5122

JET PROPULSION LABORATORY

CALIFORNIA INSTITUTE OF TECHNOLOGY

PASADENA, CALIFORNIA

A COMPUTATIONAL ALGORITHM FOR SEQUENTIAL ESTIMATION *

Richard J. Hanson
Jet Propulsion Laboratory
Section 314

Peter Dyer
Jet Propulsion Laboratory
Section 311

* 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.

ABSTRACT

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.

KEYWORDS: Kalman filters, square root filters, sequential least squares estimation, numerical solutions of linear least squares.

ALGORITHM FOR SEQUENTIAL ESTIMATION

Introduction

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,

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

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,

$$z(k) = H(k)x(k) + Q(k)v(k) \tag{2}$$

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 (Ref. 1). That paper showed that this procedure was equivalent to, and substantially more accurate than, the Kalman Filter (Ref. 2) and gave some numerical results. However, few details were given of the computational algorithm and there was little effort made to maximize the efficiency of the routine. In this paper details of a refined form of the algorithm are given. This new algorithm is substantially faster and requires only half the storage of the algorithm indicated in Ref. 1. These points are described in Appendices A and B.

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 and roundoff errors are minimized. This program can, therefore, be a welcome component of many automatic control systems.

We will now review the algebraic operations required in the algorithm.

Description of the Algorithm

As stated above the problem is to estimate the state of a dynamic system in the presence of noise. It is assumed that the components of the noise vectors ω and v are statistically independent and gaussian with zero means and unit **variances.** This assumption is not restrictive because any set of correlated gaussian random **variables may** be linearly transformed to a new set of independent gaussian random variables. One technique which effects this transformation is as follows: Let $\omega(k)$ denote correlated **process** noise with covariance $C(k)$. Now employing the Cholesky square root algorithm (Ref. 3) a matrix $D(k)$ is found such that $C(k) = D(k)D(k)^T$. By setting $\omega = D(k)w$, equation (1) may be written

$$x(k+1) = F(k)\, x(k) + G(k)D(k)w(k)$$

where the components of $w(k)$ are independent random parameters with zero mean and unit covariance.

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

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

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 $J_{opt}(k)$ denote the minimum return function[1] for this problem expressed in terms of $x(k)$. Then

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

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,

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

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

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

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

$$\Lambda(k) = R^{-1}(k)\,R^{-1}(k)^T \tag{7}$$

The algorithm shall be developed in two steps. First, the measurements at the k^{th} stage will be incorporated with the <u>a priori</u> 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 minimizing,

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

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

---

1 See Cox, (Ref. 4) for the formulation of sequential estimation in terms of dynamic programming.

Note that,

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

Step 1: Measurements

Now assume that $\tilde{J}(k)$ has been transformed to,

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

(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 inclusion of measurements implies the minimization of,

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

Substituting[2] for v(k) from equation (2) gives,

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

which may be written,

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

Now an orthogonal (n+m) x (n+m) matrix, P, is constructed such that,

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

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

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

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

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

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

where

$$P_i = I_\ell + u_i^T u_i / \beta_i \quad , \quad (i = 1,\ldots,n) \quad , \quad (\ell = m + n) \quad .$$

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

The return J(k) may now be written,

$$J(k) = \|R(k)x(k) - d(k)\|^2 + r^2(k) \tag{15}$$

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

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

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

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,

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

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),

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

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

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

The matrix I_n of Eq. (19) is the n x n identity matrix.

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

A product of n - 1 Householder orthonormal transformations $S = S_{n-1} \dots S_1$, $S_i = (I_n + u_i u_i^T/\beta_i)$, $(i = 1,\dots,n-1)$, is found such that

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

where T is upper triangular.

Since F(k) is nonsingular,

$$F^{-1}(k) = T^{-1}S_{n-1} \dots S_1 \ . \tag{21}$$

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

$$R(k)F^{-1}(k)G(k) = R(k)(T^{-1}(S_{n-1}\ldots S_1G(k))) \tag{22}$$

while,

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

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

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

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

is now chosen such that,

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

In Algorithm 7 we will show that the right member of Eq. (24) can be generated in such a way that only $2.5n^2 + 3.5n+1$ memory locations are needed at each step of the calculation. Exactly $2.5n^2$ 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 Eq. (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,

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

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

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

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

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

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

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

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

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

The hypothesis that $\tilde{R}(k+1)$ be nonsingular is not critical. We can replace the indicated inverse in Eq. (25) by a pseudoinverse (Ref. 3) which always exists.

In this case the covariance matrix of Eq. (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.

APPENDIX B

Many of the algorithms presented below have appeared in a slightly different form in Ref. 5. They are repeated and expanded here for the sake of the completeness of this paper. Algorithm 3 is essentially due to Businger and Golub using a special case of Algorithm 1.

Algorithm 1

The basic Householder transformation; its construction and application.

PURPOSE

Suppose that $y = [y_1,\dots,y_m]^T$ is an arbitrary vector of length m. Given three nonnegative integers ℓ, t, and m. We wish to construct an orthonormal transformation $Q = I_m + uu^T/\beta$ such that for Qw:

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

This can be accomplished with the following

METHOD

Let $p = \ell + 1$ and $q = \ell + t + 2$.

With,

$$u = [0,\dots,0,\ u_p,\dots,u_q,\dots,u_m] \qquad \text{A1.1}$$

$$\sigma = [y_p^2 + y_q^2 + \dots + y_m^2]^{\frac{1}{2}}\cdot(-\mathrm{sgn}(y_p)) \qquad \text{A1.2}$$

$$u_p = y_p - \sigma \qquad \text{A1.3}$$

$$\beta = \sigma\cdot u_p \ \ (= -\|u\|^2/2) \qquad \text{A1.4}$$

$$u_i = y_i,\ (i = q,\dots,m) \qquad \text{A1.5}$$

the matrix,

$$Q = I_m + uu^T/\beta \qquad \text{A1.6}$$

is orthonormal and,

$$Qy = \begin{cases} y + (u^T y/\beta)u & , \quad \beta \neq 0 \\ \\ y & , \quad \beta = 0 \end{cases} \qquad \text{A1.7}$$

$$= [y_1,\ldots,y_\ell,\ \sigma,\ y_{\ell+2},\ldots,y_{q-1},\ 0,\ldots,0]^T \qquad \text{A1.8}$$

which satisfies the requirements of a) through d) above.

(In Eq. A1.2, $\operatorname{sgn}(y_p) = 1$, if $y_p \geq 0$, and equals -1 if $y_p < 0$.)

The algorithm for computing the vector u and the scalar σ is now given. The input to this algorithm will consist of three previously mentioned integers ℓ, t, and m, the m-vector y and a single free cell to hold u_p upon output.

For later reference we will designate the output of this algorithm $H1(\ell, t, m, u_p, y)$. The vector u will occupy just those positions of y which were implicitly zeroed plus the one extra location labeled u_p. The scalar σ replaces y_p in storage.

Type:	Integer	ℓ, t, m, i, p, q;
	Real	y_i, $(i = 1,\ldots,m)$;
	Double Precision	s;

Procedure: $H1(\ell, t, m, u_p, y)$

Step Number	Description
1	Set $p := \ell + 1$, $q := \ell + t + 2$, $s := y_p^2$, $i := q$.
2	If $i \leq m$ set $s := s + y_i^2$, $i := i + 1$ and go to step 2. Else
3	Set $\sigma := [-\operatorname{sgn}(y_p)]s^{\frac{1}{2}}$.

Step Number	Description
4	Set $u_p := w_p - \sigma$.
5	Set $y_p := \sigma$.

REMARK

The vector u has now been calculated; the scalar $\beta = \sigma u_p$ is later available as the indicated product and need not be explicitly saved.

The scalers $u_i = y_i$, $(i = q,\ldots,m)$, require no change of (or extra) storage.

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

The matrix product Qc is given by,

$$Qc = c + [(u^Tc)/\beta]u \qquad \text{A1.9}$$

and so the matrix Q need not be explicitly formed.

We will now present an algorithm for computing the matrix product indicated in A1.9. This procedure will be designated by the symbol $H2(\ell, t, m, u, u_p, c)$.

Type: Integer ℓ, t, m, i, p, q;
Real c_i, $(i = 1,\ldots,m)$, u_p, u_i, $(i = 1,\ldots,m)$, σ, β;
Double Precision s;

Procedure; $H2(\ell, t, m, u, u_p, c)$

Step Number	Description
1	Set $p := \ell + 1$, $q := \ell + t + 2$, $s := u_p \cdot c_p$, $i := q$.

Step Number	Description
2	If $i \le m$ set $s := s + u_i \cdot c_i$, $i := i + 1$, and go to step 2. Else
3	If $s = 0$ go to step 9. Else
4	Set $\beta := \sigma \cdot u_p$.
5	If $\beta = 0$ go to step 9. Else
6	Set $s := s/\beta$.
7	Set $c_p := c_p + u_p \cdot s$, $i := q$.
8	If $i \le m$ set $c_i := c_i + u_i \cdot s$, $i := i + 1$, and go to step 8. Else
9	The vector c has been replaced by Qc.

REMARK

Note that only those components numbered $p = \ell + 1$, and $q, \ldots, m$, $(q = \ell + t + 2)$, 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,

$$Ax = b \qquad \text{A2.1}$$

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

$$A = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_q \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \cdot \\ \cdot \\ \cdot \\ b_q \end{bmatrix} \qquad \text{A2.2}$$

where each matrix A_i is m_i x 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

$$Q_qP_q \ldots Q_2P_2Q_1[A,b] = \begin{array}{c} \begin{array}{cc} \overbrace{}^{n} & \overbrace{}^{1} \end{array} \\ \begin{bmatrix} R & , & d \\ 0 & , & r \\ 0 & , & 0 \end{bmatrix} \end{array} \begin{array}{l} \\ \}n \\ \}1 \\ \}m\text{-}n\text{-}1 \end{array} \qquad \text{A2.3}$$

Here R is upper triangular, d is an n-vector and $|r|$ is the residual vector length if R is nonsingular.

To this end set $\mu = \max(m_1,\ldots,m_q)$ and let W denote a compute working array with $\nu \geq n + 1 + \mu$ rows and $n + 1$ columns. 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.

Type:	Integer	$t, \ell, r, i, j, m, m_t, n;$
	Real	$A_i, b_i, (i = 1,\ldots,q), s;$

Procedure: Sequential Triangularization

Step Number	Description
1	Set $t := 1$ and $\ell = 0$. (*)
2	Set $r := \ell + m_t$
3	Set $W(\ell + 1:r, 1:n+1) := [A_t, b_t]$
4	Set $i := 1$
	Compute
5	$H1(i - 1, \max(0,\ell-i), r, s, W(1:r,i:i))$;
6	If $i \leq \min(r,n)$, compute $H2(i-1,\max(0,\ell-i), r,W(1:r,i:i), s, W(1:r,j,j))$, $(j = i+1,\ldots,n+1)$. Set $i := i+1$, and go to step 6. Else
7	If $t \leq q$, set $t := t + 1$, $\ell := \min[n+1,r]$, and go to step 2. Else
8	The matrix A has been reduced to upper triangular form as indicated in A2.3.

(*) If there is an a priori matrix present in the first n rows of the working array W which is zero below the main diagonal, then one may start with $\ell = n$. This a priori matrix will usually be the matrix A_1 of Eq. A2.2.

REMARKS

As we mentioned previously, one can have $m_i = 1$, $(t = 2,\ldots,q)$. Then W need occupy at most $(n+2)\cdot(n+1)$ computer words.

Following these transformations the strictly lower triangular part of W may contain remnants of the processing; these cells should be zeroed. The matrix R of A2.3 is in the upper triangular part of W; the vector d occupies $W(1{:}n,\ n+1{:}n+1)$; the residual vector length (except possibly for sign) occupies $W(n+1{:}n+1,\ n+1{:}n+1)$.

Algorithm 3

PURPOSE

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

METHOD

Suppose we wish to solve a n x n system (which may have a singular coefficient matrix) in the least squares sense.

$$Ax = b \qquad \text{A3.1}$$

Here A is an n x 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 $Q = Q_{n-1} \cdots Q_1$, $(Q_i = I_n + u_i u_i^T/\beta_i)$, $(i = 1,\ldots,n-1)$, such that

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

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 Eq. 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 $\mathcal{Q}$ 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 $A \in \mathcal{Q}$ such that rank $(\tilde{A}) < \max_{A \in \mathcal{Q}}$ [rank (A)]; it is such a matrix $\tilde{A}$ which replaces A in Eqs. A3.1 and A3.2.

We now describe the details of this forward triangularization procedure. Let W denote an n x (n+1) working array; $W(i_1:i_2,j_1:j_2)$, as before, will denote the subarray of W consisting of rows i_1 through i_2 and columns j_1 through j_2.

Type:	Integer	i, j, n, p(1:n), rank;
	Real	t, c, d(1:n), u(1:n), eps;

Procedure: Forward Triangularization

Step Number	Description
1	W := [A,b]; Scale the n columns of W. Save the reciprocals of these scale factors in d(j), (j = 1,...,n).

REMARK

The optimal choice of scaling, when one is presented with data which is uncertain, is beyond the scope of this paper. One method which is simple to describe and has worked satisfactorily for us is to set the columns of W to have euclidean length one (unless they are identically zero).

2	Set u(j) := square of the length of the j^{th} column of W following the scaling of step 1, (j = 1,...,n).
3	Set j := 1, p(i) := i, (i = 1,...,n), and rank := n.
4	If $1 < j < n$ set $u(i) := u(i) - W(j-1:j-1,i:i)^2$, (i = j,...,n). Else
5	If j = n go to step 13. Else
6	Find the smallest $i \geq j$, such that $u(i) \geq u(\ell)$, $(\ell \geq j)$.
7	If i = j go to step 9. Else

Step Number	Description
8	Exchange columns i and j of W; Set u(i) := u(j); Exchange p(j) and p(i).
9	If u(j) ≤ eps, set rank := min(rank, j - 1);

REMARK

It may be that the rank of the matrix which is to replace the matrix in W is already known and need not be calculated as in step 8. This prior calculation of rank can be done quite effectively by computing a singular value decomposition for A. See Ref. 5 for further details.

10	Compute H1(j-1, 0, n, u(j), W(1:n, j:j))
11	Compute H2(j-1, 0, n, W(1:n, j:j), u(j), W(1:n,i:i)), (i = j+1,...,n+1).
12	Set j := j:1 and go to step 4.
13	The algorithm indicated in A3.2 is completed.

In case the matrix A of Eq. A3.1 is nonsingular (or of rank n) we may compute the unique solution to this problem with the following steps:

14	Solve the triangular system Ty = d for y; The matrix T is in the upper triangular part of the array W; the vector d is in W(1:n, n+1:n+1).
15	Apply the permutation matrix P to y.
16	Form the product x = D(Py) to obtain the unique solution.

REMARK

The steps 14 through 16 described directly above can each replace the result of the previous one in storage. The details in steps 14 through 16 above are not completely described here due to the fact that they constitute straightforward and extremely well-known computing methods.

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:

$$A = Q^T T P^T D^{-1} \quad . \qquad \text{A4.1}$$

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.

Let us suppose, then, that W is again a working array as in Algorithm 3 and that T is in the first r rows of the upper triangular part of W.

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

$$TK_r \dots K_1 = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} \qquad \text{A4.2}$$

where S is r x r upper triangular and nonsingular.

The solution of minimum length or the pseudoinverse solution (Ref. 3) (with the norm $\|x\|^2 = x^T D^{-2} x$) is given by

$$y = (Q_{n-1} \dots Q_1 b) \qquad \text{A4.3}$$

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

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

$$e = K_r \ldots K_1 \begin{bmatrix} d \\ 0 \end{bmatrix} \begin{matrix} \}r \\ \}n-r \end{matrix} \qquad \text{A4.6}$$

and

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

We now describe the computation of Eqs. A4.3 through A4.7.

Type:	Integer	r, i, j, n;
	Real	t(1:n);

Procedure: Backward Triangularization

<u>Step Number</u>	<u>Description</u>
1	Use Algorithm 3 to compute y of Eq. A4.3. Place y in W(1:n, n+1:n+1).
2	Set j := r.
3	If j > 0, compute H1(j-1, r-j, n, t(j), W(j:j, 1:n)) and compute H2(j-1, r-j, n, W(j:j, 1:n), t(j), W(i:i, 1:n)), (i = j-1,...,1), (in this order). Then set j := j+1 and go to step 3. Else
4	Multiply the first r components of the vector in W(1:n, n+1: n+1) by S^{-1}. Here S is the r x r upper triangular matrix in the first r rows of the upper triangular part of W. This multiplication should be accomplished by solving Sd = c of Eq. A4.5.
5	Then compute H2(i-1, r-i, n, W(i:i, 1:n), t(i), W(1:n, n+1: n+1)), (i = 1,...,r), (in this order).

Step Number	Description
6	Set W(i:i, n+1: n+1) := 0, (i = r+1,...,n).
7	Apply the permutation matrix P to the vector y in W(1:n, n+1: n+1).
8	Form the matrix product $x = D(Py)$ in W(1:n, n+1: n+1).
9	The pseudoinverse solution of $Ax = b$ (with respect to the norm $\|x\|^2 = x^T D^{-2} x$) is now in W(1:n, n+1:n+1).

Often the pseudoinverse solution, whose calculation is defined above, must be replaced by the approximate solution obtained by setting the last $n - r$ components of the vector x to zero.

Thus

$$x = D(P \begin{bmatrix} x_1 \\ 0 \end{bmatrix}) \qquad \text{A4.8}$$

where

$$x_1 = T_{11}^{-1} y_1 \qquad \text{A4.9}$$

Here T_{11} is the r x r upper triangular matrix formed with the first r columns of the matrix T of Eq. A4.1, while y_1 is the first r components of the vector y of Eq. A4.3.

We will comment further on this in Algorithm 5.

Algorithm 5

PURPOSE

Computation of the covariance matrix.

METHOD

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

$$A = Q^T T P^T D^{-1} \qquad \text{A5.1}$$

Let

$$T = \begin{array}{cc} \begin{array}{cc} r & n-r \end{array} & \\ \begin{bmatrix} T_{11} & T_{12} \\ 0 & 0 \end{bmatrix} & \begin{array}{l} \}r \\ \}n-r \end{array} \end{array} \qquad \text{A5.2}$$

where T_{11} is r x 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 $P^T D^{-1} X$ 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 Ref. 7.)

We will now describe the algorithm for computing the right side of Eq. 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,...,r).
2	If r = 1 go to step 17. 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)·W(ℓ:ℓ, k:k), (ℓ = p + 1,...,k).
8	Set W(p:p, k:k) := -s·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)·W(i:i, j:j), (j = i,...,r).

Step Number	Description
15	Set $W(\ell:\ell, i:i) := s$.
16	If $i < r$, $i := i + 1$ and go to step 13. Else
17	If $\ell < r$, set $\ell := \ell + 1$ 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 Eq. 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 Eqs. (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 x n.

Analogous with Eq. (22) set

$$F^{-1} = T^{-1}Q_{n-1} \dots Q_1 \quad , \quad (Q_i = I_n + u_i u_i^T/\beta_1 \quad , \quad i=1,\dots,n-1) \;. \qquad A6.1$$

Then

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

and

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

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 x 2n working array Y.

Type:	Integer	i, j;
	Real	u(1:n), t(1:n);

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,...,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.

Step Number	Description
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=1,...,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) := W(i:i, j:j) and then W(i:i, j:j) := 0, (i=j+1,...,n). Next compute H2(j-1, 0, n, t(1:n), u(j), Y(i:i, 1:n)), (i = 1,...,n). 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 triangularization when mapping forwards with process noise.

METHOD

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

$$XS = X\begin{bmatrix} I_n & , & 0 & , & 0 \\ C_1 & , & C_2 & , & d \end{bmatrix} = \begin{bmatrix} A & , & B & , & e_1 \\ 0 & , & \tilde{R} & , & e_2 \end{bmatrix} \qquad \text{A7.1}$$

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

If the n x 2n matrix $[C_1 , C_2]$ occupies part of an (n+1)(2n+1) working array Y, and if an n x n working array W is available, then the right hand side of Eq. 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 Eq. A7.1.

Let $[c_1,\ldots,c_{2n}]$ denote the 2n column vectors of the n x 2n matrix $[C_1 , C_2]$. The first column of the matrix which is the right factor of the middle term of Eq. 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

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

The details of Algorithm 1 show that:

(1) After the matrix products $X_1[e_i^T, c_i^T]$, $(i=2,\ldots,n)$, $X_1[\overbrace{0}^{n}, c_i^T]$, $(i=n+1,\ldots,2n)$, and $X_1[0, d_1^T]$ are computed, only the first component or the last n components are possibly nonzero. The vectors e_i are the unit coordinate vectors.

(2) Thus only one row of the matrices A and B and one component of the vector $\underset{\sim}{e_1}$ will be calculated at each step in the construction of a matrix $\tilde{X} = X_n \ldots X_1$ such that

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

The matrix $\hat{R}$ of Eq. A7.4 is n x n but is not necessarily upper triangular.

(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 labor saving remarks.

Type:	Integer	i, j, n;
	Real	t;

Step Number	Description
1	Move the 2n + 1 components of the n + 1[st] row of S (now in the 1[st] row of Y, say) to the 2n[th] row of the working array Y.
2	Set j := 1.

Step Number	Description
3	Set Y(1:1, i:i) := 0, (i = j,...,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,...,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,...,2n), W(j:j, i:i) := Y(1:1, i:i), (i = j+1,...,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	Triangularize 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 Eq. (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 $\tilde{d}'(k+1)$ of Eq. (27) is in Y(1:1, 1:n).

To restart the basic cycle the upper triangular matrix $\tilde{R}$ together with the vector e_2 of Eq. 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

1. Dyer, P., and McReynolds, S. R., "The Extension of Square-Root Filtering to Include Process Noise", Journal of Opt.: Theory and Appl., 1969.

2. Kalman, R. E., "A new Approach to Linear Filtering and Prediction Problems", Journal Basic Eng. 82, D, 1960, 35-45.

3. Faddeeva, V. N., Computational Methods of Linear Algebra, Dover, 1959, pp. 81-85.

4. Cox, H. C., "Estimation of State Variables via Dynamic Programming", Proc. 1964, J. A. C. C., 376-381.

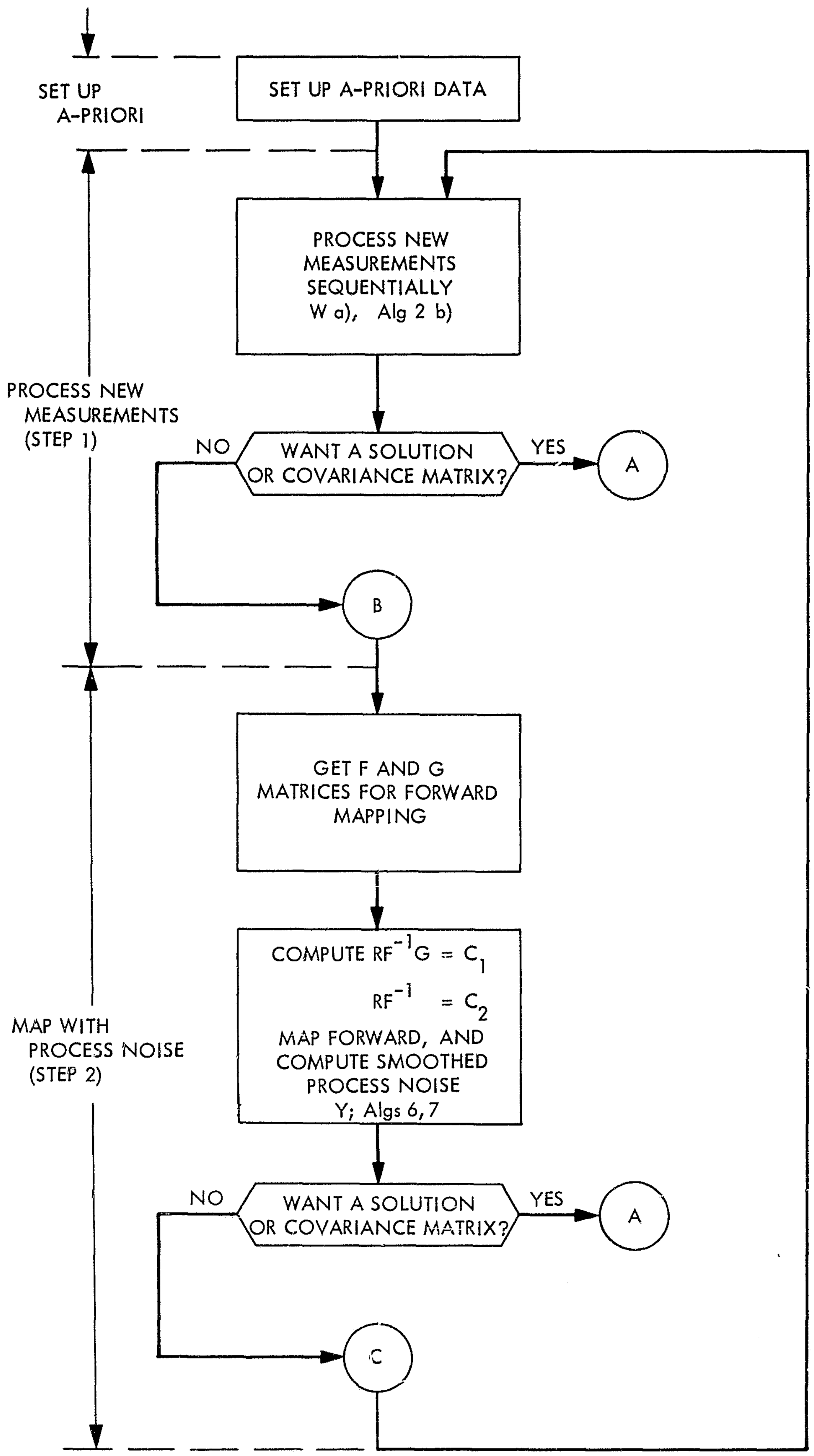
5. Lawson, C. L., and Hanson, R. J., "Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems", Jet Propulsion Laboratory, Section 314 Technical Memorandum No. 200, 12 July 1968. (To appear in Math. of Comp.)

6. Businger, P., and Golub, G., "Least Squares, Singular Values and Matrix Approximations; An ALGOL Procedure for Computing the Singular Value Decomposition", Stanford Computer Sciences Dept., Technical Report No. CS73, July 1967, (Mimo, 12 leaves).

7. Businger, P., and Golub, G., "Linear Least Squares Solution by Householder transformation", Numer. Math. 7, p. 269-275, 1965.

APPENDIX A

FLOW SEQUENCE FOR THE FILTER WITH PROCESS NOISE



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 μ = MAX NUMBER OF NEW MEASUREMENTS PROCESSED AT ONE TIME.)

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

DETERMINATION OF RANK, COMPUTATION OF SOLUTION AND COVARIANCE