SOLUTION OF LINEAR EQUATIONS
WITH SKYLINE-STORED SYMMETRIC MATRIX

CARLOS A. FELIPPA
Structural Mechanics Laboratory, Lockheed Palo Alto Research Laboratory, 3251 Hanover Street, Palo Alto, California 94304, U.S.A.

(Received in revised form 26 March 1974)

Abstract—Fortran IV subroutines for the in-core solution of linear algebraic systems with a sparse, symmetric, skyline-stored coefficient matrix are presented. Such systems arise in a variety of applications, notably the numerical discretization of conservative physical systems by finite differences or finite element techniques. The routines can be used for processing constrained systems without need for prearranging equations. The application to ‘superelement’ condensation of large-scale systems is discussed.

NOMENCLATURE

- $A$ symmetric coefficient matrix
- $A_{mn}, A_{i,j}$ block partitions of $A$ in constrained system
- $A^*$ condensed $A$ upon elimination of $x_i$ in superelement condensation process
- $B$ $A_{nc} - A^*$
- $C(A)$ Euclidean condition number of $A$
- $L$ lower triangular matrix
- $D$ diagonal matrix
- $U$ upper triangular matrix
- $X$ see equation (19)
- $b$ right hand side vector
- $b_0, b_1$ prescribed and unknown portions of $b$, respectively, in constrained linear system
- $b^*$ condensed $b$ upon elimination of $x_i$ in superelement condensation process
- $r$ residual vector $b - Ax$
- $x, x_0, x_r$ prescribed and unknown portions of $x$, respectively, in constrained linear system
- $x_n$ solution iterates produced in iterative refinement process
- $\Delta x_n$ solution corrections in iterative refinement process
- $y, z$ intermediate vectors in solution process, equation (3)
- $A$ linear array containing $A$ or factorization thereof
- $D, D_n$ array of pointers to diagonal elements of $A$ in $A$
- $b_{ns} b_{n+1}$ j-th element of $b, x$
- $d_j$ j-th diagonal element of $D$
- $k$ number of iterative refinement cycles
- $n$ order of $A$
- $n_f, n_c$ number of free and constrained equations, respectively
- $m$ number of random r.h.s. vectors used for estimating $C(A)$
- $t_i$ tolerance for singularity test
- $e$ floating point machine accuracy
- $\delta$ relative solution error
- $\|A\|_2, \|B\|_2$ 2-norm of matrix or vector
- $|A|^{-1}, |B|^{-1}$ matrix inverse
- $|A|^{T}, |B|^{T}$ matrix or vector transpose

INTRODUCTION

Large sparse symmetric systems of equations generally arise in practice as a result of numerical discretization of self-adjoint problems by variational finite difference or finite element methods. In the latter case, direct elimination techniques have been used to solve the resulting linear systems $Ax = b$ since the advent of the finite element method in the mid-1950's [1] (this is in contrast to classical finite difference discretization, where iterative solution techniques still prevail). Commonly used algorithms include Gauss elimination, standard Cholesky ($A = LL^T$) and modified Cholesky ($A = LDL^T$). The main implementation differences, however, concern the manner in which nonzero coefficients of $A$ are arranged in the various storage devices (e.g. core, extended core, random access mass storage, tapes), utilized during the solution process. Excellent surveys on this subject are presently available [2-4].

The routines presented here utilize only one storage level, namely, high-speed memory. The storage arrangement of matrix $A$ is the so-called skyline, profile, envelope, or variable bandwidth method [5-6]. The symmetric matrix $A$ is stored in a linear array as a string of ‘active’ columns of the upper triangle (or, equivalently, rows of the lower triangle). The active portion of each column is bounded by the diagonal and the furthest nonzero element.

It should be mentioned that more sophisticated sparse matrix storage schemes are available [2-4, 7-9]. However, implementation of most such schemes requires considerable programming skill to ensure that the data handling overhead (addressing, fetching, storing, etc.) is minimized. The skyline storage scheme offers a reasonable compromise between organizational simplicity and computational efficiency. Moreover, since the associated solution process consists largely of vector inner products, it is a natural choice for implementation on the ‘fifth-generation’ parallel and pipeline computers [10].

ALGORITHM DESCRIPTION

Purpose

Subroutines SKYFAC and SKYSOL are designed to solve the linear equation system

$$Ax = b$$

where $A$ is an $n \times n$ sparse symmetric nonsingular matrix entirely core stored in skyline form, and $x$ and $b$ are column vectors. The routines can handle the case in which some of the components of $x$ are prescribed.
Method

The symmetric factorization

$$A = LDL'$$

where $L$ is a unit lower triangular matrix, $D$ a nonsingular diagonal matrix and $U$ is the transpose of $L$, is carried out by subroutine SKYFAC by a compact Cholesky-type factorization algorithm similar to that described in $[11, \text{Contrs. I/1 and I/4}]$. The original matrix $A$ is overwritten by elements of $D'$ and $U$ in the manner discussed in the 'Organizational Details' subsection. No pivoting is used in the factorization process.

Once $A$ has been factorized, subroutine SKYSOL can be repeatedly invoked to solve equation (1) for any right hand side $b$. The solution vector is obtained by the usual three-stage process:

$$Lz = b \quad \text{(forward substitution)} \quad (3a)$$
$$y = D'y \quad \text{(scaling)} \quad (3b)$$
$$ux = y \quad \text{(backsubstitution)} \quad (3c)$$

Constrained system

SKYFAC-SKYSOL can also be used to solve the more general case of a constrained linear system:

$$\begin{bmatrix} A_f & A_{fc} \\ A_{cf} & A_c \end{bmatrix} \begin{bmatrix} x_f \\ x_c \end{bmatrix} = \begin{bmatrix} b_f \\ b_c \end{bmatrix} \quad n_f \text{ equations}$$

where both $b_f$ and $x_f$ are prescribed (subscripts $f$ and $c$ stand for free and constrained, respectively). Equations associated with $x_c$ are called constraint equations or simply constraints; they result from the discretization of essential boundary conditions. A constraint equation corresponding to a prescribed zero (nonzero) component of $x$ is called homogeneous (nonhomogeneous). If all constraints are homogeneous, i.e. $x_c = 0$, the system (4) is said to be homogeneously constrained.

System (4) can be solved for $x_f$ upon reduction to the form

$$A_f x_f = b_f - A_{cf} x_c = b_f$$

where $b_f = b_f$ if the system is homogeneously constrained. The subvector $b_f$ may then be obtained from the second matrix equation in (4), if desired.

It is important to note that it is not necessary to physically arrange a constrained system (1) to put it into the partitioned form (4). Constrained and unconstrained equations may be freely intermixed.\footnotemark[1]

Organizational details

The storage arrangement of the coefficient matrix $A$ is best illustrated by a simple example. Consider the $(6 \times 6)$ symmetric matrix:

$$A = \begin{bmatrix}
  a_{11} & a_{12} & a_{13} & 0 & 0 & 0 \\
  a_{21} & a_{22} & 0 & 0 & 0 & 0 \\
  a_{31} & 0 & a_{33} & 0 & 0 & 0 \\
  0 & a_{44} & a_{46} & a_{46} & 0 & 0 \\
  \text{symm.} & a_{55} & a_{56} & a_{56} & a_{56} & 0 \\
  0 & 0 & 0 & a_{66} & a_{66} & a_{66}
\end{bmatrix}$$

$\mathbf{b}_f$ is often called the constraint reaction vector.

\footnotetext[1]{If the number $n_c$ of constrained equations is significant (say $n_c \geq 0.3 n$), it may be computationally advantageous to preorder (1) into the form (4) before presenting it to SKYFAC-SKYSOL.

in which entries outside the skyline template are not shown. Matrix (6) is stored as a 15-work template $A$:

$$A: a_{11}, a_{22}, a_{13}, 0, a_{33}, a_{14}, a_{36}, a_{55}, a_{16}, 0, 0, a_{46}, a_{66}, a_{66}.$$ 

This array is complemented by a $(n + 1)$-word array $LD$ containing diagonal location pointers:

$$LD: 0, 1, 2, 5, 8, 9, 15.$$ 

For a general matrix, $LD(1 + n)$ points to $a_{ii}$ in array $A$ for $i = 1, \ldots, n$, whereas $LD(1)$ contains zero.

During the factorization process, elements of the strict upper triangle of $U$ replace the corresponding super-diagonal entries of $A$, and the reciprocals of the diagonal elements of $D$ overwrite the diagonal elements of $A$. Thus the factorization (2) of the example matrix (6) would be arranged as follows:

$$\begin{bmatrix}
  1/d_{11} & u_{12} & u_{14} & u_{16} \\
  1/d_{22} & u_{22} & u_{24} & u_{26} \\
  1/d_{33} & u_{33} & u_{36} & u_{36} \\
  1/d_{44} & u_{44} & u_{46} & u_{46} \\
  1/d_{55} & u_{55} & u_{56} & u_{56} \\
  1/d_{66} & u_{66} & u_{66} & u_{66}
\end{bmatrix}$$

Constrained equations are flagged by a negative $LD$ pointer. For instance, if $x_3$ and $x_5$ are to be prescribed in the linear system with coefficient matrix (6)

$$LD: 0, 1, 2, -5, 8, -9, 15.$$ 

In this case, the decomposition (2) is performed only on the unconstrained equations of (4), i.e. $A_g$ is replaced by $D_g$ and $U_g$. Constrained rows and columns are not altered.

Decomposition failure

The factorization process (1) may be aborted if one of the following conditions is encountered.

1. Singularity: matrix $A$ ($A_g$ if constrained) appears singular to machine accuracy. The singularity test performed at the $j$-th elimination stage is $|11, \text{Contr. I/7}|$:

$$d_i < t_i = 8\epsilon r$$

where $d_i$ is the $j$-th diagonal entry of $D$, $\epsilon$ the smallest positive floating-point number for which $1 + \epsilon > 1$ on the computer being used, and $r$ is the Euclidean norm (length) of the $j$-th row of $A$ excluding constrained columns.

2. Indefiniteness: matrix $A$ ($A_g$ if constrained) appears to be indefinite and a check on positive definiteness is requested by the user. The $j$-th stage test is:

$$d_i < -t_i$$

Note: The numerical stability of the symmetric factorization (3) is not guaranteed if $A$ is not positive definite. However, in many practical situations, such as mixed finite element analysis$[12]$, a stable symmetric factorization of an indefinite matrix is guaranteed provided some mild restrictions on the arrangement of nodal equations are observed.
Solution accuracy estimation

Two optional mechanisms for assessing the accuracy of computed solutions are provided:

1. Matrix condition evaluation (a priori estimate). A lower bound estimate of the Euclidean condition number of A\[13, Sec. 3.11\]:

\[ C(A) = \|A\|_\infty \|A^{-1}\|_\infty \]  

where \( \| \cdot \|_\infty \) denotes the Euclidean matrix norm, is returned by SKYFAC on request. The estimation is based on performing one cycle of iterative refinement on a specified number \( m \) of 'random' right hand side vectors \( b \), and applying equation (13.3) of chapter 3 of [13]. The larger \( m \), the better the estimate, but in practice \( m = 2 \) is often sufficient. If \( C(A) \leq 1, \log_{10} C(A) \) may be used as a gross estimate of the number of significant digits expected to be lost in the solution process.

2. Iterative refinement of solution (a posteriori estimate). Given the initial computed solution \( x_0 \) of (1), a \( k \)-cycle refinement process is defined by the algorithm [11, Contr. I/2]:

\[
\begin{align*}
  r_i &= b - A x_{i-1} \\
  LDU A x_i &= r_i & i &= 1, \ldots, k \\
  x_i &= x_{i-1} + \Delta x_i 
\end{align*}
\]

(14)

where only the calculation of the residual \( r_i \) has to be carried out in higher precision arithmetic. Given \( k > 0 \), subroutine SKYSOL returns \( x_k \) and the relative error

\[ \delta_k = \| \Delta x_k \|_\infty / \| x_k \|_\infty. \]  

(15)

If \( \delta_k < 1 \), \( \delta_k \) provides an estimate of the accuracy of \( x_{k-1} \). Note: In practice it is pointless to specify \( k > 1 \) unless the entries of \( A \) and \( b \) are exact within their single precision representation (as it happens, for instance, in some finite difference schemes on regular grids), since the sequence \( x_k \) will not generally converge to that solution of system (1) associated with infinite precision representation of \( A \) and \( b \).

Partial factorization

Subroutine SKYFAC allows the user to specify that the decomposition of \( A \) is to be performed from equation (NBEG + 1) through NEND, where NBEG, NEND are input parameters. The usual complete factorization of \( A \) is achieved by setting NBEG = 0 and NEND = matrix order. A partial factorization may be requested by suitably constraining that range. For instance, if matrix (6) is submitted to SKYFAC with NBEG = 0 and NEND = 3, the output is

\[
\begin{bmatrix}
  1/d_{11} & u_{13} & a_{14} \\
  1/d_{12} & u_{23} & a_{24} & 0 \\
  1/d_{13} & a_{34} & 0 & 0 \\
  a_{44} & a_{45} & a_{46} \\
  a_{55} & a_{56} \\
  a_{66}
\end{bmatrix}
\]  

(16)

A subsequent call with NBEG = 3 and NEND = 6 would produce (9).

This capability is useful in the following cases:

1. The calling program may wish to examine factorization of principal minors (e.g. in local stability analysis of physical systems).

2. Problems involving local nonlinearities, if these can be confined to the bottom of the discretization operators. Then the (linear) upper portion of \( A \) can be factorized once and for all during the nonlinear iterative solution process (a similar situation occurs in the implicit time integration of locally nonlinear dynamic systems).

Usage

The routines presented here include the equation solver proper (SKYFAC-SKYSOL) and a test package (SKYTEST). Relationships among the various routines are illustrated in Fig. 1.

Documentation for using subroutines SKYFAC and SKYSOL is embodied in the source listing. Both subroutines utilize an external, user-supplied vector inner product function whose name is transferred in the formal argument list. Fortran samples of such function are provided in the listing, but for maximum efficiency the function should be handcoded in assembly language (many computer installations have such a procedure in the utility library).

The test program SKYTEST is intended to illustrate the use of SKYFAC-SKYSOL, and to certify their implementation on specific computers. SKYTEST uses four auxiliary routines: SKYLDU, SKYPORC, SKYMAP and SKYPRT. The last two are likely to be useful on their own for matrix display purposes. Guidelines for setting up and running the verification program are provided in the comment block heading SKYTEST. A test example produced by the test program on the UNIVAC 1108 is also given.

Program modifications

Only one machine-dependent constant is utilized in this package: the floating-point precision constant \( e \) denoted by EPSMAC in subroutine SKYFAC. Appropriate values of \( e \) for various computers are indicated there.

Conversion of SKYFAC-SKYSOL to operate in full double precision is straightforward: all REAL declarations should be changed to DOUBLE PRECISION (or REAL*8 on IBM 360/370 models), or appropriate IMPLICIT statements inserted. If this is done, the optional iterative refinement mechanism loses meaning.
THE UPPER TRIANGLE OF THE INPUT SYMMETRIC MATRIX (A) IS STORED IN COLUMN-WISE "SKYLINE" FASHION IN THE FIRST NWA WORDS OF
ARRAY A. THE STORAGE SCHEME IS BEST ILLUSTRATED BY THE (4 X 4) MATRIX (DOTS DENOTE ZEROS OUTSIDE SKYLINE):

\[
\begin{bmatrix}
A_{11} & A_{12} & A_{13} & A_{14} \\
A_{21} & A_{22} & A_{23} & A_{24} \\
A_{31} & A_{32} & A_{33} & A_{34} \\
A_{41} & A_{42} & A_{43} & A_{44}
\end{bmatrix}
\]

IN ARRAY A, THE POSITION OF THE DIAGONAL ELEMENTS IS MARKED BY A (N+1)-WORD INTEGER ARRAY LD, SUCH THAT LD(I) = I AND
LD(I+1) POINTS TO A(I,I) IN ARRAY A FOR I = 1:N.
FOR THE EXAMPLE MATRIX (1):

\[
LD = 0, 1, 2, 5, 8, 9, 15
\]

THE SYMMETRIC DECOMPOSITION

\[
A_1 = L_1 D_1 U_1
\]

WHERE (U_1) IS THE TRANSPOSE OF (L_1), WHICH IS A UNIT LOWER
TRIANGULAR MATRIX AND (D_1) IS A DIAGONAL MATRIX. IT IS PERFORMED,
THE STRICT UPPER TRIANGLE OF (D_1) DIAGONALIZES THE OFF-DIAGONAL
ENTRIES OF (A), WHEREAS THE RECIPROCALS OF THE ELEMENTS OF
(D_1) REPLACE THE DIAGONAL ELEMENTS OF (A). THE OUTPUT
STORAGE SCHEME FOR THE EXAMPLE MATRIX (1) IS THEN:

\[
\begin{bmatrix}
I & U_{11} & U_{12} & U_{13} \\
I & U_{21} & U_{22} & U_{23} \\
I & U_{31} & U_{32} & U_{33} \\
I & U_{41} & U_{42} & U_{43}
\end{bmatrix}
\]

THE DECOMPOSITION PROCESS MAY BE ABORTED UNDER EITHER OF THE
FOLLOWING CONDITIONS:

(A) THE MATRIX (A_1) MODIFIED BY THE ROUNDING ERRORS, APPEARS
SINGULAR. THE SINGULARITY TEST USED AT THE J-TH STEP IS

\[
R(J,J) \leq EPSMAC*P(J)
\]

WHERE R(J,J) DENOTES THE EUCLIDEAN NORM OF THE J-TH ROW OF
(A), AND EPSMAC IS THE SMALLEST POSITIVE FLOATING-POINT
NUMBER SUCH THAT 1 + EPSMAC \approx 1. IN THE COMPUTER
BEING USED, THIS SINGULARITY TEST IS ONLY ENFORCED IF
INPUT FLAG SIMBXP IS TRUE. (SEE INPUT ARGUMENTS).

(B) THE MATRIX (A_1) IS INDEFINITE. I.E.,

\[
D(J,J) \leq 0
\]

THE POSITIVE DEFINITENESS TEST IS ONLY ENFORCED IF INPUT
FLAG PCHEX IS TRUE. (SEE INPUT ARGUMENTS). IT SHOULD BE
STRESSED THAT THE NUMERIC STABILITY OF THE DECOMPOSITION
IS NOT GUARANTEED IF MATRIX (A) IS INDEFINITE.

THIS SUBROUTINE MAY ALSO BE UTILIZED TO HANDLE THE MORE
GENERAL CASE OF A CONSTRAINED LINEAR SYSTEM, IN WHICH SOME OF
THE X-ENTRIES ARE PRESCRIBED. A CONSTRAINED SYSTEM MAY BE
FORMALLY WRITTEN IN THE PARTITIONED MATRIX FORM:

\[
\begin{bmatrix}
A_{ff} & A_{fd} & A_{fc} \\
A_{df} & d_{f} & c_{d} \\
A_{cf} & c_{f} & c_{c}
\end{bmatrix}
\]

\[
\begin{bmatrix}
V_{ff} & V_{fd} & V_{fc} \\
V_{df} & V_{df} & V_{dc} \\
V_{cf} & V_{cd} & V_{cc}
\end{bmatrix}
\]

\[
\begin{bmatrix}
X_{ff} & X_{fd} & X_{fc} \\
X_{df} & X_{df} & X_{dc} \\
X_{cf} & X_{cd} & X_{cc}
\end{bmatrix}
\]

\[
\begin{bmatrix}
B_{ff} & B_{fd} & B_{fc} \\
B_{df} & B_{df} & B_{dc} \\
B_{cf} & B_{cd} & B_{cc}
\end{bmatrix}
\]

\[
\begin{bmatrix}
A_{ff} & A_{fd} & A_{fc} \\
A_{df} & d_{f} & c_{d} \\
A_{cf} & c_{f} & c_{c}
\end{bmatrix}
\]

\[
\begin{bmatrix}
V_{ff} & V_{fd} & V_{fc} \\
V_{df} & V_{df} & V_{dc} \\
V_{cf} & V_{cd} & V_{cc}
\end{bmatrix}
\]

\[
\begin{bmatrix}
X_{ff} & X_{fd} & X_{fc} \\
X_{df} & X_{df} & X_{dc} \\
X_{cf} & X_{cd} & X_{cc}
\end{bmatrix}
\]

\[
\begin{bmatrix}
B_{ff} & B_{fd} & B_{fc} \\
B_{df} & B_{df} & B_{dc} \\
B_{cf} & B_{cd} & B_{cc}
\end{bmatrix}
\]
Solution of linear equations with skyline-stored symmetric matrix

* WHERE [BF] AND [DF] ARE GIVEN AND [DF] AND [AC] ARE TO BE DETERMINED. COEFFICIENTS 'F' AND 'C' STAND FOR 'FREE' AND 'CONSTRAINED', RESPECTIVELY. HOWEVER, IT MUST BE EMPHASIZED THAT (A) NEED NOT BE PHYSICALLY ARRANGED TO CONFORM TO (7).

* IF X(1) IS TO BE PRESCRIBED, THE 1-TH ROW AND COLUMN OF [A] ARE TO BE CONSIDERED. SUCH A CONDITION IS MARKED BY A NEGATIVE VALUE OF LD(1)=1. FOR INSTANCE, IF N=5 AND 6 OF THE EXAMPLE MATRIX (7) ARE CONSTRAINED, THE LD ARRAY MUST BE:

```
LD = 0, 1, 2, -5, 0, 9, -15
```

* CONSTRAINED ROWS AND COLUMNS ARE IGNORED IN THE FACTORIZATION OF A CONSTRAINED MATRIX AND VALUES THEREIN RETURN UNALTED. FOR INSTANCE, THE OUTPUT FROM SKYFAC FOR THE EXAMPLE MATRIX (7) WITH THE LD ARRAY (7A) WOULD BE:

```
   1  0  0  0  0
   -5 0  0  0  0
   2  0  0  0  0
   -15 0  0  0  0
   0  0  0  0  0
   0  0  0  0  0
   0  0  0  0  0
```

* THAT IS: ONLY THE (UNREARRANGED) FREE-FREE PORTION (DF) IS EFFECTIVELY FACTORIZED INTO (LPF) (DF) (DF).

* ADDITIONAL FEATURES OF THIS ROUTINE INCLUDE:

1. THE DETERMINANT OF (A) OR (DF) IF CONSTRAINED MAY BE COMPUTED ON REQUEST.

2. AN ESTIMATE OF THE EUCLIDEAN CONDITION NUMBER OF (A) OR (DF) IF CONSTRAINED MAY BE OBTAINED.

3. ACCUMULATION OF INNER PRODUCTS IN DIFFERENT PRECISION MODULI IS FACILITATED BY INCLUDING THE INNER PRODUCT IN THE FORMAL ARGUMENT LIST. VARIOUS ENTRY POINT NAMES MAY BE USED WITHOUT NEED FOR EDITING-RECOMPILING.

4. A CAPABILITY FOR PARTIAL FACTORIZATION IS INCLUDED. THE FACTORIZATION (6) IS ACTUALLY CARRIED OUT FROM EQUATION (NBEG=1) THROUGH NEND WHERE NBEG NEND ARE TWO INPUT PARAMETERS, NORMALLY NBEG=0 AND NEND=I (MATRIX ORDER), HOWEVER, IN CERTAIN CASES (E.G., LOCAL STABILITY ANALYSIS, LOCAL NONLINEARITIES), A PARTIAL FACTORIZATION MAY BE USEFUL.

*----------

* ** US E R A G E **

*----------

**** 1. INPUT ARGUMENTS ****

- **NBEG** HEND = TWO PARAMETERS SPECIFYING FACTORIZATION RANGE.

AC: EXTENDING FROM EQUATION NBEG=1) THROUGH NEND. (SEE DISCUSSION). COMPLETE FACTORIZATION OF (A) IS OBTAINED WITH NBEG=0 AND NEND=I (MATRIX ORDER).

- **N** ORDER OF MATRIX (A).

- **LD** (N+1)-WORD ARRAY OF DIAGONAL LOCATION POINTERS (SEE DISCUSSION).

- **V** A SCRATCH FLOATING-POINT ARRAY OF LENGTH AT LEAST N IF ICOND = 0; 4N IF ICOND IS POSITIVE.

- **DOTPRD** FUNCTION RETURNING INNER (DOT, CHFURP) PRODUCT OF TWO VECTORS. THIS FUNCTION IS INVOKED BY DOTPRD(P,Q,N) AND SHOULD EVALUATE THE INNER PRODUCT OF THE TWO N-WORD VECTORS P AND Q.

- **SINGRB** LOGICAL FLAG CONTROLLING ACTION TO BE TAKEN IF THE SINGULARITY TEST (5) IS VERIFIED:

  - **SINGRB** = .TRUE. ABORT SETTING IFAIL = J
  - **SINGRB** = .FALSE. SET D(J) TO ZERO AND PROCEED.

- **IORD** FOR NORMAL USE OF SKYFAC IN EQUATION SOLVING, ONE SHOULD ALWAYS SET SINGRB = .TRUE. THE FALSE SETTING IS RECOMMENDED ONLY IF THE DECOMPOSITION (3) IS TO BE USED IN THE CALCULATION OF EIGENVECTORS OF (DF) BY INVERSE POWER ITERATION.

- **PCHECK** LOGICAL FLAG INDICATING WHETHER (A) OR (DF) IF CONSTRAINED IS TO BE CHECKED FOR POSITIVE DEFINITENESS I.E., CONDITION (6) TESTED.

  - **PCHECK** = .TRUE. ABORT SETTING IFAIL = J
  - **PCHECK** = .FALSE. DON'T TEST FOR INDEFINITENESS

- **ICOND** IF POSITIVE, A LOWER BOUND ESTIMATE OF THE EUCLIDEAN CONDITION NUMBER (AND) OF (A) OR (DF) IF CONSTRAINED WILL BE CALCULATED BY PERFORMING ONE ITERATIVE REFINEMENT CYCLE ON ICOND "ARRANGING" THE VECTORS. THE LARGER ICOND, THE BETTER THE ESTIMATE. HOWEVER, FOR PRACTICAL, A PRIORI ASSESSMENT OF SOLUTION ACCURACY, ICOND = 2 IS OFTEN SUFFICIENT.
C. A. Felippa

C* IFILE IF NONZERD, COPIES OF (A) AND ITS FACTORIZATION WILL BE WRITTEN IN LOGICAL UNIT IFILE (SEE 'OUTPUT FILES' BELOW). IF ZERO, NO COPIES ARE WRITTEN. A NONZERD IFILE IS REQUIRED IN THE FOLLOWING TWO CASES:
C* (A) ICOND IS POSITIVE (C(A) REQUESTED)
C* (B) ITERATIVE REFERENCE OF SOLUTION VECTORS WILL BE REQUESTED WHEN CALLING SKYOL.
C* WARNING - DO NOT USE FEATURES (A) OR (B) IF THE PARTIAL FACTORIZATION OPTION IS BEING EXERCISED.
C***** 2. INPUT-OUTPUT ARGUMENTS -
C* A ARRAY CONTAINING INPUT MATRIX TO BE FACTORIZED (OR UNFACTORIZED PORTION THEREOF IF NBEG GT 0)
C* OUTPUT FACTORIZATION-DEFINITION ELEMENTS, AS DESCRIBED IN THE DISCUSSION.
C***** 3. OUTPUT ARGUMENTS -
C* ICOND ESTIMATE OF C(A) IF ICOND GT 0 (AND IFAIL = 0)
C* DETC+IDETX ND NUMBER: REAL INTEGER WHICH TOGETHER GIVE THE DETERMINANT OF (A) OR (AFF) IF (A) IS COMPLAINED: IN THE FORM:
C* DETC = DETC * 2**IDETX
C* WHERE DETC IS IN THE RANGE [-1.0, 1.0], PROVIDES DETC IS PRESET TO 1.0 ON INPUT, IF DETC IS ZERO ON INPUT, THE DETERMINANT IS NOT CALCULATED.
C* IF OUTPUT IFAIL = -. J, DETC IS SET TO ZERO.
C* IF OUTPUT IFAIL = -. J (.0 OF 1), THE ULTIMATE
C* (-1.0) IS THAT OF THE UPPER (.U) PRINCIPAL MINOR.
C* NEGEIG COUNT OF NEGATIVE EIGENVALUES OF (A) OR (AFF) IF (A) IS COMPLAINED: IF IFAIL = 0 ON OUTPUT.
C* IFAIL FACTORIZATION FAILURE INDICATOR:
C* IFAIL= 0 FACTORIZATION SUCCESSFULLY EXECUTED
C* IFAIL= J SIMPLIARDY CONDITION (J) DETECTED
C* IFAIL=-J INDEFINITE CONDITION (J) DETECTED
C***** 4. COMMON INPUTS -
C* NONE.
C***** 5. COMMON INPUT-OUTPUTS -
C* NONE.
C***** 6. COMMON OUTPUTS -
C* NONE.
C***** 7. INPUT FILES -
C* NONE.
C***** 8. OUTPUT FILES -
C* IFILE IF NONZERD, LOGICAL UNIT IFILE RECEIVES COPIES OF THE ORIGINAL AND OUTPUT CONTENTS OF ARRAY A. SAVED AS 2 FORTRAN BINARY RECORDS OF SIZE LD(N+1).
C***** 9. SCRATCH FILES -
C* NONE.
C***** 10. PROGRAM SUBROUTINES -
C* DOTFPD, SYRD, SKYOL.
C***** II. LIBRARY SUBROUTINES -
C* NONE.

C TYPE AND DIMENSION STATEMENTS
C REAL A(I,J), A(I),/1) REAL D, DETC, DOTFPD, EPSM9C INTEGER LD(/1) LOGICAL PICKER, SIMAB
C EQUIVALENCE (A, D, .I,1)
C
C NOTE - APPROPRIATE VALUES OF EPSM9C FOR VARIOUS COMPUTERS ARE:
C CDC 6000-7000 SERIES 7.11E-15 (SINGLE PRECISION)
C IBM 360/370 SERIES 9.54E-07 (REAL4 PRECISION)
C IBM 360/370 SERIES 2.25E-16 (REAL8 PRECISION)
C UNIVAC 1108/110; IBM 7094 1.49E-08 (SINGLE PRECISION)
C DATA EPSM9C /1.49E-08/
C
C INITIALIZATION
C IFAIL = 0 IDETEX = 0 NEGEIG = 0
C SAVE ORIGINAL MATRIX IF IFILE NE 0 AND NBEG = 0
C NAR = IABS(LD(/1))
C IF (IFILE.EQ.0) GO TO 200 IF (NBEG.NE.0) GO TO 200
C PRINTING IMPLD
C WRITE (IFILE) (A(I,J),J=1,NAR)
Solution of linear equations with skyline-stored symmetric matrix

C COMPARE SQUARED LENGTHS OF UNCONSTRAINED ROWS NEGE1 THRU N

C 200 NEGE1 = NEGE + 1
C DO 1000 I = NEGE1,N
C II = LD(I,I)
C IF (II) 1000,1000,400
C 400 V(I) = M(I)**2
C M = II - 1
C K = MAX0(NEGE1,INBS(LD(I),M+1))
C L = MIN0(NEND,I)-1
C IF (L) 500,500,1000
C 500 DO 800 J = K,L
C IF (LD(J,I)) 800,800,600
C V(J) = V(I) + M(I,2)
C V(J) = V(J) + M(I,2)
C 800 CONTINUE
C 1000 CONTINUE
C
C FACTORIZATION SECTION
C DO 4000 J = NEGE1,NEND
C COMPUTE SUPERDIAGONAL ENTRIES OF J-TH COLUMN OF (U)
C IF UNCONSTRAINED
C JJ = LD(J+1)
C IF (JJ) 4000,4000,1200
C 1200 D = A(J,J)
C NMJ = INBS(LD(J,J))
C NR = JJ - 1
C KU = JK - 1
C IF (KU(0,0)) 600 TO 2200
C DO 2000 K = 1,KU
C I = J - JY + K
C V(K) = 0.0
C II = LD(I+1)
C IF (II) 2200,2200,1800
C 1800 CONTINUE
C 2200 CONTINUE
C COMPUTE DIAGONAL ELEMENT D(J)
C D = D - DOTPRD(A(J,J),V(K))
C SINGULARITY TEST
C TOLROW = 8.0*EPSMRC*SQR1 R(J)>
C IF (ABS(D),GT,TOLROW) GO TO 2500
C IF (SINGAB) GO TO 6000
C 2500 R(J) = 1.0/D
C UPDATE DETERMINANT IF DETCF IS NONZERO
C 3200 IF (DET CF,LE,0.0) GO TO 3500
C DETCF = DETCF*D
C 3200 IF (ABS(DET CF),LT,1.0) GO TO 3400
C DETCF = DETCF*EPSMRC
C IDEX = IDEX + 4
C GO TO 3200
C 3400 IF (ABS(DET CF),GE,0.0625) GO TO 3500
C DETCF = DETCF*16.
C IDEX = IDEX - 4
C GO TO 3400
C POSITIVE DEFINITNESS CHECK (IF PDCHK = .TRUE.)
C 3500 IF (D,GT,0.0) GO TO 4000
C NEGE1 = NEGE1 + 1
C IF (PDCHK) GO TO 6500
C 4000 CONTINUE
C SAVE FACTORIZATION IF IAFILE NE 0 AND N = NEND
C IF (IAFILE,LE,0) GO TO 5000
C IF (NEND,LE,IA) GO TO 5000
C WRITE (IAFILE) (R(J),J=1,IA),V(K)
C IF (ICONO.EQ.0) GO TO 5000
C MATRIX CONDITION ESTIMATION
C K = 0
C DMRX = 0.0
C DO 4200 I = 1,ICOND
C 4200 CONTINUE
C NOW ESTIMATE COV(A) FROM LARGEST DELTA AND MACHINE PRECISION
C ACOND = DMRX/<(1.0+DMRX)*EPSMRC>
C 5000 RETURN
SUBROUTINE SKYSOL (A, N, LD, DOTFPD, IDF, IBX, B, X)

* IFREF, IFFILE, V, DELTA

**---------------------------------------------------------------
** PURPOSE
**---------------------------------------------------------------
** TO SOLVE THE LINEAR SYSTEM (A) (X) = (B), (A) BEING A SKYLINE-
** STORED SYMMETRIC MATRIX PREFACTORED BY SUBROUTINE SKYFAC.
**---------------------------------------------------------------
** PROGRAMMED - CARLOS A. FELIPPA  MARCH 1974.
** UPDATE - MAY 1974.
** LANGUAGE - FORTAN IV (FAC) EXCEPT FDP OCCASIONAL
** (REAL)-TYPE SUBSCRIPTS.
** EQUIPMENT - MACHINE INDEPENDENT

---------------------------------------------------------------
** DISCUSSION
**---------------------------------------------------------------
** SKYSOL CAN DEAL EITHER WITH THE UNCONSTRAINED LINEAR SYSTEM
** [(A) (X)] = (B)
** WHERE THE ENTIRE MATRIX [A] IS PRESCRIBED AND [X] HAS BEEN FDE-
** FACTORED INTO [L] [L] [U] [D] BY SKYFAC OR WITH THE MORE GENERAL
** CASE OF A CONSTRAINED SYSTEM (CF, EO, +) IN SKYFAC:
** [(A) (X)] + [(O)] + [(E)] = [(B)]
** IS FACTORED INTO [L] [L] [U] [D] BY SKYFAC.
** (A) IS THE CONSTRAINED CASE WHERE THE UNCONSTRAINED ONE. THE
** RULES DESCRIBED BELOW ASSUME A CONSTRAINED SYSTEM.

---------------------------------------------------------------
** SOLUTION PROCESS IN SKYSOL CONSISTS OF UP TO SIX STEPS
** 1. PAS MODIFICATION: [A] IF REPLACED BY [E] - [(O)] EXC.
** THIS STEP IS SKIPPED IF THE LINEAR SYSTEM IS UNCONSTRAINED
** OR HOMOGENEOUSLY CONSTRAINED (EO = 0).
** 2. FORWARD REDUCTION: THE TRIANGULAR SYSTEM [L] [F] = [B]
** IS SOLVED FOR [F]
** 3. SCALING: THE DIAGONAL SYSTEM [D] [F] = [F] IS SOLVED
** 4. BACKSUBSTITUTION: THE TRIANGULAR SYSTEM [U] [D] [F] = [O]
** IS SOLVED FOR [D]
** 5. ITERATIVE REFINEMENT: IF ARGUMENT IFREF GT 0, THE RESIDUAL
** IS COMPUTED USING HIGHER PRECISION ARITHMETIC, AND STAGES 2-4 REPEATED TO
** SOLVE [U] [D] [F] = [D] (NOTE THAT [E] = [O]). THE
** CORRECTION [IF] IS ADDED TO [F] AND THE PROCESS REPEATED
** IFTimes.
** 6. CONSTRAINED RHS RECOVERY: IF [O] = [O] [D] [F] + [E] [O] [D] [O]
** IS CALCULATED IF REQUESTED (SEE INPUT ARGUMENT IBX).

**---------------------------------------------------------------
** USAGE
**---------------------------------------------------------------
** 1. INPUT ARGUMENTS
** A         OUTPUT MATRIX FACTORS PRODUCED BY SKYFAC
** N         SYSTEM ORDER
** LD        (M+1)-WORD ARRAY OF DIAGONAL LOCATION POINTERS
** DOTFPD    ENTRY POINT OF INNERPRODUCT FUNCTION (CF, SKYFAC)
** IDF       UPDIAGNUM LOWER RAM PLACES
** IBX        FLAG CONTROLLING OUTPUT CONFIGURATION OF ARRAY B.
** IBX = 0 : B AND X ARE IDENTIFIED IN THE CALLING
** PROGRAM, THE SOLUTION VECTOR X THEN OVERWRITES B.
SOLUTION OF LINEAR EQUATIONS WITH SKYLINE-STORED SYMMETRIC MATRIX

THIS OPTION MAY BE USED ONLY IF

(A) THE SYSTEM IS UNCONSTRAINED, OR HOMOGENEOUSLY

CONSTRAINED WITH NO RECOVERY OF B(C) DESIRED.

(B) NO ITERATIVE REFINEMENT REQUESTED (REF=0)

IBX = -1 : B AND X ARE NOT IDENTIFIED IN THE
CALLING PROGRAM BUT COMPUTATION OF B(C) IS NOT
DESIRED.

IBX = 1 : B AND X ARE NOT IDENTIFIED IN THE
CALLING PROGRAM AND RECOVERY OF B(C) IS REQUIRED.

NOTE - IF THE INPUT SYSTEM CONTAINS NONHOMOGENEOUS
CONSTRAINTS, IBX MUST BE 1 OR -1, I.E., ARRAYS
B AND X MAY NOT SHARE THE SAME STORAGE SPACE.

REF

IF GT 0, PERFORM IREF CYCLES OF ITERATIVE
REFINEMENT OF THE SOLUTION VECTOR, IN PRACTICE
ONE SHOULD NOT REQUEST IREF GT 1 UNLESS THE
ENTRIES OF B(C) HAPPEN TO BE EXACT NUMBERS

FILE

LOGICAL UNIT CONTAINING COPIES OF B(C) AND ITS
FACTORIZATION (SEE SKYFAC), REQUIRED IF IREF GT 0

V

SCRATCH FLOATING-POINT ARRAY OF LENGTH 2*N
TO BE USED IN ITERATIVE REFINEMENT PROCESS.
MAY BE A DUMMY ARGUMENT IF IREF = 0.

***** 2. INPUT-OUTPUT ARGUMENTS -
B
ON INPUT, ARRAY B CONTAINS PRESCRIBED COMPONENTS
OF (B) AND (C), I.E., IF THE I-TH EQUATION IS FREE
<CONSTRAINED>, I.I. STORES THE RHS (LSB) VALUE.
THE OUTPUT CONFIGURATION OF B IS CONTROLLED BY
INPUT FLAG IBX (SEE INPUT ARGUMENTS).

**** 3. OUTPUT ARGUMENTS -
X
COMPUTED SOLUTION VECTOR
DELTA
IF IREF GT 0, RATIO OF LAST CORRECTION LENGTH
TO FINAL SOLUTION LENGTH, NOT USED IF IREF = 0.

***** 4. COMMON INPUTS -
NONE.
***** 5. COMMON INPUT-OUTPUTS -
NONE.
***** 6. COMMON OUTPUTS -
NONE.
***** 7. INPUT FILES -
FILE SEE SUBROUTINE SKYFAC.
***** 8. OUTPUT FILES -
NONE.
***** 9. SCRATCH FILES -
NONE.
***** 10. PROGRAM SUBROUTINES -
DOTPRD, SKYFAC
***** 11. LIBRARY SUBROUTINES -
SGRT, FORTRAN BINARY I/O

*** TYPICAL DIMENSION STATEMENTS

INTEGER LD(I)
REAL A(I), B(I), B(C), EPS, DELTA
EQUIVALENCE (B(I), EPS, DELTA)

INITIALIZATION

IF IOP.GT.0, 130 TO 1800
IF IBX.EQ.0 50 TO 1100

RHS MODIFICATION

EQ 1000 I = 1+N
I = B(LD(I)+1)
300 IF (I) B(I)
300 IF (B(J)) GO TO 1000
1000 J = 11
K = I - 11 = IABS(LD(I)) + 1
9000 J = K+N
J5 = B(LD(J))
9000 M = J - I
5000 M = R(I+1)
6000 IJ = J - M

BEGIN PROGRAM

INPUT ARGUMENTS

FREE = 0
EPS = 0.0
IF (IBX.EQ.0) GO TO 200
EPS = 1.0
150 I = 1+N
200 IF (IDP.GT.0) GO TO 1000
200 IF (IBX.EQ.0) GO TO 1100

DATA 3100 TO END

GO TO 1000

SUBROUTINE SKYFAC

DO 999 I = 1,N
999 RETURN

C. A. BLIPPA

IF (I-J-1)/2 > 1000 CONTINUE
C
C FORWARD SUBSTITUTION PASS
C
1100 DO 1500 I = 1,N
   1200 X(I) = 0.0
   1300 IM1 = IHBS(LD(I+1))
   1400 M = I - IM1 - 1
   1500 CONTINUE
IF (IPR.NE.0) GO TO 5000
C
C CALING FACE
C
1900 DO 2000 I = 1,N
   2000 X(I) = M(I+1) * (I)
C
C BACK SUBSTITUTION PASS
C
I = N
DO 2200 N = 1,N
   2200 X(I) = X(I) * BAC(I)
   2200 CONTINUE
GO TO 2800
X(I) = 0.0
CONTINUE
2800 I = I - 1
3000 CONTINUE
IF (IPR.LE.0) GO TO 4000
GO TO NEXT, 3500, 3100.
C
C ITERATIVE REFINEMENT SECTION
C
3100 REF = XREF + 1
IF (IPR-REF) GO TO 3200
C
C CALCULATE RESIDUAL VECTOR (R) = (A) (X) USING SKYLINE.
CR) RETURNS IN R, AND LDL (R) IN V
C
3200 REWIND IFILE
MAR = IHBS(LD(N+1))
READ (IFILE) (R(J),J=1,N)
CALL SKYLINE (A, M, LD, X, V, X, N)
C
C SOLVE FOR CORRECTION (B) WHICH APPEARS IN X, CORRECT OLD SOLUTION AND EVALUATE 2-NORM RELATIVE ERROR DELTA
READ (IFILE) (R(J),J=1,N)
BFACTOR = 0.0
HF = NORM (R) / NORM (A)
DO 3500 J = 1,N
   3500 B(J) = BFACTOR * R(J)
   3500 CONTINUE
DO 3300 J = 1,N
   3300 V(J) = V(J) - B(J)
   3300 CONTINUE
C
C CONstrained RHS RECOVERY
C
4000 IF (IPR.LE.0) GO TO 5000
   4100 I = I(J+1)
   4200 B(J) = BFACTOR * R(J)
   4300 CONTINUE
4400 R(I) = R(I) + X(I) * B(J)
4600 CONTINUE
5000 RETURN
C
END

SUBROUTINE SKYLINE (A, M, LD, X, R, IOP, BFACTOR, HFACTOR)
C
DOUBLE PRECISION A(M,N), R(M,N), BFACTOR, HFACTOR
INTEGER M, N, LD, IOP, I

THIS SUBROUTINE POSTMULTIPLIES SKYLINE-STORED MATRIX (A) BY
VECTOR (X). THE RESULT (R) IS PRODUCED IN DOUBLE PRECISION.
FURTHER OPERATIONS ARE CONTROLLED BY INPUT FLAG IOP :
IOP = 0 EXIT AFTER OBTAINING (R)
IOP = 1 FORM RESIDUAL VECTOR (B) = (A) - (R) IN R
IOP = -1 MOVE (X) TO R, AND FORM (B) - (R) IN X.
C
ARRAYS A (M, N) AND R MAY BE EQUIVALENT IN THE CALLING PROGRAM
C
DOUBLE PRECISION A(M,N), R(M,N), X(N), BFACTOR, HFACTOR
INTEGER M, N, LD
REAL A(M,N), B(M,N), X(N), R(M,N)
C
DO 2000 N = 1,M
   M = M + 1
   IF (M.EQ.0) GO TO 2000
   IF (M.EQ.1) GO TO 1500
   DO 1500 K = 1,M
      J = K + 1
      DO 1500 J = K + 1,M
         IF (A(K,J).NE.0) GO TO 1500
   CONTINUE
   1500 CONTINUE
2000 CONTINUE
   CONTINUE
   2500 CONTINUE
   I = I - 1
   IF (I.EQ.0) GO TO 2000
   J = I - 1
   DO 2500 J = I - 1,M
      ALU = A(J,K) + A(J+1,K)
      A(J+1,K) = A(J+1,K) - ALU
   CONTINUE
   2500 CONTINUE
3000 CONTINUE
   DO 3500 K = 1,N
      XI = X(K)
      IF (A(K,I).NE.0) XI = XI + A(K,I) * DBLE02
   CONTINUE
3500 CONTINUE
   RETURN
END
DOUBLE PRECISION DBEX(100)
EXTERNAL VIPSS, VIPSD
INTEGER BLRNK, CMHRK, LD(101)
LOGICAL PDCHEK, SINGRB
REAL R(2000), BEX(100), X(400), XEX(100)
DATA BLANK (/1H), CMARK (=1H), INFLE /2/
DATA IPRT /2/, UPLD/2/
DATA NMAX /100/, NUMAX /2000/
DATA NWIN(1), NUM(100)
C
C READ TEST PARAMETERS FROM CARD FILE (UNIT 5)
100 READ (5,10) ISKY, N, LEVC, ICOND, IREF, IDU
IF (N.LE.0) STOP
WRITE (6,20) ISKY, N, LEVC, ICOND, IREF, IDU
IF (N.GT.NNMAX) GO TO 5200
IRDN = 0
BUILD TEST SYSTEM
C CONSTRUCT DIAGONAL LOCATION POINTER ARRAY IN LD
LD(1) = 0
GO TO 1000
C ISKY = 1: FULL SYMMETRIC MATRIX
400 DO 450 I = 1,N
450 LD(I+1) = LD(I) + I
GO TO 1000
C ISKY = 2: FIXED-BAND MATRIX (HALFWIDTH M = MAXO(0.4*N))
500 M = MAXO(0.4*N,0)
DO 550 I = 1,M
550 LD(I+1) = LD(I) + M
GO TO 1000
C ISKY = 3: RANDOM SKYLINE
600 CALL SKYRDM (N, M, 0.5, IRDM)
C = 0.5*FLOAT(N)
DO 650 I = 1,N
650 K = RNRM(I,C*V(I), 1.000*>)
LD(I+1) = LD(I) + K
1000 NWIN = IRBX(LD(N+1))
IF (NW,5.GT.NWAMRX) GO TO 5400
C MAX APPROXIMATELY LEVC+N/100 CONSTRAINTS
'RANDOMLY' ON LD IF LEVC IS NONZERO
IF (LEVC.EQ.0) GO TO 1500
CALL SKYRDM (N, N, 0.5-0.01*FLOAT(LEVC), IRDM)
DO 1200 I = 1,N
1200 IF (V(I).LT.0.0) LD(I+1) = -LD(I+1)
C
C GENERATE TEST DI AND UI IN ARRAY DU
1500 CALL SKYRDM (DU, N, 0.0, IRDM)
DO 1600 I = 1,N
1600 DU(I) = 1.0*DU(I)
IF (IDU.EQ.0) GO TO 2000
WRITE (6,25)
CALL SKYRPT (DU, N, LD, UPLD/612.66, IPRT)
C ASSEMBLE MATRIX DU = DI*DU
C
C
C GENERATE RANDOM LHS VECTOR IN XEX AND ASSOCIATE RHS VECTOR
BEX = DU*DUEX IN DOUBLE PRECISION
CALL SKYRDM (BEX, N, 0.0, IRDM)
CALL SKYRDL (BEX, N, LD, VIPSS, VIPSD)
C
C SK Y R A C T E S T

C DETCF = 1.0
DUNMY = PDCHEK
CALL SKYRAC (0, N, N, LD, DU, VIPSS, SINGRB, PDCHEK,
* ICOND, INFLE, ACMOD, DETCF, IDETEX, NEGEIG, IFAIL,
TOP = 1.0+OPTIME(1)/FLOAT(KOUNT)
WRITE (6,50) SINGRB, PDCHEK, IFAIL, DETCF, IDETEX, NEGEIG
IF (IPRT.EQ.0) GO TO 100
WRITE (6,54) KOUNT, TOP
IF (ICOND.NE.0) WRITE (6,56) ACMOD
IF (IDU.EQ.0) GO TO 2000
WRITE (6,60)
CALL SKYRDP (BEX, N, LD, UPLD/612.66, IPRT)
C
C S K Y S Y M T E S T

C SET UP SINGLE PRECISION RHS VECTOR B
2000 DO 2500 I = 1,N
2500 BEX(I) = BEX(I)
WRITE (6,70) B(I) = BEX(I)
IF (LD(I+1).EQ.0) B(I) = XEX(I)
2500 CONTINUE
Solution of linear equations with skyline-stored symmetric matrix

CALL SOLVE(LIN, N, LD, VIPSS, 0, 1, 6, IREF, IFILE, 4, DELTA)

WRITE (6,70)
KXNUM = 0.0
KXNORM = 0.0
DO 3200 I = 1,N
KXNORM = KXNORM + 1.0
KX = KX/1.0
KXNORM = KXNORM + KX**2
IF (LD(I),LE,0) KXNORM = 0.0
3200 CONTINUE

WRITE (6,74)
IF (IREF.EQ.1) WRITE (6,75) DELTA
GO TO 100

DIAGNOSTICS FOR ILLEGAL TEST PARAMETERS
WRITE (6,76) NMAX
GO TO 100
WRITE (6,82) NUH, NULAMX
GO TO 100

FORMAT (614,
11H LEVEL = 13, 22H O-O, 1COND, !REF, ID" = I2/22H,,1,
5x 67<1H-)

25 FORMAT ('5A34HTEST MATRIX CR, = CL, CD, >
30 FORMAT ('5A3STEST LINE MQR OF CR :I
50 FORMAT ('5A34HTEST RESULTS '0 = CONSTRAINT EQS) :
70 FORMAT ('5A3HTESTUHL REL. ERROR OF COMPUTED 
16 FORMAT ('5A3HESTIMATED REL. ERROR
e
82 FORMAT (,'5A34H ORDER N = I4r15H EXCEED' NNRX = I4,
86 FORMAT (,'5A34H . . . MATRIX WORDS NWH = 15r17H EXCEEDS NYRX = 15)
END

SUBROUTINE SKYLDL(NBEG, NEND, LD, COUNT)
RETURNS NUMBER OF
PERRITION UNITS (COUNT) REQUIRED TO
FRCTOR h SKYLINE-STORED SYMMETRIC MhTRIX FROM ROW
 NBEG+1 THRU NEND (EACH N-WORD INNERPRODUCT IS
COUNTED AS N OPERATION UNITS)
INTEGER LD(I>
C
DO 4000 J = NBEG+1,NEND
JM = IABS(LD(J))
JJ = LD(J+1)
IF (JJ,LE,0) GO TO 3000
J = JM - JJ
V(J) = 1.0/AFAC(JJ)
3000 CONTINUE
GO TO 4000

SUBROUTINE SKYDPC(NBEG, NEND, LD, COUNT)
C
RETURNS NUMBER OF OPERATION UNITS (COUNT) REQUIRED TO
FACTOR A SKYLINE-STORED SYMMETRIC MATRIX FROM ROW
NBEG+1 THRU NEND SUCH THAT INNERPRODUCT IS
COUNTED AS N OPERATION UNITS
INTEGER LD(I>
C
DO 4000 J = NBEG+1,NEND
JM = IABS(LD(J+1))
JJ = LD(J)
IF (JJ,LE,0) GO TO 3000
J = JM - JJ
GO TO 4000

SUBROUTINE (IMAP (AN M, LD, IUPLO, DEVICE))
C
INTEGER A(M), N, LD, IUPLO, DEVICE
REAL (M,LD) X, Y, Z

DO 1000 J = 1, N
   DO 900 I = 1, J
      K = J - I + 1
      C(J,K) = C(J,K) + A(I,J) - A(J,I)
   END

1000 CONTINUE
RETURN
END

SUBROUTINE (IMAP (AN M, LD, IUPLO, DEVICE))
C
INTEGER A(M), N, LD, IUPLO, DEVICE
REAL (M,LD) X, Y, Z

DO 1000 J = 1, N
   DO 900 I = 1, J
      K = J - I + 1
      C(J,K) = C(J,K) + A(I,J) - A(J,I)
   END

1000 CONTINUE
RETURN
END
Solution of linear equations with skyline-stored symmetric matrix

1000 JREF = 0
C
SET COL PRINT RANGE (JBEG TO JEND) AND LIST COL IDENTIFIERS
1100 JBEG = JREF + 1
JEND = MIN0 (JREF+CMAX(N), N)
K = JEND - JREF
ID 1300 J = JBEG,JEND
I = J - JREF
ICONS(I) = BLANK
IF (LB(I,J)) 1200 1200
1200 IDCONS(I) = CMHYC
1300 CONTINUE
WRITE (*,40, (C(ILLRB,J, IDCONS(J-JREF), J=JBEG, JEND))
40 FORMAT (14X 10(I5),14X(I4))
C
SET ROW PRINT RANGE (JBEG THRU JEND)
C
1500 DO 2000 I = JBEG,JEND
IF (LUPLO.EQ.1) GO TO 1500
IBEG = 1
IEND = JEND
IF (LUPLO.EQ.2) GO TO 2000
IBEG = JBEG
IEND = N
1500 DO 2000 I = IBEG, IEND
IF (LUPLO.EQ.1) GO TO 1500
IBEG = JBEG
IEND = N
1500 Do 2000 I = IBEG, IEND
IF (LUPLO.EQ.2) GO TO 2000
IBEG = JBEG
IEND = N
2000 IF (IUPLOW.EQ.1) GO TO 2100
IBEG = JBEG
IEND = N
2000 IF (IUPLOW.EQ.2) GO TO 2100
IBEG = JBEG
IEND = N
2000 IJ = I - JBEG
M = IJ - 1
IF (M.LT.0) GO TO 2100
2100 IF (IJ+M.GT.N) GO TO 2200
J = IJ + M
DO 2200 K = 1,L.R
2200 PWH<K> = PWH<K>
LIST I-TH ROW
C
IF (IUPLOW.EQ.1) GO TO 2100
IF (IUPLOW.EQ.2) GO TO 2100
IF (IUPLOW.EQ.3) GO TO 2100
C
PREPARE I-TH ROW OF UPPER TRIANGLE
KDQFM<X+1> = DQFM<X+1>
J = JREF + K
M = J - I
IF (MLT.J) GO TO 2200
I = IJ + M
M = IJ - M
IF (IUPLOW.EQ.1) GO TO 2200
IF (IUPLOW.EQ.2) GO TO 2200
DQFM<X+2> = FMT(1)
KR = KR + 1
KDQFM<KR> = KDQFM<KR>
2200 IJ = IJ + 1
C
PREPARE I-TH ROW OF LOWER TRIANGLE
2000 IF (IUPLOW.EQ.1) GO TO 2100
IBEG = IJ
IEND = JEND
M = IJ - I - 1
IF (M.LT.0) GO TO 2200
I = IJ + M
J = IJ + M
DO 2200 K = 1,L.R
2200 PWH<K> = PWH<K>
LIST I-TH ROW
C
DO 3000 PWH<1> = VIPS
VIPS = 0.0
IF (IUPLOW.EQ.3) GO TO 3000
3000 CONTINUE
4000 JREF = JEND
IF (JREF=1) GO TO 4000
5000 RETURN
C
REAL FUNCTION VIPSD (A, B, N)
C
SAMPLE INNERPRODUCT FUNCTION PROCEDURE SERVING SKYFAC AND SKYSOL,
SINGLE PRECISION ACCUMULATION VERSION (ASSEMBLY LANGUAGE
SHOULD BE USED IN ACTUAL IMPLEMENTATION ON A GIVEN MACHINE)
REAL A(I), B(I)
VIPS = 0.0
IF (IUPLOW.EQ.1) GO TO 3000
VIPS = VIPS + A(I)*B(I)
3000 CONTINUE
5000 RETURN
END
C
REAL FUNCTION VIPSD (A, B, N)
C
SAMPLE INNERPRODUCT FUNCTION PROCEDURE SERVING SKYFAC AND SKYSOL,
DOUBL E PRECISION ACCUMULATION VERSION (ASSEMBLY LANGUAGE
SHOULD BE USED IN ACTUAL IMPLEMENTATION ON A GIVEN MACHINE)
REAL A(I), B(I)
DOUBLE PRECISION SUM
SUM = 0.000
IF (IUPLOW.EQ.1) GO TO 3000
SUM = SUM + DBLE(A(I))*DBLE(B(I))
3000 CONTINUE
5000 RETURN
END
**SYNOPSIS TEST RESULTS**

- **OPERATION UNITS**: 226
- **NEGATIVE EIGENVALUES**: 5
- **AVG TIME/UNIT**: 34.50 MICROSEC

**TEST MATRIX**

<table>
<thead>
<tr>
<th>COL 1</th>
<th>COL 2</th>
<th>COL 3</th>
<th>COL 4</th>
<th>COL 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125976</td>
<td>-0.772934</td>
<td>0.611516</td>
<td>-0.326797</td>
<td>-0.777990</td>
</tr>
<tr>
<td>0.018694</td>
<td>-0.772934</td>
<td>0.611516</td>
<td>-0.326797</td>
<td>-0.777990</td>
</tr>
<tr>
<td>-0.013856</td>
<td>-0.772934</td>
<td>0.611516</td>
<td>-0.326797</td>
<td>-0.777990</td>
</tr>
<tr>
<td>-0.44013</td>
<td>-0.326797</td>
<td>-0.156199</td>
<td>0.0</td>
<td>-0.483778</td>
</tr>
<tr>
<td>-0.44013</td>
<td>-0.326797</td>
<td>-0.156199</td>
<td>0.0</td>
<td>-0.483778</td>
</tr>
<tr>
<td>-0.44013</td>
<td>-0.326797</td>
<td>-0.156199</td>
<td>0.0</td>
<td>-0.483778</td>
</tr>
<tr>
<td>-0.236190</td>
<td>-0.156199</td>
<td>0.153916</td>
<td>0.0</td>
<td>-0.083927</td>
</tr>
<tr>
<td>-0.236190</td>
<td>-0.156199</td>
<td>0.153916</td>
<td>0.0</td>
<td>-0.083927</td>
</tr>
<tr>
<td>-0.236190</td>
<td>-0.156199</td>
<td>0.153916</td>
<td>0.0</td>
<td>-0.083927</td>
</tr>
<tr>
<td>-0.236190</td>
<td>-0.156199</td>
<td>0.153916</td>
<td>0.0</td>
<td>-0.083927</td>
</tr>
<tr>
<td>-0.236190</td>
<td>-0.156199</td>
<td>0.153916</td>
<td>0.0</td>
<td>-0.083927</td>
</tr>
</tbody>
</table>

**HCTUR! RELTIVE ERROR OF COMPUTED *Xi**: 1.24-03

**ESTIMATED REL. ERROR OF UNFINISHED *Xi**: 1.97-07
and should be removed, unless extended double precision arithmetic (e.g. REAL*16 on IBM 370 computers) is available for the residual calculation in subroutine SKYMUL.

A slightly more efficient implementation of these procedures is possible if array LD is used to pack the following information:

(a) diagonal location pointer
(b) local skyline width excluding diagonal
(c) constraint flag

into 3 bit fields. Such a version is not presented here since partial word manipulation with Fortran statements is highly machine dependent.

APPLICATIONS

General

The equation solving procedures presented here may be useful in the following application areas:

1. As fast linear equation solvers for small-scale systems (normally not exceeding 1000 equations). This is typical of most finite element codes written in universities and research institutions, as well as pilot programs developed in industry for special applications or feasibility studies of nonlinear or time-dependent problems.

2. As freedom “condensation” processors for first-level superelement analysis. This application is elaborated in the following subsection.

3. As a conceptual aid in the design and coding stages of the development of large-scale equation solvers based on the skyline storage approach.

Superelement condensation

Superelement techniques[14] (also called ‘substructuring’ or ‘tearing’ by structural engineers and ‘dissection’ or ‘blocking’ by numerical analysts[9]) are model-partitioning procedures intended for a multistage analysis of complex finite element idealizations. A block of finite elements modeling a simply connected portion of a complex system is called a level one superelement. A connected set of level one superelements makes a level two superelement and so forth until the complete model is realized. The fundamental characteristic of the associated matrix assembly process is that ‘internal’ degrees of freedom of all superelements of given level (those freedoms not shared by two or more superelements) are reduced out or ‘condensed’ before proceeding to the next level.

The governing linear† equations for a particular superelement may be represented by equation (4), in which subvector \( x^e \) collects all internal freedoms whereas \( x_e \) includes external or connected freedoms. Elimination of \( x_e \) yields:

\[
A^e x^e = b^e
\]

where the (symmetric) condensed superelement matrix \( A^e \) and right hand side vector \( b^e \) are given by

\[
A^e = A_e - A_e^T A_e^{-1} A_e^T
\]

\[
b^e = b_e - A_e^T A_e^{-1} b_e
\]

provided \( A_e \) is nonsingular. The matrix condensation process (18) can usually be carried out in core for level one (and sometimes level two) superelements. If such is the case, the routines presented here can be made use of as follows:

1. Use SKYFAC to factor \( A_e \) upon marking the \( n_e \) equations associated with the external freedoms \( x_e \) as constrained.

2. Solve the homogeneously constrained block system

\[
\begin{bmatrix}
A^e & A_e & A_e
\end{bmatrix}
\begin{bmatrix}
X
\end{bmatrix}
= \begin{bmatrix}
A_e & 0
\end{bmatrix}
\begin{bmatrix}
b_e
\end{bmatrix}
\]

by submitting to SKYSOL (with argument IBX = 1) \( n_e \) right hand sides consisting of the \( n_e \) columns of \( A_e \) completed with zeros on the constraint equation positions.

3. The condensed matrix \( A^* \) is

\[
A^* = A_e - A^e A^e\inv A_e - b_e
\]

so that one should simply subtract each column of \( B \) (as they are being put out by SKYSOL) from the appropriate column of \( A_e \) to produce \( A^* \) in situ (overwriting \( A_e \)). Note that \( X \) is not used.

The reader may verify that the entire process can be carried out without prearranging \( A \) and that only two auxiliary floating-point vectors of length \( n \) are required in addition to the \( A \)-definition arrays \( A \) and \( LD \). The reduction (19) of the load vector can be executed in a similar manner.

REFERENCES


†Superelement techniques are most effective in linear problems.