JPL PUBLICATION 77-26

A Parameter Estimation Subroutine Package

(NASA-CR-154109) A PARAMETER ESTIMATION SUBRCUTINE PACKAGE (Jet Propulsion Lab.) HC A05/MF A01 CSCL 09B

N77-28828

Unclas G3/61 39276

REPRODUCED BY NATIONAL TECHNICAL INFORMATION SERVICE U. S DEPARTMENT OF COMMERCE SPRINGFIELD, VA 22161

National Aeronautics and Space Administration

Jet Propulsion Laboratory California Institute of Technology Pasadena, California 91103

NOTICE

THIS DOCUMENT HAS BEEN REPRODUCED FROM THE BEST COPY FURNISHED US BY THE SPONSORING AGENCY. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE.

TECHNICAL REPORT STANDARD TITLE PAGE

4. Title and Subtitle A PARAMETER ESTIMATION SUE	ROUTINE PACKAGE	5. R	eport Date	
A PARAMETER ESTIMATION SUE	ROUTINE PACKAGE		1	1077
		6. P	erforming Organization	n Code
7. Author(s)		8. P	erforming Organizatio	n Report No.
G. J. Bierman/M. W. Nead				·
9. Performing Organization Name	and Address	10. V	York Unit No.	•
JET PROPULSION LA	BORATORY	11 0	Contract or Grant No.	
California Institu 4800 Oak Grove Dr	ite of Technology	11. C	NAS 7-100	
Pasadena, Californ	nia 91103	13 . T	ype of Report and Per	iod Covered
12. Sponsoring Agency Name and A	ddress	J	PL Publication	
NATIONAL AERONAUTICS AND Washington, D.C. 20546	SPACE ADMINISTRATION	14.5	ponsoring Agency Coc	le
7 Al				
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 parameter 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 <u>listings</u> are given, along with examples of how these routine can be used. The following outline indicates the scope of this report: Section I, introduction with reference to background materials; 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 data processing algorithms that are often used for least squares analyses.				ary of iety multi- mal amount ren, along ates the materials; summary; usage spact gorithms
17. Key Words (Selected by Author Computer Programming and So Numerical Analysis Statistics and Probability	(s)) 18. D	Distribution Stat	ement - Unlimited	
19. Security Classif. (of this report) Unclassified	20. Security Classif. Unclassified	(of this page)	21. No. of Pages 101	22. Price

HOW TO FILL OUT THE TECHNICAL REPORT STANDARD TITLE PAGE

Make items 1, 4, 5, 9, 12, and 13 agree with the corresponding information on the report cover. Use all capital letters for title (item 4). Leave items 2, 6, and 14 blank. Complete the remaining items as follows:

- 3. Recipient's Catalog No. Reserved for use by report recipients.
- 7. Author(s). Include corresponding information from the report cover. In addition, list the affiliation of an author if it differs from that of the performing organization.
- 8. Performing Organization Report No. Insert if performing organization wishes to assign this number.
- 10. Work Unit No. Use the agency-wide code (for example, 923-50-10-06-72), which uniquely identifies the work unit under which the work was authorized. Non-NASA performing organizations will leave this blank.
- 11. Insert the number of the contract or grant under which the report was prepared.
- 15. Supplementary Notes. Enter information not included elsewhere but useful, such as: Prepared in cooperation with... Translation of (or by)... Presented at conference of... To be published in...
- 16. Abstract. Include a brief (not to exceed 200 words) factual summary of the most significant information contained in the report. If possible, the abstract of a classified report should be unclassified. If the report contains a significant bibliography or literature survey, mention it here.
- 17. Key Words. Insert terms or short phrases selected by the author that identify the principal subjects covered in the report, and that are sufficiently specific and precise to be used for cataloging.
- Distribution Statement. Enter one of the authorized statements used to denote releasability to the public or a limitation on dissemination for reasons other than security of defense information. Authorized statements are "Unclassified-Unlimited," "U.S. Government and Contractors only," "U.S. Government Agencies only," and "NASA and NASA Contractors only."
- Security Classification (of report). NOTE: Reports carrying a security classification will require additional markings giving security and downgrading information as specified by the Security Requirements Checklist and the DoD Industrial Security Manual (DoD 5220, 22-M).
- 20. Security Classification (of this page). NOTE: Because this page may be used in preparing announcements, bibliographies, and data banks, it should be unclassified if possible. If a classification is required, indicate separately the classification of the title and the abstract by following these items with either "(U)" for unclassified, or "(C)" or "(S)" as applicable for classified items.
- 21. No. of Pages. Insert the number of pages.
- 22. Price. Insert the price set by the Clearinghouse for Federal Scientific and Technical Information or the Government Printing Office, if known.

JPL PUBLICATION 77-26

A Parameter Estimation Subroutine Package

G. J. Bierman M. W. Nead

July 1, 1977

.

National Aeronautics and Space Administration

Jet Propulsion Laboratory California Institute of Technology Pasadena, California 91103 Prepared Under Contract No NAS 7-100 National Aeronautics and Space Administration PREFACE

77-26

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

77-26

AKNOWLEDGEMENT

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. Discussions with T. 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. Koble and N. Mottinger who had a different but related problem that involved combining estimates from different missions. Thanks and credit are also due to J. Ellis, N. Hamata, and F. Peters for contributing to and experimenting with this package of subroutines. 77-26

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 parameter estimation problems. Our purpose is to present an easy to use multipurpose 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 data processing algorithms that are often used for least squares analyses.

v

77-26

CONTENTS

I.	Introduction	1
II.	Applications and Examples	4
III.	Subroutine Directory Summary	3
IV.	Subroutine Directory User Description	3
v.	FORTRAN Subroutine Listings 6	3

Preceding page blank

. .

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

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

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

1

subroutine to another are compatible, and vector arrays displayed using the print subroutine TRIMAT appear in a triangular matrix format. <u>Aside</u>: The most notable exception is that matrix problems are generally formulated using doubly subscripted arrays. Transforming a double subscripted symmetric or upper triangular matrix $A(\cdot, \cdot)$ to a vector stored form, $U(\cdot)$ is quite simply accomplished in FORTRAN via

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

1

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

JJ = 0		JJ = N*(N+1)/2
DO 1 J = $1,N$	or	DO 1 J = $N, 1, -1$
JJ = JJ+J		U(JJ) = D(J)
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

 \cap

Although this package of subroutines is designed in the main, for the analysis of parameter estimation problems one can use it to solve problems that involve process noise. With modest amounts of additional programming one can even apply our package to filtering problems that involve colored noise and mapping. In the latter case, however, reductions gained from our use of vector storage are for the most part lost.

2

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 are not truly portable, and users can, when necessary make modifications so that the subroutine package can operate on systems other than the UNIVAC 1108. 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.

77-26

II. <u>APPLICATIONS AND EXAMPLES</u>

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)	AGTRN	(A G Turner)	Agee-Turner rank 1 update
2)	A2A1	(A to A one)	Matrix A to matrix Al
3)	COMBO	(combo)	Combine R and A namelists
4)	COV2RI	(cov to R I)	Covariance to R inverse
5)	COV2UD	(cov to U D)	Covariance to U-D factors
6)	C2C	(C to C)	Permute the rows and columns of matrix C
7)	INF2R	(inf to R)	Information matrix to (triangular) R
8)	PERMUT	(permute)	Permute the columns of matrix A
9)	RINCON	(rin con)	R inverse with condition number bound
10)	RI2COV	(R I to cov)	R inverse to covariance
11)	R2A	(R to A)	Triangular R to matrıx A
12)	R2RA	(R to R A)	Transfer a triangular block of R to trian- gular RA
13)	RUDR	(rudder)	SRIF R to U-D factors or vice versa
14)	ТНН	(T H H)	Triangular Householder data processing
15)	TRIMAT	(tri mat)	Triangular matrix print
16)	TTHH	(ТТНН)	Two triangular matrix Householder processing
17)	TZERO	(T zero)	Zero a horizontal segment of a triangular matrix

18)	UDMES	(U D measurement)	U-D measurement updating
19)	UD2COV	(U D to cov)	U-D factors to covariance
20)	UD2SIG	(U D to sig)	U-D factors to sigmas
21)	UTINV	(U T inverse)	Upper triangular matrix inverse
22)	UTIROW		Upper triangular inverse, inverting only the upper rows
23)	WGS	(W G-S)	Weighted Gram-Schmidt triangular 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 this package of subroutines has a wide scope of applicability.

II.1 <u>Simple Least Squares</u>

Given data in the form of an overdetermined systems 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

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

Remarks: The array [A z] corresponds to the equations Ax = z - v, $v \in N(0, I)$; the array $[\hat{R} \ z]$ corresponds to the triangular data equation $\hat{R}x = \hat{z} - \hat{v}$, $v \in N(0, I)$ and $e = ||z - A\hat{x}||$ $\hat{R} \ z] \xrightarrow{\text{UTINV}} [\hat{R}^{-1} \ \hat{x}]$ Remark: $\hat{x} = \hat{R}^{-1} \ \hat{z}$

77-26

One may be concerned with the integrity of the computed inverse and the estimate. If one uses subroutine RINCON instead of UTINY 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 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 $[R_0 \ z_0]$ using either subroutine THH, or subroutine TTHH when the a priori is diagonal or triangular, e.g.,

$$\begin{array}{c} \hat{\left[\begin{array}{c} \hat{R} & \hat{z} \end{array} \right]} \\ \vdots \\ \left[\begin{array}{c} \hat{R} & \hat{z} \end{array} \right]} \end{array} \xrightarrow{\text{TTHH}} \left[\hat{R} & \hat{z} \right]^{*}$$

^{*} The new result overwrites the old.

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

77-26

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;

$$\Lambda \xrightarrow{\text{INF2R}} R_{0}$$

If there were a normal equation estimate $z = A^{T}b$, 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 TTH and UDMES 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-v with normal equations, $A^{T}Ax_{o} = A^{T}b$; $\Lambda = A^{T}A$ is the information matrix.

The acronym SRIF represents Square Root Information Filter. The SRIF is discussed at length in reference [3].

- c) $\hat{[R z]} \xrightarrow{\text{THH}} \hat{[R z]}^*$
- d) If there is more data go back to b)
- e) Compute estimates and/or covariances using UTINV and RI2COV (as in example II.1)

U-D** Processing

- a') Initialize [U-D x] with a priori U-D information and estimate
- b') Read the next [A z] scalar measurement from the file
- c') $[\hat{U}-\hat{D} \hat{x}]$ [A z] UDMES $[\hat{U}-\hat{D} \hat{x}]^*$
- d') If there is more data go back to b')
- e') Compute standard deviations or covariances using UD2SIG or UD2COV.

Note that subroutine THH is best (most efficiently) used with data batches of substantial size (say 5 or more) and that UDMES 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 UDMES 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 awhile 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.

<u>Remarks</u>: 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 UDMES 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 $\begin{bmatrix} A & A \\ R & z \end{bmatrix}$ corresponding to the 14 parameter names, a_r , a_x , a_y , x, y, z, v_x , v_y , v_z , GM, CU41, L041, CU43, L043 (constant spacecraft accelerations, position and velocity, target body gravitational constant, and spin axis and longitude station location errors).

Let us ask first what would the computed error covariance be of a model containing only the first 10 variables, i.e., by ignoring the effect of the station location errors. One would apply UTINV and RI2COV just as in example II.1, <u>except</u> 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

9

77-26

• $[\hat{R} \hat{z}] \xrightarrow{R2A} [A]$ NAME ORDER OF A

77-26

x, y, z, v_x, v_y, v_z, GM, CU41, LO41, CU43, LO43, RHS^{*}, a_r, a_x, a_y,

<u>Remark</u>: 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.

Remark: R here can replace the original R and z.

• [R]
$$\xrightarrow{\text{UTINV}}$$
 [R⁻¹ x_{est}] $\xrightarrow{\text{RI2COV}}$ [COV x_{est}]

<u>Remarks</u>: 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_v .

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 ll 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,

•
$$[\hat{R} \hat{z}]$$

INPUT NAMES:
 $a_r, a_x, a_y, x, y, z, v_x, v_y, v_z, GM$
 $CU41, L041, CU43, L043, RHS$
Remark: The lower triangle starting with x is copied into R_A .
 $Remark:$ $Remark$: R

•
$$[R_A \ z_A] \xrightarrow{\text{RZA}} [A, \ z_A]$$
 (Reordering)
NAMES: GM, CU41, LO41, CU43, LO43,
 $x, y, z, v_x, v_y, v_z, \text{RHS}$
• $[A, \ z_A] \xrightarrow{\text{THH}} [\hat{R}_A \ \hat{z}_A]$ (Triangularizing)
• $[\hat{R}_A \ \hat{z}_A] \xrightarrow{\text{RZRA}} [R_x \ z_x]$ (Shifting array)
NAMES: $x, y, z, v_x, v_y, v_z, \text{RHS}$

<u>Remark</u>: The lower right triangle starting with x is copied into $\frac{R}{x}$. 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,

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

ORIGINAL NAMES DESIRED NAMES: x, y, z, v_x, v_y, v_z, RHS <u>Remark</u>: By using COMBO the columns of $[\hat{R} \ z]$ are ordered corresponding to the names a_r , a_x , a_y , GM, CU41, LO41, CU43, and LO43, followed by the desired names list.

11

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

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

•
$$[\hat{R} \hat{z}] \xrightarrow{R2RA} [R_x z_x]$$

٦

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

To delete the last k parameters from a SRIF array, it is not necessary to use subroutines R2A and THH. The first N - k = \overline{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</u>, Perturbation, Computed Covariance and Consider Covariance Matrix Computation

Suppose that one is given a SRIF array

$$\begin{bmatrix} N_{x} & N_{y} & 1 \\ M_{x} & M_{y} & M_{y} \\ R_{x} & R_{xy} & z_{x} \\ 0 & R_{y} & z_{y} \end{bmatrix} \begin{cases} N_{x} \\ N_{y} \end{cases}$$
(II.5a)

in which the N 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

$$R_{x} + R_{y} = z_{x} - v_{x}, \quad v_{x} \in \mathbb{N}(0, 1)$$
(II.5b)

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

$$x - x_{c} = -R_{x}^{-1} R_{xy} y - R_{x}^{-1} v_{x},$$
 (II.5c)

and from this one obtains

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

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

Pert = Sen Diag
$$(\sigma_y(1), \dots, \sigma_y(N_y))$$
 (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.)

$$P_{con} = R_x^{-1} R_x^{-T} + Sen P_y Sen^T$$
(II.5f)
= $P_c + (Pert)(Pert)^T$ if P_y is diagonal*

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 $[P_y^{l_2}(0) \ y_o]^{**}$ into the lower right

 $\frac{}{Pert} = Sen P_{y}^{\frac{1}{2}}$

The a priori estimate y of consider parameters is generally zero.

corner of (II.5a), replacing R $_{\rm y}$ and z $_{\rm y},$ i.e.,

$$\begin{bmatrix} \dot{R} & z \end{bmatrix} \xrightarrow{R2RA} \begin{bmatrix} R_{x} & R_{xy} & z_{x} \\ R_{y}^{\frac{1}{2}}(0) & A_{y} \end{bmatrix} \xrightarrow{R2RA} \begin{bmatrix} R_{y} & R_{y} & z_{y} \\ 0 & P_{y}^{\frac{1}{2}}(0) & A_{y} \end{bmatrix}$$

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

77-26

R x	R Xy	z _x		R_1	Pert**	×c
0	₽ ¹ 2(0)	^ У _о	UTIROW	0	$\mathbb{P}_{y}^{\frac{1}{2}}(0)$	^ У _о
0	0	→1		0	0	-1

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}

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

 $[R_x^{-1} \text{ Pert}] \xrightarrow{\text{RI2COV}} [P_{\text{con}}]$ 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 $R_y(0)$ is diagonal.

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

$$\operatorname{Sen}_{j} = \operatorname{Sen}_{j-1} - K_{j} \begin{pmatrix} a^{T} & \operatorname{Sen}_{j-1} + a^{T} \\ x & j-1 \end{pmatrix}$$

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

$$\begin{bmatrix} s_1, \dots, s_n \end{bmatrix} = Sen_j$$
 (estimate error sensitivities)

then compute

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

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

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

If
$$P_y(0) = U D U^T_y$$
, instead of $P_y(0) = Diag(\sigma_1^2, \dots, \sigma_n^2)$, then in the

U-D recursion one should replace the Sen columns by those of Sen U 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 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 subcoutine 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 namelisc; then use subroutine THH to retriangularize and finally use subroutine R2RA to put the y independent subarray in position for further use; viz.

77-26

16

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

The rows $\begin{bmatrix} R & R & z \\ y & yx & y \end{bmatrix}$ 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

$$A_{j}x + B_{j}y_{j} = z_{j} - v_{j} \qquad j = 1, \dots, N$$

where there is generally a priori information on the vector y_j variables. Rather than form a concatenated state vector composed of x, y_1, \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.

$$\begin{bmatrix} \mathbf{R} & \mathbf{R} & \mathbf{z} \\ \mathbf{y}_{1} & \mathbf{y}_{1} \mathbf{x} & \mathbf{y}_{1} \\ \mathbf{0} & \mathbf{R} & \mathbf{z} \\ \mathbf{x}_{1} & \mathbf{x}_{1} \end{bmatrix}$$

Copy the top N rows if one will later want an estimate or covariance of y_1 the y_1 parameters. Apply subroutine TZERO to zero the top N rows and y_1 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 [B₂ A₂ Z₂] and the procedure repeated.

77-26

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

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

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

$$R_{j}x_{j} = z_{j} - v_{j}; \quad v_{j} \in \mathbb{N}(0, I_{n})$$
$$Q_{j}^{-1_{2}}w_{j} = 0 - v_{j}^{(w)}; \quad v_{j}^{(w)} \in \mathbb{N}(0, I)$$

where $Q_j^{\frac{1}{2}}$ 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 variables have the same dimension. This assumption, though not essential, simplifies our description of the procedure.

$$\begin{bmatrix} I & 0 \\ n_{W} & \\ -R_{J}G_{j}Q_{J}^{\frac{1}{2}} & R_{j} \end{bmatrix} \begin{bmatrix} \Lambda \\ J \\ x_{j+1} \end{bmatrix} = \begin{bmatrix} 0 \\ z_{j} \end{bmatrix} - \begin{bmatrix} \nu_{(W)} \\ \nu_{j} \\ \nu_{J} \end{bmatrix}$$

where $Q_{j}^{\frac{1}{2}w} = w_{j}$. This is the equation to be triangularized with subroutine THH, i.e.,

77-26

If the problem is arranged so that Q_J is diagonal one can reduce storage and computation. The form of this algorithm is designed to allow the use of singular Q_j matrices.

When the a priori for x and Q 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 $\overline{G} = G_J U^{(q)} = [g_1, \dots, g_n]$, $D^{(q)} = Diag(d_1, \dots, d_n)$
Apply subroutine AGTRN n_w times, with $\overline{U}_1 = \overline{U}_j$, $\overline{D}_1 = D_j$

$$(\overline{\overline{U}}-\overline{\overline{D}})_{k} ; d_{k}, g_{k} \xrightarrow{AGTRN} (\overline{\overline{U}}-\overline{\overline{D}})_{k+1}$$

$$k = 1, \dots, n_{w}$$

$$(\overline{\overline{U}}_{k}\overline{\overline{D}}_{k}\overline{\overline{U}}_{k}^{T} + d_{k}g_{k}g_{k}^{T} = \overline{\overline{U}}_{k+1}\overline{\overline{D}}_{k+1}\overline{\overline{U}}_{k+1}^{T})$$

Then $U_{j+1} = \overline{U}_{w}$, $D_{j+1} = \overline{D}_{w}$.

Certain filtering problems involve dynamic models of the form

77-26

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

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

$$\widetilde{x}_{j+1} = \Phi_j \hat{x}_j$$

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

$$\begin{bmatrix} \Phi_{j} & U_{j} & \bar{G} \end{bmatrix}$$
, Diag $(D_{j}, D^{(q)}) \xrightarrow{W \in S} (\widetilde{U}_{j+1} - \widetilde{D}_{j+1})$

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 ℓ 2Al used to copy A into A₁, which then can be modified as dictated by the analysis.

Covariance analyses sometimes are initialized using a covariance matrix from a different problem (or a differed 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

The 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, R12COV and UD2COV. The vector stored covariance results are displayed in a triangular format using subroutine TRIMAT. Aside: After careful consideration it was decided that subroutines to multiply matrices would not be included in our ESP. Our reasons are that parameter estimation does not, in the main, involve matrix multiplication; and when such products occur they generally involve matrices with special structures (viz. rectangle x triangle, triangle x rectangle, diagonal x triangle, etc). To see that these computations are not lengthy or complicated we illustrate how to compute z = Rxwhere R is a triangular vector stored matrix and x is an N vector,

77-26

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

21

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.

A later more encyclopedic subroutine directory may include the . various matrix products that occur in linear algebra applications.

End of Aside

III. SUBROUTINE DIRECTORY SUMMARY

1. AGTRN - (Agee-Turner)

Computes updated U-D factors corresponding to a rank 1 matrix modification; i.e., given U-D, a scalar c, and vector v, \overline{U} and \overline{D} are computed so that U D 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.

77-26

2. <u>A2A1</u> - (A to A1)

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

Example III.1

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^{*} , a_x^{*} , a_y^{*} , x, y, z, v_x , v_y , v_z , GM and station location errors. A2Al 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.

3. <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 <u>that</u> 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.





4

in track and cross track accelerations

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.

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

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. C2C - (C to C)

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

Example III.4

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

25

common practice to omit a priori information on parameters that are either poorly known or which will be well determined by the data.

77-26

8. 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.

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

Computes the inverse of an upper triangular vector stored matrix R using subroutine UTINV. 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 $CNB/N \le condition$ number $\le CNB$. When this bound is within several orders of magnitude of the machine accuracy the computed inverse is not to be trusted, (viz if $CNB \ge 10^{15}$ on an 18 decimal digit machine R is ill-conditioned).

10. 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, COVOUT can overwrite RINV.

11. 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.

26
Example III.5

В	С	D	Ε		Έ	F	С	В
4	8	14	22 -		22	0	8	4
6	10	16	24		24	0	10	6
0	12	18	26	R2A	26	0	12	0
0	0	20	28		28	0	0	0
0	0	0	30		30	0	0	0
	R		-	I	L.		A	
	В 6 0 0	 B C 4 8 6 10 0 12 0 0 0 0 0 R 	 B C D 4 8 14 6 10 16 0 12 18 0 0 20 0 0 R 	B C D E 4 8 14 22 6 10 16 24 0 12 18 26 0 0 20 28 0 0 0 30 R - - -	B C D E 4 8 14 22 6 10 16 24 0 12 18 26 0 0 20 28 0 0 0 30 R	B C D E E 4 8 14 22 6 10 16 24 0 12 18 26 0 0 20 28 0 0 0 30	B C D E E F 4 8 14 22 0 24 0 6 10 16 24 24 0 24 0 0 12 18 26 R2A 26 0 0 0 20 28 0 30 30 30 R	B C D E E F C 4 8 14 22 22 0 8 6 10 16 24 24 24 0 10 0 12 18 26 R2A R2A 26 0 12 0 0 20 28 0 30 30 0 0 R R A A A A

77-26

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, sensitivites, etc. can be obtained; or so that data sets containing different parameters can be combined.

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

A triangular portion of the vector stored upper triangular matrix R 1s 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.

* see also the aside in the introduction



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.

13. <u>RUDR</u> - (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.

28

77-26

14. THH - (Triangular Householder data packing)

An upper triangular vector stored matrix R is combined with a rectangular doubly subscripted matrix A by means of Householder orthogonal transformations. The result overwrites R, and A is destroyed in the process.



15. TRIMAT - (Triangular matrix print)

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

	Α	В	С	D
A	2	4	8	14
В		6	10	16
С			12	18
Ð				20

(The numbers are printed to 8 significant floating point digits).

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

16. <u>TTHH</u> - (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.

The elements are not explicitly set to zero.



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

77-26

Example III.8

To zero row 2 and 3 of R(15), in the example of subroutine 11.

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

$$R(15) = (2,4,0,8,0,0,14,0,0,20,22,0,0,28,30)$$

1.e.,

4					_						-
	2	4	8	14	22		2	4	8`	14	22
-	0	б	10	16	24		0	0	0	0	0
	0	0	12	18	26	TZERO	0	0	0	0	0
	0	0	0	20	28		0	0	0	20	28
	0	0	0	0	30		0	0	0	0	30
1					_		_				-
	R-vector stored							R-vec	tor	store	ed

* The elements are not explicitly set to zero.

18. UDMES - (U-D measurement update)

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

19. 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.

20. 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.

21. 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 tup of RIN has a -1 set in then ROUT will have the solution in the place of the right hand side.

77-26

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

77-26

INPUT OUTPUT

$$n_{y} \begin{bmatrix} R & R \\ -\frac{R}{x} & \frac{xy}{y} \\ 0 & R^{-1} \\ y \end{bmatrix} \underbrace{\text{UTIROW}}_{y} \begin{bmatrix} R^{-1} & -R^{-1}_{x} & R^{-1}_{y} \\ -\frac{R^{-1}_{x} & -R^{-1}_{x} & R^{-1}_{y} \\ 0 & R^{-1} \\ y \end{bmatrix}$$

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 <u>sensitivity</u> 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 <u>perturbation</u>. 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 equivalent to UTINV.

23. WGS - (Weighted Gram Schmidt matrix triangularization)

An input rectangular (possibly square) matrix W and a diagonal weight matrix, $D_{_{\rm UV}}$, 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. AGTRN (Agee-Turner U-D rank one modification)

Purpose

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

CALL AGTRN (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 result UOUT=UIN is allowed
N	Matrix dimension
С	Input scalar, destroyed by the algorithm
V(N)	Input vector, 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. Functional Description

This rank one modification is based on a result published by Agee and Turner (1972), White Sands Missile Range Tech. Report No. 38. See also Ref. [3] where the algorithm is derived using geometric arguments.

2. 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 Al, IAl.GE.IR
LAI	Number of columns in Al; this also represents the number of parameter names associated with Al
NAMA1(LA1)	Input list of parameter names to be associated with the output matrix Al

Remarks and Restrictions

Al <u>cannot</u> 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 Al in an order corresponding to the NAMAl parameter namelist. Columns of zeros are inserted in those Al columns which do not correspond to names in the input parameter namelist NAMA. 3. COMB0 (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 Defi	nitions					
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 (1.2)		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 out- put A matrix; consists of names in NAM1 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. 4. COV2RI (Covariance to Cholesky Square Root, RI)

Purpose

To construct the upper triangular Cholesky factors of a positive semi-definite matrix. Both the input covariance and the output Cholesky factor (square root) are vector stored. The output overwrites the input. Covariance (input) = U*U**T (output U = Rinverse).

CALL COV2RI(U,N)

Argument Definitions

U(N*(N+1)/2)

Contains the input vector stored covariance matrix (assumed positive definite) and on output it contains the upper triangular square root factor

Ν

Dimension of the matrices involved

Remarks and Restrictions

No check is made that the input matrix is positive semidefinite. 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].

U(input) = U(output) * U(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 covari-
	ance matrix; on output it contains the
	vector stored U-D covariance factors.

Ν

Matrix dimension

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].

 $U(input) = U*D*U^T$, U-D stored in U on output 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.CE.L = MAX(L1,L2)
Ll	No. of parameter names associated with the input C
NAM1(L)	Parameter names associated with C on input. (Only the first Ll 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

Ł

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.

39

۰.

7. INF2R (Information matrix to R)

Purpose

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

CALL INF2R(P,N)

Argument Definitions

P(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

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].

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.

8. 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.

CITTE I DECIDI (TI STILLS TICS TILS ANTHIN STAR STAR STAR STAR	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
Ll	Number of parameter names associated with the input A matrix
NAM1 (L)	Parameter names associated with A on input (only the first Ll 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.

9. 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
ROUT(N*(N+1)/2)	Output vector stored matrix inverse (RIN = ROUT is permitted)
CNB	Condition number bound. If K is the condition number of RIN, then CNB/N.LE.K.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. The matrix inversion is computed using subroutine UTINV.

Functional Description

The matrix inversion, a triangular back substitution, is accomplished via subroutine UTINV. If any diagonal element of the input R matrix is zero the inversion is not attempted; instead a message is printed. The condition number bound is computed as follows:

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

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

 $CNB/N \le condition$ number $R \le CNB$

is a simple consequence of the Frobenius norm inequalities given in Lawson-Hanson "Solving Least Squares," page 234. 10. RI2COV (RI Triangular to covariance)

Purpose

To compute the covariance matrix and/or the standard deviation of a vector stored upper triangular square root covariance matrix. The output covariance matrix, also vector stored, may overwrite the input.

•

CALL RI2COV(RINV,N,SIG,COVOUT,KOV)

Argument Definitions

RINV(N*(N+1)/2)		Input vector stored upper triangular covariance square root (RINV=R inverse 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)
(.GT.O	Compute covariance and sigmas using the first KOV rows of RINV
KOV).LT.0	Compute only the sigmas using the first KOV rows of RINV
	.EQ.0	No covariance, but all sigmas (e.g. use all N rows of RINV)

Remarks and Restrictions

Replacing N by KOV corresponds to computing the covariance of

a lower dimensional system.

Functional Description

COVOUT=RINV*RINV**T.

11. R2A (R to A)

Purpose

n

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	Row dimension of vector stored 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.

12. R2RA (Permute a subportion R_A of a vector stored triangular matrix) <u>Purpose</u>

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

Remarks and Restrictions

RA and NAMA can overwrite R and NAM. The meaning of the NRA = 0

used here denotes the output value of NRA.

option is clarified by the following example:



[†]see the concluding paragraph of Remarks and Restrictions

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

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



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

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

77-26

13. 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.O represents an R to U-D conversion and N.LT.O represents a U-D to R conversion.
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: UTINV

Functional Description

Consider the N>O case. RIN = R is transformed to ROUT = R inverse using subroutine UTINV 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 overwrites ROUT.

Tf N<O the input is assumed to be a U-D array. This array is converted to ROUT=UD^{$\frac{1}{2}$} and then using UTINV, 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.

14. THH (Triangular Householder Orthogonalization)

Purpose

То	coi	mput	e [Rz]	such	th	at
т		Ĩ	~z	=	R	۲z	T - orthogonal
		A	z		0	е	

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

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

Argument Definitions

R(N*(N+3)/2)	Input upper triangular vector stored square root information matrix. If estimates are involved SOS.GE.O and R is augmented with the right hand side (stored in the last N locations of R). If SOS.LT.O only the first N*(N+1)/2 locations of R are used. The result of the subroutine overwrites the input R
Ν	No. of parameters
A(M,N+1)	Input measurement matrix. The N+lst column is only used if SOS.GE.O, in which case it represents the right side of the equation $v + AX = z$. A is destroyed by the algorithm, but it is not explicitly set to zero.
IA	Row dimension of A
М	The number of rows of A that are to be combined with R $\$
SOS	Accumulated residual sum of squares corresponding to the data processed prior to this time. On exit SOS represents the updated sum of squares of the residuals $\sum_{i=1}^{2} z_{i}^{-A} \sum_{i=1}^{2} z_{i}^{2}$,
	summed over the old and new data. It also includes the a priori term
	$ \mathbf{R}_{o}\mathbf{X}_{est}-\mathbf{z}_{o} ^{2}$. Because SOS cannot
	be used if data, z, is not included we

not included.

NSTRT 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.

Perarks 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 ≠ 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, [1] and Bierman [3]. 15. TRIMAT (Triangular matrix print)

Purpose

To display a vector stored upper triangular matrix in a two dimensional 8-digit triangular format.

CALL TRIMAT (A, N, CAR, TEXT, NCHAR, NAMES)

Argument Definitions

A(N*N+1)/2)	Vector stored upper triangular matrix
N	Dimension of A
CAR(N)	Parameter names (alphanumeric) associated with A
TEXT (NCHAR)	An array of field data characters to be printed as a title preceding the matrix
NCHAR	No. of characters (including spaces) that are to be printed in text() ABS(NCHAR).LE.126.NCHAR negative is used to avoid skipping to a new page to start printing
NAMES	A logical flag. If NAMES=.F. the CAR namelist is ignored and the columns and rows of A on output appear with numerical column heads

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. 16. TTHH (Two triangular matrix Householder reduction)

Purpose

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



Argument Definitions

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

Remarks and Restrictions

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

17. 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



18. UDMES (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 $z = A^{T}x + v$ is processed, the covariance U-D factors and estimate (if included) are updated, and the Kalman gain and innovations variance are computed.

CALL UDMES(U,N,R,A,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 the state vector
R	Measurement variance
A(N)	Vector of Measurement coefficients; if data then A(N+1) = z
ALPHA	If ALPHA.LT.zero no estimates are computed (and X and z need not be included)
<u>OUTPUTS</u>	
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.
A	Contains U**T*A(input) and when ALPHA (input) is nonnegative A(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 sometimes introduces numerical errors.

The algorithm holds for R = 0 (a perfect measurement) but the code may fail (zero divides occur) if any of the ALPHA terms appearing in the code vanish. Changes in the code which remove the zero divide problems are commented in the code.

Functional Description

The algorithm updates the columns of U, from left to right, using Bierman's algorithm, cf Proc. 1975 Conf. Dec. Control, Houston, Texas, pp 337-346.

19. 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,N,POUT)

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.

20. 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
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.

Functional Description

If U and D are written as doubly subscripted matrices then

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

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

21. 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 per- mitted

Remarks and Restrictions

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

This routine can also be used to produce the solution to RX = Z. Place Z in column N+1 (viz. RIN(N*(N+1)/2+1) = Z(1), etc.), define RIN((N+1)(N+2)/2) = -1 and call the subroutine using N+1 instead of N. On return the first N entries of column N+1 contain the solution (e.g. ROUT(N*(N+1)/2+1) = X(1), etc.).

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].

22. 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 $R_x + R_{xy}y = z_x$ (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).

23. 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
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, ϕ_1 , ϕ_2 ,..., ϕ_{TW} , 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.
V. FORTRAN Subroutine Listings

SUBROUTINE AGTRN (UIN, UOUT, N, C, V) AGTRN010 С AGTRN020 Ċ AGEE-TURNER U-D FACTOR RANK 1 UPDATE AGTRN030 С AGTRN040 С (UOUT)*DOUT*(UOUT)**T=(UIN)*DIN*(UIN)**T+C*V*V**T AGTRN050 Ç AGTRN060 C INPUT VECTOR STORED POSITIVE SEMI-DEFINITE U-D UIN(N*(N+1)/2)AGTRN070 С ARRAY, WITH D ELEMENTS STORED ON THE DIAGONAL AGTRN080 Ċ UOUT(N*(N+1)/2) OUTPUT VECTOR STORED POSITIVE (POSSIBLY) SEMI-AGTRN090 C DEFINITE U-D RESULT. UOUT=UIN IS PERMITTED AGTRN100 C C DIMENSION OF THE STATE AGTRN110 N SCALAR. SHOULD BE NON-NEGATIVE С AGTRN120 с с с C IS DESTROYED DURING THE PROCESS INPUT VECTOR FOR RANK ONE MODIFICATION. V IS AGTRN130 V(N)AGTRN140 DESTROYED DURING THE PROCESS AGTRN150 AGTRN160 COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, FEB, 1977) AGTRN170 AGTRN180 IMPLICIT DOUBLE PRECISION (A-H, 0-Z) AGTRN190 DIMENSION UIN(1), UOUT(1), V(1) AGTRN200 AGTRN210 Z=0.0 AGTRN220 IF (C.EQ.Z) RETURN AGTRN230 AGTRN240 JJ=N*(N+1)/2 AGTRN250 DO 50 J=N+2+-1 AGTRN260 S=V(J) AGTRN270 D=UIN(JJ)+C*S*S AGTRN280 IF (D) 5,10,30 AGTRN290 5 WRITE (6+100) AGTRN300 RETURN AGTRN310 10 しし=しし-し AGTRN320 WRITE (6:110) AGTRN330 DO 20 K=1,J AGTRN340 20 UOUT(JJ+K)=ZAGTRN350 GO TO 50 AGTRN360 30 B=C/D AGTRN370 BETA=S*B AGTRN380 C=B*UIN(JJ) AGTRN390 1001(11)=DAGTRN400 しし=しし-し AGTRN410 JM1=J-1 AGTRN420 DO 40 I=1.JM1 AGTRN430 V(I)=V(I)=S*UIN(JJ+I)AGTRN440 40 UOUT(JJ+I)=UIN(JJ+I)+BETA*V(I) AGTRN450 50 CONTINUE AGTRN460 AGTRN470 UOUT(1)=UIN(1)+C*V(1)**2 AGTRN480 RETURN AGTRN490 AGTRN500 100 FORMAT (1H0,10X, ** * ERROR RETURN DUE TO A COMPUTED NEGATIVE COMAGTRN510 1PUTED DIAGONAL IN AGTRN * * **) AGTRN520 110 FORMAT (1H0,10X, ** * NOTE: U-D RESULT IS SINGULAR * * **) AGTRN530 END AGTRN540

¢

Ċ

С

С

С

C

С

77-26

64

		SUBROUTINE A2A1 (A,IA,IR,LA,NAMA,A1,IA1,LA1,NAMA1)	
С			A2A10010
С		SUBROUTINE TO REARRANGE THE COLUMNS OF A (IR, LA), IN NAMA ORDER	A2A10020
С		AND PUT THE RESULT IN A1(IR,LA1) IN NAMA1 ORDER. ZERO COLUMNS	A2A10030
C		ARE INSERTED IN AT CORRESPONDING TO THE NEWLY DEFINED NAMES.	A2A10040
C			A2A10050
С		A(IR/LA) INPUT RECTANGULAR MATRIX	A2A10060
С		IA ROW DIMENSION OF A. IR.LE.IA	A2A10070
С		IR NO. OF ROWS OF A THAT ARE TO BE REARRANGED	A2A10080
С		LA NO. OF PARAMETER NAMES ASSOCIATED WITH A	A2A10090
С		NAMA(LA) PARAMFTER NAMES ASSOCIATED WITH A	A2A10100
C		A1(IR,LA1) OUTPUT RECTANGULAR MATRIX	A2A10110
C		A AND A1 CANNOT SHARE COMMON STORAGE	A2A10120
С		IA1 ROW DIMENSION OF A1, IR.LE.IA1 -	A2A10130
С		LA1 NO. OF PARAMETER NAMES ASSOCIATED WITH A1	A2A10140
С		NAMA1(LA1) INPUT LIST OF PARAMETER NAMES TO BE ASSOCIATED	A2A10150
С		WITH THE OUTPUT MATRIX A1	A2A10160
С			A2A10170
С		COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976)	A2A10180
Ç			A2A10190
		DIMENSION A(IA,1), NAMA(1), A1(IA1,1),NAMA1(1)	A2A10200
		IMPLICIT DOUBLE PRECISION (A-H,O-Z)	A2A10210
С			A2A10220
		ZERO=0.	A2A10230
		D0 100 J=1,LA1	A2A10240
		DO 60 I=1,LA	A2A10250
		IF (NAMA(I)•EQ•NAMA1(J)) GO TO 80	A2A10260
	60	CONTINUE	A2A10270
		DO 70 K=1,IR	A2A10280
	70	A1(K,J)=ZERO @ ZERO COL. CORRES. TO NEW NAME	A2A10290
		GO TO 100	A2A10300
	80	DO 90 K=1,IR	A2A10310
	90	A1(K,J)=A(K,I) 🛛 COPY COL. ASSOC. WITH OLD NAME	A2A10320
	100	CONTINUE	A2A10330
С			A2A10340
		RETURN	A2A10350
		END	A2A10360

SUBROUTINE COMBO (R,L1,NAM1,L2,NAM2,A,IA,LA,NAMA) С С TO REARRANGE A VECTOR STORED TRIANGULAR MATRIX AND STORE COMB0010 č THE RESULT IN MATRIX A. THE DIFFERENCE BETWEEN THIS SUR-COMB0020 ROUTINE AND R2A IS THAT THERE THE NAMELIST FOR A IS INPUT. COMB0030 HERE IT IS DETERMINED BY COMBINING THE LIST FOR R WITH COMB0040 A LIST OF DESIRED NAMES. COMB0050 COMB0060 R(L1*(L1+1)/2)INPUT VECTOR STORED UPPER TRIANGULAR MATRIX COMB0070 NO. OF PARAMETERS IN R (AND IN NAM1) L1 COMB0080 NAM1(L1) NAMES ASSOCIATED WITH R COMB0090 NO. OF PARAMETERS IN NAM2 COMB0100 L2 NAM2(L2) PARAMETER NAMES THAT ARE TO BE COMBINED WITH R COMBOLLO (NAM1 LIST), THESE NAMES MAY OR MAY NOT BE IN COMB0120 NAM1. COMB0130 OUTPUT ARRAY CONTAINING THE REARRANGED A(L1/LA) COMB0140 R MATRIX, L1.LE.IA. COMB0150 ROW DIMENSION OF A IA COMB0160 LA NO. OF PARAMETER NAMES IN NAMA, AND THE COMB0170 COLUMN DIMENSION OF A. LA=L1+L2-NO. NAMES COMB0180 COMMON TO NAM1 AND NAM2. LA IS COMPUTED AND COMB0190 OUTPUT. COMB0200 NAMA(LA) PARAMETER NAMES ASSOCIATED WITH THE OUTPUT A COMB0210 MATRIX. CONSISTS OF NAMES IN NAM1 NOT IN COMB0220 NAM2 FOLLOWED BY NAM2. COMB0230 COMB0240 COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976) COMB0250 ¢ COMB0260 IMPLICIT DOUBLE PRECISION (A-H, 0-Z) COMB0270 DIMENSION R(1), A(IA,1), NAM1(1), NAM2(1), NAMA(1)COMB0280 С COMB0290 ZER0=0.0 COMB0300 K=1 COMB0310 DO 100 I=1,L1 COMB0320 DO 50 J=1,L2 COMB0330 IF (NAM1(I).EQ.NAM2(J)) GO TO 100 COM80340 50 CONTINUE COMB0350 NAMA(K)=NAM1(I) COMB0360 JJ=I*(I-1)/2 COMB0370 D0 60 L=1,I COMB0380 $A(L_{iK})=R(JJ+L)$ 60 COMB0390 IF (I.EQ.L1) GO TO 80 COMP0400 IP1 = I+1COMB0410 D0 70 L=IP1+L1 COMB0420 70 A(L,K) = ZEROCOMB0430 K=K+1 80 COMB0440 100 CONTINUE COMB0450 C NAMES UNIQUE TO NAM1 ARE NOW IN NAMA COMB0460 D0 200 J=1.L2 COMB0470 DO 150 I=1+L1 COMB0480 IF (NAM2(J).EQ.NAM1(I)) GO TO 170 COMB0490 150 CONTINUE COM80500 NAMA(K)=NAM2(J) COMB0510 DO 160 L=1.L1 COMB0520 160 A(L+K)=ZERO COMB0530

77=26

	••• =-	
С	NAMES UNIQUE TO NAM2 ARE NOW IN NAMA	COMB0540
	GO TO 190	COMB0550
	170 NAMA(K)=NAM2(J)	COMB0560
С	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
	ÎP1=I+1	COMB0620
	D0 185 L=IP1+L1	COMB0630
	185 A(L+K)=ZERO	COM80640
	190 K=K+1	COMB0650
	200 CONTINUE	COMB0660
	ĽA=K-1	COMB0670
С	NAMES MUTUAL TO NAME AND NAME ARE NOW IN NAMA	COMB0680
	RETURN	COM80690
	END	COMB0700

		SUBROUTINE COV2R	I(U,N)		
С				, 3	COV2R010
C		TO CONSTRUCT	THE UPPER TRIANGULAR	CHOLESKY FACTOR OF A	C0V2R020
C		POSITIVE SEMI	-DEFINITE MATRIX. BOT	H THE INPUT COVARIANCE	COV2R030
С		AND THE OUTPU	T CHOLESKY FACTOR (SQ	UARE ROOT) ARE VECTOR	C0V2R040
С		STORED. THE O	UTPUT OVERWRITES THE	INPUT.	CoV2R050
ç		COVARIANCE(IN	PUT)=U*U**T (U IS OU	TPUT).	CoV2R060
С				-	CoV2R070
С		IF THE INPUT (COVARIANCE IS SINGULA	R THE OUTPUT FACTOR HAS	CoV2R080
С		ZERO COLUMNS+			CoV2R090
С					COV2R100
С		U(N*(N+1)/2)	CONTAINS THE INPUT V	ECTOR STORED COVARIANCE	COV2R110
С			MATRIX (ASSUMED POSI	TIVE DEFINITE) AND ON OUTPUT	CoV2R120
С			IT CONTAINS THE UPPE	R TRIANGULAR SQUARE ROOT	CoV2R130
C			FACTOR.		CoV2R140
С		N	DIMENSION OF THE MAT	RICES INVOLVED	CoV2R150
Ċ					COV2R160
Ċ		COGNIZANT PER	SONS: G.J.BIFRMAN/M.	W.NEAD (JPL, FEB, 1977)	COV2R170
č					CoV2R180
•		TMPLICIT DOUBLE	PRECISION (A-H.O-Z)		COV2R190
		DIMENSION U(1)			COV2R200
С					COV2R210
-		7EB0=0.0			COV2R220
		ONET1-			COV2R230
		$(1)=1 \times (1)+1 /2$			Cov2R240
					CoV2R250
c		5611-56			COV2R260
Ŭ		DQ 5 1=N+2+-1			CoV2R270
		TE (11(.1.1).1 T	7580) 11(Jd)-7580		COV28280
		11(11)= SORT(COV2R290
			ZERO) AL PHA-ONE/ULL)	COV2R300
ĉ		1P (0(00).01	• ZEROT REFINE ONE OU	1	COV2R300
C		VK-0			COV2R310
				O NEXT DIACONAL	COV2R320
				M NEXT DIAGONAL	COV2R330
					COV2R330
					COVERSED
			K)		COV2R570
	-	DO 3 1=1			COV2R380
	3	U(KK+1)=0(KK+1)=5*((JJN+1))	[V] KK+I=(I)K)	COV2R390
	4	KK=KK+K			COV2R400
	5				COV2R410
		IF (U(1).LT.ZERO) U(1)=ZEKO		COV2R420
		U(1) = SQRT(U(1))			COV2R430
C					COV2R440
		RETURN			COV2R450
		END			COV2R460

- c

	SUBROUTINE COV2UD (U.N)	
	TO OBTAIN THE U-D FACTORS OF A POSITIVE SEMI-DEFINITE MATRIX. THE INPUT MATRIX VECTOR STORED IS OVERWRITTEN BY THE OUTPUT U-D FACTORS WHICH ARE ALSO VECTOR STORED.	CoV2U010 CoV2U020 CoV2U030 CoV2U040 CoV2U040
	U(N*(N+1)/2) CONTAINS INPUT VECTOR STORED COVARIANCE MATRIX. ON OUTPUT IT CONTAINS THE VECTOR STORED U-D COVARIANCE FACTORS. N MATRIX DIMENSION	COV2U060 COV2U070 COV2U080 COV2U080 COV2U090
	SINGULAR INPUT COVARIANCES RESULT IN OUTPUT MATRICES WITH ZERO COLUMNS	COV20100 COV20110 COV20120 COV20130 COV20130
	COGNIZANT PERSONS: G.J.BIFRMAN/R.A.JACORSON (JPL, FEB. 1977)	COV2U150 COV2U160
	IMPLICIT DOUBLE PRECISION (A-H,0-Z)	CoV2U170 CoV2U180
	7=0-0	COV2U190 COV2U200
	ONE=1.0	COV2U220 COV2U220 COV2U230
	JJ=N*(N+1)/2 D0 50 J=N+2+-1	CoV2U240 CoV2U250
	ALPHA=Z IF (U(JJ).LT.Z) U(JJ)=Z IF (U(JJ).CT.Z) ALBUATONS(U(J))	CoV2U260 CoV2U270
	IF (0(00).0(.2) ALPHA-ONE/((0)) JJ=JJ-J KK=0	C0V2U290 C0V2U290
	KJ=JJ JM1=J−1	Cov2U310 Cov2U320
	DO 40 K=1,JM1 KJ=KJ+1	CoV2U330 CoV2U340
	BETA=U(KJ) U(KJ)=ALPHA*U(KJ) TJ=JJ	COV2U350 COV2U360
	IK=KK D0 30 I=1,K	CoV2U380 CoV2U390
	IK=IK+1 IJ=IJ+1	COV2U400 COV2U410
30 40 50	U(IK)=U(IK)=BETA*U(IJ) KK=KK+K CONTINUE	COV2U420 COV2U430
50	IF (U(1).LT.7) U(1)=7 RETURN	COV2U450 COV2U450
	END	COV2U470

С

С

С

٠

REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

SUBROUTINE C2C (C,IC,L1,NAM1,L2,NAM2) C2C00010 SUBROUTINE TO REARRANGE THE ROWS AND COLUMNS OF MATRIX C2C00020 C(L1+L1) IN NAM1 ORDER AND PUT THE RESULT IN C2C00030 C(L2+L2) IN NAM2 ORDER. ZERO COLUMNS AND ROWS ARE C2C00040 ASSOCIATED WITH OUTPUT DEFINED NAMES THAT ARE NOT CONTAINED C2C00050 IN NAM1. C2C00060 C2C00070 INPUT MATRIX C(L1,L1) C2C00080 IC ROW DIMENSION OF C, IC.GE.L=MAX(L1.L2) CSC00090 NO. OF PARAMETER NAMES ASSOCIATED WITH THE INPUT C C2C00100 L1 NAM1(L) PARAMETER NAMES ASSOCIATED WITH C ON INPUT. (ONLY C2C00110 THE FIRST L1 ENTRIES APPLY TO THE INPUT C) C2C00120 NO. OF PARAMETER NAMES ASSOCIATED WITH THE OUTPUT CC2C00130 1.2 NAM2(L2) PARAMETER NAMES ASSOCIATED WITH THE OUTPUT C C2C00140 C2C00150 COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, SEPT, 1976) C2C00160 C2C00170 IMPLICIT DOUBLE PRECISION (A-H, 0-Z) C2C00180 C2C00190 DIMENSION C(IC,1), NAM1(1), NAM2(1) C2C00200 ZERO=0. C2C00210 L=MAX(L1,L2)C2C00220 IF (L.LE.L1) GO TO 5 C2C00230 NM=L1+1 C2C00240 DO 1 K=NM+L C2C00250 NAM1(K) = ZFROD ZERO REMAINING NAM1 LOCNS C2C00260 1 5 DO 90 J=1,L2 C2C00270 DO 10 I=1,L C2C00280 IF (NAM1(I).EQ.NAM2(J)) GO TO 30 C2C00290 CONTINUE C2C00300 10 GO TO 90 C2C00310 IF (I.EQ.J) GO TO 90 30 C2C00320 DO 40 K=1+L C2C00330 H=C(K+J) INTERCHANGE COLUMNS I AND J C2C00340 C(K,J)=C(K,I)C2C00350 40 C(K+I)=H C2C00360 DO 80 K=1,L C2C00370 $H=C(J_{i}K)$ **D** INTERCHANGE ROWS I AND J C2C00380 C(J,K)=C(I,K)C2C00390 80 C(I,K)=H C2C00400 D INTERCHANGE LABELS I AND J C2C00410 NM=NAM1(I) NAM1(I)=NAM1(J) C2C00420 C2C00430 NAM1(J)=NM 90 CONTINUE C2C00440 C2C00450 FIND NAM2 NAMES NOT IN NAM1 AND SET CORRESPONDING ROWS AND C2C00460 COLUMNS TO ZERO C2C00470 C2C00480 DU 120 J=1,L2 C2C00490 DO 100 I=1,L C2C00500 IF (NAM1(I).EQ.NAM2(J)) GO TO 120 C2C00510 100 CONTINUE C2C00520 DO 110 K=1+L2 C2C00530 C(J,K)=ZFRO C2C00540 110 C(K+J)=ZERO C2C00550 120 CONTINUE C2C00560 C2C00570 70 RETURN C2C00580 FND C2C00590

C

Ç

C C

C

C C

С

C	SUBROUTINE IN	F2R (P+N)			INF2R010
C				-D	INF2R020
ç	TO CHOLES	KY FACTOR AN	INFORMATION MAT	RIX	INFERUOU
C					INF2RU4U
C	COMPUTES A LO	WER TRIANGULA	R VECTOR STORED	CHOLESKI FACTORIZATIO	N INFERDO
C	OF A PUSITIVE	SEMI-OFFINI	E MATRIX. PERV	FCH TC OVER IRIANGUL	AK . INF 2RUDU
Č	BUTH MATRICES	ARE VECTOR 5	TURED AND THE R	ESULIS OVERWRITES	1NF2RU/U
Š	THE INPUT				INFERDOD
L C	D(11, (11, 11, (0))				INF2RU9U
5	P(N*(N+1)/2)	ON INPUT THI	S IS A POSITIVE	SEMITUREINITE MATRIXE	THEODIIA
		AND ON OUTPU	1 II IS A IRIAN	COLAR FACTORS IF THE	THEORIOO
C A		INPUT MATRIX	A IS SINGULAR TH	E OUIPUI MAIRIX WILL	INFERIEU
ç		HAVE ZERO DI	AGONAL ENTRIES	NET	INFERIO
C C	N	DIMENSION OF	MATRICES INVUL	.VED	10528140
L C				(10) (550 1077)	INFERIOU
C C	CUGNIZANI PER	SON: G.J.BIER	MANZM WINEAU		1NF2R100
C			(1.1).0-7)		INFZKI/U
~	IMPLICIT DOOR	LE PRECISION	(A-H+U-Z)		10550100
C					INF2R190
~	DIMENSION P(1	1			INFZRZUU
C					
	Z=U+U				INFZRZZU
	ONE=1+U				INF2R250
	JJ=0				INF2R240
	NN=N*(N+1)/2				INF2R250
	NM1=N-1				1NF2R260
	DO 10 J=1,NM1	,			INF2R270
			ີ 10=(ງາ	(J)	1NF2R280
	IF (P(JJ).L	.T.Z) P(JJ)=Z	-		INF2R290
	P(JJ)=SQRT(P(JJ))			INF2R300
	ALPHA=Z				INF2R31U
	IF (P(JJ)+G	T.Z) ALPHA=ON	4E/P(JJ)	<i>¥</i>)	1NF2R320
	JK=NN+J			• K J	1NF2K330
	1+0=140				INF2K340
		104 . 1		JII SIARI	INF2R00U
		15.14=1			INF/KODU INF/KODJIO
					INF2RJ/U
		HA*P(UK)			INF2R380
	BETATPLUK				INF2R390
		* 1/ *			INF2R41U
	DO IU I=N	J # K # - 1			INF2R42U
	KI=KI=1				
		⊾ \///Т_D////\~∩/	CT A		11152K44U
~	10 P(KI)=F	(KI)=P(OI)*B	11A		
C		7) 5/5/11-7			
	IF (P(NN)+L)+	121 PUNNJ=2			INF2K4 (U
	PUNNJESURICPU	NN//			INFZK48U
					INF2R49U
	END				THESTOR

PERMU530

SUBROUTINE PERMUT (A+IA+IR+L1+NAM1+L2+NAM2) PERMU010 SUBROUTINE TO REARRANGE PARAMETERS OF A(TR+L1), NAM1 ORDER PERMU020 TO A(IR, L2), NAM2 ORDER. ZERO COLUMNS ARE INSERTED PERMU030 CORRESPONDING TO THE NEWLY DEFINED NAMES. PFRMU040 PERMU050 INPUT RECTANGULAR MATRIX, L=MAX(L1,L2) PERMU060 A(IR+L) ROW DIMENSION OF A. IA.GE.IR PFRMU070 ΪA NUMBER OF ROWS OF A THAT ARE TO BE REARRANGED PERMU080 IR NUMBER OF PARAMETER NAMES ASSOCIATED WITH THE INPUT PERMU090 L1 PERMU100 A MATRIX PARAMETER NAMES ASSOCIATED WITH A ON INPUT PERMU110 NAM1(L) (ONLY THE FIRST L1 ENTRIES APPLY TO THE INPUT A) PERMU120 NUMBER OF PARAMETER NAMES ASSOCIATED WITH THE OUTPUT PERMU130 L2 PERMU140 A MATRIX PARAMETER NAMES ASSOCIATED WITH THE OUTPUT A PERMU150 NAM2 PERMU160 COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976) PERMU170 PFRMU180 IMPLICIT DOUBLE PRECISION (A-H, 0-Z) PERMU190 PERMU200 DIMENSION A(IA,1), NAM1(1), NAM2(1) PERMU210 PFRMU220 ZERO=0. L=MAX(L1,L2)PFRMU230 IF (L.LE.L1) GO TO 50 PERMU240 PFRMU250 NM=L1+1 PERMU260 DO 40 K=NM+L R ZERO REMAINING NAM1 LOCS PERMU270 ЦN NAM1(K)=0 50 DO 100 J=1,L2 PFRMU280 PERMU290 DO 60 I=1/L PERMU300 IF (NAM1(I).EQ.NAM2(J)) GO TO 65 PFRMU310 CONTINUE 60 GO TO 100 PERMU320 65 CONTINUE PERMU330 PERMU340 IF (I.EQ.J) GO TO 100 DO 70 K=1.IR PERMU350 □ INTERCHANGE COLS I AND J PFRMU360 W=A(K + J)A(K,J) = A(K,I)PFRMU370 PERMU380 70 $A(K \cdot I) = W$ NM=NAM1(I) R INTERCHANGE I AND J COL. LABELS PERMU390 PFRMU400 NAM1(I)=NAM1(J) PERMU410 NAM1(J)=NM PERMU420 100 CONTINUE REPEAT TO FILL NEW COLS PFRMU430 DO 200 J=1+L2 PFRMU440 DO 160 I=1,L PERMU450 PERMU460 IF (NAM1(I).E0.NAM2(J)) G0 TO 200 CONTINUE PFRMU470 160 PFRMU480 DO 170 K=1, IR 170 A(K,J) = ZEROPERMU490 CONTINUE PERMU500 200 PFRMU510 RETURN PERMU520

77-26

С

С

C

С

С

C

С

0000000

С

c c

Ċ

C

С

С

END

c			
-		SUBROUTINE RINCON (RIN, N, ROUT, CNB)	RINCO010
C			RINC0020
С		TO COMPUTE THE INVERSE OF THE UPPER TRIANGULAR VECTOR STORED	RINC0030
С		INPUT MATRIX RIN AND STORE THE RESULT IN ROUT. (RIN=ROUT IS	RINCO040
C		PERMITTED) AND TO COMPUTE A CONDITION NUMBER ESTIMATE.	RINC0050
ç		CNB=FROB.NORM(R)*FROB.NORM(R**-1).	RINC0060
C		THE FROBENIUS NORM IS THE SQUARE ROOT OF THE SUM OF SQUARES	RINCO070
C C		AN HOUSE DOLLING AND IT ACTS AS A LOWER BOUND IS USED AS	RINCOURU
č		AN UPPER BUUND AND IT ACTS AS A LUNER RUUND UN THE ACTUAL	RINC0100
č		CONDITION NONDER OF THE PROOF ME ASE THE BROK SOLVING LEAST	RTNC0110
č		SWOARES I DI LANSON AND HANDON)	RINCOID
č		RIN(N*(N+1)/2) INPUT VECTOR STORED UPPER TRIANGULAR MATRIX	RINC0130
č		N DIMENSION OF R MATRICES	RINC0140
Ċ		ROUT(N*(N+1)/2) OUTPUT VECTOR STORED UPPER TRIANGULAR MATRIX	RINCO150
Ċ		INVERSE (RIN=ROUT IS PERMITTED)	RINCO160
С		CNB CONDITION NUMBER BOUND	RINCO170
С			RINCO180
Ç		COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL/FEB.1977)	RINC0190
С			RINCO200
С		SUBROUTINES REQUIRED: UTINV	RINCO210
С			RINCO220
		IMPLICII DOUBLE PRECISION (A-H+0+Z)	RINC0230
~		DIMENSION RIN(1), ROUT(1)	RINCO240
L		7=0.0	RINCO250
		2-0+0 NTOT-N±(N+1)/2	RINCO280
c			RINCO2AD
<u> </u>		RNM=7	RINC0290
			RTNC0300
	10	RNM=RNM+RIN(J) **2	RINCO310
С			RINC0320
		CALL UTINV (RININROUT)	RINC0330
		RNMOUT=Z	RINCO340
		DO 20 J=1+NTOT	RINC0350
~	20	RNMOUT=RNMOUT+ROUT(J)*+2	RINC0360
С			RINC0370
~		CNB=SQRT(RNM*RNMQUT)	RINC0380
C		WDITE (6.30) CUD	RINCU390
			RTNC0400
C			RINCO420
v	30	FORMAT(1H0,5X, CONDITION NUMBER BOUNDELLD18, 10, 2X, CNB/N.LE.CONDI	TRINC0430
		110N NUMBER.LE.CNB'//)	RINC0440
		END	RINC0450
		•	

SUBROUTINE RI2COV (RINV, N, SIG, COVOUT, KOV)

R12C0010 С TO COMPUTE THE COVARIANCE MATRIX AND/OR THE STANDARD DEVIATIONSRI2C0020 С OF A VECTOR STORED UPPER TRIANGULAR SQUARE ROOT COVARIANCE RI2C0030 С Ċ THE OUTPUT COVARIANCE MATRIX IS ALSO VECTOR STORED. R12C0040 MATRIX. С **RI2CO050** С INPUT VECTOR STORED UPPER TRIANGULAR COVARI- RI2C0060 RINV(N*(N+1)/2) С ANCE SQUARE ROOT. (RINV=R INVERSE IS THE RI2C0070 С INVERSE OF THE SRIF MATRIX) R12C0080 С DIMENSION OF THE RINV MATRIX RI2C0090 N С OUTPUT VECTOR OF STANDARD DEVIATIONS SIG(N) RT2C0100 RI2C0110 С COVOUT(N*(N+1)/2) OUTPUT VFCTOR STORED COVARIANCE MATRIX Ċ R12C0120 (COVOUT = RINV IS ALLOWED) COMPUTE COVARIANCE AND SIGMAS USING KOV ROWS RI2C0130 C KOV .GT.0 С R12C0140 OF RINV. č COMPUTE ONLY THE SIGMAS USING KOV ROWS OF R12C0150 .LT.0 С RI2C0160 RINV. Ĉ .EQ.0 NO COVARIANCE, BUT ALL SIGMAS (E.G. USE RI2C0170 С N ROWS OF RINV). RT2C0180 С RI2C0190 G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976) С COGNIZANT PERSONS: R12C0200 RT2C0210 С IMPLICIT DOUBLE PRECISION (A-H+0-Z) RI2C0220 DIMENSION RINV(1), SIG(1), COVOUT(1) RI2C0230 RJ2C0240 С RI2C0250 ZERO=0.0 RI2C0260 LIM=N IF (KOV.NE.0) LIM=IABS(KOV) RT2C0270 С ***** COMPUTE SIGMAS** RI2C0280 RI2C0290 IKS=0 RJ2C0300 D0 2 J=1,LIM IKS=IKS+J RI2C0310 SUM=ZER0 RT2C0320 IK=IKS RI2C0330 DO 1 K=J+N PI2C0340 SUM=SUM+RINV(IK)**2 PI2C0350 PT2C0360 1 IK=IK+K SIG(J)=SQRT(SUM) PI2C0370 2 RI2C0380 C IF (KOV.LE.0) RETURN R12C0390 С ***** COMPUTE COVARIANCE** R12C0400 ປປ=0 RI2C0410 RI2C0420 NM1=LIM-1 R12C0430 DO 10 J=1,NM1 RI2C0440 11=11+1 COVOUT(JJ) = SIG(J) * * 2**RI2C0450** R12C0460 IJS=JJ+J JP1=J+1 **PI2C0470** DO 10 I=JP1,N RI2C0480 IK=IJS RI2C0490 IMJ=I-J P12C0500 SUM=ZER0 RT2C0510 D0 5 K=I+N PT2C0520 IJK=IK+IMJ RT2C0530 PT2C0540 SUM=SUM+RINV(IK)*RINV(LUK) RJ2C0550 IK=IK+K 5 RI2C0560 COVOUT(IJS)=SUM RT2C0570 10 IJS=IJS+I R12C0580 IF (KOV.EQ.N) COVOUT(JJ+N)=SIG(N)**2 RT2C0590 С R12C0600 74 RETURN RI2C0610 END

		R2A00010
	TO PLACE THE TRIANGULAR VECTOR STORED MATRIX R INTO THE	R2A00020
	MATRIX A AND TO ARRANGE THE COLUMNS TO MATCH THE DESIRED	R2A00030
	NAMA PARAMETER LIST. NAMES IN THE NAMA LIST THAT DO NOT	P2A00040
	CORRESPOND TO ANY NAME IN NAME HAVE ZERO ENTRIES IN THE	R2A00050
	CORRESPONDING A COLUMN.	R2A00060
•		R2400070
	PULPEURALLY2) TNELT LEPED TETANGLUAR VECTOR STORED ARRAY	R2400080
		R24000000
	NAMP(I) DADAMETER NAMES ASSOCIATED WITH P. ONLY THE	R2400100
	THAT THE THE ADDITION AND THE THAT THE THE	R2400110
	A (TD J A) MATDIX TO DATE THE PEADANCED D MATDIX	R2400110
		R2400120
		R2400100
		R2400140
	NAMA (LA) DARAMETER NAMES FOR THE OUTPUT A MATRIX	R2400160
		R2A00170
	COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL SEPT. 1976)	R2A00180
		R2A00190
	TMPLICIT DOUBLE PRECISION (A=++.0=7)	R2A00200
	DIMENSION R(1), NAMR(1), A(IA, 1), NAMA(1)	R2A00210
		R2A00220
	ZERO=0.	R2A00230
	DO 5 J=1,LA	R2A00240
	DO 5 K=1+LR	R2A00250
5	$A(K,J) = ZERO$ \square ZERO $A(IR,LA)$	R2A00260
-	DO 40 J=1/LA	R2A00270
	DO 10 $I=1$, LR	R2A00280
	IF $(NAMR(I) \cdot FQ \cdot NAMA(J))$ GO TO 20	R2A00290
10	CONTINUE	F2A00300
	GO TO 40	R2A00310
20	JJ=I*(I-1)/2	R2A00320
	DO 30 K=1,I	R2A00330
30	$A(K \cdot J) = R(JJ + K)$	R2A00340
40	CONTINUE	R2A00350
		R2A00360
	RETURN	R2A00370
	END	R2A00380

SUBROUTINE R2A (R, LR, NAMR, A, IA, LA, NAMA)

С

¢

75

		SUBROUTINE R2RA (RINRINAMIRAINRAINAMA)	
C			R2RA0010
Ç		TO COPY THE UPPER LEFT (LOWFR RIGHT) PORTION OF A VECTOR	R2RA0020
С		STORED UPPER TRIANGULAR MATRIX R INTO THE LOWER RIGHT	R2RA0030
С		(UPPER LEFT) PORTION OF A VECTOR STORED TRIANGULAR	R2RA0040
C		MATRIX RA.	R2RA0050
С	r		R2RA0060
Ç		R(NR*(NR+1)/2) INPUT VECTOR STORED UPPER TRIANGULAR MATRIX	R2RA0070
C		NR DIMENSION OF R	R2RA0080
Ç		NAM(NR) NAMES ASSOCIATED WITH R	R2RA0090
Ç		RA(NRA*(NRA+1)/2) OUTPUT VFCTOR STORED UPPER TRIANGULAR MATRIX	R2RA0100
C		NRA DIMENSION ASSOCIATED WITH RA	P2RA0110
Ç		NAMA(NRA) NAMES ASSOCIATED WITH RA	R2RA0120
Ç			R2RA0130
ç		IF NRA=0 ON INPUT. THEN NAMA(1) SHOULD HAVE THE FIRST NAME OF THE	R2RA0140
C		OUTPUT NAMELIST AND THE NUMBER OF NAMES IN NAMA IS COMPUTED.	R2RA0150
Ç		THE LOWER RIGHT BLOCK OF R WILL BE THE UPPER LEFT BLOCK OF RA.	R2RA0160
C			R2RA0170
Č		IF NRA=LAST NAME OF THE UPPER LEFT BLOCK THAT IS TO BE MOVED,	R2RA0180
C		THEN THE UPPER BLOCK IS TO BE MOVED TO THE LOWER RIGHT POSITION.	R2RA0190
C		WHEN USED IN THIS MODE NRA=NR ON OUTPUT.	R2RA0200
C			R2RA0210
С С		THE NAMES OF THE RELOCATED BLOCK ARE ALSO MOVED. THE RESULT	R2RA0220
С С		CAN COINCIDE WITH R AND NAMA WITH NAM.	R2RA0230
Š			R2RA0240
		COGNIZANI PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, SEPT. 1976)	R2RA0250
C			R2RA0260
		$\frac{1000}{1000} = \frac{1000}{1000} = \frac{1000}{1000$	R2RAU270
		DIMENSION RUITRAUIT NAM(1) NAMA(1)	R2RA0280
c		EVOLAL 15	R2RAU290
6			R2RAU3UU
			R2RAU310
с	-		R2RAU32U
č		10HEB T. CONVER OF BA	R2RAU33U
Ý			R2RAU34U
			DODA0360
		TSE.TRUE	
С	•	ISSTRUE CORRESPONDS TO MOVING LOWER LET. CORNER OF R TO	82846380
č		UPPER RT. CORNER OF RA	RORANSON
-	1	DO 3 I=1+NR	R2R40400
	-	IF (NAM(I) FOLLOCN) GO TO 4	R2RA0400
	3	CONTINUE	R2RA0420
	-	WRITE (6,100)	R2RA0430
	100	FORMAT (1H0,20X, *NAMA(1) NOT IN NAMELIST OF R MATRIX*)	R2RA0440
		RETURN	R2RA0450
C			R2RA0460
	4	K=I	P2RA0470
		KM1=K-1	R2RA0480
		IF (IS) GO TO 15	R2RA0490
С			R2RA0500
		IJS=K*(K+1)/2-1	P2RA0510
		NRA=NR-K+1	R2PA0520
		IJA=0	R2RA0530
		KOLA=0	R2RA0540

77-26

			77-26	
		DO 10 KOL=K+NR		R2RA0550
		KOLA=KOLA+1		R2RA0560
		NAMA(KOL-KM1),=NAM(KOL)		R2RA0570
		DO 5 IR=1+KOLA		R2RA0580
		IJA=IJA+1		R2RA0590
	5	RA(IJA)=R(IJS+IR)		R2RA0600
	10	IJS=IJS+KOL		R2RA0610
		RETURN		R2RA0620
C				R2RA0630
	15	IJ=K*(K+1)/2		R2RA0640
		IJA=NR*(NR+1)/2		R2RA0650
		L=NR-KM1		R2RA0660
		KOL=K		P2RA0670
		DO 25 KOLA=NR+L+-1		R2RA0680
		IJS=IJA		R2RA0690
		NAMA (KOLA) = NAM (KOL)		R2RA0700
		DO 20 IR=KOLA,L,-1		R2RA0710
		RA(IJS)=R(IJ)		R2RA0720
		IJS=IJS-1		R2RA0730
	20	IJ=IJ-1		R2RA0740
		IJA=IJA-KOLA		R2RA0750
	25	KOL=KOL-1		R2RA0760
		NRA=NR		R2RA0770
C				- R2RA0780
		RETURN		R2RA0790
		END		R2RA0800

77_26

SUBROUTINE RUDR (RIN, N, ROUT, IS) С RUDR0010 C FOR N.GT.D THIS SUBROUTINE TRANSFORMS AN UPPER TRIANGULAR VECTOR RUDR0020 STORED SRIF MATRIX TO U-D FORM, AND WHEN N.LT.O THE U-D VECTOR RUDR0030 STORED ARRAY IS TRANSFORMED TO A VECTOR STORED SRIF ARRAY -RUDR0040 RUDR0050 RIN((N+1)*(N+2)/2)INPUT VECTOR STORED SRIF OR U-D ARRAY RUDR0060 ROUT((N+1)*(N+2)/2) OUTPUT IS THE CORRESPONDING U-C OR SRIF RUDR0070 ARRAY (RIN=ROUT IS PERMITTED) RUDR0080 ABS(N) = MATRIX DIMENSION RUDR0090 Ν THE (INPUT) SRIF ARRAY IS OUTPUT IN U-D FORM RUDRO100 N.GT.O THE (INPUT) U-D ARRAY IS OUTPUT IN SRIF FORM RUDRO110 N.LT.O THERE IS NO RT. SIDE OR ESTIMATE STORED IN RUDR0120 -IS = 0 COLUMN N+1, AND RIN NEFD HAVE ONLY, PUDR0130 N COLUMNS, I.E. RIN(N*(N+1)/2) RUDR0140 THERE IS A RT. SIDE INPUT TO THE SRIF AND RUDR0150 IS = 1 AN ESTIMATE FOR THE U-D ARRAY. THESE RESIDE RUDR0160 RUDR0170 IN COLUMN N+1. ¥ RUDR0180 . THIS SUBROUTINE USES SUBROUTINE UTINV RUDR0190 Č C RUDR0200 COGIZANT PERSONS G.J.BIERMAN/M.W.NEAD (JPL, FEB, 1977) RUDR0210 С RUDR0220 IMPLICIT DOUBLE PRECISION (A-H, 0-Z) RUDR0230 RUDR0240 DIMENSION RIN(1), ROUT(1) RUDR0250 С ONE= 1.0 PUDR0260 NP1 = IS + ABS(N)RUDR0270 ■ INITIALIZE DIAGONAL INDEX PUDR0280 ______1 RUDR0290 IDIMR= NP1*(NP1 +1)/2 IF (IS.EQ.1) RIN(IDIMR)= - ONF **RUDR0300** RUDR0310 С IF (N.LT.0) GO TO 30 RUDP0320 RUDR0330 CALL UTINV(RIN, NP1, ROUT) RUDR0340 ROUT(1) = ROUT(1) * * 2RUDR0350 IF (N.EQ.1) RETURN RUDR0360 D0 20 J=2.N S=ONE/ROUT (JJ+J) RUDR0370 ROUT(JJ+J) = ROUT(JJ+J)**2RUDR0380 RUDR0390 JM1=J-1 RUDR0400 DO 10 I=1, JM1 10 ROUT(JJ+I)= ROUT(JJ+I)*S RUDR0410 RIJDR0420 20 JJ=JJ+ J RETURN RUDR0430 С RUDR0440 RUDR0450 30 N≖-N REPRODUCIBILITY OF THE ROUT(1) = SQRT(RIN(1)) **RUDR0460** ORIGINAL PAGE IS POOR RUDR0470 IF(N.EQ.1) GO TO 60 RUDR0480 DO 50 J=2+N ROUT(JJ+J)= SQRT(RIN(JJ+J)) RUDR0490 S=ROUT(JJ+J) RUDR0500 JM1=J-1 RUDR0510 DO 40 I=1, JM1 RUDR0520 40 ROUT(JJ+I) = RIN(JJ+I)*SRUDR0530 50 JJ=JJ+J RUDR0540 60 CALL UTINV (ROUT , NP1, ROUT) RUDR0550 RUDR0560 С RUDR0570 78 RUDR0580 RETURN RUDR0590

77-26

. .

SUBROUTINE THH(R, N, A, IA, M, SOS, NSTRT)

THH00010 THIS SUBROUTINE PERFORMS A DOUBLE PRECISION TRIANGULARIZATION THH00020 OF A RECTANGULAR MATRIX INTO A SINGLY-SUBSCRIPTED ARRAY BY THH00030 APPLICATION OF HOUSEHOLDER ORTHONORMAL TRANSFORMATIONS. THH00040 THH00050 R(N*(N+3)/2) VECTOR STORED SQUARE ROOT INFORMATION MATRIX THH00060 (LAST N LOCATIONS MAY CONTAIN A RIGHT HAND SIDE) THH00070 N NUMBER OF PARAMETERS THH00080 A(IA N+1)MEASUREMENT MATRIX THH00090 IΑ ROW DIMENSION OF A THH00100 NUMBER OF OBSERVATIONS IN THIS BATCH Μ THH00110 **S0S** ACCUMULATED SUM OF SQUARES OF THE RESIDUALS THH00120 (Z-A*X(EST)**2), INCLUDFS A PRIORI THH00130 FIRST COL OF THE INPUT A MATRIX THAT HAS A NONZERO NSTRT THH00140 ENTRY. IF NSTRT.LE.1, IT IS SET TO 1. THIS OPTION THH00150 IS CONVENIENT WHEN PACKING A PRIORI BY BATCHES AND THH00160 THE A MATRIX HAS LEADING COLUMNS OF ZEROS. THH00170 THH00180 ON ENTRY R CONTAINS A PRIORI SQUARE ROOT INFORMATION FILTER (SRIF) THH00190 ARRAY, AND ON EXIT IT CONTAINS THE & POSTERIORI (PACKED) ARRAY.THH00200 ON ENTRY A CONTAINS OBSERVATIONS WHICH ARE DESTROYED BY THE ТЧН00210 INTERNAL COMPUTATIONS. THH00220 ON ENTRY IF SOS IS .LT. ZERO , PROGRAM WILL ASSUME THERE IS NO THH00230 RIGHT HAND SIDE DATA AND WILL NOT COMPUTE SOS OR USE LAST N THH00240 LOCATIONS OF VECTOR R. THH00250 THH00260 COGNIZANT PERSONS G.J.BIFRMAN/N.HAMATA (JPL, OCT.1975) THH00270 THH00280 IMPLICIT DOUBLE PRECISION (A-H, 0-Z) THH00290 DIMENSION A(IA,1),R(1) THH00300 DOUBLE PRECISION SUM THH00310 DATA ZERO/0.D0// GNE/1.D0/ THH00320 THH00330 IF (NSTRT.LE.O) NSTRT=1 THH00340 NP1=N+1 a NO. COLUMNS OF R THH00350 IF(SOS,LT.ZERO) NP1=N R NO COLS. = N IF SOS.LT.0 THH00360 KK=NSTRT*(NSTRT-1)/2 THH00370 @ J-TH STEP OF HOUSEHOLDFR REDUCTION DO 100 J=NSTRT,N THH00380 KK=KK+J THH00390 SUM=ZER0 THH00400 DU 20. I=1.M THH00410 20 SUM=SUM+A(I,J)**2 THH00420 IF (SUM.LE.ZERO) GO TO 100 D JF J-TH COL. OF A.EQ.O GO TO STEP J+1THH00430 SUM=SUM+R(KK)**2 THH00440 SUM=DSQRT(SUM) THH00450 IF(R(KK).GT.7ERO) SUM=-SUM THH00460 DELTA=R(KK)-SUM THH00470 R(KK)=SUM THH00480 BETA=ONE/(SUM*DELTA) THH00490 JJ=KK THH00500 ∟=J THH00510 J1=J+1 THH00520 ** READY TO APPLY J-TH HOUSFHOLDER TRANS. THH00530 THH00540 D0 40 K=J1,NP1

77-26

С

С

			77–26	THROOFEO
		JJ=JJ+L		100550
		L=L+1		THH00560
		SUM=DELTA*R(JJ)		THH00570
		DO 30 I=1+M		THH00580
	30	SUM=SUM+A(I+J)*A(I+K)		THH00590
		IF (SUM.EQ.ZERO) GO TO 40		ТНН00600
		SUM=SUM*RFTA		ТнН00610
		$R(J_{i}) = R(J_{i}) + SIM + nELTA$		THH00620
		D0 35 T=1+M		THH00630
	35	$\Delta(T \cdot K) = \Delta(T \cdot K) + SUM * A(T \cdot J)$		THH00640
	40	CONTINUE		ТнН00650
	100	CONTINUE		THH00660
	100	TE(SOSALTAZERO) RETURN		Тнноо670
C				Тинорбар
č		CALCHLATE SOS		Тннооб90
č		CAECOERTE SUS		
				TuH00710
				TuH00720
	110			Tuunn720
	110			
-		SUS=DSQR1(SUS**2+SUM)		1HHUU/40
Ç				THH00750
		RETURN		THH00760
		END		THH00770

SUBROUTINE TRIMAT (A,N,CAR,TEXT,NCHAR,NAMES) C TRIMODIO ¢ TO DISPLAY A VECTOR STORED UPPER TRIANGULAR MATRIX IN A TRIMO020 C TWO-DIMENSIONAL TRIANGULAR FORMAT TRIMO030 С TRIMD040 С A(N+(N+1)/2) VECTOR CONTAINING UPPER TRIANGULAR MATRIX TRIMO050 (DP) С DIMENSION OF MATRIX **{I}** N TRIMOD60 С CAR(N) PARAMETER NAMES (1) TRAMOD70 С AN ARRAY OF FIELDATA CHARACTERS TO BE PRINTED AS TEXT() TRIMOD80 C A TITLE PRECEDING THE MATRIX TRIMOD90 ¢ NCHAR NUMBER OF CHARACTERS, INCLUDING SPACES, THAT TRIMO100 С ARE TO BE PRINTED IN TEXT() TRIMO110 С ABS(NCHAR).LE.126. NCHAR NEGATIVE IS USED **TRIMO120** ¢ TO AVOID SKIPPING TO A NEW PAGE TO START TRIMO130 С PRINTING TRIMO140 ¢ NAMES TRUE TO PRINT PARAMETER NAMES TRIMOISO С TRIMO160 С COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL, OCT.1975) TRIMO170 С TRIM0180 DOUBLE PRECISION A(N) TRIMD190 INTEGER CAR(N), TEXT(1), L(7), LIST(7) TRIM0200 LOGICAL NAMES TRIMO210 INTEGER V(4), VFMT(7) TRIM0220 DATA V/*(2X,*,*A6,1X,*,***,*D17,8)*/, 1 VFMT/*7*,*D17X,6*,*D34X,5*,*O51X,4*,*D68X,3*,*D85X,2*,*102X,1*/ TRIMD230 TRIM0240 ¢ TRIMO250 ¢ M1 M2 ROW LIMITS FOR EACH PRINT SEQUENCE TRIM0260 C N1,M2 COL LIMITS FOR EACH LINE OF PRINT TRIMD270 С L(I) LOC OF EACH COLUMN IN A ROW TRIM0280 C KΤ ROW COUNTER **TRIM0290** ¢ KP PRINT COUNTER TRIMOSOO С TRIMO310 С ۰ ¢ * * INITIALIZE COUNTERS TRIMO320 С TRIM0330 M1 = 1TRIMO340 M2=7 TRIMOSSO N1 **= 1** TRIM0360 KT=0 TRIMO370 KP=0 TRIM0380 IF (.NOT.NAMES) V(2)='15,2X' TRIM0390 C TRIMO400 NC=IABS(NCHAR)/6 TRIMU410 IF (MOD(NCHAR,6),NE+D) NC=NC+1 TR1M0420 IF (NCHAR, GE, D) WRITE (6,200) (TEXT(1), I=1, NC) TRIMD430 IF (NCHAR, LT. 0) WRITE (6,205) (TEXT(1), I=1, NC) TRIM0440 10 IF (M2.GT.N) M2=N TRIM0450 IF (.NOT.NAMES) GO TO 20 TRIM0460 WRITE (6,210) (CAR(I), I=N1, M2) TRIMD470 GO TO 40 TRIM0480 20 M≃N1 TRIM0490 L2=H2=N1+1 TRIMOSOO DO 30 I=1,L2 TRIMO510 TRIM0520 LIST(I) = M M=M+1 30 **TRIM0530** WRITE (6,220) (LIST(1),I=1,L2) TRIMO540

C • • • • • • • • • • • • • • • • • • •		40	CONTINUE		TRIMOREO
D0 190 IC=H1,H2 K=1 IF (IC+LE+(KT+7)) G0 T0 60 JJ=0 TRIM050 TRIM050 D0 50 J=1,IC TRIM0610 TRIM0620 IJ=JJ+J IF (IJ+EQ-7) 60 T0 90 G0 T0 70 C TRIM0660 TRIM0660 TRIM0670 TRIM0670 TRIM0670 TRIM0670 TRIM0670 TRIM07	C		* * * * *		1610000 TRINCEAC
K=1 TRIMOSO IF (IC*LE*(KT*7)) GO TO 4D TRIMOSO D0 5D J=1,IC TRIMOSO D0 5D J=1,IC TRIMOSO JJ=JJ-J TRIMOSO L(K)=JJ TRIMOSO 1=IC-KT*7 TRIMOSO G0 TO 70 TRIMOSO 60 CONTINUE TRIMOSO C TRIMOSO 1=1 TRIMOSO L(K)=L(K)+1 TRIMOSO 70 CONTINUE TRIMOSO 00 L(K)=L(K)+1 TRIMOSO 70 CONTINUE TRIMOSO 90 L(K)=L(K-1)+11 © OBTAIN COL INDEX FOR ROW TRIMO7O 91 L(K)=L(K-1)+11 © OBTAIN COL INDEX FOR ROW TRIMO7O 91 L(K)=L(K-1)+11 © OBTAIN COL INDEX FOR ROW TRIMO7O 92 CONTINUE TRIMO7AO TRIMO7AO 93 CONTINUE TRIMO7AO TRIMO7AO 94 CONTINUE TRIMO7AO TRIMO7AO 95 CONTINUE TRIMO7AO TRIMO7AO 96 CONTINUE TRIMO7AO TRIMO7AO 97 CONTINUE TRIMO7AO TRIMO7AO 98 CONTINUE TRIMO7AO TRIMO8AO			DO 190 IC=M1.M2		TRIM0570
<pre>IF (ICi_E.(KT+7)) GO TO 40 JJ=0 JJ=1 J JI=1 JII=1 JIII JIIII JIIIII JIIIIIIII</pre>			K=1		TRIMOSRO
JJ=0 D0 50 J=1,1C TRIM0400 TRIM0400 L(K)=JJ L(K)=JJ TRIM0400 TRIM040			IF (IC+LE+(KT+7)) GO TO 60		TRIMOSO
D0 50 J=1,1C TRIM0410 50 J=1,1C TRIM0410 50 J=1,1C TRIM0410 L(X)=JJ TRIM0430 I=IC-KT=7 TRIM0430 IF(I)=E4,7) 60 T0 90 TRIM0460 60 T0 70 TRIM0460 60 CONTINUE TRIM0460 70 CONTINUE TRIM0760 70 CONTINUE TRIM0700 70 CONTINUE TRIM0700 70 CONTINUE TRIM0740 80 L(K)=L(K+1)+II © OBTAIN COL INDEX FOR ROW 90 CONTINUE TRIM0740 90 CONTINUE TRIM0740 90 CONTINUE TRIM0740 90 CONTINUE TRIM0740 91 CONTINUE TRIM0740 92 CONTINUE TRIM0740 93 CONTINUE TRIM0740 94 CONTINUE TRIM0740 95 CONTINUE TRIM0740 96 CONTINUE TRIM0740 97 CONTINUE TRIM0740 98 (4) CAR(C), (A(L(1)),I=1,I2) TRIM0740 99 CONTINUE TRIM0740 90 CONTINUE TRIM0800			0=UL		TRIMOADO
50 JJ=J=J TRIM0400 L(K)=JJ TRIM0640 Ii=IC=KT=7 TRIM0640 IF (II=EQ=7) GO TO 90 TRIM0640 GO TO 70 TRIM0660 GO TO 70 TRIM0660 C TRIM0660 II=I TRIM0670 C TRIM0670 70 CONTINUE TRIM0700 70 CONTINUE TRIM0710 70 CONTINUE TRIM0770 70 </td <td></td> <td></td> <td>00 58 J=1,IC</td> <td></td> <td>TRIMOGIO</td>			00 58 J=1,IC		TRIMOGIO
L(K)=JJ II=IC=KT=7 II=IC=KT=7 IF (II=EQ,7) GO TO 90 GO TO 70 GO TO 70 GO TO 70 GO TO 70 CONTINUE C II=I L(K)=L(K)+1 TRIM0680 TRIM0700 DO 80 I=I1,6 K=K+1 II=1*KT=7 GO CONTINUE C CONTINUE C CONTINUE C I2=MIN0(8,(M2+1-KT=7))=11 V(3)=VFMT(11) IF (=N07+NAMES) GO TO 180 WRITE (6,V) CAR(IC),(A(L(I)),I=1,I2) GO TO 190 WRITE (6,V) CAR(IC),(A(L(I)),I=1,I2) IF (M02+0 TRIM0750 TRIM0750 TRIM0750 TRIM0750 TRIM0760 TRIM0750 TRIM0760 TRIM0770 C TRIM0770 C TRIM0770 TRIM0700 T		50	↓ + ↓ ↓		TRIMO620
I1=IC-KT+7 TRIMD400 I1=IC-KT+7 TRIMD400 G0 T0 70 TRIMD650 G0 CONTINUE TRIM0640 C TRIM0640 C TRIM0670 C TRIM0670 C TRIM0670 C TRIM0700 TO CONTINUE TRIM0700 D0 80 I=11,6 TRIM0700 K=K+1 TRIM0710 80 L(K)=L(K-1)+II Ø CONTINUE TRIM0750 90 CONTINUE TRIM0760 Y CONTINUE TRIM0760 C I2=MIN0(8,(M2+1=KT+7))=I1 Ø OBTAIN COL INDEX FOR ROW TRIM0760 Y Y CONTINUE TRIM0760 TRIM0760 C I2=MIN0(8,(M2+1=KT+7))=I1 Ø OBTAIN COL INDEX FOR ROW TRIM0760 Y Y CONTINUE TRIM0760 TRIM0760 G CONTINUE TRIM0760 TRIM0760 Y S0 CONTINUE TRIM0760 TRIM0760 Y TRIM0760 TRIM0760 TRIM0760 Y S0 TO 180 WRITE (6,V) CAR(IC),(A(L(L(K)=JJ		TRIMO630
IP (II+EQ.7) GO TO 90 TRIMO450 GO TO 70 TRIMO640 60 CONTINUE TRIMO640 C TRIMO670 C TRIMO670 0 II=1 TRIMO670 70 CONTINUE TRIMO700 90 CONTINUE TRIM0740 70 CONTINUE TRIM0740 70 CONTINUE TRIM0770 90 CONTINUE TRIM0770 90 CONTINUE TRIM0780 90 CONTINUE TRIM0780 90 CONTINUE TRIM0780 90 CONTINUE TRIM0780 90 CONTINUE TRIM0800 90 CONTINUE TRIM08010 90 FILE (6,V) CAR(IC), (A(L(I)), I=1, I2) TRIM08010 90 FILE (6,V) CAR(IC), (A(L(I)), I=1, I2) TRIM08010 910 FORMAT (140, SX,7(11), (I=1, I=1, NC) TRIM08010			I1=IC=KT+7		TRIMD640
GO 10 70 TRIM0680 60 CONTINUE TRIM0680 11=1 TRIM0680 14(x)=L(x)=L(x)+1 TRIM0700 70 CONTINUE TRIM0700 71 12=MIN0(8,(M2+1=KT*7))=11 WOBTAIN COL INDEX FOR ROW 70 CONTINUE TRIM0700 71 TRIM080 TRIM0700 72 TRIM080 TRIM0700 73 GO TO 190 TRIM0700 74 TRIM0810 TRIM0810 75 GO TO 190 TRIM0810 76 TRIM0810 TRIM0810 77 M2=M2+1 TRIM0810 78 TRIM0810 TRIM0810 79 TRIM0910 TRIM0910 76 <t< td=""><td></td><td></td><td>1F (I1=EQ,7) GO TO 90</td><td></td><td>TRIMO650</td></t<>			1F (I1=EQ,7) GO TO 90		TRIMO650
C TRIM0670 C I1=1 I(x)=L(K)+1 TRIM0680 70 CONTINUE TRIM0700 D0 80 I=11,6 TRIM070 K=K+1 TRIM070 0 L(K)=L(K-1)+II © OBTAIN COL INDEX FOR ROW 80 L(K)=L(K-1)+II © OBTAIN COL INDEX FOR ROW 90 CONTINUE TRIM0700 71 IA=1+KT=7 TRIM0760 90 CONTINUE TRIM0760 70 CONTINUE TRIM0760 71 IA=1+KT=7) TRIM0760 70 CONTINUE TRIM0780 71 CONTINUE TRIM0780 71 TRIM0780 TRIM0780 71 TRIM0780 TRIM0780 71 TRIM0810 TRIM0780 71 TRIM0810 TRIM0800 72 TRIM0810 TRIM0800 730 CONTINUE TRIM0800 741 TRIM0800 TRIM0800 750 CONTINUE TRIM0800 760 TRIM0810 TRIM0800 771 TRIM0810		40	GU TU 70 Continuir		TRIMO660
I1=1 TRIM0680 L(K)=L(K)+1 TRIM0670 70 CONTINUE TRIM0700 00 80 I=11,6 K=K+1 TRIM0700 11=1*KT*7 F0 80 L(K)=L(K-1)*II F0 90 CONTINUE TRIM0730 71 TRIM0740 TRIM0730 70 CONTINUE TRIM0760 70 CONTINUE TRIM0760 70 CONTINUE TRIM0760 70 CONTINUE TRIM0770 71 Textino770 TRIM0760 72 CONTINUE TRIM0760 73 GO TO 180 TRIM0760 74 TRIM0760 TRIM0770 75 TRIM0770 TRIM0770 76 CONTINUE TRIM0800 77 TRIM0800 TRIM0800 78 TRIM080 TRIM0800 79 CONTINUE TRIM0800 79 CONTINUE TRIM0800 79 TRIM080 TRIM0800 79 TRIM0800 TRIM0800 79 CONTINUE TRIM0800 79 TRIM0800 TRIM0800 70 TRIM0800 TRIM0800 70	r	00	CONTINUE		TRIMO670
1181 TRIMO600 70 CONTINUE TRIM0700 70 CONTINUE TRIM0700 70 CONTINUE TRIM0700 70 CONTINUE TRIM0700 70 L(K)=L(K)+1 FOR ROW TRIM0700 70 CONTINUE TRIM0740 TRIM0740 70 CONTINUE TRIM0760 TRIM0760 70 CONTINUE TRIM0760 TRIM0760 70 CONTINUE TRIM0800 TRIM0800 70 CONTINUE TRIM0800 TRIM0800 70 CONTINUE TRIM0800 TRIM0800 70 CONTINUE TRIM0800 TRIM0800 710 CONTINUE TRIM0800 TRIM0800			T 1 - 1		TRIMO680
TO CONTINUE TRIM0760 TO CONTINUE TRIM0710 D0 80 I=I1,6 TRIM0710 K=K+1 TRIM0730 II=I+KT+7 TRIM0730 80 L(K)=L(K-1)+II © OBTAIN COL INDEX FOR ROW TRIM0740 90 CONTINUE TRIM0760 TRIM0760 70 CONTINUE TRIM0760 TRIM0760 71 I2=MIN0(8,(M2+1=KT+7))=11 TRIM0760 TRIM0760 71 TAINOTAL (110) TRIM0760 TRIM0760 71 F(NOT+AMES) GO TO 180 TRIM0801 71 TRIM0710 TRIM0801 TRIM0801 71 TRIM0710 TRIM0801 TRIM0801 71 TRIM0710 TRIM0801 TRIM0801 71 TRIM0710 TRIM0801 TRIM0801 <			1181		TRIM0690
10 CONTINUE TRIM0710 11=1+KT+7 TRIM0720 11=1+KT+7 TRIM0720 90 CONTINUE TRIM0730 90 CONTINUE TRIM0760 70 CONTINUE TRIM0770 12=MIN0(8,(M2+1=KT+7))=11 © OBTAIN COL INDEX FOR ROW TRIM0750 70 CONTINUE TRIM0770 12=MIN0(8,(M2+1=KT+7))=11 TRIM0760 TRIM0770 14 (4,v) CAR(IC),(A(L(I)),I=1,I2) TRIM0800 0 GO TO 190 TRIM08010 TRIM08010 180 WRITE (6,v) IC,(A(L(I)),I=1,I2) TRIM08010 TRIM08010 190 CONTINUE TRIM08010 TRIM08010 190 CONTINUE TRIM08010 TRIM08010 190 CONTINUE TRIM08010 TRIM08010 190 CONTINUE TRIM08010 TRIM08010 191 F(M2+Eq.N) RETURN TRIM08010 TRIM08010 N1=M2+1 TRIM08010 TRIM08010 TRIM08010 M2=M9+7 KT=K+1+1 TRIM0910 TRIM0910 10 GO TO 10 TRIM0910 T		70			TRIM0700
K=K-1 TRIM0720 II=1+KT=7 FRIM0730 B0 L(K)=L(K-1)+II FOR COL INDEX FOR ROW TRIM0740 90 CONTINUE TRIM0760 TRIM0760 C I2=MIN0(8,(M2+1-KT*7))=I1 TRIM0760 TRIM0760 V(3)=VFMT(II) TRIM0760 TRIM0760 if (*NOT*NAMES) GO TO 180 TRIM0790 TRIM0800 WRITE (6,V) CAR(IC),(A(L(I)),I=1,I2) TRIM0810 TRIM0810 180 WRITE (6,V) IC,(A(L(I)),I=1,I2) TRIM0810 TRIM0810 190 CONTINUE TRIM0810 TRIM0810 197 CONTINUE TRIM0810 TRIM0810 190 CONTINUE TRIM0810 TRIM0810 190 CONTINUE TRIM0810 TRIM0810 N1=M2+1 M2=M2+7 TRIM0810 TRIM0810 KT=KT+1 TRIM0810 TRIM0800 TRIM0800 KT=KC+1.3) GO TO 10 TRIM0810 TRIM0810 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0800 TRIM0910 GO TO 10 TRIM0920 TRIM0920 TRIM0920 200 FORMAT (1H1,2X,21A6) TITLE TRIM092					TRIMD710
II=I+KT*7 IRIM0740 80 L(K)=L(K-1)+II © OBTAIN COL INDEX FOR ROW TRIM0740 90 CONTINUE TRIM0740 TRIM0750 90 CONTINUE TRIM0760 TRIM0760 60 I2=MIN0(8,(M2+1-KT*7))=11 TRIM0760 TRIM0770 1 YEMM0760 TRIM0780 TRIM0770 1 IF (*NOT*NAMES) GO TO 180 TRIM0790 TRIM0790 1 F(*NOT*NAMES) GO TO 180 TRIM0800 TRIM0800 0 GO TO 190 IF (*NOT*NAMES) GO TO 180 TRIM0800 180 WRITE (6,v) IC.(A(L(I)),I=1,I2) TRIM0800 TRIM0800 180 WRITE (6,v) RETURN TRIM0800 TRIM0800 N1=M2+1 TRIM080 TRIM0800 TRIM0800 M2=M2+7 TRIM0800 TRIM0800 TRIM0800 KT=K+1 TRIM0800 TRIM0800 TRIM0800 KT=K+1 TRIM0800 TRIM0800 TRIM0800 KT=K+1 TRIM0800 TRIM0910 TRIM0910 K0 TO 10 TRIM0920 TRIM0920 WRITE (6,2000) (TEXT(I),I=I,NC) © TITLE			K=K+1		TRIMO720
B0 L(K)=L(K-1)+II © OBTAIN COL INDEX FOR ROW TRIM0750 90 CONTINUE TRIM0760 TRIM0760 0 I2=MIND(8,(M2+1=KT*7))=I1 TRIM0760 TRIM0760 V(3)=VFMT(I1) TRIM0760 TRIM0760 0 GO TO.180 TRIM0760 0 GO TO.180 TRIM0760 0 GO TO.190 TRIM0800 180 WRITE (6,V) CAR(IC),(A(L(I)),I=1,I2) TRIM0800 180 WRITE (6,V) IC,(A(L(I)),I=1,I2) TRIM0800 190 CONTINUE TRIM0800 190 CONTINUE TRIM0800 190 CONTINUE TRIM0800 191 F(M2+EQ*N) RETURN TRIM0800 N1=M2+1 TRIM0800 TRIM0800 M2=M2+7 TRIM0800 TRIM0800 KT=KT+1 TRIM0800 TRIM0800 KP=KP+1 TRIM0800 TRIM0900 GO TO 10 TRIM0900 TRIM0900 200 FORMAT (1H1,2X,21A6) TRIM0900 210 FORMAT (1H0,2X,21A6) TRIM0900 220 FORMAT (1H0,3X,7(11X,A6)) P HORIZONTAL N			1 == 1 == 2		
90 CONTINUE TRIM0750 C TRIM0760 TRIM0760 12=MIN0(8,(M2+1=KT*7))=11 TRIM0760 V(3)=VFMT(11) TRIM0760 IF (*NOT*NAMES) GO TO 180 TRIM0760 WRITE (6,V) CAR(IC),(A(L(I)),I=1,I2) TRIM0800 GO TO 190 TRIM0800 IB0 WRITE (6,V) IC,(A(L(I)),I=1,I2) TRIM0800 IB0 WRITE (6,V) IC,(A(L(I)),I=1,I2) TRIM0800 IB0 WRITE (6,V) RETURN TRIM0800 N1=M2+1 TRIM0800 TRIM0800 M2=M2+7 TRIM0800 TRIM0800 KT=KT+1 TRIM0800 TRIM0800 KP=KP+1 TRIM0800 TRIM0800 IF (KP+LT.3) GO TO 10 TRIM0900 TRIM0900 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 TRIM0900 GO TO 10 TRIM0900 TRIM0900 TRIM0900 C0 TO TRIM0900 TRIM0900 200 FORMAT (1H1,2X,21A6) W TITLE TRIM0940 210 FORMAT (1H0,5X,7(11X,A6)) W HORIZONTAL NAMES TRIM0970 220 FORMAT (1H0,3X,7(11X,16)) TRIM0970 TRIM0970		80	£(K)=L(K=1)+II	B OBTAIN COL INDEX FOR R	0W TRIMO740
C IZ=MIN0(8,(M2+1-KT*7))=11 IZ=MIN0(8,(M2+1-KT*7))=11 V(3)=VFMT(I1) IF (*NOT*NAMES) GO TO 180 WRITE (6,V) CAR(IC),(A(L(1)),I=1,I2) GO TO 190 IBO WRITE (6,V) IC,(A(L(I)),I=1,I2) IBO WRITE (6,V) IC,(A(L(I)),I=1,I2) IF (M2=EQ.N) RETURN N1=M2+1 M2=M2+7 KT=KT+1 KTH0840 IF (KP=LT*3) GO TO 10 WRITE (6,200) (TEXT(I),I=1,NC) GO TO 10 C C C C C C C C C C C C C		90	CONTINUE	a caller cof inefr i fr v	U" INTHOYOU TRIMATIO
I2=MIN0(8,(M2+1=KT*7))=11 TRIM0780 V(3)=VFMT(I1) TRIM0790 IF (*NOT*NAMES) GO TO 180 TRIM0800 WRITE (6,V) CAR(IC),(A(L(I)),I=1,I2) TRIM0810 GO TO 190 TRIM0820 IB0 WRITE (6,V) IC,(A(L(I)),I=1,I2) TRIM0800 IF (M2*EQ.N) RETURN TRIM0800 N1=M2+1 TRIM0800 M2=M2+7 TRIM0800 KT*KT+1 TRIM0800 KP=KP+1 TRIM0800 IF (KP*LT*3) GO TO 10 TRIM0800 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 GO TO 10 TRIM0900 C TRIM0910 200 FORMAT (1H1,2X,21A6) TITLE TRIM0910 210 FORMAT (1H0,2X,21A6) TITLE TRIM0920 210 FORMAT (1H0,3X,7(11X,A6)) HORIZONTAL NAMES TRIM0950 220 FORMAT (1H0,3X,7(11X,A6)) HORIZONTAL NAMES TRIM0970 C TRIM0970 TRIM0970 <td>С</td> <td></td> <td></td> <td></td> <td>TRIM0700</td>	С				TRIM0700
V(3)=VFMT(I1) TRIMO790 IF (*NOT.NAMES) GO TO 180 TRIM0800 WRITE (6,V) CAR(IC),(A(L(I)),I=1,I2) TRIM0810 GO TO 190 TRIM0810 180 WRITE (6,V) IC,(A(L(I)),I=1,I2) TRIM0810 190 CONTINUE TRIM0810 190 CONTINUE TRIM0840 N1=M2+1 TRIM0840 TRIM0850 M2=M2+7 TRIM0840 TRIM0860 KT=KT+1 TRIM0860 TRIM0870 KT=KT+1 TRIM0890 TRIM0890 IF (KP.LT.3) GO TO 10 TRIM0800 TRIM0900 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 TRIM0900 GO TO 10 TRIM0900 TRIM0900 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 TRIM0900 GO TO 10 TRIM0900 TRIM0900 200 FORMAT (1H1,2X,21A6) © TITLE TRIM0900 210 FORMAT (1H0,2X,21A6) © TITLE TRIM0950 210 FORMAT (1H0,3X,7(11X,A6)) © HORIZONTAL NAMES TRIM0960 220 FORMAT (1H0,3X,7(11X,16)) TRIM0980 TRIM0980 C TRIM0980 TRIM0980 <td></td> <td></td> <td>I2=MINC(8,(M2+1-KT*7))=I1</td> <td></td> <td>TRIM0780</td>			I2=MINC(8,(M2+1-KT*7))=I1		TRIM0780
IF (*NOT*NAMES) GO TO 180 TRIMOBOD WRITE (6,V) CAR(IC),(A(L(I)),I=1,I2) TRIMOBID GO TO 190 TRIMOBID IB0 WRITE (6,V) IC,(A(L(I)),I=1,I2) TRIMOBID 190 CONTINUE TRIMOBID 190 CONTINUE TRIMOBID 190 CONTINUE TRIMOBID 191 F(M2.EQ.N) RETURN TRIMOBID N1=M2+1 TRIMOBID TRIMOBID M2=M2+7 TRIMOBID TRIMOBID KT=KT+1 TRIMOBID TRIMOBID KP=KP+1 TRIMOBID TRIMOBID IF (KP+LT+3) GO TO 10 TRIMOBID TRIMOBID WRITE (6,200) (TEXT(I),I=1,NC) TRIMOBID TRIMOPOD GO TO 10 TRIMOBID TRIMOPOD TRIMOPOD VRITE (6,200) (TEXT(I),I=1,NC) TRIMOPOD TRIMOPOD GO TO 10 TRIMOPOD TRIMOPOD TRIMOPOD C 200 FORMAT (1H1,2X,21A6) TITLE TRIMOPOD 210 FORMAT (1H0,2X,21A6) TITLE TRIMOPOD TRIMOPOD 220 FORMAT (1H0,3X,7(11X,A6)) THORIZONTAL NAMES TRIM0960 220 FORMAT (1H0,			V(3) = VFMT(11)		TRIMO790
WRITE (6,V) CAR(IC), (A(L(I)), I=1, I2) TRIMOBID GO TO 190 TRIMOBID 180 WRITE (6,V) IC, (A(L(I)), I=1, I2) TRIMOBID 190 CONTINUE TRIMOBID 190 CONTINUE TRIMOBID IF (M2.EQ.N) RETURN TRIMOBOD TRIMOBOD N1=M2+1 TRIMOBOD TRIMOBOD M2=M2+7 TRIMOBOD TRIMOBOD KT=KT+1 TRIMOBOD TRIMOBOD KP=KP+1 TRIMOBOD TRIMOBOD IF (KP.LT.3) GO TO 10 TRIMOPOD TRIMOPOD WRITE (6,200) (TEXT(I), I=1,NC) TRIMO90D TRIMO90D GO TO 10 TRIMO90D TRIMO90D C 200 FORMAT (1H1,2X,21A6) O TITLE TRIM090D 205 FORMAT (1H0,2X,21A6) O TITLE TRIM090D 205 FORMAT (1H0,5X,7(11X,A6)) O HORIZONTAL NAMES TRIM095D 210 FORMAT (1H0,3X,7(11X,A6)) O HORIZONTAL NAMES TRIM096D 220 FORMAT (1H0,3X,7(11X,I6)) TRIM097D TRIM096D C END TRIM097D TRIM097D			IF (+NOT+NAMES) GO TO 180		TRIMOSOO
GO TO 190 TRIM0820 180 WRITE (6,V) IC,(A(L(I)),I=1,I2) TRIM0830 190 CONTINUE TRIM0840 190 IF (M2.EQ.N) RETURN TRIM0840 N1=M2+1 TRIM0850 TRIM0850 M2=M2+7 TRIM08070 TRIM08070 KT=KT+1 TRIM0800 TRIM0800 KP=KP+1 TRIM0800 TRIM0800 IF (KP.LT.3) GO TO 10 TRIM0800 TRIM0800 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 TRIM0900 GO TO 10 TRIM0900 TRIM0900 C 200 FORMAT (1H1,2X,21A6) © TITLE TRIM0900 200 FORMAT (1H0,2X,21A6) © TITLE TRIM0900 210 FORMAT (1H0,2X,21A6) © TITLE TRIM0900 210 FORMAT (1H0,3X,7(11X,A6)) © HORIZONTAL NAMES TRIM0960 220 FORMAT (1H0,3X,7(11X,I6)) © HORIZONTAL NAMES TRIM0960 C END TRIM0980 TRIM0980			WRITE (6,V) CAR(IC),(A(L(I)),	I=1:12)	TRIMOSIO
180 WRITE (6,V) IC,(A(L(I)),I=1,I2) TRIMO830 190 CONTINUE TRIM0840 1F (M2.EQ,N) RETURN TRIM0850 N1=M2+1 TRIM0850 M2=M2+7 TRIM0870 KT=KT+1 TRIM0870 KP=KP+1 TRIM0890 IF (KP.LT.3) GO TO 10 TRIM0900 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 GO TO 10 TRIM0900 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 GO TO 10 TRIM0900 Z00 FORMAT (1H1,2X,21A6) © TITLE 205 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,3X,7(11X,A6)) © HORIZONTAL NAMES 210 FORMAT (1H0,3X,7(11X,A6)) © HORIZONTAL NAMES 220 FORMAT (1H0,3X,7(11X,16)) TRIM0900 C END TRIM0900			GO TO 190		TRIMO820
190 CONTINUE TRIMD840 IF (M2+EQ.N) RETURN TRIMD850 N1=M2+1 TRIM080 M2=M2+7 TRIM0870 KT=KT+1 TRIM080 KP=KP+1 TRIM0800 IF (KP+LT+3) GO TO 10 TRIM0900 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 GO TO 10 TRIM0900 Z00 FORMAT (1H1,2X,21A6) © TITLE 205 FORMAT (1H1,2X,21A6) © TITLE 205 FORMAT (1H0,3X,7(11X,A6)) © HORIZONTAL NAMES 210 FORMAT (1H0,3X,7(11X,A6)) © HORIZONTAL NAMES 220 FORMAT (1H0,3X,7(11X,I6)) TRIM0900 C TRIM0900 END TRIM0900		180	WRITE (6,V) IC, $(A(L(I)), I=1, I)$	2)	TRIMD830
IF (M2+EQ.N) RETURN TRIM0850 N1=M2+1 TRIM0860 M2=M2+7 TRIM0870 KT=KT+1 TRIM0870 KP=KP+1 TRIM0890 IF (KP+LT-3) GO TO 10 TRIM0900 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 GO TO 10 TRIM0900 C TRIM0900 200 FORMAT (1H1,2X,21A6) © TITLE 205 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,3X,7(11X,A6)) © HORIZONTAL NAMES 220 FORMAT (1H0,3X,7(11X,I6)) TRIM0960 C TRIM0970 END TRIM0980		190	CONTINUE		TRIM0840
NI=M2+1 TRIM0860 M2=M2+7 TRIM0870 KT=KT+1 TRIM0800 KP=KP+1 TRIM0800 IF (KP+LT+3) GO TO 10 TRIM0900 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 GO TO 10 TRIM0900 C TRIM0900 200 FORMAT (1H1,2X,21A6) © TITLE 205 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,5X,7(11X,A6)) © HORIZONTAL NAMES 220 FORMAT (1H0,3X,7(11X,I6)) C C TRIM0900 END TRIM0900			IF (M2+EQ+N) RETURN		TRIMO850
TRIM0870 KT=KT+1 TRIM0800 KP=KP+1 TRIM0800 IF (KP+LT-3) GO TO 10 TRIM0900 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 GO TO 10 TRIM0900 C TRIM0900 200 FORMAT (1H1,2X,21A6) © TITLE 205 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,5X,7(11X,A6)) © HORIZONTAL NAMES 220 FORMAT (1H0,3X,7(11X,I6)) C C TRIM0980 END TRIM0980					TRIMO860
KIEKT#I TRIMD880 KP=KP+1 TRIMD800 IF (KP+LT.3) GO TO 10 TRIMD900 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 GO TO 10 TRIM0910 C TRIM0930 200 FORMAT (1H1,2X,21A6) © TITLE 205 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,5X,7(11X,A6)) © HORIZONTAL NAMES 220 FORMAT (1H0,3X,7(11X,I6)) © HORIZONTAL NAMES C TRIM0960 TRIM0970 TRIM0980 TRIM0980 TRIM0980			112402+/ *****		TRIMO870
IF (KP+LT,3) G0 T0 10 TRIM0890 WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 G0 T0 10 TRIM0920 C TRIM0930 200 FORMAT (1H1,2X,21A6) © TITLE 205 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,3X,7(11X,A6)) © HORIZONTAL NAMES 220 FORMAT (1H0,3X,7(11X,I6)) C C TRIM0960 TRIM0980 TRIM0980 TRIM0980 TRIM0980			NI≕N] # 4 KP=Kp + 1		TRIMOBSO
WRITE (6,200) (TEXT(I),I=1,NC) TRIM0900 GO TO 10 TRIM0910 C TRIM090 200 FORMAT (1H1,2X,21A6) © TITLE TRIM0900 205 FORMAT (1H0,2X,21A6) © TITLE TRIM0900 210 FORMAT (1H0,2X,21A6) © TITLE TRIM0900 210 FORMAT (1H0,3X,7(11X,A6)) © HORIZONTAL NAMES TRIM0950 220 FORMAT (1H0,3X,7(11X,I6)) C TRIM0900 C TRIM0900 TRIM0900 TRIM0900 TRIM0900 TRIM0900 TRIM0900 TRIM0900 TRIM0900 TRIM0900 TRIM0900 TRIM0900			$\frac{1}{1} \frac{1}{1} \frac{1}$		TRIMO890
GO TO 10 TRIMU910 C TRIM0920 200 FORMAT (1H1,2X,21A6) © TITLE 205 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,5X,7(11X,A6)) © TITLE 220 FORMAT (1H0,3X,7(11X,A6)) © HORIZONTAL NAMES C TRIM0960 TRIM0970 TRIM0980 TRIM0980 TRIM0980			WRITE (A.200) (TEX+(1).1#1.NC)		TRIM0900
C TRIM0920 200 FORMAT (1H1,2X,21A6) © TITLE 205 FORMAT (1H0,2X,21A6) © TITLE 210 FORMAT (1H0,5X,7(11X,A6)) © HORIZONTAL NAMES 220 FORMAT (1H0,3X,7(11X,I6)) © RITLE C TRIM0980 END TRIM0980			GO TO 10		
200 FORMAT (1H1,2X,21A6) © TITLE TRIM0930 205 FORMAT (1H0,2X,21A6) © TITLE TRIM0940 210 FORMAT (1H0,2X,21A6) © TITLE TRIM0950 210 FORMAT (1H0,5X,7(11X,A6)) © HORIZONTAL NAMES TRIM0960 220 FORMAT (1H0,3X,7(11X,I6)) © RIM0970 TRIM0980 C TRIM0980 TRIM0980	с				
205 FORMAT (1H0,2X,21A6) 0 TITLE TRIM0950 210 FORMAT (1H0,5X,7(11X,A6)) 0 HORIZONTAL NAMES TRIM0960 220 FORMAT (1H0,3X,7(11X,I6)) 0 HORIZONTAL NAMES TRIM0970 C TRIM0980 TRIM0980 END TRIM0970 TRIM0980		200	FORMAT (1H1.2X.21A6)		
210 FORMAT (1H0,5X,7(11X,A6)) @ HORIZONTAL NAMES TRIM0950 220 FORMAT (1H0,3X,7(11X,16)) TRIM0970 C END TRIM0980		205	FORMAT (1H0,2X,21A4)	Q TITLE	15400770 TRIMOSES
220 FORMAT (1H0,3X,7(11X,16)) C END TRIM0920		210	FORMAT (1H0,5X,7(11X,A6))	P HORIZONTAL NAMES	TR!Mno40
C TRIM0980 END TRIM0980		220	FORMAT (1H0,3X,7(11X,16))	A MANINE VIGA	TRIM0970
END	C		· · · ·		TRIMOSAN
			END		TRIM0990

Ç. TTHH0010 THIS SUBROUTINE COMBINES TWO SINGLE SUBSCRIPTED SRIF ARRAYS TTHH0020 , Ĉ. 8000000000 TTHH0030 USING HOUSEHOLDER ORTHOGONAL TRANSFORMATIONS TTHH0040 R(N*(N+1)/2)VECTOR STORED SRIF ARRAY. TTHH0050 RA(N*(N+1)/2) THE SECOND VECTOR STORED SRIF ARRAY TTHH0060 DIMENSION OF THE ESTIMATED PARAMETER VECTOR. TTHH0070 Ν . TTHH0080 A NEGATIVE VALUE FOR N IS USED TO NOTE THAT TTHH0090 R AND RA HAVE RT. HAND SIDES INCLUDED AND TTHH0100 HAVE DIM=ARSN*(ABSN+3)/2. **TTHH0110** ON EXIT RA IS CHANGED AND R CONTAINS THE RESULTING SRIF ARRAY TTHH0120 TTHH0130 С TTHH0140 COGNIZANT PERSONS G.J.BIERMAN/M.W.NEAD (JPL, JAN. 1976) С TTHH0150 С TTHH0160 IMPLICIT DOUBLE PRECISION(A-H, 0-Z) **TTHH0170** DIMENSION RA(1), R(1) DOUBLE PRECISION SUM & FOR USE IN SINGLE PRECISION VERSION **TTHH0180** С TTHH0190 С TTHH0200 ZERO=0. TTHH0210 ONE=1. TTHH0220 NP1=N TTHH0230 IF (N.GT.0) GO TO 10 TTHH0240 N--N TTHH0250 NP1=N+1 TTHH0260 ⋒ IJ(START) 10 1JS=1 TTHH0270 KK=0 TTHH0280 DO 100 J=1.N ■ J-TH STEP OF HOUSEHOLDER REDUCTION TTHH0290 KK=KK+J TTH40300 SUM=R(KK)**2 TTHH0310 DO 20 I=IJS,KK TTHH0320 SUM=SUM+RA(I)**2 20 TTHH0330 IF (SUM.LE.ZERO) GO TO 100 TTHH0340 SUM=SQRT(SUM) TTHH0350 IF (R(KK).GT.ZERO) SUM=-SUM TTHH0360 DELTA=R(KK)-SUM TTHH0370 R(KK)=SUM TTHH0380 BETA=ONE/(SUM*DELTA) TTHH0390 JJ=KK TTHH0400 ∟≕ປ TTHH0410 JP1=J+1 TTHH0420 IKS=KK+1 TTHH0430 * * * J-TH HOUSFHOLDER TRANS. DEFINED С 40 LOOP APPLIES TRANSFORM. TO COLS. J+1 TO NP1 TTHH0440 С DO 40 K=JP1.NP1 TTHH0450 TTHH0460 JJ=JJ+L TTHH0470 L=L+1TTHH0480 IK=IKS TTHH0490 SUM=DELTA*R(JJ) TTHH0500 DO 30 I=1JS+KK SUM=SUM+RA(IK)*RA(I) TTH:H0510 TTHH0520 30 IK=IK+1 TTHH0530 IF (SUM.EQ.ZERO) GO TO 40 TTHH0540 SUM=SUM*BETA

77-26

SUBROUTINE TTHH(R, RA, N)

	TTUINEEA
R(JJ)=R(JJ)+SUM*DELIA	I I HHUSSU
IK=IKS	Ттнно560
DO 35 I=IJS+KK	TTHH0570
RA(IK) = RA(IK) + SUM + RA(I)	TTHH0580
IK=IK+1	TTHH0590
IKS=IKS+K	ТТННОБОО
IJS=KK+1	TTHH0610
	TTHH0620
RETURN	TTHH0630
END	TTHH0640
	R(JJ)=R(JJ)+SUM*DELTA IK=IKS D0 35 I=IJS+KK RA(IK)=RA(IK)+SUM*RA(I) IK=IK+1 IKS=IKS+K IJS=KK+1 RETURN END

77–26

SUBROUTINE TZERO (R,N, IS, IF)

С

С

10	TO ZERO OUT F STORED UPPER R(N*(N+1)/2) N IS IR COGNIZANT PEF IMPLICIT DOUBLE DIMENSION R(1) ZERO=0.0 IJS=IS*(IS-1)/ DO 10 I=IS.IF IJS=IJS+I IJ=IJS DO 10 J=I.N R(IJ)=ZERO IJ=IJ+J CONTINUE	ROWS IS (ISTART) TO IF (IFINAL) OF A VECTOR TRIANGULAR MATRIX INPUT VECTOR STORED UPPER TRIANGULAR MATRIX DIMENSION OF R FIRST ROW OF R THAT IS TO RE SET TO ZERO LAST ROW OF R THAT IS TO BF SET TO ZERO RSONS: G.J.BIERMAN/C.F.PETERS (JPL, NOV. 1975) PRECISION (A-H,O-Z)	TZER0010 TZER0020 TZER0030 TZER0040 TZER0050 TZER0050 TZER0070 TZER0070 TZER0070 TZER0100 TZER0100 TZER0100 TZER0120 TZER0120 TZER0130 TZER0140 TZER0140 TZER0150 TZER0160 TZER0190 TZER0190 TZER0200 TZER0210 TZER0220 TZER0230
10	IJ=IJ+J CONTINUE RETURN END		TZER0220 TZER0230 TZER0240 TZER0250 TZER0260

SUBROUTINE UDMES (U, N, R, A, G, ALPHA) UDMES010 UDMES020 COMPUTES ESTIMATE AND U-D MEASUREMENT UPDATED UDMES030 UDMES040 COVARIANCE, P=UDU**T UDMES050 *** INPUTS *** UDMES060 UDMES070 UPPER TRIANGULAR MATRIX, WITH D ELEMENTS STORED AS THE UDMES080 U DIAGONAL. U IS VECTOR STORED AND CORRESPONDS TO THE UDMES090 A PRIORI COVARIANCE. IF STATE ESTIMATES ARE COMPUTED. UDMES100 THE LAST COLUMN OF U CONTAINS X. UDMES110 DIMENSION OF THE STATE ESTIMATE. UDMES120 Ν MEASUREMENT VARIANCE UDMES130 R VECTOR OF MEASUREMENT COEFFICIENTS, IF DATA THEN A(N+1)=ZUDMES140 Δ UDMES150 IF ALPHA LESS THAN ZERO NO ESTIMATES ARE COMPUTED ALPHA (AND X AND Z NEED NOT BE INCLUDED) UDMES160 UDMES170 *** OUTPUTS *** UDMES180 UDMES190 UPDATED, VECTOR STORED FACTORS AND ESTIMATE AND U UDMES200 U((N+1)(N+2)/2) CONTAINS (Z-A**T*X) UDMES210 UDMES220 INNOVATIONS VARIANCE OF THE MEASUREMENT RESIDUAL ALPHA UDMES230 VECTOR OF UNWEIGHTED KALMAN GAINS, K=G/ALPHA UDMES240 G CONTAINS U**TA AND (Z-A**T*X)/ALPHA UDMES250 А UDMES260 COGNIZANT PERSONS: G.J. BIERMAN/M.W. NEAD (JPL, SEPT.1976) UDMES270 UDMES280 IMPLICIT DOUBLE PRECISION (A-H, 0-Z) UDMES290 DIMENSION U(1), A(1), G(1)UDMES300 DOUBLE PRECISION SUM UDMES310 UDMES320 LOGICAL IEST UDMES330 ZERO=0.0 UDMES340 IEST=.FALSE. UDMES350 UDMES360 ONE=1. UDMES370 NP1=N+1 NTOT=N*NP1/2 UDMES380 IF (ALPHA.LT.ZERO) GO TO 3 UDMES390 SUM=A(NP1) UDMES400 DO 1 J=1,N UDMES410 SUM=SUM-A(J)*U(NTOT+J) UDMES420 1 @ Z=Z=A**T*X U(NTOT+NP1)=SUM UDMES430 UDMES440 ILST=.TRUE. UDMES450 UDMES460 3 KJ=NTOT UDMES470 DO 10 J=N+2+-1 UDMES480 SUM=A(J) UDMES490 JM1=J-1 D0 5 K=JM1,1,-1 UDMES500 UDMES510 KJ=KJ-1 SUM=SUM+U(KJ)*A(K)UDMES520 5 UDMES530 A(J) = SUMKJ=KJ-1 UDMES540 UDMES550 10 G(J)=SUM*U(KJ+J)

86

77-26

С

С

			1-	20	
с с		G(1)=U(1)*A(1) A=U**T*A AND G=D*(U**T*A)			UDMES560 UDMES570 UDMES580
с с		SUM=R+G(1)*A(1) GAMMA=0 IF (G(1).EQ.ZERO) GO TO 11	ព	SUM(1) P FOR R=0 CASE P FOR R=0 CASE	UDMES590 UDMES600 UDMES610
с		U(1)=U(1)*R*GAMMA	ß	D(1)	UDMES630 UDMES640
	11	KJ=2 DO 20 J=2+N BETA=SIM	ត	BETA=SUM(J=1)	UDMES650 UDMES660 UDMES670
		SUM=SUM+G(J) *A(J) P=-A(J) *GAMMA	ត ត	$SU_{M}(J)$ P=-F(J)*(1/SUM(J-1)) EQN(21)	UDMES680 UDMES690
		JM1=J-1 DO 15 K=1,JM1			UDMES700 UDMES710
		U(KJ)=S+P*G(K) G(K)=G(K)+G(J)*S	ର ର	EQN(22) EQN(23)	UDMES720 UDMES730 UDMES740
с	15	KJ=KJ+1 IF (G(J).EQ.ZERO) GO TO 20	~	D FOR R=0 CASE	UDMES750 UDMES760
	20	GAMMA=ONE/SUM U(KJ)=U(KJ)*BETA*GAMMA KJ=KJ+1	ត្រ	$\begin{array}{l} GAMMA=1/SOM(J)\\ D(J) \qquad EQN(19) \end{array}$	UDMES770 UDMES780 UDMES790
с		ALPHA=SUM			UDMES800 UDMES810
C C		EQN. NOS. REFER TO BIERMAN'S	19	75 CDC PAPER; PP. 337-346.	UDMES820 UDMES830
		A(NP1)=U(NTOT+NP1)*GAMMA DO 30 J=1/N			UDMES850 UDMES860
С	30	U(NTOT+J)=U(NTOT+J)+G(J)*A(NP1)			UDMES870 UDMES880
		RETURN END			UDMES890 UDMES900

SUBROUTINE UD2COV (UIN, POUT, N) UD2C0010 UD2C0020 TO OBTAIN A COVARIANCE FROM ITS U-D FACTORIZATION. BOTH MATRICES UD2C0030 ARE VECTOR STORED AND THE OUTPUT COVARIANCE CAN OVERWRITE THE Un2C0040 INPUT U-D ARRAY. UIN=U-D IS RELATED TO POUT VIA POUT=UDU(**T) Un2C0050 UD2C0060 UIN(N*(N+1)/2)INPUT U-D FACTORS, VECTOR STORED WITH THE D UD2C0070 ENTRIES STORED ON THE DIAGONAL OF UIN Un2C0080 POUT(N*(N+1)/2) OUTPUT COVARJANCE, VECTOR STORED. Un2C0090 (POUT=UIN IS PFRMITTED) Up2C0100 DIMENSION OF THE MATRICES INVOLVED N UD2C0110 UD2C0120 COGNIZANT PERSONS: G.J.BIERMAN/M.W.NEAD (JPL: FEB. 1977) Un2C0130 Un2C0140 IMPLICIT DOUBLE PRECISION (A-H, 0-Z) UD2C0150 UD2C0160 UIN(1)+ POUT(1)DIMENSION UD2C0170 UD2C0180 UD2C0190 POUT(1)=UIN(1) JJ=1 UD2C0200 D0 20 J=2+N UD2C0210 JJL≃JJ UD2C0220 ს+სს=სს UD2C0230 POUT(JJ)=UIN(JJ) UD2C0240 S=POUT(JJ) UD2C0250 II=0 U02C0260 JM1=J-1 Un2C0270 DO 20 I=1,JM1 Un2c0280 Un2C0290 II=II+I ALPHA=S*UIN(JJL+I) □ JJL+I=(I,J) UD2C0300 IK=II UD2C0310 DO 10 K=I,JM1 UD2C0320 POUT(IK)=POUT(IK)+ALPHA*UIN(JJL+K) \square JJL+K=(K)J) Un2C0330 IK=IK+K Un2c0340 10 POUT(JJL+I)=ALPHA 20 UD2C0350 Un2C0360 RETURN Un2C0370 END UD2C0380

7**7**–26

С

С

С

C C

0000000

С

С

С

С

С

88

С									
C C		SUBROUTINE UD2SIG(U,N,SIG,TEXT,NCT)							
		COMPUTE STANDARD DEVIATIONS (SIGMAS) FROM U-D COVARIANCE FACTORS							
С				UD2SI040					
С		U(N*(N+1)/2)	INPUT VECTOR STORED ARRAY CONTAINING THE U-D	UD2SI050					
С			FACTORS, THE D (DIAGONAL) ELEMENTS ARE STORED	Up251060					
С			ON THE DIAGONAL	Un2SI070					
С		SIG(N)	VECTOR OF OUTPUT STANDARD DEVIATIONS	UD251080					
С		TEXT()	ARRAY OF FIELDATA CHARACTERS TO BE PRINTED	UD2S1090					
Ć			PRECEDING THE VECTOR OF SIGMAS	UD2SI100					
Ċ		NCT .	NUMBER OF CHARACTERS IN TEXT, 0.LE.NCT.LE.126	UD2SI110					
С			IF NCT=0, NO STGMAS ARE PRINTED	UD251120					
Ċ				UD2SI130					
С		COGNIZANT PER	SONS: G.J.BIERMAN/M.W.NEAD (JPL, FEB, 1977)	UD2SI140					
С				Up2SI150					
		IMPLICIT DOUR	LE PRECISION (A-H.O-Z)	UD2SI160					
		INTEGER TEXT(1)	UD251170					
		DIMENSION U(1), SIG(1)	UD2SI180					
С				UD25I190					
		JJ=1		UD2SI200					
		SIG(1)=U(1)		UD251210					
		D0 10 J=2/N		UD2SI220					
		JUL=JU	<pre></pre>	UD251230					
		L+PP=PP		Un251240					
		S=U(JJ)		Un2SI250					
		SIG(J)=S		UD251260					
		JM1=J-1							
		DO 10 I=1, JM1							
	10	SIG(I)=SIG(I)+S*U(JJL+I)**2							
Ç									
С		WE NOW HAVE VARIANCES							
С				Un2SI320					
		D0 20 J=1+N		Un2SI330					
	20	SIG(J)=SQRT	'(SIG(J))	Un2SI340					
		IF (NCT.EQ.0) GO TO 30							
		NC=NCT/6		Un2SI360					
		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		Un2SI400					
¢				UD2SI410					
	40	FORMAT (1H0)2	X+21A6)	Un2SI420					
	50	FORMAT (1H0)(6018.10))	Un2SI430					
		END		UD2SI440					

89

С С С С С С С C С С С С ¢ Č С С

C C C

¢ С С

4

UTINV100 INPUT VECTOR STORED UPPER TRIANGULAR MATRIX RIN(N*(N+1)/2)MATRIX DIMENSION **UTINV110** N OUTPUT VECTOR STORED UPPFR TRIANGULAR MATRIX ROUT(N*(N+1)/2)UTINV120 INVERSE UTINV130 UTINV140 COGNIZANT PERSONS: G.J.BIERMAN/J.ELLIS (JPL, SEPT. 1976) **UTINV150 UTINV160** DOUBLE PRECISION RIN(1), ROUT(1), WORK, ONF, ZERO , DIN UTINV170 DATA ONE/1.0D0/.ZER0/ 0.0D0/ UTINV180 IPV = N*(N+1)/2 **UTINV190** IN = IPV UTINV200 I=1+N D0 6 UTINV210 IF (RIN(IPV).NE.ZERO) GO TO 1 UTINV220 WRITE (6,10) I UTINV230 RETURN UTINV240 ONE/ RIN(IPV) **UTINV250** 1 DIN ROUT(IPV) = DIN UTINV260 UTINV270 MIN =N KEND = I-1**UTINV280** LANF = N = KENDUTINV290 IF (I.EQ.1) GO TO 5 2 J= IN UTINV300 UTINV310 UTINV320 INITIALIZE ROW LOOP UTINV330 UTINV340 D0 4 K=1+KEND UTINV350 WORK =ZERO UTINV360 MIN= MIN -**UTINV370** 1 LIN= IPV UTINV380 LOT= J UTINV390 UTINV400 START INNER LOOP **UTINV410 UTINV420** D0 3 L=LANF, MIN UTINV430 LIN= LIN+L UTINV440 LOT= LOT+1 UTINV450 3 WORK = WORK + RIN(LIN)* ROUT(LOT) UTINV460 ROUT(J) =-WORK* DIN UTINV470 J= J- MIN UTINV480 5 IPV = IPV -MIN UTINV490 6 IN= IN -1 UTINV500 UTINV510 RETURN 10 FORMAT (1H0,10X, UTINV DIAGONAL', I4, 'IS ZERO') UTINV520 UTINV530 END

77-26

TO INVERT AN UPPER TRIANGULAR VECTOR STORED MATRIX AND STORE

THE RESULT IN VECTOR FORM. THE ALGORITHM IS SO ARRANGED THAT

IN ADDITION TO SOLVE RX=Z, SET RIN(N*(N+1)/2+1)=Z(1), ETC.,

AND SET RIN((N+1)*(N+2)/2)=-1. CALL THE SUBROUTINE USING N+1

INSTEAD OF N. ON RETURN THE FIRST N ENTRIES OF COLUMN N+1

UTINV010

UTINV020

UTINV030

UTINV040

UTINV050

UTINV060

UTINV070

UTINV080

UTINV090

SUBROUTINE UTINV (RIN, N, ROUT)

WILL CONTAIN X.

THE RESULT CAN OVERWRITE THE INPUT.

	SUBRO	DUTINE	UTIRO	W (RIN+N	ROU	T,NRY)					UTIR0010
	х х									UTIR0020	
	TO CO	OMPUTE	THE IN	VERSE OF	AN	UPPER T	RIANGUL	AR (VE	CTOR	STOREn)	UTIR0030
	MATR	[X ∦HE	N THE L	OWER POR	TION	_OF THE	INVERS	F IS G	IVFN		UTIR0040
		_		,				•			UTIRO050
	(DN INP	UT:	·							UTIR0060
											UTIR0070
	_	RX	RXY		*	*			RX	RXY	UTIR008 0
	RIN=			ROUT=			WHERE	R≕			UTIR0090
		*	*		0	₽Y**-1		ţ,	0	RY	UTIP0100
											UTIR0110
	ON OUTPUT: RIN IS UNCHANGED AND ROUT=R**-1								UTIR0120		
	<pre>>THE RESULT CAN OVER-WRITE THE INPUT (I.E. RIN=ROUT)</pre>									UTIR0130	
											UTIR0140
	RIN(↓*(N+1)/2)	INPUT V	Εςτο	R STORE	D TRIAN	GULAR	MATRI	X	UTIR0150
				THE BOT	том	NRY ROW	'S ARE I	GNORED			UTIR0160
	N			MATRIX	DIMF	NSION					UTIR0170
	ROUT	(N*(N+	1)/2)	OUTPUT	VECT	OR STOR	ED MATR	IX. ON	INPL	IT THE	UTIR0180
				BOTTOM	NRY	ROWS CO	NTAIN T	HE LOW	ER PC	RTION	UTIR0190
				0F R**-	1. 0	N OUTPU	T ROUT=	R**-1			UTIRO200
	NRY DIMENS					F LOWER	(ALREA	nY INV	ERTEL))	UTIRO210
			•	TPIANGU	LAR	R. IF N	RY=0, 0	RDINAR	Υ ΜΑΤ	RIX	UTIR0220
				INVERSI	ON R	ESULTS.					UTIR0230
	COGN	EZANT	PERSONS	: G.J.	BIER	MAN/M•W	 NEAD 	(JPL	MARCH	1977)	UTIP0250
											UTIR0260
	DOOBI	E PRE	CISION	RIN(1)	+ R0	UT(1)+	SUM, ZE	RO; ON	E+ D1	INV	UTIR0270
	DATA	ONE	/1.D0/,	ZER0/0.	D0/						UTIR0280
											UTIR0290
		INITIA	LIZATIO	N							UTIR0300
											UTIRO310
	NK=N>	⊧(N+1)	/2			A NO. ELÉMENTS IN R					UTIR0320
	ISTR	r=n−nr	Y			⊜ FIRST	ROW TO	BE IN	VERTE	D	UTIR0330
	IRLS	r=istr	T+1			ស IRLST	UTIR0340				
	Il=IS	STRT*I	RLST/2			R II=DI	UTIR0350				
	DO 40) IROW	=ISTRT,	1 -1							UTIR0360
	IF	(RIN(II).NE.	ZERO) GO	то	10					UTIR0370
	WR.	[TE (6	,50) IR	OW							UTIP0380
	RE	TURN					UTIR0390				
10	DII	4V=0NE	/RIN(II)							UTIR0400
	ROL	JT(II)	=DTNV								UTIR0410
	KJ	5=NR+I	ROW			ត KJ(ST	ART)				UTIR0420
	IKS	5=II+I	ROW			⊜ IK(ST	ART)				UTIR0430
											UTIRO440
	IF	(IRLS	T.GT.N)	GO TO 3	5						UTIR0450
	DO	30 J=	N IRLST	• <u>-1</u>							UTIR0460
	í	(JS≃KJ	s-J								UTIR0470
	SUM=ZERO IK=IKS								UTIRO480		
									UTIR0490		
	ł	(J≒KJS									UTIR0500
			_								UTIR0510
	DO 20 K=IRLST,J						UTIR0520				
		KJ=K	J+1								UTIR0530
1		SUM≕	SUM+RIN	(IK)*ROU	E(KJ)					UTIR0540
/											

-

С

1

C C C C

с

Ŷ

С

	20	IK=IK+K	UTIR0550
С			UTIR0560
	30	ROUT(KJS)=-SUM*DINV	UTIR0570
	35	IRLST=IROW	UTIR0580
	40	II=II-IROW	UTIR0590
		RETURN	- UTIR0600
	50	FORMAT (1H0,10X, RIN DIAGONAL, 14, IS ZERO)	UTIR0610
		END	UTIR0620

77–26

С									
•		SUBROUTINE WGS	(W, IMAXW, IW, JW, D, U, V)	/GS00010					
с c		MODIFIED GRAMM-SCHMIDT ALGORITHM FOR REDUCING WDW(**T) TO UDU(**T)							
č		PERMITING D FLE	MENTS STOPE ON THE DIAGONAL	KGS00000					
č			PUCKER ON THE DIROUME	WGS00050					
č		W(IW.JW)	INPUT MATRIX TO BE REDUCED TO TRIANGULAR FORM.	WGS00060					
С		,	THIS MATRIX IS DESTROYED BY THE CALCULATION "	WGS00070					
C			IW.LE.IMAXW.	WGS00080					
ç		D(IW)	VECTOR OF NON-NEGATIVE WEIGHTS FOR THE	¥6500090					
с С			URTHOGONALIZATION PROCESS. THE DYS ARE UNCHANGED	WG500100					
ĉ		EI(TW*(TW+1)/2)	OUTPUT UPPER TRIANGULAR VECTOR STORED OUTPUT	WG500120					
č		A(MM)	WORK VECTOR	WGS00130					
Ĉ				W6S00140					
С			(SEE BOOK:	WGS00150					
C		FACTORIZATION METHODS FOR DISCRETE SEQUENTIAL ESTIMATION **							
ç		Forestateou	BY G.J.BIERMAN)	WG500170					
L C		ESTIMATION		WG500100					
č		COGNIZANT PERS	DNS: G.J.BIERMAN/M.W.NEAD (JPL: MARCH 1977)	W6500200					
č		CONTEXAL IERS	MAT GEOTOTEMPROVENENCE COLOR PARION 1997	WGS00210					
-		IMPLICIT DOUBLE	PRECISION (A-H, 0-Z)	W6500220					
		DIMENSION W(IM	AXW,1), D(1), U(1), V(1)	W6S00230					
С				WGS00240					
		Z=0.0		WGS00250					
		ONE=1.0	1	WG500200					
			*1	WGS00270					
		DO 40 K=1.JW		WGS00290					
		$V(K) = W(J \cdot K)$							
		U(K)=D(K)*!	V(K) DU HERE IS USED AS A WORK VECTOR	WGS00310					
	40								
			© TO 100	W6500330					
			30 10 100	W6500350					
•	,	IF (SUM+GT+Z) DINV=ONE/SUM	W6500360					
		JM1=J-1		W6S00370					
		DO 70 K=1,JM	1	WGS00380					
		SUM=Z		WGS00390					
				WGS00400					
	50		1/*U(1)+SUM	MC200410					
с		2014-2014-011	1 A	WG500430					
•		DO 60 I=1.	WU	WG500440					
	60	W(K'I)=M	(K,I)-SUM*V(I)	₩¢Sn0450					
	70	W(JIK)=SUM	R EQ.(4.10) OF BOOK	WGS00460					
	100	CONTINUE	D U(K+J) STORED IN W(J+K)	WGS00470					
C									
5		THE LOWER PART OF WID UTRANSPOSE							
C		1=LT							
		DO 110 J=1,IW							
		DO 110 I=1+J							
	-	IJ=IJ+1		WGS00540					
C	110	U(I))#4(J⁺	I)	W6500550					
			02	WG500560					
			22	10300379 W6500580					

References

- Lawson, C. L., Hanson, R. J., <u>Solving Least Squares Problems</u>, 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., <u>Factorization Methods for Discrete Sequential</u> Estimation, Academic Press, New York (1977).

-

•

v