JPL PUBLICATION 77-26, REVISION 1

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

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

$$\begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} & \\ & U_{22} & U_{23} & U_{24} & \text{etc.} \\ & & U_{33} & U_{34} & \\ 0 & & & U_{44} & \end{bmatrix} = \begin{bmatrix} U(1) & U(2) & U(4) & U(7) & \\ & U(3) & U(5) & U(8) & \text{etc.} \\ & & U(6) & U(9) & \\ 0 & & & U(10) & \end{bmatrix}$$

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(·,·) to a vector stored form, U(·) 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(·) of diagonal positions of a vector stored form, U(·), 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 nd 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 T 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

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

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

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

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.

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

<u>Remarks</u>: $C = \hat{R}^{-1} \hat{R}^{-T}$ = 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 $[R_o:z_o]$ using either subroutine THH, or subroutine TTHH when the a priori is diagonal or triangular, e.g.,

$$\left.\begin{matrix}[\hat{R}:\hat{z}] \\ [R_o:z_o]\end{matrix}\right\} \xrightarrow{\text{TTHH}} [\hat{R}:\hat{z}]^*$$

*The new result overwrites the old.

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

If a priori is given in the form of an information matrix, Λ, (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_o from Λ using INF2R;

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

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

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,... 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 $Ax = b - \nu$ with normal equations, $A^T A\hat{x}_o = A^T b$; $\Lambda = A^T A$ 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) $\left.\begin{matrix}[R:\hat{z}] \\ [A:z]\end{matrix}\right\} \xrightarrow{\text{THH}} [\hat{R}:\hat{z}]^*$

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 $[\hat{U}\text{-}\hat{D}:\hat{x}]$ with a priori U-D covariance factors and the initial estimate

b') Read the next [A:z] scalar measurement from the file

c') $\left.\begin{matrix}[\hat{U}\text{-}\hat{D}:\hat{x}] \\ [A:z]\end{matrix}\right\} \xrightarrow{\text{UDMEAS}} [\hat{U}\text{-}\hat{D}:\hat{x}]^*$

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, LO41, CU43, LO43 (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

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

NAME ORDER OF A

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

GM, CU41, LO41, CU43, LO43,

RHS*, $a_r, a_x, a_y,$

Remark: One could also have used subroutine COMBO, with the desired namelist as simply a_r, a_x, a_y. This would achieve the same A matrix form.

● $[A] \xrightarrow{THH} [R]$

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

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

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_r, a_x, or a_y.

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,

- $[\hat{R}:\hat{z}] \xrightarrow{\text{R2RA}} [R_A:z_A]$

INPUT NAMES:	OUTPUT NAMES:
a_r, a_x, a_y, x, y, z, v_x, v_y, v_z, GM	x, y, z, v_x, v_y, v_z, GM
CU41, LO41, CU43, LO43, RHS	CU41, LO41, CU43, LO43, RHS

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

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

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

- $[A\ :z_A] \xrightarrow{\text{THH}} [\hat{R}_A:\hat{z}_A]$ (Triangularizing)
- $[\hat{R}_A:\hat{z}_A] \xrightarrow{\text{R2RA}} [R_x:z_x]$ (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,

- $[\hat{R}:\hat{z}] \xrightarrow{\text{COMBO}} [A:z]$

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.

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

<u>Remark</u>: 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.

- $[\hat{R}:\hat{z}] \xrightarrow{\text{R2RA}} [R_x:z_x]$

<u>Remark</u>: 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,...,k.

<u>Remark</u>: 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 <u>Sensitivity, Perturbation, Computed Covariance and Consider Covariance Matrix Computation</u>

Suppose that one is given a SRIF array

$$\begin{array}{c} \begin{matrix} \underbrace{}_{N_x} & \underbrace{}_{N_y} & \underbrace{}_{1} \end{matrix} \\ \begin{bmatrix} R_x & R_{xy} & z_x \\ 0 & R_y & z_y \end{bmatrix} \begin{matrix} \}N_x \\ \}N_y \end{matrix} \end{array} \qquad \text{(II.5a)}$$

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

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

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

$$x - x_c = -R_x^{-1} R_{xy} y - R_x^{-1} \nu_x , \tag{II.5c}$$

and from this one obtains

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

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

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

where $\sigma_y(1),\ldots,\sigma_y(N_y)$ 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.)

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

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

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

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

---

† $\text{Pert} = \text{Sen}\, P_y^{1/2}$

†† The a priori estimate y_o of consider parameters is generally zero.

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

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

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

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

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}

$$[R_x^{-1}] \xrightarrow{\text{RI2COV}} [P_c] \qquad \text{computed covariance}$$

$$[R_x^{-1}\ :\ \text{Pert}] \xrightarrow{\text{RI2COV}} [P_{con}] \qquad \text{consider covariance}$$

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 ($z = a_x^T x + a_y^T y + \nu$) is processed.

$$Sen_j = Sen_{j-1} - K_j (a_x^T Sen_{j-1} + a_y^T)$$

where Sen_{j-1} is the sensitivity prior to processing this (j-th) measurement, and K_j is the Kalman gain vector.[†]

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

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

$[s_1, \ldots, s_{n_y}] = Sen_j$ (estimate error sensitivities)

then recursively compute

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

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

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

If $P_y(0) = U_y D_y U_y^T$, instead of $P_y(0) = \text{Diag}\ (\sigma_1^2, \ldots, \sigma_{n_y}^2)$, then in the U-D recursion one should replace the Sen_j columns by those of $Sen_j * U_y$ and σ_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

† $K = g/\alpha$ where g and α 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.

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

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 TTHH, assuming that subroutines R2A, THH 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

$$A_j x + B_j y_j = z_j - \nu_j \qquad j = 1,\ldots,N$$

where there is generally a priori information on the vector y_j variables. Rather than form a concatenated state vector composed of x, $y_1,\ldots,y_N$ which might create a problem involving exhorbitant amounts of storage and computation we solve the problem as follows. Apply subroutine THH 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.

$$N_{y_1}\left\{\begin{bmatrix} R_{y_1} & R_{y_1x} & z_{y_1} \\ 0 & R_{x_1} & z_{x_1} \end{bmatrix}\right.$$

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 $w_j \varepsilon N(0,Q_j)$. 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]),

$$R_j x_j = z_j - \nu_j \quad ; \quad \nu_j \varepsilon N(0,I)$$

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

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.

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

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

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

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

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

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

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

$$[\Phi_j U_j : G];\ \mathrm{Diag}\ (D_j, D^{(q)}) \xrightarrow{W\ G\ S} (\tilde{U}_{j+1} - \tilde{D}_{j+1})$$

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, $p_{j+1} = m\, p_j + w_j$, 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

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

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

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.

---

† in track and cross track accelerations

2. <u>COMBO</u> - (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.

<u>Example III.3</u>

$$
\begin{array}{cccc} \alpha & B & C & D \end{array}
\qquad\qquad
\begin{array}{cccccc} C & \overbrace{B \;\; E \;\; \alpha \;\; F \;\; D}^{\text{NAM2 list}} \end{array}
$$

$$
\begin{bmatrix} 1 & 2 & 4 & 7 \\ 0 & 3 & 5 & 8 \\ 0 & 0 & 6 & 9 \\ 0 & 0 & 0 & 10 \end{bmatrix}
\longrightarrow
\begin{bmatrix} 4 & 2 & 0 & 1 & 0 & 7 \\ 5 & 3 & 0 & 0 & 0 & 8 \\ 6 & 0 & 0 & 0 & 0 & 9 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}
$$

R-Vector stored A-Double subscripted

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. <u>COVRHO</u> - (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 (|RHO(IJ)|.LE.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 <u>negative</u> eigenvalue.

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

4. <u>COV2RI</u> - (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. <u>COV2UD</u> - (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. <u>C2C</u> - (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.

<u>Example III.4</u>

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

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 Λ 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.LE.N) is triangularized, and overwritten, by post-multiplying it by an implicitly defined orthogonal transformation, i.e.

$$[\ W\ ]T \longrightarrow [\ 0 \diagdown S\ ]$$

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. $W = [P^{1/2}_{computed} : Sen * P_y^{1/2}]$).

9. PERMUT

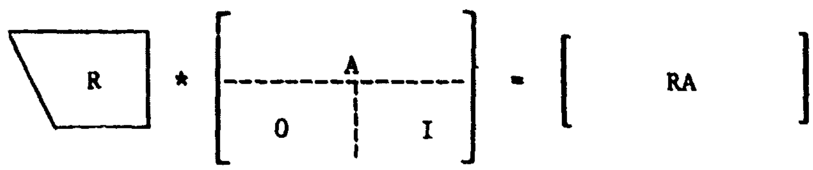
Reorders the columns of matrix A, storing the result back in A. This routine differs from A2A1 principally in that here the result overwrites A. PERMUT is especially useful in applications where storage is at a premium or where the problem is of a recursive nature.

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

$$[\ PHI\ ]\ \begin{bmatrix} \diagdown & U \\ & \diagdown \end{bmatrix} = [\ PHIU\ ]$$

The matrices PHI and PHIU are double subscripted, and U is vector subscripted with implicitly defined unit diagonal elements. It is not necessary to include trailing columns of zeros in the PHI matrix; they are accounted for implicitly. To economize on storage the output PHIU matrix can overwrite the input PHI matrix. For problems involving sparse PHI matrices it is more efficient to use the sparse matrix multiplication subroutine, SFU. When the last column of U contains the estimate, x, the last column of W represents the mapped elements of PHI*x. The principal use of this subroutine is the mapping of covariance U factors, where $P = UDU^T$, and estimates.

11. <u>RA</u> - (R(triangular) * A(rectangular))



Square root information matrix mapping involves matrix multiplication of the form indicated in the figure, i.e. with the bottom portion of A only implicitly defined as a partial identity matrix. Features of this subroutine are that the resulting RA matrix can overwrite the input A, and one can compute RA based on a trapezoidal input R matrix (i.e. only compute part of R*A).

12. <u>RANK1</u> - (U-D covariance factor rank 1 modification)

Computes updated U-D factors corresponding to a rank 1 matrix modification; i.e., given U-D, a scalar c, and vector v, $\bar{U}$ and $\bar{D}$ are computed so that $\bar{U}\,\bar{D}\,\bar{U}^T = U\,D\,U^T + c\,v\,v^T$. Both c and v are destroyed during the computation, and the resultant (vector stored) U-D array replaces the original one. Uses for this routine include (a) adding process noise effects to a U-D factored Kalman filter; (b) computing consider covariances (cf Section II.5); (c) computing "actual" covariance factors resulting from the use of suboptimal Kalman filter gains; and (d) adding measurements to a U-D factored information matrix.

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 $CNB/N \leq$ 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

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

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.

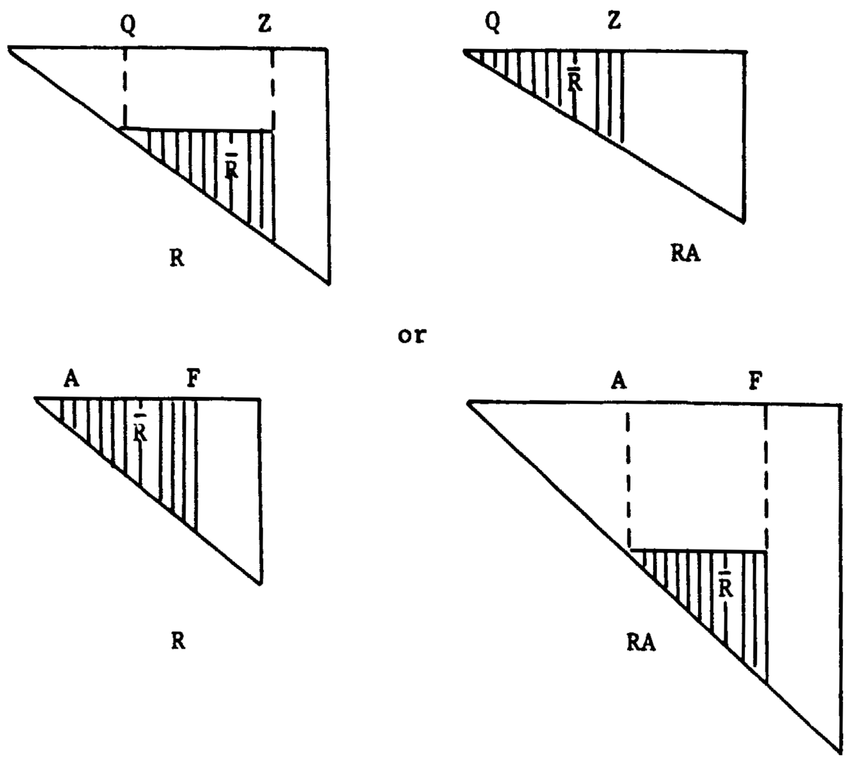
One trivial, but perhaps useful, application is to convert a vector stored matrix to a double subscripted form.† R2A is used most often when one wants to rearrange the columns of a SRIF array so that reduced order estimates, sensitivities, etc. can be obtained; or so that data sets containing different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

† see also the aside in the introduction

17. <u>R2RA</u> - (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.

<u>Examples III.6</u>



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))

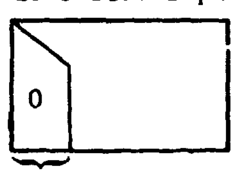
[Sparse F] 

= [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)

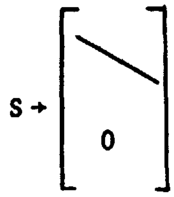
Implicitly defined Householder orthogonal transformations are used to triangularize an input two dimensional rectangular array, S(M,N). Computation can be reduced if S starts partially triangular;

S = 

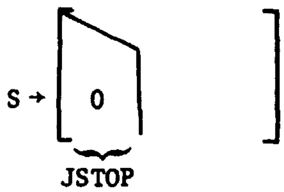
JSTART

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

when M.LT.N S → 

when M.GT.N 

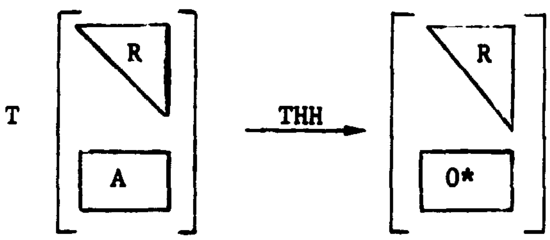
and finally when an intermediate form is desired



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. <u>THH</u> - (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, cf ref. [3].

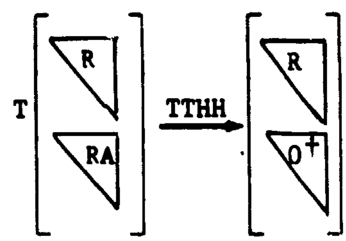


---

* 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.,

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

R-vector stored R-vector stored

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, $z = Ax + \nu$ 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)

$$\text{INPUT} \qquad\qquad\qquad\qquad \text{OUTPUT}$$

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

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 $R_y^{-1} = \text{Diag}(\sigma_y, \ldots, \sigma_{n_y})$ 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. <u>WGS</u> - (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,NAMA1)
```

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
NAMA1(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 NAMA1 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) = $1/\sqrt{\text{COV(I,I)}}$ if COV(I,I).GT.0 and 0. 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].

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

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.

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

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 THH. The orthogonolization 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].

$$R(input) = [R(output)]^T * [R(output)]$$

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.GE.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.

[PHI] [U] = [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.

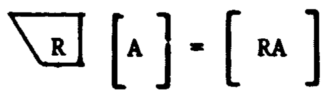
Functional Description

PHIU = PHI*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.




```
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.

$$A = \left[\begin{array}{cc} \multicolumn{2}{c}{\text{(Input)}} \\ \hline 0 & I \end{array}\right] \begin{array}{l} \}IA \\ \}N\text{-}IA \end{array}$$

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 o 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 paramters. 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. $p_{j+1} = \exp(-DT/\tau)* p_j + w_j$, $Rw = 1/\sigma_w$

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.

<u>Remarks and Restrictions</u>

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.

<u>Functional Description</u>

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 σ^* and S^* terms as follows:

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

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

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 σ^* 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 κ is the condition number of RIN, then CNB/N.LE.κ.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:

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

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

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

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

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†
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†.
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:

R

A	B	C	D	E
2	4	8	14	22
	6	10	16	24
		12	18	26
			20	28
				30

→

RA

C	D	E
12	18	26
	20	28
		30

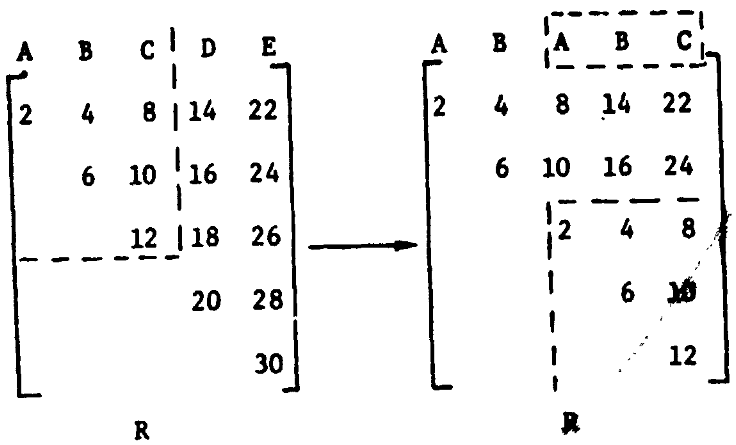
INPUT
NR = 5
NAM = 'A','B','C','D','E'
NRA = 0
NAMA(1) = 'C'
R

OUTPUT
NAMA = 'C', 'D', 'E'

†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. Incidently, 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;



INPUT
NR = 5
NAM = 'A','B','C','D','E'
NRA = 'C'
R

OUTPUT
NRA = 5
NAMA(3-5) = 'A','B','C'
RA

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 poriton 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, ≥ 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>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+1st 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 computaton. Not 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,IROW,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 implicity 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.LE.MAXFU, and IFU.GE. Max (IROW(K), K=1,...,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.

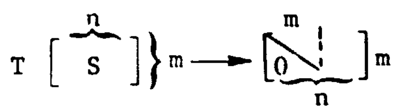
```
CALL TDHHT(S,MAXS,IRS,JCS,JSTART,JSTOP,V)
```

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

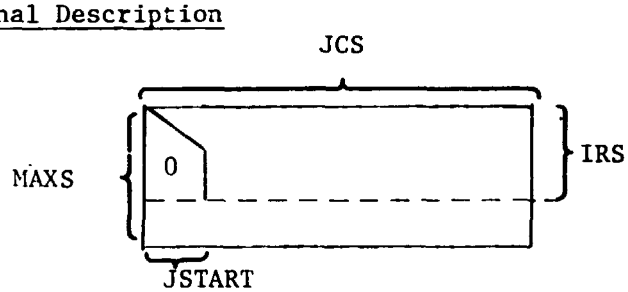
Remarks and Restrictions

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



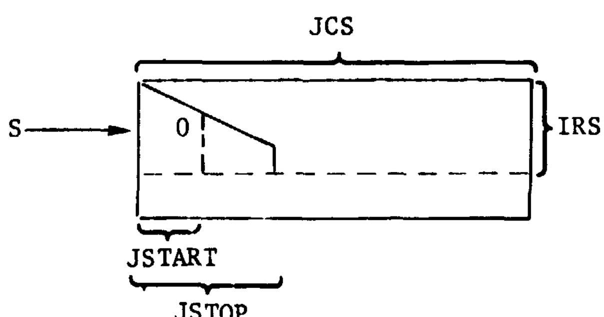
The code is set up so that it defaults to the largest possible upper triangularization.

Functional Description



Input S

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 $\nu + AX = z$. 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 $\left[\sum_i \lVert z_i - A_i X_{est} \rVert^2\right]^{1/2}$, summed over the old and new data. It also includes the a priori term

	$\|R_o X_{est} - z_o\|^2$. 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.

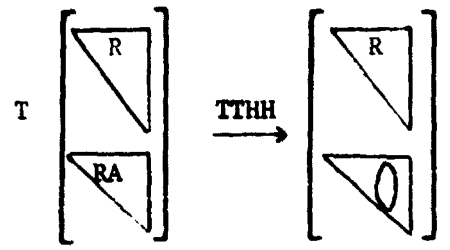
Functional Description

Assume for simplicity that NSTRT = 1. Then at step j, j = 1,...,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 jth. 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.



CALL TTHH (R,RA,N)

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

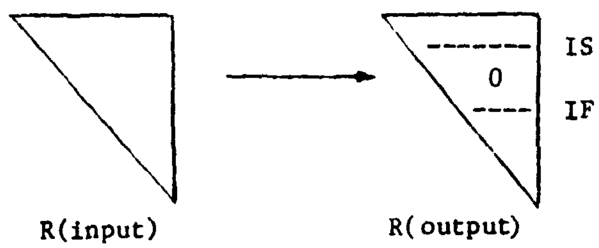
To zero out rows IS(Istart) to IF(Ifinal) of the vector stored upper triangular matrix R.

```
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 Φ 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, cf 1975 CONF. DEC. CONTROL paper. A scalar measurement $z = A^T x + \nu$ 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 algorithms 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, O.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 decompostiion 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, $\phi_1, \phi_2, \ldots, \phi_{IW}$, 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 ϕ 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 ϕ at step j of algorithm. The fact that the computed ϕ 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)
C
C        SUBROUTINE TO REARRANGE THE COLUMNS OF A(IR,LA), IN NAMA ORDER
C        AND PUT THE RESULT IN A1(IR,LA1) IN NAMA1 ORDER.  ZERO COLUMNS
C        ARE INSERTED IN A1 CORRESPONDING TO THE NEWLY DEFINED NAMES.
C
C        A(IR,LA)      INPUT RECTANGULAR MATRIX
C        IA            ROW DIMENSION OF A, IR.LE.IA
C        IR            NO. OF ROWS OF A THAT ARE TO BE REARRANGED
C        LA            NO. COLUMNS IN A, ALSO THE
C                      NO. OF PARAMETER NAMES ASSOCIATED WITH A
C        NAMA(LA)      PARAMETER NAMES ASSOCIATED WITH A
C        A1(IR,LA1)    OUTPUT RECTANGULAR MATRIX
C                      A AND A1 CANNOT SHARE COMMON STORAGE
C        IA1           ROW DIMENSION OF A1, IR.LE.IA1
C        LA1           NO. COLUMNS IN A1, ALSO THE
C                      NO. OF PARAMETER NAMES ASSOCIATED WITH A1
C        NAMA1(LA1)    INPUT LIST OF PARAMETER NAMES TO BE ASSOCIATED
C                      WITH THE OUTPUT MATRIX A1
C
C        COGNIZANT PERSONS:  G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976)
C
      DIMENSION A(IA,1), NAMA(1), A1(IA1,1),NAMA1(1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      ZERO=0.
      DO 100 J=1,LA1
        DO 60 I=1,LA
          IF (NAMA(I).EQ.NAMA1(J)) GO TO 80
   60     CONTINUE
        DO 70 K=1,IR
   70     A1(K,J)=ZERO        @ ZERO COL. CORRES. TO NEW NAME
        GO TO 100
   80   DO 90 K=1,IR
   90     A1(K,J)=A(K,I)      @ COPY COL. ASSOC. WITH OLD NAME
  100   CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE COMBO (R,L1,NAM1,L2,NAM2,A,IA,LA,NAMA)
C
C         TO REARRANGE A VECTOR STORED TRIANGULAR MATRIX AND STORE
C         THE RESULT IN MATRIX A.  THE DIFFERENCE BETWEEN THIS SUB-
C         ROUTINE AND R2A IS THAT THERE THE NAMELIST FOR A IS INPUT.
C         HERE IT IS DETERMINED BY COMBINING THE LIST FOR R WITH
C         A LIST OF DESIRED NAMES.
C
C        R(L1*(L1+1)/2)   INPUT VECTOR STORED UPPER TRIANGULAR MATRIX
C        L1               NO. OF PARAMETERS IN R (AND IN NAM1)
C        NAM1(L1)         NAMES ASSOCIATED WITH R
C        L2               NO. OF PARAMETERS IN NAM2
C        NAM2(L2)         PARAMETER NAMES THAT ARE TO BE COMBINED WITH R
C                         (NAM1 LIST). THESE NAMES MAY OR MAY NOT BE IN
C                         NAM1.
C        A(L1,LA)         OUTPUT ARRAY CONTAINING THE REARRANGED
C                         R MATRIX, L1.LE.IA.
C        IA               ROW DIMENSION OF A
C        LA               NO. OF PARAMETER NAMES IN NAMA, AND THE
C                         COLUMN DIMENSION OF A.  LA=L1+L2-NO. NAMES
C                         COMMON TO NAM1 AND NAM2.  LA IS COMPUTED AND
C                         OUTPUT.
C        NAMA(LA)         PARAMETER NAMES ASSOCIATED WITH THE OUTPUT A
C                         MATRIX.  CONSISTS OF NAMES IN NAM1 WHICH ARE
C                         NOT IN NAM2 FOLLOWED BY NAM2.
C
C        COGNIZANT PERSONS:  G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION  R(1), A(IA,1), NAM1(1), NAM2(1), NAMA(1)
C
      ZERO=0.0
      K=1
      DO 100 I=1,L1
        DO 50 J=1,L2
          IF (NAM1(I).EQ.NAM2(J)) GO TO 100
   50     CONTINUE
        NAMA(K)=NAM1(I)
      JJ=I*(I-1)/2
        DO 60 L=1,I
   60     A(L,K)=R(JJ+L)
      IF (I.EQ.L1) GO TO 80
      IP1 = I+1
      DO 70 L=IP1,L1
   70 A(L,K) = ZERO
   80   K=K+1
  100   CONTINUE
C              NAMES UNIQUE TO NAM1 ARE NOW IN NAMA
      DO 200 J=1,L2
        DO 150 I=1,L1
          IF (NAM2(J).EQ.NAM1(I)) GO TO 170
  150     CONTINUE
        NAMA(K)=NAM2(J)
        DO 160 L=1,L1
  160     A(L,K)=ZERO
```

```
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)
C
C     TO COMPUTE THE CORRELATION MATRIX RHO, FROM AN INPUT COVARIANCE
C     MATRIX COV. BOTH MATRICES ARE UPPER TRIANGULAR VECTOR STORED.
C     THE CORRELATION MATRIX RESULT CAN  OVERWRITE THE INPUT COVARIANCE
C     COV(N*(N+1)/2)  INPUT VECTOR STORED POSITIVE SEMI-DEFINITE
C                     COVARIANCE MATRIX
C     N               NUMBER OF PARAMETERS, N.GE.1
C     RHO(N(N+1)/2)   OUTPUT VECTOR STORED CORRELATION MATRIX,
C                     RHO(IJ)=COV(IJ)/(SIGMA(I)*SIGMA(J))
C     V(N)            WORK VECTOR
C
C         COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD   (JPL,FEB.1978)
C
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION  COV(1), RHO(1), V(1)
C
      ONE=1.D0
      Z=0.D0
C
      JJ=0
      DO 10 J=1,N
        JJ=JJ+J
        V(J)=Z
        IF (COV(JJ).GT.Z) V(J)=ONE/ SQRT(COV(JJ))
C
C     **** SOME MACHINES REQUIRE DSQRT FOR DOUBLE PRECISION
C
   10   CONTINUE
C
      IJ=0
      DO 20 J=1,N
        S=V(J)
        DO 20 I=1,J
          IJ=IJ+1
   20     RHO(IJ)=COV(IJ)*S*V(I)
      RETURN
      END
```

```
      SUBROUTINE COV2RI(U,N)
C
C        TO CONSTRUCT THE UPPER TRIANGULAR CHOLESKY FACTOR OF A
C        POSITIVE SEMI-DEFINITE MATRIX. BOTH THE INPUT COVARIANCE
C        AND THE OUTPUT CHOLESKY FACTOR (SQUARE ROOT) ARE VECTOR
C        STORED. THE OUTPUT OVERWRITES THE INPUT.
C        COVARIANCE(INPUT)=U*U**T   (U IS OUTPUT).
C
C        IF THE INPUT COVARIANCE IS SINGULAR THE OUTPUT FACTOR HAS
C        ZERO COLUMNS.
C
C        U(N*(N+1)/2)   CONTAINS THE INPUT VECTOR STORED COVARIANCE
C                       MATRIX (ASSUMED POSITIVE DEFINITE) AND ON OUTPUT
C                       IT CONTAINS THE UPPER TRIANGULAR SQUARE ROOT
C                       FACTOR.
C        N              DIMENSION OF THE MATRICES INVOLVED
C
C        COGNIZANT PERSONS:  G.J.BIERMAN/M.W.NEAD (JPL, FEB. 1977)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION U(1)
C
      ZERO=0.0
      ONE=1.
      JJ=N*(N+1)/2
C
      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)
C
         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
    3          U(KK+I)=U(KK+I)-S*U(JJN+I)       @ KK+I=(I,K)
    4        KK=KK+K
    5 JJ=JJN
      IF (U(1).LT.ZERO) U(1)=ZERO
      U(1)= SQRT(U(1))
C
      RETURN
      END
```

```
      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
C        COGNIZANT PERSONS:  G.J.BIERMAN/R.A.JACOBSON (JPL, FEB. 1977)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION U(1)
C
      Z=0.D0
      ONE=1.D0
      NONE=1
C
      JJ=N*(N+1)/2
      NP2=N+2
      DO 50 L=2,N
        J=NP2-L
        ALPHA=Z
        IF (U(JJ).GE.Z) GO TO 10
        WRITE (6,100) J,U(JJ)
        U(JJ)=Z
   10   IF (U(JJ).GT.Z) ALPHA=ONE/U(JJ)
        JJ=JJ-J
        KK=0
        KJ=JJ
        JM1=J-1
        DO 40 K=1,JM1
          KJ=KJ+1
          BETA=U(KJ)
          U(KJ)=ALPHA*U(KJ)
          IJ=JJ
          IK=KK
          DO 30 I=1,K
            IK=IK+1
            IJ=IJ+1
   30       U(IK)=U(IK)-BETA*U(IJ)
   40     KK=KK+K
   50   CONTINUE
      IF (U(1).GE.Z) GO TO 60
      WRITE (6,100) NONE, U(1)
      U(1)=Z
   60 RETURN
C
```

```
100 FORMAT (1H0,20X,' AT STEP',I4,'DIAGONAL ENTRY =',F12.4)           COV2U560
    END                                                                COV2U570
```

```
      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.BIERMAN/M.W.NEAD (JPL, SEPT. 1976)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION  C(IC,1), NAM1(1), NAM2(1)
C
      ZERO=0.
      L=MAX(L1,L2)
      IF (L.LE.L1) GO TO 5
      NM=L1+1
      DO 1 K=NM,L
    1   NAM1(K)= ZERO          @ ZERO REMAINING NAM1 LOCNS
    5 DO 90 J=1,L2
        DO 10 I=1,L
          IF (NAM1(I).EQ.NAM2(J)) GO TO 30
   10     CONTINUE
        GO TO 90
   30   IF (I.EQ.J) GO TO 90
        DO 40 K=1,L
          H=C(K,J)             @ INTERCHANGE COLUMNS I AND J
          C(K,J)=C(K,I)
   40     C(K,I)=H
        DO 80 K=1,L
          H=C(J,K)             @ INTERCHANGE ROWS I AND J
          C(J,K)=C(I,K)
   80     C(I,K)=H
          NM=NAM1(I)           @ INTERCHANGE LABELS I AND J
          NAM1(I)=NAM1(J)
          NAM1(J)=NM
   90   CONTINUE
C
C         FIND NAM2 NAMES NOT IN NAM1 AND SET CORRESPONDING ROWS AND
C         COLUMNS TO ZERO
C
      DO 120 J=1,L2
        DO 100 I=1,L
          IF (NAM1(I).EQ.NAM2(J)) GO TO 120
  100     CONTINUE
        DO 110 K=1,L2
          C(J,K)=ZERO
```

```
  110      C(K,J)=ZERO
  120   CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE HHPOST(S,W,MROW,NROW,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 SQRT
C                        COVARIANCE MATRIX
C     W(NROW,NCOL)       INPUT RECTANGULAR SQRT 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.NROW)
C     V(NCOL)            WORK VECTOR
C
C     COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD     (JPL, NOV.1977)
C
      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)
      DOUBLE PRECISION  SUM,BETA
      DIMENSION  S(1),W(MROW,NCOL),V(NCOL)
C
      ZERO=0.D0
      ONE=1.D0
C
      JCOL=NCOL
      NSYM=NROW*(NROW+1)/2
      JC=NROW+2
      DO 150 L=2,NROW
        IROW=JC-L
        SUM=ZERO
        DO 100 K=1,JCOL
          V(K)=W(IROW,K)
  100     SUM=SUM+V(K)**2
        SUM=DSQRT(SUM)
        IF (V(JCOL).GT.ZERO) SUM=-SUM      @ DIAGONAL ENTRY (JCOL,JCOL)
C
        S(NSYM)=SUM
        NSYM=NSYM-IROW
        V(JCOL)=V(JCOL)-SUM
        IF (SUM.NE.ZERO) BETA=-ONE/(SUM*V(JCOL))
C         T(ORTHOG. TRANS.)=I-BETA*V*V**T
        IROWM1=IROW-1
        JCOLM1=JCOL-1
        DO 140 I=1,IROWM1
          SUM=ZERO
          DO 110 K=1,JCOL
  110       SUM=SUM+V(K)*W(I,K)
            SUM=BETA*SUM
          DO 120 K=1,JCOLM1
  120       W(I,K)=W(I,K)-SUM*V(K)
  140     S(NSYM+I)=W(I,IROW)-SUM*V(IROW)
  150   JCOL=JCOLM1
C
      JC=NCOL-NROW+1
      SUM=ZERO
```

```
      DO 160 J=1,JC                                                     HHPOS560
  160   SUM=SUM+W(I,J)**2                                               HHPOS570
      S(1)=DSQRT(SUM)                                                   HHPOS580
C                                                                       HHPOS590
      RETURN                                                            HHPOS600
      END                                                               HHPOS610
```

```
C
      SUBROUTINE INF2R (R,N)
C
C         TO CHOLESKY FACTOR AN INFORMATION MATRIX
C
C     COMPUTES A LOWER TRIANGULAR VECTOR STORED CHOLESKY 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 CHOLESKY FACTOR. IF THE
C                   INPUT MATRIX IS SINGULAR THE OUTPUT MATRIX WILL
C                   HAVE ZERO DIAGONAL ENTRIES
C     N             DIMENSION OF MATRICES INVOLVED, N.GE.2
C
C     COGNIZANT PERSON: G.J.BIERMAN/M.W.NEAD      (JPL,FEB,1977)
C
      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)
C
      DIMENSION R(1)
C
      Z=0.D0
      ONE=1.D0
      JJ=0
      NN=N*(N+1)/2
      NM1=N-1
      DO 10 J=1,NM1
        JJ=JJ+J                               @ JJ=(J,J)
        IF (R(JJ).GE.Z)  GO TO 5
        WRITE (6,20) J,R(JJ)
        R(JJ)=Z
    5   R(JJ)= SQRT(R(JJ))
C
C **** SOME MACHINES REQUIRE DSQRT FOR DOUBLE PRECISION
C
        ALPHA=Z
        IF (R(JJ).GT.Z) ALPHA=ONE/R(JJ)
        JK=NN+J                               @ JK=(J,K)
        JP1=J+1
        JIS=JK                                @ JIS=(J,I) START
        NPJP1=N+JP1
        DO 10 L=JP1,N
          K=NPJP1-L
          JK=JK-K
          R(JK)=ALPHA*R(JK)
          BETA=R(JK)
          KI=NN+K
          JI=JIS
          NPK=N+K
          DO 10 M=K,N
            I=NPK-M
            KI=KI-I
            JI=JI-I
```

```
10        R(KI)=R(KI)-R(JI)*BETA
C
      IF (R(NN).GE.Z) GO TO 15
      WRITE (6,20) N,R(NN)
      R(NN)=Z
   15 R(NN)= SQRT(R(NN))
      RETURN
C
   20 FORMAT (1H0,20X,' AT STEP',I4,'DIAGONAL ENTRY =',E12.4,
     1 ', IT IS RESET TO ZERO')
      END
```

```
      SUBROUTINE PERMUT (A,IA,IR,L1,NAM1,L2,NAM2)
C
C        SUBROUTINE TO REARRANGE PARAMETERS OF A(IR,L1), NAM1 ORDER
C        TO A(IR,L2), NAM2 ORDER.  ZERO COLUMNS ARE INSERTED
C        CORRESPONDING TO THE NEWLY DEFINED NAMES.
C
C        A(IR,L)    INPUT RECTANGULAR MATRIX, L=MAX(L1,L2)
C        IA         ROW DIMENSION OF A, IA.GE.IR
C        IR         NUMBER OF ROWS OF A THAT ARE TO BE REARRANGED
C        L1         NUMBER OF PARAMETER NAMES ASSOCIATED WITH THE INPUT
C                   A MATRIX
C        NAM1(L)    PARAMETER NAMES ASSOCIATED WITH A ON INPUT
C                   (ONLY THE FIRST L1 ENTRIES APPLY TO THE INPUT A)
C                   NAM1 IS DESTROYED BY PERMUT
C        L2         NUMBER OF PARAMETER NAMES ASSOCIATED WITH THE OUTPUT
C                   A MATRIX
C        NAM2       PARAMETER NAMES ASSOCIATED WITH THE OUTPUT A
C
C        COGNIZANT PERSONS:  G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(IA,1), NAM1(1), NAM2(1)
C
      ZERO=0.
      L=MAX(L1,L2)
      IF (L.LE.L1) GO TO 50
      NM=L1+1
      DO 40 K=NM,L
   40   NAM1(K)=0           @ ZERO REMAINING NAM1 LOCS
   50 DO 100 J=1,L2
        DO 60 I=1,L
          IF (NAM1(I).EQ.NAM2(J)) GO TO 65
   60     CONTINUE
      GO TO 100
   65 CONTINUE
        IF (I.EQ.J) GO TO 100
        DO 70 K=1,IR        @ INTERCHANGE COLS I AND J
          W=A(K,J)
          A(K,J)=A(K,I)
   70     A(K,I)=W
          NM=NAM1(I)        @ INTERCHANGE I AND J COL. LABELS
          NAM1(I)=NAM1(J)
          NAM1(J)=NM
  100   CONTINUE
C                    REPEAT TO FILL NEW COLS
        DO 200 J=1,L2
          DO 160 I=1,L
            IF (NAM1(I).EQ.NAM2(J)) GO TO 200
  160       CONTINUE
          DO 170 K=1,IR
  170       A(K,J)=ZERO
  200     CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE PHIU(PHI,MAXPHI,IRPHI,ICPHI,U,N,PHIU,MPHIU)
C
C     THIS SUBROUTINE COMPUTES W=PHI*U WHERE PHI IS A RECTANGULAR MATRIX
C     WITH IMPLICITLY DEFINED COLUMNS OF TRAILING ZEROS AND U IS A
C     VECTOR STORED UPPER TRIANGULAR MATRIX
C
C     PHI(IRPHI,ICPHI) INPUT RECTANGULAR MATRIX, IRPHI.LE.MAXPHI
C     MAXPHI           ROW DIMENSION OF PHI
C     IRPHI            NO. ROWS OF PHI
C     ICPHI            NO. COLS OF PHI
C     U(N*(N+1)/2)     UPPER TRIANGULAR VECTOR STORED MATRIX
C     N                DIMENSION OF U MATRIX (ICPHI.LE.N)
C     PHIU(IRPHI,N)    OUTPUT, RESULT OF PHI*U, PHIU CAN
C                      OVERWRITE PHI
C     MPHIU            ROW DIMENSION OF PHIU
C
C       COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD   (JPL, FEB.1978)
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION PHI(MAXPHI,1),U(1),PHIU(MPHIU,1)
      DOUBLE PRECISION SUM
C
      DO 10 I=1,IRPHI
   10 PHIU(I,1)=PHI(I,1)
C
      NP2=N+2
      KJS=N*(N+1)/2
      DO 40 L=2,N
        J=NP2-L
        KJS=KJS-J
      JM1=J-1
      DO 30 I=1,IRPHI
      SUM=PHI(I,J)
      IF (J.LE.ICPHI) GO TO 15
      SUM=0.D0
      JM1=ICPHI
   15 DO 20 K=1,JM1
   20 SUM=SUM+PHI(I,K)*U(KJS+K)
   30 PHIU(I,J)=SUM
   40    CONTINUE
C
      RETURN
      END
```

```
      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
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION  R(1),A(MAXA,1),RA(MAXRA,1)
      DOUBLE PRECISION SUM
C
      IJ=IA*(IA+1)/2               @ IJ=JJ(IA)
C
      DO 30 J=1,JA
        II=0                       @ TO BE REMOVED IF JJ(I) IS USED
        DO 20 I=1,NRA
          II=II+I                  @ II=(I,I)=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=JJ(J)
C
          SUM=0.D0
          IF (I.GT.IA) GO TO 15
          IK=II
          DO 10 K=I,IA
            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
   20   RA(I,J)=SUM
   30   IF (J.GT.IA) IJ=IJ+J       @ IJ=JJ(J)
C
      RETURN
      END
```

```
      SUBROUTINE RANK1 (UIN,UOUT,N,C,V)
C
C     STABLE  U-D FACTOR RANK 1 UPDATE
C
C               (UOUT)*DOUT*(UOUT)**T=(UIN)*DIN*(UIN)**T+C*V*V**T
C
C     UIN(N*(N+1)/2)  INPUT VECTOR STORED POSITIVE SEMI-DEFINITE U-D
C                     ARRAY, WITH D ELEMENTS STORED ON THE DIAGONAL
C     UOUT(N*(N+1)/2) OUTPUT VECTOR STORED POSITIVE (POSSIBLY) SEMI-
C                     DEFINITE U-D RESULT. UOUT=UIN IS PERMITTED
C     N               MATRIX DIMENSION, N.GE.2
C     C               INPUT SCALAR. SHOULD BE NON-NEGATIVE
C                     C IS DESTROYED DURING THE PROCESS
C     V(N)            INPUT VECTOR FOR RANK ONE MODIFICATION.
C                     V IS DESTROYED DURING THE PROCESS
C
C     COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD   (JPL,SEPT.1977)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION     UIN(1),   UOUT(1),  V(1)
      DOUBLE PRECISION ALPHA, BETA, S, D, EPS, TST
C
      DATA EPS/0.D0/, TST/.0625D0/
C      IN SINGLE PRECISION EPSILON IS MACHINE ACCURACY
C
C     TST=1/16 IS USED FOR RANK1 ALGORITHM SWITCHING
C
      Z=0.D0
      JJ=N*(N+1)/2
      IF (C.GT.Z) GO TO 4
      DO 1 J=1,JJ
    1   UOUT(J)=UIN(J)
      RETURN
C
    4 NP2=N+2
      DO 70 L=2,N
        J=NP2-L
        S=V(J)
        BETA=C*S
        D=UIN(JJ)+BETA*S
        IF (D.GT.EPS) GO TO 30
        IF (D.GE.Z) GO TO 10
    5   WRITE (6,100)
        RETURN
   10   JJ=JJ-J
        WRITE (6,110)
        DO 20 K=1,J
   20   UOUT(JJ+K)=Z
        GO TO 70
   30   BETA=BETA/D
        ALPHA=UIN(JJ)/D
        C=ALPHA*C
        UOUT(JJ)=D
        JJ=JJ-J
        JM1=J-1
```

```
      IF (ALPHA.LT.TST) GO TO 50
      DO 40 I=1,JM1
        V(I)=V(I)-S*UIN(JJ+I)
   40   UOUT(JJ+I)=BETA*V(I)+UIN(JJ+I)
      GO TO 70
   50 DO 60 I=1,JM1
        D=V(I)-S*UIN(JJ+I)
        UOUT(JJ+I)=ALPHA*UIN(JJ+I)+BETA*V(I)
   60   V(I)=D
   70 CONTINUE
C
      UOUT(1)=UIN(1)+C*V(1)**2
      RETURN
C
  100 FORMAT (1H0,10X,'* * * ERROR RETURN DUE TO A COMPUTED NEGATIVE COM
     1PUTED DIAGONAL IN RANK1 * * *')
  110 FORMAT (1H0,10X,'* * * NOTE: U-D RESULT IS SINGULAR * * *')
      END
```

```
      SUBROUTINE RCOLRD(S,MAXS,IRS,JCS,NPSTRT,NP,EM,RW,ZW,V,SGSTRT)
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
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION S(MAXS,JCS),EM(NP),RW(NP),ZW(NP), V(IRS),SGSTAR(1)
      DOUBLE PRECISION ALPHA,SIGMA,BETA,GAMMA
C
      ZERO=0.D0
      ONE=1.D0
      NPCOL=NPSTRT   @ COL NO OF COLORED NOISE TERM TO BE OPERATED ON
C
      DO 70 JCOLRD=1,NP
        ALPHA=-RW(JCOLRD)*EM(JCOLRD)
        SIGMA=ALPHA**2
        DO 10 K=1,IRS
          V(K)=S(K,NPCOL)   @ FIRST IRS ELEMENTS OF HOUSEHOLDER
```

```
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
```

```
      SUBROUTINE RINCON (RIN,N,ROUT,CNB)
C
C     TO COMPUTE THE INVERSE OF THE UPPER TRIANGULAR VECTOR STORED
C     INPUT MATRIX RIN AND STORE THE RESULT IN ROUT. (RIN=ROUT IS
C     PERMITTED) AND TO COMPUTE A CONDITION NUMBER ESTIMATE.
C     CNB=FROB.NORM(R)*FROB.NORM(R**-1).
C     THE FROBENIUS NORM IS THE SQUARE ROOT OF THE SUM OF SQUARES
C     OF THE ELEMENTS. THIS CONDITION NUMBER BOUND IS USED AS
C     AN UPPER BOUND AND IT ACTS AS A LOWER BOUND ON THE ACTUAL
C     CONDITION NUMBER OF THE PROBLEM. (SEE THE BOOK 'SOLVING LEAST
C     SQUARES', BY LAWSON AND HANSON)
C
C     IF RIN IS SINGULAR, RINCON COMPUTES THE INVERSE TO THE LEFT OF
C     THE FIRST ZERO DIAGONAL. A MESSAGE IS PRINTED AND THE CONDITION
C     NUMBER BOUND COMPUTATION IS ABORTED.
C
C     RIN(N*(N+1)/2)  INPUT VECTOR STORED UPPER TRIANGULAR MATRIX
C     N               DIMENSION OF R MATRICES, N.GE.2
C     ROUT(N*(N+1)/2) OUTPUT VECTOR STORED UPPER TRIANGULAR MATRIX
C                     INVERSE (RIN=ROUT IS PERMITTED)
C     CNB             CONDITION NUMBER BOUND. IF C IS THE CONDITION
C                     NUMBER OF RIN, THEN CNB/N.LE.C.LE.CNB
C
C     COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD    (JPL,FEB.1978)
C
C
      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)
      DOUBLE PRECISION RNM,DINV,SUM,RNMOUT
      DIMENSION RIN(1), ROUT(1)
C
      Z=0.D0
      ONE=1.D0
      NTOT=N*(N+1)/2
C
      RNM=Z
      DO 10 J=1,NTOT
   10   RNM=RNM+RIN(J)**2
C
C      REPLACE CALL UTINV (RIN,N,ROUT) BY UTINV CODE
C
      IF (RIN(1).NE.Z) GO TO 20
      J=1
      WRITE (6,100) J,J
      RETURN
C
   20 ROUT(1)=ONE/RIN(1)
C
      JJ=1
      DO 50 J=2,N
        JJOLD=JJ
        JJ=JJ+J
        IF (RIN(JJ).NE.Z) GO TO 30
        WRITE (6,100) J,J
        RETURN
C
```

```
 30    DINV=ONE/RIN(JJ)
       ROUT(JJ)=DINV
       II=0
       IK=1
       JM1=J-1
       DO 50 I=1,JM1
         II=II+I
         IK=II
         SUM=Z
         DO 40 K=I,JM1
           SUM=SUM+ROUT(IK)*RIN(JJOLD+K)
 40        IK=IK+K
 50      ROUT(JJOLD+I)=-SUM*DINV
C
C
C
      RNMOUT=Z
      DO 60 J=1,NTOT
 60     RNMOUT=RNMOUT+ROUT(J)**2
C
      RNM=DSQRT(RNM*RNMOUT)
      CNB=RNM
C
      WRITE (6,110) RNM
      RETURN
C
100 FORMAT (1H0,10X,'* * * MATRIX INVERSE COMPUTED ONLY UP TO BUT NOT
   1INCLUDING COLUMN',I4,' * * * MATRIX DIAGONAL ',I4,' IS ZERO * * *'
   2)
110 FORMAT(1H0,5X,'CONDITION NUMBER BOUND=',D18.10,2X,'CNB/N.LE.CONDIT
   1ION NUMBER.LE.CNB',/)
      END
```

```
      SUBROUTINE RI2COV (RINV,N,SIG,COVOUT,KROW,KCOL)
C
C        TO COMPUTE THE COVARIANCE MATRIX AND/OR THE STANDARD DEVIATIONS
C        OF A VECTOR STORED UPPER TRIANGULAR SQUARE ROOT COVARIANCE
C        MATRIX.  THE OUTPUT COVARIANCE MATRIX IS ALSO VECTOR STORED.
C
C        RINV(N*(N+1)/2)     INPUT VECTOR STORED UPPER TRIANGULAR
C                            COVARIANCE SQUARE ROOT. (RINV=RINVERSE
C                            IS THE INVERSE OF THE SRIF MATRIX)
C        N                   DIMENSION OF THE RINV MATRIX, N.GE.2
C        SIG(N)              OUTPUT VECTOR OF STANDARD DEVIATIONS
C        COVOUT(N*(N+1)/2)   OUTPUT VECTOR STORED COVARIANCE MATRIX
C                            (COVOUT = RINV IS ALLOWED)
C        KROW  .GT.0         COMPUTES THE COVARIANCE AND SIGMAS
C                            CORRESPONDING TO THE FIRST KROW VARIABLES
C                            OF THE RINV MATRIX.
C              .LT.0         COMPUTES ONLY THE SIGMAS OF THE FIRST KROW
C                            VARIABLES OF THE RINV MATRIX.
C                            RINV.
C              .EQ.0         NO COVARIANCE, BUT ALL SIGMAS (E.G. USE
C                            N ROWS OF RINV).
C        KCOL                NO. OF COLUMNS OF COVOUT THAT ARE COMPUTED
C                            IF KCOL.LE.0 THEN KCOL=KROW. IF KROW.LE.0
C                            THIS INPUT IS IGNORED.
C
C        COGNIZANT PERSONS:  G.J.BIERMAN/M.W.NEAD (JPL, MARCH 1978)
C
      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)
      DOUBLE PRECISION SUM
      DIMENSION RINV(1), SIG(1), COVOUT(1)
C
      ZERO=0.D0
      LIM=N
      KKOL=KCOL
      IF (KKOL.LE.0) KKOL=KROW
      IF (KROW.NE.0) LIM=IABS(KROW)
C         *** COMPUTE SIGMAS
      IKS=0
      DO 2 J=1,LIM
        IKS=IKS+J
        SUM=ZERO
        IK=IKS
        DO 1 K=J,N
          SUM=SUM+RINV(IK)**2
    1     IK=IK+K
    2   SIG(J)=DSQRT(SUM)
C
      IF (KROW.LE.0) RETURN
C         *** COMPUTE COVARIANCE
      JJ=0
      NM1=LIM
      IF (KROW.EQ.N) NM1=N-1
      DO 10 J=1,NM1
        JJ=JJ+J
        COVOUT(JJ)=SIG(J)**2
```

```
      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)*RINV(IJK)
    5     IK=IK+K
        COVOUT(IJS)=SUM
   10   IJS=IJS+I
      IF (KROW.EQ.N) COVOUT(JJ+N)=SIG(N)**2
C
      RETURN
      END
```

```
      SUBROUTINE R2A(R,LR,NAMR,A,IA,LA,NAMA)
C
C        TO PLACE THE TRIANGULAR VECTOR STORED MATRIX R INTO THE
C        MATRIX A AND TO ARRANGE THE COLUMNS TO MATCH THE DESIRED
C        NAMA PARAMETER LIST.  NAMES IN THE NAMA LIST THAT DO NOT
C        CORRESPOND TO ANY NAME IN NAMR HAVE ZERO ENTRIES IN THE
C        CORRESPONDING A COLUMN.
C
C        R(LR*(LR+1)/2)  INPUT UPPER TRIANGULAR VECTOR STORED ARRAY
C        LR              DIMENSION OF R
C        NAMR(L)         PARAMETER NAMES ASSOCIATED WITH R
C        A(LR,LA)        MATRIX TO HOUSE THE REARRANGED R MATRIX
C        IA              ROW DIMENSION OF A, IA.GE.LR
C        LA              NO. OF PARAMETER NAMES ASSOCIATED WITH THE
C                        OUTPUT A MATRIX
C        NAMA(LA)        PARAMETER NAMES FOR THE OUTPUT A MATRIX
C
C        COGNIZANT PERSONS:  G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION R(1),NAMR(1),A(IA,1),NAMA(1)
C
       ZERO=0.
      DO 5 J=1,LA
      DO 5 K=1,LR
    5 A(K,J)=ZERO            @ ZERO A(LR,LA)
      DO 40 J=1,LA
        DO 10 I=1,LR
          IF (NAMR(I).EQ.NAMA(J)) GO TO 20
   10     CONTINUE
        GO TO 40
   20 JJ=I*(I-1)/2
        DO 30 K=1,I
   30     A(K,J)=R(JJ+K)
   40   CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE R2RA (R,NR,NAM,RA,NRA,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                            THIS INPUT NAMELIST IS DESTROYED
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 NAMELIST.
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        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 NAMELIST 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
      IMPLICIT DOUBLE PRECISION   (A-H,O-Z)
      DIMENSION     R(1),RA(1), NAM(1), NAMA(1)
      LOGICAL       IS
C
      IS=.FALSE.
      LOCN=NAMA(1)
C        IS=FALSE CORRESPONDS TO MOVING UPPER LFT. CORNER OF R TO
C        LOWER RT. CORNER OF RA
      IF (NRA.EQ.0) GO TO 1
      LOCN=NRA
      IS=.TRUE.
C        IS=TRUE CORRESPONDS TO MOVING LOWER LFT. CORNER OF R TO
C        UPPER RT. CORNER OF RA
    1 DO 3 I=1,NR
       IF (NAM(I).EQ.LOCN) GO TO 4
    3 CONTINUE
       WRITE (6,100)
  100 FORMAT (1H0,20X,'NAMA(1) NOT IN NAMELIST OF R MATRIX')
```

```
      RETURN
C
    4 K=I
      KM1=K-1
      IF (IS) GO TO 15
C
      IJS=K*(K+1)/2-1
      NRA=NR-K+1
      IJA=0
      KOLA=0
      DO 10 KOL=K,NR
        KOLA=KOLA+1
      NAMA(KOL-KM1)=NAM(KOL)
        DO 5 IR=1,KOLA
          IJA=IJA+1
    5     RA(IJA)=R(IJS+IR)
   10   IJS=IJS+KOL
      RETURN
C
   15 IJ=K*(K+1)/2
      IJA=NR*(NR+1)/2
      L=NR-KM1
      KOL=K
        DO 25 KOLA=NR,L,-1
        IJS=IJA
        NAMA(KOLA)=NAM(KOL)
          DO 20 IR=KOLA,L,-1
            RA(IJS)=R(IJ)
            IJS=IJS-1
   20       IJ=IJ-1
        IJA=IJA-KOLA
   25   KOL=KOL-1
      NRA=NR
C
      RETURN
      END
```

```
      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 (RIN=ROUT IS PERMITTED)
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     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     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.NEAD  (JPL, FEB.1978)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION RIN(1), ROUT(1)
C
      ONE= 1.D0
      NP1= IS + IABS(N)
      JJ=1                                 @ INITIALIZE DIAGONAL INDEX
      IDIMR= NP1*(NP1 +1)/2
      IF (IS.EQ.0) GO TO 5
      RNN=RIN(IDIMR)
      RIN(IDIMR)=-ONE
C
    5 IF (N.LT.0) GO TO 30
      CALL RINCON(RIN,NP1,ROUT,CNB)
      ROUT(1)= ROUT(1)**2
      DO 20 J=2,N
      S=ONE/ROUT(JJ+J)
      ROUT(JJ+J)= ROUT(JJ+J)**2
      JM1=J-1
      DO 10 I=1,JM1
   10 ROUT(JJ+I)= ROUT(JJ+I)*S
   20 JJ=JJ+ J
      GO TO 70
C
   30 NN=-N                          @ NN=NEGATIVE N
      ROUT(1)= SQRT(RIN(1))
C
C     *** SOME MACHINES REQUIRE DSQRT FOR DOUBLE PRECISION
C
      DO 50 J=2,NN
      ROUT(JJ+J)= SQRT(RIN(JJ+J))
```

```
      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
```

```
      SUBROUTINE SFU(FEL,IROW,JCOL,NF,U,N,FU,MAXFU,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 FU MATRIX
C     IFU              NUMBER OF ROWS IN FU,
C                      (IFU.LE.MAXFU.AND.IFU.GE.MAX(IROW(K)), K=1,...,NF,
C                      I.E. FU MUST HAVE AT LEAST AS MANY ROWS AS DOES F.
C                      ADDITIONAL ROWS OF FU 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.BIERMAN/M.W.NEAD     (JPL, FEB.1978)
C
      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)
      DIMENSION  FEL(NF),U(1),FU(MAXFU,N),IROW(NF),JCOL(NF),JDIAG(N)
C
      ZERO=0.D0
C * * * * INITIALIZE FU
      DO 10 J=1,N
        DO 10 I=1,IFU
   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
      DO 30 NEL=1,NF
C         NEL REPRESENTS THE ELEMENT NUMBER OF THE F MATRIX
        I=IROW(NEL)
         J=JCOL(NEL)
        FIJ=FEL(NEL)
        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

        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
        JP1=J+1
```

```
      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
```

```
      SUBROUTINE TDHHT(S,MAXS,IRS,JCS,JSTART,JSTOP,V)
C
C         TDHHT TRANSFORMS A RECTANGULAR DOUBLE SUBSCRIPTED 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.BIERMAN, 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.BIERMAN/M.W.NEAD  (JPL, FEB.1978)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION S(MAXS,JCS), V(IRS)
      DOUBLE PRECISION SUM, DELTA
C
      ONE=1.D0
      ZERO=0.D0
      JSTT=JSTART
      JSTP=JSTOP
      IF (JSTT.LT.1) JSTT=1
      IF (JSTP.GE.JSTT.AND.JSTP.LE.JCS) GO TO 5
      IF (IRS.LE.JCS) JSTP=IRS-1
      IF (IRS.GT.JCS) JSTP=JCS
C
    5 DO 40 J=JSTT,JSTP
        SUM=ZERO
        DO 10 I=J,IRS
          V(I)=S(I,J)
          S(I,J)=ZERO
   10     SUM=SUM+V(I)**2
        IF (SUM.LE.ZERO) GO TO 40
C         IF SUM=ZERO, COLUMN J IS ZERO AND THIS STEP OF THE
C         ALGORITHM IS OMITTED
        SUM=DSQRT(SUM)
        IF (V(J).GT.ZERO) SUM=-SUM
        S(J,J)=SUM
        V(J)=V(J)-SUM
```

```
      SUM=ONE/(SUM*V(J))                                                TDHHT560
C       THE HOUSEHOLDER TRANSFORMATION IS T=I-SUM*V*V**T                TDHHT570
      JP1=J+1                                                           TDHHT580
      IF (JP1.GT.JCS) GO TO 40                                          TDHHT590
      DO 30 K=JP1,JCS                                                   TDHHT600
        DELTA=ZERO                                                      TDHHT610
        DO 20 I=J,IRS                                                   TDHHT620
   20     DELTA=DELTA+S(I,K)*V(I)                                       TDHHT630
        DELTA=DELTA*SUM                                                 TDHHT640
        DO 30 I=J,IRS                                                   TDHHT650
   30     S(I,K)=S(I,K)+DELTA*V(I)                                      TDHHT660
   40 CONTINUE                                                          TDHHT670
C                                                                       TDHHT680
      RETURN                                                            TDHHT690
      END                                                               TDHHT700
```

```
      SUBROUTINE THH(R,N,A,IA,M,SOS,NSTRT)
C
C       THIS SUBROUTINE PERFORMS A TRIANGULARIZATION OF A RECTANGULAR
C       MATRIX INTO A SINGLY-SUBSCRIPTED 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 R 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                    SQRT(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
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
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(IA,1),R(1)
      DOUBLE PRECISION SUM, ONE, BETA, DELTA
C
      EPS=-1.D-200            @ MACHINE DEPENDENT ACCURACY TERM
      ZERO=0.D0
      ONE=1.D0
      NSTART=NSTRT
C
      IF (NSTART.LE.0) NSTART=1
      NP1=N+1                      @ NO. COLUMNS OF R
      IF(SOS.LT.ZERO) NP1=N        @ NO COLS. = N IF SOS.LT.0
      KK=NSTART*(NSTART-1)/2
      DO 100 J=NSTART,N             @ J-TH STEP OF HOUSEHOLDER REDUCTION
      KK=KK+J
      SUM=ZERO
      DO 20 I=1,M
   20 SUM=SUM+A(I,J)**2
      IF(SUM.LE.ZERO) GO TO 100  @ IF J-TH COL. OF A.EQ.0 GO TO STEP J+1
      SUM=SUM+R(KK)**2
      SUM=DSQRT(SUM)
```

```
      IF(R(KK).GT.ZERO) SUM=-SUM                                        THH00560
      DELTA=R(KK)-SUM                                                   THH00570
      R(KK)=SUM                                                         THH00580
      JP1=J+1                                                           THH00590
      IF (JP1.GT.NP1) GO TO 105                                         THH00600
      BETA=SUM*DELTA                                                    THH00610
      IF (BETA.GT.EPS) GO TO 100                                        THH00620
      BETA=ONE/BETA                                                     THH00630
      JJ=KK                                                             THH00640
      L=J                                                               THH00650
C        ** READY TO APPLY J-TH HOUSEHOLDER TRANS.                      THH00660
      DO 40 K=JP1,NP1                                                   THH00670
      JJ=JJ+L                                                           THH00680
      L=L+1                                                             THH00690
      SUM=DELTA*R(JJ)                                                   THH00700
      DO 30 I=1,M                                                       THH00710
   30 SUM=SUM+A(I,J)*A(I,K)                                             THH00720
      IF(SUM.EQ.ZERO) GO TO 40                                          THH00730
      SUM=SUM*BETA                                                      THH00740
C        BETA DIVIDE USED HERE TO AVOID OVERFLOW IN                     THH00750
C        PROBLEMS WITH NEAR COLUMN COLLINEARITY. IN THAT CASE           THH00760
C        COMMENT OUT LINE 630 AND CHANGE * TO / IN LINE 740             THH00770
      R(JJ)=R(JJ)+SUM*DELTA                                             THH00780
      DO 35 I=1,M                                                       THH00790
   35 A(I,K)=A(I,K)+SUM*A(I,J)                                          THH00800
   40 CONTINUE                                                          THH00810
  100 CONTINUE                                                          THH00820
  105 IF(SOS.LT.ZERO) RETURN                                            THH00830
C                                                                       THH00840
C     CALCULATE SOS                                                     THH00850
C                                                                       THH00860
      SUM=ZERO                                                          THH00870
      DO 110 I=1,M                                                      THH00880
  110 SUM=SUM+A(I,NP1)**2                                               THH00890
      SOS=DSQRT(SOS**2+SUM)                                             THH00900
C                                                                       THH00910
      RETURN                                                            THH00920
      END                                                               THH00930
```

```
      SUBROUTINE TTHH(R,RA,N)
C
C       THIS SUBROUTINE COMBINES TWO SINGLE SUBSCRIPTED SRIF ARRAYS
C       USING HOUSEHOLDER ORTHOGONAL TRANSFORMATIONS
C
C       R(N*(N+1)/2)    INPUT VECTOR STORED UPPER TRIANGULAR MATRIX.
C                       RESULT IS IN R
C       RA(N*(N+1)/2)   THE SECOND INPUT VECTOR STORED UPPER TRIANGULAR
C                       MATRIX. THIS MATRIX IS DESTROYED BY THE
C                       COMPUTATION
C       N               DIMENSION OF THE ESTIMATED PARAMETER VECTOR.
C                       A NEGATIVE VALUE FOR N IS USED TO NOTE THAT
C                       R AND RA HAVE RT. HAND SIDES INCLUDED AND
C                       HAVE DIM=ABS(N)*(ABS(N)+3)/2.
C
C       ON EXIT RA IS CHANGED AND R CONTAINS THE RESULTING SRIF ARRAY
C
C         COGNIZANT PERSONS G.J.BIERMAN/M.W.NEAD (JPL, JAN.1976)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION RA(1), R(1)
C     DOUBLE PRECISION SUM  @ FOR USE IN SINGLE PRECISION VERSION
C
     ZERO=0.
     ONE=1.
     NP1=N
     IF (N.GT.0) GO TO 10
     N=-N
     NP1=N+1
   10  IJS=1                 @ IJ(START)
      KK=0
       DO 100 J=1,N          @ J-TH STEP OF HOUSEHOLDER REDUCTION
         KK=KK+J
       SUM=R(KK)**2
       DO 20 I=IJS,KK
   20 SUM=SUM+RA(I)**2
     IF (SUM.LE.ZERO) GO TO 100
      SUM=SQRT(SUM)
      IF (R(KK).GT.ZERO) SUM=-SUM
      DELTA=R(KK)-SUM
      R(KK)=SUM
      BETA=ONE/(SUM*DELTA)
      JJ=KK
      L=J
      JP1=J+1
      IKS=KK+1
C         * * * J-TH HOUSEHOLDER TRANS. DEFINED
C               40 LOOP APPLIES TRANSFORM. TO COLS. J+1 TO NP1
      DO 40 K=JP1,NP1
      JJ=JJ+L
      L=L+1
      IK=IKS
      SUM=DELTA*R(JJ)
      DO 30 I=IJS,KK
      SUM=SUM+RA(IK)*RA(I)
```

```
   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
```

```
      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          NUMBER 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
      PARAMETER J12=12, J7=7
      DOUBLE PRECISION  A(N)
      INTEGER CAR(N), TEXT(1), L(J12), LIST(J12)
       LOGICAL NAMES
      INTEGER  V(4),VFMT(J12),V7MT(J7),V12MT(J12)
      DATA  V/'(2X,','A6,1X,',' ','E10.5)'/,(V12MT(I),I=1,12)
     1 /'12','10X,11','20X,10','30X,9','040X,8','050X,7',
     2 '060X,6','070X,5','080X,4','090X,3','100X,2','110X,1'/,
     1 V7MT/'7','017X,6','034X,5','051X,4','068X,3','085X,2','102X,1'/
      DATA KON7/'D17.8)'/, 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
      IF (LEN.EQ.J0) GO TO 5
      IF (LEN.EQ.7) GO TO 1
      IF (LEN.EQ.12) GO TO 2
      WRITE (6,230) LEN
      LEN=12
      GO TO 2
    1 V(4)=KON7; J0=7; J0M1=J0-1; J0P1=J0+1;
     1 REPEAT I=1,J0; VFMT(I)=V7MT(I)
      GO TO 5
    2 V(4)=KON12; J0=12; J0M1=J0-1; J0P1=J0+1;
     1 REPEAT I=1,J0; VFMT(I)=V12MT(I)
    5 M1=1
      M2=J0
      N1=1
      KT=0
      V(2)='A6,1X,'
      IF (.NOT.NAMES) V(2)='I5,2X'
```

```
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
          II=I+KT*J0
   80     L(K)=L(K-1)+II                  @ OBTAIN COL INDEX FOR ROW
   90   CONTINUE
C
        I2=MINO(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)               @ TITLE
  205 FORMAT (1H0,2X,21A6)               @ TITLE
```

```
  201 FORMAT (1H1,2X,'(CONTINUE)  ',19A6)              @ TITLE
  206 FORMAT (1H0,2X,'(CONTINUE)  ',19A6)              @ TITLE
  210 FORMAT (1H0,5X,7(11X,A6))         @ HORIZONTAL NAMES
  220 FORMAT (1H0,3X,7(11X,I6))
  211 FORMAT (1H0,5X,12(4X,A6))         @ HORIZONTAL NAMES
  221 FORMAT (1H0,3X,12(4X,I6))
  230 FORMAT (1H0,20X,'TWOMAT CALLED WITH LENGTH = ',I3)
C
      END
```

```
      SUBROUTINE TZERO (R,N,IS,IF)
C
C        TO ZERO OUT ROWS IS (ISTART) TO IF (IFINAL) OF A VECTOR
C        STORED UPPER TRIANGULAR MATRIX
C
C        R(N*(N+1)/2)  INPUT VECTOR STORED UPPER TRIANGULAR MATRIX
C        N             DIMENSION OF R
C        IS            FIRST ROW OF R THAT IS TO BE SET TO ZERO
C        IR            LAST ROW OF R THAT IS TO BE SET TO ZERO
C
C        COGNIZANT PERSONS:  G.J.BIERMAN/C.F.PETERS (JPL, NOV. 1975)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION R(1)
C
      ZERO=0.D0
        IJS=IS*(IS-1)/2
        DO 10 I=IS,IF
        IJS=IJS+I
        IJ=IJS
        DO 10 J=I,N
        R(IJ)=ZERO
        IJ=IJ+J
   10 CONTINUE
C
        RETURN
        END
```

```
      SUBROUTINE UDCOL(U,N,KS,NCOLOR,V,EM,Q)
C
C     COLORED NOISE UPDATING OF THE U-D COVARIANCE FACTORS, I.E.
C       U*D*(U**T)-OUTPUT=PHI*U*D*(U**T)*(PHI**T)+Q
C       PHI=DIAG(0(KS-1),EM(1),...,EM(NCOLOR),0(N-(KS-1+NCOLOR)))
C     Q=DIAG(0(KS-1),Q(1),...,Q(NCOLOR),0(N-(KS-1+NCOLOR)))
C     0(K) IS A VECTOR OF ZEROS
C
C     THE ALGORITHM USED IS THE BIERMAN-THORNTON ONE COMPONENT
C     AT-A-TIME UPDATE. CF.BIERMAN "FACTORIZATION METHOD
C     FOR DISCRETE SEQUENTIAL ESTIMATION", ACADEMIC PRESS (1977)
C     PP.147-148
C
C     U(N*(N+1)/2)  INPUT U-D VECTOR STORED COVARIANCE FACTORS.
C                   THE COLORED NOISE UPDATE RESULT RESIDES
C                   IN U ON OUTPUT
C     N             FILTER DIMENSION. IF THE LAST COLUMN OF U
C                   HOUSES THE FILTER ESTIMATES, THEN
C                   N=NUMBER FILTER VARIABLES + 1
C     KS            THE LOCATION OF THE FIRST COLORED NOISE TERM
C                   (KS.GE.1.AND.KS.LE.N)
C     NCOLOR        THE NUMBER OF COLORED NOISE TERMS (NCOLOR.GE.1)
C     V(KS-1+NCOLOR)  WORK VECTOR
C     EM(NCOLOR)    INPUT VECTOR OF COLORED NOISE MAPPING TERMS
C                   (UNALTERED BY PROGRAM)
C     Q(NCOLOR)     INPUT VECTOR OF PROCESS NOISE VARIANCES
C                   (UNALTERED BY PROGRAM)
C
C         SUBROUTINE REQUIRED: RANK1
C
C      COGNIZANT PERSON:  G.J.BIERMAN  (JPL, JAN. 1978)
      DOUBLE PRECISION TMP,S
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION U(1),V(1),EM(1),Q(1)
C
C * * * * * * INITIALIZATION
      NM1=N-1
      KSM1=KS-1
      JJOLD=KS*KSM1/2
      KOL=KSM1
C * * * * * *
C
      DO 50 K=1,NCOLOR
        KOLM1=KOL
        KOL=KOL+1
        JJ=JJOLD+KOL
        TMP=U(JJ)*EM(K)
        C=Q(K)*U(JJ)
        S=TMP*EM(K)+Q(K)                 QD(J) UPDATE
        U(JJ)=S
C
        IF (KOL.GE.N) GO TO 20
        IJ=JJ
        DO 10 J=KOL,NM1
          IJ=IJ+J
```

```
   10      U(IJ)=U(IJ)*EM(K)       @ UPDATING ROW KOL ENTRIES
C
   20    IF (JJ.EQ.1) GO TO 50          @ (WHEN KS=1, N=1)
         IF (S.LE.0.D0) GO TO 30
         TMP=TMP/S              @ TMP=EM(K)*D(KOL)-OLD/D(KOL)-NEW
         C=C/S                  @ C=Q(K)*D(KOL)-OLD/D(KOL)-NEW
   30    DO 40 I=1,KOLM1
           V(I)=U(JJOLD+I)
   40      U(JJOLD+I)=TMP*V(I)
           IF (KOLM1.GT.1) GO TO 45
           U(1)=U(1)+C*V(1)**2
           GO TO 50
   45      CALL RANK1(U,U,KOLM1,C,V)
   50      JJOLD=JJ
C
       RETURN
       END
```

```
      SUBROUTINE UDMEAS (U,N,R,A,F,G,ALPHA)
C
C       COMPUTES ESTIMATE AND U-D MEASUREMENT UPDATED
C       COVARIANCE, P=U*D*U**T
C
C       *** INPUTS ***
C
C       U       UPPER TRIANGULAR MATRIX, WITH D ELEMENTS STORED AS THE
C               DIAGONAL. U IS VECTOR STORED AND CORRESPONDS TO THE
C               A PRIORI COVARIANCE. IF STATE ESTIMATES ARE COMPUTED,
C               THE LAST COLUMN OF U CONTAINS X.
C       N       DIMENSION OF THE STATE ESTIMATE, N.GT.1
C       R       MEASUREMENT VARIANCE
C       A       VECTOR OF MEASUREMENT COEFFICIENTS, IF DATA THEN A(N+1)=Z
C       ALPHA   IF ALPHA LESS THAN ZERO NO ESTIMATES ARE COMPUTED
C               (AND X AND Z NEED NOT BE INCLUDED)
C
C       *** OUTPUTS ***
C
C       U         UPDATED, VECTOR STORED FACTORS AND ESTIMATE AND
C                 U((N+1)(N+2)/2) CONTAINS (Z-A**T*X)
C
C       ALPHA     INNOVATIONS VARIANCE OF THE MEASUREMENT RESIDUAL
C       G         VECTOR OF UNWEIGHTED KALMAN GAINS, THE KALMAN
C                 GAIN K IS EQUAL TO G/ALPHA
C       F         CONTAINS U**T*A AND (Z-A**T*X)/ALPHA
C                 ONE CAN HAVE F OVERWRITE A TO SAVE STORAGE
C
C          COGNIZANT PERSONS: G.J. BIERMAN/M.W. NEAD (JPL, FEB.1978)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION U(1), A(1), F(1), G(1)
      DOUBLE PRECISION SUM,BETA,GAMMA
      LOGICAL IEST
C
      ZERO=0.D0
      IEST=.FALSE.
      ONE=1.D0
      NP1=N+1
      NP2=N+2
      NTOT=N*NP1/2
      IF (ALPHA.LT.ZERO) GO TO 3
      SUM=A(NP1)
      DO 1 J=1,N
    1    SUM=SUM-A(J)*U(NTOT+J)
      U(NTOT+NP1)=SUM                        @ Z=Z-A**T*X
      IEST=.TRUE.
C
    3 JJN=NTOT
      DO 10 L=2,N
         J=NP2-L
         JJ=JJN-J
         SUM=A(J)
         JM1=J-1
         DO 5 K=1,JM1
```

```
    5         SUM=SUM+U(JJ+K)*A(K)
          F(J)=SUM
      G(J)=SUM*U(JJN)
   10 JJN=JJ
      F(1)=A(1)
      G(1)=U(1)*F(1)
C         F=U**T*A AND G=D*(U**T*A)
C
      SUM=R+G(1)*F(1)                   @ SUM(1)
      GAMMA=0                            @ FOR R=0 CASE
      IF (SUM.GT.ZERO) GAMMA=ONE/SUM     @ FOR R=0 CASE
      IF (F(1).NE.ZERO) U(1)=U(1)*R*GAMMA                 @ D(1)
C
      KJ=2
      DO 20 J=2,N
          BETA=SUM                      @ BETA=SUM(J-1)
          TEMP=G(J)
          SUM=SUM+TEMP*F(J)             @ SUM(J)
          P=-F(J)*GAMMA                 @ P=-F(J)*(1/SUM(J-1)) EQN(21)
          JM1=J-1
          DO 15 K=1,JM1
            S=U(KJ)
            U(KJ)=S+P*G(K)              @ EQN(22)
            G(K)=G(K)+TEMP*S            @ EQN(23)
   15          KJ=KJ+1
      IF (TEMP.EQ.ZERO) GO TO 20         @ FOR R=0 CASE
          GAMMA=ONE/SUM                 @ GAMMA=1/SUM(J)
          U(KJ)=U(KJ)*BETA*GAMMA        @ D(J)      EQN(19)
   20 KJ=KJ+1
      ALPHA=SUM
C
C         EQN. NOS. REFER TO BIERMAN'S 1975 CDC PAPER, PP. 337-346.
C
      IF (.NOT.IEST) RETURN
      F(NP1)=U(NTOT+NP1)*GAMMA
      DO 30 J=1,N
   30   U(NTOT+J)=U(NTOT+J)+G(J)*F(NP1)
C
      RETURN
      END
```

```
      SUBROUTINE UD2COV (UIN,POUT,N)
C
C     TO OBTAIN A COVARIANCE FROM ITS U-D FACTORIZATION. BOTH MATRICES
C     ARE VECTOR STORED AND THE OUTPUT COVARIANCE CAN OVERWRITE THE
C     INPUT U-D ARRAY. UIN=U-D IS RELATED TO POUT VIA POUT=UDU(**T)
C
C     UIN(N*(N+1)/2)  INPUT U-D FACTORS, VECTOR STORED WITH THE D
C                     ENTRIES STORED ON THE DIAGONAL OF UIN
C     POUT(N*(N+1)/2) OUTPUT COVARIANCE, VECTOR STORED.
C                     (POUT=UIN IS PERMITTED)
C     N               DIMENSION OF THE MATRICES INVOLVED, N.GT.1
C
C     COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD  (JPL, FEB. 1977)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
      DIMENSION   UIN(1),   POUT(1)
C
      POUT(1)=UIN(1)
      JJ=1
      DO 20 J=2,N
        JJL=JJ                          @ (J-1,J-1)
        JJ=JJ+J
        POUT(JJ)=UIN(JJ)
        S=POUT(JJ)
        II=0
        JM1=J-1
        DO 20 I=1,JM1
          II=II+I
          ALPHA=S*UIN(JJL+I)              @ JJL+I=(I,J)
          IK=II
          DO 10 K=I,JM1
            POUT(IK)=POUT(IK)+ALPHA*UIN(JJL+K)     @ JJL+K=(K,J)
   10       IK=IK+K
   20   POUT(JJL+I)=ALPHA
C
      RETURN
      END
```

```
      SUBROUTINE UD2SIG(U,N,SIG,TEXT,NCT)
C
C     COMPUTE STANDARD DEVIATIONS (SIGMAS) FROM U-D COVARIANCE FACTORS
C
C     U(N*(N+1)/2)  INPUT VECTOR STORED ARRAY CONTAINING THE U-D
C                   FACTORS. THE D (DIAGONAL) ELEMENTS ARE STORED
C                   ON THE DIAGONAL
C     N             U MATRIX DIMENSION, N.GT.1
C     SIG(N)        VECTOR OF OUTPUT STANDARD DEVIATIONS
C     TEXT( )       ARRAY OF FIELDATA CHARACTERS TO BE PRINTED
C                   PRECEDING THE VECTOR OF SIGMAS
C     NCT           NUMBER OF CHARACTERS IN TEXT, 0.LE.NCT.LE.126
C                   IF NCT=0, NO SIGMAS ARE PRINTED
C
C     COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD   (JPL, FEB. 1977)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER TEXT(1)
      DIMENSION U(1), SIG(1)
C
      JJ=1
      SIG(1)=U(1)
      DO 10 J=2,N
        JJL=JJ                          @ (J-1,J-1)
        JJ=JJ+J
        S=U(JJ)
        SIG(J)=S
        JM1=J-1
        DO 10 I=1,JM1
   10     SIG(I)=SIG(I)+S*U(JJL+I)**2
C
C          WE NOW HAVE VARIANCES
C
      DO 20 J=1,N
   20   SIG(J)=SQRT(SIG(J))
      IF (NCT.EQ.0) GO TO 30
      NC=NCT/6
      IF (MOD(NC,6).NE.0) NC=NC+1
      WRITE (6,40) (TEXT(I),I=1,NC)
      WRITE (6,50) (SIG(I),I=1,N)
   30 RETURN
C
   40 FORMAT (1H0,2X,21A6)
   50 FORMAT (1H0,(6D18.10))
      END
```

```
      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.BIERMAN/M.W.NEAD  (JPL, JAN.1978)
C
      DOUBLE PRECISION  RIN(1), ROUT(1), ZERO, DINV, ONE, SUM
C
      ZERO=0.D0
      ONE=1.D0
C
      IF (RIN(1).NE.ZERO) GO TO 5
      J=1
      WRITE (6,100) J,J
      RETURN
C
    5 ROUT(1)=ONE/RIN(1)
C
      JJ=1
      DO 20 J=2,N
        JJOLD=JJ
        JJ=JJ+J
        IF (RIN(JJ).NE.ZERO) GO TO 10
        WRITE (6,100) J,J
        RETURN
C
   10   DINV=ONE/RIN(JJ)
        ROUT(JJ)=DINV
        II=0
        IK=1
        JM1=J-1
        DO 20 I=1,JM1
          II=II+I
          IK=II
          SUM=ZERO
          DO 15 K=I,JM1
            SUM=SUM+ROUT(IK)*RIN(JJOLD+K)
   15       IK=IK+K
   20     ROUT(JJOLD+I)=-SUM*DINV
C
      RETURN
C
  100 FORMAT (1H0,10X,'* * * MATRIX INVERSE COMPUTED ONLY UP TO BUT NOT
     1INCLUDING COLUMN',I4,' * * * MATRIX DIAGONAL ',I4,' IS ZERO * * *'
```

```
     2)                                                                 UTINV560
C                                                                       UTINV570
      END                                                               UTINV580
```

```
C
      SUBROUTINE  UTIROW (RIN,N,ROUT,NRY)
C
C     TO COMPUTE THE INVERSE OF AN UPPER TRIANGULAR (VECTOR STORED)
C     MATRIX WHEN THE LOWER PORTION OF THE INVERSE IS GIVEN
C
C         ON INPUT:
C
C            RX    RXY                  *      *                RX    RXY
C     RIN=                    ROUT=                 WHERE     R=
C            *     *                    0    RY**-1              0     RY
C
C         ON OUTPUT: RIN IS UNCHANGED AND ROUT=R**-1
C         THE RESULT CAN OVER-WRITE THE INPUT (I.E. RIN=ROUT)
C
C     RIN(N*(N+1)/2)      INPUT VECTOR STORED TRIANGULAR MATRIX
C                         THE BOTTOM NRY ROWS ARE IGNORED
C     N                   MATRIX DIMENSION
C     ROUT(N*(N+1)/2)     OUTPUT VECTOR STORED MATRIX. ON INPUT THE
C                         BOTTOM NRY ROWS CONTAIN THE LOWER PORTION
C                         OF R**-1. ON OUTPUT ROUT=R**-1
C     NRY                 DIMENSION OF LOWER (ALREADY INVERTED)
C                         TRIANGULAR R. IF NRY=0, ORDINARY MATRIX
C                         INVERSION RESULTS.
C
C     COGNIZANT PERSONS:   G.J.BIERMAN/M.W.NEAD   (JPL MARCH 1977)
C
      DOUBLE PRECISION   RIN(1), ROUT(1), SUM, ZERO, ONE, DINV
      DATA   ONE/1.D0/, ZERO/0.D0/
C
C          INITIALIZATION
C
      NR=N*(N+1)/2                   @ NO. ELEMENTS IN R
      ISTRT=N-NRY                    @ FIRST ROW TO BE INVERTED
      IRLST=ISTRT+1                  @ IRLST=PREVIOUS IROW INDEX
      II=ISTRT*IRLST/2               @ II=DIAGONAL
      DO 40 IROW=ISTRT,1,-1
        IF (RIN(II).NE.ZERO) GO TO 10
        WRITE (6,50) IROW
        RETURN
   10   DINV=ONE/RIN(II)
        ROUT(II)=DINV
        KJS=NR+IROW                  @ KJ(START)
        IKS=II+IROW                  @ IK(START)
C
        IF (IRLST.GT.N) GO TO 35
        DO 30 J=N,IRLST,-1
          KJS=KJS-J
          SUM=ZERO
          IK=IKS
          KJ=KJS
C
          DO 20 K=IRLST,J
            KJ=KJ+1
            SUM=SUM+RIN(IK)*ROUT(KJ)
```

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

T
2
E

```
      SUBROUTINE WGS (W,IMAXW,IW,JW,DW,U,V)
C     MODIFIED GRAMM-SCHMIDT ALGORITHM FOR REDUCING WDW(**T) TO UDU(**T)
C     FORM WHERE U IS A VECTOR STORED TRIANGULAR MATRIX WITH THE
C     RESULTING D ELEMENTS STORE ON THE DIAGONAL
C
C     W(IW,JW)         INPUT MATRIX TO BE REDUCED TO TRIANGULAR FORM.
C                      THIS MATRIX IS DESTROYED BY THE CALCULATION
C                      IW.LE.IMAXW.AND.IW.GT.1
C     IMAXW            ROW DIMENSION OF W MATRIX
C     IW               NO. ROWS OF W MATRIX, DIMENSION OF U
C     JW               NO. COLS OF W MATRIX
C     DW(JW)           VECTOR OF NON-NEGATIVE WEIGHTS FOR THE
C                      ORTHOGONALIZATION PROCESS. THE D'S ARE UNCHANGED
C                      BY THE CALCULATION.
C     U(IW*(IW+1)/2)   OUTPUT UPPER TRIANGULAR VECTOR STORED OUTPUT
C     V(JW)            WORK VECTOR
C
C                      (SEE BOOK:
C     ' FACTORIZATION METHODS FOR DISCRETE SEQUENTIAL ESTIMATION ',
C                         BY G.J.BIERMAN)
C      ESTIMATION
C
C     COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD   (JPL, FEB.1978)
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION SUM,Z,DINV
      DIMENSION W(IMAXW,1), DW(1), U(1), V(1)
C
      Z=0.D0
      ONE=1.D0
      IWP2=IW+2
      DO 100 L=2,IW
        J=IWP2-L
        SUM=Z
        DO 40 K=1,JW
          V(K)=W(J,K)
          U(K)=DW(K)*V(K)                 @U HERE IS USED AS A WORK VECTOR
   40     SUM=V(K)*U(K)+SUM
        W(J,J)=SUM                        @ EQ.(4.9) OF BOOK, NEW DW(J)
        DINV=SUM
        JM1=J-1
        IF (SUM.GT.Z) GO TO 45
C     W(J,.)=0. WHEN DINV=0 (DINV=NORM(W(J,.)**2))
        DO 44 K=1,JM1
   44     W(J,K)=Z
        GO TO 100
   45   DO 70 K=1,JM1
          SUM=Z
          DO 50 I=1,JW
   50       SUM=W(K,I)*U(I)+SUM
          SUM=SUM/DINV
C     DIVIDE HERE (IN PLACE OF RECIPROCAL MULTIPLY) TO AVOID
```

```
C      POSSIBLE OVERFLOW
C
          DO 60 I=1,JW
   60       W(K,I)=W(K,I)-SUM*V(I)
   70     W(J,K)=SUM                          @ EQ.(4.10) OF BOOK
  100   CONTINUE                              @ U(K,J) STORED IN W(J,K)
C
C         THE LOWER PART OF W IS U TRANSPOSE
C
      SUM=Z
      DO 105 K=1,JW
  105   SUM=DW(K)*W(1,K)**2+SUM
      U(1)=SUM
      IJ=1
      DO 110 J=2,IW
        DO 110 I=1,J
          IJ=IJ+1
  110     U(IJ)=W(J,I)
C
      RETURN
      END
```