SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

450001

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

SHARE PROGRAM LIBRARY SUBMITTAL FORM

SHARE PROGRAM LIBRARY AGENCY
Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park, North Carolina
27709 USA

SPLA CONTROL NUMBER: SHR-174

This form should be completed and submitted with the program package to the SHARE Program Library Agency at the address shown above. Standards and instructions for submitting programs are in the "SHARE Reference Manual".

(1) Program Number (to be filled in by SPLA)..... 360D-45.0.001

(2) System Type (machine)..... S/360, S/370

(3) Search Key..... Mathematical, Linear Algebra, and Utility routines for PL/I. The PL/I Subprocedure Collection

(4) Programming Systems/Languages..... PL/I

(5) Author's Name and Address.....

MR. HANK HAMILTON
TYMSHARE TRANSACTION SERVICES
900 FRONT STREET
SAN FRANCISCO, CA. 94111

(6) Direct Technical Inquiries to Name & Address (if different than Author)

(7) Title of Program..... PL/I Subprocedure Collection - Release 1

(8) Submitter's Installation Membership Code..... NCS

(9) Submitter's Own Program Identification and Suffix(Optional)..

(10) Primary Subject Code..... 45.0

(11) Minimum System Requirements PL/I Compiler

(12) New or Revision Code (if revision, show prior Program Number in Item 1)

(13) Year Completed..... 1974

(14) Date of Submittal..... 14 February 1975

(15) Documentation (number of original pages submitted)..... 175

(16) Abstract (should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.

Revised 4/74

SHARE PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide:

a. Purpose
b. Programming Language used
c. Version and modification level or release number
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements

ABSTRACT

The Subprocedure Collection (SPC) is a library of subprocedures written in PL/I for use by PL/I programs. This release of the SPC contains about 170 procedures mostly in the area of mathematics; mostly, linear algebra.

Some of the procedures in this collection represent original algorithms and original code. Some were taken from the algorithms section of "Communications of the ACM". Some are adopted from the IBM Scientific Subroutine Package (PL/I). Currently, the majority are implementations of algorithms from the Handbook for Automatic Computation, Vol 2, Linear Algebra by Wilkinson and Reinsch (Springer-Verlag, 1971). This latter group are similar to the subroutines comprising the EISPAC FORTRAN code distributed by Argonne Labs.

All procedures were developed in an Optimizer/Checkout Compiler environment No deliberate steps were taken to be compatible with PL/I(F), but nothing deliberate was done to not be.

(Please attach additional pages if necessary).....Total pages attached

Permission to Publish

"I hereby give the SHARE Program Library Agency permission to reprint, reproduce, and distribute this program."

(17) Signature of Submitter and Date H.R. Hamilton 12 Feb 1975

(18) Signature of Installation Addressee H.R. Hamilton

NORTH CAROLINA STATE UNIVERSITY | AT RALEIGH

COMPUTING CENTER
P. O. BOX 5445
RALEIGH, N. C. 27607

The PL/I Subprocedure Collection

H.R. Hamilton
12 Feb 1975

distributed by
The SHARE Program Library Agency

The PL/I Subprocedure Collection

The Subprocedure Collection (SPC) is a library of subprocedures written in PL/I for use by PL/I programs. This release of the SPC contains about 170 procedures mostly in the area of mathematics; mostly, linear algebra.

Some of the procedures in this collection represent original algorithms and original code. Some were taken from the algorithms section of "Communications of the ACM". Some are adopted from the IBM Scientific Subroutine Package (PL/I). Currently, the majority are implementations of algorithms from the Handbook for Automatic Computation, Vol 2, Linear Algebra by Wilkinson and Reinsch (Springer-Verlag, 1971). This latter group are similar to the subroutines comprising the EISPAC FORTRAN code distributed by Argonne Labs.

The documentation consists of the user program writeups as originally published by the Triangle Universities Computation Center where this package was orginially implemented. The task of rearranging, editing, and republishing this documentation is currently beyond our abilities.

The Distribution Tape

The distribution consists of the source code of all procedures in the collecti on, in a single sequential data set which can be used as input to IEBUPDTE to create a PDS with each procedure as a member.

The DCB parameters of the tape are

```
RECFM = FB
LRECL = 80
BLKSIZE = 6400
```

and LABEL = (1,NL).

The density and track coding will be as specified when the tape was ordered.

Each procedure is preceded by a control statement of the form (Δ is "blank"):

```
./ΔΔΔΔΔΔΔADDΔΔΔNEW=PD,NAME=proc-name
```

and the last statement in the data set is

```
./ΔΔΔΔΔΔΔENDUP
```

There will also be a few statements to assign aliases:

```
./ΔΔΔΔΔΔΔALIASΔNAME=proc-alias-name
```

The data set contains approximately 15,220 80 character records.

TELECOMPUTING

TUCC MEMORANDUM

March 31, 1971

Library Services Series
Document No. LS-14-1

ROUTING
Board of Directors
Computation Centers for Distribution
TUCC/NCECS Staff

FROM: NCSU Programming Services

SUBJECT: CLOCK (CCLOCK,RCLOCK), a Subroutine for Timing Computer Runs

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus User/Programming Services

EFFECTIVE DATE: April 12, 1971. *** There have been major changes in this routine since the last issue of this memo. The document should be reviewed in its entirety.

FUNCTION: CLOCK was designed to provide a calling program with various types of timing information, principally: elapsed time between two points in a program, and the time remaining until system time cut-off. Three types of time can be read: "CPU-time"(CCLOCK), "real-time"(RCLOCK), and "TUCC-time" (CLOCK) from the TUCC accounting clock.

USE: Each of the routines can be called with one to three arguments in the following way (where "sub" may be CLOCK, RCLOCK, or CCLOCK):

CALL sub (I)

where I is a full-word binary integer scalar variable (INTEGER*4 or FIXED BINARY(31,0)), which will be assigned the current reading of the appropriate clock. The value is an integer in units of hundredths of a second. If CLOCK is called, this value will be the time remaining until system time cut-off. If RCLOCK is called, the value will be the current value of the 360/75 real time (time of day) clock. If CCLOCK is called, this value will be the time remaining in a CPU-timing interval initialized by CCLOCK.

CALL sub (I,J)

where J is a full-word binary integer scalar variable (INTEGER*4 or

Exceptions and Misc Info

1. The IBM SSP routines which were adopted were converted to the 60-character set and many were rewritten into a GOTO-less form.

2. A few of the procedures are written in S/360 ALC:

CALDATE	CLOCK	SHRU
JULDATE	SVC	TPUT

3. Procedure SOLVE, contained in the PDS, does not have documentation in this package. It is used by procedure LEAION and should be linked with it or included in it as an internal procedure. For a description, see procedure "least squares solution" in the Handbook by Wilkinson. (Similarly: DSOLVE, used by DLEAIO

4. For those who will want to use the Handbook in conjunction with these procedures, we used the text's procedure name wherever possible. Where necessary, we truncated the names to 6 characters by taking the first three and last three letters of the texts names. This name was preceded by a letter "D" for a "double precision" version.

5. The procedures from the Handbook were tested with the author's own test cases supplemented by problems from A Collection of Matrices for Testing Computational Algorithms by Gregory & Karney (Wiley), and some from CACM. A "double-precision" version of a procedure was produced in only one of two circumstances:
   - double-precision was necessary to pass the tests
   - the procedure had to be used in conjunction with others which process double-precision data.

   Here "a double-precision version" means a version which processes user data recorded in double-precision. Many of the "single-precision" procedures utilize double-precision internally to retain significance.

6. Procedures adopted from the IBM SSP (PL/I) are not documented in this package. The documentation in the IBM manual is still appropriate.

7. All procedures were developed in an Optimizer/Checkout Compiler environment. No deliberate steps were taken to be compatible with PL/I(F), but nothing deliberate was done to not be.

FIXED BINARY (31,0)), and I is exactly as described above. In this form of the CALL, I is assumed to have been initialized by an earlier CALL sub (I) for the same "sub". J will be given a value equal to the time elapsed since the earlier reading in the same units as I, and I will return the current reading of the clock.

CALL sub (I,J,A)

where A is a one-word floating-point scalar variable (REAL*4, FLOAT DECIMAL (6)) and I and J are exactly as described above. This form of CALL has the same function as CALL sub (I,J) except that, in addition the elapsed time is returned as the value of A and in units of seconds.

Many different "clocks" can be running in the same program as illustrated by:

```
   /*GET OVERALL CPU TIME, USING 'CLOCK' IT */
A:CALL CCLOCK (IT);
      ...
      ...
   /*GET DO LOOP CPU TIME, USING 'CLOCK' JT */
     CALL CCLOCK (JT);
     DO I=1 TO N;
      ...
      ...
     END;
     CALL CCLOCK (JT, JTE);
      ...
      ...
B:CALL CCLOCK (IT, ITE);
/* JTE WILL HAVE CPU TIME FOR LOOP */
/* ITE WILL HAVE CPU TIME FOR ALL CODE BETWEEN A AND B */
```

The routine can be called by PL/I or FORTRAN programs or programs with FORTRAN-like calling sequences. The routine is not re-entrant.

***When reading the "TUCC-time" clock, the user should remember that this clock may be updated in relatively large "time" increments due to accounting of EXCPs. The usual increment is one second. The CPU and real-time clocks are updated approximately every 16.7 milli-seconds.

STORAGE: This routine is one CSECT with 3 entry points. The storage requirement for the routine is 274_{10} bytes.

TELECOMPUTING

TUCC MEMORANDUM

July 7, 1970

Library Services Series
Memorandum No. LS-90

ROUTING
Board of Directors
Computation Centers for Distribution
TUCC/NCECS Staff

FROM: NCSU Programming Services

SUBJECT: A Uniform Random Number Generator for PL/I and FORTRAN

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus User/Programming Services

Purpose

A sometimes undesirable characteristic of congruential pseudo-random number generators is a type of predictability in the numbers. This phenomonen is described by Martin Greenberger (CACM, Vol 8 No 3, March 1965) and would affect the results of calculations based on the numbers which are dependent upon uniform density of points in 2 space, or planes in 3-space, etc.

This relationship can be hidden or destroyed by "shuffling" the random sequence produced by the generator. Essentially, one uses one congruential generator, G1, to generate a sequence of random numbers and a second, G2, to select numbers "at random" from that sequence.

SHRU is a shuffling generator of this type.

The sequence generator (G1) is of the form r=r*65533 where r_0 is supplied by the user. The shuffling generator (G2) is of the form u=u*32771 with u_0 = 314159265.

The period of this generator will be at least 2^{29}.

USE

This generator is available to both FORTRAN and PL/I programmers.

Two CALLs are necessary to use the generator:

CALL SHRUØ(I)
will initialize the generator with the first 128 numbers from the G1 sequence. This CALL is performed only once for each sequence to be generated, and will destroy any earlier sequence generated.

The value of I is the initial value of r("seed" or r_0) in the G1 generator and should be a nine significant digit odd number. The value of I is undisturbed by the CALL.

CALL SHRU(Y)
will return to Y a pseudo-random floating-point short-precision number scaled to the range Ø<Y<1. The G2 generator is used to select a number from the table of 128 G1 numbers. G1 is then used to replace the number selected. The minimum number actually produced is 392ØØØØØ (hex.float) or about .4656613E-9 (decimal). The maximum number is 4ØFFFFFF (hex float) or about .9999999EØ (decimal).

In FORTRAN, I should be INTEGER*4 and Y, REAL*4.

In PL/I, I should be BIN FIXED (31,Ø) and Y, BIN FLOAT (24,Ø) or DECIMAL FLOAT (6,Ø). I and Y must be arithmetic elements and I may be a constant.

SHRU requires about 20 microsec to obtain one number on a S/360/75I and occupies 700_{10} bytes of storage.

In the TUCC system it resides in SYS1.SUBLIB and SYS1.PL1SUBS as member SHRU. SHRUØ is a secondary entry into that member.

TELECOMPUTING

TUCC MEMORANDUM

February 2, 1973

Library Services Series
Document No. LS-195-0

FROM: NCSU Programming Services

SUBJECT: COMDET, for Factorization and Determinant of a Complex Unsymmetric Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

A PL/I subprocedure which performs the factorization A=LU, where A is a complex unsymmetric matrix, L is a lower triangular matrix, and U is a unit upper triangular matrix (diagonal =1,1,1,...,1). L is overwritten on the lower triangle and diagonal elements of A. U is overwritten on the upper triangle of A, omitting the constant diagonal.

A record of any interchanges made to the rows of A is kept in the vector INT, such that the I-th row and the INT(I)-th row were interchanged at the i-th step.

Two numbers, D1 and D2, are computed, which can be used to determine the value of the determinant:

$$\det A = D1 * 2^{**}D2 \left(\frac{1}{16} \le |D1| < 1\right).$$

The procedure will fail if A, or A modified by computational rounding errors, is singular or almost singular.

This procedure is based on the procedure COMPDET described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; ~~Wilinson~~ Wilkinson, ~~Reinch~~ Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) COMDET(N,AA,D1,D2,INT,FAIL)
where

N	BIN FIXED(31)	gives the order of the matrix A.
AA	(*,*)BIN FLOAT(21)COMPLEX	on call, contains the matrix, A, to be factored, in its first N rows and N columns; on return, contains L and U as described above.

D1	BIN FLOAT(21)	COMPLEX described above.
D2	BIN FIXED(31)	described above.
INT	(*)BIN FIXED(15)	as described above; LBOUND(INT,1) should be same as LBOUND(AA,1).
FAIL	LABEL	a label in the user program, to which the procedure will transfer, if the process fails.

2) DCOMDET(N,AA,D1,D2,INT,FAIL)
where parameters have the same meaning as with COMDET except for:

AA	(*,*)BIN FLOAT(53)	COMPLEX
D1	BIN FLOAT(53)	COMPLEX

[redacted]

FROM: NCSU Programming Services

SUBJECT: SVC, a Subprocedure for Making Supervisor Calls in PL/I (Optimizer Only)

SUPPORT TYPE: A

DIRECT INQUIRIES TO: NCSU Programming Services

DISTRIBUTION: Full-Time Programming Services & Judy Hallman, UNC
Neal Paris, Duke

SVC is a subprogram callable from PL/I programs that are compiled by the Optimizing Compiler. The user specifies a number corresponding to a specific system SVC routine and supplies the necessary values for that SVC routine. On return, the values normally returned in registers 1, 0 and 15 by the SVC routine are returned to the user.

If a system abend occurs during execution of subprocedure SVC, the PL/I ERROR condition is raised.

SVC routines are discussed in several IBM manuals depending on the application of the SVC:

GC28-6550 IBM S/360 OS: Data Management for System Programmers

GC28-6646 IBM S/360 OS: Supervisor Services and Macro Instructions

GC26-3794 OS Data Management Macro Instructions

GC28-6670 IBM S/360 OS: Programmer's Guide to Debugging

In addition there are some locally created SVC's which are known to various system programmers. Some of these may be described elsewhere in this document series.

Normally, the person using procedure SVC should be familiar with assembler language and SVC routines.

```
DCL  SVC ENTRY(/* SVC# */BIN FIXED(15), /* R1 */ BIN FIXED(31),
         /* RØ  */BIN FIXED(31), /* R15 */BIN FIXED(31));
```

where

SVC# — On entry, contains the decimal number of the SVC routine to be executed.

R1 — On entry, contains the value needed in register 1 by the SVC routine indicated by SVC#. On return, contains the value returned in register 1 by the SVC routine indicated by SVC#.

RØ	On entry, contains the value needed in register Ø by the SVC routine indicated by SVC#. On return, contains the value returned in register Ø by the SVC routine indicated by SVC#.
R15	On entry, contains the value needed in register 15 by the SVC routine indicated by SVC#. On return, contains the value returned in register 15 by the SVC routine indicated by SVC#.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October 11, 1974

Library Services Series
Document No. LS-272-0

FROM: NCSU Programming Services

SUBJECT: DIRHES, for Reduction of a Real Square Matrix to Upper-Hessenberg Form.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which will reduce a specified submatrix of a real, square matrix to upper-Hessenberg form by elementary similiarity transformations.

This procedure will be more effective if the matrix to be reduced is preconditioned by BALNCE (LS-267) or some similar procedure.

This procedure reduces rounding errors by accumulating inner-products wherever possible.

If eigenvectors of the derived matrix are later computed, the eigenvectors of the original matrix can be obtained with DIRBAK (LS-278).

This procedure is based on the procedure DIRHES described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) DIRHES (N,K,L,A,INT)
where

N	BIN FIXED(31,0)	is the order of the full matrix.
(K,L)	BIN FIXED(31,0)	the procedure reduces the submatrix of order L-K+1 which starts at the element A(K,K) and finishes at A(L,1). If the entire matrix is to be reduced: K=LBOUND(A,L),L=K+N-1. If BALNCE has been previously used, its output LOW and HI are used for K and L.
A (*,*)	BIN FLOAT(21)	on call, contains in its first N columns and N rows, the matrix to be reduced. On return, the derived Hessenberg matrix has replaced the specified submatrix. The zeros in the lower

portion of the Hessenberg matrix are replaced with multipliers used in the reduction. LBOUND(A,1) should equal LBOUND(A,2);

INT (*)BIN FIXED(15,0) on return, contains, in elements K through L, a description of the row and column interchanges involved in the reduction. This will be needed later by DIRBAK, if that is used. LBOUND(INT,1) should equal LBOUND(A,1).

2) DDIRHES (N,K,L,A,INT)
where the parameters have the same meaning as above, except for:
A (*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October 11, 1974

Library Services Series
Document No. LS-276-0

FROM: NCSU Programming Services

SUBJECT: HQR2, for Finding All Eigenvalues and Eigenvectors of a Real Upper Hessenberg Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes all eigenvalues and unnormalized vectors of a real upper Hessenberg matrix by QR trianguiarization.

If the Hessenberg matrix was arrived at by reduction of a general matrix, it is assumed that DIRHES (LS-272) and DIRANS (LS-274) or ORTHES (LS-273) and ORTANS (LS-271) or similar procedures performed the reduction. HQR2 finds the eigenvectors of the original matrix, not the Hessenberg matrix eigenvectors, unless the original matrix was in Hessenberg form. It is recommended, when starting with a general matrix, to balance the matrix with BALNCE (LS-267) prior to reduction. The vectors of the original unbalanced matrix can be obtained by BALBAK (LS-268).

This procedure is based on the procedure HQR2 in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) HQR2 (N,LOW,HI,MACHEPS,H,VECS,W,CNT,FAIL)
where

N	BIN FIXED(31,0)	contains the order of the matrix.
(LOW,HI)	BIN FIXED(31,0)	if the matrix has been balanced by BALNCE, these are the indices LOW and HI returned by that procedure. If not, set LOW=LBOUND(A,1) and HI=LOW+N-1.
MACHEPS	BIN FLOAT(21)	should be .953675E-6.
H	(*,*)BIN FLOAT(21)	contains, in its first N rows and N columns, the Hessenberg matrix as described above. The matrix is destroyed by HQR2. LBOUND (H,1) should equal LBOUND(H,2).
VECS	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and N columns, the matrix defining the similarity transformations used in reduction

to Hessenberg form (produced by DIRANS or ORTANS). If the original matrix was in Hessenberg form, this should be the order-N identity matrix. On return, the first N columns will contain the eigenvectors of the original matrix. If the i-th eigenvalue is real (IMAG(W(I))=0), the i-th column of VECS is the corresponding eigenvector. If eigenvalues i and i+1 are a complex pair, then column i and i+1 of VECS give the real and imaginary part of the eigenvector corresponding to the eigenvalue with positive imaginary part. LBOUND(VECS,1) should equal LBOUND(H,1) and LBOUND(VECS,2).

W (*)BIN FLOAT(21) COMPLEX on return contains, in its first N elements, the computed eigenvalues. LBOUND(W,1) should equal LBOUND(VECS,2).

CNT (*)BIN FIXED(15,0) on return, contains the number of iterations required for each eigenvalue. If two values are found simultaneously as a pair, then the number of iterations is given with a positive sign for the first and a negative sign for the second. LBOUND(CNT,1) should equal LBOUND(W,1).

FAIL LABEL is a label in the user program to which HQR2 will transfer when 30 iterations fail to isolate the current eigenvalue.

2) DHQR2 (N,LOW,HI,MACHEPS,H,VECS,W,CNT,FAIL)
where the parameters have the same meaning as with HQR2 except for:

MACHEPS BIN FLOAT(53) should be .2220446049250314E-15;

H (*,*)BIN FLOAT(53)

VECS (*,*)BIN FLOAT(53)

W (*)BIN FLOAT(53) COMPLEX

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October 11, 1974

Library Services Series
Document No. LS-277-0

FROM: NCSU Programming Services

SUBJECT: INVIT, for Finding Selected Eigenvectors of a Real Upper Hessenberg Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which uses inverse iteration to find selected eigenvectors of a real upper Hessenberg matrix, given the eigenvalues of the matrix. Only one complex vector, corresponding to the eigenvalue with positive imaginary part, is formed for a complex pair.

It is assumed the eigenvalues have been found by HQR (LS-275) or a similar procedure, so that they are ordered so as to have the correct affiliation.

The procedure is based on the procedure INVIT in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) INVIT (N,MACHEPS,C,A,W,EXCPT,Z)
where

N	BIN FIXED(31,0)	contains the order of the matrix.
MACHEPS	BIN FLOAT(21)	should be .953675E-7.
C	(*)BIT(1)	indicates the eigenvalues, in W below, for which the eigenvectors are required. If the vector corresponding to W(I) is desired, C(I)='1'B; otherwise, C(I)='0'B. If the eigenvectors for a complex pair are desired, only the C(I) corresponding to the eigenvalue with positive imaginary part needs to be '1'B. LBOUND(C,1) should equal LBOUND(W,1) and the first N elements of C are used.
A	(*,*)BIN FLOAT(21)	contains the upper Hessenberg matrix in its first N rows and N columns. LBOUND(A,1) should equal LBOUND(A,2).
W	(*)BIN FLOAT(21) COMPLEX	on call contains, in its first N elements, the eigenvalues of A. On return, the

real parts of the selected eigenvalues have been perturbed where necessary to separate close eigenvalues. LBOUND(W,1) should equal LBOUND(A,1).

EXCPT (*)BIT(1) on return, contains a '1'B in EXCPT(I) if eigenvector Z(*,I) meets the convergence criteria. If not, EXCPT(I) contains '0'B. In many of these cases, however, it has been found that the returned eigenvector is nearly as good of an approximation as the vectors that passed the convergence criteria. Residuals might be checked for these eigenvectors. LBOUND(EXCPT,1) should equal LBOUND(Z,2).

Z (*,*)BIN FLOAT(21) on return contains, in its first r columns, the computer eigenvectors, (r=number of eigenvectors specified in the C vector; note that the complex pair takes two columns.) If the eigenvector corresponding to the complex eigenvalue with positive imaginary part is desired, and is the Jth eigenvalue for which an eigenvector is desired, the eigenvector will occupy columns, J and J+1 of Z. LBOUND(Z,1) should equal LBOUND(A,1).

2) DINVIT (N,MACHEPS,C,A,W,EXCPT,Z)
where the parameters have the same meaning as with INVIT except for:

MACHEPS BIN FLOAT(53) should be .222044604925314E-15

A (*,*)BIN FLOAT(53)

W (*)BIN FLOAT(53) COMPLEX

Z (*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October 11, 1974

Library Services Series
Document No. LS-278-0

FROM: NCSU Programming Services

SUBJECT: DIRBAK, for Obtaining the Original Eigenvectors from Those of a Derived Hessenberg Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

After finding the eigenvectors of a Hessenberg matrix, of order n, which was obtained by applying DIRHES (LS-272) to a general real matrix, this subprocedure can be used to obtain the vectors of the original matrix.

This procedure is based on the procedure DIRBAK described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) DIRBAK (LOW,HI,R,A,INT,Z)
where

(LOW,HI)	BIN FIXED(31,0)	contains the indices LOW and HI returned by BALNCE (LS-267) if that was used to prepare the matrix for reduction to Hessenberg form. Otherwise LOW=LBOUND(A,1) and HI=LOW+n-1 where n is the order of the matrix specified for DIRHES.
R	BIN FIXED(31,0)	contains the number of vectors of the Hessenberg matrix which have been found.
A (*,*)	BIN FLOAT(21)	contains the matrix returned by DIRHES.
INT (*)	BIN FLOAT(15,0)	contains the array describing the interchanges used in DIRHES.
Z (*,*)	BIN FLOAT(21)	on call contains, in its first R columns, R eigenvectors of the derived Hessenberg matrix. The elements of the eigenvectors occupy the first n elements of each column. On return, these vectors will be replaced by the vectors of the original matrix. These vectors are not normalized. Note that the storage correspondence of eigenvectors to eigenvalues depends on the choice of subprocedure for finding the vectors. LBOUND(Z,1) should equal LBOUND(A,1).

2) DDIRBAK (LOW,HI,R,A,INT,Z).
where the parameters have the same meaning as above, except for:

A	(*,*)BIN FLOAT(53)
Z	(*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October 11, 1974

Library Services Series
Document No. LS-279-0

FROM: NCSU Programming Services

SUBJECT: COMEIG, for Finding the Eigenvalues and Eigenvectors of a Complex Matrix.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which uses a norm reducing Jacobi-type method to find all eigenvalues and eigenvectors of a general square complex matrix.

This procedure is generally recommended over EIGEN (LS-211) even for real matrices, except when all roots are real. COMEIG does not fail, as does EIGEN, in the complex defective case, but convergence will be linear. If the matrix is normal (AA*=A*A), other methods are faster.

The procedure is based on the procedure COMEIG in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) COMEIG (N,A,T,TIMES,FAIL)
where

N	BIN FIXED(31,0)	contains the order of the matrix
A	(*,*)BIN FLOAT(21) COMPLEX	on call contains, in its first N rows and N columns, the matrix to be solved. On return, the eigenvalues appear on the diagonal of A, other elements being destroyed. LBOUND(A,1) should equal LBOUND(A,2).
T	(*,*)BIN FLOAT(21) COMPLEX	on return contains in its first N columns the eigenvectors whose elements occupy the first N elements of each column. LBOUND(T,1) should equal LBOUND(A,1) and LBOUND(T,2).
TIMES	BIT(1)	should be '1'B on the first entry into COMEIG. On subsequent calls for the problem, should be '0'B.
FAIL	LABEL	is a label in the user program to which the procedure will transfer if convergence is not achieved in 35 iterations. The procedure may be re-entered with the parameters that have just been found (see TIMES), but convergence is again not guaranteed.

2) DCOMEIG (N,A,T,TIMES,FAIL)
where the parameters have the same meaning as above, except for:

A	(*,*)BIN FLOAT(53) COMPLEX
T	(*,*)BIN FLOAT(53) COMPLEX

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

June 22, 1973

Library Services Series
Document No. LS-220-0

FROM: NCSU Programming Services

SUBJECT: TPUT - to Allow PL/I Programs under TSO to Send Special Characters to a Terminal

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

AUTHOR: Mac Frye, NCSU Computing Center

TPUT is a subprocedure enabling a PL/I program, executing in a TSO environment, to send a string of characters to the terminal without having any characters added or deleted. The subprocedure requires only one argument: a character string to be sent to the terminal.

The character string is translated according to TUCC TSO standards (each character, except for certain control characters and the "printable" characters, is translated to a colon). Each terminal type has certain defined control characters and printable characters. If in doubt about these, contact your computing center. This translation can be bypassed by giving, as the first character of the string, a "bypass" character (hexadecimal '24'). In this case, the bypass character is changed to a hex 'DF', and the string is sent to the terminal. Any DF characters found in the string during this transmission are physically deleted from the string and do not arrive at the terminal.

TPUT is invoked by a CALL such as:

```
CALL TPUT(STRING);
```

where:

STRING CHAR(*) VARYING is the character string of length 0 through 32767 to be sent to the terminal, including, possibly, the bypass character.

In the event that TPUT detects an error, the TPTFAIL condition is signaled. The user may trap this error with an ON CONDITION (TPTFAIL) statement. Whenever the TPTFAIL condition is signaled by TPUT, the identifier ERROR is assigned an error code which gives more information about the cause of the error.

The values and meanings of the error codes are:

- '1' TPUT has been called outside the TSO environment. No output is produced.
- '2' An attention interrupt occurred during execution of the TPUT SVC used by the procedure.
- '3' Invalid parameters were detected by the TPUT SVC used by the procedure, probably caused by an incorrect type argument being passed to TPUT.
- '4' Internal error in TPUT procedure.
- '5' Undefined error. Specifically, the TPUT SVC used by the procedure has returned an error code which was not defined in the -1 edition of the manual referenced below.

The user can access the identifier ERROR by including in his program:

```
DCL  ERROR CHAR(4) STATIC EXTERNAL;
```

Error codes 3 and 4 "should never appear", and 5 is unlikely. If they appear, the user should report the occurrence.

Normal return from the ON CONDITION(TPTFAIL) on-unit, in the case of codes 1 and 2 above, will lead to immediate return from the TPUT procedure to the calling procedure. In the case of codes 3, 4, and 5, normal return will lead to SIGNAL ERROR. If there is no ON statement for the TPTFAIL condition in effect, the action is as if normal return from an on-unit for that condition had occurred.

For information on the TPUT SVC used by this procedure, see Guide to Writing a Terminal Monitor Program or a Command Processor, IBM order number GC28-6764 (in which case the user should know that the TPUT SVC is issued with the CONTROL option).

TELECOMPUTING

TUCC MEMORANDUM

July 14, 1970

Library Services Series
Memorandum No. LS-91

ROUTING
Board of Directors
Computation Centers for Distribution
TUCC/NCECS Staff

FROM: NCSU Computation Center

SUBJECT: Date Conversion Function for PL/I - CALDATE, JULDATE

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus User/Programming Services

Purpose

CALDATE - converts a serial day date (Julian date) to a standard calendar date.

JULDATE - converts a standard calendar date to a serial day date (Julian date).

Both routines produce valid results for dates between 1 March 1901 and 31 December 1999.

Both are called by a standard PL/I calling sequence.

CALDATE and JULDATE are adaptations of Algorithm 199, from Collected Algorithms from CACM, originally written by R. G. Tantzen.

Use

CALDATE and JULDATE are PL/I function procedures of single arguments whose calling sequence can be defined by:

```
DCL  CALDATE ENTRY (CHAR(5)) RETURNS(CHAR(8));

DCL  JULDATE ENTRY (CHAR(6)) RETURNS(CHAR(5));
```

The argument of CALDATE is a string of the form YYDDD and the returned value is of the form DD/MM/YY.

The argument of JULDATE is a string of the form DDMMYY and the returned value is of the form YYDDD.

Neither the input string lengths nor the output string lengths are checked by the routines but assumed to be correct. The "current length" portion of the output dope vector is set to be the correct value on return from the routines.

The contents of the input strings are not checked for validity but assumed to be valid EBCDIC character representation of the decimal digits.

CALDATE occupies 316_{10} bytes of storage and requires about 100 microseconds

JULDATE occupies 188_{10} bytes of storage and requires about 90 microseconds

(Timings are for a S/360/75I.)

Both CALDATE and JULDATE reside in SYS1.PL1SUBS at TUCC.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October 1, 1974

Library Services Series
Document No. LS-280-0

FROM: NCSU Programming Services

SUBJECT: Bessel Functions of the First Kind

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

Described below are PL/I procedures that evaluate J or I Bessel functions of positive or negative order.

The J Bessel function of order v, $J_v(X)$, is a solution of the Bessel equation

$$X^2Y'' + XY' + (X^2 + v^2)Y = 0$$

The I Bessel function of order v, $I_v(X)$, is a solution of the modified Bessel equation

$$X^2Y'' + XY' - (X^2 + v^2)Y = 0$$

With these procedures, the user can request that results be computed to a specified number of significant decimal digits. In all the procedures, accuracy may deteriorate below that requested near a zero of the function. The procedures are most efficient for arguments of small or moderately large modulus. The procedures that evaluate negative order Bessel functions deteriorate in accuracy when the order is nearly integral.

The procedures are based on "Algorithm 236, Bessel Functions of the First Kind", by Walter Gautschi, Communications of the ACM, August 1964.

Implemented versions are:

1) JAPOSN (X,A,NMAX,D,J,ERR)

JAPOSN evaluates to D significant decimal digits the Bessel functions $J_{A+n}(X)$ for given A,X and for n=0,1,...,NMAX. The results are stored in the array J.

The parameters are:

X	BIN FLOAT(53) independent variable value for which J Bessel function is to be evaluated. X>0.
A	BIN FLOAT(53) the J Bessel functions of order (A + n), n=0,1,...,NMAX, are evaluated at X. 0≤A<1.
NMAX	BIN FIXED(15) See description of A. NMAX≥0.
D	BIN FIXED(15) indicates the number of significant decimal digits requested for all the returned values. 0≤D≤16.
J	(L:K) BIN FLOAT(53) on return, J(0) contains $J_{A+0}(X)$, J(1) contains $J_{A+1}(X)$, etc. L≤0 and K≥NMAX.
ERR	BIN FIXED(15) on return, an error indicator whose values have the following meanings:

-1	Requested accuracy unattainable. Results are probably still quite accurate.
0	successful completion
1	A<0 or A≥1 on input
2	X≤0 on input
3	NMAX<0 on input
4	D<0 on input
5	dimensions of array J incorrect. LBOUND(J,1)>0 or HBOUND(J,1)<NMAX.

2) JANEGN (X,A,NMAX,D,J,ERR)

JANEGN evaluates to D significant decimal digits the Bessel functions $J_{A-n}(X)$ for given A,X and for n=0,1,...,NMAX. The results are stored in the array J.

The parameters have the same meaning, precision, and type as defined for JAPOSN except for:

A	BIN FLOAT(53) the J Bessel functions of order (A-n), n=0,1,...,NMAX, are evaluated at X. 0≤A<1.

J (L:K) BIN FLOAT(53) on return, J(0) contains $J_{A-0}(X)$, J(1) contains $J_{A-1}(X)$, etc. L≤0 and K≥NMAX.

3) CJAPOSN (Z,A,NMAX,D,J,ERR)

CJAPOSN evaluates to D significant decimal digits the Bessel functions $J_{A+n}(Z)$ for given real A and complex Z, and for n=0,1,...,NMAX. The results are stored in the complex array J.

The parameters are:

Z BIN FLOAT(53) COMPLEX independent variable value for which J Bessel function is to be evaluated. IMAG(Z)≠0 if REAL(Z)≤0.

A BIN FLOAT(53) the J Bessel functions of order (A + n), n=0,1,...,NMAX, are evaluated at Z. 0≤A<1.

NMAX BIN FIXED(15) See description of A. NMAX≥0.

D BIN FIXED(15) indicates the number of significant decimal digits requested for all the returned values. 0≤D≤16.

J (L:K) BIN FLOAT(53) COMPLEX on return, J(0) contains $J_{A+0}(Z)$, J(1) contains $J_{A+1}(Z)$, etc. L≤0 and K≥NMAX.

ERR BIN FIXED(15) on return, an error indicator whose values have the following meanings:

-1 Requested accuracy unattainable. Results are probably still quite accurate.
0 Successful completion.
1 A<0 or A≥1 on input
2 REAL(Z)≤0 and IMAG(Z)=0 on input
3 NMAX<0 on input
4 D<0 on input
5 dimensions of array J incorrect. LBOUND(J,1)>0 or HBOUND(J,1)<NMAX.

4) IAPOSN (X,A,NMAX,D,I,ERR)

IAPOSN evaluates to D significant decimal digits the modified Bessel functions $I_{A+n}(X)$ for given A,X and for n=0,1,...,NMAX. The results are stored in the array I.

The parameters are:

X	BIN FLOAT(53) independent variable for which I Bessel function is to be evaluated. X>0.
A	BIN FLOAT(53) the I Bessel functions of order (A + n), n=0,1,...,NMAX, are evaluated at X. 0≤A<1.
NMAX	BIN FIXED(15) See description of A. NMAX≥0.
D	BIN FIXED(15) indicates the number of significant decimal digits requested for all the returned values. 0≤D≤16.
I	(L:K) BIN FLOAT(53) on return, I(0) contains $I_{A+0}(X)$, I(1) contains $I_{A+1}(X)$, etc. L≤0 and K≥NMAX.
ERR	BIN FIXED(15) on return, an error indicator whose values have the following meanings:

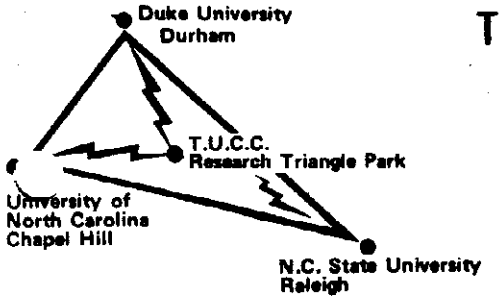
-1 Requested accuracy unattainable. Results are probably still quite accurate.
0 Successful completion.
1 A<0 or A≥1 on input
2 X≤0 on input
3 NMAX<0 on input
4 D<0 on input
5 dimensions of array I incorrect. LBOUND(I,1)>0 or HBOUND(I,1)<NMAX.

5) IANEGN (X,A,NMAX,D,I,ERR)

IANEGN evaluates to D significant decimal digits the modified Bessel function $I_{A-n}(X)$ for given A,X and for n=0,1,...,NMAX. The results are stored in the array I.

The parameters have the same meaning, precision, and type as defined for IAPOSN except for:

A	BIN FLOAT(53) the I Bessel functions of order (A - n), n=0,1,...,NMAX, are evaluated at X. 0≤A<1.
I	(L:K) BIN FLOAT(53) on return, I(0) contains $I_{A-0}(X)$, I(1) contains $I_{A-1}(X)$, etc. L≤0 and K≥NMAX.



TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

April 24, 1974

Library Services Series
Document No. LS-242-0

FROM: NCSU Programming Services

SUBJECT: SFCFIT, ITPLBV - Smooth Bivariate Interpolation

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

PL/I subprocedures SFCFIT and ITPLBV determine a single-valued bivariate function Z(X,Y) which assumes given values at the nodes of a rectangular X-Y grid, and return values of the function at selected points in the X-Y plane. The grid need not be evenly spaced in either direction. The interpolating function Z(X,Y) is smooth, i.e., the function and its first-order partial derivatives are continuous. Emphasis is on avoiding excessive undulation between given grid points.

The procedures are based on the procedures of the same name described in "Algorithm 474, Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures" by Hiroshi Akima, Communications of the ACM, January, 1974, where a further discussion of the problem and algorithm can be found.

In certain cases it is desirable to control the values of the partial derivatives along a boundary of the X-Y grid. This can be accomplished by extending the input data with two additional grid lines parallel to the boundary. For example, if the interpolating function is to exhibit periodicity in X and/or Y, then the input data should consist of the data that cover a whole period and two additional grid lines on each side of them.

ITPLBV (IERR,LX,LY,X,Y,ZZ,N,U,V,W)

Procedure ITPLBV returns the values, W, of the interpolating function Z(X,Y) for a specified set of N points (U,V) in the X-Y plane, where

IERR — BIN FIXED (15) on return contains one of the following values:

0 interpolation successful
1 either LX or LY is less than 2
2 N is less than 1
3 identical X values given
4 the X values are not in ascending order
5 identical Y values given

6 the Y values are not in ascending order
7 the lower bounds of all array argument dimensions are not equal
8 less than N elements in the array argument W

LX — BIN FIXED (15) on entry is the number of X-coordinates in the X-Y grid. (LX>=2)

LY — BIN FIXED (15) on entry is the number of Y- coordinates in the X-Y grid. (LY>=2)

X — (*) BIN FLOAT (21) on entry contains in its first LX elements the X- coordinates of the grid in ascending order.

Y — (*) BIN FLOAT (21) on entry contains in its first LY elements the Y- coordinates of the grid in ascending order. LBOUND (Y,1) must equal LBOUND(X,1).

ZZ — (*,*) BIN FLOAT (21) on entry contains in its first LX rows and LY columns the desired values of the interpolating function at the nodes of the X-Y grid. ZZ(I,J) is the value at the node (X(I),Y(J)). LBOUND(ZZ,1) and LBOUND(ZZ,2) must equal LBOUND(X,1).

N — BIN FIXED (15) , on entry is the number of points at which the value of Z(X,Y) is desired. (N>=1)

U — (*) BIN FLOAT (21) on entry contains in its first N elements the X- coordinates of points at which Z(X,Y) is to be evaluated. LBOUND(U,1) must equal LBOUND(X,1).

V — (*) BIN FLOAT (21) on entry contains in its first N elements the Y- coordinates of points at which Z(X,Y) is to be evaluated. LBOUND(V,1) must equal LBOUND(X,1).

W — (*) BIN FLOAT (21) on return contains in its first N elements the computed values of Z(X,Y). W(I) is the value at (U(I),V(I)). LBOUND(W,1) must equal LBOUND(X,1).

SFCFIT (IERR,LX,LY,X,Y,ZZ,MX,MY,U,V,W)

Given function values, ZZ, at the nodes of a X-Y grid, procedure SFCFIT returns interpolated function values, W, at the nodes of a U-V grid. The U-V grid is a refinement of the X-Y grid in that it contains the nodes of the X-Y grid and nodes introduced by subdividing each interval in X and Y into MX and MY equal subintervals, respectively.

IERR BIN FIXED (15) on return contains one of the following values:

0 interpolation succeasful
1 either LX or LY is less than 2
2 either MX or MY is less than 2
3 either the number of elements in U or the number of rows in W is less than MX*(LX-1)+1
4 either the number of elements in V or the number of columns in W is less than MY*(LY-1)+1.
5 identical X values given
6 the X values are not ordered
7 identical Y values given
8 the Y values are not ordered
9 lower bounds of all array argument dimensions are not equal.

LX,LY,X,Y,ZZ as described for procedure ITPLBV except that the elements of X and/or Y can be in either ascending or descending order.

MX BIN FIXED (15) on entry is the desired number of subintervals between successive X- coordinates of the X-Y grid.

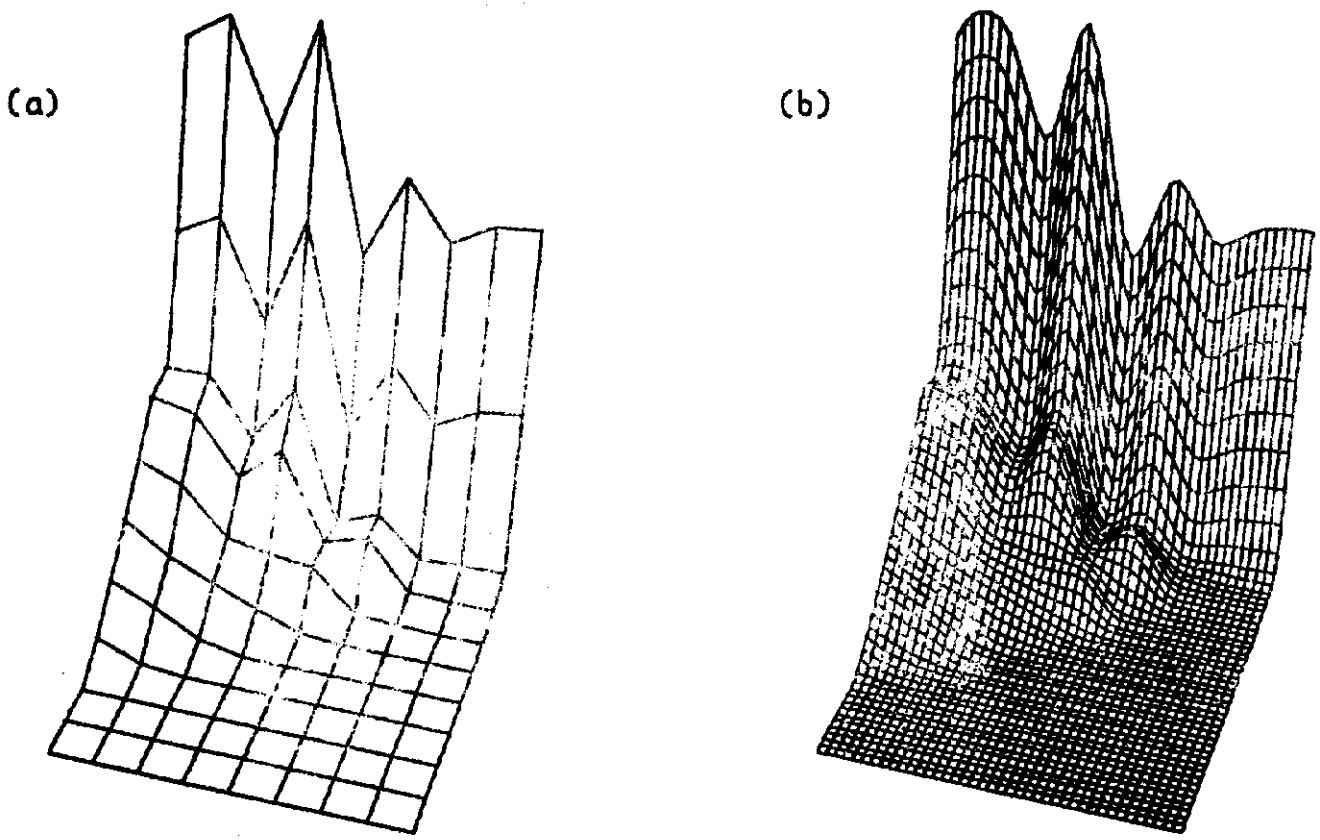
MY BIN FIXED (15) on entry is the desired number of subintervals between successive Y- coordinates of the X-Y grid.

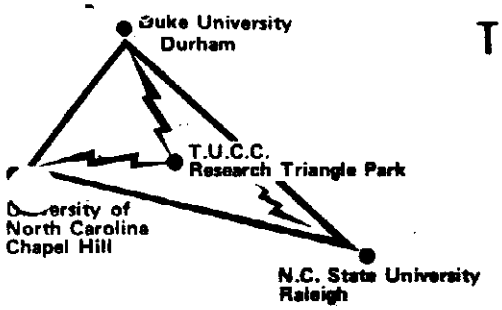
U (*) BIN FLOAT (21) on return contains in its first MX*(LX-1)+1 elements the X- coordinates of the refined grid. LBOUND(U,1) must equal LBOUND(X,1).

V (*) BIN FLOAT (21) on return contains in its first MY*(LY-1)+1 elements the Y- coordinates of the refined grid. LBOUND(V,1) must equal LBOUND(X,1).

W (*,*) BIN FLOAT (21) on return contains in its first MX*(LX-1)+1 rows and MY*(LY-1)+1 columns the interpolated values of the function at the nodes of the U-V grid. W(I,J) contains the value of Z(X,Y) at the node (U(I),V(J)).
LBOUND(W,1) and LBOUND(W,2) must equal LBOUND(X,1).

Perspective representation of (a) original data and of (b) the surface fitted by SFCFIT with LX=11, LY=9, MX=5, and MY=5.





TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

March 8, 1974

Library Services Series
Document No. LS-236-0

FROM: NCSU Programming Services

SUBJECT: STIFDIF, for Integration of Stiff Ordinary Differential Equations

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

STIFDIF is a PL/I subprocedure which solves an initial value problem for a stiff system of first order ordinary differential equations by the "Gear" algorithm. On each invocation, STIFDIF attempts to integrate the system over one step of length H, where H is initially specified by the calling procedure but may be changed by STIFDIF to limit the estimated truncation error to within a specified tolerance, or to take advantage of areas of smoothness. The order of approximation of the integration method is automatically chosen to maximize step size and minimize solution time.

The user must provide a subprocedure which evaluates the first derivatives of the dependent variables with respect to the independent variable and, optionally, one which evaluates the partial derivatives of the first derivatives with respect to the dependent variables.

The general method of operation using STIFDIF is:

1. The user initially invokes the procedure giving an initial value of the independent variable T, the initial solutions at T, a suggested value of the integration step size (always held in H), and a minimum value for that step size, among other parameters.

2. STIFDIF will attempt to compute a new solution at the point T plus the step specified. If it determines the problem is not well conditioned for solution over that interval, it will reduce H and try again. It will not reduce H below the minimum the user has specified.

3. When STIFDIF has completed one integration step with whatever step size is settled on, it returns to the user program. After that return, the value of T has been increased by that step size. The value of H is a suggested value of the interval size for the next integration step. The initial values of the solutions have been replaced with the solutions at the new T.

4. The user must then check the value of T to see if the total problem has been solved. If not, STIFDIF is recalled with the output from this as input, and the cycle from 2 through 4 is repeated. The user can modify operation of the procedure if he desires. Modification of H may be necessary to force the final value of T to be some specified value.

The integration technique for stiff systems is described in "The Automatic Integration of Ordinary Differential Equations", C. W. Gear, Communications of the ACM, March, 1971, p. 176.

Implemented versions are:

1. STIFDIF (N,T,Y,H,HMIN,HMAX,EPS,MF,YMAX,ERROR,KFLAG,JSTART,MAXDER,NQ, DIFFUN,PEDERV)

N BIN FIXED(15,0) contains the number of first order equations in the system. The user may decrease N during the solution cycle if the number of active equations reduces. N must not be increased without setting parameter JSTART to zero.

T BIN FLOAT(53) on call, if JSTART>=0, this contains the value of the independent variable at which the solution Y is known. On return, if KFLAG=1, this contains the value of the independent variable to which the solution has advanced.

Y (*,*)BIN FLOAT(53) must have row dimension 0:7. The column dimension can be whatever is convenient for the user. On first call (with JSTART=0) Y must contain the N initial values of the dependent variables in its first row. On return the first row will contain the computed values of the dependent variables corresponding to the returned value of T. Succeeding rows will contain scaled values of derivatives of successively higher order computed by STIFDIF. These are needed here in subsequent calls with JSTART>0. The number of derivatives present on each return will vary and is returned in parameter NQ, below.

H BIN FLOAT(53) on entry is the step size to be attempted by STIFDIF. This value will be used if it does not cause a larger truncation error than requested by parameter EPS. If a step is successful, then H returns the step size which should be attempted on the next entry.

To save computing time, a fairly small step should be specified on the first entry since a first order method is used initially. It will be automatically increased later. H can be negative.

HMIN BIN FLOAT(53) is the minimum step length that may be used. Note that on starting, this must be much smaller in magnitude than the average H expected.

HMAX BIN FLOAT(53) on call contains the upper limit on the magnitude of H which can be returned by STIFDIF, i.e.: the suggested next step size.

EPS BIN FLOAT(53) is the truncation error test constant. For a successful step, the single-step truncation error estimates ERROR divided by YMAX must be less than EPS in the Euclidean norm. The step size and/or order of approximation will be adjusted to achieve this. EPS>=1E-14 is suggested.

MF BIN FIXED(15,0) is an input parameter that indicates the method desired. The following values are allowed:

1 A multi-step method suitable for stiff systems is used. It will also work for non-stiff systems. The user must provide a subprocedure PEDERV to evaluate the partial derivatives of the differential equations with respect to the dependent variables.

2 The same as 1, except that STIFDIF computes the partial derivatives by numerical differencing of the derivatives. Hence PEDERV is not called.

YMAX (*)BIN FLOAT(53) on return contains the maximum magnitude of each dependent variable through the current step. It should normally be set to 1 before the first call. (See EPS above.) YMAX can be changed by the calling procedure to influence the step control. LBOUND(YMAX,1) should equal LBOUND(Y,2).

ERROR (*)BIN FLOAT(53) on return contains the estimated single step truncation error in each dependent variable. LBOUND(ERROR,1) should equal LBOUND(Y,2).

KFLAG BIN FIXED(15,0) on return contains a completion code with the following meanings:

+1 the step was successful.
-1 the step was taken with H=HMIN, but the requested truncation error was not achieved.
-2 the value of MAXDER on entry is greater than 6.
-3 corrector convergence could not be achieved for H greater than HMIN.
-4 the requested truncation error EPS is too small for an efficient solution of this system by STIFDIF.

JSTART BIN FIXED(15,0) on call must contain one of the following values to indicate the desired stepping procedure:

0 perform the first step.
1 take a new step continuing from the last.
-1 repeat the previous step with a new value of H.

The calling procedure must define JSTART prior to each invocation of STIFDIF. Note that JSTART=1 is the usual choice following a successful step (KFLAG=1).

MAXDER BIN FIXED(15,0) is the maximum order derivative that STIFDIF may use in the current step. This parameter also restricts the order of approximation. 0<MAXDER<7 is required.

NQ BIN FIXED(15,0) on return contains the order of the maximum order derivative used, which is also the current order of approximation, and the index of the last row of Y containing useful values.

DIFFUN ENTRY(/*T*/BIN FLOAT(53),/*Y*/(*,*)BIN FLOAT(53), /*DY*/(*)BIN FLOAT(53))
is the name of the procedure supplied by the user to evaluate the differential equations. The first parameter will be the value of the independent variable. The second parameter is the current Y array as described above. Only the first row, Y(0,*), containing current solution values is of interest at this point. The third parameter is where N equation values are returned. The equations are functions of T and Y. LBOUND of this parameter is equal to LBOUND(Y,2);

PEDERV ENTRY(/*T*/BIN FLOAT(53),/*Y*/(*,*)BIN FLOAT(53), /*PW*/(*,*)BIN FLOAT(21))
is the name of the procedure supplied by the user to evaluate the partial derivatives of the differential equations. The first parameter will be the value of the independent variable. The second parameter is the current Y array as described above. Only the first row, Y(0,*), containing current solution values is of interest at this point. The third parameter is where the N*N derivative values are returned. The equations are functions of T and Y. LBOUND(PW,1) and LBOUND(PW,2) equal the LBOUND(Y,2). PW(I,J) returns the value of the partial derivative of the differential equation with index I with respect to the dependent variable Y(0,J).

If MF=2, then PEDERV can be any identifier with the ENTRY attribute.

Example 1:

An example of the user-supplied procedures is, for the system

$$Y_1' = 4Y_1 + 3Y_2$$

$$Y_2' = 2Y_1 + Y_2$$

```
DIFFUN:  PROC (T,Y,DY);
    DCL(T,Y(*,*),DY(*))BIN FLOAT(53);
       DY(1)=4*Y(0,1) + 3*Y(0,2);
       DY(2)=2*Y(0,1) + Y(0,2);
END DIFFUN;

PEDERV:  PROC (T,Y,PW);
    DCL(T,Y(*,*))BIN FLOAT(53),PW(*,*) BIN FLOAT(21);
   PW(1,1)=4.0 ; PW(1,2)=3.0;
   PW(2,1)=2.0 ; PW(2,2)=1.0;
END PEDERV;
```

where the user chose to declare Y in his program with dimensions (0:7, 1:2).

Example 2:

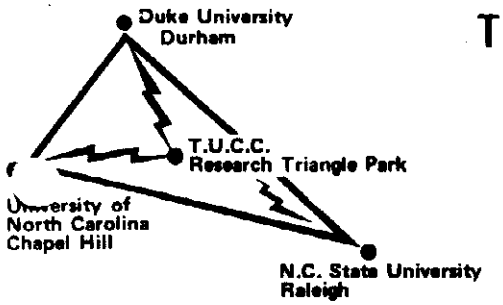
The scaled values of derivatives returned in the array Y can be used to interpolate the solution between successive values of T. Y(J,I) returns the J-th derivative of the I-th dependent variable at T scaled by H**J/J! The solution YE(I) of the I-th equation at T-E could be defined by:

YE(I)= sum, for J=0 TO NQ, of Y(J,I)*(-E/H)**J

References:

Gear, C. W. (1971), "Choosing a Method", Numerical Initial Value Problems in Ordinary Differential Equations, pp. 231-236.

Gear, C. W. (1967), "Numerical Integration of Stiff Ordinary Differential Equations", Report No. 221, Department of Computer Science, University of Illinois.



TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

March 21, 1974

Library Services Series
Document No. LS-235-0

FROM: NCSU Programming Services

SUBJECT: DIFSUB, for Integration of Ordinary Differential Equations

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

DIFSUB is a PL/I subprocedure which solves an initial value problem for a non-stiff system of first order ordinary differential equations by the "Gear" algorithm. On each invocation, DIFSUB attempts to integrate the system over one step of length H, where H is initially specified by the calling procedure but may be changed by DIFSUB to limit the estimated truncation error to within a specified tolerance, or to take advantage of areas of smoothness. The order of approximation of the integration method is automatically chosen to maximize step size and minimize solution time.

The user must provide a subprocedure which evaluates the first derivatives of the dependent variables with respect to the independent variable.

The general method of operation using DIFSUB is:

1. The user initially invokes the procedure giving an initial value of the independent variable T, the initial solutions at T, a suggested value of the integration step size (always held in H), and a minimum value for that step size, among other parameters.

2. DIFSUB will attempt to compute a new solution at the point T plus the step specified. If it determines the problem is not well conditioned for solution over that interval, it will reduce H and try again. It will not reduce H below the minimum the user has specified.

3. When DIFSUB has completed one integration step with whatever step size is settled on, it returns to the user program. After that return, the value of T has been increased by that step size. The value of H is a suggested value of the interval size for the next integration step. The initial values of the solutions have been replaced with the solutions at the new T.

4. The user must then check the value of T to see if the total problem has been solved. If not, DIFSUB is recalled with the output from this as input, and the cycle from 2 through 4 is repeated. The user can modify operation of the procedure if he desires. Modification of H may be necessary to force the final value of T to be some specified value.

The integration technique for non-stiff systems is described in "The Automatic Integration of Ordinary Differential Equations", C. W. Gear, Communications of the ACM, March, 1971, p. 176.

Implemented versions are:

1. DIFSUB (N,T,Y,H,HMIN,HMAX,EPS,NQ,YMAX,ERROR,KFLAG,JSTART,MAXDER,DIFFUN) where

N — BIN FIXED(15,0) contains the number of first order equations in the system. The user may decrease N during the solution cycle if the number of active equations reduces. N must not be increased without setting parameter JSTART to zero.

T — BIN FLOAT(53) on call, if JSTART>=0, this contains the value of the independent variable at which the solution Y is known. On return, if KFLAG=1, this contains the value of the independent variable to which the solution has advanced.

Y — (*,*)BIN FLOAT(53) must have row dimension 0:7. The column dimension can be whatever is convenient for the user. On first call (with JSTART=0),Y must contain the N initial values of the dependent variables in its first row. On return the first row will contain the computed values of the dependent variables corresponding to the returned value of T. Succeeding rows will contain scaled values of derivatives of successively higher order computed by DIFSUB. These are needed here in subsequent calls with JSTART>0. The number of derivatives present on each return will vary and is returned in parameter NQ, below.

H — BIN FLOAT(53) on entry is the step size to be attempted by DIFSUB. This value will be used if it does not cause a larger truncation error than requested by parameter EPS. If a step is successful, then H returns the step size which should be attempted on the next entry.

To save computing time, a fairly small step should be specified on the first entry since a first order method is used initially. It will be automatically increased later. H can be negative.

HMIN BIN FLOAT(53) is the minimum step length that may be used. Note that on starting, this must be much smaller in magnitude than the average H expected.

HMAX BIN FLOAT(53) on call contains the upper limit on the magnitude of H which can be returned by DIFSUB, i.e.: the suggested next step size.

EPS BIN FLOAT(53) is the truncation error test constant. For a successful step, the single-step truncation error estimates ERROR divided by YMAX must be less than EPS in the Euclidean norm. The step size and/or order of approximation will be adjusted to achieve this. EPS>=1E-14 is suggested.

NQ BIN FIXED(15,0) on return contains the order of the maximum order derivative used, which is also the current order of approximation and the index of the last row of Y containing useful values.

YMAX (*)BIN FLOAT(53) on return contains the maximum magnitude of each dependent variable through the current step. It should normally be set to 1 before the first call. (See EPS above.) YMAX can be changed by the calling procedure to influence the step control. LBOUND(YMAX,1) should equal LBOUND(Y,2).

ERROR (*)BIN FLOAT(53) on return contains the estimated single step truncation error in each dependent variable. LBOUND(ERROR,1) should equal LBOUND(Y,2).

KFLAG BIN FIXED(15,0) on return contains a completion code with the following meanings:

+1 the step was successful.
-1 the step was taken with H=HMIN, but the requested truncation error was not achieved.
-2 the value of MAXDER on entry is greater than 7.
-3 corrector convergence could not be achieved for H greater than HMIN.
-4 the requested truncation error EPS is too small for an efficient solution of this system by DIFSUB.

JSTART — BIN FIXED(15,0) on call must contain one of the following values to indicate the desired stepping procedure:

0 perform the first step.
1 take a new step continuing from the last.
-1 repeat the previous step with a new value of H.

The calling procedure must define JSTART prior to each invocation of DIFSUB. Note that JSTART=1 is the usual choice following a successful step (KFLAG=1).

MAXDER — BIN FIXED(15,0) is the maximum order derivative that DIFSUB may use in the current step. This parameter also restricts the order of approximation. 0<MAXDER<8 is required.

DIFFUN — ENTRY(/*T*/BIN FLOAT(53),/*Y*/(*,*)BIN FLOAT(53), /*DY*/(*)BIN FLOAT(53)) is the name of the procedure supplied by the user to evaluate the differential equations. The first parameter will be the value of the independent variable. The second parameter is the current Y array as described above. Only the first row, Y(0,*), containing current solution values is of interest at this point. The third parameter is where N equation values are returned. The equations are functions of T and Y. LBOUND of this parameter is equal to LBOUND(Y,2).

Example 1:

An example of a user-supplied procedure is, for the system

$$Y'_1 = 4Y_1 + 3Y_2$$

$$Y'_2 = 2Y_1 + Y_2$$

```
DIFFUN:  PROC (T,Y,DY);
   DCL(T,Y(*,*),DY(*)) BIN FLOAT(53);
      DY(1)=4*Y(Ø,1) + 3*Y(Ø,2);
      DY(2)=2*Y(Ø,1) + Y(Ø,2);
END DIFFUN;
```

where the user chose to declare Y in his program with dimensions (0:7, 1:2).

Example 2:

The scaled values of derivatives returned in the array Y can be used to interpolate the solution between successive values of T. Y(J,I) returns the J-th derivative of the I-th dependent variable at T scaled by H**J/J! The solution YE(I) of the I-th equation at T-E could be defined by:

YE(I)= sum, for J=0 TO NQ, of Y(J,I)*(-E/H)**J

Reference: Gear, C. W. (1971), "Choosing a Method", Numerical Initial Value Problems in Ordinary Differential Equations, pp. 231-236.

TELECOMPUTING

TUCC MEMORANDUM

May 4, 1973

Library Services Series
Document No. LS-164-0

FROM: NCSU Programming Services

SUBJECT: UNSSOL, for Solving Systems of Linear Equations with a Real Matrix of Coefficients

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the systems of equations defined by AX=B where A is a real matrix of order n; and B is a n x r real matrix of r right-hand sides.

UNSSOL must be preceded by UNSDET (LS-196) or any equivalent routine which performs the decomposition (A=LU) of the coefficient matrix.

AX=B is solved in three steps: interchange the elements of B, Ly=B and UX=y. The solutions, X, replace the matrix B.

This procedure is based on the procedure UNSYMSOL described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) UNSSOL (N,R,AA,INT,BB)
   where:

N	BIN FIXED(31)	gives the order, n, of the matrix A.
R	BIN FIXED(31)	gives the number, r, of right-hand sides.
AA	(*,*)BIN FLOAT(21)	should contain, in its first N rows and N columns, the result of the UNSDET decomposition.
INT	(*)BIN FIXED(31)	contains the record of interchanges produced by UNSDET. LBOUND(INT,1) should equal LBOUND(AA,1).

BB (*,*)BIN FLOAT(21) on call, contains in its first N rows and R columns the matrix of right-hand sides. LBOUND(BB,1) should equal LBOUND(AA,1). On return, BB will contain the elements of X.

2) DUNSSOL (N,R,AA,INT,BB)
where parameters have the same meaning as with UNSSOL except for:

AA (*,*)BIN FLOAT(53)

BB (*,*)BIN FLOAT(53)

TELECOMPUTING

TUCC MEMORANDUM

May 4, 1973

Library Services Series
Document No. LS-166-0

FROM: NCSU Programming Services

SUBJECT: ACCLVE, for Solving Systems of Linear Equations with a Symmetric Positive-Definite Matrix of Coefficients, with Iterative Improvement.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the systems of equations defined by AX=B where A is a real symmetric positive-definite matrix of order n; and B is a n x r real matrix of r right-hand sides.

ACCLVE must be preceded by CHOLT1 (LS-192) or an equivalent routine which performs the decomposition (A=L'L).

ACCLVE uses the procedure CHOLL1 (LS-209), followed by iterative improvement. After each solution, the residuals S=B-AX are calculated and Ay=S is solved. The refinement is continued as long as the maximum correction at any stage is less than half that at the previous stage, until the maximum correction is less than 2*EPS times the maximum $abs(X_{ij})$.

The procedure will exit to a user specified label if iteration fails to improve the solution.

This procedure is based on the procedure ACCSOLVE described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implementations are:

1) ACCLVE (N,R,AA,P,BB,EPS,XX,S,LL,LIST,ILL)
where

N	BIN FIXED(31)	contains the order, n, of the matrix A.
R	BIN FIXED(31)	contains the number, r, of right-hand sides.

AA (*,*)BIN FLOAT(21) contains, in its first N rows and N columns, the CHOET1 decomposition. LBOUND(AA,2) should equal LBOUND(AA,1).

P (*)BIN FLOAT(21) contains, in the first N elements, the reciprocals of the diagonal elements of L as produced by CHOET1. LBOUND(P,1) should equal LBOUND(AA,1).

BB (*,*)BIN FLOAT(21) contains in its first N rows and R columns the matrix of right-hand sides, B. LBOUND (BB,1) should equal LBOUND(AA,1).

EPS BIN FLOAT(21) should be .953674E-6.

XX (*,*)BIN FLOAT(21) on return, will contain, in its first N rows and R columns, the solutions to the R systems of equations. LBOUND(XX,1) should equal LBOUND(AA,1). LBOUND(XX,2) should equal LBOUND(BB,2).

S (*,*)BIN FLOAT(21) on return, will contain, in its first N rows and R columns, the R residual vectors. LBOUND of all dimensions of S should equal corresponding LBOUND of dimensions of BB.

LL BIN FIXED(31) will contain the number of iterations.

LIST (*)BIT(8) In case of failure, this vector will indicate, in the first R elements, the solutions that are correct and those that are incorrect since the routine will fail if even only one solution is incorrect. If the Ith element of LIST equals '0'B then the Ith solution is incorrect; not-equal to '0'B indicates a correct solution. LBOUND(LIST,1) should equal LBOUND(BB,2).

ILL LABEL a user specified label to which ACCLVE will transfer if iteration fails to improve the solution.

2) DACCLVE(N,R,AA,P,BB,EPS,XX,S,LL,LIST,ILL)
where parameters have the same meaning as with ACCLVE except for:

AA (*,*)BIN FLOAT(53)

P (*)BIN FLOAT(53)

BB (*,*)BIN FLOAT(53)

EPS BIN FLOAT(53)

XX (*,*)BIN FLOAT(53)

S (*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

June 13, 1973

Library Services Series
Document No. LS-169-0

FROM: NCSU Programming Services

SUBJECT: UNSLVE, for Solving Systems of Linear Equations with an Unsymmetric Matrix of Coefficients

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the systems of equations defined by AX=B where A is a real unsymmetric matrix of order n; and B is a n x r real matrix of r right-hand sides.

UNSLVE must be preceded by UNSDET (LS-196) or an equivalent routine which performs the decomposition (A=LU).

UNSLVE uses the procedure, UNSSOL (LS-164) followed by iterative improvement. After each solution, the residuals S=B-AX are calculated and Ad=S is solved, overwriting d on S. The refinement is continued as long as the maximum correction at any stage is less than half that at the previous stage, until the maximum correction is less than 2*EPS times the maximum abs(X_{ij}).

The procedure will exit to a user specified label if iteration fails to improve the solution.

This procedure is based on the procedure UNSYMACCSOLVE described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) UNSLVE (N,R,AA,A1,P,BB,EPS,XX,S,L,ILL)
where

N	BIN FIXED(31)	is the order, n, of the matrix A.
R	BIN FIXED(31)	is the number, r, of right-hand sides.
AA	(*,*)BIN FLOAT(21)	contains, in its first N rows and N columns, the elements of matrix A. LBOUND(AA,2) should equal LBOUND(AA,1).

A1 (*,*)BIN FLOAT(21) contains, in its first N rows and N columns, the UNSDET decomposition. LBOUND of all dimensions of A1 should equal corresponding LBOUND of dimensions of AA.

P (*)BIN FIXED(31) contains, in its first N elements, a set of integers describing the interchanges as produced by UNSDET. LBOUND(P,1) should equal LBOUND(AA,1).

BB (*,*)BIN FLOAT(21) contains, in its first N rows and R columns, the matrix of right-hand sides, B. LBOUND(BB,1) should equal LBOUND(AA,1).

EPS BIN FLOAT(21) should be .953674E-6.

XX (*,*)BIN FLOAT(21) on return, will contain, in its first N rows and R columns, the solutions to the R systems of equations. LBOUND(XX,1) should equal LBOUND(AA,1).

S (*,*)BIN FLOAT(21) on return, will contain, in its first N rows and R columns, the R residual vectors. LBOUND of all dimensions of S should equal corresponding LBOUND of dimensions of BB.

L BIN FIXED(31) will contain the number of iterations taken.

ILL LABEL a user specified label to which UNSLVE will transfer if iteration fails to improve the solution.

2) DUNSLVE (N,R,AA,A1,P,BB,EPS,XX,S,L,ILL)
where parameters have the same meaning as with UNSLVE except for:

AA (*,*)BIN FLOAT(53)

A1 (*,*)BIN FLOAT(53)

BB (*,*)BIN FLOAT(53)

EPS BIN FLOAT(53) should be .222044604925031E-15.

XX (*,*)BIN FLOAT(53)

S (*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

June 13, 1973

Library Services Series
Document No. LS-170-0

FROM: NCSU Programming Services

SUBJECT: SYMION, Computes the Inverse of a Positive Definite Symmetric Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which, given a real positive definite symmetric matrix, calculates the inverse.

The upper triangle is not disturbed and is available for use in iterative improvement.

The procedure will fail if the matrix, or the matrix modified by computational rounding errors, is not positive definite.

This procedure is based on the procedure SYMINVERSION described in Handbook of Automatic Computation, Vol 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) SYMION (N,AA,FAIL)
where

N BIN FIXED(31) is the order of the matrix to be inverted.

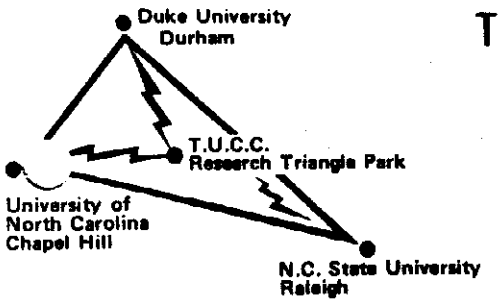
AA (*,*)BIN FLOAT(21) is the working matrix containing the elements of the matrix to be inverted stored in the first N rows and N columns of the upper triangle; on return, the elements of the inverse will be found in the lower triangle beginning at row 1 + LBOUND(AA,1). Note this requires that AA be dimensioned with a number of rows at least equal to N+1.

FAIL LABEL is the label to which the routine is to transfer when the matrix is not positive definite.

2) DSYMION (N,AA,FAIL)
where parameters have the same meaning as with SYMION except for:
AA (*,*)BIN FLOAT(53)

Note: For convenience in referring to the result stored in AA, the user may wish to make use of iSUB defining. If that is the case, the iSUB algorithm is illustrated by the following:

```
DCL AA(6,5),RESULT(5,5) DEF AA(1SUB+1,2SUB);
```



TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

June 13, 1973

Library Services Series
Document No. LS-171-0

FROM: NCSU Programming Services

SUBJECT: CXALVE, for Solving Systems of Equations with a Complex Matrix of Coefficients

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the systems of equation defined by AX=B where A is a complex matrix of order n; and B is a n x r complex matrix of r right-hand sides.

CXALVE must be preceded by COMDET(LS-195) or an equivalent routine which performs the decompositon (A=LU).

CXALVE uses the procedure, COMSOL(LS-174), followed by iterative improvement. After each refinement, the residuals S=B-AX are calculated and Ad=S solved, overwriting d on S. The refinement is continued as long as the maximum correction at any stage is less than half that at the previous stage, until the maximum correction is less than 2*EPS times the maximum abs (X_{ij}).

The procedure will exit to a user specified label if iteration fails to improve the solution.

This procedure is based on the procedure CXACCSOLVE described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) CXALVE (N,R,A1,AA,P,BB,EPS,XX,S,L,ILL)
where

N	BIN FIXED(31)	is the order, n, of the matrix A.
R	BIN FIXED(31)	is the number, r, of right-hand sides.

A1 (*,*)BIN FLOAT(21) COMPLEX contains in its first N rows and N columns the matrix A. LBOUND(A1,2) should equal LBOUND(A1,1).

AA (*,*)BIN FLOAT(21) COMPLEX contains, in its first N rows and N columns, the COMDET decomposition of A. LBOUND of all dimensions of AA should equal corresponding LBOUND of dimensions of A1.

P (*)BIN FIXED(31) contains as first N elements, the vector output by COMDET which designates the row interchanges. LBOUND(P,1) should equal LBOUND(A1,1).

BB (*,*)BIN FLOAT(21) COMPLEX contains in its first N rows and R columns, the matrix of right-hand sides, B. LBOUND (BB,1) should equal LBOUND(A1,1). LBOUND(BB,2) should equal LBOUND(XX,2).

EPS BIN FLOAT(21) should be .953674E-6.

XX (*,*)BIN FLOAT(21) COMPLEX on return, will contain, in its first N rows and R columns, the solutions to the R systems of equations. LBOUND(XX,1) should equal LBOUND(AA,1).

S (*,*)BIN FLOAT(21) on return, will contain, in its first N rows and R columns, the R residual vectors. LBOUND of all dimensions of S should equal corresponding LBOUND of dimensions of BB.

L BIN FIXED(31) will contain the number of iterations taken.

ILL LABEL a user specified label to which CXALVE will transfer if iteration fails to improve the solution.

2) DCXALVE (N,R,A1,AA,P,BB,EPS,XX,B1,L,ILL)
where parameters have the same meaning as with CXALVE except for:

A1 (*,*)BIN FLOAT(53) COMPLEX

AA (*,*)BIN FLOAT(53) COMPLEX

BB (*,*)BIN FLOAT(53) COMPLEX

EPS BIN FLOAT(53) should be .222044604925031E-15

S (*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

April 24, 1973

Library Services Series
Document No. LS-173-0

FROM: NCSU Programming Services

SUBJECT: CHOSOL, for Solving Systems of Equations with a Positive-Definite Symmetric Band Matrix of Coefficients

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the systems of equations defined by AX=B for X, where A is a positive-definite symmetric band matrix, of order n, with m bands on either side of the diagonal and B is an n x r matrix of r right-hand sides. The matrix A must be previously decomposed into A=LL' by CHODET (LS-198) or an equivalent procedure.

Only the lower triangle and the reciprocals of the diagonal elements of L are supplied, with elements stored in the array LL as illustrated (for n=4, m=2) by:

	first column		(m+1)st column
first row →	0	0	L_{11}
	0	L_{21}	L_{22}
	L_{31}	L_{32}	L_{33}
nth row →	L_{42}	L_{43}	L_{44}

Note that the reciprocals of the diagonal elements are in the (m+1)st column. This is the storage mode returned by CHODET.

AX=B is solved in two steps, Ly=B and L'X=y. The matrix B is retained in order to facilitate the refinement of X. X and B can be the same array in the call of the procedure, however, then B will not be retained.

This procedure is based on the procedure CHOBANDSOL described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) CHOSOL (N,M,R,LL,BB,XX)
where

N	BIN FIXED(31)	the order (n) of the matrix A.
M	BIN FIXED(31)	the number (m) of subdiagonal bands.
R	BIN FIXED(31)	the number (r) of right-hand sides.
LL	(*,*)BIN FLOAT(21)	contains in its first N rows and (M+1) columns the result of the CHODET decomposition.
BB	(*,*)BIN FLOAT(21)	contains in its first N rows and R columns the matrix of right-hand sides, B. LBOUND(BB,1) should equal LBOUND(LL,1).
XX	(*,*)BIN FLOAT(21)	on return, will contain in its first N rows and R columns the matrix of solutions. LBOUND of each dimension of XX should equal LBOUND of the corresponding dimension of BB.

2) DCHOSOL (N,M,R,LL,BB,XX)
where parameters have the same meaning as with CHOSOL except for:

LL	(*,*)BIN FLOAT(53)
BB	(*,*)BIN FLOAT(53)
XX	(*,*)BIN FLOAT(53)

TELECOMPUTING

TUCC MEMORANDUM

May 4, 1973

Library Services Series
Document No. LS-174-0

FROM: NCSU Programming Services

SUBJECT: COMSOL, for Solving Systems of Linear Equations with a Complex Matrix of Coefficients

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming

This is a PL/I subprocedure which solves the systems of equations defined by AX=B where A is a complex matrix of order n; and B is a n x r complex matrix of r right-hand sides.

COMSOL must be preceded by COMDET (LS-195) or any equivalent routine which performs the decomposition (A=LU) of the coefficient matrix.

AX=B is solved in three steps: interchange the elements of B, Ly=B, and UX=y. The solutions, X, are returned in matrix B.

This procedure is based on the procedure COMPSOL described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag, (1971), where a description of the method can be found.

Implementation versions are:

1) COMSOL (N,R,AA,INT,BB);

where

N	BIN FIXED(31)	contains the order, n, of the matrix A.
R	BIN FIXED(31)	contains the number, r, of right-hand sides.
AA	(*,*)BIN FLOAT(21) COMPLEX	should contain, in its first N rows and N columns, the result of the COMDET decomposition.

INT (*)BIN FIXED(31) contains a record of interchanges produced by COMDET. LBOUND(INT,1) should equal LBOUND(AA,1).

BB (*,*)BIN FLOAT(21) COMPLEX contains in its first N rows and R columns the matrix of right-hand sides. LBOUND(BB,1) should equal LBOUND(AA,1).

2) DCOMSOL (N,R,AA,INT,BB)
where parameters have the same meaning as with COMSOL except for:

AA (*,*)BIN FLOAT(53) COMPLEX

BB (*,*)BIN FLOAT(53) COMPLEX

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

July 2, 1973

Library Services Series
Document No. LS-179-0

FROM: NCSU Programming Services

SUBJECT: LEAION, for Obtaining the Linear Least Squares Solution of Systems of Linear Equations

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which finds the solutions of AX=B, p over-determined systems of m equations in n unknowns (m≥n). The least squares solution is found.

Householder orthogonal transformations are used and performed by procedure DECOSE (LS-197). Iterative improvement of the results is performed. If the norm of the first correction is not significantly smaller than (.0625 of) the norm of the initial solution, the procedure will fail. The procedure will also fail if the rank (A)<n.

The procedure is based on the procedure LEAST SQUARES SOLUTION described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) LEAION (AA,XX,BB,M,N,P,ETA,SINGULAR)
where

AA	(*,*)BIN FLOAT(21)	contains in its first m rows and n columns, the given matrix (A) of an overdetermined system of m linear equations in n unknowns. AA is undistrubed.
XX	(*,*)BIN FLOAT(21)	on return, contains in its first n rows and p columns, the least squares solution (X). LBOUND(XX,1) should equal LBOUND(AA,2).

BB	(*,*)BIN FLOAT(21)	contains in its m rows and p columns, the p right-hand sides (B) stored as columns. BB is undisturbed. LBOUND(BB,1) should equal LBOUND(AA,1). LBOUND(BB,2) should equal LBOUND(XX,2).
M	BIN FIXED(31)	is the number of equations (m) in the system (rows in A).
N	BIN FIXED(31)	is the number of unknowns (n) in the system (columns in A).
P	BIN FIXED(31)	is the number of systems of equations (p) (columns in the arrays X and B).
ETA	BIN FLOAT(21)	should be .9536743E-6.
SINGULAR	LABEL	is a label in the user program to which LEAION will transfer if the procedure fails.

2) DLEAION (AA,XX,BB,M,N,P,ETA,SINGULAR);
where parameters have the same meaning as with LEAION except for:

AA	(*,*)BIN FLOAT(53)	
XX	(*,*)BIN FLOAT(53)	
BB	(*,*)BIN FLOAT(53)	
ETA	BIN FLOAT(53)	should be .222044604925031E-15.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

July 2, 1973

Library Services Series
Document No. LS-183-0

FROM: NCSU Programming Services

SUBJECT: UNSRAY, for Computing Eigenvectors of a Nonsymmetric Band Matrix, by Iteration

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which, given the band matrix, A, will compute the right and left handed eigenvectors, using inverse iteration and partial pivoting.

The nonsymmetric band matrix, A, is of the order n with ml subdiagonal and m2 superdiagonal bands. The elements of A are stored as n x (ml+m2+1) array. The scheme of storage is illustrated by the case; ml=2, m2=1, n=4:

	first column			(ml+m2+1)th column
first row→	0	0	a_{11}	a_{12}
	0	a_{21}	a_{22}	a_{23}
	a_{31}	a_{32}	a_{33}	a_{34}
nth row→	a_{42}	a_{43}	a_{44}	0

Note that the zeros shown are required and that the diagonal elements occupy column (ml+1) in the array.

The procedure will fail if A-lambda*I is singular, or becomes so due to computational rounding errors.

The procedure is based on the procedure UNSRAY described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) UNSRAY (N,M1,M2,R,MACHEPS,AA,LA,LAMBDA1,L,C,ZR,ZL,FAIL)
where

N	BIN FIXED(31)	gives the order, n, of the band matrix A.
(M1,M2)	BIN FIXED(31)	contains the number of sub-diagonal bands (m1) and superdiagonal bands (m2) as described above.
R	BIN FIXED(31)	gives the number, r, of given eigenvalues in LAMBDA1, below.
MACHEPS	BIN FLOAT(21)	should be .953674E-6.
AA	(*,*)BIN FLOAT(21)	on call, contains in its first N rows and M1+M2+1 columns, the elements of the band matrix A, stored as illustrated above. LBOUND(AA,2) should equal LBOUND(AA,1).
LA	BIN FIXED(31)	contains the limit of the number of iterations to be permitted in the computation of each eigenvector.
LAMBDA1	(*)BIN FLOAT(21)	contains in its first R elements, the given approximate eigenvalues of A. LBOUND(LAMBDA,1) should equal LBOUND(ZR,2).
L	(*)BIN FIXED(15)	on return, the first R elements are such that the ith element of L represents the number of iterations that the ith right-hand eigenvector's computation required. LBOUND(L,1) should equal LBOUND(ZR,2).
C	(*)BIN FIXED(15)	on return, the first R elements are such that the ith element of C represents the number of iterations that the ith left-hand eigenvector's computation required. LBOUND(C,1) should equal LBOUND(ZR,2).
ZR	(*,*)BIN FLOAT(21)	on return, contains in its first N rows and R columns, the R right-hand eigenvectors of A. LBOUND(ZR,1) should equal LBOUND(AA,1).
ZL	(*,*)BIN FLOAT(21)	on return, contains in its first N rows and R columns, the R left-hand eigenvectors of A. LBOUND of all dimensions of ZL should equal LBOUND of the corresponding dimensions of ZR.
FAIL	LABEL	is a user specified label to which UNSRAY will transfer if the procedure fails.

2) DUNSRAY (N,M1,M2,R,MACHEPS,AA,LAMBDA1,L,C,ZR,ZL,FAIL)
where parameters have the same meaning as with UNSRAY except for:

AA	(*,*)BIN FLOAT(53)	
LAMBDA1	(*)BIN FLOAT(53)	
MACHEPS	BIN FLOAT(53)	should be .222044604924031E-15.
ZR	(*,*)BIN FLOAT(53)	
ZL	(*,*)BIN FLOAT(53)	

Note: For convenience in referring to elements of the "compressed" band matrix used by this procedure, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated, for the above example, by:

```
DCL AA(4,-2:1),
    BANDM(4,4) DEF AA(1SUB,2SUB-1SUB);
```

TELECOMPUTING

TUCC MEMORANDUM

January 17, 1973

Library Services Series
Document No. LS-189-0

FROM: NCSU Programming Services

SUBJECT: ORTHO1, for Computing the Inverse of a Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the inverse of a matrix, A, of order n, using iterative improvement of the initial inverse.

This procedure is based on the procedure ORTHO1 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a full description of the method can be found. The method is essentially a modified Schmidt orthogonalization process.

Implemented versions are:

1) ORTHO1 (AA,N,EPS,INV,STOP)
where

AA	(*,*)BIN FLOAT(21)	contains, in its first N rows and N columns, the matrix, A, to be inverted and is unchanged by the process.
N	BIN FIXED(31)	contains the order, n, of the matrix.
EPS	BIN FLOAT(21)	is the maximal desired relative rounding error and should not be less than .953674E-6.
INV	(*,*)BIN FLOAT(21)	will contain the inverse computed by the subprocedure. LBOUND of each dimension should be the same as for the corresponding dimension of AA.
STOP	LABEL	is a label in the user program to which the procedure will transfer if iteration is ineffective. INV will contain arbitrarily approximate values.

2) DORTHO1 (AA,N,EPS,INV,STOP)
where parameters have the same meaning as with ORTHO1 except for:

AA	(*,*)BIN FLOAT(53)	
EPS	BIN FLOAT(53)	should not be less than .2220446049250313E-15
INV	(*,*)BIN FLOAT(53)	

TELECOMPUTING

TUCC MEMORANDUM

January 17, 1973

Library Services Series
Document No. LS-190-0

FROM: NCSU Programming Services

SUBJECT: SYMDET, for Decomposition of a Real Positive Definite Symmetric Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

A PL/I subprocedure which, given a real symmetric positive definite matrix, A, of order n, determines the decomposition A=LDL'; where L is a unit lower triangular matrix (diagonal=1,1,1,...,1), L' is the transpose of L and D is a diagonal matrix. Only the upper triangular portion of A is used and is retained so that the solution can be subsequently improved by iteration.

L is computed and stored in the subdiagonal part of A. The reciprocals of the elements of D are computed and stored in the vector P.

The routine also computes two numbers D1 and D2 from which the determinant can be computed: det A=D1*2**D2 $(1/16 \leq |D1| < 1)$.

The procedure will fail if A, or A modified by computational rounding errors, is not positive definite.

This procedure is based on the procedure SYMDET described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971) where a description of the method can be found.

Implemented versions are:

1) SYMDET (N,AA,P,D1,D2,FAIL)
where

N		BIN FIXED(31) contains the order, n, of the matrix A.
AA	(*,*)	BIN FLOAT(21) contains the matrix to be decomposed in the first N rows and columns of its upper triangle and on return, will have the elements of L in the subdiagonal part.
P	(*)	BIN FLOAT(21) will contain, as its first N elements, the reciprocals of the elements of D. LBOUND(P,1) should equal LBOUND(AA,1).

D1	BIN FLOAT(21)	as described above.
D2	BIN FIXED(31)	as described above.
FAIL	LABEL	is the label to which SYMDET will transfer when A is not positive definite.

2) DSYMDET (N,AA,P,D1,D2,FAIL)
where parameters have the same meaning as with SYMDET except for:

AA	(*,*)BIN FLOAT(53)
P	(*)BIN FLOAT(53)
D1	BIN FLOAT(53)

TELECOMPUTING

TUCC MEMORANDUM

January 17, 1973

Library Services Series
Document No. LS-191-0

FROM: NCSU Programming Services

SUBJECT: GJDEF1, for In-Place Inversion of a Positive Definite Symmetric Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

A PL/I subprocedure which, given a symmetric positive definite matrix, A, of order n, will perform the in-place inversion using the Gauss-Jordan algorithm. Values need be given only in the lower triangle, including the diagonal. The computed inverse will replace these lower triangle elements. The upper triangle is not disturbed nor used by the procedure.

If the matrix, or the matrix modified by computational rounding errors, is not positive definite, the procedure will fail.

This procedure is based on the procedure GJDEF1 described in Handbook of Automatic Computation, Vol 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) GJDEF1 (N,AA,FAIL)

where

N	BIN FIXED(31)	contains the order, n, of matrix A.
AA	(*,*)BIN FLOAT(21)	contains, in the first N rows and N columns of its lower triangle, the matrix, A, to be inverted, as described above. On return, the lower triangle contains the computed inverse.
FAIL	LABEL	is the label in the user program to which GJDEF1 should transfer if inversion fails.

TELECOMPUTING

TUCC MEMORANDUM

January 17, 1973

Library Services Series
Document No. LS-192-0

FROM: NCSU Programming Services

SUBJECT: CHOET1, for Cholesky Decomposition of a Positive Definite Symmetric Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

A PL/I subprocedure which, given a positive definite symmetric matrix, A, of order n, determines the Cholesky decomposition A=LL'. Only the upper triangle of A is used, and is retained through the process so it may be later used for iterative improvement. L, except for its diagonal, is computed and stored in the lower triangle of A. The reciprocals of the diagonal elements of L are stored in a vector P.

The routine also computes two numbers D1 and D2 from which the determinant can be computed: det A=D1*2**D2 $(1/16 \leq |D1| < 1)$.

The procedure will fail if A, or A modified by computational rounding errors, is not positive definite.

This procedure is based on the procedure CHOLDET1 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) CHOET1 (N,AA,P,D1,D2,FAIL)

where

N	BIN FIXED(31)	contains the order, n, of the matrix A.
AA	(*,*)BIN FLOAT(21)	contains the matrix A in its first N rows and N columns. Only the upper triangle elements need be correct.
P	(*)BIN FLOAT(21)	is a vector to contain, as its first N elements, the reciprocals of the diagonal elements of L. LBOUND(P,1) should be the same as LBOUND(AA,1).

D1	BIN FLOAT(21)	as described above
D2	BIN FIXED(31)	as described above
FAIL	LABEL	is the label to which CHOET1 will transfer when A is not positive definite.

2) DCHOET1 (N,AA,P,D1,D2,FAIL)
where parameters have the same meaning as with CHOET1 except for:

AA	(*,*)BIN FLOAT(53)
P	(*)BIN FLOAT(53)
D1	BIN FLOAT(53)

TELECOMPUTING

TUCC MEMORANDUM

February 7, 1973

Library Services Series
Document No. LS-196-0

FROM: NCSU Programming Services

SUBJECT: UNSDET, for Crout Factorization and Determinant of a Real Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

A PL/I subprocedure which gives the Crout factorization of a real square matrix, A, and produces the determinant of A as a by-product. A, of order n, is decomposed into the product LU where L is a lower triangular matrix and U is a unit upper triangular matrix (diagonal=1,1,1,...,1). L is overwritten on the lower triangle and diagonal elements of A. U is overwritten on the upper triangle of A, omitting the constant diagonal.

A record of any interchanges made to the rows of A is kept in the vector INT, such that the I-th row and the INT(I)-th row were interchanged at the i-th step.

Two numbers, D1 and D2, are computed which can be used to determine the value of the determinant:

$$\det A = D1 * 2^{**}D2 \quad \left(\frac{1}{16} \le |D1| < 1\right).$$

The procedure will fail if A, or A modified by computational rounding errors, is singular or almost singular.

This procedure is based on the procedure UNSYMDET described in Handbook of Automatic Computation, Vol 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) UNSDET(N,EPS,AA,D1,D2,INT,FAIL)
where

N	BIN FIXED(31)	gives the order (n) of the matrix A,
EPS	BIN FLOAT(21)	should be .9536743E-6

AA	(*,*)BIN FLOAT(21)	on call, contains the matrix to be factored in its first N rows and N columns; on return, contains the L and U matrices, as described above.
D1	BIN FLOAT(21)	described above
D2	BIN FIXED(31)	described above
INT	(*)BIN FIXED(15)	contains a record of interchanges, as mentioned above, in its first N elements. LBOUND (INT,1) should equal LBOUND(AA,1).
FAIL	LABEL	a label in the user program to which the procedure will transfer, if the process fails.

2) DUNSDET(M,EPS,AA,D1,D2,INT,FAIL)
where parameters have the same meaning as with UNSDET except for:

EPS	BIN FLOAT(53)	
AA	(*,*)BIN FLOAT(53)	
D1	BIN FLOAT(53)	

TELECOMPUTING

TUCC MEMORANDUM

February 8, 1973

Library Services Series
Document No. LS-197-0

FROM: NCSU Programming Services

SUBJECT: DECOSE, for Reducing a Rectangular Matrix to Upper Right Triangular Form

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

A PL/I subprocedure which, given the real matrix Q (m x n,m>=n), reduces it to upper right triangular form by means of n elementary orthogonal transformations (I-beta*U*U'). The diagonal elements of the reduced matrix are stored in the vector A; the off-diagonal elements in the upper right triangular part of Q. The components of the vectors U are stored on and below the leading diagonal of Q. (By definition, the first k-1 elements of the kth U are zero. These zeros are omitted in the storage.)

Pivoting is done by choosing at each step the column with the largest sum of squares to be reduced next. These interchanges are recorded in the vector P.

If at any stage, the sum of squares of the column to be reduced is exactly equal to zero, then the procedure exits to a user specified label.

This procedure is based on the procedure DECOMPOSE (contained in procedure LEAST SQUARES SOLUTION) described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) DECOSE (M,N,QQ,A,P,SINGULAR)
where

M	BIN FIXED(31)	is the number of rows (m) in array Q
N	BIN FIXED(31)	is the number of columns (n) in array Q
QQ	(*,*)BIN FLOAT(21)	contains, in its first M rows and N columns, Q, the array to be reduced, on return, it will contain elements of the reduced matrix and transformation vectors as described above.

A (*)BIN FLOAT(21) will contain the diagonal elements of the reduced matrix in its first N elements, as described above. LBOUND(A,1) should be equal to LBOUND(QQ,2).

P (*)BIN FIXED(15) will contain the record of interchanges, in its first N elements, as mentioned above. LBOUND(P,1) should be equal to LBOUND(QQ,2).

SINGULAR LABEL the label to which the routine is to transfer if the process fails as described above.

2) DDECOSE(M,N,QQ,A,P,SINGULAR)
where parameters have the same meaning as with DECOSE except for:

QQ (*,*)BIN FLOAT(53)

A (*)BIN FLOAT(53)

TELECOMPUTING

TUCC MEMORANDUM

February 21, 1973

Library Services Series
Document No. LS-198-0

FROM: NCSU Programming Services

SUBJECT: CHODET, for Decomposition of a Positive Definite Symmetric Band Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

A PL/I subprocedure which, given a positive definite symmetric band matrix A, of order n, with m bands on either side of the diagonal, computes the Cholesky decomposition $A=LL^t$.

Only the lower triangle and diagonal of A are supplied, with elements stored in the array AA as illustrated, for n=4, m=2, by:

	first column		(m+1)st column
first row →	0	0	a_{11}
	0	a_{21}	a_{22}
	a_{31}	a_{32}	a_{33}
nth row →	a_{42}	a_{43}	a_{44}

Note that the zeros shown are required and that the diagonal is in the (m+1)st column.

The banded lower triangle result is stored in the same fashion in array L. However, the reciprocals of the diagonal elements are stored rather than the diagonals themselves. A is retained so the solution can be subsequently improved through iteration. However, AA and L can be the same array on call, in which case A will be destroyed.

The procedure computes two numbers, D1 and D2, which can be used to determine the determinant: det A=D1*2**D2 ($1/16 \leq |D1| < 1$).

The procedure will fail if A, or A modified by computational rounding errors, is not positive definite.

This procedure is based on the procedure CHOBANDDET described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) CHODET (N,M,AA,L,D1,D2,FAIL)
where

N	BIN FIXED (31)	the order (n) of the matrix A
M	BIN FIXED (31)	the number (m) of subdiagonal bands
AA	(*,*)BIN FLOAT(21)	contains the matrix A, stored in the manner described above, in its first N rows and M+1 columns.
L	(*,*)BIN FLOAT(21)	will contain the result of the decomposition as described above. LBOUND of each dimension of L should equal LBOUND of the corresponding dimension of AA.
D1	BIN FLOAT(21)	as described above.
D2	BIN FIXED(31)	as described above.
FAIL	LABEL	is a label in the user program to which CHODET will transfer if the procedure fails.

2) DCHODET (N,M,AA,L,D1,D2,FAIL)
where parameters have the same meaning as with CHODET except for:

AA	(*,*)BIN FLOAT(53)
L	(*,*)BIN FLOAT(53)
D1	BIN FLOAT(53)

Note: For convenience in referring to elements of the "compressed" band matrices used by this procedure, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated, for the above example by:

```
DCL AA(4,-2:0), L(4,-2:0) FLOAT;
    A(4,4) DEF AA(1SUB,2SUB-1SUB);
    RESULT(4,4) DEF L(1SUB,2SUB-1SUB);
```

TELECOMPUTING

TUCC MEMORANDUM

March 14, 1973

Library Services Series
Document No. LS-199-0

FROM: NCSU Programming Services

SUBJECT: BANET1, for Factorization of a Real Band Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure, which given the band matrix, A, and a number, lambda, factors the matrix (A-lambda*I) into the product of a lower-triangular matrix and an upper-triangular matrix, where I is the identity matrix. The method is a modified Crout decomposition with partial pivoting. (See the text referenced below.)

The band matrix, A, is of order n with m1 subdiagonal bands and m2 superdiagonal bands. The elements of A are stored as an n by (m1+m2+1) array, in the array AA. The scheme of storage is illustrated by the case m1=2, m2=1, n=4:

	first column			(m1+m2+1)th column
first row →	0	0	a_{11}	a_{12}
	0	a_{21}	a_{22}	a_{23}
	a_{31}	a_{32}	a_{33}	a_{34}
nth row →	a_{42}	a_{43}	a_{44}	0

Note that the zeros shown are required and that the diagonal elements occupy column m1+1 in the array.

The banded upper triangle of the result will be left in AA in the same storage fashion (from the (m1)th column to the right). The banded lower triangle results are left, stored in the same fashion, in the first n rows and m1+1 columns of the array M (the diagonal is in the (m1)th column).

The procedure also computes two numbers, D1 and D2, which can be used to determine the determinant of (A-lambda*I): det A=D1*2**D2 $(1/16 \leq |D1| < 1)$.

A user supplied parameter E is used to select the course of action when a zero pivot arises. When E=1, a zero pivot is replaced by approximately MACHEPS * $||A||_\infty$. When E=Ø, and a zero pivot is encountered (implying that (A-lambda*I) is computationally singular), the procedure fails.

The procedure is based on the procedure BANDET1 described in Handbook of Automatic Computation, Vol. II, Linear Algebra, Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) BANET1(N,M1,M2,E,LAMBDA,MACHEPS,AA,D1,D2,M,INT,FAIL)
where

N	BIN FIXED(31)	is the order, n, of the matrix A.
(M1,M2)	BIN FIXED(31)	are the number of subdiagonal bands (m1) and superdiagonal bands (m2), as described above.
E	BIN FIXED(31)	as described above.
LAMBDA	BIN FLOAT(21)	lambda, as described above.
MACHEPS	BIN FLOAT(21)	should be .953674E-6.
AA (*,*)	BIN FLOAT(21)	on call, contains, in its first N rows and M1+M2+1 columns, the elements of the band matrix A, as described above; on return, contains the resultant upper triangle, stored as described above.
D1	BIN FLOAT(21)	as described above.
D2	BIN FIXED(31)	as described above.
M (*,*)	BIN FLOAT(21)	on return, will contain, in its first N rows and M1 columns, the elements of the resultant lower triangle, as described above. LBOUND(M,1) should equal LBOUND(AA,1).

INT (*)BIN FIXED(15) is a vector of at least N elements in which details of the pivoting interchanges will be stored. LBOUND(INT,1) should equal LBOUND(AA,1).

FAIL LABEL is a label in the user program to which the procedure will transfer if the process fails, as described above.

2) DBANET1(N,M1,M2,E,LAMBDA,MACHEPS,AA,D1,D2,M,INT,FAIL)
where parameters have the same meaning as with BANET1 except for:

LAMBDA BIN FLOAT(53)

MACHEPS BIN FLOAT(53) should be .222044604925031E-15

AA (*,*)BIN FLOAT(53)

D1 BIN FLOAT(53)

M (*,*)BIN FLOAT(53)

Note: For convenience in referring to elements of the "compressed" band matrices used by this procedure, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated, for the above example, by:

```
DCL AA(4,-2:1), M(4,2) FLOAT,
    BANDM(4,4) DEF AA(1SUB,2SUB-1SUB),
    LOWTRI(4,4) FLOAT DEF M(1SUB,2SUB-1SUB),
    UPTRI(4,4 DEF AA(1SUB,2SUB-1SUB);
```

TELECOMPUTING

TUCC MEMORANDUM

Library Services Series
Document No. LS-200-0

February 21, 1973

FROM: NCSU Programming Services

SUBJECT: CHOON1, for Computing the Inverse of a Real Positive Definite Symmetric Matrix by Cholesky Decomposition

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

A PL/I subprocedure which, given a real positive definite symmetric matrix, calculates the inverse by Cholesky decomposition.

The upper triangle is not disturbed and is available for use in iterative improvement.

The procedure will fail if the matrix, or the matrix modified by computational rounding errors, is not positive definite.

This procedure is based on the procedure CHOLINVERSION1 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) CHOON1(N,AA,FAIL)
where

N	BIN FIXED(31)	gives the order of the matrix to be inverted.
AA	(*,*)BIN FLOAT(21)	is the working matrix containing the elements of the matrix to be inverted stored in the first N rows and N columns of the upper triangle; on return, the elements of the inverse will be found in in the lower triangle beginning at row 1 + LBOUND(AA,1). Note this requires that AA be dimensioned with a number of rows at least equal to N + 1.
FAIL	LABEL	is the label to which the routine is to transfer when the matrix is not positive definite.

2) DCHOON1(N,AA,FAIL)
where parameters have the same meaning as with CHOON1 except for:

AA (*,*)BIN FLOAT(53)

Note: For convenience in referring to the result stored in AA, the user may wish to make use of iSUB defining. If that is the case, the iSUB algorithm is illustrated by the following:

```
DCL AA(6,5), RESULT(5,5) DEF AA(1SUB + 1, 2SUB);
```

TELECOMPUTING

TUCC MEMORANDUM

March 9, 1973

Library Services Series
Document No. LS-202-0

FROM: NCSU Programming Services

SUBJECT: CHOON2, for Computing the Inverse of a Real Positive Definite Symmetric Matrix by Cholesky Decomposition

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which, given a real positive definite symmetric matrix, A, calculates the inverse by Cholesky decomposition.

Only the lower triangle of the matrix to be inverted is stored, row by row, in a vector of $n*(n+1)/2$ elements, where n is the order of the matrix; e.g. a 3x3 matrix, A, would be stored as $(A_{11},A_{21},A_{22},A_{31},A_{32},A_{33})$. CHOON2 is essentially the same as CHOON1, except for the compressed storage of A.

Elements of the computed inverse replace those of the original matrix.

The procedure will fail if the matrix, or the matrix modified by computational rounding errors, is not positive definite.

This procedure is based on the procedure CHOLINVERSION2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) CHOON2(N,AA,FAIL)
where

N BIN FIXED(31) gives the order of the matrix to be inverted.

AA (*)BIN FLOAT(21) is the working vector containing the elements of the matrix to be inverted, stored in the first N*(N+1)/2 elements of the vector. On return, the elements of the inverse will be contained in the first N*(N+1)/2 elements of the vector.

FAIL LABEL is the label to which the routine is to transfer when the matrix is not positive definite.

2) DCHOON2(N,AA,FAIL)
where parameters have the same meaning as with CHOON2 except for:
AA (*)BIN FLOAT(53)

Note: For convenience in referring to elements of the "compressed" symmetric matrix, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated by:

```
DCL AA(6),A(3,3) DEF AA(2SUB+(1SUB*(1SUB-1))/2);
```

TELECOMPUTING

T U C C MEMORANDUM

March 5, 1973

Library Services Series
Document No. LS-203-0

FROM: NCSU Programming Services

SUBJECT: SVD, for Computing the Singular Values and Orthogonal Decomposition of a Real Rectangular Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which, given a real m by n matrix (m>=n), A, decomposes the matrix into the form

$$A=UQV'$$

where

$$U'U=V'V=VV'=I \text{ and } Q=\text{diag}(\sigma_1,\ldots,\sigma_n)$$

The matrix U consists of n orthonormalized eigenvectors associated with the n largest eigenvalues of AA'; and V, the orthonormalized eigenvectors of A'A. The σ_i are the non-negative square roots of the eigenvalues of A'A; they are called the "singular values" of A. The above description is called the "singular value decomposition" (SVD).

Householder reduction to bidiagonal form followed by QR iteration is used.

A few of the applications of SVD are:

calculation of the pseudoinverse of an m by n matrix
solution of homogeneous equations
determining the rank of matrices
a generalization of the least squares problem

This procedure is based on the procedure SVD described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) SVD (M,N,WITHU,WITHV,EPS,TOL,AA,QQ,UU,VV)
where

M	BIN FIXED(31)	contains the number of rows (m) in matrix A.
N	BIN FIXED(31)	contains the number of columns (n) in matrix A.
WITHU	BIT(8)	'0'B if U is not to be computed, non-zero otherwise.
WITHV	BIT(8)	'0'B if V is not to be computed, non-zero otherwise.
EPS	BIN FLOAT(21)	a constant used for convergence testing; should not be less than .953674E-6.
TOL	BIN FLOAT(21)	should be .353737E-73.
AA	(*,*)BIN FLOAT(21)	contains the matrix A in its first M rows and N columns. AA will be changed only if used as UU. (Cf.)
QQ	(*)BIN FLOAT(21)	will contain the singular values of A in its first N elements; they are non-negative but not necessarily ordered in descending sequence. LBOUND (QQ,1) should be the same as LBOUND(AA,2).
UU	(*,*)BIN FLOAT(21)	If WITHU is non-zero, then UU will contain the matrix U in its first M rows and N columns; otherwise UU is used for working storage and could be the same array as AA. LBOUND(UU,1) should be the same as LBOUND(AA,1). LBOUND(UU,2) should be the same as LBOUND(AA,2).
VV	(*,*)BIN FLOAT(21)	If WITHV is non-zero, then VV will contain the matrix V in its first M rows and N columns; and LBOUND(VV,1) and LBOUND(VV,2) should be the same as LBOUND(AA,2). If WITHV='Ø'B, then VV is not used and can be any 2-dimensional array; for example, the same array as AA or UU.

TELECOMPUTING

TUCC MEMORANDUM

March 7, 1973

Library Services Series
Document No. LS-204-0

FROM: NCSU Programming Services

SUBJECT: ORTHO2, for Inversion of a Real Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the inverse of a matrix, A, of order n. The inversion is performed in-place, so that on return, A contains the inverse. A modified Schmidt orthogonalization process is used. The same algorithm with iterative improvement is offered by ORTHO1. No tests for singularity are made.

This procedure is based on the procedure ORTHO2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) ORTHO2(N,AA)
where
N BIN FIXED(31) gives the order of the matrix.
AA (*,*)BIN FLOAT(21) contains in its first N rows and N columns the matrix to be inverted; on return, it will contain the computed inverse. LBOUND(AA,2) should equal LBOUND(AA,1).

2) DORTHO2(N,AA)
where parameters have the same meaning as with ORTHO2 except for:
AA (*,*)BIN FLOAT(53)

TELECOMPUTING

T U C C

MEMORANDUM

March 8, 1973

Library Services Series
Document No. LS-205-0

FROM: NCSU Programming Services

SUBJECT: CG, for Iterative Solution of a System of Linear Equations with a Symmetric Coefficient Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus-NCECS Programming Services

This is a PL/I subprocedure which, using the Conjugate Gradient (CG) algorithm (an iterative method), solves the linear equations Ax+b=0, where A is a symmetric positive definite coefficient matrix of order n.

This procedure can also be used for investigating the spectrum of A.

This method is advantageous for sparse matrices A. For use in this procedure, the coefficient matrix A is not given as an array of numbers. The user is expected to supply a procedure which computes, from any given vector V, the product A x V.

A should not be too badly "scaled": the euclidian lengths of the n columns should not be of totally different order of magnitude. This may require a transformation of the system to be solved; the safest way being to make all diagonal elements 1 while preserving symmetry. Such a transformation might be to replace A with RAR and b with Rb, where R is a diagonal matrix in which $R_{ii} = 1/\sqrt{|A_{ii}|}$. The solutions, x, obtained from CG can then be transformed back to solutions of the original system by Rx.

The procedure will transfer to a user specified label if the process fails either because the matrix is not positive definite or is too poorly conditioned (the smallest eigenvalue is in the roundoff error level). In either case, no results are given.

The results of the procedure are left in three vectors, X, Q and E. X contains the approximate solution to the system. Q and E contain values q_k and e_k useful in the spectrum investigation and study of convergence properties, as described in the following reference.

The procedure is based on the procedure CG described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) CG(N,P,B,OP,X,NN,Q,E,INDEF)
where

N BIN FIXED(31) gives the order (n) of the matrix A, also see X.

P BIN FIXED(31) gives the upper bound in the declaration of the arrays Q and E (see later), or the limit to which they are to be used.

B (*)BIN FLOAT(21) contains the constant terms, b_k, of the equations in its first N elements. LBOUND(B,1) should equal LBOUND(X,1).

OP ENTRY(BIN FIXED(31),(*)BIN FLOAT(21),(*)BIN FLOAT(21)) is the name of a user supplied call-procedure with the parameter list (N,V,AV). This procedure is to compute the product AV=A x V. Neither N nor V should be changed by the user procedure. The bounds of V and AV should be the same as the bounds of X. N will be ABS(N) as passed to CG.

X (*)BIN FLOAT(21) on return, this vector will contain, in its first N elements, the approximate solutions to the system. On call, in the normal case (N>0), the values in X are irrelevant. However, if N is negative, X is assumed to contain n initial guesses of the solutions, and ABS(N) is used for N in this procedure

NN BIN FIXED(31) gives the maximum number of CG steps to be performed. On return, this will be the actual number performed. If this is equal to the original value, more steps may have produced a more precise solution.

(Q,E) (*)BIN FLOAT(21) will contain the values q_k, e_k as their first elements. If the returned value of NN<P, the vectors have only NN elements; otherwise, P elements, beginning at the LBOUND of the vectors. LBOUND(Q,1) should equal LBOUND(E,1).

INDEF LABEL is the label in the user program to which the procedure should transfer if A is not positive definite or is too ill conditioned (used the first time a q_k becomes non-positive). In either case no results are given.

2) DCG (N,P,B,OP,X,NN,Q,E,INDEF)
where parameters have the same meaning as with CG except for

B	(*)BIN FLOAT(53)
OP	ENTRY(BIN FIXED(31),(*)BIN FLOAT(53),(*)BIN FLOAT(53))
X	(*)BIN FLOAT(53)
(Q,E)	(*)BIN FLOAT(53)

TELECOMPUTING

TUCC MEMORANDUM

March 15, 1973

Library Services Series
Document No. LS-206-0

FROM: NCSU Programming Services

SUBJECT: BANET2, for Factorization of a Real Band Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure, which given the band matrix, A, and a number, lambda, factors the matrix (A-lambda*I) into the product of a lower-triangular matrix and an upper-triangular matrix. The method is a modified Crout decomposition with partial pivoting. The number of agreements in sign between consecutive leading principal minors is also given in paramter C. When A is symmetric, C is the number of eigenvalues greater than lambda. This procedure differs from BANET1 in that an alternative factorization yielding C is used. If C is not required, BANET1 is preferred.

The band matrix, A, is of order n with m1 subdiagonal bands and m2 superdiagonal bands. The elements of A are stored as an n x (m1+m2+1) matrix, in the array AA. The scheme of storage is illustrated by the case m1=2, m2=1, n=4:

	first column			(m1+m2+1)th column
first row →	0	0	a_{11}	a_{12}
	0	a_{21}	a_{22}	a_{23}
	a_{31}	a_{32}	a_{33}	a_{34}
nth row →	a_{42}	a_{43}	a_{44}	0

Note that the zeros shown are required and that the diagonal elements occupy column m1+1 in the array.

The banded upper triangle of the result will be left in AA in the same storage fashion (from the (ml+1)st column to the right). The banded lower triangle results are left, stored in the same fashion, in the first n rows and ml columns of the array M (the diagonal is in the (ml)th column).

The procedure also computes two numbers, D1 and D2, which can be used to determine the determinant of (A-lambda*I): which is A=D1*2**D2 ($1/16 \leq |D1| < 1$).

A parameter E is used to select the course of action when a zero pivot arises. When E=1, a zero pivot is replaced by approximately MACHEPS * $||A||_\infty$ (MACHEPS is a user supplied constant). When E=$\emptyset$, and a zero pivot is encountered implying that (A-lambda*I) is computationally singular), the procedure fails.

The procedure is based on the procedure BANDET2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) BANET2 (N,M1,M2,E,LAMBDA,MAChEPS,AA,D1,D2,C,M,INT,FAIL)
where

N	BIN FIXED(31)	is the order, n, of the matrix A.
(M1,M2)	BIN FIXED(31)	are the number of subdiagonal bands (m1) and superdiagonal bands (m2), as described above.
E	BIN FIXED(31)	as described above.
LAMBDA	BIN FLOAT(21)	lambda, as described above.
MACHEPS	BIN FLOAT(21)	should be .953674E-6.
AA (*,*)	BIN FLOAT(21)	on call, contains, in its first N rows and M1+M2+1 columns, the elements of the band matrix A, as described above; on return, contains the resultant upper triangle, stored as described above.
D1	BIN FLOAT(21)	as described above.
D2	BIN FIXED(31)	as described above.

C BIN FIXED(31) as described above.

M (*,*)BIN FLOAT(21) on return, will contain, in its first N rows and M1 columns, the elements of the resultant lower triangle, as described above. LBOUND(M,1) should equal LBOUND(AA,1)

INT (*,*)[illegible] Boolean array which on return, will contain, in its first N rows and M1 columns, details of the [illegible] row interchanges used in the triangulation. LBOUND(INT,1) should equal LBOUND(AA,1). LBOUND(AA,2) should equal LBOUND(M,2)

FAIL [illegible] program to which [illegible] process fails, [illegible]

2) DBANET2 (N,M1,M2,[illegible],M,INT,FAIL)
where parameters [illegible] BANET2 except for:
LAMBDA BIN FLOAT(53)

MACHEPS BIN [illegible]E-15.

AA (*,*)BIN FLOAT(53)

D1 BIN FLOAT(53)

M (*,*)BIN FLOAT(53)

Note: For convenience in referring to elements of the "compressed" band matrices used by this procedure, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated, for the above example, by:

```
DCL AA(4,-2:1),M(4,2) FLOAT,
    BANDM(4,4) DEF AA(1SUB,2SUB-1SUB)
    LOWTRI(4,4) FLOAT DEF M(1SUB,2SUB-1SUB),
    UPTRI(4,4) DEF AA(1SUB,2SUB-1SUB);
```

TELECOMPUTING

TUCC MEMORANDUM

March 15, 1973

Library Services Series
Document No. LS-207-0

FROM: NCSU Programming Services

SUBJECT: GJDEF2, for In-Place Inversion of a Positive Definite Symmetric Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which, given a symmetric positive definite matrix A, of order n, will perform an in-place inversion using the Gauss-Jordan algorithm. Only the lower triangle and diagonal of the matrix to be inverted need be given, stored as a vector of n*(n+1)/2 elements. A 3x3 matrix A would be stored as (A_{11},A_{21},A_{22},A_{31},A_{32},A_{33}). The elements of the computed inverse replace the original elements of A. GJDEF2 is essentially the same as GJDEF1, except for the compressed storage of A.

If the matrix, or the matrix modified by computational rounding errors, is not positive definite, the procedure will fail.

This procedure is based on the procedure GJDEF2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implementation restrictions are:

1) GJDEF2 (N, AA, FAIL)
where

N BIN FIXED(31) contains the order, n, of matrix A.

AA (*)BIN FLOAT(21) contains, in its first N*(N+1)/2 elements, the matrix, A, to be inverted, as described above. On return, the first N*(N+1)/2 elements contains the computed inverse.

FAIL LABEL specifies a label in the user program to which GJDEF2 should transfer if inversion fails.

Note: For convenience in referring to elements of the "compressed" symmetric matrix, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated by:

```
DCL AA(6),A(3,3)DEF AA(2SUB+(1SUB*(1SUB-1))/2);
```

TELECOMPUTING

TUCC MEMORANDUM

April 6, 1973

Library Services Series
Document No. LS-209-0

FROM: NCSU Programming Services

SUBJECT: CHOOL1, for Solving Systems of Equations with a Positive-Definite Symmetric Matrix of Coefficients

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the systems of equations defined by AX=B for X, where A is a positive-definite symmetric matrix, of order n, and B is a n by r matrix of r right-hand sides. CHOOL1 must be preceded by CHOET1 (LS-192) or an equivalent procedure which performs the decomposition (A=LL') of the coefficient matrix. CHOOL1 may be used any number of times, for different B, after one use of CHOET1.

Only the lower triangle of A needs to be correct. The reciprocals of the diagonal elements of the decomposition must be supplied in a vector P, as provided by CHOET1.

AX=B is solved in two steps, Ly=B and L'X=y. The matrix B is retained in order to facilitate the later refinement of X. X and B can be the same array in the call of the procedure; however, then B will not be retained.

This procedure is based on the procedure CHOLSOL1 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) CHOOL1 (N,R,AA,P,BB,XX)
   where

N	BIN FIXED(31)	is the order (n) of matrix A.
R	BIN FIXED(31)	is the number (r) of right-hand sides.

AA (*,*)BIN FLOAT(21) contains, in its first N rows and N columns, the CHOET1 decomposition of A.

P (*)BIN FLOAT(21) contains in, its first N elements, the reciprocals of the diagonal elements of the CHOET1 decomposition. LBOUND(P,1) should equal LBOUND(AA,1).

BB (*,*)BIN FLOAT(21) contains, in its first N rows and R columns, the matrix B of right-hand sides. LBOUND(BB,1) should equal LBOUND(AA,1).

XX (*,*)BIN FLOAT(21) on return, will contain, in its first N rows and R columns, the matrix of solutions. LBOUND of each dimension of XX should equal LBOUND of the corresponding dimension of BB.

2) DCHOOL1 (N,R,AA,P,BB,XX)
where parameters have the same meaning as with CHOOL1 except for:

AA (*,*)BIN FLOAT(53)

P (*)BIN FLOAT(53)

BB (*,*)BIN FLOAT(53)

XX (*,*)BIN FLOAT(53)

TELECOMPUTING

TUCC MEMORANDUM

March 21, 1973

Library Services Series
Document No. LS-210-0

FROM: NCSU Programming Services

SUBJECT: ORTIN1, for Least Squares Solution of a Set of Linear Equations

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the least squares solution, X, for k systems of n linear equations with m unknowns (m<=n). A is the n by m matrix of the coefficients of the systems. B is the constant n by k matrix of right-hand sides. For the case k=1, more efficient code is provided by ORTIN2.

Iterative improvement is applied to the initial solutions. If the iteration is ineffective, the procedure exits to a user specified label.

This procedure is based on the procedure ORTHOLIN1 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) ORTIN1 (AA,N,M,BB,K,EPS,XX,STOP)
where

AA	(*,*)BIN FLOAT(21)		contains in its first N rows and M columns, the matrix A.
N	BIN FIXED(31)		is the number of equations (n).
M	BIN FIXED(31)		is the number of unknowns (m).
BB	(*,*)BIN FLOAT(21)		contains in its first N rows and K columns the matrix of right-hand sides. LBOUND(BB,1) should equal LBOUND(AA,1).

K	BIN FIXED(31)	is the number of right-hand sides (k).
EPS	BIN FLOAT(21)	is the maximal desired relative rounding error and cannot be less than .953674E-6.
XX	(*,*)BIN FLOAT(21)	will contain in its first M rows and K columns the solutions. LBOUND(XX,1) should equal LBOUND(AA,2). LBOUND(XX,2) should equal LBOUND(BB,2).
STOP	LABEL	specifies the label to which the procedure should transfer if iterative improvement fails.

2) DORTIN1 (AA,N,M,BB,K,EPS,XX,STOP)
where parameters have the same meaning as with ORTIN1 except for:

AA	(*,*)BIN FLOAT(53)	
BB	(*,*)BIN FLOAT(53)	
EPS	BIN FLOAT(53)	cannot be less than .222044604925031E-15
XX	(*,*)BIN FLOAT(53)	

TELECOMPUTING

TUCC MEMORANDUM

March 22, 1973

Library Services Series
Document No. LS-211-0

FROM: NCSU Programming Services

SUBJECT: EIGEN, for Solving the Eigenproblem of a Square Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the eigenproblem for a n by n matrix, A. (For specifically symmetric matrices, other methods are much faster.) When A is symmetric, the algorithm reduces to the (cyclic) Jacobi Method. A series of norm-reducing Jacobi-like similarity transformations are used. The appearance of multiple complex roots will slow convergence considerably; and if these roots are defective, convergence is markedly slow. ("Defective" here means that the multiplicity of the root is less than the number of independent vectors associated with the root). The procedure contains an upper limit of 50 to the number of iterations (complete sweeps).

A complex matrix C=A+iB may be treated by working with the augmented matrix:

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix}$$

as explained in the text referenced below.

On return, the real eigenvalues occupy diagonal elements of A, while the real and imaginary parts of complex eigenvalues $\lambda \pm i\mu$ occupy, respectively, the diagnonal and off-diagonal corners of 2x2 blocks on the main diagonal so that $\lambda = A_{j,j} = A_{j+1,j+1}$ and $A_{j,j+1} = -A_{j+1,j}$ and $abs(A_{j,j+1}) = abs(\mu)$.

The parameter,TMX, is used to control the operation of the procedure and, on return, is used as an error indicator. If on input TMX=0, no eigenvectors are produced. TMX<0, or TMX>0 indicates that, respectively, left or right eigenvectors are to be produced. On return, abs(TMX) gives the number of iterations performed. Convergence will have occurred if 0<TMX<50. Failure

to converge is indicated by TMX=50 or, if all transformations in one iteration are identity matrices, by TMX<0.

The procedure is based on the procedure EIGEN described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reich; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) EIGEN (N,TMX,AA,T)
where

N	BIN FIXED(31)	is the order, n, of the matrix A.
TMX	BIN FIXED(31)	is described above.
AA	(*,*)BIN FLOAT(21)	contains in its first N rows and N columns the matrix for which the eigenproblem is to be solved; on return, contains the eigenvalues as as described above. LBOUND(AA,2) should equal LBOUND(AA,1).
T	(*,*)BIN FLOAT(21)	on return, will contain in its first N rows and N columns the left (or right) transformation matrices depending on the value of TMX. If TMX=0 on call, this can be any array of 2 dimensions, e.g., A.

2) DEIGEN(N,TMX,AA,T)
where parameters have the same meaning as with EIGEN except for:

AA	(*,*)BIN FLOAT(53)	
T	(*,*)BIN FLOAT(53)	

TELECOMPUTING

TUCC MEMORANDUM

March 22, 1973

Library Services Series
Document No. LS-212-0

FROM: NCSU Programming Services

SUBJECT: ORTIN2, for Least Squares Solution of a Single System of Equations

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the least squares solution, X, for a system of n linear equations with m unknowns (m<=n). A is the n by m matrix of the coefficients of the system. B is the constant right-hand side. ORTIN1 is provided for the case of more than one right-hand side.

Iterative improvement is applied to the initial solution. If the iteration is ineffective, the procedure exits to a user specified label.

This procedure is based on the procedure ORTHOLIN2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) ORTIN2 (AA,N,M,BB,EPS,XX,STOP)

where

AA	(*,*)BIN FLOAT(21)	contains in its first N rows and M columns, the matrix A.
N	BIN FIXED(31)	is the number of equations(n).
M	BIN FIXED(31)	is the number of unknowns(m).
BB	(*)BIN FLOAT(21)	contains in its first N elements, the right-hand side. LBOUND(BB,1) should equal LBOUND(AA,1).
EPS	BIN FLOAT(21)	is the maximal desired relative rounding error and cannot be less than .953674E-6.
XX	(*)BIN FLOAT(21)	will contain, in its first M elements, the solution. LBOUND(XX,1) should equal LBOUND(AA,2).
STOP	LABEL	specifies the label to which the procedure should transfer if iterative improvement fails.

2) DORTIN2 (AA,N,M,BB,EPS,XX,STOP)
where parameters have the same meaning as with ORTIN2 except for:

AA	(*,*)BIN FLOAT(53)	
BB	(*)BIN FLOAT(53)	
EPS	BIN FLOAT(53)	cannot be less than .222044604925031E-15.
XX	(*)BIN FLOAT(53)	

TELECOMPUTING

TUCC MEMORANDUM

March 28, 1973

Library Services Series
Document No. LS-213-0

FROM: NCSU Programming Services

SUBJECT: SYMSOL, for Solving Systems of Linear Equations with a Symmetric Positive-Definite Matrix of Coefficients

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the systems of equations defined by AX=B, where A is a real positive-definite symmetric matrix of order n; and B is a n by r real matrix of r right-hand sides.

SYMSOL must be preceded by SYMDET (LS-190) or any similar routine which performs the decomposition (A=LDL') of the coefficient matrix.

AX=B is solved in two steps: LY=B and DL'X=Y. The matrix B is retained to facilitate a later refinement of X. X and B can be the same array in the call of the procedure; however, then B will not be retained.

This procedure is based on the procedure SYMSOL described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer Verlag (1971), where a description of the method can be found.

Implementations are:

1) SYMSOL (N,R,AA,P,BB,XX)

N	BIN FIXED(31)	is the order, n, of the matrix A.
R	BIN FIXED(31)	is the number, r, of right-hand sides, and the number of columns used in B and X.
AA	(*,*)BIN FLOAT(21)	should contain, in its first N rows and N columns, the result of the decomposition performed by SYMDET or a similar procedure. LBOUND (AA,2) should equal LBOUND(AA,1).

P (*)BIN FLOAT(21) should contain in its first N elements the reciprocal diagonal elements of the SYMDET decomposition. LBOUND(P,1) should equal LBOUND(AA,1).

BB (*,*)BIN FLOAT(21) contains, in its first N rows and R columns, the matrix of right-hand sides. LBOUND (BB,1) should equal LBOUND(AA,1). LBOUND(BB,2) should equal LBOUND(AA,1).

XX (*,*)BIN FLOAT(21) on return, will contain, in its first N rows and R columns, the solutions to the R systems of equations. LBOUND(XX,1) should equal LBOUND(AA,1) LBOUND(XX,2) should equal LBOUND(AA,1).

2) DSYMSOL (N,R,AA,P,BB,XX)
where parameters have the same meaning as with SYMSOL except for:

AA (*,*)BIN FLOAT(53)

P (*)BIN FLOAT(53)

BB (*,*)BIN FLOAT(53)

XX (*,*)BIN FLOAT(53)

TELECOMPUTING

TUCC MEMORANDUM

April 1, 1973

Library Services Series
Document No. LS-214-0

FROM: NCSU Programming Services

SUBJECT: ACCRSE, for Computing the Inverse of a Positive Definite Symmetric Matrix, with Iterative Improvement

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the inverse of a positive definite symmetric matrix, A, of order n, by Cholesky decomposition using the procedure CHOON1, followed by iterative improvement of the result. After inversion, the inverse, X, is improved by calculating X=X+Z until the ~~correlation~~ correction, Z, is such that maximum abs (Z_{ij}) is less than 2*EPS times the maximum abs(X_{ij}), where Z=XB and B=I-AX.

If A, or A modified by computational rounding error, is not positive definite, the procedure fails and exits to a user specified label. If the maximum correction at any stage is not less than half that at any previous stage, then the procedure exits to another user specified label.

Only the upper triangle need be correct on input. The upper triangle elements are undisturbed. The inverse is stored in the remainder of A. The reciprocals of its diagonal elements are stored instead of the elements themselves.

This procedure is based on the procedure ACCINVERSE described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) ACCRSE (N,EPS,AA,L,FAIL,ILL)
where

N	BIN FIXED(31)	is the order of the matrix A.
EPS	BIN FLOAT(21)	should be .953674E-6.
AA	(*,*)BIN FLOAT(21)	contains the elements of the matrix to be inverted, stored in the first N rows and N columns of the upper triangle; on return, the elements of the inverse will be found in the lower triangle beginning at row 1+LBOUND(AA,1). Note this requires AA to be dimensioned with a number of rows at least equal to N+1.
L	BIN FIXED(31)	will contain the number of iteration steps.
FAIL	LABEL	specifies the label to which the procedure is to transfer if the matrix is not positive definite.
ILL	LABEL	specifies the label to which the procedure is to transfer if the maximum correction at any stage is not less than half that at the previous stage.

2) DACCRSE (N,EPS,AA,L,FAIL,ILL)
where parameters have the same meaning as with ACCRSE except for:

EPS	BIN FLOAT(53)	should be .2220446049250313E-15.
AA	(*,*)BIN FLOAT(53)	

TELECOMPUTING

TUCC MEMORANDUM

May 2, 1973

Library Services Series
Document No. LS-215-1

FROM: NCSU Programming Services

SUBJECT: BISECT, for Calculating Selected Eigenvalues of a Symmetric Tridiagonal Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which calculates selected eigenvalues of a symmetric tridiagonal matrix, of order n, by the method of bisection. This procedure is beneficial if clusters of eigenvalues are to be expected.

(Unsymmetric tridiagonal matrices A with $f_i=a_{i,i+1}$ and $g_i=a_{i+1,i}$ may be treated by taking b_{i+1} as the elements of the subdiagonal, where $b^2_{i+1}=f_i g_i$, provided $f_i g_i \geq 0$.)

The eigenvalues $\lambda_{m1}, \lambda_{m1+1}, \ldots, \lambda_{m2}$ ($\lambda_{i+1} \geq \lambda_i$) are computed ($0<m1\leq m2\leq n$). m1 must be less than or equal to m2 or no eigenvalues are computed.

A parameter EPS1 controls the duration of bisection iteration in roughly the following way: bisection continues until the upper and lower bounds for the value of an eigenvalue differ by less than EPS1, unless at some earlier stage, the upper and lower bounds differ only in the least significant digits. If EPS1 is specified as non-positive, a suitable small value is used instead. EPS2, North Carolina the procedure, gives an extreme upper bound for the error in any eigenvalue. The following reference contains an expanded discussion of these two parameters.

This procedure is based on the procedure BISECT described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) BISECT (C,B,BETA,N,M1,M2,EPS1,RELFEH,EPS2,Z,X)
where

C	(*)BIN FLOAT(21)	contains the elements of the diagonal stored in its first N elements.
B	(*)BIN FLOAT(21)	contains the elements of the subdiagonal as described above in its first N elements. The first element may be arbitrary but is set to zero by the procedure. LBOUND(B,1) should equal LBOUND(C,1).
BETA	(*)BIN FLOAT(21)	contains the squares of the subdiagonal elements in its first N elements. The first element may be arbitrary but is set to zero by the procedure. LBOUND(BETA,1) should equal LBOUND(C,1).
N	BIN FIXED(31)	contains the number of diagonal elements, n (or the order of matrix A).
(M1,M2)	BIN FIXED(31)	are the indices m1 and m2 as described above.
EPS1	BIN FLOAT(21)	as described above.
RELFEH	BIN FLOAT(21)	should be .953674E-6.
EPS2	BIN FLOAT(21)	output, as described above.
Z	BIN FIXED(31)	will, on return, contain the total number of bisections to find all required eigenvalues.
X	(*)BIN FLOAT(21)	contains, in elements X(M1) through X(M2), the computed eigenvalues.

2) DBISECT (C,B,BETA,N,M1,M2,EPS1,RELFEH,EPS2,Z,X)
where parameters have the same meaning as with BISECT except for:

C	(*)BIN FLOAT(53)	
B	(*)BIN FLOAT(53)	
BETA	(*)BIN FLOAT(53)	
EPS1	BIN FLOAT(53)	
RELFEH	BIN FLOAT(53)	should be .222044604925031E-15.
EPS2	BIN FLOAT(53)	
X	(*)BIN FLOAT(53)	

TELECOMPUTING

TUCC MEMORANDUM

April 3, 1973

Library Services Series
Document No. LS-216-0

FROM: NCSU Programming Services

SUBJECT: BANOL1, for Solving Systems of Equations with a Real Band Matrix of Coefficients

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the systems of equations defined by $(A-\lambda I)X=B$ for X, where A is a real band matrix of order n, B is a n by r real matrix of r right-hand sides, and λ is any number. $(A-\lambda I)$ has been previously factorized by BANET1 (LS-199) or an equivalent procedure. BANOL1 can be used any number of times for different B after one application of BANET1 to $(A-\lambda I)$.

This procedure can be used, along with BANET1, to obtain a complete solution to the system $(A-\lambda I)X=B$ where λ is any number except an eigenvalue of A, or it can be used, along with BANET1, in the inverse iteration process to obtain the eigenvalues (λ) and eigenvectors (X) of A. If the parameter E is specified as zero, a forward and backward substitution is performed. If E is specified as non-zero, only a backward substitution is performed which would be the first step in the inverse iteration.

BANOL2 (LS-217) is preferred to this procedure if A is symmetric.

This procedure is based on the procedure BANSOL1 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) BANOL1(N,M1,M2,E,R,AA,M,INT,BB)
where

N	BIN FIXED(31)	is the order, n, of the matrix A.
M1	BIN FIXED(31)	is the number of subdiagonal bands of A.

M2	BIN FIXED(31)	is the number of superdiagonal bands of A.
E	BIN FIXED(31)	modifier for inverse iteration described above.
R	BIN FIXED(31)	the number, r, of right-hand sides.
AA	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and columns M1+1 through M1+M2+1, the upper triangle of the BANET1 factorization.
M	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and M1 columns, the elements of the lower triangle of the BANET1 factorization. LBOUND(M,1) should equal LBOUND(AA,1).
INT	(*)BIN FIXED(15)	is a vector of at least N elements which contains details of the pivoting interchanges produced by the factorization procedure. LBOUND(INT,1) should equal LBOUND(AA,1).
BB	(*,*)BIN FLOAT(21)	on call, contains in its first N rows and R columns the array of right-hand sides. On return, will contain the R solutions. LBOUND(BB,1) should equal LBOUND(AA,1).

2) DBANOL1(N,M1,M2,E,R,AA,M,INT,BB)
where parameters have the same meaning as with BANOL1 except for:

AA	(*,*)BIN FLOAT(53)
M	(*,*)BIN FLOAT(53)
BB	(*,*)BIN FLOAT(53)

TELECOMPUTING

TUCC MEMORANDUM

April 3, 1973

Library Services Series
Document No. LS-217-0

FROM: NCSU Programming Services

SUBJECT: BANOL2, for Solving Systems of Equations with a Real Band Matrix of Coefficients

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the systems of equations defined by $(A-\lambda I)X=B$ for X, where A is a real band matrix of order n, B is a n by r real matrix of r right-hand sides, and λ is any number. $(A-\lambda I)$ has been previously factorized by BANET2 (LS-206) or an equivalent procedure. BANOL2 can be used any number of times for different B after one application of BANET2 to $(A-\lambda I)$. If BANET1 (LS-199) was used to obtain the factorization, BANOL1 (LS-216) is used instead of this procedure.

This procedure can be used, along with BANET2, to obtain a complete solution to the system $(A-\lambda I)X=B$ where λ is any number except an eigenvalue of A, or it can be used, along with BANET2, in the inverse iteration process to obtain the eigenvalues (λ) and eigenvectors (X) of A. If the parameter E is specified as zero, a forward and backward substitution is performed. If E is specified as non-zero, only a backward substitution is performed which would be the first step in the inverse iteration.

This procedure is preferred to BANOL1 when A is symmetric.

This procedure is based on the procedure BANSOL2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) BANOL2(N,M1,M2,E,R,AA,M,INT,BB)
where

N	BIN FIXED(31)	is the order, n, of the matrix A.
M1	BIN FIXED(31)	is the number of subdiagonal bands of A.

M2 BIN FIXED(31) is the number of superdiagonal bands of A.

E BIN FIXED(31) modifier for inverse iteration described above.

R BIN FIXED(31) the number, r, of right-hand sides.

AA (*,*)BIN FLOAT(21) on call contains, in its first N rows and columns M1+1 through M1+M2+1, the upper triangle of the BANET2 factorization.

M (*,*)BIN FLOAT(21) on call contains, in its first N rows and M1 columns, the elements of the lower triangle of the BANET2 factorization. LBOUND(M,1) should equal LBOUND(AA,1).

INT (*,*)BIT(8) contains in its first N rows and M1 columns, the details of the binary row interchange produced by BANET2 or an equivalent procedure. LBOUND(INT,2) should equal LBOUND(M,2). LBOUND(INT,1) should equal LBOUND(AA,1).

BB (*,*)BIN FLOAT(21) on call, contains, in its first N rows and R columns, the array of right-hand sides; on return, will contain the R solutions. LBOUND(BB,1) should equal LBOUND(AA,1).

2) DBANOL2(N,M1,M2,E,R,AA,M,INT,BB)
where parameters have the same meaning as with BANOL2 except for:

AA (*,*)BIN FLOAT(53)

M (*,*)BIN FLOAT(53)

BB (*,*)BIN FLOAT(53)

TELECOMPUTING

TUCC

MEMORANDUM

April 6, 1973

Library Services Series
Document No. LS-219-0

FROM: NCSU Programming Services

SUBJECT: CHOET2, for Cholesky Decomposition of a Positive Definite Symmetric Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which, given a positive definite symmetric matrix, A, of order n, determines the Cholesky decomposition A=LL'. Only the lower triangle of the matrix to be decomposed is stored, row by row, in a vector of n*(n+1)/2 elements. A 3 by 3 matrix, A, would be stored as $(A_{11},A_{21},A_{22},A_{31},A_{32},A_{33})$.

Elements of L, of the LL' decomposition, replace those of the matrix A, except that the reciprocals of the diagonal elements of L are stored instead of the diagonal elements.

The routine then computes two numbers D1 and D2 from which the determinant can be computed: $\det A = D1*2^{**}D2$ $(1/10 \le |D1| < 1)$.

The procedure will fail if A, or A modified by computational rounding errors, is not positive definite.

This procedure is based on the procedure CHOLDET2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1973), where a description of the method can be found.

Implemented versions are:

1) CHOET2 (N,AA,D1,D2,FAIL)
where

N	BIN FIXED(31)	contains the order, n, of the matrix A.

AA	(*)BIN FLOAT(21)	contains the matrix A in its first N*(N+1)/2 elements. On return, will contain the elements of the decomposition stored in the first N*(N+1)/2 elements as described above.
D1	BIN FLOAT(21)	as described above.
D2	BIN FIXED(31)	as described above.
FAIL	LABEL	specifies the label to which CHOET2 will transfer when A is not positive definite.

2) DCHOET2 (N,AA,D1,D2,FAIL)
where parameters have the same meaning as with CHOET2 except for:

AA	(*)BIN FLOAT(53)	
D1	BIN FLOAT(53)	

Note: For convenience in referring to elements of the "compressed" symmetric matrix, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated by:

```
DCL AA(6),A(3,3)  DEF  AA(2SUB+(1SUB*(1SUB-1))/2);
```

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

July 2, 1973

Library Services Series
Document No. LS-221-0

FROM: NCSU Programming Services

SUBJECT: MINFIT, for Finding Singular Values, Minimal Length Solutions, and Solutions of Homogeneous Linear Equations.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which, given a real m x n matrix A and a real m x p matrix B, computes V,C and diag(q) such that

U'AV=diag(q) and U'B=C

with orthogonal matrices U and V. The elements of q are the "singular values" of A.

These results can be used to determine X minimizing the norm of both AX-B and X. This solution is

X=V*(pseudo-inverse of diag(q))*C, where the ii-th element of the pseudo-inverse of diag(q) is $1/q_{ii}$ for $q_{ii}\neq 0$ and 0 for $q_{ii}=0$.

The procedure can also be used to determine the complete solution of homogeneous equations (in which case p=0).

Householder reduction to bidiagonal form followed by QR iterations are used to produce the singular values and the decomposition.

This procedure is based on the procedure MINFIT described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) MINFIT (M,N,P,EPS,TOL,AB,QQ);
where

M	BIN FIXED(31)	is the number of rows (m) of the array A.
N	BIN FIXED(31)	is the number of columns (n) of the array A.
P	BIN FIXED(31)	is the number of columns (p) of the array B (P≥0).
EPS	BIN FLOAT(21)	a constant used for convergence testing; should not be less than .953674E-6.
TOL	BIN FLOAT(21)	should be .353737E-73.
AB	(*,*)BIN FLOAT(21)	on call, contains in its first M rows and N columns, the elements of A, and in its first M rows and columns N+1 through N+P, the elements of B. On return, the first N rows and N columns of AB will contain the elements of V. The elements of C will be found in columns N+1 through N+P and the first max(N,M) rows. Note that this dual matrix storage requires that AB be dimensioned with at least max(M,N) rows and (N+P) columns.
QQ	(*)BIN FLOAT(21)	will contain the singular values of A in its first N elements; they are non-negative but not necessarily ordered in descending sequence. LBOUND(QQ,1) should equal LBOUND(AB,2).

2) DMINFIT (M,N,P,EPS,TOL,AB,QQ);
where parameters have the same meaning as with MINFIT except for:

EPS	BIN FLOAT(53)	should be .222044604925031E-15.
TOL	BIN FLOAT(53)	should be .35373746401662E-73.
AB	(*,*)BIN FLOAT(53)	
QQ	(*)BIN FLOAT(53)	

NOTE: For convenience in referring to elements of the matrices stored within AB, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated, in which m=3, n=4, and p=1:

```
DCL  AB(4,5)  FLOAT,
     A(3,4)  DEF  AB(1SUB,2SUB),
     B(4,1)  DEF  AB(1SUB,2SUB+N),
     V(4,4)  DEF  AB(1SUB,2SUB),
     C(4,1)  DEF  AB(1SUB,2SUB+N);
```

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

November 2, 1973

Library Services Series
Document No. LS-223-0

FROM: NCSU Programming Services

SUBJECT: CHOOL2, for Solving Systems of Equations with a Positive-Definite Matrix of Coefficients

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which solves the systems of equations defined by AX=B for X, where A is a positive-definite symmetric matrix, of order n, and B is a n by r matrix of r right-hand sides. The procedure CHOOL2 must be preceded by CHOET2 (LS-219) or an equivalent procedure which performs the decomposition (A=LL') of the coefficient matrix. CHOOL2 may be used any number of times, for different B, after one use of CHOET2.

AX=B is solved in two steps, Ly=B and L'X=y. The matrices y, and then X, are overwritten on B;

The procedure is based on the procedure CHOLSOL2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) CHOOL2 (N,R,AA,BB);
where

N	BIN FIXED(31)	is the order, n, of matrix A.
R	BIN FIXED(31)	is the number, r, of right-hand sides.
AA	(*)BIN FLOAT(21)	contains the elements of the lower triangle L as produced by the CHOET2 decomposition of A.
BB	(*,*)BIN FLOAT(21)	contains, in its first N rows and R columns, the matrix B of right-hand sides. LBOUND(BB,1) should equal LBOUND(AA,1).

2) DCHOOL2 (N,R,AA,BB);
where parameters have the same meaning as with CHOOL2 except for:

AA	(*)BIN FLOAT(53)
BB	(*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

November 2, 1973

Library Services Series
Document No. LS-224-0

FROM: NCSU Programming Services

SUBJECT: RATQR, for computation of the lowest eigenvalues of a symmetric tridiagonal matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the m lowest eigenvalues of a symmetric tridiagonal matrix. A rational variant of the QR transformation is used. A quadratic rate of convergence is obtained. The procedure BISECT (LS-215) is preferred if clusters of eigenvalues are expected.

The upper eigenvalues are found if the signs of the diagonal elements are reversed before and after using the procedure. A non-symmetric tridiagonal matrix, T, can be handled, provided T(I-1,I)*T(I,I-1)>=0, by giving the elements of B2, below, as that product.

This procedure is based on the procedure RATQR described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) RATQR (N,M,POSDEF,DLAM,EPS,D,B2)
where

N	BIN FIXED(31,0)	contains the order of the matrix
M	BIN FIXED(31,0)	specifies the number of smallest eigenvalues wanted.
POSDEF	BIT(1)	is given a '1'B if the matrix is known to be positive definite; '0'B, otherwise.
DLAM	BIN FLOAT(21)	specifies a tolerance for the theoretical error of the computed eigenvalues. This error for the k-th value is usually not greater than k*DLAM. DLAM=0 is admissible.

EPS	BIN FLOAT(21)	should be .953674E-6.
D	(*)BIN FLOAT(21)	on call contains, in its first N elements, the diagonal elements of the tri-diagonal matrix. On return contains, in its first M elements, the computed eigenvalues in non-decreasing sequence. The remaining elements are lost.
B2	(*)BIN FLOAT(21)	on call contains, in its second through Nth elements, the squares of the off-diagonal elements of the matrix. The first element is not used and may be of arbitrary value. On return, B2(K) is a bound (not including the effect of roundoff errors) for the theoretical error of the K-th computed eigenvalue, D(K). The remaining elements are lost. LBOUND(B2,1) should equal LBOUND(D,1).

2) DRATQR (N,M,POSDEF,DLAM,EPS,D,B2)
where the parameters have the same meaning as with RATQR, except for:

DLAM	BIN FLOAT(53)	
EPS	BIN FLOAT(53),	should be .222044604925031E-15
D	(*)BIN FLOAT(53)	
B2	(*)BIN FLOAT(53)	

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

November 2, 1973

Library Services Series
Document No. LS-225-0

FROM: NCSU Programming Services

SUBJECT: JACOBI, for Eigenvalues and Eigenvectors of a Real Symmetric Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which will compute the approximate eigenvalues and, optionally, the corresponding normalized vectors of a real symmetric matrix by Jacobi's method. The iteration is designed to operate until no further significant change occurs.

The procedure is based on the procedure JACOBI described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) JACOBI (N,E,A,D,V,ROT)
where

N	BIN FIXED(31)	is the order of the matrix
E	BIT(1)	is '1'B if eigenvectors are to be computed '0'B, if not.
A	(*,*)BIN FLOAT(21)	on call, contains the matrix in its first N rows and N columns. Only the upper triangle need be correct. On return, the superdiagonal elements have been destroyed.
D	(*)BIN FLOAT(21)	on return will contain, in its first N elements, the computed eigenvalues. LBOUND(D,1) must equal LBOUND(A,1).
V	(*,*)BIN FLOAT(21)	if E was specified as '1'B, on return, will contain the computed eigenvectors. The Kth column contains the vector corresponding to D(K). LBOUND(V,1) must equal LBOUND(A,1). If E was specified as '0'B, some two dimensional array must still be specified here, e.g.: A.

ROT BIN FIXED(31) on return, will contain the number of Jacobi rotations required to achieve the solution. One complete sweep over the matrix requires $N^2/2$ rotations.

2) DJACOBI(N,E,A,D,V,ROT)
where the parameters have the same meaning as with JACOBI, except for:

A (*,*)BIN FLOAT(53)

D (*)BIN FLOAT(53)

V (*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

December 4, 1973

Library Services Series
Document No. LS-229-0

FROM: NCSU Programming Services

SUBJECT: SYMRAY, for Computing the Eigenvectors of a Symmetric Band Matrix, by Iteration

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes by inverse iteration the r eigenvectors of a symmetric band matrix, A, corresponding to a given set of r approximate eigenvalues.

The symmetric band matrix, A, is of order n with m subdiagonal and superdiagonal bands. The elements of A are stored as a n x (2m+1) matrix, in the array AA. The scheme of storage is illustrated by the case m=1, n=4;

	first column		(2m+1)th column
first row →	0	a_{11}	a_{12}
	a_{21}	a_{22}	a_{23}
	a_{32}	a_{33}	a_{34}
nth row →	a_{43}	a_{44}	0

Note that the zeros shown are required and that the diagonal elements occupy column (m+1) in the array.

The procedure will fail if A-lambda*I is singular, or becomes so due to computational rounding errors.

The procedure is based on the procedure SYMRAY described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) SYMRAY (N,M,R,MACHEPS,AA,LA,LAMBDA1,L,C,Z,FAIL)
where

N	BIN FIXED(31)	gives the order, n, of the matrix A.
M	BIN FIXED(31)	gives the number, m, of subdiagonal and superdiagonal bands, as described above.

R	BIN FIXED(31)	contains the number, r, of given eigenvalues.
MACHEPS	BIN FLOAT(21)	should be .953674E-6.
AA	(*,*)BIN FLOAT(21)	on call, contains in its first N rows and (2M+1) columns, the elements of the symmetric band matrix A, as described above. LBOUND(AA,2) should equal LBOUND(AA,1).
LA	BIN FIXED(31)	contains the limit of the number of iterations permitted in the determination of any eigenvector.
LAMBDA1	(*)BIN FLOAT(21)	contains in its first R elements the given approximate eigenvalues of A. LBOUND(LAMBDA1,1) should equal LBOUND(Z,2).
L	(*)BIN FIXED(31)	on return, the first R elements are such that the ith element of L represents the number of iterations that the ith eigenvector's computation required. LBOUND(L,1) should equal LBOUND(Z,2).
C	(*)BIN FIXED(31)	on return, the first R elements are such that the ith element of C represents the number of eigenvalues greater than the ith element of LAMBDA1. LBOUND(C,1) should equal LBOUND(Z,2).
Z	(*,*)BIN FLOAT(21)	on return, contains in its first N rows and R columns, the R eigenvectors of A corresponding to the eigenvalues in LAMBDA1. LBOUND(Z,1) should equal LBOUND(AA,1).
FAIL	LABEL	is a user specified label to which SYMRAY will transfer if the procedure fails.

2) DSYMRAY (N,M,R,MACHEPS,AA,LA,LAMBDA1,L,C,Z,FAIL)
where the parameters have the same meaning as with SYMRAY except for:

MACHEPS	BIN FLOAT(53)	should be .222044604925031E-15
AA	(*,*)BIN FLOAT(53)	
LAMBDA1	(*)BIN FLOAT(53)	
Z	(*,*)BIN FLOAT(53)	

Note: For convenience in referring to elements of the "compressed" band matrices used by this procedure, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated, for the above example, by:

```
DCL  AA(4,-1:1) FLOAT,
     BANDM(4,4) DEF AA(1SUB,2SUB-1SUB);
```

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

December 4, 1973

Library Services Series
Document No. LS-230-0

FROM: NCSU Programming Services

SUBJECT: PERC, to Compute a New Inverse after Perturbation of a Column in the Original Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

Given R, the inverse of A, and a vector X, this PL/I subprocedure computes the inverse of $A + Xe_k$ (the matrix A after perturbation of its k-th column by the vector X).

The calculation of the new inverse requires many fewer operations than recalculation of the inverse directly from the perturbed matrix.

The procedure will fail if the k-th row of R times X equals -1, which will occur if the perturbation would make A computationally, or in fact, singular.

The procedure is based on an algorithm (equation 3.26) given in Matrix Calculus; Bodewig; North-Holland Publishing (1956).

Implemented version is:

1) PERC (N,RR,K,XX,FAIL)

where

N	BIN FIXED(31,0)	contains the order of the matrix R (and A).
RR	(*,*)BIN FLOAT (21)	on call contains in its first N rows and N columns, the inverse, R. On return, this is replaced by the new inverse. LBOUND (RR,2) should equal LBOUND (RR,1).
K	BIN FIXED (31,0)	contains, k, the index of the column of A to be perturbed.
XX	(*)BIN FLOAT(21)	contains the vector by which the matrix A is to be perturbed. LBOUND (XX,1) should equal LBOUND (RR,1).
FAIL	LABEL	contains a label in the calling procedure to which PERC will transfer if the procedure fails as described above. In this event, RR is unchanged.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

May 30, 1974

Library Services Series
Document No. LS-244-0

FROM: NCSU Programming Services

SUBJECT: TRED1, for Reduction of a Symmetric Matrix to Tridiagonal Form

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming

This is a PL/I subprocedure which uses Householder's reduction to reduce a symmetric matrix to tridiagonal form

This is a useful transformation prior to the computation of roots and vectors by some algorithms. If the vectors of a matrix reduced by this procedure have been found, TRBAK1 (LS-245) can be used to obtain the vectors of the original matrix.

TRED1 should be used if only selected eigenvalues and corresponding vectors are to be determined by BISECT (LS-215) or TRIURM (LS-255), for example. It can be used with any procedure for finding the eigenvalues of a symmetric tridiagonal matrix (for example: LS-256, TQL1; LS-224, RATQR). If multiple or close eigenvectors are suspected, TRED3 (LS-247) is preferred.

If the original matrix has elements of widely varying orders of magnitude, it is important that the smaller elements be in the upper left corner (and larger elements in the lower right corner) to reduce roundoff in this method.

This procedure is based on the procedure TRED1 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) TRED1 (N,TOL,A,D,E,E2)

where

N	BIN FIXED(31,0)	contains the order of the matrix to be reduced
TOL	BIN FLOAT(21)	should be 565980E-72.
A	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and N columns, the matrix to be reduced. Only the lower triangular elements are used. On return, the strict lower triangle will contain details on the transformation which will be needed by TRBAK1. The upper triangle is undisturbed.
D	(*)BIN FLOAT(21)	on return will contain, in its first N elements, the diagonal elements of the reduced matrix. LBOUND(D,1) should equal LBOUND(A,1).
E	(*)BIN FLOAT(21)	on return will contain, in its second through Nth elements, the N-1 subdiagonal elements of the reduced matrix. The first elements of E is set to zero. LBOUND(E,1) should equal LBOUND(A,1).
E2	(*)BIN FLOAT(21)	on return, E2(K)=E(K)**2. These squares are used by procedure BISECT. If squares are not desired, E2 may be the same name as E. LBOUND(E2,1) should equal LBOUND(A,1).

2) DTRED1 (N,TOL,A,D,E,E2)

where the parameters have the same meaning as with TRED1, except for;

TOL	BIN FLOAT(53) should be .24308653429E-62.
A	(*,*)BIN FLOAT(53)
D	(*)BIN FLOAT(53)
E	(*)BIN FLOAT(53)
E2	(*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

June 3, 1974

Library Services Series
Document No. LS-245-0

FROM: NCSU Programming Services

SUBJECT: TRBAK1, for Obtaining Original Eigenvectors from Those of a Derived Tridiagonal Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

After finding the eigenvectors of a tridiagonal matrix which was obtained by Householder reduction of a symmetric matrix by procedure TRED1 (LS-244) or similar procedure, this PL/I subprocedure can be used to obtain the vectors of the original matrix.

This procedure is based on the procedure TRBAK1 in the Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) TRBAK1 (N,M1,M2,A,E,Z)

where

N	BIN FIXED(31,0)	gives the order of the symmetric matrix.
(M1,M2)	BIN FIXED(31,0)	give two values, m1 and m2, indicating that eigenvectors m1 thru m2 of the derived tridiagonal matrix have been found.
A	(*,*)BIN FLOAT(21)	contains in the subdiagonal elements of its first N rows and N columns, the details of the Householder reduction as produced by TRED1. LBOUND(A,1) should equal LBOUND(A,2).
E	(*)BIN FLOAT(21)	contains, in its second through Nth elements, the N-1 off-diagonal elements of the tridiagonal matrix, with its first element equal to zero, as produced by TRED1. LBOUND(E,1) should equal LBOUND(A,1).
Z	(*,*)BIN FLOAT(21)	on call contains, in its first M2-M1+1 columns, eigenvectors M1 thru M2 of the tridiagonal matrix normalized according to the Euclidean norm. The vectors occupy the first N elements of each column. On return, these vectors are replaced by the original problem. LBOUND(Z,1) should equal LBOUND(A,1).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

June 4, 1974

Library Services Series
Document No. LS-246-0

FROM: NCSU Programming Services

SUBJECT: TRED2 for Reduction of a Symmetric Matrix to Tridiagonal Form

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which uses Householder's reduction to reduce a symmetric matrix to tridiagonal form.

This is a useful transformation prior to the computation of roots and vectors by some algorithms. This procedure should never be used if vectors are not required. Its use is mandatory preceding use of TQL2 (LS-257) or IMTQL2 (LS-259), for general symmetric matrices.

If the original matrix has elements of widely varying orders of magnitude, it is important that the smaller elements be in the upper left corner (and larger elements, in the lower right corner) to reduce roundoff in this method.

This procedure is based on the procedure TRED2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) TRED2 (N,TOL,A,D,E,Z)

where

N	BIN FIXED(31,0)	contains the order of the matrix to be reduced.
TOL	BIN FLOAT(21)	should be .565980E-72.
A	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and N columns, the matrix to be reduced. Only the lower triangle is used in the computations. This name can be the same name as Z below, in which case, on return, the values in A are destroyed; otherwise the values in A are undisturbed. LBOUND(A,2) should equal LBOUND(A,1).

D (*)BIN FLOAT(21) on return contains, in its first N elements, the diagonal elements of the tridiagonal matrix. LBOUND(D,1) should equal LBOUND(A,1).

E (*)BIN FLOAT(21) on return contains, in its second through Nth elements, the N-1 elements of the subdiagonal of the tridiagonal matrix. The first element is set to zero. LBOUND(E,1) should equal LBOUND(A,1).

Z (*,*)BIN FLOAT(21) on return contains, in its first N rows and N columns, the orthogonal matrix which is the product of the Householder transformation matrices. Z can be specified as the same name as A, in which case the original matrix is overwritten. LBOUND(Z,1) and LBOUND(Z,2) should equal LBOUND(A,1) and LBOUND(A,2) respectively. This matrix will be required by TQL2, or IMTQL2.

2) DTRED2 (N,TOL,A,D,E,Z)

where the parameters have the same meaning as with TRED2, except for:

TOL BIN FLOAT(53) should be .24308653429E-62.

A (*,*)BIN FLOAT(53)

D (*)BIN FLOAT(53)

E (*)BIN FLOAT(53)

Z (*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

July 11, 1974

Library Services Series
Document No. LS-247-0

FROM: NCSU Programming Services

SUBJECT: TRED3 for Reduction of a Symmetric Matrix to Tridiagonal Form

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which uses Householder's reduction to reduce a symmetric matrix to tridiagonal form. The symmetric matrix is presented in a compressed form.

TRED1 (LS-244) offers the same function for the matrix stored in general form.

This is a useful transformation prior to the computation of roots and vectors by some algorithms. If the vectors of a matrix reduced by this procedure have been found, TRBAK3 (LS-248) can be used to obtain the vectors of the original matrix. If multiple or close roots are suspected, TRED3 is preferred over TRED1.

If the original matrix has elements of widely varying orders of magnitude, it is important that the smaller elements be in the upper left corner (and larger elements in the lower right corner) to reduce roundoff in this method.

This procedure is based on the procedure TRED3 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) TRED3 (N,TOL,A,D,E,E2)
where

N	BIN FIXED(31,0)	contains the order of the matrix to be reduced.
TOL	BIN FLOAT(21)	should be .565980E-72.

A	(*)BIN FLOAT(21) on call contains, in its first N*(N+1)/2 elements, the elements of the lower triangle of the matrix to be reduced stored in the order a_{11}, a_{21}, a_{22}, a_{31}, a_{32}, a_{33}, etc. On return, the matrix has been replaced with details on the transformation.
D	(*)BIN FLOAT(21) on return will contain, in its first N elements, the diagonal elements of the reduced matrix. LBOUND(D,1) should equal LBOUND(A,1).
E	(*)BIN FLOAT(21) on return will contain, in its second through Nth elements, its N-1 subdiagonal elements of the reduced matrix. The first elements of E are set to zero. LBOUND(E,1) should equal LBOUND(A,1).
E2	(*)BIN FLOAT(21) on return, E2(K)**2. These squares are used by procedure BISECT (LS-215). If squares are not desired, E2 may be the same name as E. LBOUND(E,1) should equal LBOUND(A,1).

2) DTRED3 (N,TOL,A,D,E,E2)
where the parameters have the same meaning as with TRED3, except for:

TOL	BIN FLOAT(53) should be .24308653429E-62.
A	(*)BIN FLOAT(53)
D	(*)BIN FLOAT(53)
E	(*)BIN FLOAT(53)
E2	(*)BIN FLOAT(53)

Note: For convenience in referring to elements of the "compressed" symmetric matrix, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated by:

```
DCL AA(6),A(3,3) DEF  AA (2SUB+(1SUB*(1SUB-1))/2);
```

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

July 18, 1974

Library Services Series
Document No. LS-248-0

FROM: NCSU Programming Services

SUBJECT: TRBAK3 for Obtaining Original Eigenvectors from those of a Derived Tridiagonal Matrix.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

After finding the eigenvectors of a tridiagonal matrix which was obtained by Householder reduction of a symmetric matrix by procedure TRED3 (LS-247) or similar procedure, this PL/I subprocedure can be used to obtain the vectors of the original matrix.

This procedure is based on the procedure TRBAK3 in the Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) TRBAK3 (N,M1,M2,A,Z)
where

N	BIN FIXED(31,0)	gives the order of the symmetric matrix.
(M1,M2)	BIN FIXED(31,0)	give two values, m1 and m2, indicating that eigenvectors m1 thru m2 of the derived tridiagonal matrix have been found.
A	(*)BIN FLOAT(21)	contains in its first (N*(N+1))/2 elements, details of the Householder reduction, as produced by TRED3.
Z	(*,*)BIN FLOAT(21)	on call contains, in its first M2-M1+1 columns, eigenvectors M1 and M2 of the tridiagonal matrix, normalized according to the Euclidean norm. The vectors occupy the first N elements of each column. On return, these vectors are replaced by the vectors of the original problem. LBOUND(Z,1) should equal LBOUND(A,1).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

July 18, 1974

Library Services Series
Document No. LS-249-0

FROM: NCSU Programming Services

SUBJECT: REDUC1, for Reducing a General Symmetric Eigenproblem to the Standard Symmetric Eigenproblem.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure for reducing the problem $Ax=\lambda Bx$ to the standard symmetric eigenproblem $Pz=\lambda z$, where $B=LL'$ and $P=L^{-1}A(L')^{-1}$ and A is symmetric.

The derived system can be tridiagonalized by using procedure TRED1 (LS-244).

If the eigenvectors of the derived system have been found, the eigenvectors of the original system can be found by use of procedure REBAKA (LS-250).

The procedure will fail if B, or B modified by computational rounding errors, is not positive definite.

If in calling the procedure, N, below, is given as a negative value, REDUC1 assumes that the decomposition B=LL' has already been performed and elements of L are in place in BB and DL as described below.

The procedure is based on the procedure REDUC1 in the Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) REDUC1 (N,AA,BB,DL,FAIL)
where

N	BIN FIXED(31,0)	gives the order of the matrices A, B, and P.
AA	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and N columns, the elements of A. Only the upper triangle elements need be correct. On return, it contains the lower triangle of P. The strict upper triangle of A is preserved but its diagonal is destroyed. LBOUND(AA,1) should equal LBOUND(AA,2).

BB	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and N columns, the elements of B Only the upper triangle elements need be correct. On return, it contains the subdiagonal elements of the matrix L stored in its strict lower triangle The upper triangle of B is undisturbed. LBOUND(BB,1) should equal LBOUND(BB,2) and LBOUND(AA,1).
DL	(*)BIN FLOAT(21)	on return contains, in its first N elements, the diagonal elements of L. LBOUND(DL,1) should equal LBOUND(AA,1).
FAIL	LABEL	is a label in the calling program to which REDUCE will transfer if the procedure fails.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

July 18, 1974

Library Services Series
Document No. LS-250-0

FROM: NCSU Programming Services

SUBJECT: REBAKA, for Obtaining Eigenvectors of Some General Eigenproblems from Vectors of Derived Standard Problems.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the eigenvectors, x, of the symmetric problems $Ax=\lambda Bx$, $ABx=\lambda x$ and $x'BA=\lambda x'$ from the corresponding vectors $z=L'x$ of the derived standard symmetric problem. It is assumed that the derived problem was obtained by REDUC1 (LS-249), REDUC2 (LS-251) or some similar procedure.

The procedure is based on the procedure REBAKA in the Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) REBAKA (N,M1, M2,LL,DL,Z)
where

N	BIN FIXED(31,0)	gives the order of A and B
(M1,M2)	BIN FIXED(31,0)	contain values m1 and m2 indicating that vectors m1 thru m2 of the derived problem have been found.
LL	(*,*)BIN FLOAT(21)	contains, in the strict lower triangle of its first N rows and N columns, the subdiagonal elements of L above (B=LL'). REDUC1 and REDUC2 leave L in this form. LBOUND(LL,1) should equal LBOUND(LL,2).
DL	(*)BIN FLOAT(21)	contains, in its first N elements, the diagonal elements of L. REDUC1 and REDUC2 leave diag(L) in this form. LBOUND(DL,1) should equal LBOUND(LL,1).
Z	(*,*)BIN FLOAT(21)	on call contains, in its first M2-M1+1 columns, eigenvectors M1 thru M2 of the derived problem. The vectors occupy the first N elements of each column. On return, these vectors are replaced by the vectors of the original problem. LBOUND(Z,1) should equal LBOUND(LL,1).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

July 18, 1974

Library Services Series
Document No. LS-251-0

FROM: NCSU Programming Services

SUBJECT: REDUC2, for Reducing Some Symmetric Eigenproblems to the Standard Symmetric Eigenproblems.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure for reducing the problems

$y'AB=\lambda y'$ $\quad ABx=\lambda x$

$BAy=\lambda y$ $\quad x'BA=\lambda x'$

to the standard symmetric eigenproblem $Qz=\lambda z$, where A is symmetric, $Q=L'AL$, and $B=LL'$.

The derived system can be tridiagonalized by using procedure TRED1 (LS-244).

If the eigenvectors of the derived system have been found, the eigenvectors of the original system can be found by use of procedure REBAKB (LS-252) in the case of the left two above problems, and by procedure REBAKA (LS-250) in the case of the right two.

The procedure will fail if B, or B modified by computational rounding errors, is not positive definite.

If in calling the procedure, N, below, is given as a negative value, REDUC2 assumes that the decomposition B=LL' has already been performed and elements of L are in place in BB and DL as described below.

This procedure is based on the procedure REDUC2 in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) REDUC2 (N,AA,BB,DL,FAIL)
where

N	BIN FIXED(31,0)	gives the order of A, B, and Q.
AA	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and N columns, the elements of A. Only the upper triangle need be correct. On return,

the lower triangle of A is replaced by the lower triangle of Q. The strict upper triangle of A is undisturbed but its diagonal is lost. LBOUND(AA,1) should equal LBOUND(AA,2).

BB (*,*)BIN FLOAT(21) on call contains, in its first N rows and N columns, the elements of B. Only the upper triangle need be correct. On return, the strict lower triangle of B is replaced by the subdiagonal elements of L. The upper triangle of B is undisturbed. LBOUND(BB,1) should equal LBOUND(BB,2) and LBOUND(AA,1).

DL (*)BIN FLOAT(21) on return contains, in its first N elements, the diagonal elements of L. LBOUND(DL,1) should equal LBOUND(AA,1).

FAIL LABEL is a label in the calling program to which REDUC2 will transfer if the procedure fails.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

July 18, 1974

Library Services Series
Document No. LS-252-0

FROM: NCSU Programming Services

SUBJECT: REBAKB, for Obtaining Eigenvectors of Some General Eigenproblems from Vectors of Derived Standard Problems.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the eigenvectors, x, of the symmetric problems $y'AB=\lambda y'$, $BAy=\lambda y$ from the corresponding vectors $z=L^{-1}y$ of the derived standard symmetric problem. It is assumed that the derived problem was obtained by REDUC2 (LS-251) or some similar procedure.

The procedure is based on the procedure REBAKB in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) REBAKB (N,M1,M2,LL,DL,Z)
where

N	BIN FIXED(31,0)	gives the order of A and B
(M1,M2)	BIN FIXED(31,0)	contain values m1 and m2 indicating that vectors m1 thru m2 of the derived problem have been found.
LL	(*,*)BIN FLOAT(21)	contains, in the strict lower triangle of its first N rows and N columns, the subdiagonal elements of L above (B=LL'). REDUC2 leaves L in this form. LBOUND(LL,1) should equal LBOUND(LL,2).
DL	(*)BIN FLOAT(21)	contains, in its first N elements, the diagonal elements of L. REDUC2 leaves diag(L) in this form. LBOUND(DL,1) should equal LBOUND(LL,1).
Z	(*,*)BIN FLOAT(21)	on call contains in its first M2-M1+1 columns, eigenvectors M1 thru M2 of the derived problem. The vectors occupy the first N elements of each column. On return, these vectors are replaced by the vectors of the original problem. LBOUND(Z,1) should equal LBOUND(LL,1).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

August 21, 1974

Library Services Series
Document No. LS-253-0

FROM: NCSU Programming Services

SUBJECT: BANDRD, for Reducing a Real Symmetric Band Matrix to Tridiagonal Form.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which transforms a real symmetric band matrix to tridiagonal form by a sequence of Jacobi rotations. The matrix obtained is J=V'AV where A is the original matrix. V can be optionally obtained as a result.

The eigenvalues of the derived matrix can be found by TQL1 (LS-256), IMTQL1 (LS-258), BISECT (LS-215) or RATQR (LS-224).

The matrix A is presented in the compressed form illustrated by:

	first column		third column
first row	a_{11}	a_{12}	a_{13}
	a_{22}	a_{23}	a_{24}
	a_{33}	a_{34}	x
4th row	a_{44}	x	x

for an order 4 matrix with two bands above (and below) the diagonal. The x's represent irrelevant values not used by the procedure.

The triangular matrix replaces the diagonal and first superdiagonal band of the matrix A (first and second columns of the above). Other elements of the returned matrix are irrelevant.

The procedure is based on the procedure BANDRD in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) BANDRD (N,M,MATV,A,V)
where

N BIN FIXED(31,0) contains the order of the matrix.

M BIN FIXED(31,0) contains the number of bands above (and below) the diagonal (>1).

MATV BIT(1) contains '1'B if the transformation matrix V is to be computed; '0'B, if not.

A (*,*)BIN FLOAT(21) on call contains, in its first N rows and M+1 columns the band matrix stored as described above. On return, contains the tridiagonal matrix, as described above.

V (*,*)BIN FLOAT(21) If MATV='0'B, this matrix is unused and can be any two dimensional matrix, e.g.: A. If MATV='1'B, on return this will contain the computed orthogonal transformation matrix in its first N rows and N columns. LBOUND(V,1) and LBOUND(V,2) should equal LBOUND(A,1) if MATV='1'B.

2) DBANDRD (N,M,MATV,A,V)
where the parameters have the same meaning as with BANDRD, except for:

A (*,*)BIN FLOAT(53)

V (*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

August 21, 1974

Library Services Series
Document No. LS-254-0

FROM: NCSU Programming Services

SUBJECT: BQR, for Finding an Eigenvalue of a Symmetric Band Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes an eigenvalue of A+tI, where A is a symmetric band matrix and t is a user supplied parameter. Usually the eigenvalue found is the one nearest t but this is not guaranteed. The QR algorithm with shifts of origin is used. More than one eigenvalue can be found by repeated uses of BQR, but if all values are needed, BANDRD (LS-253) is preferred.

To compute a smallest eigenvalue of B, provide the procedure with t and A=B-tI.

The procedure returns a reduced matrix and a modified value of the parameter t which can be used in repeated uses of the procedure to find additional eigenvalues. A value of zero can be initially specified for t.

If it is necessary to guarantee the order of the values, this can be done with BANET2 (LS-206). Vectors can be found by inverse iteration (see SYMRAY, LS-229).

Only the upper triangle and diagonal of A are supplied, with elements stored in the array AA, as illustrated for an order four matrix with 2 bands above (and below) the diagonal (m=2):

	first column		(m+1)st column
first row →	a_{11}	a_{12}	a_{13}
	a_{22}	a_{23}	a_{24}
	a_{33}	a_{34}	0
nth row →	a_{44}	0	0

The procedure will fail if more than 30 QR transformations are required to isolate a value.

The procedure is based on the procedure BQR described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) BQR (N,M,TOL,AA,T,R,FAIL)
where

N	BIN FIXED(31,0)	contains the order of the matrix
M	BIN FIXED(31,0)	contains the number of bands above (or below) the diagonal.
TOL	BIN FLOAT(21)	should be .565980E-72.
AA	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and M+1 columns, the band matrix stored as illustrated above. On return, contains the matrix reduced as a result of removing the found root. This matrix can be used in another use of the procedure to find another eigenvalue. Note that the user must reduce N by one before a second use.
T	BIN FLOAT(21)	on call, contains the value of the parameter t. On return, contains the computed eigenvalue unless the failure exit was taken. The returned value is used as input in the next use of BQR, if the next required eigenvalue is the one nearest to the currently computed one.
R	BIN FLOAT(21)	is given as zero in the first of a series of repeated uses. On return it will be MAX(S,R) where S=sum-norm of the last column of the input matrix. This returned value is used as input in the next use of BQR. It is used to determine when the last row and column of the current transformed matrix can be regarded as negligible.
FAIL	LABEL	is a label in the user program to which BQR is to transfer if the process fails.

2) DBQR (N,M,TOL,AA,T,R,FAIL)
where the parameters have the same meaning as with BQR except for:

TOL	BIN FLOAT(53)	should be .24308653429E-62
AA	(*,*)BIN FLOAT(53)	
T	BIN FLOAT(53)	
R	BIN FLOAT(53)	

Note: For convenience in referring to elements of the "compressed" band matrix AA, the user might wish to use iSUB defining. In this case, the iSUB algorithm can be illustrated, for the above example, by:

```
DCL   AA (4,0:2),
      UP_TRI(4,4) DEF AA(1SUB,2SUB-1SUB);
```

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

August 21, 1974

Library Services Series
Document No. LS-255-0

FROM: NCSU Programming Services

SUBJECT: TRIURM, for Selected Eigenvalues of a Symmetric Tridiagonal Matrix, and the Corresponding Eigenvectors

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which finds the eigenvalues of a symmetric tridiagonal matrix lying between two given values, and the corresponding eigenvectors. When the number of values to be found is small, say less than 25% of the total, this is perhaps the most efficient algorithm. If a larger number of values are to be found, TQL2 (LS-257) or IMTQL2 (LS-259) are more efficient.

A full symmetric matrix can be reduced to tridiagonal form for use with TRIURM by TRED1 (LS-244), TRED3 (LS-247) or a similar procedure.

The vectors found by this procedure are Euclidean normalized.

The method of bisection is used to find the eigenvalues and inverse iteration is used to find the eigenvectors.

A call of the procedure with a negative value for argument M below will cause the procedure to return through the failure exit after setting M to the number of eigenvalues in the specified interval, but not performing computation of the values.

This procedure is based on the procedure TRISTURM in the Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) TRIURM (N,LB,UB,MACHEPS,EPS1,C,B,BETA,M,ROOT,VEC,COUNT,FAIL)
where

N	BIN FIXED(31,0)	gives the order of the tridiagonal matrix.
(LB,UB)	BIN FLOAT(21)	all eigenvalues between LB and UB are to be computed.

MACHEPS BIN FLOAT(21) should be .953675E-6

EPS1 BIN FLOAT(21) gives the error to be tolerated in the eigenvalues of smallest modules. If EPS1 is not positive, the quantity MACHEPS*(ABS(LB)+ABS(UB)) is used so that on entry EPS1=0 is admissible.

C (*)BIN FLOAT(21) contains, in its first N elements, the diagonal of the tridiagonal matrix.

B (*)BIN FLOAT(21) on call contains, in its second thru Nth elements, the subdiagonal elements of the matrix with its first element equal to zero. On return, negligible elements have been replaced with zero. LBOUND(B,1) should equal LBOUND(C,1).

BETA (*)BIN FLOAT(21) on call BETA(I)=B(I)**2 and on return these elements are replaced by the squares of the returned Bs. LBOUND(BETA,1) should equal LBOUND(C,1).

M BIN FIXED(31,0) on call, contains an estimate of the maximum number of eigenvalues in the interval. The procedure will fail if this turns out to be less than the actual number in the interval. On return, this will contain the number of values in the interval, thus a call with M=-1 will determine the correct number.

ROOT (*)BIN FLOAT(21) on normal return, will contains, in its first M elements, the M eigenvalues, not always in ascending sequence. If input M is less than returned M, this vector is unused.

VEC (*,*)BIN FLOAT(21) on normal return, will contain in its first M columns the eigenvectors corresponding to the computed eigenvalues. The vectors occupy the first N elements of each column. LBOUND(VEC,2) should equal LBOUND(ROOT,1). LBOUND(VEC,1) should equal LBOUND(C,1). If input M is less than returned M, VEC is unused.

COUNT (*)BIN FIXED(15,0) on normal return contains in its ith element the number of iterations required to find the ith eigenvector. LBOUND(COUNT,1) should equal LBOUND(ROOT,1). If input M is less than returned M, COUNT is unused.

FAIL LABEL gives a label in the user program to which TRIURM will transfer if the procedure fails: when input M is less than returned M, or when more than 5 iterations are required for any eigenvector. In the latter case, the corresponding component of COUNT is set to 6. All M values and the preceding vectors are correct.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

August 21, 1974

Library Services Series
Document No. LS-256-0

FROM: NCSU Programming Services

SUBJECT: TQL1, for Computing All the Eigenvalues of a Real Symmetric Tridiagonal Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the eigenvalues of a real symmetric tridiagonal matrix, using explicit QL transformations.

The procedure will fail if more than 30 iterations are needed to isolate any one eigenvalue.

If the matrix has elements of widely varying orders of magnitude, to minimize the effects of round-off, it is important that the smaller elements be in the upper left corner (and larger elements, in the lower right corner).

This procedure may be preceded by TRED1 (LS-244) which reduces a general symmetric matrix to tridiagonal form. If the above ordering condition is satisfied for TRED1, it will be satisfied for TQL1 in the reduced matrix. If this is not the condition, IMTQL1 (LS-258) is preferred over TQL1.

This procedure is based on the procedure TQL1 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) TQL1 (N,MACHEPS,D,E,FAIL)
where

N	BIN FIXED(31)	contains the order of the given tridiagonal matrix.
MACHEPS	BIN FLOAT(21)	should be .953675E-6.
D	(*)BIN FLOAT(21)	on call, contains in its first N elements, the diagonal elements of the tridiagonal matrix; on return, contains the N eigenvalues in ascending order of magnitude.

E (*)BIN FLOAT(21) on call, contains in its 2nd through Nth elements, the subdiagonal elements of the tridiagonal matrix. The first element of E is not used and may be arbitrary. On return, the original information has been destroyed. LBOUND(E,1) should equal LBOUND(D,1).

FAIL LABEL is a label in the user program to which the procedure will transfer if the process fails, as described above.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

August 21, 1974

Library Services Series
Document No. LS-257-0

FROM: NCSU Programming Services

SUBJECT: TQL2, for Computing All Eigenvalues and Eigenvectors of a Symmetric Tridiagonal Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the eigenvalues and vectors of a real symmetric tridiagonal matrix, T, using explicit QL transformations. If the matrix T was obtained by Householder reduction, using TRED2 (LS-246), of a symmetric matrix A, then TQL2 will find the vectors of A directly without first finding the vectors of T Computed vectors are orthonormal almost to working accuracy.

If the matrix has elements of widely varying orders of magnitude, to minimize the effects of roundoff, it is important that the smaller elements be in the upper left corner (and larger elements, in the lower right corner). If this condition was satisfied for TRED2, it will be satisfied for this procedure. If this is not the condition, IMTQL2 (LS-259) is preferred over TQL2.

This procedure may be used in combination with REDUC1 (LS-249) or REDUC2 (LS-251) for solving the generalized eigenproblem.

The procedure is based on the procedure TQL2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) TQL2 (N,MACHEPS,D,E,Z,FAIL)
where

N	BIN FIXED(31,0)	contains the order of the matrix.
MACHEPS	BIN FLOAT(21)	should be .953675E-6.
D	(*)BIN FLOAT(21)	on call contains, in its first N elements, the diagonal elements. On return, they are replaced by the computed eigenvalues, in ascending order.

E (*)BIN FLOAT(21) on call contains, in its second through Nth elements, the N-1 elements of the subdiagonal of T. The first element of E is not used and can be of arbitrary value. On return, these elements will be lost. LBOUND(E,1) should equal LBOUND(D,1).

Z (*,*)BIN FLOAT(21) on call, the first N rows and N columns, will contain either a) the identify matrix, if the vectors of T itself are desired or b) if T was derived from a full symmetric matrix by TRED2, and it is the vectors of that matrix that are desired: the matrix Z which is output by TRED2. On return, the columns of Z will contain the appropriate normalized eigenvectors. LBOUND(Z,1) should equal LBOUND(Z,2) and LBOUND(D,1).

FAIL LABEL is a label in the user program to which the procedure will transfer if the more than 30 iterations are required to isolate any single eigenvalue.

2) DTQL2 (N,MACHEPS,D,E,Z,FAIL)
where the parameters have the same meaning as with TQL2, except for:

MACHEPS BIN FLOAT(53) should be .2220446049250314E-15.

D (*)BIN FLOAT(53)

E (*)BIN FLOAT(53)

Z (*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

August 21, 1974

Library Services Series
Document No. LS-258-0

FROM: NCSU Programming Services

SUBJECT: IMTQL1, for Computing All Eigenvalues of a Symmetric Tridiagonal Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the eigenvalues of a real symmetric tridiagonal matrix using implicit QL transformations. This algorithm is advantageous over the TQL1 (LS-256) algorithm in that ordering of the matrix elements is not vital to roundoff control.

Use of this procedure may be preceded by a procedure like TRED1 (LS-244, Householder reduction) if the original matrix is a general symmetric matrix.

The procedure will fail if more than 30 iterations are needed to isolate any one eigenvalue.

The procedure is based on the procedure IMTQL1 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) IMTQL1 (N,MACHEPS,D,E,FAIL)
where

N	BIN FIXED(31,0)	contains the order of the matrix.
MACHEPS	BIN FLOAT(21)	should be .953675E-6.
D	(*)BIN FLOAT(21)	on call contains, in its first N elements, the diagonal elements of the tridiagonal matrix. On return, these elements are replaced by the computed eigenvalues, in ascending order.
E	(*)BIN FLOAT(21)	on call contains, in its second through Nth elements, the N-1 elements of the

subdiagonal of the matrix. The first element of E is unused and may be arbitrary in value. On return, these elements are lost. LBOUND(E,1) should equal LBOUND(D,1).

FAIL LABEL is a label in the user program to which IMTQL1 will transfer if the procedure fails.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

August 21, 1974

Library Services Series
Document No. LS-259-0

FROM: NCSU Programming Services

SUBJECT: IMTQL2, for Computing All Eigenvalues and Eigenvectors of a Symmetric Tridiagonal Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes the eigenvalues and vectors of a real symmetric tridiagonal matrix using implicit QL transformations. This algorithm is advantageous over the TQL2 (LS-257) algorithm in that ordering of the matrix elements is not vital to roundoff control.

Use of this procedure may be preceded by a procedure like TRED2 (LS-246, Householder reduction) if the original matrix, A, is a general symmetric matrix.

If the tridiagonal matrix, T, was obtained by such a reduction, then IMTQL2 will find the vectors of A directly without first finding the vectors of T.

The process will fail if more than 30 iterations are required to isolate any eigenvalue.

The procedure is based on the procedure IMTQL2 described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) IMTQL2 (N,MACHEPS,D,E,Z,FAIL)
where

N	BIN FIXED(31,0)	contains the order of the matrix.
MACHEPS	BIN FLOAT(21)	should be .953675E-6.
D	(*)BIN FLOAT(21)	on call contains, in its first N elements, the diagonal elements of T. On return, these elements are replaced by the computed eigenvalues in ascending order.
E	(*)BIN FLOAT(21)	on call contains, in its second through Nth elements, the N-1 elements of the subdiagonal of T. The first element of E is not used and is arbitrary. On return, these elements will be lost. LBOUND(E,1) should equal LBOUND(D,1).

Z (*,*)BIN FLOAT(21) on call, the first N rows and N columns will contain either a) the identity matrix, if the vectors of T itself are desired or b) if T was derived from a full symmetric matrix by TRED2, and it is the vectors of that matrix that are desired: the matrix Z which is output by TRED2. On return, the columns of Z will contain the appropriate normalized eigenvectors. LBOUND(Z,1) should equal LBOUND(Z,2) and LBOUND(D,1).

LABEL FAIL is a label in the user program to which IMTQL2 will transfer if the procedure fails.

2) DIMTQL2 (N,MACHEPS,D,E,Z,FAIL)
where the parameters have the same meaning as with IMTQL2 except for:

MACHEPS BIN FLOAT(53) should be .222044604924E-15.

D (*)BIN FLOAT(53)

E (*)BIN FLOAT(53)

Z (*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

September 4, 1974

Library Services Series
Document No. LS-260-0

FROM: NCSU Programming Services

SUBJECT: COMNCE, Norm Reducing Transformation for Complex Square Matrices

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which performs a norm-reducing diagonal similarity transformation on a complex square matrix. This is a useful and often desirable transformation to perform prior to computing eigenvalues by many algorithms as it will reduce computational roundoff errors. The procedure also recognizes "isolated" eigenvalues (available by inspection without any computation) and its use insures that such eigenvalues are determined exactly, however ill-conditioned they may be. The effect of the procedure is to transform the matrix so that rows and columns have nearly the same norm.

If the eigenvectors of a matrix which have been balanced by this procedure are found, the vectors of the original matrix can be obtained through use of the procedure COBBAK (LS-266).

The procedure is based on the procedure BALANCE described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) COMNCE (N,B,A,LOW,HI,D)
where

N	BIN FIXED(31,0)	contains the order of the matrix.
B	BIN FIXED(31,0)	should be 16.
A	(*,*)BIN FLOAT(21) COMPLEX	on call contains, in its first N rows and N columns the matrix to be balanced. On return, the original matrix is overwritten by the balanced matrix. LBOUND(A,1) should equal LBOUND(A,2).
(LOW,HI)	BIN FIXED(31,0)	on return, are two integers such that A(I,J) is zero if I>J and J=1 to LOW-1 or I=HI+1 to N.

D (*)BIN FLOAT(21) on return, contains information on the transformations, and the scaling factors used, which will be needed by certain other procedures, like COBBAK. LBOUND(D,1) should equal LBOUND(A,1).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

September 4, 1974

Library Services Series
Document No. LS-261-0

FROM: NCSU Programming Services

SUBJECT: COMHES, for Reduction of a Complex Matrix to Upper-Hessenberg Form

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which will reduce a specified submatrix of a complex matrix to upper-Hessenberg form by stabilized elementary (non-orthogonal) similarity transformations.

This procedure will be more effective if the matrix to be reduced is pre-conditioned by COMNCE (LS-260) or some similar procedure.

If eigenvectors of the derived matrix are later computed, the eigenvectors of the original matrix can be obtained by COMBAK (LS-265).

This procedure is based on the procedure COMHES described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) COMHES (N,K,L,A,INT)
where

N	BIN FIXED(31,0)	is the order of the full matrix.
(K,L)	BIN FIXED(31,0)	the procedure reduces the submatrix of order L-K+1 which starts at A(K,K) and finishes at A(L,L). If the entire matrix is to be reduced: K=LBOUND(A,1) and L=K+N-1. If COMNCE has been previously used, its output LOW and HI are used for K and L.
A	(*,*)BIN FLOAT(21) COMPLEX	on call contains, in its first N rows and N columns, the matrix to be reduced. On return, the derived Hessenberg matrix has replaced the specified submatrix. The zeros in the lower portion of the Hessenberg matrix are replaced with multipliers used in the reduction. LBOUND(A,1) should equal LBOUND(A,2).

INT (*)BIN FIXED(15,0) on return, contains in elements K through L a description of the row and column interchanges involved in the reduction. This will be needed later by COMBAK, if that is used. LBOUND(INT,1) should equal LBOUND(A,1).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

September 4, 1974

Library Services Series
Document No. LS-262-0

FROM: NCSU Programming Services

SUBJECT: COMLR, for Finding All Eigenvalues of a Complex Upper Hessenberg Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure for finding all the eigenvalues of a complex upper Hessenberg matrix by LR triangularizations. If the original matrix was not in Hessenberg form, it is assumed to have been reduced to that form by COMHES (LS-261), or some similar procedure, and balanced by COMNCE (LS-260).

This procedure is based on the procedure COMLR in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) COMLR (N,MACHEPS,H,W,FAIL)
where

N	BIN FIXED(31,0)	contains the order of the matrix.
MACHEPS	BIN FLOAT(21)	should be .953675E-6.
H	(*,*)BIN FLOAT(21) COMPLEX	on call contains, in its first N rows and N columns, the Hessenberg matrix. On return, the matrix has been destroyed. LBOUND(H,2) should equal LBOUND(H,1).
W	(*)BIN FLOAT(21) COMPLEX	on return contains, in its first N elements, the computed eigenvalues. LBOUND(W,1) should equal LBOUND(H,1).
FAIL	LABEL	is a label in the calling program to which COMLR will transfer if any eigenvalue takes more than 30 iterations.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

September 4, 1974

Library Services Series
Document No. LS-263-0

FROM: NCSU Programming Services

SUBJECT: COMLR2, for Finding All Eigenvalues and Eigenvectors of a Complex Upper Hessenberg Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which computes all eigenvalues and unnormalized vectors of a complex upper Hessenberg matrix, by LR triangularization. If only a few selected vectors are required, COMLR (LS-262) followed by CXIVIT (LS-264) is more efficient.

If the Hessenberg matrix was arrived at by reduction of a general matrix, it is assumed COMHES (LS-261), or a similar procedure, performed the reduction. It is recommended when starting with a general matrix, to balance the matrix with COMNCE (LS-260) prior to COMHES. The vectors of the original unbalanced matrix can be obtained by COBBAK (LS-266).

This procedure is based on the procedure COMLR2 in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) COMLR2 (N,LOW,HI,MACHEPS,INT,H,W,V,FAIL)
where

N	BIN FIXED(31,0)	gives the order of the matrix.
(LOW,HI)	BIN FIXED(31,0)	if the matrix has been balanced by COMNCE, these are the indices LOW and HI returned by that procedure. If not, set LOW=LBOUND(H,1) and HI=LOW+N-1.
MACHEPS	BIN FLOAT(21)	should be .953675E-6.
INT	(*)BIN FIXED(15,0)	if the Hessenberg matrix was obtained by COMHES, this is the result INT produced by COMHES. If the original matrix was in Hessenberg form, set INT(I)=I, for the first N elements. LBOUND(INT,1) should equal LBOUND(H,1).

H (*,*)BIN FLOAT(21) COMPLEX on call contains the upper Hessenberg matrix stored in the relevant parts of the first N rows and N columns. If the matrix was obtained from COMHES, the remaining elements will contain the multipliers used in the reduction. If the original matrix was in Hessenberg form, the remaining elements must be zero. On return, the strict upper triangle contains the components of the eigenvectors of the triangular matrix produced by the LR algorithm. LBOUND(H,1) should equal LBOUND(H,2).

W (*)BIN FLOAT(21) COMPLEX on return contains, in its first N elements, the computed eigenvalues. LBOUND(W,1) should equal LBOUND(H,1).

V (*,*)BIN FLOAT(21) COMPLEX on return contains, in its first N columns, the vectors of the original matrix. If COMHES was used, V is the eigenvectors of the input matrix to COMHES. LBOUND(V,1) should equal LBOUND(H,1) and LBOUND(V,2).

FAIL LABEL is a label in the user program to which the procedure will transfer if more than 30 iterations are used to find any one eigenvalue.

2) DCOMLR2 (N,LOW,HI,MACHEPS,INT,H,W,V,FAIL)
where the parameters have the same meaning as with COMLR2 except for:

MACHEPS BIN FLOAT(53) should be .2220446049250314E-15

H (*,*)BIN FLOAT(53) COMPLEX

W (*)BIN FLOAT(53) COMPLEX

V (*,*)BIN FLOAT(53) COMPLEX

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

September 4, 1974

Library Services Series
Document No. LS-264-0

FROM: NCSU Programming Services

SUBJECT: CXIVIT, for Finding Selected Eigenvectors of a Complex Upper Hessenberg Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which uses inverse iteration to find r selected eigenvectors of a complex upper Hessenberg matrix, given the eigenvalues of the matrix.

It is assumed the eigenvalues have been found by COMLR (LS-262) or a similar procedure so that they are ordered so as to have the correct affiliation.

If more than, say 25% of the eigenvectors are required, COMLR2 (LS-263) is to be preferred.

This procedure is based on the procedure CXINVIT in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) CXIVIT (N,MACHEPS,C,A,W,Z)
where

N BIN FIXED(31,0) contains the order of the matrix.

MACHEPS BIN FLOAT(21) should be .953675E-6.

C (*)BIT(1) indicates the eigenvalues, in W below, for which the eigenvectors are required. If the vector corresponding to W(I) is desired, C(I)='1'B; otherwise, C(I)='0'B. LBOUND(C,1) should equal LBOUND(W,1) and the first N elements of C are used.

A (*,*)BIN FLOAT(21) COMPLEX contains the upper Hessenberg matrix in its first N rows and N columns. LBOUND(A,1) should equal LBOUND(A,2).

W (*)BIN FLOAT(21) COMPLEX on call contains, in its first N elements, the eigenvalues of A. On return, the real parts of the selected eigenvalues have been perturbed where necessary to separate close eigenvalues. LBOUND(W,1) should equal LBOUND(A,1).

Z	(*,*)BIN FLOAT(21) COMPLEX	on return contains, in its first r columns, the computed eigenvectors. Each vector occupies the first N elements of a column. Any vector that is not accepted is set to zero. LBOUND(Z,1) and LBOUND(Z,2) should equal LBOUND(A,1).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

September 4, 1974

Library Services Series
Document No. LS-265-0

FROM: NCSU Programming Services

SUBJECT: COMBAK, for Obtaining the Original Eigenvectors from Those of a Derived Complex Hessenberg Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

After finding the eigenvectors of a complex Hessenberg matrix, of order n, which was obtained by applying COMHES (LS-261) to a general complex matrix, this subprocedure can be used to obtain the vectors of the original matrix.

This procedure is based on the procedure COMBAK described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) COMBAK (LOW,HI,R,A,INT,Z)
where

(LOW,HI)	BIN FIXED(31,0)	contains the indices LOW and HI returned by COMNCE (LS-260) if it was used to prepare the matrix for reduction to Hessenberg form. Otherwise, LOW=LBOUND(A,1) and HI=LOW+n-1, where n is the order of the matrix.
R	BIN FIXED(31,0)	contains the number of vectors of the Hessenberg matrix which have been found.
A	(*,*)BIN FLOAT(21) COMPLEX	contains the matrix returned by COMHES.
INT	(*)BIN FIXED(15,0)	contains the array describing the interchanges used in COMHES.
Z	(*,*)BIN FLOAT(21) COMPLEX	on call contains, in its first R columns, R eigenvectors of the derived Hessenberg matrix. The elements of the eigenvectors occupy the first n elements of each column. On return, these vectors will be replaced by the vectors of the original matrix. These vectors are not normalized. LBOUND(Z,1) should equal LBOUND(A,1) and LBOUND(Z,2).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

September 4, 1974

Library Services Series
Document No. LS-266-0

FROM: NCSU Programming Services

SUBJECT: COBBAK, for Obtaining the Original Eigenvectors from Those of a Complex Matrix Which Had Been Balanced.

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

After finding the right-hand eigenvectors of a matrix which had been balanced by use of COMNCE (LS-260), this subprocedure can be used to obtain the eigenvectors of the original matrix.

The procedure is based on the procedure BALBAK in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) COBBAK (N,LOW,HI,M,D,Z)
where

N	BIN FIXED(31,0)	contains the order of the eigenvectors.
(LOW,HI)	BIN FIXED(31,0)	contain the indices LOW and HI returned by COMNCE.
M	BIN FIXED(31,0)	contains the number of eigenvectors to be transformed.
D	(*)BIN FLOAT(21)	contains the information on the COMNCE transformations, as returned by COMNCE.
Z	(*,*)BIN FLOAT(21) COMPLEX	on call contains, as its first M columns, the eigenvectors to be transformed. The vectors occupy the first N elements of each column. On return, these vectors will be replaced with vectors of the original matrix.

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

September 4, 1974

Library Services Series
Document No. LS-267-0

FROM: NCSU Programming Services

SUBJECT: BALNCE, Norm Reducing Transformation for Real Square Matrices

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which performs a norm-reducing diagonal similarity transformation on a real square matrix. This is a useful and often desirable transformation to perform prior to computing eigenvalues by many algorithms as it will reduce computational roundoff errors. The procedure also recognizes "isolated" eigenvalues (available by inspection without any computation) and its use insures that such eigenvalues are determined exactly, however ill-conditioned they may be. The effect of the procedure is to transform the matrix so that rows and columns have nearly the same norm.

If the eigenvectors of the balanced matrix are found, the vectors of the original matrix can be obtained through use of the procedure BALBAK (LS-268).

The procedure is based on the procedure BALANCE described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) BALNCE (N,B,A,LOW,HI,D)
where

N	BIN FIXED(31,0)	contains the order of the matrix.
B	BIN FIXED(31,0)	should be 16.
A	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and N columns, the matrix to be balanced. On return, the original matrix is overwritten by the balanced matrix. LBOUND(A,1) should equal LBOUND(A,2).
(LOW,HI)	BIN FIXED(31,0)	on return, are two integers such that A(I,J) is zero if I>J and J=1 to LOW-1 or I=HI+1 to N.
D	(*)BIN FLOAT(21)	on return, contains information on the transformations, and the scaling factors used, which will be needed by certain other procedures, like BALBAK. LBOUND(D,1) should equal LBOUND(A,1).

2) DBALNCE (N,B,A,LOW,HI,D)
where the parameters have the same meaning as with BALNCE except for:

A	(*,*)BIN FLOAT(53)
D	(*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

September 5, 1974

Library Services Series
Document No. LS-268-0

FROM: NCSU Programming Services

SUBJECT: BALBAK, for Obtaining the Original Eigenvectors from Those of a Real Matrix Which Had Been Balanced

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

After finding the right-hand eigenvectors of a matrix which had been balanced by use of BALNCE (LS-267), this subprocedure can be used to obtain the eigenvectors of the original matrix.

The procedure is based on the procedure BALBAK in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) BALBAK (N,LOW,HI,M,D,Z)
where

N	BIN FIXED(31,0)	contains the order of the eigenvectors
(LOW,HI)	BIN FIXED(31,0)	contain the indices LOW and HI returned by BALNCE.
M	BIN FIXED(31,0)	contains the number of eigenvectors to be transformed.
D	(*)BIN FLOAT(21)	contains the information on the BALNCE transformations, as returned by BALNCE.
Z	(*,*)BIN FLOAT(21)	on call contains, as its first M columns, the eigenvectors to be transformed. The vectors occupy the first N elements of each column. On return, these vectors will be replaced with vectors of the original matrix. Note that the storage correspondence of eigenvectors to eigenvalues depends on the choice of subprocedure for finding the vectors. LBOUND(Z,1) should equal LBOUND(A,1).

2) DBALBAK (N,LOW,HI,M,D,Z)
where the parameters have the same meaning as with BALBAK, except for:

D	(*)BIN FLOAT(53)
Z	(*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October 11, 1974

Library Services Series
Document No. LS-270-0

FROM: NCSU Programming Services

SUBJECT: ORTBAK, for Obtaining the Original Eigenvectors from Those of a Derived Hessenberg Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

After finding the eigenvectors of a Hessenberg matrix, of order n, which was obtained by applying ORTHES (LS-273) to a general real matrix, this subprocedure can be used to obtain the vectors of the original matrix.

This procedure is based on the procedure ORTBAK described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) ORTBAK (LOW,HI,R,A,D,Z)
where

(LOW,HI)	BIN FIXED(31,0)	contains the indices LOW and HI returned by BALNCE (LS-267) if that was used to prepare the matrix for reduction to Hessenberg form. Otherwise LOW=LBOUND(A,1) and HI=LOW+n-1 where n is the order of the matrix specified for ORTHES.
R	BIN FIXED(31,0)	contains the number of eigenvectors of the Hessenberg matrix which have been found.
A	(*,*)BIN FLOAT(21)	contains the matrix returned by ORTHES.
D	(*)BIN FLOAT(21)	on call contains information about the orthogonal transformations as returned by ORTHES. This information is destroyed by ORTBAK.
Z	(*,*)BIN FLOAT(21)	on call contains, in its first R columns, R eigenvectors of the derived Hessenberg matrix. The elements of the eigenvectors occupy the first n elements of each column. On return, these vectors will be replaced by the vectors of the original matrix. These vectors are not normalized. Note that the storage correspondence of eigenvectors to eigenvalues depends on the choice of subprocedure for finding the vectors. LBOUND(Z,1) should equal BLOUND(A,1).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October 11, 1974

Library Services Series
Document No. LS-271-0

FROM: NCSU Programming Services

SUBJECT: ORTANS, for Finding the Transformation Matrix in a Hessenberg Reduction

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which forms the matrix of accumulated transformations involved in the Hessenberg reduction in ORTHES (LS-273).

This matrix is required by HQR2 (LS-276) which finds all the eigenvalues and eigenvectors of the original matrix input to ORTHES.

The procedure is based on the procedure ORTRANS described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) ORTANS (N,LOW,HI,H,D,V)
where

N	BIN FIXED(31)	is the order of the full matrix.
(LOW,HI)	BIN FIXED(31)	contains the indices LOW and HI returned by BALNCE (LS-267) is that was used to prepare the matrix for reduction to Hessenberg form. Otherwise, LOW=LBOUND(H,1) and HI=LOW+N-1
H	(*,*)BIN FLOAT(21)	contains the Hessenberg matrix produced by ORTHES. LBOUND(H,1) should equal LBOUND(H,2).
D	(*)BIN FLOAT(21)	contains information produced by ORTHES. This information is destroyed by ORTRANS.
V	(*,*)BIN FLOAT(21)	on return, contains, in its first N rows and N columns, the matrix defining the similarity reduction from the original matrix to H. LBOUND(V,1) should equal LBOUND(V,2) and LBOUND(H,1).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October 11, 1974

Library Services Series
Document No. LS-273-0

FROM: NCSU Programming Services

SUBJECT: ORTHES, for Reduction of a Real Square Matrix to Upper-Hessenberg Form

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which will reduce a specified submatrix of a real, square matrix to upper-Hessenberg form by orthogonal transformations.

This procedure will be more effective if the matrix to be reduced is preconditioned by BALNCE (LS-267) or some similar procedure.

If eigenvectors of the derived matrix are later computed, the eigenvectors of the original matrix can be obtained from ORTBAK (LS-270).

The procedure is based on the procedure ORTHES described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) ORTHES (N,K,L,TOL,A,D)
where

N	BIN FIXED (31,0)	contains the order of the matrix to be reduced.
(K,L)	BIN FIXED(31,0)	the procedure reduces the submatrix of order L-K+1 which starts at A(K,K) and finishes at A(L,L). If the entire matrix is to be reduced: K=LBOUND(A,1) and L=K+N-1. If BALNCE has been previously used, its output LOW and HI are used for K and L.
TOL	BIN FLOAT(21)	should be .565980E-72
A	(*,*)BIN FLOAT(21)	on call contains, in its first N rows and N columns, the matrix to be reduced. On return, the derived Hessenberg matrix has replaced the specified submatrix. The zeros in the lower portion of the Hessenberg matrix are replaced with multipliers used in the reduction. LBOUND(A,1) should equal LBOUND(A,2).

D	(*)BIN FLOAT(21)	on return contains, in elements K through L, additional information on the transformations used. This will be required in ORTBAK, if that is used. LBOUND(D,1) should equal LBOUND (A,1).

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October , 1974

Library Services Series
Document No. LS-274-0

FROM: NCSU Programming Services

SUBJECT: DIRANS, for Finding the Transformation Matrix in a Hessenberg Reduction

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I subprocedure which forms the matrix of accumulated transformations involved in the Hessenberg reduction in DIRHES (LS-272).

This matrix is required by HQR2 (LS-276) which finds all the eigenvalues and eigenvectors of the original matrix input to DIRHES.

The procedure is based on the procedure DIRRANS described in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) DIRANS (N,LOW,HI,INT,H,V)
where

N	BIN FIXED(31)	is the order of the full matrix.
(LOW,HI)	BIN FIXED(31)	contains the indices LOW and HI returned by BALNCE (LS-267) if that was used to prepare the matrix for reduction to Hessenberg form. Othe wise, LOW=LBOUND(H,1) and HI=LOW+N-1.
INT	(*)BIN FIXED(15)	contains information produced by DIRHES. LBOUND(INT,1) should equal LBOUND(H,1).
H	(*,*)BIN FLOAT(21)	contains the Hessenberg matrix produced by DIRHES. LBOUND(H,1) should equal LBOUND(H,2).
V	(*,*)BIN FLOAT(21)	on return, contains, in its first N rows and N columns, the matrix defining the similarity reduction from the original matrix to H. LBOUND(V,1) should equal LBOUND(V,2) and LBOUND(H,1).

2) DDIRANS (N,LOW,HI,INT,H,V)
where the parameters have the same meaning as above, except for:

H	(*,*)BIN FLOAT(53)
V	(*,*)BIN FLOAT(53)

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

October 11, 1974

Library Services Series
Document No. LS-275-0

FROM: NCSU Programming Services

SUBJECT: HQR, for Finding All Eigenvalues of a Real Upper Hessenberg Matrix

SUPPORT TYPE: A

DIRECT INQUIRIES TO: Campus/NCECS Programming Services

This is a PL/I procedure which computes all eigenvalues of a real upper Hessenberg matrix by QR transformations.

If the Hessenberg matrix was obtained by reduction of a general matrix, it is assumed that the reduction was carried out by DIRHES (LS-272), ORTHES (LS-273) or some similar procedure. If this was the case, it is recommended that the original matrix be balanced by BALNCE (LS-267) prior to reduction.

This procedure is based on the procedure HQR in Handbook of Automatic Computation, Vol. 2, Linear Algebra; Wilkinson, Reinsch; Springer-Verlag (1971), where a description of the method can be found.

Implemented versions are:

1) HQR (N,MACHEPS,H,W,CNT,FAIL)
where

N		BIN FIXED(31,0)	contains the order of the matrix.
MACHEPS		BIN FLOAT(21)	should be .953675E-6.
H	(*,*)	BIN FLOAT(21)	on call contains, in its first N rows and N columns, the upper Hessenberg matrix. On return, the matrix has been destroyed. LBOUND(H,2) should equal LBOUND(H,1).
W	(*)	BIN FLOAT(21) COMPLEX	on return contains, in its first N elements, the computed eigenvalues. LBOUND(W,1) should equal LBOUND(H,1).
CNT	(*)	BIN FIXED(15,0)	on return contains the number of iterations required for each eigenvalue. If two values are found simultaneously as a pair, then the number of iterations is given with a positive sign for the first and a negative sign for the second. LBOUND(CNT,1) should equal LBOUND(W,1).

FAIL LABEL is a label in the user program to which HQR will transfer when 30 iterations fail to isolate the current eigenvalue.

2) DHQR (N,MACHEPS,H,W,CNT,FAIL)
where the parameters have the same meaning as with HQR except for:

MACHEPS BIN FLOAT(53) should be .2220446049250314E-15

H (*,*)BIN FLOAT(53)

W (*)BIN FLOAT(53) COMPLEX

Duke University
Durham
T.U.C.C.
Research Triangle Park
University of
North Carolina
Chapel Hill
N.C. State University
Raleigh

TRIANGLE UNIVERSITIES COMPUTATION CENTER

MEMORANDUM

June 6, 1975

Library Services Series
Document No. LS-071-1

FROM: Dr. J. R. Denk, UNC at Greensboro

SUBJECT: LSQFIT - Subroutine for Least Square Fit on Linear Data

SUPPORT TYPE: A

LSQFIT is a PL/I subroutine that does a least squares fit on linear data. It calculates the slope and intercept of the best fit line, the best fit ordinate value for each experimental abscissa value, the deviation of each experimental ordinate value from the best-fit value, and the root mean square standard deviation.

I. Calling sequence and definition of parameters

LSQFIT may be called by any PL/I program with the following statement:

```
CALL LSQFIT  (N,X,Y,YLSF,DEV,RMSY,AØ,A1,NOGO);
```

Input parameters:

```
    DECIMAL FITED (5,0)--
N:  Number of ordered pairs (points).
    ARRAY (N) DECIMAL FLOAT (SINGLE)--
X:  Abscissa values (any units).
    ARRAY (N) DECIMAL FLOAT (SINGLE)--
Y:  Ordinate values (any units).
```

Output parameters:

```
    ARRAY(N) DECIMAL FLOAT (SINGLE)--
YLSF: Best fit values of Y.
    ARRAY(N) DECIMAL FLOAT (SINGLE)--
DEV:  Deviation of Y values from YLSF values.
    DECIMAL FLOAT (SINGLE)--
RMSY: Root mean square deviation.
    DECIMAL FLOAT (SINGLE)--
AØ:   Y intercept of best fit line.
```

DECIMAL FLOAT (SINGLE)--

A1: Slope of best fit line.

DECIMAL FIXED(SP)--

NOGO: Switch to check for input error. If value is set to one, then subroutine has detected an error and issued a message.

II. System Information

Source Language: PL/I

Storage Required: 4088_{16}