AD-A042 807 DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/G 12/1
CSKYTI: AN OUT-OF-CORE CHOLESKY ALGORITHM EQUATION SOLVER (WITH--ETC(U)
JUL 77 D A GIGNAC

UNCLASSIFIED DTNSRDC-77-0082 NL

| OF |
AD
A042807

END
DATE
FILMED
9 -77
DDC

DAVID W. TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER

Bethesda, Md. 20084

Report 77-0082

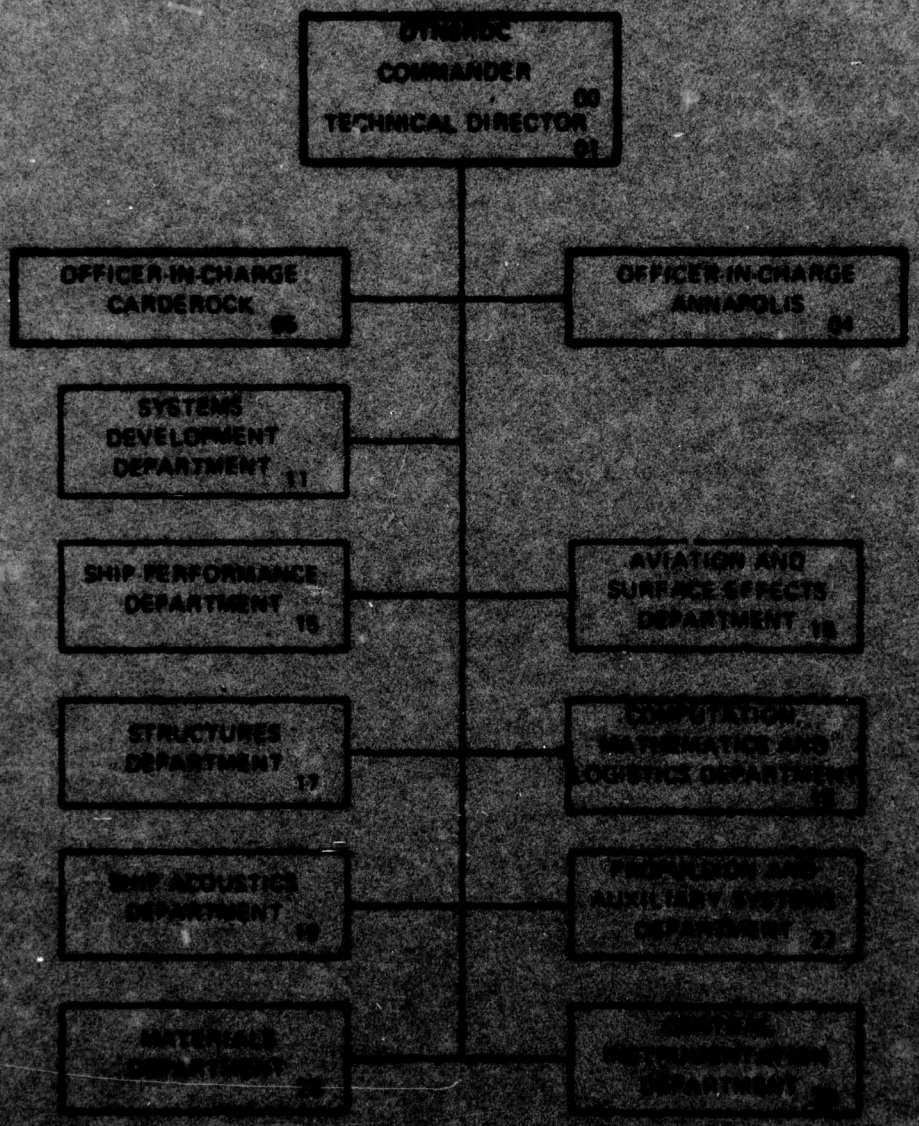
CSKYTI: AN OUT-OF-CORE CHOLESKY ALGORITHM EQUATION SOLVER (WITH RESPECT TO PROFILE) FOR THE TEXAS INSTRUMENTS' ADVANCED SCIENTIFIC COMPUTER

by

Donald A. Gignac

CSKYTI: AN OUT-OF-CORE CHOLESKY ALGORITHM EQUATION SOLVER
(WITH RESPECT TO PROFILE) FOR THE TEXAS INSTRUMENTS' ADVANCED SCIENTIFIC COMPUTER
Report 77-0082

MAJOR DTNSRDC ORGANIZATIONAL COMPONENTS



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: DTNSRDC-77-0082	2. GOVT ACCESSION NO.: (9) Research and
	3. RECIPIENT'S CATALOG NUMBER: development rep
4. TITLE (and Subtitle): CSKYTI: An Out-of-Core Cholesky Algorithm Equation Solver (with Respect to Profile) for the Texas Instruments' Advanced Scientific Computer,	5. TYPE OF REPORT & PERIOD COVERED
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): (10) Donald A. Gignac	8. CONTRACT OR GRANT NUMBER(s): (16) (17)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: David W. Taylor Naval Ship R&D Center, Bethesda, Maryland 20084	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61153N, R01403, SR0140301, Task 15321, Work Unit 1808-010
11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE: (11) July 1977
	13. NUMBER OF PAGES: 27
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): (12) 25 p.	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Matrix Analysis
Linear Equations
Cholesky Algorithm
Out-of-Core Solution
Profile Approach

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

CSKYTI is an out-of-core profile approach Cholesky algorithm equation solver for large sparse positive definite systems of linear equations. CSKYTI is an adaptation of an existing CDC 6000 series program, CSKYDG2, for the ~~Texas Instruments'~~ Advanced Scientific Computer (ASC 6), ~~at the Naval Research Laboratory.~~ The preprocessor program SETUP for CSKYTI was similarly obtained from CSKYDG2's SETUP program. This report documents a comparison of the performance of CSKYTI and its SETUP on the ASC 6 with that of CSKYDG2 and its →

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Item 20. ABSTRACT (cont.)

SETUP on the CDC 6600 at DTNSRDC. It would appear that the ASC 6 requires double precision arithmetic to obtain the significance of single precision computation on the CDC 6600. Even so, the ASC 6 was found to be fast compared to the CDC 6600.

TABLE OF CONTENTS

	Page
ABSTRACT	1
INTRODUCTION	1
MATHEMATICAL DETAILS	3
SETUP AND CSKYTI	6
THE TEST EXAMPLES	7
TABLE NOTATION	8
OBSERVATIONS	11
ACKNOWLEDGMENTS	11
PROGRAM LISTINGS	12

LIST OF TABLES

Table 1. A Comparison of CSKYTI and CSKYDG2 Execution Times for the First Set of Systems $A_N^1 X = B_N$ (NS=10) 9

Table 2. A Comparison of CSKYTI and CSKYDG2 Execution Times for the Second Set of Systems $A_N^2 X = C_N$ (NS=10) 10

ABSTRACT

CSKYTI is an out-of-core profile approach Cholesky algorithm equation solver for large sparse positive definite systems of linear equations. CSKYTI is an adaptation of an existing CDC 6000 series program, CSKYDG2, for the Texas Instruments' Advanced Scientific Computer (ASC 6) at the Naval Research Laboratory. The preprocessor program SETUP for CSKYTI was similarly obtained from CSKYDG2's SETUP program. This report documents a comparison of the performance of CSKYTI and its SETUP on the ASC 6 with that of CSKYDG2 and its SETUP on the CDC 6600 at DTNSRDC. It would appear that the ASC 6 requires double precision arithmetic to obtain the significance of single precision computation on the CDC 6600. Even so, the ASC 6 was found to be fast compared to the CDC 6600.

INTRODUCTION

One of the long range projects of the Computation, Mathematics, and Logistics Department has been the development of mathematical subroutines suitable for use in the computer-aided structural analysis of ships. Many unrelated efforts in both government and industry have resulted in computer programs that treat particular classes of structural problems. These programs often involve the solution of similar mathematical problems but, since the solutions are reached independently, the efficiency and accuracy of the various algorithms used may vary greatly. The need to coordinate these diverse efforts, to develop improved methods of more general applicability, and to produce more comprehensive programs for solving Navy structural problems became obvious. A project was therefore established to coordinate research efforts involving mathematical and computational methods in the area of structural mechanics and to integrate the work of mathematicians, computer specialists, and structural

engineers in this field.

The present considerable interest in the finite element approach to structural analysis is evidenced by the widespread use of NASTRAN (NAsa STRuctural ANalysis program) and other such programs. According to the NASTRAN Theoretical Manual,[1] "From a theoretical viewpoint, the formulation of a static structural problem for solution by the displacement method is completely described by the matrix equation KU=P." Thus, there is a need for accurate efficient computer subroutines capable of solving these large sparse positive definite systems of simultaneous linear equations. However, the order of K is often so large that, even when advantage is taken of K's symmetry and banded structure, it is not feasible, and sometimes not even possible, to store K in the core memory of a computer.

To handle such a case the computer program CSKYDG2 was developed. CSKYDG2 is an out-of-core Cholesky algorithm equation solver which takes the profile approach.[2] CSKYDG2 makes use of SETUP, a preprocessor program which stores the profile of the upper triangular half of the matrix K in random access files. CSKYDG2 and SETUP were written in FORTRAN Extended for the CDC 6000 series of computers. The present program CSKYTI was obtained by modifying CSKYDG2 so as to take as much advantage as feasible of the special features of the Texas Instruments' Advanced Scientific Computer (TI ASC 6) at the Naval Research Laboratory. The SETUP preprocessor program for CSKYTI was obtained similarly from the SETUP program for CSKYDG2. This conversion was undertaken chiefly to investigate the effectiveness of the ASC's "pipe line" architecture with respect to the "vectorization" of computer code by comparing the performance of the two versions of the same program on the ASC and the CDC 6600, respectively.

---

[1] "The NASTRAN Theoretical Manual," edited by R.H. MacNeal, National Aeronautics and Space Administration, Washington, D.C., (1969) p. 3.1-1.

[2] Gignac, D.A., "CSKYDG2: An Out-of-Core Cholesky Algorithm Equation Solver (with Respect to Profile) for Large Positive Definite Systems of Linear Equations," Naval Ship Research & Development Center Report 4655 (March 1975).

MATHEMATICAL DETAILS

DEFINITION OF TERMS

- Positive Definite

A real symmetric matrix A of order n is said to be "positive definite" if its associated quadratic form X^TAX is positive for all vectors X other than the null vector 0. A is also characterized by the fact that all its eigenvalues are positive. (The reader will remember that all the eigenvalues of a real symmetric matrix are real.) Clearly A is also nonsingular, that is, A^{-1} exists. A system of simultaneous linear equations in n unknowns, AX=B, is said to be positive definite if A is positive definite. In theory, such a system can never be singular, though numerically it may turn out to be nearly so.

- Bandwidth

A real symmetric matrix A of order n is said to be banded and have a bandwidth of m where $1 \leq m \leq n$ if $a_{ij}=0$ for $|j-1+i| > m$. The quantity m is sometimes also called the semi-bandwidth of A.

- Profile

The "profile" of a real symmetric matrix A is the upper triangular half of A stored columnwise from the diagonal element of the column to the last non-zero element of that column.

- Active Column

An "active column" of a real symmetric matrix A is a column which results in an undesirably large bandwidth.

CHOLESKY ALGORITHM

The Cholesky algorithm solution of a positive definite system of simultaneous linear equations AX=B consists of two steps:

- The factorization of A into the product of a lower triangular matrix S and its transpose T -- that is, finding S such that A=ST. This is the Cholesky decomposition of A. S and T are the unique Cholesky factors of A.

- The joint solution of the equivalent pair of triangular systems

$$SY = B$$

$$TX = Y$$

the first of which is easily solved by forward substitution and the second by backward substitution.

The Cholesky algorithm is both accurate and efficient. It is also quite stable and does not require pivoting.

Making use of symmetry we may rewirte the usual formulas for the columnwise Cholesky decomposition of A entirely in terms of T.

$$t_{11} = \sqrt{a_{11}}$$

$$t_{12} = t_{11}^{-1} a_{12}$$

$$t_{22} = \sqrt{a_{22} - t_{12}^2}$$

$$\left.\begin{aligned} t_{1j} &= t_{11}^{-1} a_{1j} \\ t_{ij} &= t_{ii}\left(a_{ij} - \sum_{k=1}^{i-1} t_{ki} t_{kj}\right) \quad i=2,\ldots,j-1 \\ t_{jj} &= \sqrt{a_{jj} - \sum_{k=1}^{j-1} t_{kj}^2} \end{aligned}\right\} j=3,\ldots,n$$

The usual formulas for the forward substitution in terms of T are

$$y_1 = t_{11}^{-1} b_1$$

$$y_i = t_{ii}^{-1}\left(b_i - \sum_{k=1}^{i-1} t_{ki}\, b_k\right) \quad i=2,\ldots,n$$

The usual formulas for the backward substitution in terms of T are

$$x_n = t_{nn}^{-1} y_n$$

$$x_i = t_{ii}\left(y_i - \sum_{k=i+1}^{n} t_{ik}\, y_k\right) \quad i=n-1,\ldots,1$$

Examination of these formulas leads to the following observations:

- It is more efficient to incorporate the forward substitution into the Cholesky decomposition, provided that one does not wish to solve

AX=B for several right-hand sides B with the same left-hand side A.

- Formulas can be modified to take advantage of the fact that A and T have the same profile.

HYPERMATRIX APPROACH

The hypermatrix approach consists in regarding the elements of a matrix A as some partitioning set of submatrices A_{ij} rather than the scalars a_{ij}. For example, if n, the order of A, equals $n_1+n_2+\ldots n_\ell$ (where the n_i are integers), then A can be regarded as a "hypermatrix" of order ℓ whose elements are the obvious partitioning set of $\ell^2 n_i \times n_j$ submatrices. "Hypervectors" are defined similarly. Thus the system AX=B defines a hypermatrix system A'X'=B' of order ℓ.

In extending the Cholesky algorithm to hypermatrices and replacing the scalars a_{ij} by the submatrices A_{ij} in the above formulas, note the following:

- Matrix multiplication, unlike scalar multiplication is not commutative.
- Scalar division is replaced by the solution of a triangular system.
- The extraction of square roots is replaced by Cholesky decomposition.

The formulas for the hypermatrix Cholesky decomposition in terms of the submatrices of T become

$$T_{11}^T T_{11} = A_{11}$$

$$T_{12} = (T_{11}^T)^{-1} A_{12}$$

$$T_{22}^T T_{22} = A_{22} - T_{12}^T T_{12}$$

$$\left.\begin{aligned} T_{1j} &= (T_{11}^T)^{-1} A_{1j} \\ T_{ij} &= (T_{ii}^T)^{-1} \left(A_{ij} - \sum_{k=1}^{i-1} T_{ki}^T T_{kj}\right) \quad i=2,\ldots,j-1 \\ T_{jj}^T T_{jj} &= A_{jj} - \sum_{k=1}^{j-1} T_{kj}^T T_{kj} \end{aligned}\right\} j=3,\ldots,n$$

The formulas for the hypermatrix forward substitution are

$$Y_1 = (T_{11}^T)^{-1} B_1$$

$$Y_i = (T_{ii}^T)^{-1} (B_i - \sum_{k=1}^{i-1} T_{ki}^T B_k) \quad i=2,\ldots,n$$

The formulas for the hypermatrix backward substitution are

$$X_n = T_{nn}^{-1} Y_n$$

$$X_i = T_{ii}^{-1} (Y_i - \sum_{k=\ell+1}^{n} T_{ik} X_k) \quad i=n-1,\ldots,1$$

Note that the positive definiteness of A guarantees that those matrices which must be invertible or positive definite according to the above formulas are, in fact, invertible or positive definite. This statement is so easily established as not to require formal proof here.

SETUP AND CSKYTI

SETUP is a preprocessor subroutine which packs the profile of A in QDAM (Qued Direct Access Method) files on tape 1, according to the specified partition. The SETUP subroutine first reads the order of A and then reads the rows of the upper triangular half of A from tape 2. The SETUP subroutine has four arguments. The partition is specified in the first two arguments, NS and NNS, where NS is the partition vector and NNS is the number of partitions.* The profile is characterized in the output of the last two arguments with respect to the given partition. The LASTBLK array contains the locations of the diagonal submatrices of the hypermatrix. The NUMBLKS array specifies the number of submatrices stored from each column of the hypermatrix.

---

* IMPORTANT: It must be pointed out to the reader that the program published in this report expects a uniform partition mesh of 10 except for the NNSth partition whose width is i, $1 \leq i < 10$. This limitation was introduced to facilitate the use of QDAM.

The CSKYTI subroutine effects a Cholesky algorithm solution of a system of linear equations via the profile approach, using the submatrices packed in random access memory by SETUP. The CSKYTI subroutine has five arguments, NNS, NS, LASTBLK, NUMBLKS, and B -- the first four as already described. The right-hand side of a system of linear equations is passed through the B array and the solution of the system is returned in the same array.

As noted previously SETUP and CSKYTI are ASC 6 adaptations of programs written for the CDC 6000 series of computers.

THE TEST EXAMPLES

EXAMPLE 1

The matrix family of Table 1, $A^1_{N^2}$, is generated as follows: Let N be an integer ≥ 3. Let C_N be the tridiagonal of order N with 4's on the diagonal and a line of -1's above and below the diagonal. Let I_N be the identity matrix of order N. An (N+1)-banded matrix of order N^2, A^1_N is constructed by

(1) stringing N C_N submatrices along the diagonal,

(2) inserting lines of N-1 $-I_N$ submatrices above and below the diagonal, and

(3) setting the remaining elements of $A^1_{N^2}$ equal to 0.

Application of Gerschgorin's theorem shows $A^1_{N^2}$ to be positive definite. The right-hand side of the system $A^1_{N^2}X = B_{N^2}$ is chosen such that all components of the exact solution vector have the value 1.

EXAMPLE 2

The matrix family of Table 2, $A^2_{N^2}$, is generated as follows: Let N be an integer ≥ 3. Assume the real symmetric matrix of order N^2 and bandwidth N with N^2 on the diagonal and -1 elements filling out the rest of the band. Then change the value of each zero element in the last N rows and columns to a -1. As before, Gerschgorin's theorem shows $A^2_{N^2}$ to be positive

definite. Note that from the viewpoint of bandwidth, $A^2_{N^2}$ is a full matrix. The right-hand side of the system $A^2_{N^2}X = C_{N^2}$ is chosen such that all the components of the exact solution vector, except the last, which is 1, are zero.

TABLE NOTATION

The solution times for $A^1_M X = B_M$ and $A^2_M X = C_M$ are tabulated in Tables 1 and 2, respectively. These test problems were originally run on the CDC 6600 at DTNSRDC using the OPT=1 FORTRAN Extended compiler under the SCOPE 3.3 operating system.[2] They were rerun on the ASC 6 at NRL using the NX FORTRAN compiler under the General Purpose Operating System (GPOS).

The column headings are defined as follows:

- N - the order of the system,
- M - system bandwidth,
- NS - the partition vector,
- NN - the order of the hypermatrix system.
- MM - hypermatrix system's bandwidth,
- T_{SETUP} - the time required to pack the hypermatrix system into random access mass storage,
- T_{SOLVE} - the time required to solve the hypermatrix system, and
- T_{TOTAL} - the sum of the T_{SETUP} and T_{SOLVE} times.

(The above times are given in terms of CPU seconds.)

TABLE 1 - A COMPARISON OF CSKYTI AND CSKYDG2 EXECUTION TIMES FOR THE FIRST SET OF SYSTEMS $A_N^1 X = B_N$ (NS=10)

N	M	NN	MM	CSKYTI Execution Times (ASC 6 CPU Seconds)			CSKYDG2 Execution Times (CDC 6600 CPU Seconds)		
				T_{SETUP}	T_{SOLVE}	T_{TOTAL}	T_{SETUP}	T_{SOLVE}	T_{TOTAL}
25	6	3	2	.01	.01	.02	.04	.05	.09
100	11	10	2	.05	.05	.10	.45	.22	.67
225	16	23	3	.14	.25	.38	2.20	1.17	3.37
400	21	40	3	.33	.43	.76	6.77	2.12	8.89
625	26	63	4	.70	1.10	1.80	16.39	5.64	22.03
900	31	90	4	1.30	1.59	2.89	33.71	8.29	42.00
1225	36	123	5	2.26	3.17	5.43	62.86	17.11	79.97
1600	41	160	5	3.61	4.14	7.75	106.49	21.96	128.45
2025	46	203	6	5.65	7.32	12.9	170.75	39.88	210.63
2500	51	250	6	8.34	9.06	17.40	259.74	49.20	308.94

TABLE 2 - A COMPARISON OF CSKYTI AND CSKYDG2 EXECUTION TIMES FOR THE SECOND SET OF SYSTEMS $A_N^2 X = C_N$ (NS=10)

N	M	NN	MM	CSKYTI Execution Times (ASC 6 CPU Seconds)			CSKYDG2 Execution Times (CDC 6600 CPU Seconds)		
				T_{SETUP}	T_{SOLVE}	T_{TOTAL}	T_{SETUP}	T_{SOLVE}	T_{TOTAL}
25	25	3	3	.01	.01	.02	.04	.06	.10
100	100	10	10	.05	.09	.14	.47	.45	.92
225	225	23	23	.16	.42	.58	2.22	2.38	4.60
400	400	40	40	.35	.86	1.19	6.89	5.22	12.11
625	625	63	63	.76	2.25	3.01	16.68	13.99	30.67
900	900	90	90	1.39	3.44	4.83	34.28	23.10	57.38
1225	1225	123	123	2.44	6.80	9.24	63.56	47.38	110.94
1600	1600	160	160	3.88	9.34	13.22	108.83	67.55	176.38
2025	2025	203	203	6.08	16.52	22.60	173.70	118.55	292.25
2500	2500	250	250	8.89	21.19	30.08	264.18	157.16	421.34

OBSERVATIONS

At the outset it was suspected that ASC single precision computation (32-bit word) could not provide solution significance comparable to that of the CDC 6000 series (60-bit word). This was found to be the case. However, since the ASC double precision word is slightly more significant (64 bits) than the CDC 60-bit word, maintaining significance proved to be no problem, all the more so since the ASC user is not unduly penalized for resorting to double precision arithmetic to maintain significance. (Changing from single to double precision arithmetic on the CDC 6000 series almost quadruples the running time of a given program.) Both CSKYTI and CSKYDG2 provided the same significance--at least 12 to 13 significant digits--for the esamples in this report.

Clearly the improvement in computing time can only be described as phenomenal. The ASC version of SETUP runs up to 31 times faster than the CDC 6600 version, CSKYTI runs about 5 times faster than CSKYDG2 and the total time may be as much as 17 times faster. The remarkable improvement in SETUP's running time results from the more efficient reading and writing of scratch tape 2 (which is really on a disk) by the head-per-track disk drives of the ASC. The considerable improvement in CSKYTI's running time over than of CSKYDG2 is no doubt due to the fair amount of vectorization provided by the NX FORTRAN so as to exploit the "pipeline" architecture of the ASC.

ACKNOWLEDGMENTS

The author wishes to thank the following individuals for their interest and assistance: Dr. Elizabeth H. Cuthill (DTNSRDC Code 1805); Dr. Gordon Everstine and Mr. Michael Golden (DTNSRDC Code 1844); Mr. Richard Van Eseltine and Mr. Paul Morawski (DTNSRDC Code 1843); and Mrs. Barbara Brooks (NRL Code 422.23).

PROGRAM LISTINGS

(Program listings are given on the following pages.)

```
CSN           STATEMENT

0001            SUBROUTINE SETUP(NS,NNS,LASTBL ,NUMBLK )
      C
0002            IMPLICIT REAL*8(A-H,O-Z)
0003            DIMENSION HOLD(3025)
0004            DIMENSION NS(1),LASTBL (1),NUMBLK (303)
      *         COMMON A1(1600),A(6300)
0005            COMMON A1(100),A(30250)
0006            INDXX(JXYZ)=(JXYZ-1)*200+1
      C
0007            REWIND 2
0008            N=0
0009            DO 1 I=1,NNS
0010          1 N=N+NS(I)
0011            NP0=N+1
0012            KLIM=NP0
0013            DO 2 I=2,NNS
0014          2 NUMBLK(I)=-I
0015            NSS=NS(1)
0016            DO 3 I=1,NSS
0017            KLIM=KLIM-1
0018            READ(2)(HOLD(K),K=1,KLIM)
0019            DO 3 K=1,KLIM
0020          3 A(NP0+(I-1)+K)=HOLD(K)
0021            NUMBLK (1)=1
0022            LASTBL (1)=1
0023            DO 10 I=2,NNS
0024            NS0=NS(I-1)
0025            JDISP=NS0
0026            DO 7 J=I,NNS
0027            NSJ=NS(J)
0028            IF (NUMBLK (J)) 4,4,7
0029          4 IISTOR =-N+JDISP
0030            DO 5 II=1,NS0
0031            IISTOR=IISTOR+N
0032            DO 5 JJ=1,NSJ
0033            IF (A(IISTOR +JJ)) 6,5,6
0034          5 CONTINUE
0035            NUMBLK (J)=NUMBLK (J)+1
0036            GO TO 7
0037          6 NUMBLK (J)=-NUMBLK (J)
0038          7 JDISP=JDISP+NSJ
0039            NSS=NS(I)
0040            DO 8 II=1,NSS
0041            KLIM=KLIM-1
0042            READ(2)(HOLD(K),K=1,KLIM)
0043            DO 8 K=1,KLIM
0044          8 A(NP0+(II-1)+K)=HOLD(K)
0045            IF (NUMBLK (I)) 9,9,10
0046          9 NUMBLK (I)=1
0047         10 LASTBL (I)=LASTBL (I-1)+NUMBLK (I)
      C
```

```
CSN          STATEMENT

0048           REWIND 2
0049           KLIM=NP0
0050           NSS=NS(1)
0051           DO 11 I=1,NSS
0052           KLIM=KLIM-1
0053           READ(2)(HOLD(K),K=1,KLIM)
0054           DO 11 K=1,KLIM
0055        11 A(NP0*(I-1)+K)=HOLD(K)
0056           DO 12 I=1,NSS
0057           DO 12 J=I,NSS
0058        12 A1(NSS*(J-1)+I)=A(N*(I-1)+J)
0059           DO 22 I=1,NSS
0060           DO 22 J=I,NSS
0061        22 A1(NSS*(I-1)+J)=A(N*(I-1)+J)
0062           LQZ=NSS*NSS
0063           LQZ=LQZ*2
0064           CALL QPUT(1,LQZ,1,A1)
0065           IDISP=0
0066           DO 18 I=2,NNS
0067           NS0=NS(I-1)
0068           IDISP=IDISP+NS0
0069           IST0N=2-I
0070           IHOLD=I-2
0071           JDISP=NS0
0072           DO 15 J=I,NNS
0073           IF (IST0N+J-NUMBLK (J)) 13,13,15
0074        13 NSJ=NS(J)
0075           IISTOR =-NSJ
0076           IIKEEP=-N+JDISP
0077           DO 14 II=1,NS0
0078           IISTOR =IISTOR +NSJ
0079           IIKEEP=IIKEEP+N
0080           DO 14 JJ=1,NSJ
0081        14 A1(IISTOR +JJ)=A(IIKEEP+JJ)
0082           LQZ=NS0*NSJ
0083           LQZ=LQZ*2
0084           IABC=LASTBL(J-1)+J-IHOLD
0085           CALL QPUT(1,LQZ,INDXX(IABC),          A1)
0086        15 JDISP=JDISP+NS(J)
0087           NSS=NS(I)
0088           KSAVE=0
0089           DO 16 II=1,NSS
0090           KLIM=KLIM-1
0091           READ(2)(HOLD(K),K=1,KLIM)
0092           DO 16 K=1,KLIM
0093        16 A(NP0*(II-1)+K)=HOLD(K)
0094           DO 17 II=1,NSS
0095           DO 17 JJ=II,NSS
0096        17 A1(NSS*(JJ-1)+II)=A(KSAVE+N*(II-1)+JJ)
0097           DO 27 II=1,NSS
0098           DO 27 JJ=II,NSS
```

```
CSN          STATEMENT

0099      27 A1(NSS*(II-1)+JJ)=A(KSAVE+N*(II-1)+JJ)
0100         LQZ=NSS*NSS
0101         LQZ=LQZ*2
0102         IABC=LASTBL(I-1)+1
0103      18 CALL QPUT(1,LQZ,INDXX(IABC)  ,A1)
         C
0104         RETURN
0105         END
```

```
CSN             STATEMENT

0001            SUBROUTINE CSKYTI (NNS,NS,LASTBL ,NUMBLK ,B)
      C
0002            IMPLICIT REAL*8(A-H,O-Z)
0003            DIMENSION NS(1),LASTBL (1),NUMBLK (1),B(1)
0004            COMMON AA(1600),BB(40),A1(1600),BC(40),A2(1600)
0005            INDXX(JXYZ)=(JXYZ-1)*200+1
      C
0006            NSSUM1=0
0007            DO 12 J=1,NNS
0008            NSJ=NS(J)
0009            JSAVE=LASTBL (J)
0010            LLLIM=NUMBLK (J)-1
0011            LQZ=NS(J-LLLIM)*NSJ
0012            LQZ=LQZ*2
0013            LSAVE=INDXX(JSAVE)
0014            CALL QFIND(1,LQZ,LSAVE)
0015            CALL QGET(1,LQZ,LSAVE,AA)
0016            DO 1 K=1,NSJ
0017          1 BB(K)=B(NSSUM1+K)
0018            IF (LLLIM) 2,10,2
0019          2 KLIM1=J-LLLIM
0020            JM0=J-1
0021            JKEEP=LASTBL (JM0)-2
0022            NSSUM2=NSSUM1
0023            DO 3 K=KLIM1,JM0
0024          3 NSSUM2=NSSUM2-NS(K)
0025            JP0=J+1
0026            LLSAVE=NUMBLK (J)+1
0027            JH0LD=-(LLSAVE+1)
0028            LLH0LD=LASTBL(J)+1
0029            DO 9 LL=1,LLLIM
0030            L=LLSAVE-LL
0031            JP0L=JP0-L
0032            JP0L2=JP0L+2
0033            KLIM2=NS(JP0L)
0034            LQZ=KLIM2*KLIM2
0035            LQZ=LQZ*2
0036            INDEX=LASTBL(JP0L)-NUMBLK(JP0L)+1
0037            INDEX=INDXX(INDEX)
0038            CALL QFIND(1,LQZ,INDEX)
0039            CALL QGET(1,LQZ,INDEX,A1)
0040            CALL FORWDS (A1,AA,KLIM2,NSJ)
0041            LQZ=KLIM2*NSJ
0042            LQZ=LQZ*2
0043            INDEX=INDXX(LLH0LD-LL)
0044            CALL QPUT(1,LQZ,INDEX     ,AA)
0045            DO 4 K=1,KLIM2
0046          4 BC(K)=B(NSSUM2+K)
0047            CALL MLTPLY (BB,AA,BC,KLIM2,NSJ,1)
0048            NSSUM2=NSSUM2+KLIM2
0049            KLIM2=NUMBLK (JP0L+1)
```

```
CSN          STATEMENT

0050           NS1=NS(JP0L+1)
0051           LQZ=NS1*NSJ
0052           LQZ=LQZ*2
0053           INDEX=JSAVE-LL
0054           INDEX=INDXX(INDEX)
0055           CALL QFIND(1,LQZ,INDEX)
0056           CALL QGET(1,LQZ,INDEX,AA)
0057           IF (KLIM2-1) 5,9,5
0058         5 KSTORE=L+KLIM2+JHOLD
0059           IF (KSTORE) 7,7,6
0060         6 KLIM2=KLIM2-KSTORE
0061         7 KSTRE1=LASTBL(JP0L)
0062           KSTRE2=JKEEP+L
0063           DO 8 K=2,KLIM2
0064           NS2=NS(JP0L2-K)
0065           LQZ=NS2*NS1
0066           LQZ=LQZ*2
0067           INDEX=KSTRE1+K
0068           INDEX=INDXX(INDEX)
0069           CALL QFIND(1,LQZ,INDEX)
0070           CALL QGET(1,LQZ,INDEX,A1)
0071           LQZ=NS2*NSJ
0072           LQZ=LQZ*2
0073           INDEX=KSTRE2+K
0074           INDEX=INDXX(INDEX)
0075           CALL QFIND(1,LQZ,INDEX)
0076           CALL QGET(1,LQZ,INDEX,A2)
0077         8 CALL MLTPLY (AA,A1,A2,NS2,NS1,NSJ)
0078         9 CONTINUE
0079        10 CALL CHOLSK (AA,BB,NSJ)
0080           LQZ=NSJ*NSJ
0081           LQZ=LQZ*2
0082           INDEX=JSAVE-LLLIM
0083           INDEX=INDXX(INDEX)
0084           CALL QPUT(1,LQZ,INDEX,AA)
0085           DO 11 K=1,NSJ
0086        11 B(NSSUM1+K)=BB(K)
0087        12 NSSUM1=NSSUM1+NSJ
         C
0088           NNSP0=NNS+1
0089           DO 36 JJ=1,NNS
0090           J=NNSP0-JJ
0091           NSJ=NS(J)
0092           NSSUM1=NSSUM1-NSJ
0093           DO 31 K=1,NSJ
0094        31 BB(K)=B(NSSUM1+K)
0095           LLIM=NUMBLK (J)-1
0096           JSTORE=LASTBL (J)-LLIM
0097           LQZ=NSJ*NSJ
0098           LQZ=LQZ*2
0099           LSTORE=INDXX(JSTORE)
```

```
CSN             STATEMENT

0100              CALL QFIND(1,LQZ,LSTORE)
0101              CALL QGET(1,LQZ,LSTORE,AA)
0102              CALL BCKWDS (AA,BB,NSJ,1)
0103              DO 32 K=1,NSJ
0104           32 B(NSSUM1+K)=BB(K)
0105              IF (LLIM) 33,36,33
0106           33 NSSUM2=NSSUM1
0107              DO 35 L=1,LLIM
0108              KLIM=NS(J-L)
0109              NSSUM2=NSSUM2-KLIM
0110              DO 34 K=1,KLIM
0111           34 BC(K)=B(NSSUM2+K)
0112              LQZ=KLIM*NSJ
0113              LQZ=LQZ*2
0114              INDEX=JSTORE+L
0115              INDEX=INDXX(INDEX)
0116              CALL QFIND(1,LQZ,INDEX)
0117              CALL QGET(1,LQZ,INDEX,AA)
0118              CALL MLTPL(BC,AA,BB,KLIM,NSJ,1)
0119              DO 35 K=1,KLIM
0120           35 B(NSSUM2+K)=BC(K)
0121           36 CONTINUE
0122              RETURN
          C
0123              END
```

```
CSN          STATEMENT

0001         SUBROUTINE MLTPLY(A,B,C,NS1,NS2,NS3)
0002         IMPLICIT REAL*8(A-H,O-Z)
       C
0003         DIMENSION A(1600),B(1600),C(1600),X(40)
0004         DO 2 I=1,NS2
0005         DO 1 K=1,NS1
0006       1 X(K)=B(NS2*K-NS2+I)
0007         DO 2 J=1,NS3
0008         DO 2 K=1,NS1
0009       2 A(NS3*I-NS3+J)=A(NS3*I-NS3+J)+X(K)*C(NS3*K-NS3+J)
0010         RETURN
0011         END
```

```
CSN             STATEMENT

0001            SUBROUTINE MLTPL(A,B,C,NS1,NS2,NS3)
0002            IMPLICIT REAL*8(A-H,O-Z)
      C
0003            DIMENSION A(40),B(1600),C(40)
0004            DO 3 I=1,NS1
0005            DO 3 J=1,NS2
0006          3 A(I)=A(I)-B(NS2*I-NS2+J)*C(J)
0007            RETURN
0008            END
```

```
CSN             STATEMENT

0001              SUBROUTINE FORWDS(A,B,N,M)
      C
0002              IMPLICIT REAL*8(A-H,O-Z)
0003              DIMENSION A(1600),B(1600),X(40)
0004              DIMENSION S(40)
      C
0005              DO 8 J=1,M
0006              DO 1 K=1,N
0007            1 S(K)=B((K-1)*M+J)
0008              X(1)=S(1)/A(1)
0009              DO 3 I=2,N
0010              IMO=I-1
0011              DO 2 K=1,IMO
0012            2 S(I)=S(I)-X(K)*A((I-1)*N+K)
0013            3 X(I)=S(I)/A((I-1)*N+I)
0014              DO 4 K=1,N
0015            4 B((K-1)*M+J)=X(K)
0016            8 CONTINUE
0017              RETURN
      C
0018              ENTRY BCKWDS(A,B,N,M)
      C
0019              X(N)=B(N)/A(N*N)
0020              DO 6 II=2,N
0021              IPO=N-II+2
0022              DO 5 K=IPO,N
0023            5 B(N-II+1)=B(N-II+1)-X(K)*A(N*N-N*II+K)
0024            6 X(N-II+1)=B(N-II+1)/A(N*N+N+1-(N+1)*II)
0025              DO 7 I=1,N
0026            7 B(I)=X(I)
0027              RETURN
      C
0028              END
```

```
CSN             STATEMENT

0001              SUBROUTINE CHOLSK (A,B,N)
          C
0002              IMPLICIT REAL*8(A-H,O-Z)
0003              DIMENSION A(1),B(1)
          C
0004              IF (A(1)) 5,5,1
0005            1 A(1)=DSQRT(A(1))
          *     1 A(1)=SQRT(A(1))
0006              B(1)=B(1)/A(1)
0007              DO 4 I=2,N
0008              IMO=I-1
0009              IMON=IMO*N
0010              IHOLD=I-2*N
0011              SUM=A(IMON+1)
0012              DO 2 J=2,I
0013              JMO=J-1
0014              JMON=JMO*N
0015              STORE=SUM/A((J-2)*N+JMO)
0016              A(IMON+JMO)=STORE
0017              A(J*N+IHOLD)=STORE
0018              B(I)=B(I)-B(JMO)*STORE
0019              SUM=A(IMON+J)
0020              DO 2 K=1,JMO
0021            2 SUM=SUM-A(IMON+K)*A(JMON+K)
0022              IF (SUM) 5,5,3
          *     3 A(IMON+I)=SQRT(SUM)
0023            3 A(IMON+I)=DSQRT(SUM)
0024            4 B(I)=B(I)/A(IMON+I)
0025              RETURN
          C
0026            5 WRITE(6,6)
0027            6 FORMAT(1H1/14(1HO/),30X,22HBAD NEWS FROM CHOLSKY.)
0028              STOP
          C
0029              END
```

INITIAL DISTRIBUTION

Copies	
2	CHONR
	1 430/R. Lundegard
	1 432/L.D. Bram
7	NRL
	1 8441/J. Hansen
	6 4223.23/B. Brooks
2	USNA
	1 Dept Math
	1 Library
4	NAVPGSCOL
	1 59C1/G. Cantin
	1 M. Kelleher
	1 Math Dept
	1 Library
1	NAVWARCOL
1	NROTC & NAVADMINU, MIT
1	NADC
1	NELC
1	NUC
1	NWC
1	NCSL
2	NSWC, WHITE OAK
	1 331/E. Cohen
	1 331/M. Vander Vorst
1	NSWC, DAHLGREN
1	NUSC, NEW LONDON
1	NUSC, NEWPORT

Copies	
1	NAVSEA, 03F/B. Orleans
1	NAVSHIPYD BREM/LIB
1	NAVSHIPYD CHASN/LIB
1	NAVSHIPYD MARE/LIB
1	NAVSHIPYD NORVA/LIB
1	NAVSHIPYD PEARL/LIB
1	NAVSHIPYD PHILA/LIB
1	NAVSHIPYD PTSMH/LIB
12	DDC
1	AIR FORCE AERO RES LAB P. Nikolai
2	BUSTAND
	1 Dr. H. Oser
	1 M. Cordes
1	U MARYLAND Prof. Rheinboldt
1	GRUMMAN AEROSPACE CORP Bethpage, N.Y. 11714

CENTER DISTRIBUTION

Copies	Code	
1	1552	BAI, JUNE
4	1725	JONES, R.F.,JR.
		RODERICK, JOAN E
		ROTH, PETER N
		GIFFORD, LEROY N., JR.
1	1730.5	MA, JAMES H.
1	1745	NG, CHRISTOPHER
1	1800	GLEISSNER, G.H.
2	1802	FRENKIEL, F.N.
		THEILHEIMER, F.
1	1805	CUTHILL, E.H.
1	1809.3	HARRIS, D.
1	182	CAMARA, A.W.
2	1826	MEALS, L.K.
		CULPEPPER, L.M.
1	184	LUGT, H.J.
4	1843	SCHOT, J.W.
		EDDY, R.P.
		MORAWSKI, P.E.
		VAN ESELTINE, R.T.
6	1844	DHIR, S.K.
		EVERSTINE, G.C.
		GOLDEN, M.E.
		HENDERSON, F.M.
		HURWITZ, M.M.
10	1844	GIGNAC, D.A.
1	185	CORIN, T.
1	186	SULIT, R.A.
3	189	GRAY, G.R.
		TAYLOR, N.M.
		STRICKLAND, J.D.
1	1892	GOOD, S.E.
2	1966	CASPAR, J.R.
		LIU, Y.-N.
30	5214.1	REPORTS DISTRIBUTION
1	522	LIBRARY

DTNSRDC ISSUES THREE TYPES OF REPORTS

(1) DTNSRDC REPORTS, A FORMAL SERIES PUBLISHING INFORMATION OF PERMANENT TECHNICAL VALUE, DESIGNATED BY A SERIAL REPORT NUMBER.

(2) DEPARTMENTAL REPORTS, A SEMIFORMAL SERIES, RECORDING INFORMATION OF A PRELIMINARY OR TEMPORARY NATURE, OR OF LIMITED INTEREST OR SIGNIFICANCE, CARRYING A DEPARTMENTAL ALPHANUMERIC IDENTIFICATION.

(3) TECHNICAL MEMORANDA, AN INFORMAL SERIES, USUALLY INTERNAL WORKING PAPERS OR DIRECT REPORTS TO SPONSORS, NUMBERED AS TM SERIES REPORTS; NOT FOR GENERAL DISTRIBUTION.