

JPL PUBLICATION 77-26. REVISION 1

(NASA-CR-157165) A PARAMETER ESTIMATION  
SUBROUTINE PACKAGE (Jet Propulsion Lab.)  
144 p HC A07/MF A01 CSCI 09E

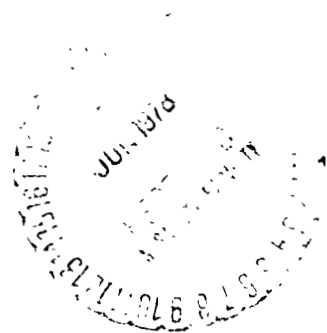
N78-25790

Unclas  
G3/61 21309

# A Parameter Estimation Subroutine Package

National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California



JPL PUBLICATION 77-26, REVISION 1

# A Parameter Estimation Subroutine Package

G. J. Bierman  
M. W. Nead

April 15, 1978

National Aeronautics and  
Space Administration

Jet Propulsion Laboratory  
California Institute of Technology  
Pasadena, California

Prepared Under Contract No NAS 7-100  
National Aeronautics and Space Administration

PREFACE

The work described in this report was performed by the Systems Division of the Jet Propulsion Laboratory.

#### ACKNOWLEDGEMENT

The construction of this Estimation Subroutine Package (ESP) was motivated by an involvement with a particular problem; construction of fast, efficient and simple least squares data processing algorithms to be used for determining ephemeris corrections. Discussion with T. C. Duxbury led to the proposal of a subroutine strategy which would have great flexibility. The general utility of such a subroutine package was made evident by H. M. Koble and N. A. Mottinger who had a different but related problem that involved combining estimates from different missions. Thanks and credit are also due to our colleagues for experimenting with this package of subroutines and letting us benefit from their experience.

## ABSTRACT

Linear least squares estimation and regression analyses continue to play a major role in orbit determination and related areas. In this report we document a library of FORTRAN subroutines that have been developed to facilitate analyses of a variety of estimation problems. Our purpose is to present an easy to use, multi-purpose set of algorithms that are reasonably efficient and which use a minimal amount of computer storage. Subroutine inputs, outputs, usage and listings are given, along with examples of how these routines can be used. The following outline indicates the scope of this report: Section I, introduction with reference to background material; Section II, examples and applications; Section III, a subroutine directory summary; Section IV, the subroutine directory user description with input, output and usage explained; and Section V, subroutine FORTRAN listings. The routines are compact and efficient and are far superior to the normal equation and Kalman filter data processing algorithms that are often used for least squares analyses.

## CONTENTS

I. Introduction . . . . .	1
II. Applications and Examples . . . . .	4
III. Subroutine Directory Summary . . . . .	23
IV. Subroutine Directory User Description . . . . .	38
V. References . . . . .	84
VI. FORTRAN Subroutine Listings. . . . .	85

PRECEDING PAGE BLANK NOT FILMED

## I. Introduction

Techniques related to least squares parameter estimation play a prominent role in orbit determination and related analyses. Numerical and algorithmic aspects of least squares computation are documented in the excellent reference work by Lawson and Hanson, Ref. [1]. Their algorithms, available from the JPL subroutine library, Ref. [2], are very reliable and general. Experience has, however, shown that in reasonably well posed problems one can streamline the least squares algorithm codes and reduce both storage and computer times. In this report, we document a collection of subroutines most of which we have written that can be used to solve a variety of parameter estimation problems.

The algorithms for the most part involve triangular and/or symmetric matrices and to reduce storage requirements these are stored in vector form, e.g., an upper triangular matrix  $U$  is written as

$$\begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ & U_{22} & U_{23} & U_{24} \text{ etc.} \\ \bigcirc & & U_{33} & U_{34} \\ & & & U_{44} \end{bmatrix} = \begin{bmatrix} U(1) & U(2) & U(4) & U(7) \\ & U(3) & U(5) & U(8) \text{ etc.} \\ & & U(6) & U(9) \\ & & & U(10) \end{bmatrix}$$

Thus, the element from row  $i$  and column  $j$  of  $U$ ,  $i \leq j$ , is stored in vector component  $j(j-1)/2 + i$ . We hasten to point out that the engineer, with few exceptions, need have no direct contact with the vector subscripting. By this we mean that the vector subscript related operations are internal to the subroutines, vector arrays transmitted from one



subroutine to another are compatible, and vector arrays displayed using the print subroutine TRIMAT appear in a triangular matrix format.

Aside: The most notable exception is that matrix problems are generally formulated using doubly subscripted arrays. Transforming a double subscripted symmetric or upper triangular matrix  $A(\cdot,\cdot)$  to a vector stored form,  $U(\cdot)$  is quite simply accomplished in FORTRAN via

```
IJ = 0
DO 1 J = 1,N
DO 1 I = 1,J
IJ = IJ+1
1 U(IJ) = A(I,J)
```

Similarly, transforming an initial vector  $D(\cdot)$  of diagonal positions of a vector stored form,  $U(\cdot)$ , is accomplished using

```
JJ = 0
DO 1 J = 1,N
JJ = JJ+J
1 U(JJ) = D(J)
```

or

```
JJ = N*(N+1)/2
DO 1 J = N,1,-1
U(JJ) = D(J)
1 JJ = JJ-J
```

The conversion on the right has the modest advantage that  $D$  and  $U$  can share common storage (i.e.,  $U$  can overwrite  $D$ ). These conversions are too brief to be efficiently used as subroutines. It seems that when such conversions are needed one can readily include them as in-line code.

#### End of Aside

This package of subroutines is designed, in the main, for the analysis of parameter estimation problems. One can, however, use it to solve problems that involve process noise and to map (time propagate) covariance or information matrix factors. In the case of mapping the storage savings associated with the use of vector stored triangular matrices is, to some extent, lost.

Mathematical background regarding Householder orthogonal transformations for least squares analyses and U-D matrix factorization for covariance matrix analyses are discussed in references [1] and [3]. Our plan is to illustrate, in Section II, with examples, how one can use the basic algorithms and matrix manipulation to solve a variety of important problems. The subroutines which comprise our estimation subroutine package are described in Section III, and detailed input/output descriptions are presented in Section IV.

Section V contains FORTRAN listings of the subroutines. There are several reasons for including such listings. Making these listings available to the engineer analyst allows him to assess algorithm complexity for himself; and to appreciate the simplicity of the routines he tends otherwise to use as a black box. The routines use only FORTRAN IV and are therefore reasonably portable (except possibly for routines which involve alphanumeric inputs). When estimation problems arise to which our package does not directly apply (or which can be made to apply by an awkward concatenation of the routines) one may be able to modify the codes and widen still further the class of problems that can be efficiently solved.

## II. APPLICATIONS AND EXAMPLES

Our purpose in this section is to illustrate, with a number of examples, some of the problems that can be solved using this ESP. The examples, in addition, serve to catalogue certain estimation techniques that are quite useful.

To begin, let us catalogue the subroutines that comprise the ESP:

- 1) A2A1 (A to A one) Matrix A to matrix A1
- 2) COMBO (combo) Combine R and A namelists
- 3) COVRHO (cov rho) Covariance to correlation matrix, RHO
- 4) COV2RI (cov to RI) Covariance to R inverse
- 5) COV2UD (cov to U-D) Covariance to U-D covariance factors
- 6) C2C (C to C) Permute the rows and columns of matrix C
- 7) INF2R (inf to R) Information matrix to (triangular) R factor
- 8) HHPOST (HH POST) Householder triangularization by post multiplication
- 9) PERMUT (permut) Permute the columns of matrix A
- 10) PHIU (PHI\*U) Multiplies a rectangular PHI matrix by the vector stored U matrix that has implicitly defined unit diagonal entries.
- 11) RA (R\*A) R(upper triangular, vector stored)\*A (rectangular)
- 12) RANK1 (rank 1) Updated U-D factors of a rank-1 modified matrix
- 13) RCOLRD (R colored) (SRIF)R colored noise time-update
- 14) RINCON (rin-con) R inverse along with a condition number bounding estimate
- 15) RI2COV (RI to cov) R inverse to covariance
- 16) R2A (R to A) Triangular R to (rectangular stored) matrix A
- 17) R2RA (R to RA) Transfer to triangular block of (vector stored) R to a triangular (vector stored) RA
- 18) RUDR (rudder) (SRIF)R to U-D covariance factors, or vice-versa
- 19) SFU (S F U) Sparse F matrix \* vector stored U matrix with implicitly defined unit diagonal entries
- 20) TDHHT (T D H H T) Two dimensional Householder matrix triangularization
- 21) THH (T H H) Triangular vector stored Householder data processing algorithm
- 22) TTHH (T T H H) Orthogonal triangularization of two triangular matrices
- 23) TWOMAT (two mat) Two dimensional labeled display of a vector stored triangular matrix

24) TZERO	(T zero)	Zero a horizontal segment of a vector stored triangular matrix
25) UDCOL	(U-D colored)	U-D covariance factor colored noise update
26) UDMEAS	(U-D measurement)	U-D covariance factor measurement update
27) UD2COV	(U-D to cov)	U-D factors to covariance
28) UD2SIG	(U-D to sig)	U-D factors to sigmas
29) UTINV	(U D inverse)	Upper triangular matrix inverse
30) UTIROW		Upper triangular inverse, inverting only the upper rows
31) WGS	(W G-S)	U-D covariance factorization using a weighted Gram-Schmidt reduction

These routines are described in succeedingly more detail in sections III, IV, and V. The examples to follow are chosen to demonstrate how these various subroutines can be used to solve orbit determination and other parameter estimation problems. It is important to keep in mind that these examples are not by any means all inclusive, and that the package of subroutines has a wide scope of applicability.

### II.1 Simple Least Squares

Given data in the form of an overdetermined system of linear equations one may want a) the least squares solution; b) the estimate error covariance, assuming that the data has normalized errors; and c) the sum of squares of the residuals. The solution to this problem, using the ESP can be symbolically depicted as

$$\bullet [A:z] \xrightarrow{\text{THH}} [\hat{R}:\hat{z}], e$$

Remarks: The array  $[A:z]$  corresponds to the equations  $Ax = z - v$ ,  $v \in N(0, I)$ ; the array  $[\hat{R}:\hat{z}]$  corresponds to the triangular data equation  $\hat{R}x = \hat{z} - \hat{v}$ ,  $v \in N(0, I)$  and  $e = ||z - Ax||$

$$\bullet [\hat{R}:\hat{z}] \xrightarrow{\text{UTINV}} [\hat{R}^{-1}:\hat{x}]$$

Remark:  $\hat{x} = \hat{R}^{-1} \hat{z}$

One may be concerned with the integrity of the computed inverse and the estimate. If one uses subroutine RINCON instead of UTINV then in addition one obtains an estimate (lower and upper bounds) for the condition number R. If this condition number estimate is large the computed inverse and estimate are to be regarded with suspicion. By large, we mean considerable with respect to the machine accuracy (viz. on an 18 decimal digit machine numbers larger than  $10^{15}$ ). Note that the condition number estimate is obtained with negligible additional computation and storage.

$$[R^{-1}] \xrightarrow{\text{RI2COV}} [C]$$

Remarks:  $C = R^{-1} R^{-T}$  = estimate error covariance. Some computation can be avoided in RI2COV if only some (or all) of the standard deviations are wanted.

## II.2 Least Squares With A Priori

If a priori information is given, it can be included as additional equations (in the A array) or used to initialize the R array in subroutine THH (see the subroutine argument description given in section IV). One is sometimes interested in seeing how the estimate and/or the formal statistics change corresponding to the use of different a priori conditions. In this case one should compute  $[\hat{R}; \hat{z}]$  as in case II.1, and then include the a priori  $[R_0; z_0]$  using either subroutine THH, or subroutine TTHH when the a priori is diagonal or triangular, e.g.,

$$\left. \begin{array}{l} [\hat{R}; \hat{z}] \\ [R_0; z_0] \end{array} \right\} \xrightarrow{\text{TTHH}} [\hat{R}; \hat{z}]^*$$

\* The new result overwrites the old.

It is often good practice to process the data and form  $[\hat{R}:z]$  before including the a priori effects. When this is done one can analyze the effect of different a priori,  $[R_0:z_0]$  without reprocessing the data.

If a priori is given in the form of an information matrix,  $\Lambda$ , (as for example would be the case if the problem is being initialized with data processed using normal equation data accumulation\*) then one can obtain  $R_0$  from  $\Lambda$  using INF2R;

$$\Lambda \xrightarrow{\text{INF2R}} R_0$$

If there were a normal equation estimate term,  $z = A^T b$ , then  $z_0 = R_0^{-T} z$ .

### II.3 Batch Sequential Data Processing

Prime reasons for batch sequential data processing are that many problems are too large to fit in core, are too expensive in terms of core cost, and for certain problems it is desirable to be able to incorporate new data as it becomes available. Subroutines THH and UDMEAS are specially designed for this kind of problem. Both of these subroutines overwrite the a priori with the result which then acts as a priori for the next batch of data. If the data is stored on a file or tape as  $A_1, z_1, A_2, z_2, \dots$  then the sequential process can be represented as follows:

#### SRIF Processing\*\*

- a) Initialize  $[R:z]$  with a priori values or zero
- b) Read the next  $[A:z]$  from the file

\* i.e., solving  $Ax = b-v$  with normal equations,  $A^T \hat{A} x_0 = A^T b$ ;  $\Lambda = A^T A$  is the information matrix.

\*\* The acronym SRIF represents Square Root Information Filter. The SRIF is discussed at length in the book by Bierman, ref. [3].

$$c) \begin{Bmatrix} \hat{R}:z \\ \hat{A}:z \end{Bmatrix} \xrightarrow{\text{THH}} \hat{R}:z^*$$

- d) If there is more data go back to b)
- e) Compute estimates and/or covariances using UTINV and RI2COV  
(as in example II.1)

#### U-D\*\* Processing

- a) Initialize  $[\hat{U}-\hat{D}:\hat{x}]$  with a priori U-D covariance factors and the initial estimate
- b) Read the next  $[\hat{A}:z]$  scalar measurement from the file
- c)  $\begin{Bmatrix} \hat{U}-\hat{D}:\hat{x} \\ \hat{A}:z \end{Bmatrix} \xrightarrow{\text{UDMEAS}} \hat{U}-\hat{D}:\hat{x}^*$
- d) If there is more data go back to b)
- e) Compute standard deviations or covariances using UD2SIG or UD2COV.

Note that subroutine THH is best (most efficiently) used with data batches of substantial size (say 5 or more) and that UDMEAS processes measurement vectors one component at a time. If the dimension of the state is small the cost of using either method is generally negligible. The UDMEAS subroutine is best used in problems where estimates are wanted with great frequency or where one wishes to monitor the effects of each update. In a given application one might choose to process data in batches for a while and during critical periods it may be

---

\* The new result overwrites the old.

\*\* U-D processing is a numerically stable algorithmic formulation of the Kalman filter measurement update algorithm, cf reference [3]. The estimate error covariance is used in its  $UDU^T$  factored form, where U is unit upper triangular and D is diagonal.

desirable to monitor the updating process on a point by point basis.

In cases such as this, one may use RUDR to convert a SRIF array to U-D form or vice-versa.

Remarks: Another case where an R to U-D conversion can be useful occurs in large order problems (with say 100 or more parameters) where after data has been SRIF processed one wants to examine estimate and/or covariance sensitivity to the a priori variances of only a few of the variables. Here it may be more convenient to update using the UDMEAS subroutine.

#### II.4 Reduced State Estimates and/or Covariances From a SRIF Array

Suppose, for example, that data has been processed and that we have a triangular SRIF array  $[\hat{R}:\hat{z}]$  corresponding to the 14 parameter names,  $a_r$ ,  $a_x$ ,  $a_y$ ,  $x$ ,  $y$ ,  $z$ ,  $v_x$ ,  $v_y$ ,  $v_z$ , GM, CU41, L041, CU43, L043 (constant spacecraft accelerations, position and velocity, target body gravitational constant, and spin axis and longitude station location errors).

Let us ask first what would the computed error covariance be of a model containing only the first 10 variables, i.e., by ignoring the effect of the station location errors. One would apply UTINV and RI2COV just as in example II.1, except here we would use N (the dimension of the filter) = 10, instead of N=14.

Next, suppose that we want the solution and associated covariance of the model without the 3 acceleration errors. One ESP solution is to use



$$\bullet [\hat{R}:\hat{z}] \xrightarrow{R2A} [A]$$

NAME ORDER OF A

x, y, z, v<sub>x</sub>, v<sub>y</sub>, v<sub>z</sub>,

GM, U41, L041, CU43, L043,

RHS<sup>\*</sup>, a<sub>r</sub>, a<sub>x</sub>, a<sub>y</sub>,

Remark: One could also have used subroutine COMBO, with the desired namelist as simply a<sub>r</sub>, a<sub>x</sub>, a<sub>y</sub>. This would achieve the same A matrix form.

$$\bullet [A] \xrightarrow{THH} [R]$$

Remark: R here can replace the original  $\hat{R}$  and  $\hat{z}$ .

$$\bullet [R] \xrightarrow{UTINV} [R^{-1}:x_{est}] \xrightarrow{RI2COV} [COV:x_{est}]$$

Remarks: Here, use only N=11, i.e., 11 variables and the RHS.  $x_{est}$  is the 11 state estimate based on a model that does not contain acceleration errors a<sub>r</sub>, a<sub>x</sub>, or a<sub>y</sub>.

Note how triangularizing the rearranged R matrix produces the desired lower dimensional SRIF array; and this is the same result one would obtain if the original data had been fit using the 11 state model.

As the last subcase of this example suppose that one is only interested in the SRIF array corresponding to the position and velocity variables. The difference between this example and the one above is that here we want to include the effects due to the other variables.

---

\* z is often given the label RHS (right hand side)

One might want this sub-array to combine with a position-velocity SRIF array obtained from, say, optical data. One method to use would be,

$$\bullet \quad [\hat{R}:\hat{z}] \xrightarrow{R2RA} [R_A:z_A]$$

INPUT NAMES:

$a_r, a_x, a_y, x, y, z, v_x, v_y, v_z, GM$   
 CU41, LO41, CU43, LO43, RHS

OUTPUT NAMES:

$x, y, z, v_x, v_y, v_z, GM$   
 CU41, LO41, CU43, LO43, RHS

Remark: The lower triangle starting with x is copied into  $R_A$ .

$$\bullet \quad [R_A:z_A] \xrightarrow{R2A} [A:z_A] \text{ (Reordering)}$$

NAMES: GM, CU41, LO41, CU43, LO43,

$x, y, z, v_x, v_y, v_z, RHS$

$$\bullet \quad [A:z_A] \xrightarrow{THH} [\hat{R}_A:\hat{z}_A] \text{ (Triangularizing)}$$

$$\bullet \quad [\hat{R}_A:\hat{z}_A] \xrightarrow{R2RA} [R_x:z_x] \text{ (Shifting array)}$$

NAMES:  $x, y, z, v_x, v_y, v_z, RHS$

Remark: The lower right triangle starting with x is copied into  $R_x$ .

We note that one could have elected to use COMBO in place of the first R2RA usage and R2A; this would have involved slightly more storage, but a lesser number of inputs. The sequence of operations is in this case,

$$\bullet \quad [\hat{R}:\hat{z}] \xrightarrow{COMBO} [A:z]$$

ORIGINAL NAMES      DESIRED NAMES:  $x, y, z, v_x, v_y, v_z, RHS$

Remark: By using COMBO the columns of  $[\hat{R}:\hat{z}]$  are ordered corresponding to the names  $a_r, a_x, a_y, GM, CU41, LO41, CU43,$  and LO43, followed by the desired names list.

$$\bullet [A:z] \xrightarrow{\text{THH}} [\hat{R}:\hat{z}]$$

Remark: The  $[\hat{R}:\hat{z}]$  array that is output from this procedure is equivalent but different from the  $[\hat{R}:\hat{z}]$  array that we began with.

$$\bullet [\hat{R}:\hat{z}] \xrightarrow{\text{R2RA}} \begin{bmatrix} \hat{R} & : & \hat{z} \\ & & \hat{z}_x \end{bmatrix}$$

Remark: As before, the lower right triangle starting with x is copied into  $R_x$ .

To delete the last k parameters from a SRIF array, it is not necessary to use subroutines R2A and THH. The first  $N - k = \bar{N}$  columns of the array already correspond to a square root information matrix of the reduced system. If estimates are involved one can simply move the z column left using:

$$R(\bar{N}*(\bar{N} + 1)/2 + i) = R(N*(N + 1)/2 + i), i = 1, \dots, k.$$

Remark: We mention in passing that if one is only interested in estimates and/or covariances corresponding to the last k parameters then one can use R2RA to transform the lower right triangle of the SRIF array to an upper left triangle after which UTINV and RI2COV can be applied.

#### II.5 Sensitivity, Perturbation, Computed Covariance and Consider Covariance Matrix Computation

Suppose that one is given a SRIF array

$$\begin{array}{ccc} \underbrace{N_x} & \underbrace{N_y} & \underbrace{1} \\ \left[ \begin{array}{ccc} R_x & R_{xy} & z_x \\ 0 & R_y & z_y \end{array} \right] & \left. \begin{array}{l} \{ N_x \\ \{ N_y \end{array} \right\} & \end{array} \quad (\text{II.5a})$$

in which the  $N_y$  variables are to be considered. (One can, of course, using subroutines R2A and THH reorder and retriangularize an arbitrarily arranged SRIF array so that a given set of variables fall at the end.) For various reasons one may choose to ignore the y variables in the equation

$$R_x x + R_{xy} y = z_x - v_x, \quad v_x \in N(0, I) \quad (\text{II.5b})$$

and take as the estimate  $x_c = R_x^{-1} z_x$ . It then follows that

$$x - x_c = -R_x^{-1} R_{xy} y - R_x^{-1} v_x, \quad (\text{II.5c})$$

and from this one obtains

$$\text{Sen} \equiv \frac{\partial(x-x_c)}{\partial y} = -R_x^{-1} R_{xy} \quad (\text{II.5d})$$

(sensitivity of the estimate error to the unmodeled y parameters)

$$\text{Pert} = \text{Sen} * \text{Diag}(\sigma_y(1), \dots, \sigma_y(N_y)) \quad (\text{II.5e})$$

where  $\sigma_y(1), \dots, \sigma_y(N_y)$  are a priori y parameter uncertainties.

(The perturbations are a measure of how much the estimate error could be expected to change due to the unmodeled y parameters.)

$$P_{\text{con}} = R_x^{-1} R_x^{-T} + \text{Sen} P_y \text{Sen}^T \quad (\text{II.5f})$$

$$= P_c + (\text{Pert})(\text{Pert})^T \text{ if } P_y \text{ is diagonal}^\dagger$$

where  $P_c$  is the estimate error covariance of the reduced model.

An easy way to compute  $P_c$ ,  $\text{Pert}$  and  $P_{\text{con}}$  is as follows: Use subroutine R2RA to place the y variable a priori  $[P_y^{1/2}(0) : \hat{y}_0]^\dagger$  into the lower right

---

<sup>†</sup>  $\text{Pert} = \text{Sen} P_y^{1/2}$

<sup>††</sup> The a priori estimate  $y_0$  of consider parameters is generally zero.

corner of (II.5a), replacing  $R_y$  and  $z_y$ , i.e.,

$$\left. \begin{array}{l} [R : z] \\ [P_y^{1/2}(0) : \hat{y}_0] \end{array} \right\} \xrightarrow{R2RA} \begin{bmatrix} R_x & R_{xy} & z_x \\ 0 & P_y^{1/2}(0) & \hat{y}_0 \end{bmatrix}$$

Now apply subroutine UTIROW to this system (with a -1 set in the lower right corner\*)

$$\left[ \begin{array}{ccc} R_x & R_{xy} & z_x \\ \hline 0 & P_y^{1/2}(0) & \hat{y}_0 \\ 0 & 0 & -1 \end{array} \right] \xrightarrow{UTIROW} \left[ \begin{array}{ccc} R_x^{-1} & \text{Pert}^{**} & x_c \\ \hline 0 & P_y^{1/2}(0) & \hat{y}_0 \\ 0 & 0 & -1 \end{array} \right]$$

Note that the lower portion of the matrix is left unaltered, i.e., the purpose of UTIROW is to invert a triangular matrix, given that the lower rows have already been inverted. From this array one can, using subroutine RI2COV, get both  $P_c$  and  $P_{con}$

$$\begin{aligned} [R_x^{-1}] &\xrightarrow{RI2COV} [P_c] \quad \text{computed covariance} \\ [R_x^{-1} : \text{Pert}] &\xrightarrow{RI2COV} [P_{con}] \quad \text{consider covariance} \end{aligned}$$

Suppose now that one is dealing with a U-D factored Kalman filter formulation. In this case estimate error sensitivities can be sequentially

\*

To have estimates from the triangular inversion routines one sets a -1 in the last column (below the right hand side).

\*\*

Strictly speaking this is not what we call the perturbation unless  $P_y(0)$  is diagonal.

calculated as each scalar measurement ( $z = a_x^T x + a_y^T y + v$ ) is processed.

$$\text{Sen}_j = \text{Sen}_{j-1} - K_j (a_x^T \text{cor}_{j-1} + a_y^T)$$

where  $\text{Sen}_{j-1}$  is the sensitivity prior to processing this (j-th) measurement, and  $K_j$  is the Kalman gain vector.<sup>†</sup>

In this formulation one computes  $P_{\text{con}}$  in a manner analogous to that described in section II.7;

$$\text{Let } \bar{U}_1 = U_j, \bar{D}_1 = D_j \quad (\text{filter U-D factors})$$

$$[s_1, \dots, s_{n_y}] = \text{Sen}_j \quad (\text{estimate error sensitivities})$$

then recursively compute

$$\bar{U}_k - \bar{D}_k, \sigma_k^2, s_k \xrightarrow{\text{RANK1}} \bar{U}_{k+1} - \bar{D}_{k+1} \quad k = 1, \dots, n_y$$

For the final  $\bar{U} - \bar{D}$  we have

$$U_{j+1}^{\text{con}} = \bar{U}_{n_y+1}, D_{j+1}^{\text{con}} = D_{n_y+1}$$

If  $P_y(0) = U_y D_y U_y^T$ , instead of  $P_y(0) = \text{Diag}(\sigma_1^2, \dots, \sigma_{n_y}^2)$ , then in the

U-D recursion one should replace the  $\text{Sen}_j$  columns by those of  $\text{Sen}_j * U_y$  and  $\sigma_j^2$  should be replaced by the corresponding diagonal elements of  $D_y$ .

## II.6 Combining Various Data Sets

In this example we collect several related problems involving data sets with different parameter lists.

Suppose that the parameter namelist of the current data does not correspond to that of the a priori SRIF array. If the new data involves a permutation or a subset of the SRIF namelist, then an application of

<sup>†</sup> $K = g/\alpha$  where  $g$  and  $\alpha$  are quantities computed in subroutine UDMEAS.

subroutine PERMUT will create the desired data rearrangement. If the data involves parameters not present in the SRIF namelist then one could use subroutine R2A to modify the SRIF array to include the new names and then if necessary use PERMUT on the data, to rearrange it compatibly.

Suppose now that two data sets are to be combined and that each contains parameters peculiar to it (and of course there are common parameters). For example let data set 1 contain names ABC and data set 2 contain names DEB. One could handle such a problem by noting that the list ABCDE contains both name lists. Thus one could use subroutine PERMUT on each data set comparing it to the master list, ABCDE, and then the results could be combined using subroutine THH. An alternative automated method for handling this problem is to use subroutine COMBO with data set 1 (assuming it is in triangular form) and namelist 2. The result would be data set 1 in double subscripted form and arranged to the namelist ACDEB (names A and C are peculiar to data set 1 and are put first). Having determined the namelist one could apply subroutine PERMUT to data set 2 and give it a compatible namelist ordering.

The process of increasing the namelist size to accommodate new variables can lead to problems with excessively long namelists, i.e., with high dimension. If it is known that a certain set of variables will not occur in future data sets then these variables can be eliminated and the problem dimension reduced. To eliminate a vector y from a SRIF array, first use subroutine R2A to put the y names first in the namelist; then use subroutine THH to retriangularize and finally use subroutine R2RA to put the y independent subarray in position for further use; viz.

$$[R] \xrightarrow{R2A} [A] \xrightarrow{TTH} \begin{bmatrix} R_y & R_{yx} & z_y \\ 0 & R_x & z_x \end{bmatrix} \xrightarrow{R2RA} [R_x : z_x]$$

The rows  $[R_y : R_{yx} : z_y]$  can be used to recover a  $y$  estimate (and its covariance) when an estimate for  $x$  (and its covariance) are determined. (See example II.4).

Still another application related to the combining of data sets involves the combining of SRIF triangular data arrays. One might encounter such problems when combining data from different space missions (that involve common parameters) or one might choose to process data of each type\* or tracking station separately and then combine the resulting SRIF arrays. Triangular arrays can be combined using subroutine TTH, assuming that subroutines R2A, TTH and R2RA have been used previously to formulate a common parameter set for each of the sub problems.

#### II.7 Batch Sequential White Noise

It is not uncommon to have a problem where each data set contains a set of parameters that apply only to that set and not to any other, viz. the data is of the form

$$A_j x + B_j y_j = z_j - v_j \quad j = 1, \dots, N$$

where there is generally a priori information on the vector  $y_j$  variables. Rather than form a concatenated state vector composed of  $x, y_1, \dots, y_N$  which might create a problem involving exorbitant amounts of storage and computation we solve the problem as follows. Apply subroutine TTH to  $[B_1 : A_1 : z_1]$ , with the corresponding  $R$  initialized with the  $y_1$  a priori. The resulting SRIF array is of the form

\* viz. range, doppler, optical, etc.



$$N_{y_1} \left\{ \begin{bmatrix} R_{y_1} & R_{y_1 x} & z_{y_1} \\ 0 & R_{x_1} & z_{x_1} \end{bmatrix} \right.$$

Copy the top  $N_{y_1}$  rows if one will later want an estimate or covariance of the  $y_1$  parameters. Apply subroutine TZERO to zero the top  $N_{y_1}$  rows and using subroutine R2RA set in the  $y_2$  a priori\*. This SRIF array is now ready to be combined with the second set of data  $[B_2; A_2; z_2]$  and the procedure repeated.

A somewhat analogous situation is represented by the class of problems that involve noisy model variations, i.e., the state at step  $j+1$  satisfies

$$x_{j+1} = x_j + G_j w_j$$

where matrix  $G_j$  is defined so that  $w_j$  is independent of  $x_j$  and  $w_j \in N(0, Q_j)$ . Models of this type are used to reflect that the problem at hand is not truly one of parameter estimation, and that some (or all) of the components vary in a random (or at least unknown) manner that is statistically bounded. To solve this problem in a SRIF formulation suppose that a priori for  $x_j$  and  $w_j$  are written in data equation form (cf ref. [3]),

$$R_j x_j = z_j - v_j \quad ; \quad v_j \in N(0, I)$$

$$Q_j^{-1/2} w_j = 0 - v_j^{(w)} \quad ; \quad v_j^{(w)} \in N(0, I_{n_w})$$

where  $Q_j^{1/2}$  is a Cholesky factor of  $Q_j$  that is obtainable from COV2RI. Combining these two equations with the one for  $x_{j+1}$  gives

\* In this example it is assumed that all of the  $y_j$  variables have the same dimension. This assumption, though not essential, simplifies our description of the procedure.

$$\begin{bmatrix} I_{n_w} & 0 \\ -R_j G_j Q_j^{1/2} & R_j \end{bmatrix} \begin{bmatrix} \hat{w}_j \\ x_{j+1} \end{bmatrix} = \begin{bmatrix} 0 \\ z_j \end{bmatrix} - \begin{bmatrix} v_j^{(w)} \\ v_j \end{bmatrix}$$

where  $Q_j^{1/2} \hat{w}_j = w_j$ . This is the equation to be triangularized with subroutine THH, i.e.,

$$\begin{array}{l} \text{Dim } w \{ \\ \text{Dim } x \{ \end{array} \begin{array}{c} \overbrace{\begin{bmatrix} I_{n_w} & 0 & 0 \\ -R_j G_j Q_j^{1/2} & R_j & z_j \end{bmatrix}}^{\text{Dim } w} \quad \overbrace{\begin{bmatrix} 0 & 0 \\ R_j & z_j \end{bmatrix}}^{\text{Dim } x} \quad \overbrace{\begin{bmatrix} 0 \\ z_j \end{bmatrix}}^1 \end{array} \xrightarrow{\text{THH}} \begin{bmatrix} R_j^{(w)} & R_j^{(wx)} & z_j^w \\ 0 & R_{j+1} & z_{j+1} \end{bmatrix}$$

When the problem is arranged so that  $Q_j$  is diagonal one can reduce storage and computation. Incidentally, the form of this algorithm allows one to use singular  $Q_j$  matrices.

When the a priori for  $x_j$  and  $Q_j$  are given in U-D factored form, one can obtain the U-D factors for  $x_{j+1}$  as follows:

Let  $Q_j = U^{(q)} D^{(q)} (U^{(q)})^T$  (use COV2UD if necessary)

Set  $\bar{G} = G_j U^{(q)} = [g_1, \dots, g_{n_w}]$ ,  $D^{(q)} = \text{Diag}(d_1, \dots, d_{n_w})$

Apply subroutine RANK1  $n_w$  times, with  $\bar{U}_0 = \bar{U}_j$ ,  $\bar{D}_0 = D_j$

$$\left. \begin{array}{l} (\bar{U}-\bar{D})_k ; d_k, g_k \xrightarrow{\text{RANK1}} (\bar{U}-\bar{D})_{k+1} \\ \text{i.e. } (\bar{U}_k \bar{D}_k \bar{U}_k^T + d_k g_k g_k^T = \bar{U}_{k+1} \bar{D}_{k+1} \bar{U}_{k+1}^T) \end{array} \right\} k = 1, \dots, n_w$$

Then  $U_{j+1} = \bar{U}_{n_w}$ ,  $D_{j+1} = \bar{D}_{n_w}$

Certain filtering problems involve dynamic models of the form

$$x_{j+1} = \phi_j x_j + G_j w_j$$

Given an estimate for  $x_j$ ,  $\hat{x}_j$ , the predicted estimate for  $x_{j+1}$ , denoted  $\tilde{x}_{j+1}$  is simply\*

$$\tilde{x}_{j+1} = \phi_j \hat{x}_j$$

The U-D factors of the estimate error corresponding to the estimate  $\tilde{x}_{j+1}$  can be obtained using the weighted Gram-Schmidt triangularization subroutine

$$[\phi_j \ U_j : G_j]; \text{Diag}(D_j, D^{(q)}) \xrightarrow{WGS} (\tilde{U}_{j+1} - \tilde{D}_{j+1})$$

Subroutine PHIU can be used to construct  $\phi_j * U_j$ . Note that this matrix multiplication updates the estimate too, because it is placed as an addended column to the U matrix.

When the  $w$  and associated  $x$  terms correspond to a colored noise model,  $p_{j+1} = \rho p_j + w_j$ , then it is easier and more efficient to use the colored noise update subroutine UDCOL. Note that here too the estimate is updated by the subroutine calculation because the estimate is an addended column of U.

## II.8 Miscellaneous Uses of the Various ESP Subroutines

In certain parameter analyses we may want to reprocess a set of data suppressing different subsets of variables. In this case the original data should be left unaltered and subroutine A2A1 used to copy A into  $A_1$ , which then can be modified as dictated by the analysis.

Covariance analysis sometimes are initialized using a covariance matrix from a different problem (or a different phase of the same problem). In such cases it may be necessary to permute, delete or insert rows and columns into the covariance matrix; and that can be achieved using subroutine C2C.

If a priori for the problem at hand is given as a covariance matrix then one can compute the corresponding SRIF or U-D initialization using

\* In statistical notation that is commonly used, one writes

$$x(j+1|j) = \phi_j x(j|j)$$

subroutines COV2RI or COV2UD. Of course, if the covariance is diagonal the appropriate R and U-D factors can be obtained more simply. To convert a priori given in the form of an information matrix to a corresponding SRIF matrix one applies subroutine INF2R. To display covariance results corresponding to the SRIF or U-D filter one can use subroutines UTINV, RI2COV and UD2COV. The vector stored covariance results can be displayed in a triangular format using subroutine TWOMAT.

Parameter estimation does not, in the main, involve matrix multiplication. Certain applications, such as coordinate transformations and time propagation are important enough to warrant inclusion in the ESP. For that reason we have included RA (to post multiply a square root information matrix) and PHIU to premultiply a U-covariance factor). Certain time propagation problems involve sparse transition matrices, and for this we have included the subroutine SFU. Other special matrix products involving triangular matrices were not included because we have had no need for other products (to date), and they are generally not lengthy or complicated to construct. We illustrate this point by showing how to compute  $z = Rx$  where R is a triangular vector stored matrix and x is an N vector,

```

II=0
DO 2 I=1,N
SUM=0.           @SUM is Double Precision
II=II+I         @II=(I,I)
IK=II
DO 1 K=1,N
SUM=SUM+R(IK)*x(K) @IK=(I,K)
1  IK=IK+K
2  z(I)=SUM      @z can overwrite x if desired

```

Note that the II and IK incremental recursions are used to circumvent the  $N(N+1)/2$  calculations of  $IK=K(K-1)/2+I$ .

### III. SUBROUTINE DIRECTORY SUMMARY

#### 1. A2A1 - (A to A1)

Reorders the columns of a rectangular matrix A, storing the result in matrix A1. Columns can be deleted and new columns added. Zero columns are inserted which correspond to new column name entries. Matrices A and A1 cannot share common storage.

##### Example III.1

$$\begin{array}{ccc} \alpha & B & C \\ \left[ \begin{array}{ccc} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{array} \right] & \xrightarrow{\text{A2A1}} & \left[ \begin{array}{ccccc} B & F & G & C & H \\ 5 & 0 & 0 & 9 & 0 \\ 6 & 0 & 0 & 10 & 0 \\ 7 & 0 & 0 & 11 & 0 \\ 8 & 0 & 0 & 12 & 0 \end{array} \right] \\ A & & A1 \end{array}$$

The new namelist (BFGCH) contains F, G and H as new columns and deletes the column corresponding to name  $\alpha$ .

##### Example III.2

Suppose one is given an observation data file with regression coefficients corresponding to a state vector with components say,  $x, y, z, v_x, v_y, v_z$  and station location errors. Suppose further, that the vector being estimated has components  $a_r^\dagger, a_x^\dagger, a_y^\dagger$ ,  $x, y, z, v_x, v_y, v_z, GM$  and station location errors. A2A1 can be used to reorder the matrix of regression coefficients to correspond to the state being estimated. Zero coefficients are set in place for the accelerations and GM which are not present in the original file.

---

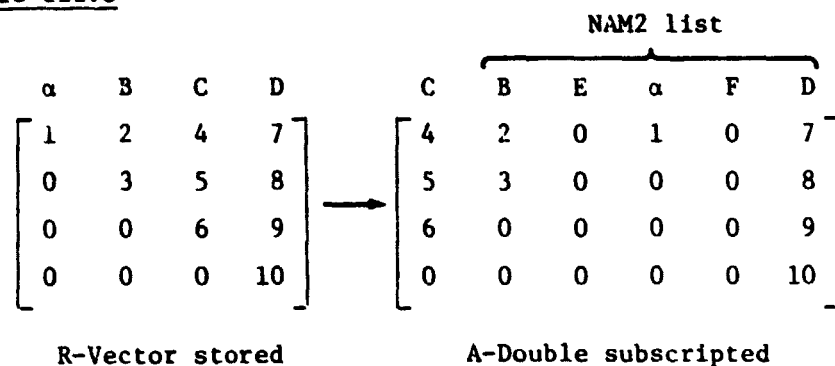
<sup>†</sup> in track and cross track accelerations

2. COMBO - (combine R and A namelists)

The upper triangular vector stored matrix R has its columns permuted and is copied into matrix A. The names associated with R are to be combined with a second namelist.

The namelist for A is arranged so that R names not contained in the second list appear first (left most). These are then followed by the second list. Names in the second list that do not appear in the R namelist have columns of zeros associated with them.

Example III.3



A principal application of this subroutine is to the problem of combining equation sets containing different variables, and automating the process of combining name lists.

3. COVRHO - (covariance to correlation matrix)

A vector stored correlation matrix, RHO, is computed from an input positive semi-definite vector stored matrix, P. Correlations corresponding to zero diagonal covariance elements are zero. To economize on storage the output RHO matrix can overwrite the input P matrix. The principal function of correlation matrices is to expose strong pairwise component correlations ( $|\text{RHO}(IJ)| \leq 1$ , and near unity in magnitude). It is sometimes erroneously assumed that numerical ill-conditioning

of the covariance matrix can be determined by inspecting the correlation matrix entries. While it is true that RHO is better conditioned than is the covariance matrix, it is not true that inspection of RHO is sufficient to detect numerical ill-conditioning. For example, it is not at all obvious that the following correlation matrix has a negative eigenvalue.

$$\text{RHO} = \begin{bmatrix} 1. & 0.49999 & 0.49999 \\ & 1. & -0.49999 \\ & & 1. \end{bmatrix}$$

4. COV2RI - (Covariance to R inverse)

An input positive semi-definite vector stored matrix P is replaced by its upper triangular vector stored Cholesky factor S,  $P = SS^T$ . The name RI is used because when the input covariance is positive definite,  $S = R^{-1}$ .

5. COV2UD - (Covariance to U-D factors)

An input positive semi-definite vector stored matrix P is replaced by its upper triangular vector stored U-D factors.  $P = UDU^T$ .

6. C2C - (C to C)

Reorders the rows and columns of a square (double subscripted) matrix C and stores the result back in C. Rows and columns of zeros are added when new column entries are added.

Example III.4

$$\begin{array}{c} \text{A} \\ \text{B} \\ \text{\Gamma} \end{array} \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \xrightarrow{\text{C2C}} \begin{array}{c} \text{\Gamma} \\ \text{P} \\ \text{B} \\ \text{Q} \end{array} \begin{bmatrix} 9 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 \\ 8 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Names P and Q have been added and name A deleted. An important application of this subroutine is to the rearranging of covariance matrices.



7. INF2R - (Information matrix to R)

Replaces a vector stored positive semi-definite information matrix  $\Lambda$  by its lower triangular Cholesky factor  $R^T$ ;  $\Lambda = R^T R$ . The upper triangular matrix  $R$  is in the form utilized by the SRIF algorithms. The algorithm is designed to handle singular matrices because it is a common practice to omit a priori information on parameters that are either poorly known or which will be well determined by the data.

8. HHPOST - (Householder orthogonal triangularization by post multiplication)

The input, double subscripted, rectangular matrix  $W(M,N)$  ( $M \leq N$ ) is triangularized, and overwritten, by post-multiplying it by an implicitly defined orthogonal transformation, i.e.

$$[ W ]^T \rightarrow [ 0 \setminus S ]$$

This subroutine is used, in the main, to retriangularize a mapped covariance square root and to include in the effects of process noise (i.e.

$W = [\Phi * P^{1/2} : B Q^{1/2}]$ ) and to compute consider covariance matrix square roots (i.e.  $W = [P_{\text{computed}}^{1/2} : S_{\text{en}} * P_y^{1/2}]$ ).

9. PERMUT

Reorders the columns of matrix  $A$ , storing the result back in  $A$ . This routine differs from  $A2A1$  principally in that here the result overwrites  $A$ . PERMUT is especially useful in applications where storage is at a premium or where the problem is of a recursive nature.

10. PHIU - ( $PHI$  (rectangular) \*  $U$  (unit upper triangular))

$$[ PHI ] \begin{array}{|c|} \hline \triangle \\ \hline U \\ \hline \end{array} = [ PHIU ]$$

The matrices  $PHI$  and  $PHIU$  are double subscripted, and  $U$  is vector subscripted with implicitly defined unit diagonal elements. It is not



13. RCOLRD - (colored noise inclusion into the SRIF)

Includes colored noise time updating into the square root information matrix. It is assumed that the deterministic portion of the time update has been completed, and that only the colored noise effects are being incorporated by this subroutine. The algorithm used is Bierman's colored noise one-component-at-a-time update, cf ref. [3], and updates the SRIF array corresponding to the model

$$\begin{bmatrix} x_1 \\ p \\ x_2 \end{bmatrix}_{j+1} = \begin{bmatrix} I & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_1 \\ p \\ x_2 \end{bmatrix}_j + \begin{bmatrix} 0 \\ w_j \\ 0 \end{bmatrix}$$

M is diagonal and  $w_j \in N(0, Q)$ . Auxiliary quantities, useful for fixed interval smoothing, are also generated.

14. RINCON - (R inverse with condition number bound, CNB)

Computes the inverse of an upper triangular vector stored matrix R using back substitution. To economize on storage the output result can overwrite the input matrix. A Frobenius bound (CNB) for the condition number of R is computed too. This bound acts as both an upper and a lower bound, because  $CNB/N \leq \text{condition number} \leq CNB$ . When this bound is within several orders of magnitude of the machine accuracy the computed inverse is not to be trusted, (viz if  $CNB \geq 10^{15}$  on an 18 decimal digit machine R is ill-conditioned).

15. RI2COV - (RI to covariance)

This subroutine computes sigmas (standard deviations) and/or the covariance of a vector stored upper triangular square root covariance matrix, RINV (SRIF inverse). The result, stored in COVOUT (covariance output) is also vector stored. To economize on storage, COVOUT can overwrite RINV.

16. R2A - (R to A)

The columns of a vector stored upper triangular matrix R are permuted and variables are added and/or deleted. The result is stored in the double subscripted matrix A. In other respects the subroutine is like A2A1.

Example III.5

$$\begin{array}{c}
 \begin{array}{ccccc}
 \alpha & B & C & D & E \\
 \left[ \begin{array}{ccccc}
 2 & 4 & 8 & 14 & 22 \\
 0 & 6 & 10 & 16 & 24 \\
 0 & 0 & 12 & 18 & 26 \\
 0 & 0 & 0 & 20 & 28 \\
 0 & 0 & 0 & 0 & 30
 \end{array} \right] & \xrightarrow{\text{R2A}} & \begin{array}{cccc}
 E & F & C & B \\
 \left[ \begin{array}{cccc}
 22 & 0 & 8 & 4 \\
 24 & 0 & 10 & 6 \\
 26 & 0 & 12 & 0 \\
 28 & 0 & 0 & 0 \\
 30 & 0 & 0 & 0
 \end{array} \right] \\
 R & & A
 \end{array}
 \end{array}$$

R is vector stored as R = (2,4,6,8,10,12,14,16,18,20,22,24,26,28,30) with namelist (α,B,C,D,E) associated with it. Names α and D are not included in matrix A, and a column of zeros corresponding to name F is added.

One trivial, but perhaps useful, application is to convert a vector stored matrix to a double subscripted form.<sup>†</sup> R2A is used most often when one wants to rearrange the columns of a SRIF array so that reduced order estimates, sensitivities, etc. can be obtained; or so that data sets containing different parameters can be combined.

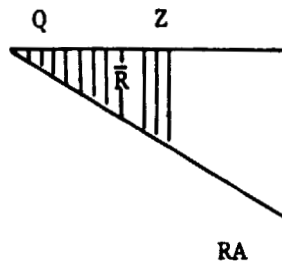
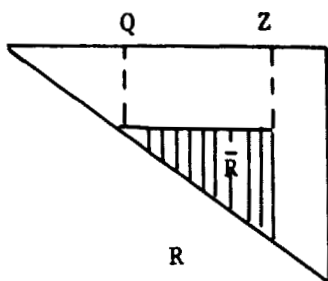
---

<sup>†</sup> see also the aside in the introduction

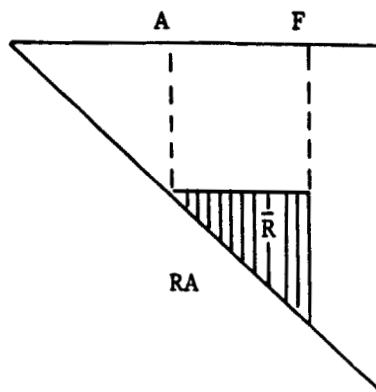
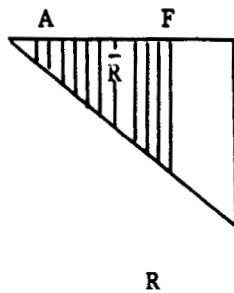
17. R2RA - (Triangular block of R to triangular block of RA)

A triangular portion of the vector stored upper triangular matrix R is put into a triangular portion of the vector stored matrix RA. The names corresponding to the relocated block are also moved. R can coincide with RA.

Examples III.6



or



Note that an upper left triangular submatrix can slide to any lower position along the diagonal, but that a submatrix moving up must go to the upper leftmost corner. Upper shifting is used when one is interested in that subsystem; and the lower shifting is used, for example, when inserting a priori information for consider analyses.

18. RUDR - (SRIF R converted to U-D form or vice versa)

A vector stored SRIF array is replaced by a vector stored U-D form or conversely. A point to be noted is that when data is involved the right side of the SRIF data equation transforms to the estimate in the U-D array.

19. SFU - (Sparse F\*U(Unit upper triangular))

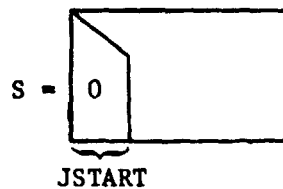
$$[\text{Sparse F}] \begin{array}{|c} \triangle \\ U \end{array} = [FU]$$

A sparse F matrix, with only its nonzero elements recorded, multiplies U which is vector stored with implicit unit diagonal entries. When the input F is sparse this routine is very efficient in terms of storage and computation. When the last column of U contains the estimate, x, the last column of FU represents elements of the mapped estimate F\*x.

20. TDHHT - (Two dimensional Householder Triangularization)

Implicitly defined Householder orthogonal transformations are used to triangularize an input two dimensional rectangular array, S(M,N).

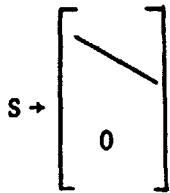
Computation can be reduced if S starts partially triangular;



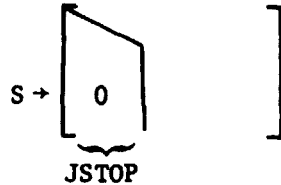
Further, the algorithm implementation is such that (a) maximum triangularization is achievable

when M.LT.N  $S \rightarrow \begin{array}{|c} \triangle \\ 0 \end{array} \quad ]$

when M.GT.N



and finally when an intermediate form is desired

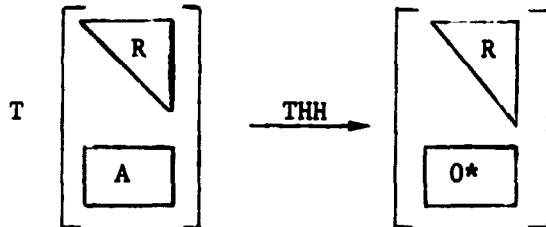


This subroutine can be used to compress overdetermined linear systems of equations to triangular form (for use in least squares analyses). The chief application, that we have in mind, of this subroutine, is to the matrix triangularization of a "mapped" square root information matrix.

This subroutine overlaps to a large extent the subroutine THH which utilizes vector stored, single subscripted, matrices. This latter routine when applicable is more efficient. The triangularization is adapted from ref. [1].

21. THH - (Triangular Householder data packing)

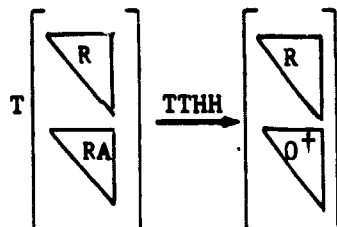
An upper triangular vector stored matrix R is combined with a rectangular doubly subscripted matrix A by means of Householder orthogonal transformations. The result overwrites R, and A is destroyed in the process. This subroutine is a key component of the square root information sequential filter, of ref. [3].



\* The elements are not explicitly set to zero.

22. TTHH - (Two triangular arrays are combined using Householder orthogonal transformations)

This subroutine combines two single subscripted upper triangular SRIF arrays, R and RA using Householder orthogonal transformations. The result overwrites R.



23. TWOMAT - (Two dimensional print of a triangular matrix)

Prints a vector stored upper triangular matrix, using a matrix format.

Example III.7

R(10) = (2,4,6,8,10,12,14,16,18,20) with associated namelist (A,B,C,D) is printed as

	A	B	C	D
A	2	4	8	14
B		6	10	16
C			12	18
D				20

(The numbers are printed as 7 columns of 8 significant floating point digits or 12 columns of 5 significant floating point digits.)

To appreciate the importance of this subroutine compare the vector R(10) with the double subscript representation.

---

† The elements are not explicitly set to zero.



24. TZERO - (Zero a horizontal segment of a vector stored upper triangular matrix)

Upper triangular vector stored matrix R has its rows between ISTART and IFINAL set to zero.

Example III.8

To zero rows 2 and 3 of R(15) of example III.5

R(15) = (2,4,6,8,10,12,14,16,18,20,22,24,26,28,30) is transformed to

R(15) = (2,4,0,8,0,0,14,0,0,20,22,0,0,28,30) i.e.,

$$\begin{array}{ccc}
 \begin{bmatrix} 2 & 4 & 8 & 14 & 22 \\ 0 & 6 & 10 & 16 & 24 \\ 0 & 0 & 12 & 18 & 26 \\ 0 & 0 & 0 & 20 & 28 \\ 0 & 0 & 0 & 0 & 30 \end{bmatrix} & \xrightarrow{\text{TZERO}} & \begin{bmatrix} 2 & 4 & 8 & 14 & 22 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 28 \\ 0 & 0 & 0 & 0 & 30 \end{bmatrix} \\
 \text{R-vector stored} & & \text{R-vector stored}
 \end{array}$$

25. UDCOL - (U-D covariance factor colored noise update)

This subroutine updates the U-D covariance factors corresponding to the model

$$\begin{bmatrix} x_1 \\ p \\ x_2 \end{bmatrix}_{j+1} = \begin{bmatrix} I & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_1 \\ p \\ x_2 \end{bmatrix}_j + \begin{bmatrix} 0 \\ w_j \\ 0 \end{bmatrix}$$

where M is diagonal and  $w_j \sim N(0,Q)$ . The special structure of the transition and process noise covariance matrices is exploited, cf Bierman, [3].

26. UDMEAS - (U-D Measurement Update)

Given the U-D factors of the a priori estimate error covariance and the measurement,  $z = Ax + v$  this routine computes the updated estimate and U-D covariance factors, the predicted residual, the predicted residual variance, and the normalized Kalman gain. This is Bierman's U-D measurement update algorithm, cf [3].

27. UD2COV - (U-D factors to covariance)

The input vector stored U-D matrix (diagonal D elements are stored as the diagonal entries of U) is replaced by the covariance P, also vector stored,  $P = UDU^T$ . P can overwrite U to economize on storage.

28. UD2SIG - (U-D factors to sigmas)

Standard deviations corresponding to the diagonal elements of the covariance are computed from the U-D factors. This subroutine, a restricted version of UD2COV can print out the resulting sigmas and a title. The input U-D matrix is unaltered.

29. UTINV - (Upper triangular matrix inversion)

An upper triangular vector stored matrix RIN (R in) is inverted and the result, vector stored, is put in ROUT (R out). ROUT can overwrite RIN to economize on storage. If a right hand side is included and the bottommost tip of RIN has a -1 set in then ROUT will have the solution in the place of the right hand side.

30. UTIROW - (Upper triangular inversion, inverting only the upper rows)

$$\begin{array}{ccc}
 \text{INPUT} & & \text{OUTPUT} \\
 \left. \begin{array}{c} n_y \\ \left\{ \begin{array}{c} \left[ \begin{array}{cc} R_x & R_{xy} \\ \hline 0 & R_y^{-1} \end{array} \right] \end{array} \right. & \xrightarrow{\text{UTIROW}} & \left[ \begin{array}{cc} R_x^{-1} & -R_x^{-1} R_{xy} R_y^{-1} \\ \hline 0 & R_y^{-1} \end{array} \right]
 \end{array}$$

An input vector stored R matrix with its lower left triangle assumed to have been already inverted is used to construct the upper rows of the matrix inverse of the result. The result, vector stored, can overwrite the input to economize on storage.

If the columns comprising  $R_{xy}$  represent consider terms then taking  $R_y^{-1}$  as the identity gives the sensitivity on the upper right portion of the result. If  $R_y^{-1} = \text{Diag}(\sigma_y, \dots, \sigma_{n_y})$  then the upper right portion of the result represents the perturbation. Note that if z (the right hand side of the data equation) is included in  $R_{xy}$  then taking the corresponding  $R_y^{-1}$  diagonal as -1 results in the filter estimate appearing as the corresponding column of the output array. When  $n_y$  is zero this subroutine is algebraically equivalent to UTINV. The subroutines differ when a zero diagonal is encountered. UTINV gives the correct inverse for the columns to the left of the zero element, whereas UTIROW gives the correct inverse for the rows below the zero element.

31. WGS - (Weighted Gram-Schmidt U-D matrix triangularization)

An input rectangular (possibly square) matrix  $W$  and a diagonal weight matrix,  $D_w$ , are transformed to (U-D) form; i.e.,

$$S D_w W^T = UDU^T$$

where  $U$  is unit upper triangular and  $D$  is diagonal. The weights  $D_w$  are assumed nonnegative, and this characteristic is inherited by the resulting  $D$ .

#### IV. SUBROUTINE DIRECTORY USER DESCRIPTION

##### 1. A2A1 (A to A1)

###### Purpose

To rearrange the columns of a namelist indexed matrix to conform to a desired namelist.

**CALL A2A1(A, IA, IR, LA, NAMA, A1, IA1, LA1, NAMAL)**

###### Argument Definitions

A(IR,LA)	Input rectangular matrix
IA	Row dimension of A, IA.GE.IR
IR	Number of rows of A that are to be arranged
LA	Number of columns in A; this also represents the number of parameter names associated with A
NAMA(LA)	Parameter names associated with A
A1(IR,LA1)	Output rectangular matrix
IA1	Row dimension of A1, IA1.GE.IR
LA1	Number of columns in A1; this also represents the number of parameter names associated with A1
NAMAL(LA1)	Input list of parameter names to be associated with the output matrix A1

###### Remarks and Restrictions

A1 cannot overwrite A. This subroutine can be used to add on columns corresponding to new names and/or to delete variables from an array.

###### Functional Description

The columns of A are copied into A1 in an order corresponding to the NAMAL parameter namelist. Columns of zeros are inserted in those A1 columns which do not correspond to names in the input parameter namelist NAMA.

## 2. COMBO (Combine parameter namelists)

### Purpose

To rearrange a vector stored triangular matrix and store the result in matrix A. The difference between this subroutine and R2A is that there the namelist for A is input; here it is determined by combining the list for R with a list of desired names.

```
CALL COMBO (R,L1,NAM1,L2,NAM2,A,IA,LA,NAMA)
```

### Argument Definitions

R(L1*(L1+1)/2)	Input vector stored upper triangular matrix
L1	No. of parameters in R (and in NAM1)
NAM1(L1)	Names associated with R
L2	No. of parameters in NAM2
NAM2(L2)	Parameter names that are to be combined with R (NAM1 list); these names may or may not be in NAM1
A(L1,LA)	Output array containing the rearranged R matrix L1.LE.IA
IA	Row dimension of A
LA	No. of parameter names in NAMA, and the column dimension of A. LA = L1 + L2 - No. names common to NAM1 and NAM2; LA is computed and output
NAMA(LA)	Parameter names associated with the output A matrix; consists of names in NAM1 which are not in NAM2, followed by NAM2

### Remarks and Restrictions

The column dimension of A is a result of this subroutine. To avoid having A overwrite neighboring arrays one can bound the column dimension of A by L1+L2.

### Functional Description

First the NAM1 and NAM2 lists are compared and the names appearing in NAM1 only have their corresponding R column entries stored in A (e.g. if NAM1(2) and NAM1(6) are the only names not appearing in the NAM2 list then columns 2 and 6 of R are copied into columns 1 and 2 of A). The remaining columns of A are labeled with NAM2. The A namelist is recorded in NAMA. The NAM1 list is compared with NAM2 and matching names have their R column entries copied into the appropriate columns of A. NAM2 entries not appearing in NAM1 have columns of zero placed in A.

### 3. COVRHO (Covariance to correlation matrix, RHO)

#### Purpose

To compute the correlation matrix RHO from an input covariance matrix COV. Both matrices are upper triangular, vector stored and the output can overwrite the input.

```
CALL COVRHO(COV,N,RHO,V)
```

#### Argument Definitions

COV(N*(N+1)/2)	Input vector stored positive semi-definite covariance matrix
N	Model dimension, N.GE.1
RHO(N*(N+1)/2)	Output vector stored correlation matrix
V(N)	Work vector

#### Remarks

No test for non-negativity of the input matrix is made. Correlations corresponding to negative or zero diagonal entries are set to zero, as is the diagonal output entry.

#### Functional Description

$$V(I) = 1/\sqrt{COV(I,I)} \text{ if } COV(I,I) > 0 \text{ and } 0 \text{ otherwise}$$
$$RHO(I,J) = COV(I,J)*V(I)*V(J)$$

The subroutine employs, however, vector stored COV and RHO matrices.



4. COV2RI (Covariance to Cholesky Square Root, RI)

Purpose

To construct the upper triangular Cholesky factor of a positive semi-definite matrix. Both the input covariance and the output Cholesky factor (square root) are vector stored. The output overwrites the input. Covariance (input) = (CF)\*(CF)\*\*T (output CF = Rinverse). If the input covariance is singular, the output factor has zero columns.

CALL COV2RI(CF,N)

Argument Definitions

CF(N*(N+1)/2)	Contains the input vector stored covariance matrix (assumed positive definite) and on output it contains the upper triangular Cholesky factor
N	Dimension of the matrices involved, N.GE.2

Remarks and Restrictions

No check is made that the input matrix is positive semi-definite. Singular factors (with zero columns) are obtained if the input is (a) in fact singular, (b) ill-conditioned, or (c) in fact indefinite; and the latter two situations are cause for alarm. Case (c) and possibly (b) can be identified by using RI2COV to reconstruct the input matrix.

Functional Description

An upper triangular Cholesky reduction of the input matrix is implemented using a geometric algorithm described in Ref. [3].

$$CF(input) = CF(output)*CF(output)^T$$

At each step of the reduction diagonal testing is used and negative terms are set to zero.

## 5. COV2UD (Covariance to UD factors)

### Purpose

To obtain the U-D factors of a positive semi-definite matrix. The input vector stored matrix is overwritten by the output U-D factors which are also vector stored.

**CALL COV2UD(U,N)**

### Argument Definitions

U(N*(N+1)/2)	Contains the input vector stored covariance matrix; on output it contains the vector stored U-D covariance factors.
N	Matrix dimension, N,GE.2

### Remarks and Restrictions

No checks are made in this routine to test that the input U matrix is positive semi-definite. Singular results (with zero columns) are obtained if the input is (a) in fact singular, (b) ill-conditioned, or (c) in fact indefinite; and the latter two situations are cause for alarm. Case (c) and possibly case (b) can be identified by using UD2-COV to reconstruct the input matrix. Note that although indefinite matrices have U-D factorizations, the algorithm here applies only to matrices with non-negative eigenvalues.

### Functional Description

An upper triangular U-D Cholesky factorization of the input matrix is implemented using a geometric algorithm described in Ref. [3].

$$U(\text{input}) = U * D * U^T, \quad U-D \text{ overwrites the input } U$$

at each step of the reduction diagonal testing is used to zero negative terms.

6. C2C (C to C)

Purpose

To rearrange the rows and columns of C, from NAM1 order to NAM2 order. Zero rows and columns are associated with output defined names that are not contained in NAM1.

`CALL C2C(C,IC,L1,NAM1,L2,NAM2)`

Argument Definitions

C(L1,L1)	Input matrix
IC	Row dimension of C IC.GE.L = MAX(L1,L2)
L1	No. of parameter names associated with the input C
NAM1(L)	Parameter names associated with C on input. (Only the first L1 entries apply to the input C)
L2	No. of parameter names associated with the output C
NAM2(L2)	Parameter names associated with the output C

Remarks and Restrictions

The NAM2 list need not contain all the original NAM1 names and L1 can be .GE. or .LE. L2. The NAM1 list is used for scratch and appears permuted on output. If L2.GT.L1 the user must be sure that NAM1 has L2 entries available for scratch purposes.

Functional Description

The rows and columns of C and NAM1 are permuted pairwise to get the names common to NAM1 and NAM2 to coalesce. Then the remaining rows and columns of C(L2,L2) are set to zero.

## 7. HHPOST (Householder Post Multiplication Triangularization)

### Purpose

To employ Householder orthogonal transformations to triangularize an input rectangular W matrix by post multiplication, i.e.

$$\begin{bmatrix} W \end{bmatrix}^T = \begin{bmatrix} 0 \backslash S \end{bmatrix}$$

This algorithm is employed in various covariance square root updates.

`CALL HHPOST(S,W,MROW,NROW,NCOL,V)`

### Argument Definitions

S(NROW*(NROW+1)/2)	Output upper triangular vector stored square root matrix
W(NROW,NCOL)	Input rectangular square root covariance matrix (W is destroyed by computations)
MROW	Maximum row dimension of W
NROW	Number of rows of W to be triangularized and the dimension of S (NROW.GE.2)
NCOL	Number of column of W (NCOL.GE.NROW)
V(NCOL)	Work vector

### Functional Description

Elementary Householder transformations are applied to the rows of W in much the same way as they are applied to obtain subroutine THY. The orthogonalization process is discussed at length in the books by Lawson and Hanson [1] and Bierman [3].

8. INF2R (Information matrix to R)

Purpose

To compute a lower triangular Cholesky factorization of an input positive semi-definite matrix. The result transposed, is vector stored; this is the form of an upper triangular SRIF matrix.

CALL INF2R(R,N)

Argument Definitions

R(N*(N+1)/2)	Input vector stored positive semi-definite (information) matrix; on output it represents the transposed lower triangular Cholesky factor (i.e. the SRIF R matrix)
N	Matrix dimension, N.GE.2

Remarks and Restrictions

No checks are made on the input matrix to guard against negative eigenvalues of the input, or to detect ill-conditioning. Singular output matrices have one or more rows of zeros.

Functional Description

A Cholesky type lower triangular factorization of the input matrix is implemented using the geometric formulation described in Ref. [3].

$$R(\text{input}) = [R(\text{output})]^T * [R(\text{output})]$$

At each step of the factorization diagonal testing is used to zero columns corresponding to negative entries. The result is vector stored in the form of a square root information matrix as it would be used for SRIF analyses.

## 9. PERMUT (Permute A)

### Purpose

To rearrange the columns of a namelist indexed matrix to conform to a desired namelist. The resulting matrix is to overwrite the input.

```
CALL PERMUT(A,IA,IR,L1,NAM1,L2,NAM2)
```

### Argument Definitions

A(IR,L)	Input rectangular matrix, $L = \max(L1,L2)$
IA	Row dimension of A, $IA \geq IR$
IR	Number of rows of A that are to be rearranged
L1	Number of parameter names associated with the input A matrix
NAM1(L)	Parameter names associated with A on input (only the first L1 entries apply to the input A)
L2	Number of parameter names associated with the output A matrix
NAM2	Parameter names associated with the output A

### Remarks and Restrictions

This subroutine is similar to A2A1; but because the output matrix in this case overwrites the input there are several differences. The NAM1 vector is used for scratch, and on output it contains a permutation of the input NAM1 list. The user must allocate  $L = \max(L1,L2)$  elements of storage to NAM1. The extra entries, when  $L2 > L1$ , are used for scratch.

### Functional Description

The columns of A are rearranged, a pair at a time, to match the NAM2 parameter namelist. The NAM1 entries are permuted along with the columns, and this is why  $\dim(NAM1)$  must be larger than L1 (when  $L2 > L1$ ). Columns of zeroes are inserted in A which correspond to output names that do not appear in NAM1.

10. PHIU (PHI-rectangular\*U-unit upper triangular)

Purpose

To multiply a rectangular two dimensional matrix PHI by a unit upper triangular vector stored matrix U, and store the result in PHIU. The PHIU matrix can overwrite PHI to economize on storage.

$$[\text{PHI}] \begin{array}{|c} \diagdown \\ \text{U} \\ \diagup \end{array} = [\text{PHIU}]$$

CALL PHIU(PHI,MAXPHI,IRPHI,JCPHI,U,N,PHIU,MPHIU)

Argument Definitions

PHI(IRPHI,JCPHI)	Input rectangular matrix IRPHI.LE MAXPHI
MAXPHI	Row dimension of PHI
IRPHI	number of rows of PHI
JCPHI	number of columns of PHI
U(N*(N+1)/2)	unit upper triangular vector stored matrix
N	U-matrix dimension, JCPHI.LE.N
PHIU(IRPHI,N)	output result PHI*U,PHIU can overwrite PHI
MPHIU	row dimension of PHIU

Remarks and Restrictions

If JCPHI.LT.N it is assumed that there are implicitly defined trailing columns of zeros in PHI. The unit diagonal entries of U are implicit, i.e. the diagonal U entries are not explicitly used.

Functional Description

$$\text{PHIU} = \text{PHI} * \text{U}$$

## 11. RA (R-upper triangular\*A-rectangular)

### Purpose

To post multiply a vector stored triangular matrix, R, by a rectangular matrix A, and if desired to store the result in A.

$$\begin{array}{|c} \text{R} \\ \hline \end{array} \begin{bmatrix} \text{A} \end{bmatrix} = \begin{bmatrix} \text{RA} \end{bmatrix}$$

`CALL RA(R,N,A,MAXA,IA,JA,RA,MAXRA,IRA)`

### Argument Definitions

R(N*(N+1)/2)	upper triangular, vector stored input
N	order of R
A(IA,JA)	Input rectangular right multiplier matrix
MAXA	Row dimension of input A matrix
IA	Number of rows of A that are input
JA	Number of columns of A
RA(IRA,JA)	Output resulting rectangular matrix RA can overwrite A
MAXRA	Row dimension of RA
IRA	Number of rows in the output result (IRA.LE.MAXRA)

### Functional Description

The first IRA rows of the product R\*A are computed using the vector stored input matrix R, and the output can, if desired, overwrite the input A matrix. When N.GT.IA (i.e. there are more columns of R than rows of A) then it is assumed that the bottom N-IA rows of A are implicitly defined as a partial identity matrix, i.e.

$$A = \left[ \begin{array}{cc} \text{-(Input)} & - \\ 0 & I \end{array} \right] \begin{array}{l} \text{ } IA \\ \text{ } N-IA \end{array}$$



12. RANK1 (Stable U-D rank one update)

Purpose

To compute the (updated) U-D factors of  $UDU^T + CVV^T$ .

CALL RANK1(UIN,UOUT,N,C,V)

Argument Definitions

UIN(N*(N+1)/2)	Input vector stored positive semi-definite U-D array (with the D entries stored on the diagonal of U)
UOUT(N*(N+1)/2)	Output vector stored positive (possibly) semi-definite U-D result, UOUT=UIN is allowed.
N	Matrix dimension, N.GE.2
C	Input scalar, which should be non-negative. C is destroyed by the algorithm.
V(N)	Input vector for the rank one modification. V is destroyed by the algorithm.

Remarks and Restrictions

If C negative is used the algorithm is numerically unstable, and the result may be numerically unreliable. Singular U matrices are allowed, and these can result in singular output U Matrices. The code switches from a 1-multiply to a 2-multiply mode at a key place, based upon a 1/16 comparison of input to output D values. Also, there is provision made to supply a machine accuracy epsilon when single precision is specified.

Functional Description

This rank one modification is based on a result published by Agee and Turner (1972), White Sands Missile Range Tech. Report No. 38 and improved on using a numerical stabilization idea due to Gentlemen (1973). The algorithm is derived in the chapter,

" $UDU^T$  Covariance Factorization For Kalman Filtering," C. L. Thornton,  
G. J. Bierman, Vol. XVI of Advances in Control of Dynamic Systems,  
Academic Press, to appear 1979.

13. RCOLRD (Colored noise time update of the SRIF R matrix)

Purpose

To include colored noise time updating into the square root information matrix. It is assumed that the deterministic portion of the time update has been completed, and that only the colored noise effects are being incorporated by this subroutine.

CALL RCOLRD(S,MAXS,IRS,JCS,NPSTRT,NP,EM,RW,ZW,V,SGSTAR)

Argument Definitions

S(IRS,JCS) Input rectangular portion of the square root information matrix corresponding to the nonconstant parameters. It is assumed that estimates are included, i.e. the last column represents the "right hand side", Z, (but see JCS description). S also houses the time updated array, and if there is smoothing there are NP extra rows adjoined to S.

MAXS Row dimension of S. If smoothing calculations are to be included then MAXS.GE.IRS+NP.

IRS The number of rows of S, i.e. the number of nonconstant parameters (including colored noise variables). IRS.GE.2

JCS The number of columns of S. If the vector ZW is zero, then the right hand side of transformed estimates need not be included.

NPSTRT Location of the first colored process noise variable.

NP The number of colored noise variables contiguous to and following the first.

EM(NP) Vector of exponential colored noise multipliers (EM = exp (-DT/TAU))

RW(NP) Vector of positive reciprocal colored process noise standard deviations, i.e.  
 $p_{j+1} = \exp(-DT/\tau) * p_j + w_j, R_w = 1/\sigma_w$

ZW(NP)                    Vector of normalized process noise a priori estimates. ZW is generally zero.

V(IRS)                    Work vector.

SGSTAR(NP)                Vector of smoothing coefficients. Needed only if smoothing is to be done.

Remarks and Restrictions

There are three lines of code associated with smoothing, and these are commented out of the nominal case. Therefore, if smoothing is contemplated the comments must be removed. The vector SGSTAR is involved only with smoothing. Last note: for smoothing, be sure that S has NP extra rows to house the smoothing coefficients.

The ZW vector is generally zero. If ZW = 0 one has the option of doing covariance only analyses and the last column of S (the right hand side of normalized estimates) can be omitted.

Because of the large number of arguments appearing in this subroutine, and because almost all of them are constant (i.e. with succeeding calls only S, and possible EM, RW, ZW and SGSTAR change) for a given problem, it is suggested that one a) introduce COMMON, b) use this as an internal subroutine, or c) write in-line code.

Functional Description

The model is

$$\begin{bmatrix} x_1 \\ p \\ x_2 \end{bmatrix}_{j+1} = \begin{bmatrix} I & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_1 \\ p \\ x_2 \end{bmatrix}_j + \begin{bmatrix} 0 \\ w_j \\ 0 \end{bmatrix} \begin{matrix} \text{]NPSTRT-1} \\ \text{]NP} \\ \text{]N-(NPSTRT-1+NP)} \end{matrix}$$

where M is diagonal, with NP non-negative entries and  $w_j$  is a white noise process with  $w_j \in N(\bar{w}, Q)$ ,  $Q = R_w^{-1} R_w^{-T}$ . The algorithm is based on Bierman's one component-at-a-time SRIF time update which economizes

on storage and computation (see Bierman-Factorization Methods for Discrete Sequential Estimation, Academic Press 1977).

When smoothing is contemplated, there is output a vector  $\sigma^*(NP)$  and a matrix  $S^*(NP, N+1)$ ;  $S^*$  occupies the bottom  $NP$  rows of the output  $S$  matrix. Smoothed estimates of the  $p$  terms can be obtained from the  $\sigma^*$  and  $S^*$  terms as follows:

Let  $X^*$  be the previously computed estimates of the  $N$  filter parameters, then for  $J = NP, NP-1, \dots, 1$  recursively compute

$$X^*(NSTRT + J-1) := (S^*(J, N+1) - \sum_{K=1}^N S^*(J, K)X^*(K)) / \sigma^*(J)$$

Note that the symbol " := " means is replaced by, so that the old values of  $X^*$ , on the right side, are over-written by the new smoothed colored noise estimates. Smoothed covariances can be obtained from the  $S^*$  and  $\sigma^*$  terms as well, but we do not go into detail here; the reader is directed to chapter 10 of the Bierman reference.

#### 14. RINCON (R inverse with condition number bound)

##### Purpose

To compute the inverse of an upper triangular vector stored triangular matrix, and an estimate of its condition number.

CALL RINCON(RIN,N,ROUT,CNB)

##### Argument Definitions

RIN(N*(N+1)/2)	Input vector stored upper triangular matrix
N	Matrix dimension, N.GE.2
ROUT(N*(N+1)/2)	Output vector stored matrix inverse (RIN = ROUT is permitted)
CNB	Condition number bound. If $\kappa$ is the condition number of RIN, then CNB/N.LE. $\kappa$ .LE CNB

##### Remarks and Restrictions

The condition number bound, CNB serves as an estimate of the actual condition number. When it is large the problem is ill-conditioned.

##### Functional Description

The matrix inversion is carried out using a triangular back substitution. If any diagonal element of the input R matrix is zero the condition number computation is aborted. When the first zero occurs at diagonal k the matrix inversion is carried out only on the first k-1 columns. The condition number bound is computed as follows:

$$\text{F.NORM } R = \sum_{J=1}^{\text{NTOT}} R(J)^2$$

$$\text{F.NORM } R^{-1} = \sum_{J=1}^{\text{NTOT}} R^{-1}(J)^2$$

where  $NTOT = N*(N+1)/2$  is the number of elements in the vector stored triangular matrix. The condition number bound, CNB, is given by

$$CNB = (F.NORM R * F.NORM R^{-1})^{1/2}$$

F.NORM is the Frobenius norm, squared. The inequality

$$CNB/N \leq \text{condition number } R \leq CNB$$

is a simple consequence of the Frobenius norm inequalities given in Lawson-Hanson "Solving Least Squares," page 234.

15. RI2COV (RI Triangular to covariance)

Purpose

To compute the standard deviations, and if desired, the covariance matrix of a vector stored upper triangular square root covariance matrix. The output covariance matrix, also vector stored, can overwrite the input.

```
CALL RI2COV(RINV,N,SIG,COVOUT,KROW,KCOL)
```

Argument Definitions

RINV(N*(N+1)/2)	Input vector stored upper triangular covariance square root (RINV=Rinverse is the inverse of the SRIF matrix).	
N	Dimension of the RINV matrix	
SIG(N)	Output vector of standard deviations	
COVOUT(N*(N+1)/2)	Output vector stored covariance matrix (COVOUT = RINV is allowed)	
KROW	{ .GT.0	Computes the covariance and sigmas corresponding to the first KROW variables of the RINV matrix
	{ .LT.0	Computes only the sigmas of the first (KROW) variables of the RINV matrix.
	{ .EQ.0	No covariance, but all sigmas (e.g. use all N rows of RINV)
KCOL	Number of columns of COVOUT that are computed, If KCOL.LE.0, then KCOL = KROW.	

Remarks and Restrictions

Replacing N by |KROW| corresponds to computing the covariance of a lower dimensional system.

Functional Description

COVOUT=RINV\*RINV\*\*T



## 16. R2A (R to A)

### Purpose

To place the upper triangular vector stored matrix R into the matrix A and to arrange the columns to match the desired NAMA parameter list. Names in the NAMA list that do not correspond to any name in NAMR have zero entries in the corresponding A columns.

```
CALL R2A(R,LR,NAMR,A,IA,LA,NAMA)
```

### Argument Definitions

R(LR*(LR+1)/2)	Input upper triangular vector stored array
LR	No. of parameters associated with R
NAMR(LR)	Parameter names associated with R
A(LR,LA)	Matrix to house the rearranged R matrix
IA	Row dimension of A, IA.GE.LR.
LA	No. of parameter names associated with the output A matrix.
NAMA(LA)	Parameter names for the output A matrix.

### Functional Description

The matrix A is set to zero and then the columns of R are copied into A.

17. R2RA (Permute a subportion  $R_A$  of a vector stored triangular matrix)

Purpose

To copy the upper left (lower right) portion of a vector stored upper triangular matrix R into the lower right (upper left) portion of a vector stored triangular matrix RA.

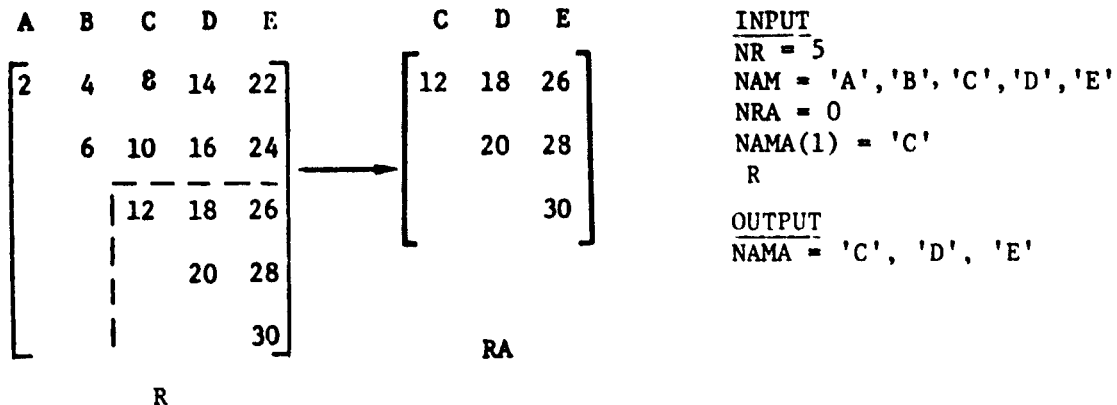
CALL R2RA(R, NR, NAM, RA, NRA, NAMA)

Argument Definitions

R(NR*(NR+1)/2)	Input vector stored upper triangular matrix
NR	Dimension of vector stored R matrix <sup>†</sup>
NAM(NR)	Names associated with R.
RA(NRA*(NRA+1)/2)	Output vector stored upper triangular matrix
NRA	If NRA = 0 on input, then NAMA(1) should have the first name of the output namelist. In this case the number of names in NAMA, NRA, will be computed. The lower right block of R will be the upper left block of RA.  If NRA = last name of the upper left block that is to be moved then this upper block is to be moved to the lower right corner of RA. When used in this mode NRA=NR on output <sup>†</sup> .
NAMA(NRA)	Names associated with RA. Note that NRA used here denotes the output value of NRA.

Remarks and Restrictions

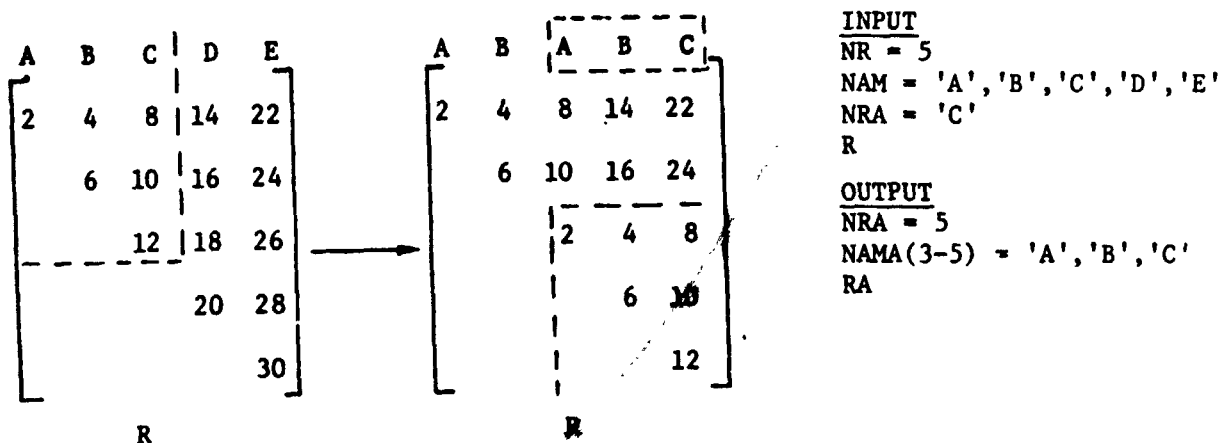
RA and NAMA can overwrite R and NAM. The meaning of the NRA = 0 option is clarified by the following example:



<sup>†</sup> see the concluding paragraph of Remarks and Restrictions

When  $NRA = 0$  and  $NAMA(1) = 'C'$  we are asking that the lower triangular portion of  $R$ , beginning at the column labeled  $C$ , be moved to form the first (in this case 3) columns of  $RA$ . Incidentally,  $RA$  could have additional columns; these columns and their names would be unaltered by the subroutine.

The meaning of the other  $NRA$  option is illustrated by the following example;



When  $NRA = 'C'$  we are asking that the upper left block of  $R$ , up to the column labeled  $C$ , be moved to the lower right portion of  $RA$  and the corresponding names be moved too. If  $RA$  overwrites  $R$ , as in the example, then the first two rows of  $R$  remain unchanged and since  $NAMA$  overwrites  $NAM$ , the labels of the first two columns remain unaltered.

The remark that  $NRA=NR$  on output means, in this example, that the column with name  $C$  in  $R$  is moved over to column 5. If one wanted to slide the upper left triangle corresponding to names  $ABC$  of  $R$  to columns 7-9 of an  $RA$  matrix (of unspecified dimension,  $\geq 9$ ), then one should set  $NR=9$  in the subroutine call. Thus  $NR$ , when used in this sliding down the diagonal mode, does not represent the dimension of  $R$ ; but indicates how far the slide will be.

18. RUDR (R to U-D or U-D to R)

Purpose

To transform an upper triangular vector stored SRIF array to U-D form or vice versa.

CALL RUDR(RIN,N,ROUT,IS)

Argument Definitions

RIN(NBAR\*(NBAR+1)/2) Input upper triangular vector stored SRIF or U-D array; NBAR = ABS(N) + 1

ROUT(NBAR\*(NBAR+1)/2) Output upper triangular vector stored U-D or SRIF array (RIN = ROUT is permitted)

N Matrix dimension, N.GT.0 represents an R to U-D conversion and N.LT.0 represents a U-D to R conversion. ABS(N).GE.2

IS If IS = 0 the input array is assumed not to contain a right side (or an estimate), and IS = 1 means an appropriate additional column is included. In the IS = 0 case the last column of RIN is ignored and NBAR = ABS(N) is used.

Subroutine used: RINCON

Functional Description

Consider the  $N \geq 0$  case. RIN = R is transformed to ROUT = R inverse using subroutine RINCON with dimension  $N + IS$ . If  $IS = 1$  the subroutine sets  $RIN((N+1)(N+2)/2) = -1$ , so that the  $N+1$ st column of ROUT will be the X estimate followed by -1.  $R^{-1} = UD^{1/2}$  so that the diagonals are square root scaled U columns. This information is used to construct the U-D array which is written in ROUT.

If  $N < 0$  the input is assumed to be a U-D array. This array is converted to  $ROUT = UD^{1/2}$  and then using RINCON, R is computed and stored in ROUT. If  $IS = 1$  the U-D matrix is assumed augmented by X (estimate), and on output the right side term of the SRIF array is obtained. When  $IS = 1$ , the initial value of  $RIN((N+1)(N+2)/2)$  is restored before exiting the subroutine.

19. SFU (Sparse F \* unit upper triangular U)

Purpose

To efficiently form the product  $F*U$  so that only the nonzero elements of  $F$  are employed and so that the structure of the  $U$  matrix is utilized (upper triangular with implicit unit diagonal elements). When  $F$  is sparse there are significant savings in storage and computation. Note that since we deal only with the nonzero elements of  $F$  we are saved the time associated with computing unnecessary  $F$  matrix element addresses.

```
CALL SFU(FEL,IPOW,JCOL,NF,U,N,FU,MAXFU,IFU,JDIAG)
```

Argument Definitions

FEL(NF)	Values of the non-zero elements of the $F$ matrix
IROW(NF)	Row indices of the $F$ elements
JCOL(NF)	Column indices of the $F$ elements $F(IROW(K), JCOL(K)) = FEL(K)$
NF	The number of non-zero elements of the $F$ matrix
$U(N*(N+1)/2)$	Upper triangular, vector stored matrix with implicitly defined unit diagonal elements. Note that $U(JJ)$ terms are not, in fact, unity.
$N$	Dimension of the $U$ matrix
FU(IFU,N)	The output result
MAXFU	Row dimension of the $FU$ matrix
IFU	Number of rows in $FU$ . $IFU \leq MAXFU$ , and $IFU \geq \max(IROW(K), K=1, \dots, NF)$ ; i.e. $FU$ must have at least as many rows as does $F$ . Additional rows of $FU$ could correspond to zero rows of $F$ .
JDIAG(N)	Diagonal element indices of a vector stored upper triangular matrix, i.e. $JDIAG(K) = K*(K+1)/2 = JDIAG(K-1) + K$ .

Example:

F(3,12) with: F(1,1) = .9, F(2,2) = .8, F(3,3) = 1.1,  
F(1,7) = 1.7, F(2,8) = -2.8 and F(3,11) = 3.11.

In this case F has NF = 6 (nonzero elements); and one may  
take

IROW(1) = 1	JCOL(1) = 1	FEL(1) = .9
IROW(2) = 2	JCOL(2) = 2	FEL(2) = .8
IROW(3) = 3	JCOL(3) = 3	FEL(3) = 1.1
IROW(4) = 1	JCOL(4) = 7	FEL(4) = 1.7
IROW(5) = 2	JCOL(5) = 8	FEL(5) = -2.8
IROW(6) = 3	JCOL(6) = 11	FEL(6) = 3.11

Remarks and Restrictions

Comments regarding increased efficiency are included in the code.

Functional Description

We write

$$F = \sum_{i,j} F_{ij} e_i e_j^T$$

where  $e_i$  is the i-th unit vector. Then

$$FU = \sum_{ij} F_{ij} e_i (e_j^T U)$$

The code is based on this equation.

20. TDHHT (Two dimensional Householder triangularization)

Purpose

To transform a two dimensional rectangular matrix to a triangular, or partially triangular form by Householder orthogonal matrix pre-multiplication. This subroutine can be used to compress overdetermined linear systems to triangular (double subscripted form) in much the same way as does the subroutine THH (which outputs a vector subscripted triangular result). For recursive applications THH is computationally more efficient and requires less storage. The chief application, that we have in mind, for this subroutine is to the matrix triangularization of "mapped" square root information matrices of the form  $S(m,n)$  with  $m$  less than  $n$ .

```
CALL TDHHT(S,MAXS,IRS,JCS,JSTART,JSTOP,V)
```

Argument Definitions

S(IRS,JCS)	Input (possibly partially) triangular matrix. The output (possibly partially) triangular result overwrites the input.
MAXS	Row dimension of S matrix
IRS	Number of rows in S (IRS.LE.MAXS), and IRS.GE.2.
JCS	Number of columns in S
JSTART	Index of first column to be triangularized. If JSTART.LT.1 then it is assumed that the triangularization starts at column 1.
JSTOP	Index of last column to be triangularized. When JSTOP is not between max(1,JSTART) and JCS then the triangularization is carried out as far as possible (i.e. to IRS if S has less rows than columns, or to JCS if it has more rows than columns).
V(IRS)	Work vector

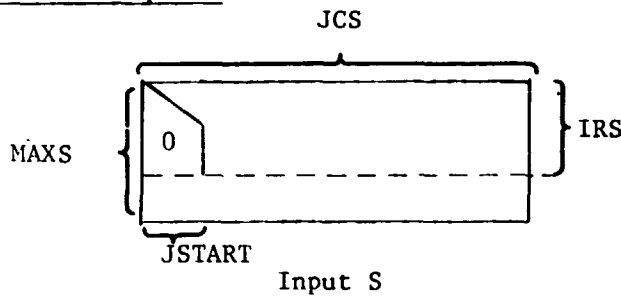
Remarks and Restrictions

The indices JSTART and JSTOP are input for efficiency purposes. When it is known that the input matrix is partially triangular one can by-pass the corresponding (initial) Householder reduction steps. Further, for certain applications it is not necessary to totally triangularize the input array. For example if  $S(m,n)$  and  $m$  is less than  $n$ , the system is in triangular form after only  $m$  elementary Householder reduction steps, i.e

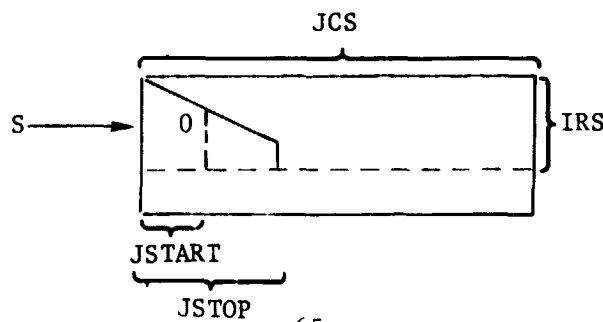
$$T \left[ \begin{array}{c} \overbrace{\hspace{1cm}}^n \\ S \end{array} \right] \}_m \rightarrow \left[ \begin{array}{c} \overbrace{\hspace{1cm}}^m \\ \underbrace{\hspace{1cm}}_n \end{array} \right] \}_m$$

The code is set up so that it defaults to the largest possible upper triangularization.

Functional Description



The dotted portion of the matrix and the block of zeros are not employed at all in the computations. The input matrix is transformed to (possibly partially) triangular form by premultiplication by a sequence of elementary Householder orthogonal transformations.





The method is described fully in the books by Lawson and Hanson -  
Solving Least Squares Problems, and in Bierman - Factorization  
Methods for Discrete Sequential Estimation.

21. THH (Triangular Householder Orthogonalization)

Purpose

To compute [R:z] such that

$$T \begin{bmatrix} \tilde{R} & \tilde{z} \\ A & z \end{bmatrix} = \begin{bmatrix} \hat{R} & \hat{z} \\ 0 & e \end{bmatrix} \quad T - \text{orthogonal}$$

This is the key algorithm used in the square root information batch sequential filter.

CALL THH(R,N,A,IA,M,RSOS,NSTRT)

Argument Definitions

R(N*(N+3)/2)	Input upper triangular vector stored square root information matrix. If estimates are involved RSOS.GE.0 and R is augmented with the right hand side (stored in the last N locations of R). If RSOS.LT.0 only the first N*(N+1)/2 locations of R are used. The result of the subroutine overwrites the input R
N	Number of parameters
A(M,N+1)	Input measurement matrix. The N+1st column is only used if RSOS.GE.0, in which case it represents the right side of the equation $v + AX = z$ . A is destroyed by the algorithm, but it is not explicitly set to zero.
IA	Row dimension of A
M	The number of rows of A that are to be combined with R (M.LE.IA)
RSOS	Accumulated residual root sum of squares corresponding to the data processed prior to this time. On exit RSOS represents the updated root sum of squares of the residuals $\left[ \sum_i \ z_i - A_i X_{i \text{ est}}\ ^2 \right]^{1/2}$ , summed over the old and new data. It also includes the a priori term

$\|R_0 X_{est} - z_0\|^2$ . Because RSOS cannot be used if data,  $z$ , is not included we use RSOS.LT.0 to indicate when data is not included.

NSTART

First column of the input A matrix that has a nonzero entry. In certain problems, especially those involving the inclusion of a priori statistics, it is known that the first NSTRT-1 columns of A all have zero entries. This knowledge can be used to reduce computation. If nothing is known about A, then NSTRT.LE.1 gives a default value of 1, i.e. it is assumed that A may have nonzero entries in the very first column.

#### Remarks and Restrictions

It is trivial to arrange the code so that R output need not overwrite the input R. This was not done because, in the author's opinion, there are too few times when one desires to have  $ROUT \neq RIN$ .

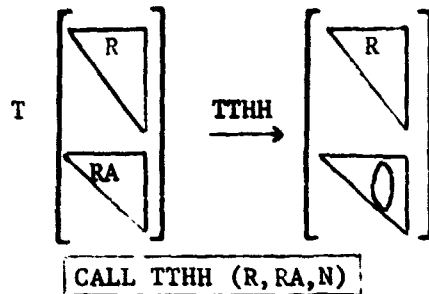
#### Functional Description

Assume for simplicity that NSTRT=1. Then at step  $j$ ,  $j=1, \dots, N$  (or  $N+1$  if data is present) the algorithm implicitly determines an elementary Householder orthogonal transformation which updates row  $j$  of R and all the columns of A to the right of the  $j$ th. At the completion of this step column  $j$  of A is in theory zero, but it is not explicitly set to zero. The orthogonalization process is discussed at length in the books by Lawson and Hanson - Solving Least Squares Problems and Bierman - Factorization Methods for Discrete Sequential Estimation.

22. TTHH (Two triangular matrix Householder reduction)

Purpose

To combine two vector stored upper triangular matrices, R and RA by applying Householder orthogonal transformations. The result overwrites R.



Argument Definitions

$R(N*(N+1)/2)$	Input vector stored upper triangular matrix, which also houses the result
$RA(N*(N+1)/2)$	Second input vector stored upper triangular matrix. This matrix is destroyed by the computation.
N	Matrix dimension N less than zero is used to indicate that R and RA have right sides ( $ N +1$ columns) and have dimension $ N *( N +3)/2$ .

Remarks and Restrictions

RA is theoretically zero on output, but is not set to zero.

23. TWOMAT (Triangular matrix print)

Purpose

To display a vector upper triangular matrix in a two dimensional triangular format. Precision output corresponds to a 7 column 8 digit, double precision format. Compact output corresponds to a 12 column, 5 digit single precision format.

```
CALL TWOMAT(A,N,LEN,CAR,TEXT,NCHAR,NAMES)
```

Argument Definitions

A(N*N+1)/2)	Vector stored upper triangular matrix (DP)
N	Dimension of A
LEN	Column format (7 or 12 columns). When LEN is different from 7 or 12 the print defaults to 12 columns.
CAR(N)	Parameter names (alphanumeric) associated with A. When NAMES is false, CAR is not used.
TEXT(NCHAR)	An array of field data characters to be printed as a title preceding the matrix
NCHAR	Number of characters (including spaces) that are to be printed in text( ) ABS(NCHAR).LE.114. If NCHAR is negative there is no page eject before printing. NCHAR positive results in a page eject so that the print starts on a fresh page.
NAMES	A logical flag. If true then the names of the parameters are used as labels for the rows and columns. If false the output labels default to numerical values.

Remarks and Restrictions

Using NCHAR nonnegative, and starting the print at the top of a new page makes it easier to locate the printed result and is

especially recommended when dealing with large dimensioned arrays. Page economy can, however, be achieved using the NCHAR negative option. In this case the print begins on the next line. The alphanumerics in this routine make it machine dependent; it is arranged for implementation on a UNIVAC 1108.

24. TZERO (Triangular matrix zero)

Purpose

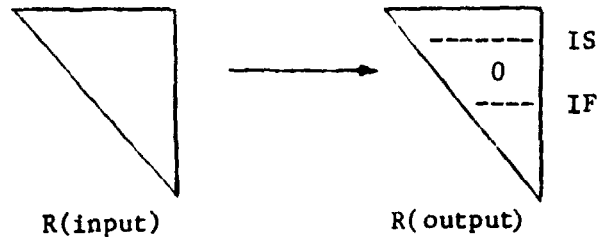
To zero out rows IS(Istart) to IF(Ifinal) of the vector stored upper triangular matrix R.

CALL TZERO(R,N,IS,IF)

Argument Definition

R(N*(N+1)/2)	Input vector stored upper triangular matrix
N	Row dimension of vector stored matrix
IS	First row of R that is to be set to zero
IF	Last row of R that is to be set to zero

Functional Description



25. UDCOL (U-D covariance factor colored noise time update)

Purpose

To time update the U-D covariance factors so as to include the effects of colored noise variables.

CALL UDCOL(U,N,KS,NCOLOR,V,EM,Q)

Argument Definitions

U(N*(N+1)/2)	Input vector stored U-D covariance factors. The updated result resides here on output.
N	Filter matrix dimension. If the last column of U houses the filter estimates, then N = number filter variables + 1.
KS	Location of the first colored noise variable (KS.GE.1.AND.KS.LE.N)
NCOLOR	The number of colored noise variables contiguous to the first, including the first. (NCOLOR.GE.1)
V(KS-1+NCOLOR)	Work vector ((KS-1+NCOLOR).LE.N)
EM(NCOLOR)	Input vector of colored noise mapping terms (unaltered by program)
Q(NCOLOR)	Input vector of process noise variances (unaltered by program)

Remarks and Restrictions

When estimates are involved they are appended as an additional column to the U-D matrix. When the subroutine is applied to the augmented matrix the estimates are correctly updated. When the colored noise terms are not contiguously located one can fill in the gaps with unit EM terms and corresponding zero Q elements. It is preferable, however, to apply the subroutine repeatedly to the individual contiguous groups.



### Functional Description

The model equation corresponding to the time update of this subroutine is

$$\begin{bmatrix} x \\ p \\ y \end{bmatrix}_{j+1} = \begin{bmatrix} I & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x \\ p \\ y \end{bmatrix}_j + \begin{bmatrix} 0 \\ I \\ 0 \end{bmatrix} w_j$$

where  $M$  is diagonal, with  $NP$  terms, and  $w_j \in N(0, Q)$  where  $Q$  is diagonal with  $NP$  terms. The output U-D array associated with this time update equation satisfies

$$UDU^T(\text{output}) = \phi UDU^T \phi^T + BQB^T$$

where  $\phi$  and  $B$  are as above. The algorithm for obtaining U-D (output) is the Bierman-Thornton one-component-at-a-time update described in Bierman - Factorization Methods for Discrete Sequential Estimation", Academic Press (1977), pp 147-148.

26. UDMEAS (U-D measurement update)

Purpose

Kalman filter measurement updating using Bierman's U-D measurement update algorithm, c. 1975 CONF. DEC. CONTROL paper. A scalar measurement  $z = A^T x + v$  is processed, the covariance U-D factors and estimate (when included) are updated, and the Kalman gain and innovations variance are computed.

CALL UDMEAS(U,N,R,A,F,G,ALPHA)

Argument Definitions

INPUTS

U(N\*(N+1)/2) Upper triangular vector stored input matrix. D elements are stored on the diagonal. The U vector corresponds to an a priori covariance. If state estimates are involved the last column of U contains X. In this case Dim U = (N+1)\*(N+2)/2 and on output (U(N+1)\*(N+2)/2 = z-A\*\*T\*X(a priori est)).

N Dimension of state vector, N.GE.2

R Measurement variance

A(N) Vector of Measurement coefficients; if data then A(N+1) = z

F(N) Input work vector. To economize on storage F can overwrite A

ALPHA If ALPHA.LT.zero no estimates are computed (and X and z need not be included).

OUTPUTS

U Updated vector stored U-D factors. When ALPHA (input) is nonnegative the (N+1)st column contains the updated estimate and the predicted residual.

ALPHA Innovations variance of the measurement residual.

F Contains U\*\*T\*A(input) and when ALPHA(input) is nonnegative F(N+1) = (z-A\*\*T\*X(a priori est))/ALPHA.

$G(N)$  Vector of unweighted Kalman gains,  
 $K = G/ALPHA$

Remarks and Restrictions

One can use this algorithm with  $R$  negative to delete a previously processed data point. One should, however, note that data deletion is numerically unstable and sometimes introduces numerical errors.

The algorithm holds for  $R = 0$  (a perfect measurement) and the code has been arranged to include this case. Such situations arise when there are linear constraints and in the generation of certain error "budgets".

Functional Description

The algorithm updates the columns of the U-D matrix, from left to right, using Bierman's algorithm, see Bierman's "Factorization Methods for Discrete Sequential Estimation," Academic Press (1977) pp 76-81 and 100-101.

27. UD2COV (U-D factor to covariance)

Purpose

To obtain a covariance from its U-D factorization. Both matrices are vector stored and the output covariance can overwrite the input U-D array. U-D and P are related via  $P = UDU^T$ .

CALL UD2COV(UIN,POUT,N)

Argument Definitions

UIN(N*(N+1)/2)	Input vector stored U-D factors, with D entries stored on the diagonal.
POUT(N*(N+1)/2)	Output vector stored covariance matrix (POUT = UIN is permitted).
N	Dimension of the matrices involved (N.GE.2)

28. UD2SIG (U-D factors to sigmas)

Purpose

To compute variances from the U-D factors of a matrix.

```
CALL UD2SIG(U,N,SIG,TEXT,NCT)
```

Argument Definitions

U(N*(N+1)/2)	Input vector stored array containing the U-D factors. The D (diagonal) elements are stored on the diagonal of U.
N	Dimension of the U matrix (N.GE.2)
SIG(N)	Output vector of standard deviations
TEXT ( )	Output label of field data characters, which precedes the printed vector of standard deviations.
NCT	Number of characters of text, 0.LE.NCT.LE.126. If NCT = 0, no sigmas are printed, i.e. nothing is printed.

Remarks and Restrictions

The user is cautioned that the text related portion of this subroutine may not be compatible with other computers. The changes that may be involved are, however, very modest.

Functional Description

If U and D are represented as doubly subscripted matrices then

$$SIG(J) = \left( D(J,J) + \sum_{K=J+1}^N D(K,K) [U(J,K)]^2 \right)^{1/2}$$

If NCT.GT.0 a title is printed, followed by the sigmas.

29. UTINV (Upper triangular matrix inverse)

Purpose

To invert an upper triangular vector stored matrix and store the result in vector form. The algorithm is so arranged that the result can overwrite the input.

CALL UTINV(RIN,N,ROUT)

Argument Definitions

RIN(N*(N+1)/2)	Input vector stored upper triangular matrix
N	Matrix dimension
ROUT(N*(N+1)/2)	Output vector stored upper triangular matrix inverse (ROUT = RIN is permitted)

Remarks and Restrictions

Ill conditioning is not tested, but for nonsingular systems the result is as accurate as is the full rank Euclidean scaled singular value decomposition inverse. Singularity occurs if a diagonal is zero. The subroutine terminates when it reaches a zero diagonal. The columns to the left of the zero diagonal are, however, inverted and the result stored in ROUT.

This routine can also be used to produce the solution to  $RX = Z$ . Place Z in column N+1 (viz.  $RIN(N*(N+1)/2+1) = Z(1)$ , etc.), define  $RIN((N+1)(N+2)/2) = -1$  and call the subroutine using N+1 instead of N. On return the first N entries of column N+1 contain the solution (e.g.  $ROUT(N*(N+1)/2+1) = X(1)$ , etc.). When only the estimate is needed, then it is more efficient to use the code described in section to II.8 to obtain X, directly.

Because matrix inversion is numerically sensitive we recommend using this subroutine only in double precision.

Functional Description

The matrix inversion is accomplished using the standard back substitution method for inverting triangular matrices, cf. the book references by Lawson and Hanson, [1] or Bierman [3].

30. UTIROW (Upper triangular inverse, inverting only the upper rows)

Purpose

To compute the inverse of a vector stored upper triangular matrix, when the lower right corner triangular inverse is given.

```
CALL UTIROW(RIN,N,ROUT,NRY)
```

Argument Definitions

RIN(N*(N+1)/2)	Input vector stored upper triangular matrix. Only the first N - NRY rows are altered by the algorithm.
N	Matrix dimension.
ROUT(N*(N+1)/2)	Output vector stored upper triangular matrix inverse. On input the lower NRY dimensional right corner contains the given (known) inverse. This lower right corner matrix is left unchanged. (ROUT = RIN is permitted.)
NRY	Number of rows, starting at the bottom, that are assumed already inverted.

Remarks and Restrictions

The purpose of this subroutine is to complete the computation of an upper triangular matrix inverse, given that the lower right corner has already been inverted. Part of the input, the rows to be inverted, are inserted via the matrix RIN. The portion of the matrix that has already been inverted is entered via the matrix ROUT. It may seem odd that part of the input matrix is put into RIN and part into ROUT. The reasoning behind this decision is that RIN represents the input matrix to be inverted (it just happens that we do not make use of the lower right triangular entries); ROUT represents the inversion result, and therefore that portion of the inversion that is given should be entered in this array.



Ill conditioning is not tested, but for nonsingular systems the result is accurate. Singularity halts the algorithm if any of the first N-NRY diagonal elements is zero. If the first zero encountered moving up the diagonal (starting at N-NRY) is at diagonal j then the rows below this element will be correctly represented in ROUT.

To generate estimates do the following: put N+1 into the matrix dimension argument; in the first N-NRY rows of the last column of RIN put the right hand side elements of the equation  $R_x x + R_{xy} y = z_x$  (i.e.,  $R_x$ ,  $R_{xy}$ , and  $z_x$  make up the first N-NRY rows of RIN); in the next NRY entries of ROUT, beginning in the (N-NRY+1)st element, put  $y_{est}$  (i.e.,  $R_y^{-1}$  and  $y_{est}$  make up rows N-NRY+1, ..., N of ROUT); and  $ROUT((N+1)(N+2)/2) = -1$ . On output, the last column of ROUT will contain  $x_{est}$ ,  $y_{est}$  and -1.

When NRY = 0 this algorithm is equivalent to subroutine UTINV.

#### Functional Description

The matrix inversion is accomplished using the standard back substitution method. The computations are arranged row-wise, starting at the bottom (from row N-NRY, since it is assumed that the last NRY rows have already been inverted).

31. WGS (Weighted Gram-Schmidt matrix triangularization)

Purpose

To compute a vector stored U-D array from an input rectangular matrix W, and a diagonal matrix  $D_w$  so that  $W D_w W^T = UDU^T$ .

CALL WGS(W, IMAXW, IW, JW, DW, U, V)

Argument Definitions

W(IW, JW)	Input rectangular matrix, destroyed by the computations
IMAXW	Row dimension of input W matrix, IMAXW.GE.IW
IW	Number of rows of W matrix, dimension of U
JW	Number of columns of W matrix
DW(JW)	Diagonal input matrix; the entries are assumed to be nonnegative. This vector is unaltered by the computations
U(IW*(IW+1)/2)	Vector stored output U-D array
V(JW)	Work vector in the computation

Remarks and Restrictions

The algorithm is not numerically stable when negative DW weights are used; negative weights are, however, allowed. If JW is less than IW (more rows than columns), the output U-D array is singular; with IW-JW zero diagonal entries in the output U array.

Functional Description

A  $D_w$ -orthogonal set of row vectors,  $\phi_1, \phi_2, \dots, \phi_{IW}$ , are constructed from the input rows of the W matrix, i.e.,  $W = U \phi, \phi D_w \phi^T = D$ . The construction is accomplished using the modified Gram-Schmidt orthogonal construction (see refs. [1] or [3]). This algorithm is reputed to have excellent numerical properties. Note that the  $\phi$  vectors are not of interest in this routine, and they are overwritten; The V vector used in the program houses vector IW-j+1 of  $\phi$  at step j of algorithm. The fact that the computed  $\phi$  vectors may not be D orthogonal is of no import in regard to the U and D computed results.

### References

- [1] Lawson, C. L. Hanson, R. J., Solving Least Squares Problems, Prentice Hall, Englewood Cliffs, N. J. (1974).
- [2] JPL FORTRAN V Subprogram Directory, JPL Internal Document 1845-23, Rev. A., Feb. 1, 1975.
- [3] Bierman, G. J., Factorization Methods for Discrete Sequential Estimation, Academic Press, New York (1977).

## V. FORTRAN Subroutine Listings

The subroutines use only FORTRAN IV, and are therefore essentially portable. The one notable exception is subroutine TWOMAT, which prints triangular, vector stored matrices. It employs FORTRAN V FORMAT statements and six character UNIVAC alphanumeric wordlength, and thus is UNIVAC dependent. Subroutine UD2SIG also involves text, and it too is therefore to some extent machine dependent. Comment statements appear occasionally to the right of the FORTRAN code, and are preceded by a "@" symbol. The subroutine user can, if necessary, transfer or remove such program commentary.

All of the subroutines employ "implicit double precision" statements. They are, however, constructed so as to operate in single precision, and the user has only to omit or comment out the implicit statements. If the subroutines are to be used in double precision on a machine that does not have the implicit FORTRAN option one should explicitly declare all of the non-integer variable names appearing in the programs as double precision variables.

If these subroutines are to be used in production code and computational efficiency is of major concern one should replace the somewhat lengthy subroutine argument lists by introducing COMMON, and including those terms in the COMMON that are redundantly computed with each subroutine call.

	SUBROUTINE A2A1 (A,IA,IR,LA,NAMA,A1,IA1,LA1,NAMA1)	A2A10010
C		A2A10020
C	SUBROUTINE TO REARRANGE THE COLUMNS OF A(IR,LA), IN NAMA ORDER	A2A10030
C	AND PUT THE RESULT IN A1(IR,LA1) IN NAMA1 ORDER. ZERO COLUMNS	A2A10040
C	ARE INSERTED IN A1 CORRESPONDING TO THE NEWLY DEFINED NAMES.	A2A10050
C		A2A10060
C	A(IR,LA) INPUT RECTANGULAR MATRIX	A2A10070
C	IA ROW DIMENSION OF A, IR.LE.IA	A2A10080
C	IR NO. OF ROWS OF A THAT ARE TO BE REARRANGED	A2A10090
C	LA NO. COLUMNS IN A, ALSO THE	A2A10100
C	NO. OF PARAMETER NAMES ASSOCIATED WITH A	A2A10110
C	NAMA(LA) PARAMETER NAMES ASSOCIATED WITH A	A2A10120
C	A1(IR,LA1) OUTPUT RECTANGULAR MATRIX	A2A10130
C	A AND A1 CANNOT SHARE COMMON STORAGE	A2A10140
C	IA1 ROW DIMENSION OF A1, IR.LE.IA1	A2A10150
C	LA1 NO. COLUMNS IN A1, ALSO THE	A2A10160
C	NO. OF PARAMETER NAMES ASSOCIATED WITH A1	A2A10170
C	NAMA1(LA1) INPUT LIST OF PARAMETER NAMES TO BE ASSOCIATED	A2A10180
C	WITH THE OUTPUT MATRIX A1	A2A10190
C		A2A10200
C	COGNIZANT PERSONS: G.J.BIERMAN/M.W.MEAD (JPL, SEPT. 1976)	A2A10210
C		A2A10220
C	DIMENSION A(IA,1), NAMA(1), A1(IA1,1), NAMA1(1)	A2A10230
C	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	A2A10240
C		A2A10250
C	ZERO=0.	A2A10260
C	DO 100 J=1,LA1	A2A10270
C	DO 60 I=1,LA	A2A10280
C	IF (NAMA(I).EQ.NAMA1(J)) GO TO 80	A2A10290
C	60 CONTINUE	A2A10300
C	DO 70 K=1,IR	A2A10310
C	A1(K,J)=ZERO # ZERO COL. CORRES. TO NEW NAME	A2A10320
C	GO TO 100	A2A10330
C	80 DO 90 K=1,IR	A2A10340
C	90 A1(K,J)=A(K,I) # COPY COL. ASSOC. WITH OLD NAME	A2A10350
C	100 CONTINUE	A2A10360
C		A2A10370
C	RETURN	A2A10380
C	END	A2A10390

	SUBROUTINE COMBO (R,L1,NAM1,L2,NAM2,A,IA,LA,NAMA)	COMB0000
C		
C	TO REARRANGE A VECTOR STORED TRIANGULAR MATRIX AND STORE	COMB0010
C	THE RESULT IN MATRIX A. THE DIFFERENCE BETWEEN THIS SUB-	COMB0020
C	ROUTINE AND R2A IS THAT THERE THE NAMELIST FOR A IS INPUT.	COMB0030
C	HERE IT IS DETERMINED BY COMBINING THE LIST FOR R WITH	COMB0040
C	A LIST OF DESIRED NAMES.	COMB0050
C		COMB0060
C	R(L1*(L1+1)/2) INPUT VECTOR STORED UPPER TRIANGULAR MATRIX	COMB0070
C	L1 NO. OF PARAMETERS IN R (AND IN NAM1)	COMB0080
C	NAM1(L1) NAMES ASSOCIATED WITH R	COMB0090
C	L2 NO. OF PARAMETERS IN NAM2	COMB0100
C	NAM2(L2) PARAMETER NAMES THAT ARE TO BE COMBINED WITH R	COMB0110
C	(NAM1 LIST). THESE NAMES MAY OR MAY NOT BE IN	COMB0120
C	NAM1.	COMB0130
C	A(L1,LA) OUTPUT ARRAY CONTAINING THE REARRANGED	COMB0140
C	R MATRIX, L1,LE,IA.	COMB0150
C	IA ROW DIMENSION OF A	COMB0160
C	LA NO. OF PARAMETER NAMES IN NAMA, AND THE	COMB0170
C	COLUMN DIMENSION OF A. LA=L1+L2-NO. NAMES	COMB0180
C	COMMON TO NAM1 AND NAM2. LA IS COMPUTED AND	COMB0190
C	OUTPUT.	COMB0200
C	NAMA(LA) PARAMETER NAMES ASSOCIATED WITH THE OUTPUT A	COMB0210
C	MATRIX. CONSISTS OF NAMES IN NAM1 WHICH ARE	COMB0220
C	NOT IN NAM2 FOLLOWED BY NAM2.	COMB0230
C		COMB0240
C	COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPI, SEPT. 1976)	COMB0250
C		COMB0260
C	IMPLICIT DOUBLE PRECISION (A-H,O-7)	COMB0270
C	DIMENSION R(1), A(IA,1), NAM1(1), NAM2(1), NAMA(1)	COMB0280
C		COMB0290
C	ZERO=0.0	COMB0300
C	K=1	COMB0310
C	DO 100 I=1,L1	COMB0320
C	DO 50 J=1,L2	COMB0330
C	IF (NAM1(I).EQ.NAM2(J)) GO TO 100	COMB0340
C	50 CONTINUE	COMB0350
C	NAMA(K)=NAM1(I)	COMB0360
C	JJ=I*(I-1)/2	COMB0370
C	DO 60 L=1,I	COMB0380
C	60 A(L,K)=R(JJ+L)	COMB0390
C	IF (I.EQ.L1) GO TO 80	COMB0400
C	IP1 = I+1	COMB0410
C	DO 70 L=IP1,L1	COMB0420
C	70 A(L,K) = ZERO	COMB0430
C	80 K=K+1	COMB0440
C	100 CONTINUE	COMB0450
C	NAMES UNIQUE TO NAM1 ARE NOW IN NAMA	COMB0460
C	DO 200 J=1,L2	COMB0470
C	DO 150 I=1,L1	COMB0480
C	IF (NAM2(J).EQ.NAM1(I)) GO TO 170	COMB0490
C	150 CONTINUE	COMB0500
C	NAMA(K)=NAM2(J)	COMB0510
C	DO 160 L=1,L1	COMB0520
C	160 A(L,K)=ZERO	COMB0530

C	NAMES UNIQUE TO NAM2 ARE NOW IN NAMA	COMB0540
	GO TO 190	COMB0550
170	NAMA(K)=NAM2(J)	COMB0560
C	LOCATE DIAGONAL OF PRECEDING COLUMN	COMB0570
	JJ=I*(I-1)/2	COMB0580
	DO 180 L=1,I	COMB0590
180	A(L,K)=R(JJ+L)	COMB0600
	IF (I.EQ.L1) GO TO 190	COMB0610
	IP1=I+1	COMB0620
	DO 185 L=IP1,L1	COMB0630
185	A(L,K)=ZERO	COMB0640
190	K=K+1	COMB0650
200	CONTINUE	COMB0660
	LA=K-1	COMB0670
C	NAMES MUTUAL TO NAM1 AND NAM2 ARE NOW IN NAMA	COMB0680
	RETURN	COMB0690
	END	COMB0700

	SUBROUTINE COVRHO(COV,N,RHO,V)	COVRH010
C		COVRH020
C	TO COMPUTE THE CORRELATION MATRIX RHO, FROM AN INPUT COVARIANCE	COVRH030
C	MATRIX COV. BOTH MATRICES ARE UPPER TRIANGULAR VECTOR STORED.	COVRH040
C	THE CORRELATION MATRIX RESULT CAN OVERWRITE THE INPUT COVARIANCE	COVRH050
C	COV(N*(N+1)/2) INPUT VECTOR STORED POSITIVE SEMI-DEFINITE	COVRH060
C	COVARIANCE MATRIX	COVRH070
C	N NUMBER OF PARAMETERS, N.GE.1	COVRH080
C	RHO(N(N+1)/2) OUTPUT VECTOR STORED CORRELATION MATRIX,	COVRH090
C	RHO(IJ)=COV(IJ)/(SIGMA(I)*SIGMA(J))	COVRH100
C	V(N) WORK VECTOR	COVRH110
C		COVRH120
C	COGNIZANT PERSONS: G.J. RIERMAN/M.W. NEAD (JPL, FEB. 1978)	COVRH130
C		COVRH140
C		COVRH150
C	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	COVRH160
C	DIMENSION COV(1), RHO(1), V(1)	COVRH170
C		COVRH180
C	ONE=1.D0	COVRH190
C	Z=0.D0	COVRH200
C		COVRH210
C	JJ=0	COVRH220
C	DO 10 J=1,N	COVRH230
C	JJ=JJ+J	COVRH240
C	V(J)=Z	COVRH250
C	IF (COV(JJ).GT.Z) V(J)=ONE/ SQRT(COV(JJ))	COVRH260
C		COVRH270
C	**** SOME MACHINES REQUIRE DSORT FOR DOUBLE PRECISION	COVRH280
C		COVRH290
C	10 CONTINUE	COVRH300
C		COVRH310
C	IJ=0	COVRH320
C	DO 20 J=1,N	COVRH330
C	S=V(J)	COVRH340
C	DO 20 I=1,J	COVRH350
C	IJ=IJ+1	COVRH360
C	20 RHO(IJ)=COV(IJ)*S*V(I)	COVRH370
C	RETURN	COVRH380
C	END	COVRH390



SUBROUTINE COV2RI(U,N)

TO CONSTRUCT THE UPPER TRIANGULAR CHOLESKY FACTOR OF A  
POSITIVE SEMI-DEFINITE MATRIX. BOTH THE INPUT COVARIANCE  
AND THE OUTPUT CHOLESKY FACTOR (SQUARE ROOT) ARE VECTOR  
STORED. THE OUTPUT OVERWRITES THE INPUT.  
COVARIANCE(INPUT)=U\*U\*\*T (U IS OUTPUT).

IF THE INPUT COVARIANCE IS SINGULAR THE OUTPUT FACTOR HAS  
ZERO COLUMNS.

U(N\*(N+1)/2) CONTAINS THE INPUT VECTOR STORED COVARIANCE  
MATRIX (ASSUMED POSITIVE DEFINITE) AND ON OUTPUT  
IT CONTAINS THE UPPER TRIANGULAR SQUARE ROOT  
FACTOR.

N DIMENSION OF THE MATRICES INVOLVED

COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, FEB. 1977)

IMPLICIT DOUBLE PRECISION (A-H,O-Z)  
DIMENSION U(1)

ZERO=0.0  
ONE=1.  
JJ=N\*(N+1)/2

DO 5 J=N,2,-1  
IF (U(JJ).LT.ZERO) U(JJ)=ZERO  
U(JJ)=SQRT(U(JJ))  
ALPHA=ZERO  
IF (U(JJ).GT.ZERO) ALPHA=ONE/U(JJ)

KK=0  
JJN=JJ-J @ NEXT DIAGONAL  
JM1=J-1

DO 4 K=1,JM1  
U(JJN+K)=ALPHA\*U(JJN+K) @ JJN+K=(K,J)

S=U(JJN+K)  
DO 3 I=1+K  
U(KK+I)=U(KK+I)-S\*U(JJN+I) @ KK+I=(I,K)

3  
4  
5  
JJ=JJN  
IF (U(1).LT.ZERO) U(1)=ZERO  
U(1)=SQRT(U(1))

RETURN  
END

COV2R010  
COV2R020  
COV2R030  
COV2R040  
COV2R050  
COV2R060  
COV2R070  
COV2R080  
COV2R090  
COV2R100  
COV2R110  
COV2R120  
COV2R130  
COV2R140  
COV2R150  
COV2R160  
COV2R170  
COV2R180  
COV2R190  
COV2R200  
COV2R210  
COV2R220  
COV2R230  
COV2R240  
COV2R250  
COV2R260  
COV2R270  
COV2R280  
COV2R290  
COV2R300  
COV2R310  
COV2R320  
COV2R330  
COV2R340  
COV2R350  
COV2R360  
COV2R370  
COV2R380  
COV2R390  
COV2R400  
COV2R410  
COV2R420  
COV2R430  
COV2R440  
COV2R450  
COV2R460  
COV2R470

```

C          SUBROUTINE COV2UD (U,N)
C
C          TO OBTAIN THE U-D FACTORS OF A POSITIVE SEMI-DEFINITE MATRIX.
C          THE INPUT VECTOR STORED MATRIX IS OVERWRITTEN BY THE OUTPUT
C          U-D FACTORS WHICH ARE ALSO VECTOR STORED.
C
C          U(N*(N+1)/2) CONTAINS INPUT VECTOR STORED COVARIANCE MATRIX.
C          ON OUTPUT IT CONTAINS THE VECTOR STORED U-D
C          COVARIANCE FACTORS.
C          N          MATRIX DIMENSION, N.GE.2
C
C          SINGULAR INPUT COVARIANCES RESULT IN OUTPUT MATRICES WITH ZERO
C          COLUMNS
C
C          COGNIZANT PERSONS: G.J.RIERMAN/R.A.JACOBSON (JPL, FEB. 1977)
C
C          IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C          DIMENSION U(1)
C
C          Z=0.D0
C          ONE=1.D0
C          NONE=1
C
C          JJ=N*(N+1)/2
C          NP2=N+2
C          DO 50 L=2,N
C             J=NP2-L
C             ALPHA=Z
C             IF (U(JJ).GE.Z) GO TO 10
C             WRITE (6,100) J,U(JJ)
C             U(JJ)=Z
10          IF (U(JJ).GT.Z) ALPHA=ONE/U(JJ)
C             JJ=JJ-J
C             KK=0
C             KJ=JJ
C             JM1=J-1
C             DO 40 K=1,JM1
C                KJ=KJ+1
C                BETA=U(KJ)
C                U(KJ)=ALPHA*U(KJ)
C                IJ=JJ
C                IK=KK
C                DO 30 I=1,K
C                   IK=IK+1
C                   IJ=IJ+1
30                 U(IK)=U(IK)-BETA*U(IJ)
40                 KK=KK+K
50          CONTINUE
C          IF (U(1).GE.Z) GO TO 60
C          WRITE (6,100) NONE, U(1)
C          U(1)=Z
60          RETURN
C
COV2U010
COV2U020
COV2U030
COV2U040
COV2U050
COV2U060
COV2U070
COV2U080
COV2U090
COV2U100
COV2U110
COV2U120
COV2U130
COV2U140
COV2U150
COV2U160
COV2U170
COV2U180
COV2U190
COV2U200
COV2U210
COV2U220
COV2U230
COV2U240
COV2U250
COV2U260
COV2U270
COV2U280
COV2U290
COV2U300
COV2U310
COV2U320
COV2U330
COV2U340
COV2U350
COV2U360
COV2U370
COV2U380
COV2U390
COV2U400
COV2U410
COV2U420
COV2U430
COV2U440
COV2U450
COV2U460
COV2U470
COV2U480
COV2U490
COV2U500
COV2U510
COV2U520
COV2U530
COV2U540
COV2U550

```

C-2

100 FORMAT (1H0,20X,' AT STEP',I4,' DIAGONAL ENTRY =',F12.4)  
END

COV21560  
COV21570

```

SUBROUTINE C2C (C,IC,L1,NAM1,L2,NAM2)
C
C      SUBROUTINE TO REARRANGE THE ROWS AND COLUMNS OF MATRIX
C      C(L1,L1) IN NAM1 ORDER AND PUT THE RESULT IN
C      C(L2,L2) IN NAM2 ORDER. ZERO COLUMNS AND ROWS ARE
C      ASSOCIATED WITH OUTPUT DEFINED NAMES THAT ARE NOT CONTAINED
C      IN NAM1.
C
C      C(L1,L1)  INPUT MATRIX
C      IC       ROW DIMENSION OF C, IC.GE.L=MAX(L1,L2)
C      L1      NO. OF PARAMETER NAMES ASSOCIATED WITH THE INPUT C
C      NAM1(L)  PARAMETER NAMES ASSOCIATED WITH C ON INPUT. (ONLY
C              THE FIRST L1 ENTRIES APPLY TO THE INPUT C)
C      L2      NO. OF PARAMETER NAMES ASSOCIATED WITH THE OUTPUT C
C      NAM2(L2) PARAMETER NAMES ASSOCIATED WITH THE OUTPUT C
C
C      COGNIZANT PERSONS:  G.J.RIFERMAN/M.W.NEAD (JPL, SEPT. 1976)
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION C(IC,1), NAM1(1), NAM2(1)
C
C      ZERO=0.
C      L=MAX(L1,L2)
C      IF (L.LE.L1) GO TO 5
C      NM=L1+1
C      DO 1 K=NM,L
C 1    NAM1(K)=ZERO      @ ZERO REMAINING NAM1 LOCNS
C 5    DO 90 J=1,L2
C      DO 10 I=1,L
C        IF (NAM1(I).EQ.NAM2(J)) GO TO 30
C 10   CONTINUE
C      GO TO 90
C 30   IF (I.EQ.J) GO TO 90
C      DO 40 K=1,L
C        H=C(K,J)      @ INTERCHANGE COLUMNS I AND J
C        C(K,J)=C(K,I)
C        C(K,I)=H
C 40   DO 80 K=1,L
C        H=C(J,K)      @ INTERCHANGE ROWS I AND J
C        C(J,K)=C(I,K)
C        C(I,K)=H
C 80   NM=NAM1(I)      @ INTERCHANGE LABELS I AND J
C        NAM1(I)=NAM1(J)
C        NAM1(J)=NM
C 90   CONTINUE
C
C      FIND NAM2 NAMES NOT IN NAM1 AND SET CORRESPONDING ROWS AND
C      COLUMNS TO ZERO
C
C      DO 120 J=1,L2
C      DO 100 I=1,L
C        IF (NAM1(I).EQ.NAM2(J)) GO TO 120
C 100  CONTINUE
C      DO 110 K=1,L2
C        C(J,K)=ZERO

```

110 C(K+J)=ZERO  
120 CONTINUE

C

RETURN  
END

C2C00550  
C2C00560  
C2C00570  
C2C00580  
C2C00590

```

SUBROUTINE HHPOST(S,W,MROW,NPOW,NCOL,V)
C
C      TRIANGULARIZES RECTANGULAR W BY POST MULTIPLYING IT BY AN
C      ORTHOGONAL TRANSFORMATION T. THE RESULT IS IN S
C
C      S(NROW*(NROW+1)/2) OUTPUT UPPER TRIANGULAR VECTOR STORED SORT
C      COVARIANCE MATRIX
C      W(NROW,NCOL)      INPUT RECTANGULAR SORT COVARIANCE MATRIX
C      (W IS DESTROYED BY COMPUTATIONS)
C      MROW              ROW DIMENSION OF W
C      NROW              NUMBER OF ROWS OF W TO BE TRIANGULARIZED
C      AND THE DIMENSION OF S (NROW.GT.1)
C      NCOL              NUMBER OF COLUMNS OF W (NCOL.GE.MROW)
C      V(NCOL)          WORK VECTOR
C
C      COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD      (JPL, NOV.1977)
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DOUBLE PRECISION SUM,BETA
C      DIMENSION S(1),W(MROW,NCOL),V(NCOL)
C
C      ZERO=0.00
C      ONE=1.00
C
C      JCOL=NCOL
C      NSYM=NROW*(NROW+1)/2
C      JC=NROW+2
C      DO 150 L=2,NROW
C          IROW=JC-L
C          SUM=ZERO
C          DO 100 K=1,JCOL
C              V(K)=W(IROW,K)
100          SUM=SUM+V(K)**2
C          SUM=DSQRT(SUM)
C          IF (V(JCOL).GT.ZERO) SUM=-SUM      @ DIAGONAL ENTRY (JCOL,JCOL)
C
C          S(NSYM)=SUM
C          NSYM=NSYM-IROW
C          V(JCOL)=V(JCOL)-SUM
C          IF (SUM.NE.ZERO) BETA=-ONE/(SUM*V(JCOL))
C          T(ORTHOG. TRANS.)=I-BETA*V*V**T
C          IROWM1=IROW-1
C          JCOLM1=JCOL-1
C          DO 140 I=1,IROWM1
C              SUM=ZERO
C              DO 110 K=1,JCOL
110          SUM=SUM+V(K)*W(I,K)
C              SUM=BETA*SUM
C              DO 120 K=1,JCOLM1
120          W(I,K)=W(I,K)-SUM*V(K)
140          S(NSYM+I)=W(I,IROW)-SUM*V(IPOW)
150          JCOL=JCOLM1
C
C      JC=NCOL-NROW+1
C      SUM=ZERO

```

```

HHPOS010
HHPOS020
HHPOS030
HHPOS040
HHPOS050
HHPOS060
HHPOS070
HHPOS080
HHPOS090
HHPOS100
HHPOS110
HHPOS120
HHPOS130
HHPOS140
HHPOS150
HHPOS160
HHPOS170
HHPOS180
HHPOS190
HHPOS200
HHPOS210
HHPOS220
HHPOS230
HHPOS240
HHPOS250
HHPOS260
HHPOS270
HHPOS280
HHPOS290
HHPOS300
HHPOS310
HHPOS320
HHPOS330
HHPOS340
HHPOS350
HHPOS360
HHPOS370
HHPOS380
HHPOS390
HHPOS400
HHPOS410
HHPOS420
HHPOS430
HHPOS440
HHPOS450
HHPOS460
HHPOS470
HHPOS480
HHPOS490
HHPOS500
HHPOS510
HHPOS520
HHPOS530
HHPOS540
HHPOS550

```

DO 160 J=1+JC  
160 SUM=SUM+W(I,J)\*\*2  
S(I)=DSORT(SUM)  
C  
RETURN  
END

HHP05560  
HHP05570  
HHP05580  
HHP05590  
HHP05600  
HHP05610

```

C
C      SUBROUTINE INF2R (R,N)
C
C      TO CHOLFSKY FACTOR AN INFORMATION MATRIX
C
C      COMPUTES A LOWER TRIANGULAR VECTOR STORED CHOLFSKY FACTORIZATION
C      OF A POSITIVE SEMI-DEFINITE MATRIX. R=R(**T)R, R UPPER TRIANGULAR.
C      BOTH MATRICES ARE VECTOR STORED AND THE RESULT OVERWRITES
C      THE INPUT
C
C      R(N*(N+1)/2) ON INPUT THIS IS A POSITIVE SEMI-DEFINITE
C      (INFORMATION) MATRIX, AND ON OUTPUT IT IS THE
C      TRANSPOSED LOWER TRIANGULAR CHOLFSKY FACTOR. IF THE
C      INPUT MATRIX IS SINGULAR THE OUTPUT MATRIX WILL
C      HAVE ZERO DIAGONAL ENTRIES
C
C      N
C      DIMENSION OF MATRICES INVOLVED, N.GE.2
C
C      COGNIZANT PERSON: G.J.BIERMAN/M.W.NEAD      (JPL,FEB,1977)
C
C      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)
C
C      DIMENSION R(1)
C
C      Z=0.00
C      ONE=1.00
C      JJ=0
C      NN=N*(N+1)/2
C      NMI=N-1
C      DO 10 J=1,NMI
C          JJ=JJ+J
C          IF (R(JJ).GE.?) GO TO 5
C          WRITE (6,20) J,R(JJ)
C          R(JJ)=Z
C      5  R(JJ)=SQRT(R(JJ))
C
C      **** SOME MACHINES REQUIRE DSORT FOR DOUBLE PRECISION
C
C      ALPHA=Z
C      IF (R(JJ).GT.?) ALPHA=ONE/R(JJ)
C      JK=NN+J
C      JP1=J+1
C      JIS=JK
C      NPJP1=N+JP1
C      DO 10 L=JP1,N
C          K=NPJP1-L
C          JK=JK-K
C          R(JK)=ALPHA*R(JK)
C          RETA=R(JK)
C          KI=NN+K
C          JI=JIS
C          NPK=N+K
C          DO 10 M=K,N
C              I=NPK-M
C              KI=KI-I
C              JI=JI-I

```

```

INF2R010
INF2R020
INF2R030
INF2R040
INF2R050
INF2R060
INF2R070
INF2R080
INF2R090
INF2R100
INF2R110
INF2R120
INF2R130
INF2R140
INF2R150
INF2R160
INF2R170
INF2R180
INF2R190
INF2R200
INF2R210
INF2R220
INF2R230
INF2R240
INF2R250
INF2R260
INF2R270
INF2R280
INF2R290
INF2R300
INF2R310
INF2R320
INF2R330
INF2R340
INF2R350
INF2R360
INF2R370
INF2R380
INF2R390
INF2R400
INF2R410
INF2R420
INF2R430
INF2R440
INF2R450
INF2R460
INF2R470
INF2R480
INF2R490
INF2R500
INF2R510
INF2R520
INF2R530
INF2R540
INF2R550

```



10	R(KI)=R(KI)-R(JI)*BETA	INF2R560
C	IF (R(NN).GE.Z) GO TO 15	INF2R570
	WRITE (6,20) N,R(NN)	INF2R580
	R(NN)=Z	INF2R590
15	R(NN)= SORT(R(NN))	INF2R600
	RETURN	INF2R610
C	20 FORMAT (1H0,20X,' AT STEP',I4,' DIAGONAL ENTRY =',E12.4,	INF2R620
	1 ' , IT IS RESET TO ZERO')	INF2R630
	END	INF2R640
		INF2R650
		INF2R660

	SUBROUTINE PERMIT (A,IA,IR,L1,NAM1,L2,NAM2)	PERMU010
C		PERMU020
C	SUBROUTINE TO REARRANGE PARAMETERS OF A(IR,L1), NAM1 ORDER	PERMU030
C	TO A(IR,L2), NAM2 ORDER. ZERO COLUMNS ARE INSERTED	PERMU040
C	CORRESPONDING TO THE NEWLY DEFINED NAMES.	PERMU050
C		PERMU060
C	A(IR,L) INPUT RECTANGULAR MATRIX, L=MAX(L1,L2)	PERMU070
C	IA ROW DIMENSION OF A, IA.GF.IR	PERMU080
C	IR NUMBER OF ROWS OF A THAT ARE TO BE REARRANGED	PERMU090
C	L1 NUMBER OF PARAMETER NAMES ASSOCIATED WITH THE INPUT	PERMU100
C	A MATRIX	PERMU110
C	NAM1(L) PARAMETER NAMES ASSOCIATED WITH A ON INPUT	PERMU120
C	(ONLY THE FIRST L1 ENTRIES APPLY TO THE INPUT A)	PERMU130
C	NAM1 IS DESTROYED BY PERMUT	PERMU140
C	L2 NUMBER OF PARAMETER NAMES ASSOCIATED WITH THE OUTPUT	PERMU150
C	A MATRIX	PERMU160
C	NAM2 PARAMETER NAMES ASSOCIATED WITH THE OUTPUT A	PERMU170
C		PERMU180
C	COGNIZANT PERSONS: G.J.RIFERMAN/M.W.NEAD (JPL, SEPT. 1976)	PERMU190
C		PERMU200
	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	PERMU210
	DIMENSION A(IA,1), NAM1(1), NAM2(1)	PERMU220
C		PERMU230
	ZERO=0.	PERMU240
	L=MAX(L1,L2)	PERMU250
	IF (L.LE.L1) GO TO 50	PERMU260
	NM=L1+1	PERMU270
	DO 40 K=NM,L	PERMU280
40	NAM1(K)=0 @ ZERO REMAINING NAM1 LOCS	PERMU290
50	DO 100 J=1,L2	PERMU300
	DO 60 I=1,L	PERMU310
	IF (NAM1(I).EQ.NAM2(J)) GO TO 65	PERMU320
60	CONTINUE	PERMU330
	GO TO 100	PERMU340
65	CONTINUE	PERMU350
	IF (I.EQ.J) GO TO 100	PERMU360
	DO 70 K=1,IR @ INTERCHANGE COLS I AND J	PERMU370
	W=A(K,J)	PERMU380
	A(K,J)=A(K,I)	PERMU390
70	A(K,I)=W	PERMU400
	NM=NAM1(I) @ INTERCHANGE I AND J COL. LABELS	PERMU410
	NAM1(I)=NAM1(J)	PERMU420
	NAM1(J)=NM	PERMU430
100	CONTINUE	PERMU440
C		PERMU450
	REPEAT TO FILL NEW COLS	PERMU460
	DO 200 J=1,L2	PERMU470
	DO 160 I=1,L	PERMU480
	IF (NAM1(I).EQ.NAM2(J)) GO TO 200	PERMU490
160	CONTINUE	PERMU500
	DO 170 K=1,IR	PERMU510
170	A(K,J)=ZERO	PERMU520
200	CONTINUE	PERMU530
C		PERMU540
	RETURN	PERMU550
	END	

	SUBROUTINE PHIU(PHI,MAXPHI,IRPHI,ICPHI,U,N,PHIU,MPHIU)	PHIU0010
C		PHIU0020
C	THIS SUBROUTINE COMPUTES W=PHI*U WHERE PHI IS A RECTANGULAR MATRIX	PHIU0030
C	WITH IMPLICITLY DEFINED COLUMNS OF TRAILING ZEROS AND U IS A	PHIU0040
C	VECTOR STORED UPPER TRIANGULAR MATRIX	PHIU0050
C		PHIU0060
C	PHI(IRPHI,ICPHI) INPUT RECTANGULAR MATRIX, IRPHI,LE,MAXPHI	PHIU0070
C	MAXPHI ROW DIMENSION OF PHI	PHIU0080
C	IRPHI NO. ROWS OF PHI	PHIU0090
C	ICPHI NO. COLS OF PHI	PHIU0100
C	U(N*(N+1)/2) UPPER TRIANGULAR VECTOR STORED MATRIX	PHIU0110
C	N DIMENSION OF U MATRIX (ICPHI,LE,N)	PHIU0120
C	PHIU(IRPHI,N) OUTPUT, RESULT OF PHI*U, PHIU CAN	PHIU0130
C	OVERWRITE PHI	PHIU0140
C	MPHIU ROW DIMENSION OF PHIU	PHIU0150
C		PHIU0160
C	COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, FFR.1978)	PHIU0170
C		PHIU0180
	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	PHIU0190
	DIMENSION PHI(MAXPHI,1),U(1),PHIU(MPHIU,1)	PHIU0200
	DOUBLE PRECISION SUM	PHIU0210
C		PHIU0220
	DO 10 I=1,IRPHI	PHIU0230
10	PHIU(I,1)=PHI(I,1)	PHIU0240
C		PHIU0250
	NP2=N+2	PHIU0260
	KJS=N*(N+1)/2	PHIU0270
	DO 40 L=2,N	PHIU0280
	J=NP2-L	PHIU0290
	KJS=KJS-J	PHIU0300
	JM1=J-1	PHIU0310
	DO 30 I=1,IRPHI	PHIU0320
	SUM=PHI(I,J)	PHIU0330
	IF (J,LE,ICPHI) GO TO 15	PHIU0340
	SUM=0.00	PHIU0350
	JM1=ICPHI	PHIU0360
15	DO 20 K=1,JM1	PHIU0370
20	SUM=SUM+PHI(I,K)*U(KJS+K)	PHIU0380
30	PHIU(I,J)=SUM	PHIU0390
40	CONTINUE	PHIU0400
C		PHIU0410
	RETURN	PHIU0420
	END	PHIU0430

```

SUBROUTINE RA (R,N,A,MAXA,IA,JA,RA,MAXRA,NRA)
C
C      TO COMPUTE RA=R*A
C
C      WHERE R IS UPPER TRIANGULAR VECTOR SUBSCRIPTED AND OF DIMENSION N,
C      A HAS JA COLUMNS AND IA ROWS. IF IA.LT.JA THEN THE BOTTOM JA-IA
C      ROWS OF A ARE ASSUMED TO BE IMPLICITLY DEFINED AS THE
C      BOTTOM JA-IA ROWS OF THE JA DIMENSION IDENTITY MATRIX.
C      ONLY NRA ROWS OF THE PRODUCT R*A ARE COMPUTED.
C
C      R(N*(N+1)/2) UPPER TRIANGULAR VECTOR STORED INPUT MATRIX
C      N DIMENSION OF R
C      A(IA,JA) INPUT RECTANGULAR MATRIX
C      MAXA ROW DIMENSION OF A
C      IA NUMBER OF ROWS IN THE A MATRIX (IA.LE.MAXA)
C      JA NUMBER OF COLUMNS IN THE A MATRIX
C      RA(NRA,N) OUTPUT RESULTING RECTANGULAR MATRIX,
C      RA=A IS ALLOWED
C      MAXRA ROW DIMENSION OF RA
C      NRA NUMBER OF ROWS OF THE PRODUCT R*A THAT ARE COMPUTED
C      (NRA.LE.MAXRA)
C
C      COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, FEB.1978)
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION R(1),A(MAXA,1),RA(MAXRA,1)
C      DOUBLE PRECISION SUM
C
C      IJ=IA*(IA+1)/2 @ IJ=JJ(IA)
C
C      DO 30 J=1,JA
C      II=0 @ TO BE REMOVED IF JJ(I) IS USED
C      DO 20 I=1,NRA
C      II=II+I @ II=(I,II)=JJ(I)
C      IT IS MORE EFFICIENT TO USE A PRESTORED VECTOR OF DIAGONALS
C      WITH JJ(I)=I*(I+1)/2, AND TO SET II=JJ(I) AND IJ=J*(J)
C
C      SUM=0.D0
C      IF (I.GT.IA) GO TO 15
C      IK=II
C      DO 10 K=I,JA
C      SUM=SUM+R(IK)*A(K,J)
10      IK=IK+K
15      IF (J.GT.IA.AND.I.LE.J) SUM=SUM+R(IJ+I)
C
C      20 RA(I,J)=SUM
C      30 IF (J.GT.IA) IJ=IJ+J @ IJ=JJ(J)
C
C      RETURN
C      END

```

C	SUBROUTINE RANK1 (UIN,UOUT,N,C,V)	RANK1010
C	STABLE U-D FACTOR RANK 1 UPDATE	RANK1020
C		RANK1030
C	(UOUT)*DOUT*(UOUT)**T=(UIN)*DIN*(UIN)**T+C*V*V**T	RANK1040
C		RANK1050
C	UIN(N*(N+1)/2) INPUT VECTOR STORED POSITIVE SEMI-DEFINITE U-D	RANK1060
C	ARRAY, WITH D ELEMENTS STORED ON THE DIAGONAL	RANK1070
C	UOUT(N*(N+1)/2) OUTPUT VECTOR STORED POSITIVE (POSSIBLY) SEMI-	RANK1080
C	DEFINITE U-D RESULT. UOIT=UIN IS PERMITTED	RANK1090
C	N	RANK1100
C	MATRIX DIMENSION, N.GE.2	RANK1110
C	C	RANK1120
C	INPUT SCALAR. SHOULD BE NON-NEGATIVE	RANK1130
C	C IS DESTROYED DURING THE PROCESS	RANK1140
C	V(N)	RANK1150
C	INPUT VECTOR FOR RANK ONE MODIFICATION.	RANK1160
C	V IS DESTROYED DURING THE PROCESS	RANK1170
C		RANK1180
C	COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL,SEPT.1977)	RANK1190
C		RANK1200
C	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	RANK1210
C	DIMENSION UIN(1), UOUT(1), V(1)	RANK1220
C	DOUBLE PRECISION ALPHA, BETA, S, D, EPS, TST	RANK1230
C		RANK1240
C	DATA EPS/0.00/, TST/.0625D0/	RANK1250
C	IN SINGLE PRECISION EPSILON IS MACHINE ACCURACY	RANK1260
C		RANK1270
C	TST=1/16 IS USED FOR RANK1 ALGORITHM SWITCHING	RANK1280
C		RANK1290
C	Z=0.00	RANK1300
C	JJ=N*(N+1)/2	RANK1310
C	IF (C.GT.Z) GO TO 4	RANK1320
C	DO 1 J=1,JJ	RANK1330
C	1 UOUT(J)=UIN(J)	RANK1340
C	RETURN	RANK1350
C		RANK1360
C	4 NP2=N+2	RANK1370
C	DO 70 L=2,N	RANK1380
C	J=NP2-L	RANK1390
C	S=V(J)	RANK1400
C	BETA=C*S	RANK1410
C	D=UIN(JJ)+BETA*S	RANK1420
C	IF (D.GT.EPS) GO TO 30	RANK1430
C	IF (D.GE.Z) GO TO 10	RANK1440
C	5 WRITE (6,100)	RANK1450
C	RETURN	RANK1460
C	10 JJ=JJ-J	RANK1470
C	WRITE (6,110)	RANK1480
C	DO 20 K=1,J	RANK1490
C	20 UOUT(JJ+K)=Z	RANK1500
C	GO TO 70	RANK1510
C	30 BETA=BETA/D	RANK1520
C	ALPHA=UIN(JJ)/D	RANK1530
C	C=ALPHA*C	RANK1540
C	UOUT(JJ)=D	RANK1550
C	JJ=JJ-J	
C	JM1=J-1	

	IF (ALPHA.LT.TST) GO TO 50	RANK1560
	DO 40 I=1,JM1	RANK1570
	V(I)=V(I)-S*UIN(JJ+I)	RANK1580
40	UOUT(JJ+I)=RETA*V(I)+UIN(JJ+I)	RANK1590
	GO TO 70	RANK1600
50	DO 60 I=1,JM1	RANK1610
	D=V(I)-S*UIN(JJ+I)	RANK1620
	UOUT(JJ+I)=ALPHA*UIN(JJ+I)+RETA*V(I)	RANK1630
60	V(I)=D	RANK1640
70	CONTINUE	RANK1650
C		RANK1660
	UOUT(1)=UIN(1)+C*V(1)**2	RANK1670
	RETURN	RANK1680
C		RANK1690
100	FORMAT (1H0,10X,'* * * ERROR RETURN DUE TO A COMPUTED NEGATIVE COM	RANK1700
	PUTED DIAGONAL IN RANK1 * * *')	RANK1710
110	FORMAT (1H0,10X,'* * * NOTE: U-D RESULT IS SINGULAR * * *')	RANK1720
	END	RANK1730

```

SUBROUTINE PCOLRD(S,MAXS,IRS,JCS,NPSTRT,NP,FM,RW,ZW,V,SGSTAR)
C
C      TO ADD IN PROCESS NOISE EFFECTS INTO THE SQUARE ROOT
C      INFORMATION FILTER, AND TO GENERATE WEIGHTING COEFFICIENTS
C      FOR SMOOTHING. IT IS ASSUMED THAT VARIABLES X(NPSTRT),
C      X(NPSTRT+1),...,X(NPSTRT+NP-1) ARE COLORED NOISE AND THAT
C      EACH COMPONENT SATISFIES A MODEL EQUATION OF THE FORM
C      X(SUB)(J+1)=EM*X(SUB)(J)+W(SUB)(J). FOR DETAILS, SEE
C      'FACTORIZATION METHODS FOR DISCRETE SEQUENTIAL ESTIMATION',
C      G.J.BIERMAN, ACADEMIC PRESS (1977)
C      FOR SMOOTHING, REMOVE THE COMMENT STATEMENTS ON THE 3 LINES
C      OF 'SMOOTHING ONLY' CODE. THE SIGNIFICANCE OF THE SMOOTHING
C      MATRIX IS EXPLAINED IN THE FUNCTIONAL DESCRIPTION.
C
C      S(IRS,JCS)  INPUT SQUARE ROOT INFORMATION ARRAY. OUTPUT COLORED
C                  NOISE ARRAY HOUSED HERE TOO. IF THERE IS SMOOTHING,
C                  NR ADDITIONAL ROWS ARE INCLUDED IN S
C      MAXS        ROW DIMENSION OF S. IF THERE ARE SMOOTHING COMPUTA-
C                  TIONS IT IS NECESSARY THAT MAXS.GE.IRS+NP BECAUSE
C                  THE BOTTOM NP ROWS OF S HOUSE THE SMOOTHING
C                  INFORMATION
C      IRS        NUMBER OF ROWS OF S (.LE. NUMBER OF FILTER VARIABLES)
C                  (IRS.GE.2)
C      JCS        NUMBER OF COLUMNS OF S (EQUALS NUMBER OF FILTER
C                  VARIABLES + POSSIBLY A RIGHT SIDE), WHICH CONTAINS
C                  THE DATA EQUATION NORMALIZED ESTIMATE (JCS.GE.1)
C      NPSTRT     LOCATION OF THE FIRST COLORED NOISE VARIABLE
C                  (1.LE.NPSTRT.LE.JCS)
C      NP         NUMBER OF CONTIGUOUS COLORED NOISE VARIABLES (NP.GE.1)
C      EM(NP)     COLORED NOISE MAPPING COEFFICIENTS
C                  (OF EXPONENTIAL FORM, EM=EXP(-DT/TAU))
C      RW(NP)     RECIPROCAL PROCESS NOISE STANDARD DEVIATIONS
C                  (MUST BE POSITIVE)
C      ZW(NP)     ZW=RW+W-ESTIMATE (PROCESS NOISE ESTIMATES ARE
C                  GENERALLY ZERO MEAN). WHEN ZW=0 ONE CAN OMIT THE
C                  RIGHT HAND SIDE COLUMN.
C      V(IRS)     WORK VECTOR
C      SGSTAR(NP) VECTOR OF SMOOTHING COEFFICIENTS. WHEN THE SMOOTHING
C                  CODE IS COMMENTED OUT SGSTAR IS NOT USED.
C
C      COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, FEB.1978)
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION S(MAXS,JCS),EM(NP),RW(NP),ZW(NP), V(IRS),SGSTAR(1)
C      DOUBLE PRECISION ALPHA,SIGMA,BETA,GAMMA
C
C      ZERO=0.00
C      ONE=1.00
C      NPCOL=NPSTRT @ COL NO OF COLORED NOISE TERM TO BE OPERATED ON
C
C      DO 70 JCOLRD=1,NP
C          ALPHA=-RW(JCOLRD)+FM(JCOLRD)
C          SIGMA=ALPHA**2
C          DO 10 K=1,IRS
C              V(K)=S(K,NPCOL) @ FIRST IRS ELEMENTS OF HOUSEHOLDER

```

```

RCOLR010
RCOLR020
RCOLR030
RCOLR040
RCOLR050
RCOLR060
RCOLR070
RCOLR080
RCOLR090
RCOLR100
RCOLR110
RCOLR120
RCOLR130
RCOLR140
RCOLR150
RCOLR160
RCOLR170
RCOLR180
RCOLR190
RCOLR200
RCOLR210
RCOLR220
RCOLR230
RCOLR240
RCOLR250
RCOLR260
RCOLR270
RCOLR280
RCOLR290
RCOLR300
RCOLR310
RCOLR320
RCOLR330
RCOLR340
RCOLR350
RCOLR360
RCOLR370
RCOLR380
RCOLR390
RCOLR400
RCOLR410
RCOLR420
RCOLR430
RCOLR440
RCOLR450
RCOLR460
RCOLR470
RCOLR480
RCOLR490
RCOLR500
RCOLR510
RCOLR520
RCOLR530
RCOLR540
RCOLR550

```

```

C                                     TRANSFORMATION VECTOR
10      SIGMA=SIGMA+V(K)**2
      SIGMA=DSQRT(SIGMA)
      ALPHA=ALPHA-SIGMA      @ LAST ELEMENT OF HOUSEHOLDER
C                                     TRANSFORMATION VECTOR
C * * * * *
C      SGSTAR(JCOLRD)=SIGMA      @ USED FOR SMOOTHING ONLY
C * * * * *
      BETA=ONE/(SIGMA+ALPHA)      @ HOUSEHOLDER=I+BETA*V+V**T
C      HOUSEHOLDER TRANSFORMATION DEFINED, NOW APPLY IT TO S, I.E. 60 LOOP
      DO 60 KOL=1,JCS
      IF (KOL.NE.NPCOL) GO TO 30
      GAMMA= RW(JCOLRD)*ALPHA*BETA
C * * * * *
C      S(IRS+JCOLRD,NPCOL)=RW(JCOLRD)+GAMMA*ALPHA      @ SMOOTHING ONLY
C * * * * *
      DO 20 K=1,IRS
20      S(K,NPCOL)=GAMMA*V(K)
      GO TO 60
30      GAMMA=ZERO
      IF (KOL.EQ.JCS) GAMMA=ZW(JCOLRD)*ALPHA
C
C      IF ZW ALWAYS ZERO, COMMENT OUT THE ABOVE IF TEST
C
      DO 40 K=1,IRS
40      GAMMA=GAMMA+S(K,KOL)*V(K)
      GAMMA= GAMMA*BETA
      DO 50 K=1,IRS
50      S(K,KOL)=S(K,KOL)+GAMMA*V(K)
C * * * * *
C      S(IRS+JCOLRD,KOL)=GAMMA*ALPHA      @ FOR SMOOTHING ONLY
C * * * * *
60      CONTINUE
C * * * * *
C      S(IRS+JCOLRD,JCS)=S(IRS+JCOLRD,JCS)+ZW(JCOLRD)
C      THE ABOVE IS FOR SMOOTHING ONLY
C      IF ZW IS ALWAYS ZERO, COMMENT OUT THE ABOVE STATEMENT
C * * * * *
70      NPCOL=NPCOL+1
C
      RETURN
      END

```

```

RCOLR560
RCOLR570
RCOLR580
RCOLR590
RCOLR600
RCOLR610
RCOLR620
RCOLR630
RCOLR640
RCOLR650
RCOLR660
RCOLR670
RCOLR680
RCOLR690
RCOLR700
RCOLR710
RCOLR720
RCOLR730
RCOLR740
RCOLR750
RCOLR760
RCOLR770
RCOLR780
RCOLR790
RCOLR800
RCOLR810
RCOLR820
RCOLR830
RCOLR840
RCOLR850
RCOLR860
RCOLR870
RCOLR880
RCOLR890
RCOLR900
RCOLR910
RCOLR920
RCOLR930
RCOLR940
RCOLR950
RCOLR960
RCOLR970

```



C	SUBROUTINE RINCON (RIN,N,ROUT,CNR)	RTNC0010
C	TO COMPUTE THE INVERSE OF THE UPPER TRIANGULAR VECTOR STORED	RTNC0020
C	INPUT MATRIX RIN AND STORE THE RESULT IN ROUT. (RIN=ROUT IS	RTNC0030
C	PERMITTED) AND TO COMPUTE A CONDITION NUMBER ESTIMATE.	RTNC0040
C	CNB=FR0B.NORM(R)*FR0B.NORM(R**-1).	RTNC0050
C	THE FROBENIUS NORM IS THE SQUARE ROOT OF THE SUM OF SQUARES	RTNC0060
C	OF THE ELEMENTS. THIS CONDITION NUMBER BOUND IS USED AS	RTNC0070
C	AN UPPER BOUND AND IT ACTS AS A LOWER BOUND ON THE ACTUAL	RTNC0080
C	CONDITION NUMBER OF THE PROBLEM. (SEE THE BOOK 'SOLVING LEAST	RTNC0090
C	SQUARES' BY LAWSON AND HANSON)	RTNC0100
C	IF RIN IS SINGULAR, RINCON COMPUTES THE INVERSE TO THE LEFT OF	RTNC0110
C	THE FIRST ZERO DIAGONAL. A MESSAGE IS PRINTED AND THE CONDITION	RTNC0120
C	NUMBER BOUND COMPUTATION IS ABORTED.	RTNC0130
C	RIN(N*(N+1)/2) INPUT VECTOR STORED UPPER TRIANGULAR MATRIX	RTNC0140
C	N DIMENSION OF R MATRICES, N.GE.2	RTNC0150
C	ROUT(N*(N+1)/2) OUTPUT VECTOR STORED UPPER TRIANGULAR MATRIX	RTNC0160
C	INVERSE (RIN=ROUT IS PERMITTED)	RTNC0170
C	CNB CONDITION NUMBER BOUND. IF C IS THE CONDITION	RTNC0180
C	NUMBER OF RIN, THEN CNB/N.LE.C.LE.CNR	RTNC0190
C	COGNIZANT PERSONS: G.J.BIFERMAN/M.W.HEAD (JPL,FEB.1978)	RTNC0200
C	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	RTNC0210
C	DOUBLE PRECISION RNM,DINV,SUM,RNMOUT	RTNC0220
C	DIMENSION RIN(1),ROUT(1)	RTNC0230
C	Z=0.00	RTNC0240
C	ONE=1.00	RTNC0250
C	NTOT=N*(N+1)/2	RTNC0260
C	RNM=Z	RTNC0270
C	DO 10 J=1,NTOT	RTNC0280
C	10 RNM=RNM+RIN(J)**2	RTNC0290
C	REPLACE CALL UTINV (RIN,N,ROUT) BY UTINV CODE	RTNC0300
C	IF (RIN(1).NE.Z) GO TO 20	RTNC0310
C	J=1	RTNC0320
C	WRITE (6,100) J,J	RTNC0330
C	RETURN	RTNC0340
C	20 ROUT(1)=ONE/RIN(1)	RTNC0350
C	JJ=1	RTNC0360
C	DO 50 J=2,N	RTNC0370
C	JJOLD=JJ	RTNC0380
C	JJ=JJ+J	RTNC0390
C	IF (RIN(JJ).NE.Z) GO TO 30	RTNC0400
C	WRITE (6,100) J,J	RTNC0410
C	RETURN	RTNC0420
C		RTNC0430
C		RTNC0440
C		RTNC0450
C		RTNC0460
C		RTNC0470
C		RTNC0480
C		RTNC0490
C		RTNC0500
C		RTNC0510
C		RTNC0520
C		RTNC0530
C		RTNC0540
C		RTNC0550

30	DINV=ONE/RIN(JJ)	RTNC0560
	ROUT(JJ)=DINV	RTNC0570
	II=0	RTNC0580
	IK=1	RTNC0590
	JM1=J-1	RTNC0600
	DO 50 I=1,JM1	RTNC0610
	II=II+1	RTNC0620
	IK=II	RTNC0630
	SUM=Z	RTNC0640
	DO 40 K=I,JM1	RTNC0650
	SUM=SUM+ROUT(IK)*RIN(JJOLD+K)	RTNC0660
40	IK=IK+K	RTNC0670
50	ROUT(JJOLD+I)=-SUM*DINV	RTNC0680
C		RTNC0690
C		RTNC0700
C		RTNC0710
	RNMOUT=Z	RTNC0720
	DO 60 J=1,NTOT	RTNC0730
60	RNMOUT=RNMOUT+ROUT(J)**2	RTNC0740
C		RTNC0750
	RNM=DSQRT(RNM*RNMOUT)	RTNC0760
	CNB=RNM	RTNC0770
C		RTNC0780
	WRITE (6,110) RNM	RTNC0790
	RETURN	RTNC0800
C		RTNC0810
100	FORMAT (1H0,10X,'* * * MATRIX INVERSE COMPUTED ONLY UP TO BUT NOT	RTNC0820
	1 INCLUDING COLUMN',I4,' * * * MATRIX DIAGONAL ',I4,' IS ZERO * * '	RTNC0830
	2)	RTNC0840
110	FORMAT(1H0,5X,'CONDITION NUMBER ROUND=',D18.10,2X,'CNB/N.LE.CONDIT	RTNC0850
	ION NUMBER.LE.CNB',/)	RTNC0860
	END	RTNC0870

C	SUBROUTINE RI2COV (RINV,N,SIG,COVOUT,KROW,KCOL)	RT2C0010
C		RT2C0020
C	TO COMPUTE THE COVARIANCE MATRIX AND/OR THE STANDARD DEVIATIONS	RT2C0030
C	OF A VECTOR STORED UPPER TRIANGULAR SQUARE ROOT COVARIANCE	RT2C0040
C	MATRIX. THE OUTPUT COVARIANCE MATRIX IS ALSO VECTOR STORED.	RT2C0050
C		RT2C0060
C	RINV(N*(N+1)/2) INPUT VECTOR STORED UPPER TRIANGULAR	RT2C0070
C	COVARIANCE SQUARE ROOT. (RINV=PINVERSE	RT2C0080
C	IS THE INVERSE OF THE SRIF MATRIX)	RT2C0090
C	N DIMENSION OF THE RINV MATRIX, N.GE.2	RT2C0100
C	SIG(N) OUTPUT VECTOR OF STANDARD DEVIATIONS	RT2C0110
C	COVOUT(N*(N+1)/2) OUTPUT VECTOR STORED COVARIANCE MATRIX	RT2C0120
C	(COVOUT = RINV IS ALLOWED)	RT2C0130
C	KROW .GT.0 COMPUTES THE COVARIANCE AND SIGMAS	RT2C0140
C	CORRESPONDING TO THE FIRST KROW VARIABLES	RT2C0150
C	OF THE RINV MATRIX.	RT2C0160
C	.LT.0 COMPUTES ONLY THE SIGMAS OF THE FIRST KROW	RT2C0170
C	VARIABLES OF THE RINV MATRIX.	RT2C0180
C	RINV.	RT2C0190
C	.EQ.0 NO COVARIANCE, BUT ALL SIGMAS (F.G. USE	RT2C0200
C	N ROWS OF RINV).	RT2C0210
C	KCOL NO. OF COLUMNS OF COVOUT THAT ARE COMPUTED	RT2C0220
C	IF KCOL.LE.0 THEN KCOL=KROW. IF KROW.LE.0	RT2C0230
C	THIS INPUT IS IGNORED.	RT2C0240
C		RT2C0250
C	COGNIZANT PERSONS: G.J. RIEMAN/M.W. NEAD (JPL, MARCH 1978)	RT2C0260
C		RT2C0270
C	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	RT2C0280
C	DOUBLE PRECISION SUM	RT2C0290
C	DIMENSION RINV(1), SIG(1), COVOUT(1)	RT2C0300
C		RT2C0310
C	ZERO=0.D0	RT2C0320
C	LIM=N	RT2C0330
C	KKOL=KCOL	RT2C0340
C	IF (KKOL.LE.0) KKOL=KROW	RT2C0350
C	IF (KROW.NE.0) LIM=IABS(KROW)	RT2C0360
C	*** COMPUTE SIGMAS	RT2C0370
C	IKS=0	RT2C0380
C	DO 2 J=1,LIM	RT2C0390
C	IKS=IKS+J	RT2C0400
C	SUM=ZERO	RT2C0410
C	IK=IKS	RT2C0420
C	DO 1 K=J,N	RT2C0430
C	SUM=SUM+RINV(IK)**2	RT2C0440
C	1 IK=IK+K	RT2C0450
C	2 SIG(J)=DSORT(SUM)	RT2C0460
C		RT2C0470
C	IF (KROW.LE.0) RETURN	RT2C0480
C	*** COMPUTE COVARIANCE	RT2C0490
C		RT2C0500
C	NM1=LIM	RT2C0510
C	IF (KROW.EQ.N) NM1=N-1	RT2C0520
C	DO 10 J=1,NM1	RT2C0530
C	JJ=J+J	RT2C0540
C	COVOUT(JJ)=SIG(J)**2	RT2C0550

```

IJS=JJ+J
JP1=J+1
DO 10 I=JP1, KKOL
  IK=IJS
  IMJ=I-J
  SUM=ZERO
  DO 5 K=I, N
    IJK=IK+IMJ
    SUM=SUM+RINV(IK)*PINV(IJK)
  5   IK=IK+K
    COVOUT(IJS)=SUM
  10  IJS=IJS+I
C   IF (KROW.EQ.N) COVOUT(JJ+N)=SIG(N)**2
RETURN
END

```

```

RT2C0560
RT2C0570
RT2C0580
RT2C0590
RT2C0600
RT2C0610
RT2C0620
RT2C0630
RT2C0640
RT2C0650
RT2C0660
RT2C0670
RT2C0680
RT2C0690
RT2C0700
RT2C0710

```

	SUBROUTINE R2A(P,LR,NAMR,A,IA,LA,NAMA)	R2A00010
C		R2A00020
C	TO PLACE THE TRIANGULAR VECTOR STORED MATRIX R INTO THE	R2A00030
C	MATRIX A AND TO ARRANGE THE COLUMNS TO MATCH THE DESIRED	R2A00040
C	NAMA PARAMETER LIST. NAMES IN THE NAMA LIST THAT DO NOT	R2A00050
C	CORRESPOND TO ANY NAME IN NAMR HAVE ZERO ENTRIES IN THE	R2A00060
C	CORRESPONDING A COLUMN.	R2A00070
C		R2A00080
C	R(LR*(LR+1)/2) INPUT UPPER TRIANGULAR VECTOR STORED ARRAY	R2A00090
C	LR DIMENSION OF R	R2A00100
C	NAMR(L) PARAMETER NAMES ASSOCIATED WITH R	R2A00110
C	A(LR,LA) MATRIX TO HOUSE THE REARRANGED R MATRIX	R2A00120
C	IA ROW DIMENSION OF A, IA.GF.LR	R2A00130
C	LA NO. OF PARAMETER NAMES ASSOCIATED WITH THE	R2A00140
C	OUTPUT A MATRIX	R2A00150
C	NAMA(LA) PARAMETER NAMES FOR THE OUTPUT A MATRIX	R2A00160
C		R2A00170
C	COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976)	R2A00180
C		R2A00190
C	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	R2A00200
C	DIMENSION R(1),NAMR(1),A(IA,1),NAMA(1)	R2A00210
C		R2A00220
C	ZERO=0.	R2A00230
C	DO 5 J=1,LA	R2A00240
C	DO 5 K=1,LR	R2A00250
C	5 A(K,J)=ZERO @ ZERO A(LR,LA)	R2A00260
C	DO 40 J=1,LA	R2A00270
C	DO 10 I=1,LR	R2A00280
C	IF (NAMR(I).EQ.NAMA(J)) GO TO 20	R2A00290
C	10 CONTINUE	R2A00300
C	GO TO 40	R2A00310
C	20 JJ=I*(I-1)/2	R2A00320
C	DO 30 K=1,I	R2A00330
C	30 A(K,J)=R(JJ+K)	R2A00340
C	40 CONTINUE	R2A00350
C		R2A00360
C	RETURN	R2A00370
C	END	R2A00380

```

SUBROUTINE R2RA (R,NR,NAM,RA,NPA,NAMA)
C
C      TO COPY THE UPPER LEFT (LOWER RIGHT) PORTION OF A VECTOR
C      STORED UPPER TRIANGULAR MATRIX R INTO THE LOWER RIGHT
C      (UPPER LEFT) PORTION OF A VECTOR STORED TRIANGULAR
C      MATRIX RA.
C
C      R(NR*(NR+1)/2)  INPUT VECTOR STORED UPPER TRIANGULAR MATRIX
C      NR              DIMENSION OF R
C      NAM(NR)         NAMES ASSOCIATED WITH R
C
C      RA(NRA*(NRA+1)/2) OUTPUT VECTOR STORED UPPER TRIANGULAR MATRIX
C      NRA              IF NRA=0 ON INPUT, THEN NAMA(1) SHOULD HAVE
C                      THE FIRST NAME OF THE OUTPUT NAMLIST.
C                      IN THIS CASE THE NUMBER OF NAMES IN NAMA AND
C                      NRA WILL BE COMPUTED. THE LOWER RIGHT BLOCK
C                      OF R WILL BE THE UPPER LEFT BLOCK OF RA.
C                      IF NRA=LAST NAME OF THE UPPER LEFT BLOCK
C                      THAT IS TO BE MOVED, THEN THIS UPPER
C                      BLOCK IS TO BE MOVED TO THE LOWER RIGHT
C                      CORNER OF RA. WHEN USED IN THIS MODE NRA=NR
C                      ON OUTPUT.
C
C      NAMA(NRA)       NAMES ASSOCIATED WITH RA
C
C      IF NRA=0 ON INPUT, THEN NAMA(1) SHOULD HAVE THE FIRST NAME OF THE
C      OUTPUT NAMLIST AND THE NUMBER OF NAMES IN NAMA IS COMPUTED.
C      THE LOWER RIGHT BLOCK OF R WILL BE THE UPPER LEFT BLOCK OF RA.
C
C      IF NRA=LAST NAME OF THE UPPER LEFT BLOCK THAT IS TO BE MOVED,
C      THEN THE UPPER BLOCK IS TO BE MOVED TO THE LOWER RIGHT POSITION.
C      WHEN USED IN THIS MODE NRA=NR ON OUTPUT.
C
C      THE NAMES OF THE RELOCATED BLOCK ARE ALSO MOVED. THE RESULT
C      CAN COINCIDE WITH R AND NAMA WITH NAM.
C
C      COGNIZANT PERSONS:  G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976)
C
C      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)
C      DIMENSION  R(1),RA(1), NAM(1), NAMA(1)
C      LOGICAL    IS
C
C      IS=.FALSE.
C      LOCN=NAMA(1)
C
C      IS=FALSE CORRESPONDS TO MOVING UPPER LEFT. CORNER OF R TO
C      LOWER RT. CORNER OF RA
C      IF (NRA.EQ.0) GO TO 1
C      LOCN=NRA
C      IS=.TRUE.
C
C      IS=TRUE CORRESPONDS TO MOVING LOWER LEFT. CORNER OF R TO
C      UPPER RT. CORNER OF RA
C
C      1 DO 3 I=1,NR
C        IF (NAM(I).EQ.LOCN) GO TO 4
C      3 CONTINUE
C        WRITE (6,100)
C      100 FORMAT (1H0,20X,'NAMA(1) NOT IN NAMLIST OF R MATRIX')
C
R2PA0010
R2PA0020
R2PA0030
R2PA0040
R2PA0050
R2PA0060
R2PA0070
R2PA0080
R2PA0090
R2PA0100
R2PA0110
R2PA0120
R2PA0130
R2PA0140
R2PA0150
R2PA0160
R2PA0170
R2PA0180
R2PA0190
R2PA0200
R2PA0210
R2PA0220
R2PA0230
R2PA0240
R2PA0250
R2PA0260
R2PA0270
R2PA0280
R2PA0290
R2PA0300
R2PA0310
R2PA0320
R2PA0330
R2PA0340
R2PA0350
R2PA0360
R2PA0370
R2PA0380
R2PA0390
R2PA0400
R2PA0410
R2PA0420
R2PA0430
R2PA0440
R2PA0450
R2PA0460
R2PA0470
R2PA0480
R2PA0490
R2PA0500
R2PA0510
R2PA0520
R2PA0530
R2PA0540
R2PA0550

```

	RETURN	R2PA0560
C	4 K=I	R2PA0570
	KM1=K-1	R2PA0580
	IF (IS) GO TO 15	R2PA0590
C	IJS=K*(K+1)/2-1	R2PA0600
	NRA=NR-K+1	R2PA0610
	IJA=0	R2PA0620
	KOLA=0	R2PA0630
	DO 10 KOL=K, NR	R2PA0640
	KOLA=KOLA+1	R2PA0650
	NAMA(KOL-KM1)=NAM(KOL)	R2PA0660
	DO 5 IR=1, KOLA	R2PA0670
	IJA=IJA+1	R2PA0680
	5        RA(IJA)=R(IJS+IR)	R2PA0690
	10      IJS=IJS+KOL	R2PA0700
	RETURN	R2PA0710
C	15 IJ=K*(K+1)/2	R2PA0720
	IJA=NR*(NR+1)/2	R2PA0730
	L=NR-KM1	R2PA0740
	KOL=K	R2PA0750
	DO 25 KOLA=NR, L, -1	R2PA0760
	IJS=IJA	R2PA0770
	NAMA(KOLA)=NAM(KOL)	R2PA0780
	DO 20 IR=KOLA, L, -1	R2PA0790
	RA(IJS)=R(IJ)	R2PA0800
	IJS=IJS-1	R2PA0810
	20      IJ=IJ-1	R2PA0820
	IJA=IJA-KOLA	R2PA0830
	25      KOL=KOL-1	R2PA0840
	NRA=NR	R2PA0850
C	RETURN	R2PA0860
	END	R2PA0870
		R2PA0880
		R2PA0890
		R2PA0900
		R2PA0910

```

SUBROUTINE RUDR(RIN,N,ROUT,IS)
C
C FOR N.GT.0 THIS SUBROUTINE TRANSFORMS AN UPPER TRIANGULAR VECTOR
C STORED SRIF MATRIX TO U-D FORM, AND WHEN N.LT.0 THE U-D VECTOR
C STORED ARRAY IS TRANSFORMED TO A VECTOR STORED SRIF ARRAY
C
C RIN((N+1)*(N+2)/2) INPUT VECTOR STORED SRIF OR U-D ARRAY
C ROUT((N+1)*(N+2)/2) OUTPUT IS THE CORRESPONDING U-D OR SRIF
C ARRAY (PIN=ROUT IS PERMITTED)
C
C N ABS(N)= MATRIX DIMENSION .GE.2
C N.GT.0 THE (INPUT) SRIF ARRAY IS (OUTPUT)
C IN U-D FORM
C N.LT.0 THE (INPUT) U-D ARRAY IS (OUTPUT)
C IN SRIF FORM
C
C IS = 0 THERE IS NO RT. SIDE OR ESTIMATE STORED IN
C COLUMN N+1, AND RIN NEED HAVE ONLY
C N COLUMNS, I.E. RIN(N*(N+1)/2)
C
C IS = 1 THERE IS A RT. SIDE INPUT TO THE SRIF AND
C AN ESTIMATE FOR THE U-D ARRAY. THESE RESIDE
C IN COLUMN N+1.
C
C THIS SUBROUTINE USES SUBROUTINE RINCON
C
C COGIZANT PERSONS G.J.BIERMAN/M.W.HEAD (JPL, FFR.1978)
C
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C DIMENSION RIN(1), ROUT(1)
C
C ONE= 1.00
C NP1= IS + IABS(N)
C JJ=1
C IDIMR= NP1*(NP1 +1)/2
C IF (IS.EQ.0) GO TO 5
C RNN=RIN(IDIMR)
C RIN(IDIMR)=-ONE
C
C 5 IF (N.LT.0) GO TO 30
C CALL RINCON(RIN,NP1,ROUT,CNR)
C ROUT(1)= ROUT(1)**2
C DO 20 J=2,N
C S=ONE/ROUT(JJ+J)
C ROUT(JJ+J)= ROUT(JJ+J)**2
C JM1=J-1
C DO 10 I=1,JM1
C 10 ROUT(JJ+I)= ROUT(JJ+I)*S
C 20 JJ=JJ+ J
C GO TO 70
C
C 30 NN=-N
C ROUT(1)= SQRT(RIN(1))
C
C *** SOME MACHINES REQUIRE DSORT FOR DOUBLE PRECISION
C
C DO 50 J=2,NN
C ROUT(JJ+J)= SQRT(RIN(JJ+J))

```

```

RUDR0010
RUDR0020
RUDR0030
RUDR0040
RUDR0050
RUDR0060
RUDR0070
RUDR0080
RUDR0090
RUDR0100
RUDR0110
RUDR0120
RUDR0130
RUDR0140
RUDR0150
RUDR0160
RUDR0170
RUDR0180
RUDR0190
RUDR0200
RUDR0210
RUDR0220
RUDR0230
RUDR0240
RUDR0250
RUDR0260
RUDR0270
RUDR0280
RUDR0290
RUDR0300
RUDR0310
RUDR0320
RUDR0330
RUDR0340
RUDR0350
RUDR0360
RUDR0370
RUDR0380
RUDR0390
RUDR0400
RUDR0410
RUDR0420
RUDR0430
RUDR0440
RUDR0450
RUDR0460
RUDR0470
RUDR0480
RUDR0490
RUDR0500
RUDR0510
RUDR0520
RUDR0530
RUDR0540
RUDR0550

```



```
S=ROUT(JJ+J)
JM1=J-1
DO 40 I=1,JM1
40 ROUT(JJ+I)=RIN(JJ+I)*S
50 JJ=JJ+J
60 CALL RINCON(ROUT,NP1,ROUT,CNB)
C
70 IF (IS.EQ.1) RIN(IDIMR)=RNN
RETURN
END
```

```
RUDR0560
RUDR0570
RUDR0580
RUDR0590
RUDR0600
RUDR0610
RUDR0620
RUDR0630
RUDR0640
RUDR0650
```

```

SUBROUTINE SFU(FEL,IROW,JCOL,NF,II,N,FII,MAXFII,IFU,JDIAG)
C
C      TO COMPUTE FU(IFU,N)=F*U WHERE F IS SPARSE AND ONLY THE
C      NON-ZERO ELEMENTS ARE DEFINED AND U IS VECTOR STORED,
C      UPPER TRIANGULAR WITH IMPLICITLY DEFINED UNIT DIAGONAL
C      ELEMENTS
C      FEL(NF)      VALUES OF THE NON-ZERO ELEMENTS OF THE F MATRIX
C      IROW(NF)     ROW INDICES OF THE F ELEMENTS
C      JCOL(NF)     COLUMN INDICES OF THE F ELEMENTS
C                  F(IROW(K),JCOL(K))=FEL(K)
C      NF          NUMBER OF NON-ZERO ELEMENTS OF THE F MATRIX
C      U(N*(N+1)/2) UPPER TRIANGULAR VECTOR STORED MATRIX WITH
C                  IMPLICITLY DEFINED UNIT DIAGONAL ELEMENTS
C                  (U(J,J) ARE NOT, IN FACT, UNITY)
C      N          DIMENSION OF U MATRIX
C      FU(IFU,N)   OUTPUT RESULT
C      MAXFU      ROW DIMENSION OF FII MATRIX
C      IFU        NUMBER OF ROWS IN FII,
C                  (IFU.LE.MAXFII.AND.(IFU.GE.MAX(IROW(K))), K=1,...,NF;
C                  I.E. FII MUST HAVE AT LEAST AS MANY ROWS AS DOES F.
C                  ADDITIONAL ROWS OF FII COULD CORRESPOND TO ZERO
C                  ROWS OF F.
C      JDIAG(N)   DIAGONAL ELEMENT INDICES OF A VECTOR STORED
C                  UPPER TRIANGULAR MATRIX,
C                  I.E. JDIAG(K)=K*(K+1)/2=JDIAG(K-1)+K
C
C      COGNIZANT PERSONS: G.J.RICHMAN/M.W.NEAD (JPL, FEB.1978)
C
C      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C      DIMENSION FEL(NF),U(1),FII(MAXFII,N),IROW(NF),JCOL(NF),JDIAG(N)
C
C      ZERO=0.00
C      * * * INITIALIZE FII
C      DO 10 J=1,N
C        DO 10 I=1,IFU
C          10  FU(I,J)=ZERO
C          IF MAXFU=IFU, IT IS MORE EFFICIENT TO REPLACE THIS LOOP BY
C
C              DO 10 IJ=1,IFUN          @ IFUN=IFU*N
C              10  FU(IJ,1)=ZERO
C
C      DO 30 NEL=1,NF
C      NEL REPRESENTS THE ELEMENT NUMBER OF THE F MATRIX
C      I=IROW(NEL)
C      J=JCOL(NEL)
C      FIJ=FEL(NEL)
C      FU(I,J)=FU(I,J)+FIJ
C      THIS ACCOUNTS FOR THE IMPLICIT UNIT DIAGONAL U MATRIX
C      ELEMENTS. WHEN NON-UNIT DIAGONALS ARE USED, DELETE
C      THE ABOVE LINE AND USE J INSTEAD OF JP1 BELOW
C
C      IF (J.EQ.N) GO TO 30
C      WHEN IT IS KNOWN THAT THE LAST COLUMN OF F IS ZERO
C      THIS 'IF' TEST MAY BE OMITTED
C      JP1=J+1

```

SFU00010  
SFU00020  
SFU00030  
SFU00040  
SFU00050  
SFU00060  
SFU00070  
SFU00080  
SFU00090  
SFU00100  
SFU00110  
SFU00120  
SFU00130  
SFU00140  
SFU00150  
SFU00160  
SFU00170  
SFU00180  
SFU00190  
SFU00200  
SFU00210  
SFU00220  
SFU00230  
SFU00240  
SFU00250  
SFU00260  
SFU00270  
SFU00280  
SFU00290  
SFU00300  
SFU00310  
SFU00320  
SFU00330  
SFU00340  
SFU00350  
SFU00360  
SFU00370  
SFU00380  
SFU00390  
SFU00400  
SFU00410  
SFU00420  
SFU00430  
SFU00440  
SFU00450  
SFU00460  
SFU00470  
SFU00480  
SFU00490  
SFU00500  
SFU00510  
SFU00520  
SFU00530  
SFU00540  
SFU00550

```
IK=JDIAG(J)+J
DO 20 K=JP1,N
FU(I,K)=FU(I,K)+FIJ*U(IK)
20 IK=IK+K
30 CONTINUE
C
RETURN
END
```

```
SFU00560
SFU00570
SFU00580
SFU00590
SFU00600
SFU00610
SFU00620
SFU00630
```

```

SUBROUTINE TDHHT(S,MAXS,IRS,JCS,JSTART,JSTOP,V)
C
C     TDHHT TRANSFORMS A RECTANGULAR DOUBLE SUBSCRIBED MATRIX S
C     TO AN UPPER TRIANGULAR OR PARTIALLY UPPER TRIANGULAR FORM
C     BY THE APPLICATION OF HOUSEHOLDER ORTHOGONAL TRANSFORMATIONS.
C     IT IS ASSUMED THAT THE FIRST 'JSTART'-1 COLUMNS OF S ARE
C     ALREADY TRIANGULARIZED. THE ALGORITHM IS DESCRIBED IN
C     'FACTORIZATION METHODS FOR DISCRETE SEQUENTIAL ESTIMATION'
C     BY G.J. RIERMAN, ACADEMIC PRESS, 1977
C
C     S(IRS,JCS)  INPUT (POSSIBLY PARTIALLY) TRIANGULAR MATRIX. THE
C                 OUTPUT (POSSIBLY PARTIALLY) TRIANGULAR RESULT
C                 OVERWRITES THE INPUT.
C     MAXS        ROW DIMENSION OF S
C     IRS         NUMBER OF ROWS IN S (IRS.LE.MAXS.AND.IRS.GE.2)
C     JCS         NUMBER OF COLUMNS IN S
C     JSTART      INDEX OF THE FIRST COLUMN TO BE TRIANGULARIZED. IF
C                 JSTART.LT.1 IT IS ASSUMED THAT JSTART=1, I.E.
C                 START TRIANGULARIZATION AT COLUMN 1.
C     JSTOP       INDEX OF LAST COLUMN TO BE TRIANGULARIZED.
C                 IF JSTOP.LT.JSTART.OR.JSTOP.GT.JCS THEN
C                     IF IRS.LE.JCS JSTOP IS SET EQUAL TO IRS-1
C                     IF IRS.GT.JCS JSTOP IS SET EQUAL TO JCS
C                     I.E. THE TRIANGULARIZATION IS COMPLETED AS FAR
C                     AS POSSIBLE
C     V(IRS)      WORK VECTOR
C
C     COGNIZANT PERSONS: G.J. RIERMAN/M.W. NEAD (JPL, FEB. 1978)
C
C     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C     DIMENSION S(MAXS,JCS), V(IRS)
C     DOUBLE PRECISION SUM, DELTA
C
C     ONE=1.00
C     ZERO=0.00
C     JSTT=JSTART
C     JSTP=JSTOP
C     IF (JSTT.LT.1) JSTT=1
C     IF (JSTP.GE.JSTT.AND.JSTP.LE.JCS) GO TO 5
C     IF (IRS.LE.JCS) JSTP=IRS-1
C     IF (IRS.GT.JCS) JSTP=JCS
C
C     5 DO 40 J=JSTT,JSTP
C       SUM=ZERO
C       DO 10 I=J,IRS
C         V(I)=S(I,J)
C         S(I,J)=ZERO
C     10   SUM=SUM+V(I)**2
C       IF (SUM.LE.ZERO) GO TO 40
C       IF SUM=ZERO, COLUMN J IS ZERO AND THIS STEP OF THE
C       ALGORITHM IS OMITTED
C       SUM=DSQRT(SUM)
C       IF (V(J).GT.ZERO) SUM=-SUM
C       S(J,J)=SUM
C       V(J)=V(J)-SUM

```

```

SUM=ONE/(SUM*V(J))
C   THE HOUSEHOLDER TRANSFORMATION IS  $T=I-SUM*V*V**T$ 
   JP1=J+1
   IF (JP1.GT.JCS) GO TO 40
   DO 30 K=JP1,JCS
     DELTA=ZERO
     DO 20 I=J,IRS
20    DELTA=DELTA+S(I,K)*V(I)
     DELTA=DELTA*SUM
     DO 30 I=J,IRS
30    S(I,K)=S(I,K)+DELTA*V(I)
40    CONTINUE
C   RETURN
   END

```

```

T0HHT560
T0HHT570
T0HHT580
T0HHT590
T0HHT600
T0HHT610
T0HHT620
T0HHT630
T0HHT640
T0HHT650
T0HHT660
T0HHT670
T0HHT680
T0HHT690
T0HHT700

```

```

SUBROUTINE THH(R,N,A,IA,M,SOS,NSTRT)
C
C   THIS SUBROUTINE PERFORMS A TRIANGULARIZATION OF A RECTANGULAR
C   MATRIX INTO A SINGLY-SUBSCRIBED ARRAY BY APPLICATION OF
C   HOUSEHOLDER ORTHONORMAL TRANSFORMATIONS.
C
C   R(N*(N+3)/2) VECTOR STORED SQUARE ROOT INFORMATION MATRIX
C                   (LAST N LOCATIONS MAY CONTAIN A RIGHT HAND SIDE)
C   N               DIMENSION OF P MATRIX
C   A(M,N+1)       MEASUREMENT MATRIX
C   IA              ROW DIMENSION OF A
C   M               NUMBER OF ROWS OF A THAT ARE TO BE COMBINED WITH R
C                   (M.LE. IA)
C   SOS             ACCUMULATED ROOT SUM OF SQUARES OF THE RESIDUALS
C                   SORT(Z-A*X(EST)**2), INCLUDES A PRIORI
C                   SOS MUST BE INPUT AS A VARIABLE, NOT AS A
C                   NUMERICAL VALUE. IF INPUT SOS.LT.0, NO SOS
C                   COMPUTATION OCCURS.
C   NSTRT           FIRST COL OF THE INPUT A MATRIX THAT HAS A NONZERO
C                   ENTRY. IF NSTRT.LE.1, IT IS SET TO 1. THIS OPTION
C                   IS CONVENIENT WHEN PACKING A PRIORI BY BATCHES AND
C                   THE A MATRIX HAS LEADING COLUMNS OF ZEROS.
C
C   ON ENTRY R CONTAINS A PRIORI SQUARE ROOT INFORMATION FILTER (SRIF)
C   ARRAY, AND ON EXIT IT CONTAINS THE A POSTERIORI (PACKED) ARRAY.
C   ON ENTRY A CONTAINS OBSERVATIONS WHICH ARE DESTROYED BY THE
C   INTERNAL COMPUTATIONS.
C   ON ENTRY IF SOS IS .LT. ZERO, PROGRAM WILL ASSUME THERE IS NO
C   RIGHT HAND SIDE DATA AND WILL NOT ALTER SOS OR USE LAST N
C   LOCATIONS OF VECTOR R.
C
C   COGNIZANT PERSONS G.J.BIERMAN/N.HAMATA (JPL, MARCH 1978)
C
C   IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C   DIMENSION A(IA,1),R(1)
C   DOUBLE PRECISION SUM, ONE, BETA, DELTA
C
C   EPS=-1.D-200          @ MACHINE DEPENDENT ACCURACY TERM
C   ZERO=0.00
C   ONE=1.00
C   NSTART=NSTRT
C
C   IF (NSTART.LE.0) NSTART=1
C   NP1=N+1                @ NO. COLUMNS OF R
C   IF(SOS.LT.ZERO) NP1=N  @ NO COLS. = N IF SOS.LT.0
C   KK=NSTART*(NSTART-1)/2
C   DO 100 J=NSTART,N     @ J-TH STEP OF HOUSEHOLDER REDUCTION
C   KK=KK+J
C   SUM=ZERO
C   DO 20 I=1,M
C 20 SUM=SUM+A(I,J)**2
C   IF(SUM.LE.ZERO) GO TO 100 @ IF J-TH COL. OF A.F0.0 GO TO STEP J+1
C   SUM=SUM+R(KK)**2
C   SUM=DSQRT(SUM)

```

```

THH00010
THH00020
THH00030
THH00040
THH00050
THH00060
THH00070
THH00080
THH00090
THH00100
THH00110
THH00120
THH00130
THH00140
THH00150
THH00160
THH00170
THH00180
THH00190
THH00200
THH00210
THH00220
THH00230
THH00240
THH00250
THH00260
THH00270
THH00280
THH00290
THH00300
THH00310
THH00320
THH00330
THH00340
THH00350
THH00360
THH00370
THH00380
THH00390
THH00400
THH00410
THH00420
THH00430
THH00440
THH00450
THH00460
THH00470
THH00480
THH00490
THH00500
THH00510
THH00520
THH00530
THH00540
THH00550

```

```

IF(R(KK).GT.ZERO) SUM=-SUM
DELTA=R(KK)-SUM
R(KK)=SUM
JP1=J+1
IF (JP1.GT.NP1) GO TO 105
BETA=SUM*DELTA
IF (BETA.GT.EPS) GO TO 100
BETA=ONE/BETA
JJ=KK
L=J
C      ** READY TO APPLY J-TH HOUSEHOLDER TRANS.
DO 40 K=JP1,NP1
  JJ=JJ+L
  L=L+1
  SUM=DELTA*R(JJ)
  DO 30 I=1,M
30  SUM=SUM+A(I,J)*A(I,K)
  IF(SUM.EQ.ZERO) GO TO 40
  SUM=SUM*BETA
C      BETA DIVIDE USED HERE TO AVOID OVERFLOW IN
C      PROBLEMS WITH NEAR COLUMN COLLINEARITY. IN THAT CASE
C      COMMENT OUT LINE 630 AND CHANGE * TO / IN LINE 740
  R(JJ)=R(JJ)+SUM*DELTA
  DO 35 I=1,M
35  A(I,K)=A(I,K)+SUM*A(I,J)
  40 CONTINUE
100 CONTINUE
105 IF(SOS.LT.ZERO) RETURN
C
C      CALCULATE SOS
C
  SUM=ZERO
  DO 110 I=1,M
110 SUM=SUM+A(I,NP1)**2
  SOS=DSQRT(SOS**2+SUM)
C
  RETURN
  END

```

```

TMM00560
TMM00570
TMM00580
TMM00590
TMM00600
TMM00610
TMM00620
TMM00630
TMM00640
TMM00650
TMM00660
TMM00670
TMM00680
TMM00690
TMM00700
TMM00710
TMM00720
TMM00730
TMM00740
TMM00750
TMM00760
TMM00770
TMM00780
TMM00790
TMM00800
TMM00810
TMM00820
TMM00830
TMM00840
TMM00850
TMM00860
TMM00870
TMM00880
TMM00890
TMM00900
TMM00910
TMM00920
TMM00930

```

	SUBROUTINE TTHH(R,PA,N)	TTHH0010
C		TTHH0020
C	THIS SUBROUTINE COMBINES TWO SINGLE SUBSCRIBED SP1E ARRAYS	TTHH0030
C	USING HOUSEHOLDER ORTHOGONAL TRANSFORMATIONS	TTHH0040
C		TTHH0050
C	R(N*(N+1)/2) INPUT VECTOR STORED UPPER TRIANGULAR MATRIX.	TTHH0060
C	RESULT IS IN P	TTHH0070
C	RA(N*(N+1)/2) THE SECOND INPUT VECTOR STORED UPPER TRIANGULAR	TTHH0080
C	MATRIX. THIS MATRIX IS DESTROYED BY THE	TTHH0090
C	COMPUTATION	TTHH0100
C	N DIMENSION OF THE ESTIMATED PARAMETER VECTOR.	TTHH0110
C	A NEGATIVE VALUE FOR N IS USED TO NOTE THAT	TTHH0120
C	R AND RA HAVE RT. HAND SIDES INCLUDED AND	TTHH0130
C	HAVE DIM=ARS(N)*(ARS(N)+3)/2.	TTHH0140
C		TTHH0150
C	ON EXIT RA IS CHANGED AND R CONTAINS THE RESULTING SP1E ARRAY	TTHH0160
C		TTHH0170
C	COGNIZANT PERSONS G.J.RIERMAN/M.W.MEAD (JPL, JAN.1976)	TTHH0180
C		TTHH0190
C	IMPLICIT DOUBLE PRECISION(A-H,O-Z)	TTHH0200
C	DIMENSION RA(1), R(1)	TTHH0210
C	DOUBLE PRECISION SUM @ FOR USE IN SINGLE PRECISION VERSION	TTHH0220
C		TTHH0230
C	ZERO=0.	TTHH0240
C	ONE=1.	TTHH0250
C	NP1=N	TTHH0260
C	IF (N.GT.0) GO TO 10	TTHH0270
C	N=-N	TTHH0280
C	NP1=N+1	TTHH0290
C	10 IJS=1 @ IJ(START)	TTHH0300
C	KK=0	TTHH0310
C	DO 100 J=1,N @ J-TH STEP OF HOUSEHOLDER REDUCTION	TTHH0320
C	KK=KK+J	TTHH0330
C	SUM=R(KK)**2	TTHH0340
C	DO 20 I=IJS,KK	TTHH0350
C	20 SUM=SUM+RA(I)**2	TTHH0360
C	IF (SUM.LE.ZERO) GO TO 100	TTHH0370
C	SUM=SQRT(SUM)	TTHH0380
C	IF (R(KK).GT.ZERO) SUM=-SUM	TTHH0390
C	DELTA=R(KK)-SUM	TTHH0400
C	R(KK)=SUM	TTHH0410
C	BETA=ONE/(SUM*DELTA)	TTHH0420
C	JJ=KK	TTHH0430
C	L=J	TTHH0440
C	JP1=J+1	TTHH0450
C	IKS=KK+1	TTHH0460
C	* * * J-TH HOUSEHOLDER TRANS. DEFINED	TTHH0470
C	40 LOOP APPLIES TRANSFORM. TO COLS. J+1 TO NP1	TTHH0480
C	DO 40 K=JP1,NP1	TTHH0490
C	JJ=JJ+L	TTHH0500
C	L=L+1	TTHH0510
C	IK=IKS	TTHH0520
C	SUM=DELTA*R(JJ)	TTHH0530
C	DO 30 I=IJS,KK	TTHH0540
C	SUM=SIJM+RA(IK)*PA(I)	TTHH0550



```
30 IK=IK+1
   IF (SUM.EQ.ZERO) GO TO 40
   SUM=SUM*BETA
   R(JJ)=R(JJ)+SUM*DELTA
   IK=IKS
   DO 35 I=IJS,KK
   RA(IK)=RA(IK)+SUM*RA(I)
35 IK=IK+1
40 IKS=IKS+K
100 IJS=KK+1
C   RETURN
   END
```

```
TTHH0560
TTHH0570
TTHH0580
TTHH0590
TTHH0600
TTHH0610
TTHH0620
TTHH0630
TTHH0640
TTHH0650
TTHH0660
TTHH0670
TTHH0680
```

```

SUBROUTINE TWOMAT (A,N,LEN,CAR,TEXT,NCHAR,NAMES)
C
C      TO DISPLAY A VECTOR STORED UPPER TRIANGULAR MATRIX IN A
C      TWO-DIMENSIONAL TRIANGULAR FORMAT
C
C      A(N*(N+1)/2) VECTOR CONTAINING UPPER TRIANGULAR MATRIX      (DP)
C      N      DIMENSION OF MATRIX      (I)
C      LEN     NUMFR OF COLUMNS TO BE PRINTED, 7 OR 12      (I)
C      CAR(N)  PARAMETER NAMES      (I)
C      TEXT( ) AN ARRAY OF FIELDATA CHARACTERS TO BE PRINTED AS
C              A TITLE PRECEDING THE MATRIX
C      NCHAR   NUMBER OF CHARACTERS, INCLUDING SPACES, THAT
C              ARE TO BE PRINTED IN TEXT( )
C              ABS(NCHAR).LE.114. NCHAR NEGATIVE IS USED
C              TO AVOID SKIPPING TO A NEW PAGE TO START
C              PRINTING
C      NAMES   TRUE TO PRINT PARAMETER NAMES
C
C      COGNIZANT PERSON: M.W.NEAD (JPL, OCT. 1977)
C
C      PARAMETER J12=12, J7=7
C      DOUBLE PRECISION A(N)
C      INTEGER CAR(N), TEXT(1), L(J12), LIST(J12)
C      LOGICAL NAMES
C      INTEGER V(4),VFMT(J12),V7MT(J7),V12MT(J12)
C      DATA V/'(2X,'A6,1X,'E10.5)'/,(V12MT(I),I=1,12)
C      1 /'12','10X,11','20X,10','30X,9','040X,8','050X,7',
C      2 '060X,6','070X,5','080X,4','090X,3','100X,2','110X,1'/,
C      1 V7MT/'7','017X,6','034X,5','051X,4','068X,3','085X,2','102X,1'//
C      DATA KON7/'017.8)'/, KON12/'E10.5)'/
C
C      M1,M2   ROW LIMITS FOR EACH PRINT SEQUENCE
C      N1,M2   COL LIMITS FOR EACH LINE OF PRINT
C      L(I)    LOC OF EACH COLUMN IN A ROW
C      KT      ROW COUNTER
C
C      * * * * * INITIALIZE COUNTERS
C
C      IF (LEN.EQ.J0) GO TO 5
C      IF (LEN.EQ.7) GO TO 1
C      IF (LEN.EQ.12) GO TO 2
C      WRITE (6,230) LEN
C      LEN=12
C      GO TO 2
C      1 V(4)=KON7; J0=7; J0M1=J0-1; J0P1=J0+1;
C      1 REPEAT I=1,J0; VFMT(I)=V7MT(I)
C      GO TO 5
C      2 V(4)=KON12; J0=12; J0M1=J0-1; J0P1=J0+1;
C      1 REPEAT I=1,J0; VFMT(I)=V12MT(I)
C      5 M1=1
C      M2=J0
C      N1=1
C      KT=0
C      V(2)='A6,1X,'
C      IF (.NOT.NAMES) V(2)='15,2X'

```

<pre> C NC=IABS(NCHAR)/6 IF (MOD(NCHAR,6).NE.0) NC=NC+1 IF (NCHAR.GE.0) WRITE (6,200) (TEXT(I),I=1,NC) IF (NCHAR.LT.0) WRITE (6,205) (TEXT(I),I=1,NC) 10 IF (M2.GT.N) M2=N IF (.NOT.NAMES) GO TO 20 IF (LEN.EQ.7) WRITE (6,210) (CAR(I),I=N1,M2) IF (LEN.EQ.12) WRITE (6,211) (CAR(I),I=N1,M2) GO TO 40 20 M=N1 L2=M2-N1+1 DO 30 I=1,L2 LIST(I)=M 30 M=M+1 IF (LEN.EQ.7) WRITE (6,220) (LIST(I),I=1,L2) IF (LEN.EQ.12) WRITE (6,221) (LIST(I),I=1,L2) 40 CONTINUE C ***** DO 190 IC=M1,M2 K=1 IF (IC.LE.(KT*J0)) GO TO 60 JJ=0 DO 50 J=1,IC 50 JJ=JJ+J L(K)=JJ I1=IC-KT*J0 IF (I1.EQ.J0) GO TO 90 GO TO 70 60 CONTINUE C I1=1 L(K)=L(K)+1 70 CONTINUE DO 80 I=I1,J0M1 K=K+1 I1=I+KT*J0 80 L(K)=L(K-1)+I1 90 CONTINUE C I2=M1N0(J0P1,(M2+1-KT*J0))-I1 V(3)=VFMT(I1) IF (.NOT.NAMES) GO TO 180 WRITE (6,V) CAR(IC),(A(L(I)),I=1,I2) GO TO 190 180 WRITE (6,V) IC,(A(L(I)),I=1,I2) 190 CONTINUE IF (M2.EQ.N) RETURN N1=M2+1 M2=M2+J0 KT=KT+1 IF (NCHAR.GE.0) WRITE (6,201) (TEXT(I),I=1,NC) IF (NCHAR.LT.0) WRITE (6,206) (TEXT(I),I=1,NC) GO TO 10 C 200 FORMAT (1H1,2X,21A6) 205 FORMAT (1H0,2X,21A6) </pre>	<pre> TWOM0560 TWOM0570 TWOM0580 TWOM0590 TWOM0600 TWOM0610 TWOM0620 TWOM0630 TWOM0640 TWOM0650 TWOM0660 TWOM0670 TWOM0680 TWOM0690 TWOM0700 TWOM0710 TWOM0720 TWOM0730 TWOM0740 TWOM0750 TWOM0760 TWOM0770 TWOM0780 TWOM0790 TWOM0800 TWOM0810 TWOM0820 TWOM0830 TWOM0840 TWOM0850 TWOM0860 TWOM0870 TWOM0880 TWOM0890 TWOM0900 TWOM0910 TWOM0920 TWOM0930 TWOM0940 TWOM0950 TWOM0960 TWOM0970 TWOM0980 TWOM0990 TWOM1000 TWOM1010 TWOM1020 TWOM1030 TWOM1040 TWOM1050 TWOM1060 TWOM1070 TWOM1080 TWOM1090 TWOM1100 TWOM1110 TWOM1120 </pre>
---	---

Q OBTAIN COL INDEX FOR ROW

Q TITLE  
Q TITLE

201	FORMAT (1H1,2X,'(CONTINUE) ',19A6)	@ TITLE	TWOM1130
206	FORMAT (1H0,2X,'(CONTINUE) ',19A6)	@ TITLE	TWOM1140
210	FORMAT (1H0,5X,7(11X,A6))	@ HORIZONTAL NAMES	TWOM1150
220	FORMAT (1H0,3X,7(11X,I6))		TWOM1160
211	FORMAT (1H0,5X,12(4X,A6))	@ HORIZONTAL NAMES	TWOM1170
221	FORMAT (1H0,3X,12(4X,I6))		TWOM1180
230	FORMAT (1H0,20X,'TWO MAT CALLED WITH LENGTH = ',I3)		TWOM1190
C	END		TWOM1200
			TWOM1210

	SUBROUTINE TZERO (R,N,IS,IF)	TZER0000
C		TZER0010
C	TO ZERO OUT ROWS IS (ISTART) TO IF (IFINAL) OF A VECTOR	TZER0020
C	STORED UPPER TRIANGULAR MATRIX	TZER0030
C		TZER0040
C	R(N*(N+1)/2) INPUT VECTOR STORED UPPER TRIANGULAR MATRIX	TZER0050
C	N DIMENSION OF R	TZER0060
C	IS FIRST ROW OF R THAT IS TO BE SET TO ZERO	TZER0070
C	IR LAST ROW OF R THAT IS TO BE SET TO ZERO	TZER0080
C		TZER0090
C	COGNIZANT PERSONS: G.J.RIERMAN/C.F.PETERS (JPL, NOV. 1975)	TZER0100
C		TZER0110
	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	TZER0120
	DIMENSION R(1)	TZER0130
C		TZER0140
	ZERO=0.D0	TZER0150
	IJS=IS*(IS-1)/2	TZER0160
	DO 10 I=IS,IF	TZER0170
	IJS=IJS+I	TZER0180
	IJ=IJS	TZER0190
	DO 10 J=I,N	TZER0200
	R(IJ)=ZERO	TZER0210
	IJ=IJ+J	TZER0220
C	10 CONTINUE	TZER0230
		TZER0240
	RETURN	TZER0250
	END	TZER0260

```

SUBROUTINE UDCOL(U,N,KS,NCOLOR,V,FM,Q)                                UNCOL010
C                                                                     UNCOL020
C COLORED NOISE UPDATING OF THE U-D COVARIANCE FACTORS, I.E.         UNCOL030
C   U*D*(U**T)-OUTPUT=PHI*U*D*(U**T)*(PHI**T)+Q                     UNCOL040
C   PHI=DIAG(O(KS-1),FM(1),...,EM(NCOLOR),O(N-(KS-1+NCOLOR)))     UNCOL050
C   W=DIAG(O(KS-1),Q(1),...,Q(NCOLOR),O(N-(KS-1+NCOLOR)))        UNCOL060
C   O(K) IS A VECTOR OF ZEROS                                       UNCOL070
C                                                                     UNCOL080
C THE ALGORITHM USED IS THE BIERMAN-THORNTON ONE COMPONENT        UNCOL090
C AT-A-TIME UPDATE. CF. BIERMAN FACTORIZATION METHOD                UNCOL100
C FOR DISCRETE SEQUENTIAL ESTIMATION, ACADEMIC PRESS (1977)      UNCOL110
C PP.147-148                                                        UNCOL120
C                                                                     UNCOL130
C U(N*(N+1)/2) INPUT U-D VECTOR STORED COVARIANCE FACTORS,        UNCOL140
C THE COLORED NOISE UPDATE RESULT RESIDES                           UNCOL150
C IN U ON OUTPUT                                                  UNCOL160
C N                                                                  UNCOL170
C FILTER DIMENSION. IF THE LAST COLUMN OF U                        UNCOL180
C HOUSES THE FILTER ESTIMATES, THEN                                UNCOL190
C N=NUMBER FILTER VARIABLES + 1
C KS                                                                 UNCOL200
C THE LOCATION OF THE FIRST COLORED NOISE TERM                    UNCOL210
C (KS.GE.1.AND.KS.LE.N)
C NCOLOR                                                            UNCOL220
C THE NUMBER OF COLORED NOISE TERMS (NCOLOR.GE.1)
C V(KS-1+NCOLOR) WORK VECTOR                                       UNCOL230
C EM(NCOLOR)                                                        UNCOL240
C INPUT VECTOR OF COLORED NOISE MAPPING TERMS
C (UNALTERED BY PROGRAM)                                         UNCOL250
C Q(NCOLOR)                                                         UNCOL260
C INPUT VECTOR OF PROCESS NOISE VARIANCES
C (UNALTERED BY PROGRAM)                                         UNCOL270
C                                                                     UNCOL280
C SUBROUTINE REQUIRED: RANK1                                         UNCOL290
C                                                                     UNCOL300
C COGNIZANT PERSON: G.J. BIERMAN (JPL, JAN. 1978)                UNCOL310
C DOUBLE PRECISION TMP,S                                           UNCOL320
C IMPLICIT DOUBLE PRECISION (A-H,O-Z)                              UNCOL330
C DIMENSION U(1),V(1),FM(1),Q(1)                                  UNCOL340
C                                                                     UNCOL350
C * * * * * INITIALIZATION                                          UNCOL360
C   NM1=N-1                                                         UNCOL370
C   KSM1=KS-1                                                       UNCOL380
C   JJOLD=KS*KSM1/2                                                 UNCOL390
C   KOL=KSM1                                                         UNCOL400
C * * * * *                                                         UNCOL410
C                                                                     UNCOL420
C   DO 50 K=1,NCOLOR                                                UNCOL430
C     KOLM1=KOL                                                       UNCOL440
C     KOL=KOL+1                                                       UNCOL450
C     JJ=JJOLD+KOL                                                  UNCOL460
C     TMP=U(JJ)*EM(K)                                                 UNCOL470
C     C=Q(K)*U(JJ)                                                   UNCOL480
C     S=TMP*EM(K)+Q(K)                                               UNCOL490
C     U(JJ)=S                                                         UNCOL500
C                                                                     UNCOL510
C   IF (KOL.GE.N) GO TO 20                                           UNCOL520
C     IJ=JJ                                                           UNCOL530
C     DO 10 J=KOL,NM1                                               UNCOL540
C       IJ=IJ+J                                                       UNCOL550

```

C	10	U(IJ)=U(IJ)*EM(K)	R UPDATING ROW KOL ENTRIES	UNCOL560
	20	IF (JJ.EQ.1) GO TO 50	R (WHEN KS=1, N=1)	UNCOL570
		IF (S.LE.0.D0) GO TO 30		UNCOL580
		TMP=TMP/S	R TMP=EM(K)*D(KOL)-OLD/D(KOL)-NEW	UNCOL590
		C=C/S	R C=G(K)*D(KOL)-OLD/D(KOL)-NEW	UNCOL600
	30	DO 40 I=1,KOLM1		UNCOL610
		V(I)=U(JJOLD+I)		UNCOL620
	40	U(JJOLD+I)=TMP*V(I)		UNCOL630
		IF (KOLM1.GT.1) GO TO 45		UNCOL640
		U(1)=U(1)+C*V(1)**2		UNCOL650
		GO TO 50		UNCOL660
	45	CALL RANK1(U,U,KOLM1,C,V)		UNCOL670
	50	JJOLD=JJ		UNCOL680
C		RETURN		UNCOL690
		END		UNCOL700
				UNCOL710
				UNCOL720

	SUBROUTINE UDMEAS (U,N,R,A,F,G,ALPHA)	UDMEA010
C		UDMEA020
C	COMPUTES ESTIMATE AND U-D MEASUREMENT UPDATED	UDMEA030
C	COVARIANCE, P=U*D*U**T	UDMEA040
C	*** INPUTS ***	UDMEA050
C		UDMEA060
C	U UPPER TRIANGULAR MATRIX, WITH N ELEMENTS STORED AS THE	UDMEA070
C	DIAGONAL. U IS VECTOR STORED AND CORRESPONDS TO THE	UDMEA080
C	A PRIORI COVARIANCE. IF STATE ESTIMATES ARE COMPUTED,	UDMEA090
C	THE LAST COLUMN OF U CONTAINS X.	UDMEA100
C	N DIMENSION OF THE STATE ESTIMATE. N.GT.1	UDMEA110
C	R MEASUREMENT VARIANCE	UDMEA120
C	A VECTOR OF MEASUREMENT COEFFICIENTS, IF DATA THEN A(N+1)=Z	UDMEA130
C	ALPHA IF ALPHA LESS THAN ZERO NO ESTIMATES ARE COMPUTED	UDMEA140
C	(AND X AND Z NEED NOT BE INCLUDED)	UDMEA150
C		UDMEA160
C	*** OUTPUTS ***	UDMEA170
C		UDMEA180
C	U UPDATED, VECTOR STORED FACTORS AND ESTIMATE AND	UDMEA190
C	U((N+1)(N+2)/2) CONTAINS (Z-A**T*X)	UDMEA200
C		UDMEA210
C	ALPHA INNOVATIONS VARIANCE OF THE MEASUREMENT RESIDUAL	UDMEA220
C	G VECTOR OF UNWEIGHTED KALMAN GAINS. THE KALMAN	UDMEA230
C	GAIN K IS EQUAL TO G/ALPHA	UDMEA240
C	F CONTAINS U**T*A AND (Z-A**T*X)/ALPHA	UDMEA250
C	ONE CAN HAVE F OVERWRITE A TO SAVE STORAGE	UDMEA260
C		UDMEA270
C		UDMEA280
C	COGNIZANT PERSONS: G.J. BIERMAN/M.W. NEAD (JPL, FEB.197A)	UDMEA290
C		UDMEA300
C	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	UDMEA310
C	DIMENSION U(1), A(1), F(1), G(1)	UDMEA320
C	DOUBLE PRECISION SUM,BETA,GAMMA	UDMEA330
C	LOGICAL IEST	UDMEA340
C		UDMEA350
C	ZERO=0.00	UDMEA360
C	IEST=.FALSE.	UDMEA370
C	ONE=1.00	UDMEA380
C	NP1=N+1	UDMEA390
C	NP2=N+2	UDMEA400
C	NTOT=N*NP1/2	UDMEA410
C	IF (ALPHA.LT.ZERO) GO TO 3	UDMEA420
C	SUM=A(NP1)	UDMEA430
C	DO 1 J=1,N	UDMEA440
C	1 SUM=SUM-A(J)*U(NTOT+J)	UDMEA450
C	U(NTOT+NP1)=SUM	UDMEA460
C	IEST=.TRUE.	UDMEA470
C		UDMEA480
C	3 JJN=NTOT	UDMEA490
C	DO 10 L=2,N	UDMEA500
C	J=NP2-L	UDMEA510
C	JJ=JJN-J	UDMEA520
C	SUM=A(J)	UDMEA530
C	JM1=J-1	UDMEA540
C	DO 5 K=1,JM1	UDMEA550



5	SUM=SUM+U(JJ+K)*I(K)		UNMFA560
	F(J)=SUM		UNMFA570
	G(J)=SUM*U(JJN)		UNMFA580
10	JJN=JJ		UNMFA590
	F(1)=A(1)		UNMFA600
	G(1)=U(1)*F(1)		UNMFA610
C	F=U**T*A AND G=n*(U**T*A)		UNMFA620
C	SUM=R+G(1)*F(1)	@ SUM(1)	UNMFA630
	GAMMA=0	@ FOR R=0 CASE	UNMFA640
	IF (SUM.GT.ZERO) GAMMA=ONE/SUM	@ FOR R=0 CASE	UNMFA650
	IF (F(1).NE.ZERO) U(1)=U(1)*R/GAMMA	@ D(1)	UNMFA660
C	KJ=2		UNMFA670
	DO 20 J=2,N		UNMFA680
	BETA=SUM	@ BETA=SUM(J-1)	UNMFA690
	TEMP=G(J)		UNMFA700
	SUM=SUM+TEMP+F(J)	@ SUM(J)	UNMFA710
	P=-F(J)*GAMMA	@ P=-F(J)*(1/SUM(J-1)) EQN(21)	UNMFA720
	JM1=J-1		UNMFA730
	DO 15 K=1,JM1		UNMFA740
	S=U(KJ)		UNMFA750
	J(KJ)=S+P*G(K)	@ EQN(22)	UNMFA760
	G(K)=G(K)+TEMP*S	@ EQN(23)	UNMFA770
15	KJ=KJ+1		UNMFA780
	IF (TEMP.EQ.ZERO) GO TO 20	@ FOR R=0 CASE	UNMFA790
	GAMMA=ONE/SUM	@ GAMMA=1/SUM(J)	UNMFA800
	U(KJ)=U(KJ)*BETA*GAMMA	@ D(J) EQN(19)	UNMFA810
20	KJ=KJ+1		UNMFA820
	ALPHA=SUM		UNMFA830
C			UNMFA840
C			UNMFA850
C			UNMFA860
	EQN. NOS. REFER TO BIERMAN'S 1975 CDC PAPER, PP. 337-346.		UNMFA870
			UNMFA880
	IF (.NOT.IEST) RETURN		UNMFA890
	F(NP1)=U(NTOT+NP1)*GAMMA		UNMFA900
	DO 30 J=1,N		UNMFA910
30	U(NTOT+J)=U(NTOT+J)+G(J)*F(NP1)		UNMFA920
C			UNMFA930
	RETURN		UNMFA940
	END		UNMFA950

```

SUBROUTINE UD2COV (UIN,POUT,N)                                UD2C0010
C                                                                 UD2C0020
C TO OBTAIN A COVARIANCE FROM ITS U-D FACTORIZATION, BOTH MATRICES UD2C0030
C ARE VECTOR STORED AND THE OUTPUT COVARIANCE CAN OVERWRITE THE UD2C0040
C INPUT U-D ARRAY. UIN=U-D IS RELATED TO POUT VIA POUT=UDU(/**T) UD2C0050
C                                                                 UD2C0060
C UIN(N*(N+1)/2) INPUT U-D FACTORS, VECTOR STORED WITH THE D UD2C0070
C ENTRIES STORED ON THE DIAGONAL OF UIN UD2C0080
C POUT(N*(N+1)/2) OUTPUT COVARIANCE, VECTOR STORED. UD2C0090
C (POUT=UIN IS PERMITTED) UD2C0100
C N DIMENSION OF THE MATRICES INVOLVED, N.GT.1 UD2C0110
C UD2C0120
C COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, FEB. 1977) UD2C0130
C UD2C0140
C IMPLICIT DOUBLE PRECISION (A-H,O-Z) UD2C0150
C UD2C0160
C DIMENSION UIN(1), POUT(1) UD2C0170
C UD2C0180
C POUT(1)=UIN(1) UD2C0190
C JJ=1 UD2C0200
C DO 20 J=2,N UD2C0210
C   JJL=JJ UD2C0220
C   JJ=JJ+J UD2C0230
C   POUT(JJ)=UIN(JJ) UD2C0240
C   S=POUT(JJ) UD2C0250
C   II=0 UD2C0260
C   JM1=J-1 UD2C0270
C   DO 20 I=1,JM1 UD2C0280
C     II=II+I UD2C0290
C     ALPHA=S*UIN(JJL+I) UD2C0300
C     IK=II UD2C0310
C     DO 10 K=I,JM1 UD2C0320
C       POUT(IK)=POUT(IK)+ALPHA*UIN(JJL+K) UD2C0330
C       IK=IK+K UD2C0340
C     10 POUT(JJL+I)=ALPHA UD2C0350
C UD2C0360
C RETURN UD2C0370
C END UD2C0380

```

C	SUBROUTINE UD2SIG(U,N,SIG,TEXT,NCT)	UN2SI010
C		UN2SI020
C	COMPUTE STANDARD DEVIATIONS (SIGMAS) FROM U-D COVARIANCE FACTORS	UN2SI030
C		UN2SI040
C	U(N*(N+1)/2) INPUT VECTOR STORED ARRAY CONTAINING THE U-D	UN2SI050
C	FACTORS. THE D (DIAGONAL) ELEMENTS ARE STORED	UN2SI060
C	ON THE DIAGONAL	UN2SI070
C	N U MATRIX DIMENSION, N.GT.1	UN2SI080
C	SIG(N) VECTOR OF OUTPUT STANDARD DEVIATIONS	UN2SI090
C	TEXT( ) ARRAY OF FIELDATA CHARACTERS TO BE PRINTED	UN2SI100
C	PRECEDING THE VECTOR OF SIGMAS	UN2SI110
C	NCT NUMBER OF CHARACTERS IN TEXT, 0.LE.NCT.LE.126	UN2SI120
C	IF NCT=0, NO SIGMAS ARE PRINTED	UN2SI130
C		UN2SI140
C	COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, FEB. 1977)	UN2SI150
C		UN2SI160
C	IMPLICIT DOUBLE PRECISION (A-H,O-Z)	UN2SI170
C	INTEGER TEXT(1)	UN2SI180
C	DIMENSION U(1), SIG(1)	UN2SI190
C		UN2SI200
C	JJ=1	UN2SI210
C	SIG(1)=U(1)	UN2SI220
C	DO 10 J=2,N	UN2SI230
C	JJL=JJ	UN2SI240
C	JJ=JJ+J	UN2SI250
C	S=U(JJ)	UN2SI260
C	SIG(J)=S	UN2SI270
C	JM1=J-1	UN2SI280
C	DO 10 I=1,JM1	UN2SI290
C	10    SIG(I)=SIG(I)+S*U(JJL+I)**2	UN2SI300
C		UN2SI310
C	WE NOW HAVE VARIANCES	UN2SI320
C		UN2SI330
C	DO 20 J=1,N	UN2SI340
C	20    SIG(J)=SQRT(SIG(J))	UN2SI350
C	IF (NCT.EQ.0) GO TO 30	UN2SI360
C	NC=NCT/6	UN2SI370
C	IF (MOD(NC,6).NE.0) NC=NC+1	UN2SI380
C	WRITE (6,40) (TEXT(I),I=1,NC)	UN2SI390
C	WRITE (6,50) (SIG(I),I=1,N)	UN2SI400
C	30    RETURN	UN2SI410
C		UN2SI420
C	40    FORMAT (1H0,2X,21A6)	UN2SI430
C	50    FORMAT (1H0,(6D18.10))	UN2SI440
C	END	UN2SI450

```

SUBROUTINE UTINV(RIN,N,ROUT)
C
C      TO INVERT AN UPPER TRIANGULAR VECTOR STORED MATRIX AND STORE
C      THE RESULT IN VECTOR FORM. THE ALGORITHM IS SO ARRANGED THAT
C      THE RESULT CAN OVERWRITE THE INPUT.
C      IN ADDITION TO SOLVE  $RX=Z$ , SET  $RIN(N*(N+1)/2+1)=Z(1)$ , ETC.,
C      AND SET  $RIN((N+1)*(N+2)/2)=-1$ . CALL THE SUBROUTINE USING N+1
C      INSTEAD OF N. ON RETURN THE FIRST N ENTRIES OF COLUMN N+1
C      WILL CONTAIN X.
C
C      RIN(N*(N+1)/2) INPUT VECTOR STORED UPPER TRIANGULAR MATRIX
C      N           MATRIX DIMENSION
C      ROUT(N*(N+1)/2) OUTPUT VECTOR STORED UPPER TRIANGULAR MATRIX
C      INVERSE
C
C      COGNIZANT PERSONS: G.J. RIEMAN/M.W. NEAD (JPL, JAN. 1978)
C
C      DOUBLE PRECISION RIN(1), ROUT(1), ZERO, DINV, ONE, SUM
C
C      ZERO=0.00
C      ONE=1.00
C
C      IF (RIN(1).NE.ZERO) GO TO 5
C      J=1
C      WRITE (6,100) J,J
C      RETURN
C
C      5 ROUT(1)=ONE/RIN(1)
C
C      JJ=1
C      DO 20 J=2,N
C      JJOLD=JJ
C      JJ=JJ+J
C      IF (RIN(JJ).NE.ZERO) GO TO 10
C      WRITE (6,100) J,J
C      RETURN
C
C      10 DINV=ONE/RIN(JJ)
C      ROUT(JJ)=DINV
C      II=0
C      IK=1
C      JM1=J-1
C      DO 20 I=1,JM1
C      II=II+I
C      IK=II
C      SUM=ZERO
C      DO 15 K=I,JM1
C      SUM=SUM+ROUT(IK)*RIN(JJOLD+K)
C      15 IK=IK+K
C      20 ROUT(JJOLD+I)=-SUM*DINV
C
C      RETURN
C
C      100 FORMAT (1HC,10X,'* * * MATRIX INVERSE COMPUTED ONLY UP TO BUT NOT UTINV540
C      INCLUDING COLUMN',I4,' * * * MATRIX DIAGONAL ',I4,' IS ZERO * * *'UTINV550

```

C 2)  
END

UTINV560  
UTINV570  
UTINV580



C  
20 IK=IK+K  
30 ROUT(KJS)=-SUM\*DIINV  
35 IRLST=IROW  
40 II=II-IROW  
RETURN  
50 FORMAT (1H0,10X,'RIN DIAGONAL',I4,' IS ZFRO')  
END

UTIP0550  
UTIP0560  
UTIP0570  
UTIP0580  
UTIP0590  
UTIP0600  
UTIP0610  
UTIP0620

T  
2  
E

```

C SUBROUTINE WGS (W,IMAXW,IW,JW,DW,U,V) WGS00010
C MODIFIED GRAMM-SCHMIDT ALGORITHM FOR REDUCING WDW(**T) TO UDU(**T) WGS00020
C FORM WHERE U IS A VECTOR STORED TRIANGULAR MATRIX WITH THE WGS00030
C RESULTING D ELEMENTS STORE ON THE DIAGONAL WGS00040
C W(IW,JW) INPUT MATRIX TO BE REDUCED TO TRIANGULAR FORM. WGS00050
C THIS MATRIX IS DESTROYED BY THE CALCULATION WGS00060
C IW.LE.IMAXW.AND.IW.GT.1 WGS00070
C IMAXW ROW DIMENSION OF W MATRIX WGS00080
C IW NO. ROWS OF W MATRIX, DIMENSION OF U WGS00090
C JW NO. COLS OF W MATRIX WGS00110
C DW(JW) VECTOR OF NON-NEGATIVE WEIGHTS FOR THE WGS00120
C ORTHOGONALIZATION PROCESS. THE D'S ARE UNCHANGED WGS00130
C BY THE CALCULATION. WGS00140
C U(IW*(IW+1)/2) OUTPUT UPPER TRIANGULAR VECTOR STORED OUTPUT WGS00150
C V(JW) WORK VECTOR WGS00160
C WGS00170
C (SEE BOOK: WGS00180
C ' FACTORIZATION METHODS FOR DISCRETE SEQUENTIAL ESTIMATION ', WGS00190
C BY G.J.BIERMAN) WGS00200
C ESTIMATION WGS00210
C WGS00220
C COGNIZANT PERSONS: G.J.BIERMAN/M.W.MEAD (JPL, FEB.1978) WGS00230
C WGS00240
C IMPLICIT DOUBLE PRECISION (A-H,O-Z) WGS00250
C DOUBLE PRECISION SUM,Z,DINV WGS00260
C DIMENSION W(IMAXW,1), DW(1), U(1), V(1) WGS00270
C WGS00280
C Z=0.00 WGS00290
C ONE=1.00 WGS00300
C IWP2=IW+2 WGS00310
C DO 100 L=2,IW WGS00320
C J=IWP2-L WGS00330
C SUM=Z WGS00340
C DO 40 K=1,JW WGS00350
C V(K)=W(J,K) WGS00360
C U(K)=DW(K)*V(K) WGS00370
C SUM=V(K)*U(K)+SUM WGS00380
40 W(J,J)=SUM WGS00390
C DINV=SUM WGS00400
C JM1=J-1 WGS00410
C IF (SUM.GT.Z) GO TO 45 WGS00420
C W(J,.)=0. WHEN DINV=0 (DINV=NORM(W(J,.)**2)) WGS00430
C DO 44 K=1,JM1 WGS00440
44 W(J,K)=Z WGS00450
C GO TO 100 WGS00460
45 DO 70 K=1,JM1 WGS00470
C SUM=Z WGS00480
C DO 50 I=1,JW WGS00490
50 SUM=W(K,I)*U(I)+SUM WGS00500
C SUM=SUM/DINV WGS00510
C DIVIDE HERE (IN PLACE OF RECIPROCAL MULTIPLY) TO AVOID WGS00520
```



C	POSSIBLE OVERFLOW		WGS00530
C			WGS00540
	DO 60 I=1,JW		WGS00550
60	W(K,I)=W(K,I)-SUM*V(I)		WGS00560
70	W(J,K)=SUM	Q EQ.(4.10) OF BOOK	WGS00570
100	CONTINUE	Q U(K,J) STORED IN W(J,K)	WGS00580
C			WGS00590
C	THE LOWER PART OF W IS U TRANSPOSE		WGS00600
C			WGS00610
	SUM=Z		WGS00620
	DO 105 K=1,JW		WGS00630
105	SUM=0W(K)*W(1,K)**2+SUM		WGS00640
	U(1)=SUM		WGS00650
	IJ=1		WGS00660
	DO 110 J=2,IW		WGS00670
	DO 110 I=1,J		WGS00680
	IJ=IJ+1		WGS00690
110	U(IJ)=W(J,I)		WGS00700
C			WGS00710
	RETURN		WGS00720
	END		WGS00730