(0)   ,MLS4A;
% REASTABLISH BASE BY LOADING TBASE INTO BASE
MLS7:     LIT(0)     =0,1,0;
          CADD(0)    $D3;
          CRDTL(0)   24;
MLS5:     LDI(1)     .ADRES2;
          LDI(2)     .ADRES;
          CADD(1)    $C0;
          LDA        0(1));
          CADD(2)    $C0;
          STA        0(2));
          TXEEM(0)   ,MLS5;
          LDI(3)     .SAV1;
          EXCHL(3)   $ICR;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TRASPOS: FILL;
%                    PROCEDURE TRASPOS FINDS THE TRANSPOSE
%                    OF THE ANGLE-MATRIX BY CHANGING THE
%                    SIGN OF THE OFF-DIAGONAL ELEMENTS
          STI(3)     .SAV1;%  SAVE RETURN ADRESS
          LDI(0)     .ANTUN;
          LDEE1      $C0;
          LDA       NUMB;%      FIND THE INDEX FOR THE ELEMENTS
%                               THAT NEED THE SIGN CHANGE
        LDS          =1;
        ADM         $S;
```

```
        STA        TEMP;
        LDA        NUMB;
        CSHR(0)    1;
        RTL        $A,1;
        LDFF1      $C0;
        LDA        $R;
        STA        TEMP;%     THE INDEX IS FOUND
        LDL(0)     .SPEC;
        LDFF1      $C0;
        LDS        TEMP;
        LDL(3)      .ADRC;
        LDA         #0(3);;
        CHSA;
        STA      #0(3);;
        LDL(3)   .SAV1;%    RETURN
        EXCHL(3) $ICR;
%   ****************************************************************************
ANGL:     FILL;
%     PROCEDURE ANGL FINDS COSINE AND SINE, TO FORM THE
% TRANSFORMATION MATRIX ANMAT.
%      SINE=SQRT(0.5-C[2K-1,2K]/H).
%      COSINE=SQRT(0.5+C[2K-1,2K]/H).
%      H=SQRT(4XC[2K-1,2K]*2+(C[2K-1,2K-1]-C[2K,2K])*2.
% THERE ARE DIFFERENT CASES TO BE CONSIDERED FOR
% THE CALCULATION OF COSINE. THEY WILL BE DISCUSSED AS THEY
% COME UP IN THE PROGRAM.
%###################################################################
         STL(3)     .SAV1;%    SAVE RETURN ADRESS
         LIT(0)     =0,1,0;
         CADD(0)    $D3;
         CROTL(0)   24;
         LDL(1)     .ADRC;
         LIT(3)     =0.;
         LDL(2)     .SPEC;
         LDFF1      $C2;
         LDA         $C3;
AG:      STA        0(1);;
         CADD(1)    $D1;
         TXFEM(0)   ,AG;%      MAKE SURE THAT THE OFF-DIAGO
%                                   NAL ELEMENTS ARE ZERO.
         LIT(0)     =0,2,0;
         CADD(0)    $D3;
         CSUB(0)    $D1;
         CROTL(0)   24;
         CLC(1);
AG1:     CCR(1)     0(0);;%    CREATE THE TURN ON PATTERN
%                             FOR THE OFF-DIAG. PICK-UP
         TXFEM(0)   ,AG1;
         STL(1)     .ANTUN;
         LDFF1      $C1;
         LDA        NUMB;
         LIT(2)     =1;
         ADM        $C2;
         STA        TEMP;
         LDA        NUMB;
         RTL        $A,1;
         CSHR(1)    1;
         LDFF1      $C1;
         LDA        $R;
         STA        TEMP;%    INDEX IS FOUND AND IT IS OF THE
% FORM 2K,2K-1;2K-1,2K. I.E.1,0,3,2,5,4,......
```

```
% NOW WE CALCULATE
% A)  HSQR=4XC[2K-1,2K]*2+(C[2K-1,2K-1]-C[2K,2K])*2
% B)  H     =SQRT(HSQR)
% C)  SINE=SQRT(0.5-C[2K-1,2K]/H)
% TO A);
        LDS         TEMP;%    INDEX IN $S FOR OFF-DIAGS.
        LDL(2)      .ADRCM;%  ADRESS OF MATRIX IN USE
        LDA         *0(2);%   C[2K-1,2K] LOADED AND
        STA         TEMP1;%   SAVED HERE FOR LATER CASE
%                             DIFFERENTIATION FOR COSINE
        LDL(3)      .NMO;
        STL(3)      .SAV2;
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        CSHL(1)     1;
        LDEE1       $C1;
        STA         TEMP1;%  TEMP1 HAS IN EVERY PE C[2K-1,2K]
%                            IN PAIRS OF TWO.
        LDL(1)      .SPEC;
        LDEE1       $C1;
        LDA         TEMP1;
        MLRN        TEMP1;%  $A=(C[2K-1,2K])*2
        LIT(3)      =4.;
        LDS         $C3;
        MLRN        $S;%     $A=4X(C[2K-1,2K])*2
        STA         TEMP2;
        LDX         NUMB;%   INDEX FOR THE DIAG.ELEMENTS
        LDA         *0(2);%  C[I,I] LOADED
        LDL(1)      .ANTUN;
         LDEE1       $C1;
       STA         GERSHG;% SAVE C[2K-1,2K-1]
        CLC(3);
        SLIT(3) =ROUTE;EXCHL(3) $ICR;
        LDS         $A;
        LDEE1       $C1;
       LDA         GERSHG;
        SBRN        $S;
        STA         GERSHG;%  C[2K-1,2K-1]-C[2K,2K]
        RTL         $A,1;
        CSHR(1)     1;
        LDEE1       $C1;
        LDA         $R;
        STA         GERSHG;
% ABOVE VALUES IN ALL PES OF GERSHG IN PAIRS OF TWO
        LDL(1)      .SPEC;
        LDEE1       $C1;
        LDA         GERSHG;
        MLRN        GERSHG;%  $A=(C[2K-1,2K-1]-C[2K,2K])*2
        ADRN        TEMP2;%   $A=HSQR
% TO B);
        CALL SQRT64();
        STA         TEMP2;%    H STORED HERE
        LIT(0)      =0.;
        IAL         $C0;
        SETE        I.AND.E;
        JAG         $C0;
        SETE        J.OR.E;
        SETE1       E.AND.E;
% TO C);
        LDA         TEMP1;
        DVRN        TEMP2;
```

```
        STA         TEMP2;%     $A=C[2K-1,2K]/H.
        LDI(1)      .SPEC;
        LDEF1       $C1;
        LIT(0)      =0.5;
        LDA         $C0;
        SBRN        TEMP2;%     $A=SINE=0.5-C[2K-1,2K]/H
        CALL  SQRT64();
        LDS          TEMP;%    $S=INDEX FOR C[2K,2K-1] & C
%                              C[2K-1,2K]
        LDI(2)      .ANTUN;
        LDEF1       $C2;
        CHSA;
        LDEF1       $C1;
        LDI(3)      .ADRC;
        STA         *0(3);%  SINE INTO ANMAT
%#C#########################
% NOW HEAR THIS: CASE DIFFERENTIATION FOR COSINE
% CASE ONE: C[2K-1,2K-1] - C[2K,2K] LSS 0 OR GTR 0
        LDA         GERSHG;
        LIT(0)      =0.;
        JAL         $C0;%      $A LSS 0
        JAG         $C0;%      $A GRT 0
        SETF        I.AND.E;
        SETC(0)     F;
        STI(0)      .SAV9;
        SETF        J.OR.F;
        SETC(0)     E;
        CAND(0)     $D5;%      CLEAN OUT WHERE $A EQUAL TO ZERO
        ZERT(0)     ,AG2;%  NO C[2K-1,2K-1]-C[2K,2K] LSS OR
%                            GRT ZERO, SO SKIP TO AG2
        SETF1       E.AND.E;
        LIT(1)      =0.5;
        LDA         $C1;
        ADRN        TEMP2;
        CALL  SQRT64();%       $A=SQRT(0.5+C[2K-1,2K]/H)
        LDI(1)      .SAV9;
        LDEF1       $C1;
        CHSA;%                  $A LSS 0 WHERE I BITS WRE SET
        LDEF1       $C0;
        LDX         NUMB;
        LDI(2)      .ADRC;
        STA         *0(2);%     COSINE STORED DIAGONALLY WHERE
% C[2K-1,2K-1] - C[2K,2K] LSS 0 OR GRT 0
%#C########################
% CASE TWO: C[2K-1,2K-1] - C[2K,2K] EQL 0
AG2:    COMPC(0);
        LDI(1)      .SPEC;
        CAND(1)     $C0;
        ZERT(1)     ,AG5;%     IF NONE ARE ZERO FINISHED
        LDEF1       $C1;%    PES TURNED ON WHERE CASE TWO = 0
% CASE 2A:  C[2K-1,2K] GRT 0
        LIT(0)      =0.;
        LDA          TEMP1;
        JAG         $C0;
        SETF        I.AND.E;
        SETC(3)     F;
        ZERT(3)     ,AG3;%   NONE ARE GRT 0
        SETF1       F.AND.F;
        LIT(0)      =1.;
        LDA          $C0;
        LDX         NUMB;
```

```
         LDI(2)    .ADRC;
         STA       *0(2);%  COSINE = 1 WHERE C[2K-1,2K] GRT 0
% CASE 2B; C[2K-1,2K] LSS 0
AG3;     COMPC(3);
         CAND(3)   $C1;
         STI(3)    .SAV9;
         LDF       $C3;
         LIT(0)    =0.;
         IAI       $C0;
         SETF      I.AND.E;
         SETC(0)   E;
         ZERT(0)   ,AG4;%    NONE ARE LSS 0
         SETF1     F.AND.E;
         LIT(1)    =0.;
         LDA       $C1;
         LDX       NUMB;
         LDI(2)    .ADRC;
         STA       *0(2);% COSINE=0., WHERE C[2K-1,2K] LSS 0
% CASE 2C;  C[2K-1,2K] = 0
AG4;     COMPC(0);
         LDI(3)    .SAV9;
         CAND(3)   $C0;
         ZERT(3)   ,AG5;%    NONE ARE EQL 0
         LDFF1     $C3;
         LIT(1)    =1.;
         LDA       $C1;
         LDX       NUMB;
         LDI(2)    .ADRC;
         STA       *0(2);%   COSINE=1., WHERE C[2K-1,2K]=0
         LIT(1)    =0.;
         LDA       $C1;
         LDS       TEMP;
         STA       *0(2);
AG5;     LDI(3)    .SAV1;%   RETURN TO THE OUTSIDE WORLD
         EXCHL(3)  $ICR;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
HYANG;   FILL;
%       PROCEDURE HYANG CALCULATES;
%   D=A[2I-1,2K-1]-A[2I,2K].
%   E=A[2I-1,2K]-A[2I,2K-1].
%   K2=SUM(D X E).
%   K1=SUM(D*2+E*2).
%   TANH=-2K2/K1.
%   TANSQ=TANH*2.
%   D1=1./SQRT(1.-TANSQ).
%   E1=SQRT[(D+1.)/2.].
%   COSH=SQRT[(E+1.)/2].
%   D2=SQRT[[(E+1.)/2]-1].
%   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.-SWRTQRT(1-TANSQ)).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         STI(3)   .SAV1;%  SAVE RETURN ADRESS
         SETF      F.OR.-F;
         SETF1     E.AND.E;
         CLRA;
         STA       KAP1;%    CLEAR KAP1 AND KAP2
         STA       KAP2;
```

```
        LIT(0)    =0,2,0;% LOOP FOR FETCHING THE PROPER ROWS
*                         OF THE MATRIX UNDER CONSIDERATION
        CADD(0)   $D3;
        CSHR(0)   $D1;
        CROTL(0)  24;
HYP::   LDI(1)    .ANTUN;% EVERY OTHER PE TURNED ON
        LDI(2)     .SPEC;
        LDEE1      $C2;
        LDX       $C0;%     INDEX FOR THE A[2I-1,2K-1]
        CSHR(1)    1;
        LDEE1      $C1;
        LIT(2)    =1;
        LDA       $X;
        ADM       $C2;
        LDR        $A;
        LDX       $R;%  INDEX FOR A[2I,2K]
        LDI(2)    .SPEC;
        LDEE1     $C2;
        LDI(2)    .ADRA;% ADRESS OF BASE.
        LDA       *0(2);
        CHSA;
        LDI(1)    .ANTUN;
        LDEE1     $C1;
        STA       TEMP;%  -A[2I-1,2K-1] STORED HERE
        LDI(3)    .NMO;
        STI(3)    .SAV2;
        CLC(3);
        SLIT(3)  =ROUTE;EXCHL(3) $ICR;
        LDEE1     $C1;
        SBRN      TEMP;%  -A[2I,2K]-(-A[2I-1,2K-1]).
        STA       TEMP;
        RTL       $A,1;
        CSHR(1)   1;
        LDEE1     $C1;
        STR       TEMP;%  TEMP:=D
        LDI(1)    .SPEC;
        LDEE1     $C1;
        LDA       TEMP;
        MLRN      TEMP;
        STA       TEMP1;% TEMP1:=D*2
        LDI(3)    .ONE;
        STI(3)    .SAV2;
        LDA       $X;
        CLC(3);
        SLIT(3)  =ROUTE;EXCHL(3) $ICR;
        LDS       $A;
        LDX       $S;%    INDEX FOR A[2I-1,2K],A[2I,2K-1]
        LDA       *0(2);% A[2I-1,2K],A[2I,2K-1] LOADED
        CHSA;
        LDI(1)    .ANTUN;
        CSHR(1)   1;
        LDEE1     $C1;
        STA       TEMP2;
        CLC(3);
        SLIT(3)  =ROUTE;EXCHL(3) $ICR;
        LDEE1      $C1;
        SBRN      TEMP2;%  -A[2I,2K-1]-(-A[2I-1,2K])
        STA       TEMP2;
        LDI(3)    .NMO;
        STI(3)    .SAV2;
        CLC(3);
```

```
         SLITT(3) =ROUTE;EXCHI(3) $ICR;
         CSHI(1)  1;
         LDEF1    $C1;
         STA      TEMP2;%   TEMP2:=F.
         LDI(1)   .SPEC;
         LDEF1    $C1;
         LDA      TEMP2;
         MLRN     TEMP2;
         STA      GERSHG;%  GERSHG:=E*2.
         LDA      TEMP;
         MLRN     TEMP2;%   D X F.
         ADRN     KAP2;
         STA      KAP2;%  KAP2:=K2.
         LDA      TEMP1;
         ADRN     GERSHG;% D*2+E*2.
         ADRN     KAP1;
         STA      KAP1;%   KAP1:=K1.
         TXFTM(0)  ,+1;
         JUMP      HYP;
         LIT(0)    =0;
HYP2::   SETE      E.OR.-F;
         SETE1     E.AND.F;
         CLRA;
         LDI(1)    .ANTUN;
         LDEF1     $C1;
         LDA       KAP2(0);
         LIT(2)    =1;
         SETE      E.OR.-F;
         SETE1     E.AND.F;
HYP1:    LDS       $A;%       LOG-SUM FOR KAP2 AND KAP1
         RTL       $S,0(2);
         LDS       $R;
         ADRN      $S;
         CADD(2)   $C2;
         LIT(3)    =64;
         EQLXE(2)  $C3,HYP1;%  END SUMING
         STA       KAP2(0);
         ALIT(0)   =1;
         GRTRT(0)  $D1,+1;
         JUMP      HYP2;
         LDI(1)    .SPEC;
         LDEF1     $C1;
         LIT(0)    =-2.;
         LDA       $C0;
         MLRN      KAP2;
         DVRN      KAP1;
         STA       KAP2;%   KAP2:=TANH
         SAP;
         LIT(0)    =1.;%  CHECK IF ABS(TANH-1) LSS 1.@-12.
% IF SO GO OUT AND TRY A NEW MATRIX BASE, FORMED BY THE TRANS-
% FORMATION FOUND UNDER ANGL.
         SBRN      $C0;
         SAP;
         LIT(0)    =0.000000000001;
         IAL       $C0;
         SETE      I.AND.F;
         SETC(0)   E;
         COMPC(0);
         CAND(0)   $D5;
         STI(0)    .MAX;
         ZERE(0)   ,+1;
```

```
        JUMP       HYP3A;
        LDEF1      $C1;
        LDA        KAP2;
        MLRN       KAP2;%   TANSQ:=TANH*2
        STA        GERSHG;%STORE TANSQ FOR CALC. OF FINVAL
        CHSA;
        LIT(0)     =-1.;
        SBRN       $C0;%    $A:=-TANSQ-(-1.)
        CALL SQRT64();
%                            $A=SQRT(1.-TANSQ)
        LDS        $A;
        CCR(0)     0;
        LDA        $C0;
        DVRN       $S;%   $A:=1./SQRT(1.-TANSQ)
        ADRN       $C0;
        LIT(0)     =2.;
        DVRN       $C0;
        CALL SQRT64();
%                            F1 FOUND
        LIT(0)     =1.;
        ADRN       $C0;
        LIT(2)     =2.;
        DVRN       $C2;
        CALL SQRT64();
        LDX        NUMB;
        LDL(2)     .ADRC;%  COSH STORED IN ANMAT
        STA        *0(2);
        MLRN       $A;%       $A:=(COSH)*2
        SBRN       $C0;
        CALL SQRT64();
        LDS        $A;
        LDA        KAP2;
        LIT(0)     =0.;
        IAL        $C0;
        SETF       I.AND.F;
        SETF1      E.AND.F;
        LDA        $S;
        CHSA;
        LDS        $A;
        LDEF1      $C1;
        STS        KAP2;%   SINH FOUND
        LDL(0)     .ANTUN;
        LDEF1      $C0;
        LDA        NUMB;
        LIT(3)     =1;
        ADM        $C3;
        LDS        $A;
       LDA        NUMB;
        RTL        $A,1;
        CSHR(0)    1;
        LDEF1      $C0;
        LDS        $R;
        LDEF1      $C1;
        LDA        KAP2;
        LDL(2)     .ADRC;
        STA        *0(2);
%   SINH AND COSH ARE STORED PROPERLY. FIND FINVAL
        LIT(0)     =1.;
        LDA        $C0;
        SBRN      GERSHG;% $A:=1.-GERSHG;GERSHG:=TANSQ
        CALL SQRT64();
```

```
         CHSA;
         CCR(0)     0;%        $C0:=-1.
         SBRN       $C0;
         MLRN       KAP1;
         LIT(0)     =0.5;
         MLRN       $C0;
         STA        KAP2;
         CLC(3);
         SLIT(3)    =KAP2;
         LOAD(3)    $C0;
         STL(0)     .FINVAL;
HYP3A::  LDL(3)     .SAV1;
         EXCHL(3)   $ICR;
%%C#####################################################################
NULCHK: FILL:
%                      THIS PROCEDURE CHECKS:
%  ABS(C[2K-1,2K-1]-C[2K,2K]) LEQ 1.0-12.
%  IF SO, IT THEN FINDS THE SIGN OF C[I,I] OPPOSITE OF THAT
%  OF C[0,0] AND THE ROWINDEX I=.MAX. IT THEN LEADS INTO
%  THE PROCEDURE SHFT WHICH EXCHANGES THE SECOND ROW
%  WITH ROW I. THE PROCEDURE SHFT IS ALSO USED FOR THE
%  MATRIX BASE.
%###################################################################
         STL(3)     .SAV1;%    SAVE RETURN ADRESS
         LDL(0)     .SPEC;
         LDEE1      $C0;
         LDX        NUMB;
         LDL(1)     .ADRCM;
         LDA        *0(1);%    LOAD DIAGONAL ELEMENTS
         CHSA;
         CLC(2);
         LIT(3)     =0,2,0;%   CREATE PATTERN TO TURN ON
%                              THE EVEN PES.
         CADD(3)    $D3;
         CSUB(3)    $D1;
         CROTL(3)   24;
NUL:     CCR(2)     0(3);
         TXEEM(3)   ,NUL;
         STL(2)     .ANTUN;
         LDEE1      $C2;
         STA        TEMP;%     -C[2K-1,2K-1] SAVED
         LDL(3)     .NMO;
         STL(3)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         SBRN       TEMP;%    DIFFERENCE OF OFF DIAGS IN EVEN
         LDEE1      $C2;%     PES
         STA        TEMP;%    -C[2K,2K]-(-C[2K-1,2K-1])
         LDL(3)     .ONE;
         STL(3)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         CSHR(2)    1;
         LDEE1      $C2;
         STA        TEMP;
         LDEE1      $C0;
         LDA        TEMP;
         SAB;%                TAKE ABS. VAL. WHERE NECESS.
         LDL(1)     .TNMTW;
         LDEE1      $C0;
         IAG        $C1;
```

```
          SETF        -I.AND.E;
          SETC(3)     E;
          STI(3)      .INDEX;
          ZERE(3)     ,+1;%       NO EXCHANGE NECESSARY IF TEST IS
          JUMP        NUL5;
%                                 SATISFIED.
          LDEE1       $C0;%       FIND THE SIGN OF C[0,0]
          LDX         NUMB;
          LDI(2)      .ADRCM;
          STI(2)      .ADRES;
          LDA         *0(2);;%    LOAD DIAGS. OF CMAT
          ISN;
          SETE        I.AND.E;
          SETC(1)     E;
          LDI(2)      .ENB;
          CAND(2)     $C1;
          ZERE(2)     ,NUL2;%     IF TEST IS SATISFIED, C[0,0]
%                                 IS NEGATIVE, ELSE POS.
          LEADO(1);
          SKIP        ,+1;
NUL2:     LEADZ(1);
          LIT(2)      =77:8;
          CAND(2)     $C1;%      THE INDEX OF C[I,I] DIFFERENT
%                           IN SIGN FROM C[0,0] IS FOUND
          STI(2)      .MAX;
          EQLXE(2)    $D1,+1;
          JUMP        NUL5;
          SKIP        ,+2;
SHFT:     FILL;
          STI(3)      .SAV1;
          LDI(0)      .SPEC;
          LDEE1       $C0;
          LDI(2)      .ADRES;%  EXCHANGE ROW 1 WITH ROW I
          CADD(2)     $D1;
          LDA         0(2);
          STA         TEMP;
          LDI(3)      $C2;
          CADD(3)     $D7;
          CSUB(3)     $D1;
          LDA         0(3);
          STA         0(2);
          LDA         TEMP;
          STA         0(3);;%    END EXCHANGE ROWS. TO EXCHANGE
% COLUMNS SKEW MATRIX FIRST,.
          LIT(0)      =0,1,0;
          CADD(0)     $D3;
          CROTL(0)    24;
NUL3:     LDI(1)      .ADRES;
          CADD(1)     $C0;
          LDA         0(1);
          STI(0)      .SAV2;
          CLC(3);
          SLIT(3) =ROUTE;EXCHI(3)  $ICR;
          STA         0(1);
          TXEEM(0)    ,NUL3;
% NOW EXCHANGE SECOND COLUMN WITH COLUMN I
          LDA         NUMB;
          LDI(1)      .ONE;
          STI(1)      .SAV2;
          CLC(3);
          SLIT(3) =ROUTE;EXCHI(3) $ICR;
```

```
         LDS        $A;
         LDX        $S;%   COLUMN INDEX FOR COLUMN 1
         LDI(1)     .MAX;
         CSUB(1)    $D1;
         STI(1)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         STA        TEMP1;%COLUMN INDEX FOR ROW I=.MAX
         LDI(1)     .ADRES;
         LDA        *0(1);%    SECOND ROW LOADED
         CLC(3);
         SLIT(3)    =ROUTE;EXCHL(3) $ICR;%  D=.MAX-1
         STA        TEMP;.
         LDS        TEMP1;
         LDA        #0(1);%     COLUMN I=MAX LOADED
         LDI(2)     .N;
         CSUB(2)    $D7;
         CADD(2)    $D1;
         STI(2)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;%  D=N-(MAX-1)
         STA        *0(1);%  COLMN. I STORED IN COL. 1
         LDA        TEMP;
         LDS        TEMP1;
         STA        #0(1);%  COL. 1 STORED INTO COL. I.
% COLUMNS ARE EXCHANGED , NOW UNSKEW THE MATRIX
         LIT(0)     =-1,1,0;
         CADD(0)    $D3;
         LDI(2)     .ADRES;
         CADD(2)    $D1;
NUL4:    LDA        0(2);
         STI(0)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         STA        0(2);
         ALIT(2)    =1;
         TXEEM(0)   ,NUL4;
NUL5::   LDI(3)     .SAV1;
         EXCHL(3)   $ICR;
%%C###############################################################
SYASY:   FILL;
%        THIS PROCEDURE MAKES THE MATRIX TRULY SYMMETRIC
% BY STORING (ABS(A[I,J]) + ABS(A[J,I]))/2 INTO
% THE A[I,J] & A[J,I] POSITIONS OF BASE. AT THE
% SAME TIME AN ERROR MATRIX EPS IS CREATED BY FIN-
% DING THE DEVIATION FROM A[I,J](OLD) & A[J,I](OLD)
% AND THE ABOVE AVERAGE.
         STI(3)     .SAV1;
         CLC(3);
         STI(3)     .ANTUN1;
%        ANTUN1 WILL CONTAINA TAG. IF :=0, THEN A[I,J]:=A[J,I]
% AND BASE WILL HAVE REAL EIGENVALUES. IF :=1 THEN
% AT LEAST ONE A[I,J]:=-A[J,I] AND BASE WILL HAVE
% COMPLEX EIGENVALUES, WHICH MEANS A PROCEDURE HAS TO BE
% CHOSEN THAT CAN HANDLE COMPLEX EIGENVALUES.
         LIT(0)     =0,1,0;%  SET UP MAIN LOOP
%        THE DIAGONALS FROM 1 TO N/2 PARTICIPATE
%        IN THE FOLLOWING ALGORITHM.
         LDI(1)     .N;%       $C1:=N
         CSHR(1)    1;%        $C1:=N/2
         CADD(0)    $C1;
```

```
         CRDTL(0)   24;
         CADD(0)    $D1;
ASY6::   LDI(1)     .SPEC;
         LDEF1      $C1;
         LDA        NUMB;
         STI(0)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHI(3) $ICR;
         LDS        $A;
         STS        TEMP2;
         LDX        $S;%       INDEX FOR THE ELEMENTS IN THE
%                              DIAGONALS TO THE RIGHT OF THE MAIN DIAGONAL
         LDI(2)     .ADRA;
         LDA        *0(2);
         ISN;
         SETH       I.AND.F;% H HAS THE SIGN PATTERN OF
%                              THE ELEMENTS UNDER CONSIDERATION
         SETC(2)    H;%        $C2 IS NEEDED TO CHECK THE
%                              DIFFERENCE IN SIGN PERFORMED LATER
         SAP;
         STA        TEMP;%     :=A[I,J](OLD)
         LDI(3)     .N;
         CSUB(3)    $C0;
         STI(3)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHI(3) $ICR;
         LDS        $A;%       INDEX FOR THE ELEMENTS IN THE
%                              DIAGONALS TO THE LEFT OF THE MAIN DIAGONAL
         STS        GERSHG;
         LDI(3)     .ADRA;
         LDA        *0(3);
         STI(0)     .SAV2;
         CLC(3);
         SLIT(3) =ROUTE;EXCHI(3) $ICR;
         JSN;
         SETJ       J.AND.F;% SAME SIGN CHECK AS BEFORE
         SETC(1)    J;
         CEXOR(2)   $C1;
         ZERT(2)    ,ASY2;%    CHECK FOR DIFFERENCE IN SIGN
         LDI(2)     .N;
         STI(2)     .ANTUN1;
ASY2:    SAP;
         STA        TEMP1;
         ADRN       TEMP;
         LIT(2)     =2.0;
         DVRN       $C2;%      FIND AVERAGE
         LDS        $A;%       A[I,J](NEW) & A[J,I](NEW)
         LDA        TEMP;
%        FIND DEVIATION FROM OLD AND NEW VALUES OF BASE
         SBRN       $S;%       :=A[I,J](OLD) - A[I,J](NEW)
         SETF       H.AND.F;
         SETF1      E.AND.E;
         CHSA;
         LDI(1)     .SPEC;
         LDEF1      $C1;
         LDX        TEMP2;
         LDI(2)     .ADRE;%    $C2:=ADDRESS OF ERROR MATRIX
         STA        *0(2);
         LDA        $S;%       $A=AVERAGE
         SETF       H.AND.F;
         SETF1      E.AND.F;
```

```
         CHSA;
         LDFF1      $C1;
         LDI(2)     .ADRA;%   STORE A[I,J](NEW)
         STA        *0(2);
         LDA        TEMP1;
         SBRN       $S;%      A[J,I](OLD) - A[I,J](NEW)
         SETF       J.AND.F;
         SETF1      F.AND.F;
         CHSA;
         LDFF1      $C1;
         LDL(2)     .N;
         CSUP(2)    $C0;
         STS        TEMP;%    SAVE A[I,J](NEW) TEMPORARILY
         STI(2)     .SAV2;
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         LDI(2)     .ADRE;
         LDS        GERSHG;%  STORE A[J,I](OLD) - A[I,J](NEW)
         STA        *0(2);
         LDA        TEMP;
         SETF       J.AND.F;
         SETF1      F.AND.E;
         CHSA;
         LDFF1      $C1;
         CLC(3);
         SLIT(3) =ROUTE;EXCHL(3) $ICR;
         LDL(2)     .ADRA;
         LDS        GERSHG;%  STORE A[J,I](NEW)
         STA        *0(2);
         TXETM(0)   ,+1;
         JUMP       ASY6;
         LDI(3)     .SAV1;
         EXCHL(3)   $ICR;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
AVER:          FILL;
%                       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 2 WITH ROW I IN SHET, 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]) LSS 1..
               STI(3)     .SAV1;%  SAVE THE RETURN ADRESS
         LIT(0)     =0,1,0;%  LOOP FOR PICKING UP THE
% OFF - DIAGONAL ERLEMENTS OF CMAT. CMAT IS SYMMETRIC.
               LDL(1)     .N;%      N:=ORDER OF THE MATRIX
               CSHR(1)    1;%       DIVIDE BY TWO(2)
               CSUR(1)    $D1;%     $C1:=(N/2)-1
               CADD(0)    $C1;
               CROTL(0)   24;%      THIS LOOP ENABVLES 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.OR.-E;
               SETF1      F.AND.F;
               CLRA;
               STA        TEMP;%   CLEAN UP TEMP;
AV:            LDL(3)     .SPEC;
               LDFF1      $C3;
```

```
            LDA         NUMB;
            STL(0)      .SAV2;
            CLC(3);
            SLIT(3) =ROUTE;EXCHL(3) $ICR;
            LDS         $A;%    $S:= INDEX FOR OFF-DIAGS.
            LDL(2)      .ADRCM;%    ADRESS OF CMAT
            LDA         #0(2);
            SAP;
            ADRN        TEMP;%      FORM PARIAL SUM
            STA         TEMP;
            TXFEM(0)    ,AV;
            LDL(3)      .SPEC;
            LDEE1       $C3;
            LDA         NUMB;%  INDEX FOR THE REST N/2
%  ELEMENTS OF CAMT NOT YET CONSIDERED
            RTL         $A,0(0);
            CSHR(3)     0(0);
            CAND(3)     $D5;
            LDEE1       $C3;
            LDS         $R;
            LDA         #0(2);
            SAP;
            ADRN        TEMP;
            STA         TEMP;
% ALL OFF-DIAGONAL ELEMENTS HAVE BEEN LOOKED AT (I.E. UPPER
% HALF OF CMAT) AND THEY ARE PARTIALLY SUMMED.
% NOW WE LOG-SUM,I.E., ACROSS PES
            LIT(2)      =1;
            SETE        E.OR.-E;
            SETE1       E.AND.E;
            LDA         TEMP;
AV1:        LDS         $A;
            RTL         $S,0(2);
            LDS         $R;
            ADRN        $S;
            CADD(2)     $C2;
            LIT(3)      =64;
            EQLXE(2)    $C3,AV1;%   END SUMMING
            LDL(2)      .G;
            SBRN        $C2;
% LET G1 BE THE SUM OF THE OFF-DIAGS. AND G BE THE LARGEST
% OFF-DIAGONAL ELEMENT: THEN $A CONTAINS G1 - G. NOW WE
% AVERAGE. THE NUMBER OIF ELEMENTS INVOLVED IS ((N*2-N)/2)-1
            LDL(0)      .N;
            LEADD(0);
            LIT(1)      =77;8;
            CAND(0)     $C1;
            LIT(1)      =16;
            CSUB(0)     $C1;
            LIT(1)      =47;
            CSUB(1)     $C0;
            LDL(0)      .N;
            CSHL(0)     0(1);
            CSUB(0)     $D2;%           $C0:=N*2-N
            CSHR(0)     1;%             $C0:=(N*2-N)/2
            CSUB(0)     $D1;%           $C0:=((N*2-N)/2)-1
            LDL(3)      $C0;%       SAVE $C0 TEMPORARILY
% CONVERT FIXED $C0 INTO FLOAT $C0
            LEADD(0);
            LIT(1)      =77;8;%
            CAND(0)     $C1;
```

```
              LIT(1)     =16;
              CSUB(0)    $C1;
              CSHL(3)    0(0);%        MANTISSA-PART ADJUSTED
              LIT(1)     =48;
              CSUB(1)    $C0;%       CALCULATE THE EXPONENT-PART
              LIT(0)     =48;
              CSHL(1)    0(0);
              CEXOR(3)   $C1;%    EXPONENT & MANTISSA JOINED
              LDL(1)     .ENR;
              CSHR(1)    1;
              CEXOR(3)   $C1;%    SIGN OF EXPONENT JOINED
              DVRN       $C3;
              LDL(0)     .G;
              DVRN       $C0;
              SAP;%                  $A;=ABS(G1/G)
              LIT(1)     =0.4;
              TAL        $C1;
              SETC(0)    I;
              CLC(2);
              ONEST(0)    ,AV2;%   IF ALL ONE"S THEN
%                    THE OFF-DIAGONALS ARE NOT IN RANGE
              LIT(2)     =1;
AV2:          LDL(3)     .SAV1;
              EXCHL(3)   $ICR;% RETURN TO MAINPROGRAM WITH
% TEST RESULT IN $C2.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
FBFRI[ENTRY]:    FILL;%    SAVE THE CONTENT OF ACAR0,ACAR1,
                     %      AND ADP-LOCATION $D32 - $D39
          STI(3)     .RETUR;%  SAVE RETURN ADDRESS
          CLC(3);
          SLIT(3)    =ADRSAV+8;
          STORE(3)   $C0;
          ALIT(3)    =1;
          STORE(3)   $C1;
          CLC(3);
          SLIT(3)    =ADRSAV;
          LIT(0)     =1,7,0;
SA;       STORE(3)   $D32(0);
          TXEEM(0) ,SA;
          LIT(0)     =1,5,0;
SA1:      CLC(3);
          LOAD(2)   $C3;%  $C2 CONTAINS ADDRESS OF LIST WHICH
%                          IN TURN CONTAINS THE ADRESSES 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;%
          TXEEM(0)   ,SA1;
          CLC(3);
          LOAD(2)    $C3;
          STI(3)     .N;
          LIT(0)     =1;
          STI(0)     .ONE;
          LIT(0)     =0;
          STI(0)     .ZERO;
% ***********************************
          LDI(0)     .N;
          CSUB(0)    $D1;
          STI(0)     .NMO;
          LIT(0)     =64;
```

```
        CSUB(0)    $D2;
        STI(0)     .ROUT;
        LIT(0)     =8000000000000000:16;
        STI(0)     .FNB;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        SETE       E.OR.-E;%    CREATE THE TURN ON PATTERN
%                               FOR THE FIRST N PES
        SETE1      E.AND.E;
        LDA        NUMB;
        LDI(0)     .N;
        TAI        $C0;
        SETC(0)    I;
        CLRA;
        STI(0)     .SPEC;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        LDEE1      $C0;%    CREATE FREIG INITIALLY AS AN
        LIT(1)     =1.;%       IDENTITY- MATRIX
        LDA        $C1;
        LDX        NUMB;
        LDI(0)     .ADRE;
        STA        *0(0);
        CLRA;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        LIT(0)     =0.000001;
        STI(0)     .TNMTW;
        CLC(3);
        STI(3)     .AB;
        STI(3)    .FINVAL1;
COMP6:: LDI(3)     .ADRA;
%  NOW WE PERFORM CMAT:=BASE X BASE(TR.) - BASE(TR.) X BASE
%  BUT FIRST: BASE X BASE(TR.X BASE(TR.) - BASE(TR.) X B)
        STI(3)     .ADRES;
        LDI(3)     .ADRCM;
        STI(3)     .ADRES1;% ADRES1 GIVES THE ADRESS OF THE
%                            MATRIX WHICH WILL CONTAIN
%                            BASE X BASE(TR.). HERE ADRES1:=
%                            CMAT.
        CLC(3);
        SLIT(3) =MLTRPS:EXCHL(3) $ICR;
% NOW WE TRANSPOSE BASE. THE RESULT WILL BE IN ADRC.
        CLC(3);
        SLIT(3) =TRPS;EXCHL(3) $ICR;
% WE FORM BASE(TR) X BASE. RESULT IS IN ADRD.
        LDI(3)     .ADRD;
        STI(3)     .ADRES;
        LDI(3)     .ADRC;
        STI(3)     .ADRES1;
        CLC(3);
        SLIT(3) =MLTRPS:EXCHL(3) $ICR;
% NOW THE COMPUTATION OF CMAT=CMAT-ADRC
        LIT(0)     =0.1,0;%    SET UP THE LOOP FOR I
        CADD(0)    $D3;
        CROTL(0)   24;
COMP:   LDI(1)     .ADRCM;
        CADD(1)    $C0;%
        LDI(2)     .ADRC;
        CADD(2)    $C0;
        LDI(3)     .SPEC;
        LDEE1      $C3;
        LDA        0(1);%  $A:= ROW I OF CMAT
        SBRN       0(2);%  SUBTRACT ROW I OF ANMAT
```

```
         STA        0(1);%  STORE  RESULT IN ROW I OF CMAT
         TXFEM(0)   ,COMP;
%        LIT(3)     =1,MESS2-1,MESS1;
%        DISPLAY    $C3,32;
%        LDL(3)     .ADRCM;CSHL(3) 6;WRTPEM;
%%%%%%%%%%%%%CMAT:=BASE X BASE(TR) - BASE(TR) X BASE %%%%%%%%%%
% LES FIND THE LARGEST OFF DIAGONAL ELEMENT
         CLC(3);
         SLIT(3) =FNDMX;EXCHL(3) $ICR;
%  .G WILL CONTAIN THE LARGEST OFF DIAGONAL ELEMENT
%  .MAXR & .MAXC WILL CONTAIN THE ROW- & COL.-INDEX
         ONESE(0)   ,+1;%         THE MAX. OFF-DIAGONAL ELEM.
         JUMP       COMP11;
%                                 WAS ZERO.
         CLC(3);
         SLIT(3) =AVER;EXCHL(3) $ICR;
         EQLXE(2)   $D0,COMP2A;
         JUMP       COMP2;
%%%%%%%%%%%%CHECK FOR C[2I-1,2I-1]=C[2I,2I]%%%%%%%%%%%%%
COMP2A:: CLC(3);
         SLIT(3) =NULCHK;EXCHL(3) $ICR;
         LDL(0)     .INDEX;
         ZERE(0)    ,+1;
         JUMP       COMP2;
         LDL(0)     .MAX;
         EQLXE(0)   $D1,+1;
         JUMP       COMP3;
         LDL(3)     .ADRA;
         STL(3)     .ADRES;
         CLC(3);
         SLIT(3) =SHFT;EXCHL(3) $ICR;
%%%%%%%%%% WE JUST EXCHANGED ROW 1 & ROW I OF BASE %%%%%%%%%
%%%%%%%%% NOW WE DO THE SAME TO EBEIG%%%%%%%%%%%%%%%%%%%%%%%%%%%
         LDL(3)     .ADRE;
         STL(3)     .ADRES;
         CLC(3);
         SLIT(3) =SHFT;EXCHL(3) $ICR;
         JUMP       COMP3;
COMP2::  LDL(0)     .MAXR;
         EQLXE(0)   $D0,COMPA;
         LDL(0)     .MAXC;
         EQLXE(0)   $D1,COMPA;
         JUMP       COMP3;
%%%%%%% WE BRING THE LARGEST OFF-DIAGONAL ELEMENT     %%%%%%%%%
%%%%%%%% INTO THE 2I,2I-1 POSITION.%%%%%%%%%%%%%%%%%%%%%%%%%%%%
COMPA:   LDL(3)     .ADRA;
         STL(3)     .ADRES;%      SHUFFLE BASE
         CLC(3);
         SLIT(3) =SHUF;EXCHL(3) $ICR;
%        LIT(3)     =1,MESS3-1,MESS2;
%        DISPLAY    $C3,32;
%        LDL(3)     .ADRA;CSHL(3) 6;WRTPEM;
         LDL(3)     .ADRCM;
         STL(3)     .ADRES:%      SHUFFLE CMAT
         CLC(3);
         SLIT(3) =SHUF;EXCHL(3) $ICR;
         LDL(3)     .ADRE;
         STL(3)     .ADRES;%       SHUFFLE EBEIG
         CLC(3);
         SLIT(3) =SHUF;EXCHL(3) $ICR;
%%%%%%%%%%% NOW WE ARE READY (OR NOT) FOR THE %%%%%%%%%%%%
```

```
%##########      TRANSFORMATION        ##################
COMP3:;  CLC(3);
         SLIT(3) =ANGL;EXCHL(3) $ICR;
%########## DO THE TRANSFORMATION BASE X ANMAT #############
         LDI(3)     .ADRA;
         STI(3)     .ADRES;
         STI(3)     .ADRES2;
         LDI(3)     .ADRC;
         STI(3)     .ADRES1;
          LIT(3)     =1;
         STI(3)     .SAV9;
          CLC(3);
         SLIT(3) =MULTPL;EXCHL(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;
         LIT(3)     =1;
         STI(3)    .SAV9;
         SLIT(3) =MULSA;EXCHL(3) $ICR;
%############ SAVE THE TRANSFORMATION MATRIX FOR ###########
%############ THE EIGENVECTOR COMPUTATION UNDER JACOBI #####
         LDI(3)    .ADRE;
         STI(3)    .ADRES;
         STI(3)    .ADRES2;
         LDI(3)    .ADRC;
         STI(3)    .ADRES1;
         LIT(3)    =1;
         STI(3)    .SAV9;
         SLIT(3) =MULTPL;EXCHL(3) $ICR;
%########### NOW WE FIND THE SECOND TRANSFOR- ############
%#########  MATION MATRIX CORRESP. TO O[I]  ##################
         CLC(3);
         SLIT(3) =HYANG;EXCHL(3) $ICR;
         LDI(0)    .MAX;
         ZERT(0)   .COMP5;
         LDI(3)    .ADRA;%   BASE X ANMAT
         STI(3)    .ADRES;
         STI(3)    .ADRES2;
         LDI(3)    .ADRC;
         STI(3)    .ADRES1;
         LIT(3)    =1;
         STI(3)    .SAV9;
         CLC(3);
         SLIT(3) =MULTPL;EXCHL(3) $ICR;
         CLC(3);
         SLIT(3) =TRASPOS;EXCHL(3) $ICR;
         LDI(3)     .ADRA;
         STI(3)    .ADRES;
         LDI(3)     .ADRC;
         STI(3)    .ADRES1;
         LDI(3)     .ADRD;
         STI(3)    .ADRES2;
         LIT(3)    =0;
         STI(3)    .SAV9;
```

```
        SLIT(3) =MULSA;FXCHL(3) $ICR;
%       LIT(3)     =1,ADRSAV-1,MESS3;
%       DISPLAY    $C3,32;
%       LDL(3)     .ADRA;CSHL(3) 6;WRTPEM;
        LDL(3)    .ADRE;%  EREIG X ANMAT(TR.)
        STL(3)    .ADRES;
        STL(3)    .ADRES2;
        LDL(3)    .ADRC;
        STL(3)    .ADRES1;
        LIT(3)    =1;
        STL(3)    .SAV9;
        SLIT(3) =MULTPL;FXCHL(3) $ICR;
COMP5:  LDL(0)    .FINVAL;
        LIT(1)    =0.000001;
        SETE      E.OR.-E;
        SETE1     E.AND.E;
        LDA       $C0;
        TAL       $C1;
        SETC(0)   I;
        CLRA;
        ONEST(0)   ,+1;
        JUMP       COMP6;
        LDL(1)    .FINVAL1;
        LDA        $C1;
        SPRN      $C0;
        SAP;
        LIT(1)    =0.0000000001;
        TAL       $C1;
        SETC(1)   I;
        ONEST(1) ,COMP9;
        LDC(1)    $A;
        STL(0)    .FINVAL1;
        LDL(0)    .AB;
        SBRN      $C0;
        SAP;
        LIT(0)    =0.00000001;
        TAL       $C0;
        SETC(0)   I;
        ONESE(0)   ,+1;
        JUMP       COMP9;
        STL(1)     .AB;
        JUMP       COMP6;
COMP9:;  CLC(3);
%#########  MAKE THE RESULTANT MATRIX SYMMETRIC AND   #########
%#########  KEEP A TAG IN CASE OF ASYMMETRY   ###############
        SLIT(3) =SYASY;FXCHL(3) $ICR;
%       LDL(3)     .ADRA;CSHL(3) 6;WRTPEM;
%       LDL(3)     .ADRE;CSHL(3) 6;WRTPEM;
COMP11:; CLC(3);
        SLIT(3)    =ADRSAV;
        RIN(3)     $D32;
        CLC(3);
        SLIT(3)    =ADRSAV+8;
        LOAD(3)    $C0;
        ALIT(3)    =1;
        LOAD(3)    $C1;
        LDL(2)     .ANTUN1;% TAKE THE PATTERN OF ANTUN1 TO THE
% OUTSIDE TO CHECK FOR THE NEXT PATH OF ACTION
% CONCERNING THE CALCULATION OF EIGENVALUES. I.E., IF
% ANTUN1 CONTAINS AT LEAST ONE ONE, THEN THE EIGEN-
% VALUE WILL BE A COMPLEX ONE RESULTING FROM THE JACOBI
```

```
* SUBROUTINE. SO WE TAKE THE PATH TO THE MODIFIED JACOBI
* WHICH CAN HANDLE THE CASE OF THE COMPLEX EIGENVALUE.
        LDL(3)      .RETUR;%   TURN BACK TO THE OUTSIDE
        EXCHL(3)    $ICR;
%#####################################################
%#                                                   #
%#                                                   #
%#                 END EBER/EBERL                    #
%#                                                   #
%#                                                   #
%#####################################################
END       EBERL.

COLUMN 10,20,30
```

APPENDIX D

Results from EIGEN

a) <u>The Original Matrix</u>

```
********** DISPLAY #        1     ICR=  002443 1:16  TIME=09:27:44:00    ELAPSED PROCESSOR TIME=00:00:05:11*************************

MEMORY:
------
LOCATION:8  C(LOCATION)                  C(LOCATION +0000 01:8)           C(LOCATION +0000 02:8)           C(LOCATION +0000 03:8)
0200 00:+.39999970707000050+0001      -.30587921358500060-0003        -.24079223557299940-0002        +.32893877723800090-0004

********** DISPLAY #        2     ICR=  002443 1:16  TIME=09:27:44:58    ELAPSED PROCESSOR TIME=00:00:05:24*************************

MEMORY:
------
LOCATION:8  C(LOCATION)                  C(LOCATION +0000 01:8)           C(LOCATION +0000 02:8)           C(LOCATION +0000 03:8)
0201 00:-.30587921358500060-0003      +.10000000408800030+0001        -.39456574815699920-0005        -.59456996816000020-0004

********** DISPLAY #        3     ICR=  002443 1:16  TIME=09:27:45:06    ELAPSED PROCESSOR TIME=00:00:05:31*************************

MEMORY:
------
LOCATION:8  C(LOCATION)                  C(LOCATION +0000 01:8)           C(LOCATION +0000 02:8)           C(LOCATION +0000 03:8)
0202 00:-.24079223557299940-0002      -.39456574815699920-0005        +.20000029208899970+0001        -.17212239523799970-0003

********** DISPLAY #        4     ICR=  002443 1:16  TIME=09:27:45:15    ELAPSED PROCESSOR TIME=00:00:05:38*************************

MEMORY:
------
LOCATION:8  C(LOCATION)                  C(LOCATION +0000 01:8)           C(LOCATION +0000 02:8)           C(LOCATION +0000 03:8)
0203 00:+.32893877723800090-0004      -.59456996816000020-0004        -.17212239523799970-0003        +.29999999730400050+0001
```

b) The Diagonalized Matrix

```
$$$$$$$$$$ DISPLAY #       12     ICR=  002454 1:16   TIME=09:44:30:27     ELAPSED PROCESSOR TIME=00:07:08:07$$$$$$$$$$$$$$$$$$$$$$$$

MEMORY:
------
LOCATION:8  C(LOCATION)              C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)
0200 00:+.400000000203181 1@+0001    +.23050072267549740@-0010     -.16087059707029200@-0010     +.61362492387324240@-0008

$$$$$$$$$$ DISPLAY #       13     ICR=  002454 1:16   TIME=09:44:31:05     ELAPSED PROCESSOR TIME=00:07:08:25$$$$$$$$$$$$$$$$$$$$$$$$

MEMORY:
------
LOCATION:8  C(LOCATION)              C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)
0201 00:+.23050072267549740@-0010    +.30000000033237630@+0001     +.41203375959377240@-0008     +.26039539332830020@-0011

$$$$$$$$$$ DISPLAY #       14     ICR=  002454 1:16   TIME=09:44:31:12     ELAPSED PROCESSOR TIME=00:07:08:32$$$$$$$$$$$$$$$$$$$$$$$$

MEMORY:
------
LOCATION:8  C(LOCATION)              C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)
0202 00:-.16087057972373140@-0010    +.41203375957208310@-0008     +.10000000079076440@+0001     -.26993808170715570@-0009

$$$$$$$$$$ DISPLAY #       15     ICR=  002454 1:16   TIME=09:44:31:19     ELAPSED PROCESSOR TIME=00:07:08:39$$$$$$$$$$$$$$$$$$$$$$$$

MEMORY:
------
LOCATION:8  C(LOCATION)              C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)


0203 00:+.61362492248545830@-0008    +.26039556684669200@-0011     -.26993808165294560@-0009     +.19999999922470120@+0001
```

c) The Eigenvector Matrix

```
********** DISPLAY #         8     ICR=  00244D 0:16  TIME=09:44:12:55    ELAPSED PROCESSOR TIME=00:07:05:12*************************
MEMORY:
------
LOCATION:8   C(LOCATION)
0205 00:+.9999992694847109E+0000          C(LOCATION +0000 01:8)             C(LOCATION +0000 02:8)            C(LOCATION +0000 03:8)
                                       -.3331730884807748E-0004        +.1019628351699596E-0003       +.1203961622744242E-0002

********** DISPLAY #         9     ICR=  00244D 0:16  TIME=09:44:15:48    ELAPSED PROCESSOR TIME=00:07:05:31*************************
MEMORY:
------
LOCATION:8   C(LOCATION)
0206 00:+.3310712060560698E-0004          C(LOCATION +0000 01:8)             C(LOCATION +0000 02:8)            C(LOCATION +0000 03:8)
                                       +.9999999842038818E+0000        +.2972924040163346E-0004       +.1720824139929307E-0003

********** DISPLAY #        10     ICR=  00244D 0:16  TIME=09:44:15:56    ELAPSED PROCESSOR TIME=00:07:05:38*************************
MEMORY:
------
LOCATION:8   C(LOCATION)
0207 00:-.1019587319959067E-0003          C(LOCATION +0000 01:8)             C(LOCATION +0000 02:8)            C(LOCATION +0000 03:8)
                                       -.2972512371758357E-0004        +.9999999943510716E+0000       -.4323882547824215E-0005

********** DISPLAY #        11     ICR=  00244D 0:16  TIME=09:44:16:03    ELAPSED PROCESSOR TIME=00:07:05:45*************************
MEMORY:
------
LOCATION:8   C(LOCATION)
0210 00:-.1203967771670256E-0002          C(LOCATION +0000 01:8)             C(LOCATION +0000 02:8)            C(LOCATION +0000 03:8)
                                       -.1720425542802309E-0003        +.4196010360106502E-0005       +.9999992604224204E+0000
```

DOCUMENT CONTROL DATA - R & 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 2b. GROUP

3. REPORT TITLE

ILLIAC IV CODES FOR JACOBI AND JACOBI-LIKE ALGORITHMS

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	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
November 5, 1971	35	4

8a. CONTRACT OR GRANT NO. DAHC04 72-C-0001 b. PROJECT NO. ARPA Order 1899 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) CAC Document No. 19 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

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						