# A computational algorithm for sequential estimation*

R. J. Hanson[†] and P. Dyer[‡]

[†]*Section 391, California Institute of Technology, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, California 91103, USA*

[‡]*Imperial Chemical Industries Ltd., Central Instrument Research Laboratory, Bozedown House, Whitchurch Hill, Reading, Berkshire*

**This paper details a highly reliable computational algorithm for sequential least squares estimation (filtering) with process noise. The various modular components of the algorithm are described in detail so that their conversion to computer code is straightforward. These components can also be used to solve any least squares problem with possibly rank deficient coefficient matrices.**

(Received February 1969, Revised October 1970)

In this paper we will describe a reliable computational procedure for estimating the state vector of a noisy system from a set of noisy measurements. The state of the system,

$$x(i) = [x_1(i), x_2(i), \ldots, x_n(i)]^T ,$$

is described by a sequence of transition equations,

$$x(k+1) = F(k)\,x(k) + G(k)\,\omega(k)\,, \quad k = 0, 1, \ldots, N-1, \quad (1)$$

where $F(k)$ and $G(k)$ are $n \times n$ matrices and $\omega(k) = [\omega_1(k), \omega_2(k), \ldots, \omega_n(k)]^T$ is the process noise vector. The measurements $z(k) = [z_1(k), z_2(k), \ldots, z_m(k)]^T$ are given by,

$$z(k) = H(k)\,x(k) + Q(k)\,v(k) \quad (2)$$

where $H(k)$ and $Q(k)$ are $m \times n$ and $m \times m$ matrices respectively and $v(k) = [v_1(k), v_2(k), \ldots, v_m(k)]^T$ is the measurement noise vector.

An estimation procedure for sequentially estimating the state $x(k)$, $(k = 0, 1, \ldots, N-1)$, using orthonormal Householder transformations was described in a recent paper by Dyer and McReynolds (1969). That paper showed that this procedure was equivalent to, and substantially more accurate than, the Kalman Filter (1960) and gave numerical results. However, few details were given of the computational algorithm and there was little effort made to maximise the efficiency of the routine. In this paper details of a refined form of the algorithm are given.

These details amount, in part, to referring the reader to the literature which describes reliable computational methods for solving a linear least squares system which may be of deficient rank (or nearly so) from a numerical point of view. This will be of great help to the reader who must implement the algorithm for some technological application.

It should be stated that this algorithm requires a rather large investment in programming to develop from the beginning. (We feel one-man year is a good estimate). We will, however, provide a set of (documented FORTRAN IV) subroutines to any interested requester. There is one feature of the present algorithm which we feel more than compensates for its complexity: it is completely reliable in the sense that rank deficiencies will not cause a system to fail. Also, the use of orthonormal transformations minimises the propogation of data errors in the computation. This program can, therefore, be a welcome component of many automatic control systems.

## Description of the algorithm

The problem of estimating $x(k)$ is equivalent to minimising

$$J(k) = \sum_{i=1}^{k} \{\|v(i)\|^2 + \|w(i)\|^2\} + \|x(1) - \bar{x}(1)\|^2_{\Lambda^{-1}(1)} \quad (3)$$

with respect to the random sequences $v(i)$ and $w(i)$, $(i = 1, 2, \ldots, k)$, subject to the constraints of equations (1) and (2). In equation (3), $\bar{x}(1)$ denotes the *a priori* mean of $x(1)$, while $\Lambda(1)$ denotes the *a priori* covariance of $x(1)$.

Let $J_{\text{opt}}(k)$ denote the minimum return function[†] for this problem expressed in terms of $x(k)$. Then

$$J_{\text{opt}}(k) = \|x(k) - \bar{x}(k)\|^2{}_{\Lambda^{-1}(k)} + r^2(k) \quad (4)$$

Here $\bar{x}(k)$ is the conditional mean of $x(k)$, $\Lambda(k)$ is the conditional covariance, and $r^2(k)$ denotes the sum of the squares of the residuals.

The Dyer-McReynolds algorithm computes $R(k)$ and $d(k)$ where,

$$\begin{aligned} R(k) &= \Lambda^{-\frac{1}{2}}(k) \\ d(k) &= \Lambda^{-\frac{1}{2}}(k)\,\bar{x}(k) \end{aligned} \quad (5)$$

In terms of $R(k)$ and $d(k)$, the return $J_{\text{opt}}(k)$ is given by

$$J_{\text{opt}}(k) = \|R(k)\,x(k) - d(k)\|^2 + r^2(k) \quad (6)$$

Clearly, if $R(k)$ is non-singular, $\bar{x}(k)$ and $\Lambda(k)$ are given by,

$$\bar{x}(k) = R^{-1}(k)\,d(k)$$

and $\quad (7)$

$$\Lambda(k) = R^{-1}(k)\,R^{-1}(k)^T$$

The details of the algorithm will be developed in two steps. First, the measurements at the $k$th stage will be incorporated with the *a priori* information. Secondly, the information will be transformed from the $k$th to the $k+1$st stage, corrupted by the effects of process noise.

The best estimate of $x(k)$ employing measurements $z(1), \ldots, z(k-1)$ is obtained by minimising,

$$\tilde{J}(k) = \sum_{i=1}^{k-1} \|v(i)\|^2 + \sum_{i=1}^{k} \|w(i)\|^2 + \|x(1) - \bar{x}(1)\|^2_{\Lambda^{-1}(1)} \quad (8)$$

subject to the constraints imposed by equations (1) and (2).

Note that,

$$J(k) = \tilde{J}(k) + \|v(k)\|^2 \ .$$

*Step* 1: *Measurements*

Now assume that $\tilde{J}(k)$ is given by

$$\tilde{J}(k) = \|\tilde{R}(k)\,x(k) - \tilde{d}(k)\|^2 + \tilde{r}^2(k-1) \quad (9)$$

(This will normally be the case. At the initial time $\tilde{R}(1)$ and $\tilde{d}(1)$ are formed from the *a priori* covariance and mean.) The

*This paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under Contract No. NAS 7-100, sponsored by the National Aeronautics and Space Administration.

[†]See Cox (1964) for the formulation of sequential estimation in terms of dynamic programming.

inclusion of measurements implies the minimisation of,

$$J(k) = \|\tilde{R}(k)\,x(k) - \tilde{d}(k)\|^2 + \|v(k)\|^2 + \tilde{r}^2(k) \qquad (10)$$

Substituting* for $v(k)$ from equation (2) gives,

$$J(k) = \|\tilde{R}(k)\,x(k) - \tilde{d}(k)\|^2 + \|Q^{-1}(k)\,H(k)\,x(k) - Q^{-1}(k)\,z(k)\|^2 + \tilde{r}^2(k) \qquad (11)$$

which may be written,

$$J(k) = \left\| \begin{bmatrix} \tilde{R}(k) \\ Q^{-1}(k)\,H(k) \end{bmatrix} x(k) - \begin{bmatrix} \tilde{d}(k) \\ Q^{-1}(k)\,z(k) \end{bmatrix} \right\|^2 + \tilde{r}^2(k) \qquad (12)$$

Now an orthogonal $(n + m) \times (n + m)$ matrix, $P$, is constructed such that,

$$P \begin{bmatrix} \overbrace{\tilde{R}(k) \qquad\qquad}^{n} \\ Q^{-1}(k)\,H(k) \end{bmatrix} \begin{matrix} \}n \\ \}m \end{matrix} = \begin{bmatrix} \overbrace{R(k)}^{n} \\ 0 \end{bmatrix} \begin{matrix} \}n \\ \}m \end{matrix} \qquad (13)$$

Here $R(k)$ is an upper triangular matrix. Let $d(k)$ and $d'(k)$ be defined by,

$$P \begin{bmatrix} \tilde{d}(k) \\ Q^{-1}z(k) \end{bmatrix} \begin{matrix} \}n \\ \}m \end{matrix} = \begin{bmatrix} d(k) \\ d'(k) \end{bmatrix} \begin{matrix} \}n \\ \}m \end{matrix} \qquad (14)$$

The matrix $P$ is a product of Householder transformations, i.e.

$$P = P_n \cdot P_{n-1} \ldots P_1$$

where

$$P_i = I_l + u_i u_i^T/\beta_i\,, \ (i = 1, \ldots, n)\,, \ (l = m+n)\ .$$

Each $P_i$ is orthonormal and symmetric. It should be noted, however, that none of the full $(n + m) \times (n + m)$ matrices $P_i$ have to be formed explicitly. It is only necessary to store the $l$-vector $u_i$ and the scalar $\beta_i$. Further details regarding the construction of these parameters are given in Appendix B, Algorithm 1.

The return $J(k)$ may now be written,

$$J(k) = \|R(k)\,x(k) - d(k)\|^2 + r^2(k) \qquad (15)$$

where $r^2(k) = \tilde{r}^2(k) + \|d'(k)\|^2$. The vectors $d(k)$ and $d'(k)$ are defined in equation (14).

The best estimate of $x(k)$ and its covariance are given by,

$$\begin{aligned} \bar{x}(k) &= R^{-1}(k)\,d(k) \\ \Lambda(k) &= R^{-1}(k)\,R^{-1}(k)^T \end{aligned} \qquad (16)$$

Details of the computation of $\bar{x}(k)$ and $\Lambda(k)$ are given in Algorithms 3, 4, and 5 of Appendix B, and the sequential processing of new data is outlined in Algorithm 2.

*Step* 2: *Mapping and Process Noise*

Mapping forwards introduces process noise, and the return $\tilde{J}(k + 1)$ is given by,

$$\tilde{J}(k + 1) = \|w(k)\|^2 + \|R(k)\,x(k) - d(k)\|^2 + r^2(k) \qquad (17)$$

From equation (1),

$$x(k) = F^{-1}(k)\,(x(k + 1) - G(k)\,w(k))$$

Hence writing equation (17) in terms of $x(k + 1)$,

$$\tilde{J}(k + 1) = \|w(k)\|^2 + \|R(k)\,F^{-1}(k)\,x(k + 1) - R(k)\,F^{-1}(k)\,G(k)\,w(k) - d(k)\|^2 \qquad (18)$$

This equation must now be minimised with respect to $w(k)$ and $w(k)$ eliminated. Equation (18) may be written,

$$\tilde{J}(k + 1) = \left\| \begin{bmatrix} I_n & 0 \\ R(k)\,F^{-1}(k)\,G(k) & R(k)\,F^{-1}(k) \end{bmatrix} \cdot \begin{bmatrix} w(k) \\ x(k + 1) \end{bmatrix} - \begin{bmatrix} 0 \\ d(k) \end{bmatrix} \right\|^2 + r^2(k) \qquad (19)$$

The matrix $I_n$ appearing in equation (19) is the $n \times n$ identity matrix.

The coefficient matrices $R(k)\,F^{-1}(k)\,G(k)$ and $R(k)\,F^{-1}(k)$ in equation (19) are computed in the following way.

A product of $n - 1$ Householder orthonormal transformations

$$S = S_{n-1} \ldots S_1,\ S_i = (I_n + u_i u_i^T/\beta_i),\ (i = 1, \ldots, n - 1),$$

is found such that

$$F(k) = S_1 \ldots S_{n-1} T \qquad (20)$$

where $T$ is upper triangular.

Since $F(k)$ is nonsingular,

$$F^{-1}(k) = T^{-1} S_{n-1} \ldots S_1 \qquad (21)$$

Then premultiplying by $R(k)$ and postmultiplying by $G(k)$ gives,

$$R(k)\,F^{-1}(k)\,G(k) = R(k)\,(T^{-1}(S_{n-1} \ldots S_1 G(k))) \qquad (22)$$

while,

$$R(k)\,F^{-1}(k) = R(k)\,(\ldots(T^{-1}S_{n-1}) \ldots S_1) \qquad (23)$$

In Algorithm 6 it will be shown that the formation of the matrix products on the left hand side of equations (22) and (23) require only $n$ additional storage locations.

A $(2n) \times (2n)$ orthonormal matrix, again a product of $2n - 1$ Householder transformations:

$$X = X_{2n-1} \ldots X_1 \ X_i = I_{2n} + u_i u_i^T/\beta_i\,,\ (i = 1, \ldots, 2n-1)\,,$$

is now chosen such that,

$$X \begin{bmatrix} I_n & 0 \\ R(k)\,F^{-1}(k)\,G(k), & R(k)\,F^{-1}(k) \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & \tilde{R}(k + 1) \end{bmatrix} \qquad (24)$$

In Algorithm 7 we will show that the right member of equation (24) can be generated in such a way that only $2{\cdot}5n^2 + 3{\cdot}5n + 1$ memory locations are needed at each step of the calculation. Exactly $2{\cdot}5n^2$ of these cells are the working arrays which initially held the matrices $F(k)$, $G(k)$, and $R(k)$.

We further remark here that the matrix $A$ in the right member of equation (24) is nonsingular. This follows from the observation that $A$ is upper triangular and the modulus of each diagonal term has the value one at least.

With,

$$X \begin{bmatrix} 0 \\ d(k) \end{bmatrix} = \begin{bmatrix} \tilde{d}'(k + 1) \\ \tilde{d}(k + 1) \end{bmatrix} \begin{matrix} \}n \\ \}n \end{matrix} \qquad (25)$$

the value of $\tilde{J}(k + 1)$ is,

$$\tilde{J}(k + 1) = \|\tilde{R}(k + 1)\,x(k + 1) - \tilde{d}(k + 1)\|^2 + \|Aw(k) + Bx(k + 1) - \tilde{d}'(k + 1)\|^2$$

If $\tilde{R}(k + 1)$ is nonsingular the best estimate of $x(k + 1)$, given measurements through the $k$th stage, is given by,

$$\tilde{x}(k + 1) = \tilde{R}^{-1}(k + 1)\,\tilde{d}(k + 1) \qquad (26)$$

The smoothed value of $w(k)$ is given by,

$$w(k) = A^{-1}[\tilde{d}'(k + 1) - Bx(k + 1)] \qquad (27)$$

The covariance associated with $\tilde{x}(k + 1)$ is given by,

*If the measurements are genuinely noisy then $Q(k)$ is nonsingular.

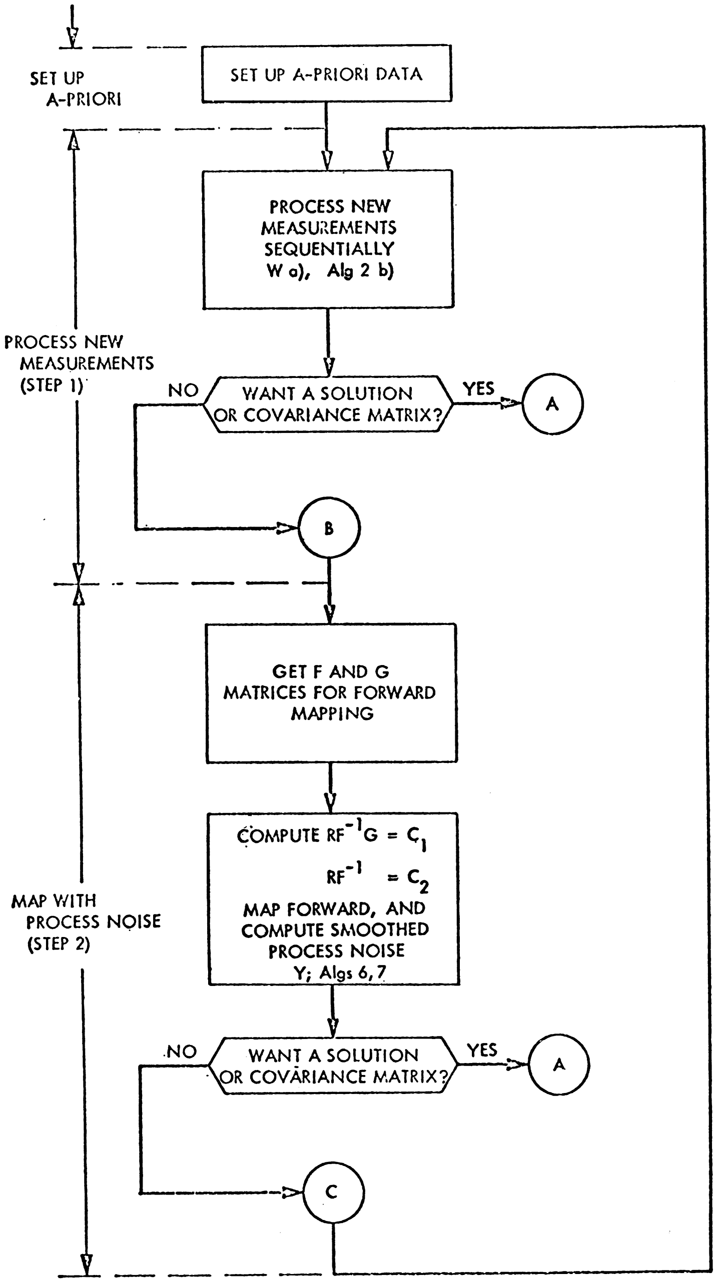
$$\tilde{\Lambda}(k + 1) = [\tilde{R}^{-1}(k + 1)]\,[\tilde{R}^{-1}(k + 1)]^T \qquad (28)$$

The hypothesis that $\tilde{R}(k + 1)$ be nonsingular is not critical. We can replace the indicated inverse in equation (25) by a pseudoinverse (Lawson and Hanson, 1968) which always exists.

In this case the covariance matrix of equation (28) no longer exists; one can, however agree to solve for certain of the variables and set the remaining ones to zero. This amounts to obtaining a pseudoinverse solution (in a limiting sense) with a weighted euclidean metric. In the latter case one can obtain a covariance matrix for the variables which were solved for. The details of this are given in Algorithm 5 of Appendix B.

## Appendix A

The flow diagram of Appendix A is intended to indicate the overall structure of the filter and how it makes use of the various component algorithms of Appendix B. These algorithms of Appendix B can be used for solving any least squares problem, of course.


SET UP A-PRIORI
SET UP A-PRIORI DATA
PROCESS NEW MEASUREMENTS SEQUENTIALLY W a), Alg 2 b)
PROCESS NEW MEASUREMENTS (STEP 1)
NO
WANT A SOLUTION OR COVARIANCE MATRIX?
YES
A
B
GET F AND G MATRICES FOR FORWARD MAPPING
COMPUTE RF$^{-1}$G = C$_1$
RF$^{-1}$ = C$_2$
MAP FORWARD, AND COMPUTE SMOOTHED PROCESS NOISE
Y; Algs 6,7
MAP WITH PROCESS NOISE (STEP 2)
NO
WANT A SOLUTION OR COVARIANCE MATRIX?
YES
A
C

a) W AND Y REFER TO WORKING AREAS IN THE COMPUTER OF DIMENSIONS $(n + 1 + \mu) \times (n + 1)$ AND $(n + 1) \times (2n + 1)$ RESPECTIVELY. (HERE $\mu$ = MAX NUMBER OF NEW MEASUREMENTS PROCESSED AT ONE TIME.)

b) Alg N DENOTES THE ALGORITHM IN APPENDIX B WHICH DESCRIBES THE RELEVANT COMPUTATION

**Fig. 1. Flow sequence for the filter with process noise**

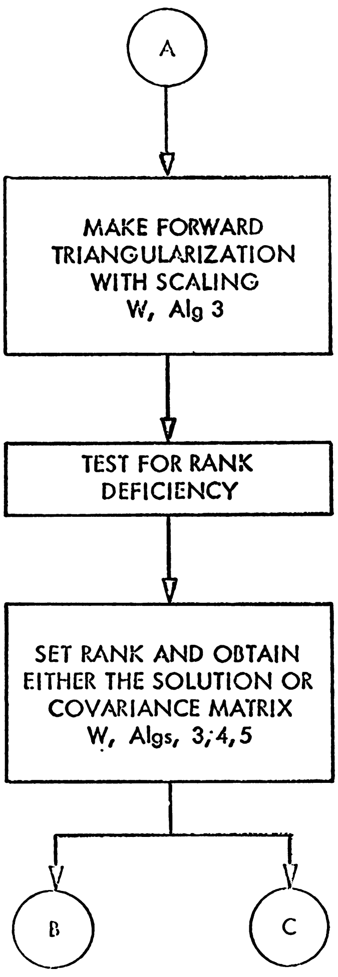

A
MAKE FORWARD TRIANGULARIZATION WITH SCALING
W, Alg 3
TEST FOR RANK DEFICIENCY
SET RANK AND OBTAIN EITHER THE SOLUTION OR COVARIANCE MATRIX
W, Algs, 3,4,5
B
C

**Fig. 2. Determination of rank, computation of solution and covariance**

## Appendix B

Many of the algorithms presented below have appeared in a slightly different form in Lawson and Hanson (1968). Algorithm 3 is essentially due to Businger and Golub (1965) using a special case of Algorithm 1.

*Algorithm* 1:
The basic Householder transformation; its construction and application.

*PURPOSE*

Suppose that $y = [y_1, \ldots, y_m]^T$ is an arbitrary vector of length $m$. Given three non-negative integers $l$, $t$, and $m$, with $l + t < m$, we wish to construct an orthonormal transformation $Q = I_m + uu^T/\beta$ such that for $Qy$:

(*a*) Components 1 through $l$ are to be left unchanged
(*b*) Components $l + 1$ is permitted to change
(*c*) Components $l + 2$ through $l + t + 1$ are to be left unchanged
(*d*) Components $l + t + 2$ through $m$ are to be zero.

*METHOD*

(See Lawson and Hanson (1968) for details.)

The input to this algorithm will consist of the three previously mentioned integers $l$, $t$, and $m$, the $m$-vector $y$ and a single free cell to hold a scalar $u_p$ upon output.

For later reference we will designate the output of this algorithm $H1(l, t, m, u_p, y)$. The vector $u$ will occupy just those positions of $y$ which were implicitly zeroed plus the one extra location labelled $u_p$.

Assume now that $c = [c_1, \ldots, c_m]^T$ is an $m$-vector, and that we wish to compute the matrix product $Qc$ and place it into the storage previously occupied by $c$.

From the equality $c^T Q = (Qc)^T$ ($Q$ is symmetric) we see that only matrix products of the form $Qc$ need be discussed here.

The matrix product $Qc$ is given by,

$$Qc = c + [(u^T c)/\beta]u \tag{A1.1}$$

and so the matrix $Q$ need not be explicitly formed. The replacement of the vector $c$ in storage by the right side of (A1.1) will be designated by the symbol $H2(l, t, m, u, u_p, c)$.

Note that only those components numbered $p = l + 1$, and $q, \ldots, m$, $(q = l + t + 2)$, are changed by premultiplication of $c$ by $Q$. Further, if these components of $c$ are known to be zero (or, more generally $u^T c = 0$) then $Qc = c$ and no explicit computation is required.

*Algorithm* 2:

*PURPOSE*

Sequential acceptance of equations to achieve upper triangular form as a preliminary step.

*METHOD*

Suppose we have a large linear least squares problem of the form,

$$Ax = b \tag{A2.1}$$

The matrix $A$ and the vector $b$ are written in partitioned form,

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_q \end{bmatrix}, \; b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{bmatrix} \tag{A2.2}$$

where each matrix $A_t$ is $m_i \times n$ and each $b_i$ is a vector of length $m_i$. The integers $m_i$ can be as small as one.

Let $m = m_1 + \ldots + m_q$. We construct orthonormal matrices $Q_1, \ldots, Q_q$, each of which are a direct sum of an identity matrix and products of at most $n$ Householder transformations and permutation matrices $P_2, \ldots, P_q$ such that

$$Q_q P_q \ldots Q_2 P_2 Q_1 [A, b] = \begin{array}{c} \begin{array}{cc} \overbrace{\;}^{n} & \overbrace{\;}^{1} \end{array} \\ \begin{bmatrix} R, & d \\ 0, & r \\ 0, & 0 \end{bmatrix} \end{array} \begin{array}{l} \\ \}n \\ \}1 \\ \}m - n - 1 \end{array} \tag{A2.3}$$

Here $R$ is upper triangular, $d$ is an $n$-vector and $|r|$ is the residual vector length if $R$ is nonsingular; no zeros are stored.

The computational details for this algorithm are found in Lawson and Hanson (1968).

*Algorithm* 3:

*PURPOSE*

Forward triangularisation of square matrices with column scaling, column interchanges and rank determination.

*METHOD*

Supose we wish to solve an $n \times n$ system (which may have a singular coefficient matrix) in the least squares sense:

$$Ax = b \tag{A3.1}$$

Here $A$ is an $n \times n$ real matrix of rank $r \leq n$ and $b$ is a real $n$-vector. We construct a nonsingular diagonal matrix $D$, a permutation matrix $P$ and an orthonormal matrix $Q = Q_{n-1} \ldots Q_1$, $(Q_i = I_n + u_i u_i^T/\beta_i)$, $(i = 1, \ldots, n - 1)$, such that

$$A = Q^T T P^T D^{-1} \tag{A3.2}$$

so that if $A$ is nonsingular,

$$A^{-1} = DPT^{-1}Q$$

Here, in general, $T$ is upper triangular with its first $r$ diagonal terms nonzero and with its last $n - r$ rows identically zero.

We remark here that the matrix in the right member of equation (A3.2) may actually be a replacement for $A$ in the following sense:

The data which constitutes the matrix $A$ in the machine is usually only a representative member of a class of matrices $\mathscr{A}$ which is determined by the original uncertainty in the data and the uncertainty caused by subsequent computer arithmetic operations on this data. Thus, it may be apparent during the calculation that there is a matrix $A \in \mathscr{A}$ such that rank $(\tilde{A}) = \min\limits_{A \in \mathscr{A}}$ rank $(A)$ it is such a matrix $\tilde{A}$ which replaces $A$ in equations (A3.1) and (A3.2).

See Lawson and Hanson (1968) and Businger and Golub (1965).

*Algorithm* 4:

*PURPOSE*

Computing the solution of minimum length for rank deficient problems.

*METHOD*

The method described in Algorithm 3 allows us to assume, with no loss of generality, that for a given system as in (A3.1), we may write:

$$A = Q^T T P^T D^{-1} \; . \tag{A4.1}$$

Here $Q^T$ is a product of $n - 1$ Householder transformations, $T$ is upper triangular with its first $r$ diagonal terms nonzero and its last $n - r$ rows identically zero, $P^T$ is a permutation matrix, and $D^{-1}$ is a diagonal matrix.

We will first find $r$ Householder transformations $K_r, \ldots, K_1$ such that

$$TK_r \ldots K_1 = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} \tag{A4.2}$$

where $S$ is $r \times r$ upper triangular and nonsingular.

The solution of minimum length or the pseudoinverse solution (Lawson and Hanson, 1959) (with the norm $\|x\|^2 = x^T D^{-2} x$) is given by

$$y = (Q_{n-1} \ldots Q_1 b) \tag{A4.3}$$

$$c = 1\text{st } r \text{ components of } y \, , \tag{A4.4}$$

$$d = S^{-1} c \tag{A4.5}$$

$$e = K_r \ldots K_1 \begin{bmatrix} d \\ 0 \end{bmatrix} \begin{array}{l} \}r \\ \}n - r \end{array} \tag{A4.6}$$

and

$$x = D(Pe) \tag{A4.7}$$

See Lawson and Hanson (1968) for more details.

*Algorithm* 5:

*PURPOSE*

Computation of the covariance matrix.

*METHOD*

Let us suppose, as in Algorithm 4, that we have

$$A = Q^T T P^T D^{-1} \; . \tag{A5.1}$$

Let

$$T = \begin{array}{c} \begin{array}{cc} \overbrace{\;}^{r} & \overbrace{\;}^{n-r} \end{array} \\ \begin{bmatrix} T_{11} & T_{12} \\ 0 & 0 \end{bmatrix} \end{array} \begin{array}{l} \\ \}r \\ \}n - r \end{array} \tag{A5.2}$$

where $T_{11}$ is $r \times r$, upper triangular and nonsingular. In case either $r = \text{rank}(A) = \text{rank}(T) = n$, or the solution is obtained by setting the last $n - r$ components of $P^T D^{-1} y$ to zero, the (unscaled) covariance matrix of those variables which were solved for can be defined by

$$C(A) = DP \begin{bmatrix} T_{11}^{-1}(T_{11}^{-1})^T & 0 \\ 0 & 0 \end{bmatrix} P^T D \qquad \text{(A5.3)}$$

If $r = \text{rank}(A) = n$, then

$$C(A) = (A^T A)^{-1} \qquad \text{(A5.4)}$$

as can easily be verified. (See Businger and Golub, 1965)

We will now describe the algorithm for computing the right side of equation (A5.3). The matrix $T$ will be in the first $r$ rows of the upper triangular part of the working array $W$.

```
Type:  Integer           $r, i, j, n, k, l, p$;
       Real              $W$;
       Double Precision  $s$;

Procedure: Covariance matrix computation
Step Number   Description
 1            $W(j{:}j, j{:}j) := 1/W(j{:}j, j{:}j), (j = 1, \ldots, r)$.
 2            If $r = 1$ go to step 11. Else
 3            Set $j := 2$.
 4            Set $k := r + 2 - j$.
 5            Set $i := 2$.
 6            Set $p := k + 1 - i, s := 0$,
 7            Set $s := s + W(p{:}p, l{:}l) \cdot W(l{:}l, k{:}k)$,
              $(l = p + 1, \ldots, k)$.
 8            Set $W(p{:}p, k{:}k) := -s \cdot W(p{:}p, p{:}p)$.
 9            If $i < k$, set $i := i + 1$ and go to step 6. Else
10            If $j < r$, set $j := j + 1$ and go to step 4. Else
11            Set $l := 1$.
REMARK
The matrix $T_{11}^{-1}$ has now replaced the matrix $T_{11}$ in the storage array $W$.
12            Set $i := l$.
13            Set $s := 0$.
14            Set $s := s + W(l{:}l, j{:}j) \cdot W(i{:}i, j{:}j)$,
              $(j = i, \ldots, r)$.
15            Set $W(l{:}l, i{:}i) := s$.
16            If $i < r, i := i + 1$ and go to step 13. Else
17            If $l < r$, set $l := l + 1$ and go to step 12. Else
REMARK
The upper triangular part of the symmetric matrix $T_{11}^{-1} (T_{11}^{-1})^T$ has now replaced $T_{11}^{-1}$ in storage.
18            Zero the last $n - r$ columns of the upper triangular part of $W$.
19            Compute $W := PWP^T$.
20            Compute $W := DWD$.
21            The upper triangular part of the symmetric matrix $C(A)$ of equation (A5.3) is now in the
              upper triangular part of the array $W$.
REMARK
In steps 19 and 20 only the upper triangular part of $W$ need be referenced. We will not comment on these details.
```

*Algorithm* 6:

*PURPOSE*

Computation of the matrix products associated with forward mapping of process noise.

*METHOD*

In equations (22) and (23) we see that matrix products of the forms $RF^{-1}G$ and $RF^{-1}$ must be formed where $R$ is upper triangular, $F$ is nonsingular and $G$ is arbitrary. All of these matrices are $n \times n$.

Analogous with equation (22) set

$$F^{-1} = T^{-1} Q_{n-1} \cdots Q_1 \,,$$
$$(Q_i = I_n + u_i u_i^T / \beta_i \,, \; i = 1, \ldots, n-1) \,. \qquad \text{(A6.1)}$$

Then

$$RF^{-1}G = R(T^{-1} Q_{n-1} \cdots Q_1 G) \qquad \text{(A6.2)}$$

and

$$RF^{-1} = R(T^{-1} Q_{n-1} \cdots Q_1) \qquad \text{(A6.3)}$$

For the purpose of describing the formation of these matrix products, suppose that $R$ is located in the upper triangular part of a working array $W$ and that $F$ and $G$ are in partitioned form in an $n \times 2n$ working array $Y$. We will let $W(i_1{:}i_2, j_1{:}j_2)$ denote the subarray of $W$ consisting of rows $i_1$ through $i_2$ and columns $j_1$ through $j_2$. Analogous comments held for the array $Y$.

```
Type:  Integer  $i, j$;
       Real     $u(1{:}n), t(1{:}n),\ Y$
Step Number   Description
 1            Set $j := 1$
 2            If $j < n$, compute $H1(j-1, 0, n, u(j), Y(1{:}n, j{:}j))$ and next compute $H2(j - 1, 0, n,$
              $Y(1{:}n, j{:}j), u(j), Y(1{:}n, i{:}i)), (i = j + 1, \ldots, n)$; then set $j := j + 1$ and go to step 2. Else
REMARK
The matrix $T$ is in the upper triangular part of the left half of $Y$; $G$ is in the right half of $Y$.
 3            Compute $F^{-1}G$ by solving the $n$ systems of $n$ equations $FX = G$ for $X$; the matrix $X$ can
              replace $G$ in storage in the right half of $Y$.
 4            Set $t(j){:} = Y(j{:}j, j{:}j), (j = 1, \ldots, n)$.
 5            Compute the matrix $T^{-1}$; this matrix can replace $T$ in storage in the upper triangular
              part of the first $n$ columns of $Y$. (See Algorithm 5, Steps 1-10.)
REMARK
The matrix $F^{-1}G$ is now in the right half of $Y$.
 6            Set $j := n - 1$.
              If $j > 0$ first set $t(i) := Y(i{:}i, j{:}j)$ and then $Y(i{:}i,\ j{:}j) := 0,\ (i = j + 1, \ldots, n)$. Next
              compute $H2(j - 1, 0, n, t(1{:}n), u(j), Y(i{:}i, 1{:}n)), i = 1, \ldots, n), j{:}{=}j{-}$. Else
REMARK
In step 6 the last $n - j + 1$ columns of the left half of $Y$ are all that is affected by multiplication from the right by $Q_j$.
 7            The working array $Y$ contains the augmented matrix $[F^{-1}, F^{-1}G]$. Note that the order of
              these matrices is reversed from that required in Algorithm 7.
 8            Compute the product $R[F^{-1}, F^{-1}G]$. This matrix can replace $[F^{-1}, F^{-1}G]$ in the $Y$
              array.
```

*Algorithm* 7:

*PURPOSE*

Forward triangularisation when mapping forwards with process noise.

*METHOD*

As indicated in equation (30), we wish to find an orthonormal matrix $X$ such that for given $n \times n$ matrices $C_i$, $(i = 1, 2)$, and a given $n$-vector $d$,

$$X\tilde{S} = X \begin{bmatrix} I_n\ , & 0\ , & 0 \\ C_1\ , & C_2\ , & d \end{bmatrix} = \begin{bmatrix} A\ , & B\ , & \hat{e}_1 \\ 0\ , & R, & \hat{e}_2 \end{bmatrix} \qquad \text{(A7.1)}$$

where both matrices $A$ and $R$ are upper triangular. The vector $d_1$ is of length $n$ as are the vectors $\hat{e}_1$, $(i = 1, 2)$. The matrix $B$ will, in general, have no special structure. The definition of the matrix $\tilde{S}$ is self-explanatory.

If the $n \times 2n$ matrix $[C_1, C_2]$ occupies part of an $(n + 1) \times (2n + 1)$ working array $Y$, and if an $n \times n$ working array $W$ is available, then the right hand side of equation (A7.1) can be computed and stored in the working array $Y$ together with the upper triangular part of the array $W$. In total this requires $2.5n^2 + 3.5n + 1$ computer words; this is in marked contrast to the $4n^2 + 2n$ cells of memory which might at first seem to be required to calculate the right side of equation (A7.1).

Let $[c_1, \ldots, c_{2n}]$ denote the $2n$ column vectors of the $n \times 2n$ matrix $[C_1, C_2]$. The first column of the matrix which is the right factor of the middle term of equation (A7.1) is the $2n$ vector

$$w_1 = [\overbrace{1, 0, \ldots, 0}^{n}, c_1^T]^T \qquad \text{(A7.2)}$$

After constructing the Householder transformation

$$X_1 = I_{2n} + u_1 u_1^T / \beta_1$$

such that

$$X_1 w_1 = \pm[1 + \|c_1\|^2]^{\frac{1}{2}} \overbrace{[1, 0, \ldots, 0]^T}^{2n} \qquad \text{(A7.3)}$$

The details of Algorithm 1 show that:

1. After the matrix products
   $X_1[e_i^T, c_i^T]^T_, (i, = 2, \ldots, n)$,
   $X_1[\overbrace{0}^{n}, c_i^T]^T, (i = n + 1, .. 2n)$,
   and $X_1[0, d_1^T]^T$ are computed, only the first component or the last $n$ components are possibly changed. The vectors $e_i$ are the unit coordinate $n$-vectors.
2. Thus only one row of the matrices $A$ and $B$ and one component of the vector $e_1$ will be calculated at each step in the construction of a matrix $X = X_n \ldots X_1$ such that

$$\tilde{X}\tilde{S} = \begin{bmatrix} A & B & \hat{e}_1 \\ 0 & R & \tilde{e}_2 \end{bmatrix} \qquad \text{(A7.4)}$$

   The matrix $\tilde{R}$ of equation (A7.4) is $n \times n$ but is not necessarily upper triangular; $\tilde{e}_2$ is an $n$ vector.
3. As the rows of $A$ and $B$ and the components of $\hat{e}_1$ are calculated they can be placed into parts of the working arrays $Y$ and $W$ where space has come available.

We now present a step-by-step procedure which effects these space-and labour saving remarks.

```
Type:   Integer  $i, j, n$;
        Real     $t, Y, W$
Step Number  Description
 1           Move the $2n + 1$ components of the $n + 1$st
             row of $\tilde{S}$ (now in the 1st row of $Y$, say) to the
             $n + 1$st row of the working array $Y$.
 2           Set $j := 1$.
 3           Set $Y(1:1, i:j) := 0, (i = j, \ldots, 2n + 1)$ and
             $Y(1:1, j:j) := 1$.
 4           If $j \leq n$, compute
             $H1(0, 0, n + 1, t, Y(1:n + 1, j:j))$,
             and next compute
             $H2(0, 0, n + 1, Y(1:n + 1, j:j), t, Y(1:n + 1, i:i)), (i = j + 1, \ldots, 2n + 1)$,
             $Y(1:1, j:j) := Y(1:1, 2n + 1:2n + 1), Y(2:i, j:j) := Y(1:1, i:i), (i = n + 1, \ldots, 2n)$,
             $W(j:j, i:i) := Y(1:1, i:i), (i = j + 1, \ldots, n)$,
             $u(j) := Y(1:1, j:j)$; then set $j := j + 1$ and
             go to step 3. Else
```

*REMARK*

At this point $B^T$ occupies $Y(2:n + 1, 1:n)$; note that each column of $B^T$ moves in to occupy the storage implicitly zeroed with the successive Householder transformations; the strictly lower triangular part of the matrix $A^T$ is in the strictly lower part of the array $W$; diagonal terms of $A^T$ are now in $u(1:n)$.

```
 5           Triangularise the matrix $\tilde{R}$ now in $Y(2:n + 1, n + 1:2n)$ with Algorithm 3.
 6           Place the strictly lower part of $A^T$ into the
             lower part of $Y(2:n + 1, n + 1: 2n)$.
```

*REMARK*

Step 6 completes the forward mapping procedure; a solution and its covariance may be obtained by means of Algorithms 3-5.

The smoothed value of the process noise is then trivially computed by means of equations (27). Recall that $B^T$ is in $Y(2:n + 1, 1:n)$, the strictly lower part of $A^T$ is in the strictly lower part of $Y(2:n + 1, n + 1: 2n)$, the diagonal entries of $A^T$ are in $u(1:n)$, and the vector $d'(k + 1)$ of equation (27) is in $Y(1:1, 1:n)$.

To restart the basic cycle the upper triangular matrix $R$ together with the vector $\hat{e}_2$ of equation (A7.1) are now in the upper part of $Y(2:n + 1, n + 1:2n + 1)$ and must be copied to the upper triangular part of $W(1:n, 1:n + 1)$.

## References

BUSINGER, P., and GOLUB, G. (1965). Linear Least Squares Solution by Householder Transformations, *Numer. Math.*, Vol. 7, pp. 269-275.

COX, H. C. (1964). Estimation of State Variables via Dynamic Programming, *Proc. J.A.C.C.*, pp. 376-381.

DYER, P., and MCREYNOLDS, S. R. (1969). The Extension of Square-Root Filtering to Include Process Noise, *Journal of Opt.: Theory and Appl.*, 3, pp. 92-105.

GOLUB, G. H., and REINSCH, C. (1970). Singular Value Decomposition and Least Squares Solutions, *Numer. Math.*, Vol. 14, pp. 403-420.

KALMAN, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems, *Journal Basic Eng.*, Vol. 82, D, pp. 35-45.

LAWSON, C. L., and HANSON, R. J. (1968). Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems, Jet Propulsion Laboratory, Section 314 Technical Memorandum No. 200. *Math. of Comp.*, October 1969, Vol. 23, No, 108, pp. 787-812.