NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Memorandum 33-735

Sequential Least-Squares Using Orthogonal Transformations

Gerald J. Bierman

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

August 1, 1975

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Technical Memorandum 33-735

Sequential Least-Squares Using Orthogonal Transformations

Gerald J. Bierman

JET PROPULSION LABORATORY
CALIFORNIA INSTITUTE OF TECHNOLOGY
PASADENA, CALIFORNIA

August 1, 1975

PREFACE

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

CONTENTS

ABSTRACT

This report traces square root information estimation, starting from its beginnings in least-squares parameter estimation. Special attention is devoted to discussions of the sensitivity and perturbation matrices, computed solutions and their formal statistics, consider-parameters and consider-covariances, and the effects of a priori statistics. The constant-parameter model is extended to include time-varying parameters and process noise, and the error analysis capabilities are generalized. Efficient and elegant smoothing results are obtained as easy consequences of the filter formulation.

The value of the techniques discussed here is demonstrated by the navigation results that were obtained for the Mariner Venus-Mercury (Mariner 10) multiple-planetary space probe and for the Viking Mars space mission.

SEQUENTIAL LEAST-SQUARES USING ORTHOGONAL TRANSFORMATIONS

I. INTRODUCTION

Outer space navigation analyses and orbit determination has relied, in the main, upon the method of least-squares introduced by Gauss (Refs. 8 and 12). In recent years science experiments and mission requirements have become more stringent, and space scientists have found it necessary to introduce more precise and sophisticated models, including stochastic process noise effects. Software specialists* were at first nonplussed because the new models involved time-varying parameters and process noise in a way which appeared to be at odds with the classical constant parameter estimation software then in use.

There was a reluctance to abandon the least-squares methods in favor of the Kalman filter, even though the latter technique is flexible enough to accommodate time-varying models with process noise. Reasons for this reluctance included cost, reliability and inertia. The constant parameter estimation software that was already developed, checked out, and proven is only a part of the lengthy and complex orbit determination (OD) process. In this framework the constant parameters are referenced to an epoch time, and thus introducing a current state Kalman filter would require costly modification of the entire OD process. Furthermore, OD problems generally involve processing thousands of data points and, for such situations, the Kalman filter is costly to operate. Least-squares analyses using orthogonal transformation techniques (Ref. 10) have proven to give reliable and accurate results, while the Kalman filter in process noise free situations exhibits numeric deterioration and instability. In light of these comments, it is not surprising that software specialists and managers were reluctant to abandon least-squares.

The apparent contrariety between the least-squares parameter estimation techniques based on orthogonal transformation methods and the Kalman covari-

*The remarks and opinions which follow reflect the author's experiences at the Jet Propulsion Laboratory but are believed to have a more general applicability.

ance oriented methods was cleverly resolved by modifying both the model and the least-squares orthogonal transformation method, cf Curkendall and Leondes (Ref. 5) and Dyer and McReynolds (Ref. 6). The Dyer-McReynolds filter algorithm, dubbed with the acronym SRIF (Square Root Information Filter) by Kaminski (Ref. 9), is of primary importance in our estimation developments and discussions. The work in this report documents estimation techniques used at the Jet Propulsion Laboratory which have been found to be efficient, have good numeric characteristics, and which have led to new and improved insights into the problem of orbit determination.

The intention here is to summarize and highlight the epoch-state model (see Eq. 3-4) SRIF formulations for filtering, smoothing, and error analysis. Specializing the results to the epoch-state model greatly simplifies the somewhat intimidating algorithms of Refs. 2, 3, 4, and 9 that were developed for more general linear models. Our techniques and algorithms are applicable to a rather broad class of problems even though our development is oriented toward, and was motivated by, orbit determination applications. The following paragraphs outline the scope of material that is to be discussed.

In Section II we discuss Golub's sequential solution to the least-squares parameter estimation problem through the use of Householder orthogonal transformations* (Ref. 7). This method is acknowledged to be numerically stable and is considerably more accurate than the classical normal equation technique introduced by Gauss (Ref. 10). Various jargon are defined here: the "data equation" corresponding to an estimate-estimate error covariance pair; "computed" solutions and covariances corresponding to restricted dimension models; "sensitivities" and "perturbations" corresponding to neglected parameters, and "consider" covariances which reflect the true error covariance of the computed estimate.

In Section III the "epoch state" model is defined along with a technique for including time propagation and white process noise. Certain computational efficiencies and algorithmic simplifications come about when the process noise entering the time propagation is exponentially correlated.

Smoothing, discussed in Section IV, is arranged so that it blends with the material of Sections II and III. Smoothed analogues are developed for the

*Householder orthogonal transformations are discussed in Appendix A.

"computed" estimate, the "computed" estimate error covariance, and the "sensitivity" matrix. These results differ from their filter counterparts, which arise from auxiliary information array computations, in that they are computed recursively. Commonalities between the filter and smoother algorithm formulations allow for a generalization of the "consider" covariance and of estimate-covariance pairs corresponding to various order estimation models.

Section V is devoted to error analyses that are related to the SRIF. The data equation representations lead straight-away to augmented information array algorithms. The error equation methodology parallels the covariance error analysis methods that are derived in Ref. 11; but the error analysis algorithms, like the SRIF, differ from their conventional covariance algorithm counterparts. The error analyses are not completely general but do cover the principal error sources occurring in orbit determination, including the effects of incorrect a priori state and process noise covariance statistics, unestimated parameters, and the effects of incorrect colored noise time constants.

II. SEQUENTIAL DATA PROCESSING USING ORTHOGONAL TRANSFORMATIONS

Accurate and stable sequential least-squares data processing methods are of major importance in orbit determination because OD problems generally have poor observability characteristics (are ill-conditioned) and involve large amounts of data. The normal equation methods, classically used in such problems, have been supplanted by the orthogonal transformation techniques that were introduced by Golub (Ref. 7). To see how orthogonal transformations enter into the least-squares problem, consider the overdetermined* system:

$$z = Ax + \nu \tag{2-1}$$

where x is an n-vector, z is an m-vector (with $m \geq n$) and ν is assumed to be scaled so that $\nu \epsilon N(0, I_m)$. The least-squares problem is formulated and solved in the following sequence of equations:

$$J_{\ell s}(x) \equiv \sum_{j=1}^{m} \nu_j^2 = ||\nu||^2 \tag{2-2}$$

$$= ||Ax - z||^2 \tag{2-3}$$

$$= ||Q_A (Ax - z)||^2 \tag{2-4}$$

$$Q_A \quad \text{such that} \quad Q_A^T Q_A = I_m$$

$$= ||Q_A Ax - Q_A z||^2 \tag{2-5}$$

$$Q_A A = \begin{bmatrix} \hat{R} \\ -- \\ 0 \end{bmatrix} \begin{matrix} \}n \\ \\ \}m-n \end{matrix}, \quad Q_A z = \begin{bmatrix} \hat{z} \\ -- \\ e \end{bmatrix} \begin{matrix} \}n \\ \\ \}m-n \end{matrix}$$

*Only overdetermined problems are discussed here because when a priori information is used the estimation problem is of this form.

$$J_{ls}(x) = \left\| \begin{pmatrix} \hat{R}x \\ -- \\ 0 \end{pmatrix} - \begin{pmatrix} \hat{z} \\ -- \\ e \end{pmatrix} \right\|^2 \tag{2-6}$$

$$= \|\hat{R}x - \hat{z}\|^2 + \|e\|^2 \tag{2-7}$$

Golub pointed out that the orthogonal transformation Q_A could be simply computed using Householder's reflections, that in this case R will be a triangular matrix, and that this reduction of the array $[A, z]$ could be accomplished without explicitly computing Q_A. Appendix A reviews the key ideas and formulae involved with Householder matrix reductions.*

From Eqs. 2-2 and 2-7 it is evident that J_{ls} is minimized for $\hat{x}$ chosen such that

$$\hat{R}\hat{x} = \hat{z} \tag{2-8}$$

and that $\|e\|^2$ is the minimum sum of squares residual error. Statistical significance for the solution $\hat{x}$ is an easy consequence of Eq. 2-1, i.e.,

$$Q_A z = Q_A A x + Q_A \nu$$

$$\begin{bmatrix} \hat{z} \\ ---- \\ e \end{bmatrix} = \begin{bmatrix} \hat{R} \\ ---- \\ 0 \end{bmatrix} x + \begin{bmatrix} \hat{\nu} \\ ---- \\ \nu_e \end{bmatrix}$$

and thus,

$$\hat{z} = \hat{R}x + \hat{\nu} \tag{2-9}$$

where $\hat{\nu} \in N(0, I_n)$, because $Q_A \nu \in N(0, I_m)$. Combining Eqs. 2-8 and 2-9 gives

$$x - \hat{x} = -\hat{R}^{-1}\hat{\nu} \quad ,$$

*For more details of this subject the reader is referred to the book by Lawson and Hanson (Ref. 10) which comprehensively treats the various numeric and computational aspects of orthogonally computed least-squares solutions.

so that

$$P_{\hat{x}} \equiv E\left[(x - \hat{x})(x - \hat{x})^T\right] = \hat{R}^{-1}\hat{R}^{-T} \qquad (2\text{-}10)^*$$

Equation 2-10 shows that $\hat{R}^{-1}$ is a square root of the error covariance matrix $P_{\hat{x}}$.

An equation of the form of Eq. 2-9, with $\hat{R}$ nonsingular, is called a data equation. From Eqs. 2-8 and 2-10 it is clear that there is an equivalence between the information array $[\hat{R}, \hat{z}]$ and the covariance-estimate pair $(P_{\hat{x}}, \hat{x})$. Information arrays are not unique, but if $[R_1, z_1]$ and $[R_2, z_2]$ correspond to the same covariance-estimate pair, then for some orthogonal matrix Q,

$$[R_1, z_1] = Q[R_2, z_2] \quad .$$

(This relation is a direct consequence of the data equation representation corresponding to the information array.) An obvious but important consequence of a data equation representation for the a priori covariance and estimate is that the least-squares orthogonalization is recursive. This result is written as theorem 1.

Theorem 1 (Recursive Least-Squares Measurement Updating)

The data $z_j = A_j x + \nu_j$, with $E(\nu_j) = 0$, $E(\nu_i, \nu_j^T) = I_{m_j}\delta_{ij}$ can be recursively least-squares fit as follows. Start with the a priori information array $[\hat{R}_0, \hat{z}_0]$, and for $j = 1, \ldots, N$, recursively compute:

$$Q_{j-1}\begin{bmatrix} \hat{R}_{j-1} & \hat{z}_{j-1} \\ A_j & z_j \end{bmatrix} = \begin{bmatrix} \hat{R}_j & \hat{z}_j \\ 0 & e_j \end{bmatrix} \qquad (2\text{-}11)$$

$$\Sigma_j = \Sigma_{j-1} + \|e_j\|^2 \quad , \qquad \Sigma_0 = 0 \qquad (2\text{-}12)$$

where Q_{j-1}, orthogonal, is implicitly defined by Eq. 2-11.

*$\left(\hat{R}^{-1}\right)^T = \left(\hat{R}^T\right)^{-1} \equiv \hat{R}^{-T}$

Then, for each j, $\hat{x}_j$ given by Eq. 2-13 is the least-squares estimate of x based on the data $\{z_i | i = 0, \cdots, j\}$, and $\hat{P}_j$ is its error covariance:

$$\hat{x}_j = \hat{R}_j^{-1} \hat{z}_j \quad , \qquad \hat{P}_j = \hat{R}_j^{-1} \hat{R}_j^{-T} \quad . \tag{2-13}$$

Moreover,

$$\Sigma_j = \left\| \hat{R}_0 \hat{x}_j - \hat{z}_0 \right\|^2 + \sum_{i=1}^{j} \left\| A_i \hat{x}_j - z_i \right\|^2 \tag{2-14}$$

NOTES:

(1) $\hat{R}_j$ are by construction triangular matrices, generally arranged so as to be upper triangular. The triangular structure reduces computations in Eq. 2-11 and also facilitates the matrix inversion appearing in Eq. 2-13.

(2) The recursion (Eq. 2-11) is similar in certain respects to the normal equation results,

$$\left[\Lambda_j, d_j \right] = \left[\Lambda_{j-1}, d_{j-1} \right] + \left[A_j^T A_j, A_j^T z_j \right] \tag{2-15}$$

In this case the estimate and covariance are given by

$$\hat{x}_j = \Lambda_j^{-1} d_j \quad , \qquad \hat{P}_j = \Lambda_j^{-1} \tag{2-16}$$

Equation 2-15 is susceptible to roundoff errors; and not infrequently one finds that the computed Λ is singular, cf Refs. 7 and 10.

(3) Estimates and covariances (Eq. 2-13) need not be computed for each j and this allows for certain economies of computation.

(4) Equation 2-12 gives the accumulated residual error corresponding to the latest estimate; but this computation does not explicitly involve the estimate.

(5) The relations in Eqs. 2-12 and 2-14 are important in residual analyses problems involving large amounts of data. In such problems it is inconvenient and costly to reread the previous data $\{A_i, z_i | i = 1, \cdots, j\}$ from storage and compute Eq. 2-14.

The data equation, as shown in Theorem 1, is a handy vehicle with which to express the state of the system. To further explore this representation, suppose that x is replaced by the partitioned vector $\begin{bmatrix} x \\ y \end{bmatrix}$ and that the data Eq. 2-9 is partitioned consistently. Assuming that R is upper triangular gives:

$$\begin{bmatrix} R_x & R_{xy} \\ 0 & R_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} z_x \\ z_y \end{bmatrix} - \begin{bmatrix} \nu_x \\ \nu_y \end{bmatrix} \tag{2-17}$$

x and y are now assumed to be vectors of dimension n_x and n_y respectively, with $n = n_x + n_y$. This partitioned data equation has an interesting and useful interpretation.

The bottom equation, $R_y y = z_y - \nu_y$ is a data equation for y and as such it implicitly contains the estimate and covariance of y. This point is of importance when the parameter estimation problem is of large dimension; but one is only interested in explicitly estimating a small subset of parameters.

Turning attention to the upper of Eqs. 2-17, we point out that a consequence of the triangular construction is that R_x and ν_x are independent of y. This observation is of major importance because it enables one to analyze the effects of various y models in a straightforward and insightful manner. With these comments in mind, Eq. 2-17 is expanded to obtain:

$$\left.\begin{aligned} x &= R_x^{-1} z_x + \left(-R_x^{-1} R_{xy}\right) y - R_x^{-1} \nu_x \\ &\equiv x_c + (SEN)\, y - R_x^{-1} \nu_x \end{aligned}\right\} \tag{2-18}$$

This innocent looking equation is actually pregnant with implications, which we now proceed to describe.

$$x_c = R_x^{-1} z_x \quad , \quad SEN = -R_x^{-1} R_{xy} \quad , \quad P_c = R_x^{-1} R_x^{-T} \tag{2-19}$$

where x_c is called the computed estimate and corresponds to a model with no y parameters; SEN is called the Sensitivity; and P_c called the computed covariance will equal $E[(x - x_c)(x - x_c)^T]$ if the model contains no y parameters.

$$\left.\begin{aligned} P_{x_c} &= E\left[(x - x_c)(x - x_c)^T\right] \\ &= P_c + Sen\, \tilde{P}_y (SEN)^T \end{aligned}\right\} \tag{2-20}$$

where $\tilde{P}_y$ is the a priori covariance of the y parameter vector. P_{x_c} is the actual or true covariance of the estimate x_c; it reflects the estimate error due to using a filter model that ignores the y parameters.

Not infrequently one includes y parameters in the model, not because they are of special interest, but because excluding them may lead to poor estimates of the x parameters. Equation 2-20 separates out that portion of the error covariance that is due to the omitted y parameters. In this context the components of y are called consider parameters and P_{x_c} is called a consider covariance.

In the special but important case where the y a priori covariance is diagonal, one can get further useful information about the effects of the individual components of y. In this case, the columns of $\Gamma \equiv SEN\ Diag\ (\tilde{\sigma}_y(1), \cdots, \tilde{\sigma}_y(n_y))$ are, so to speak, error perturbations of the estimate error; and perusal of η is helpful in determining which y parameters contribute most or least to uncertainties in particular components of the estimate. This information is a valuable aid in identifying major error sources.

The importance and utility of the SEN matrix does not end here; it appears in the optimal least-squares estimate and covariance results. From Eq. 2-18 it follows that

$$\hat{x} = x_c + SEN\, \hat{y} \tag{2-21}$$

where $\hat{y} = R_y^{-1} z_y$ and

$$\left.\begin{aligned} \hat{P}_x &= E\left[(x - \hat{x})(x - \hat{x})^T\right] \\ &= P_{x_c} + SEN\, \hat{P}_y (SEN)^T \end{aligned}\right\} \tag{2-22}$$

where $\hat{P}_y = R_y^{-1} R_y^{-T}$. Furthermore, from Eqs. 2-18 and 2-21 it follows that

$$SEN = \frac{\partial \hat{x}}{\partial \hat{y}} = \frac{\partial (x - \hat{x})}{\partial (y - \hat{y})} \tag{2-23}$$

and thus SEN is a sensitivity matrix which measures the sensitivity of the estimate to the y parameters.

These formulae involving the sensitivity matrix point up its key role in parameter estimation analysis. Although these estimation formulae are easily and simply implemented, their flexibility and utility is enhanced by noting the following:

(1) Systematically increasing the x parameter set can be done with but a modicum of effort. The necessary equations follow from the accompanying matrix identity:

$$\begin{bmatrix} R_x & R_{xy} \\ 0 & R_y \end{bmatrix}^{-1} = \begin{bmatrix} R_x^{-1} & \left(-R_x^{-1} R_{xy}\right) R_y^{-1} \\ 0 & R_y^{-1} \end{bmatrix}$$

(2) SEN is independent of the y a priori and this simplifies certain analyses.

(3) The triangular structure of R_y makes it a simple matter to delete the lowermost y's and examine models with a smaller number of parameters.

(4) One need not commit to a prespecified arrangement of the y parameters; i.e., it is possible to omit any subset of parameters from the

model. In the general case, one should permute the columns of R_v and SEN corresponding to the desired y-parameter arrangement and then use an orthogonal transformation to retriangularize R_y.

To illustrate the kind of analysis that can be performed using the ideas just described, consider the following hypothetical but typical problem. Formulation of the problem involves, say, 150 parameters; but we are interested in estimates for perhaps 10 of these, and they are designated x. Most of the remaining parameters came about because of our efforts to precisely model the problem. Based on our knowledge of this problem, we arrange the parameter name list so that those parameters believed to be most significant are at the top of the list. There are 3000 data points and these are orthogonally transformed to an upper triangular form, cf Eq. 2-11. The sensitivity matrix is formed, cf Eq. 2-18, and is 10 by 140. A perturbation matrix, Γ, is computed using nominal y-parameter a priori standard deviations, and the columns are inspected. This suggests a rearrangement of certain of the y's, i. e., moving to the end those y's that appear to have negligible significance. The columns of SEN and R_y are rearranged accordingly, and R_y is orthogonally triangularized and the y a priori is included. It is possible now to exhibit estimates and estimate error covariances based on, say, 50, 60 and 70 y-parameter models. If the higher order y's are truly less significant, the estimate and covariance should be relatively unaffected by increasing the model order. By redefining the x and y namelists, or by partitioning and expanding the appropriate equations, one could examine the effects of estimating the first n_{y_1} parameters, considering the effects of the next n_{y_2} parameters and ignoring the rest. For a problem with this structure, we would be perusing P_{x_c} where x_c^T and y^T represent $(x^T, y_1, \cdots, y_{n_{y_1}})$ and $(y_{n_{y_1}+1}, \cdots, y_{n_{y_1}+n_{y_2}})$ respectively.

The data analysis options described in this section are standard components of JPL's Orbit Determination Program.

III. MAPPING AND THE INCLUSION OF PROCESS NOISE

The discussions of the previous section were directed at constant parameter estimation. Fortunately, and not coincidentally, most of the key results and notions of that section extend to the time varying problem,

$$X_{j+1} = \Phi_j X_j + G_j w_j \tag{3-1}$$

where $\{\Phi_j\}$ are nonsingular and $\{w_j\}$ is a white noise sequence. The nature of our orbit determination problems motivates us to partition X, and we write

$$\begin{bmatrix} p \\ x \\ y \end{bmatrix}_{j+1} = \begin{bmatrix} M & 0 & 0 \\ \phi_{xp} & \phi_x & \phi_{xy} \\ 0 & 0 & I \end{bmatrix}_j \begin{bmatrix} p \\ x \\ y \end{bmatrix}_j + \begin{bmatrix} w_j \\ 0 \\ 0 \end{bmatrix} \tag{3-2}$$

where the ϕ elements are transition matrix elements $\phi_{xp}(j+1, J)$, etc. The vector p is colored process noise and y are constant parameters. In this formulation M, assumed to be diagonal, may be singular; and thus, we allow for white noise inputs to x. The model of Eq. 3-2, while somewhat specialized, is quite adequate for our applications. Spacecraft related p parameters are generally random accelerations; viz solar pressure, altitude control jet gas leaks, or satellite drag effects. OD related p parameters may reflect ephemeris variations or Deep Space Network Station location position uncertainties.

Previous orbit determination analyses which did not involve process noise were cast as parameter estimation problems by referencing the time-varying parameters to an epoch time. We retain this practice because such a formulation is compatible with existing OD software (at JPL) and because our estimation computations are somewhat simplified in this framework. Thus, we define the <u>epoch state</u> variable $\bar{x}_j$, so that

$$x_j = \phi_x(j, 0)\,\bar{x}_j + \phi_{xy}(j, 0)y \tag{3-3}$$

$\bar{x}_j$ are called <u>epoch state</u> variables because when $\phi_{xp}(j+1, j)p_j \equiv 0$, $\bar{x}_j$ reduces to the epoch state x_0. When $\bar{x}_j$ is substituted into Eq. 3-2, one obtains the simplified propagation representation (Eq. 3-4),

$$\begin{bmatrix} p \\ \bar{x} \\ y \end{bmatrix}_{j+1} = \begin{bmatrix} M & 0 & 0 \\ V_p & I & 0 \\ 0 & 0 & I \end{bmatrix}_j \begin{bmatrix} p \\ \bar{x} \\ y \end{bmatrix}_j + \begin{bmatrix} w_j \\ 0 \\ 0 \end{bmatrix} \tag{3-4}$$

where $V_p(j) = \phi_x^{-1}(j+1, 0)\, \phi_{xp}(j+1, j)$.

Let us turn now to the business of constructing the SRIF mapping algorithm corresponding to the dynamic model (Eq. 3-4). The mapping process consists simply of constructing a data equation for X_{j+1}, given an a priori data equation for X_j. Thus, suppose we are given an a priori data equation in the following form:

$$\begin{bmatrix} \hat{R}_p & \hat{R}_{px} & \hat{R}_{py} \\ \hat{R}_{xp} & \hat{R}_x & \hat{R}_{xy} \\ 0 & 0 & \hat{R}_y \end{bmatrix} \begin{bmatrix} p_j \\ \bar{x}_j \\ y \end{bmatrix} = \begin{bmatrix} \hat{z}_p \\ \hat{z}_x \\ \hat{z}_y \end{bmatrix} - \begin{bmatrix} \hat{v}_p \\ \hat{v}_x \\ \hat{v}_y \end{bmatrix} \tag{3-5}^*$$

NOTES:

(1) In the interest of notational economy we omit the j subscript on R and z, and rely on the symbol "$\wedge$" to indicate the values of R and z at the start of the mapping.

(2) A pragmatically important advantage accruing to a formulation with the p variables uppermost is that when $R_{xp} = 0$, the formulae of Section II are valid and need no modifications.

Using Eq. 3-4 it is easy to replace $\bar{x}_j$ by $\bar{x}_{j+1}$, i.e.,

$$\begin{bmatrix} \hat{R}_p - \hat{R}_{px}V_p & \hat{R}_{px} & \hat{R}_{py} \\ \hat{R}_{xp} - \hat{R}_x V_p & \hat{R}_x & \hat{R}_{xy} \\ 0 & 0 & \hat{R}_y \end{bmatrix} \begin{bmatrix} p_j \\ \bar{x}_{j+1} \\ y \end{bmatrix} = \begin{bmatrix} \hat{z}_p \\ \hat{z}_x \\ \hat{z}_y \end{bmatrix} - \begin{bmatrix} \hat{v}_p \\ \hat{v}_x \\ \hat{v}_y \end{bmatrix} \tag{3-6}$$

*Generally the $\hat{R}$ terms are the result of data processing, and in this case $\hat{R}_{xp} = 0$.

To complete our construction of the data equation at (j+1), we have to replace p_j by p_{j+1}, and this is done in the following way. Recall from Eq. 3-4 that the colored process noise terms satisfy

$$p_{j+1} = M p_j + w_j, \tag{3-7}$$

and that the statistics of w_j can be written in data equation form

$$R_w w_j = z_w - \nu_w \tag{3-8}$$

($z_w = 0$ when $E(w_j) = 0.$)

Equations 3-6 through 3-8 are combined by considering an augmented state vector, one that contains p_j and p_{j+1}:

$$\begin{bmatrix} -R_w M & R_w & 0 & 0 \\ \hat{R}_p - \hat{R}_{px} V_p & 0 & \hat{R}_{px} & \hat{R}_{py} \\ \hat{R}_{xp} - \hat{R}_x V_p & 0 & \hat{R}_x & \hat{R}_{xy} \\ 0 & 0 & 0 & \hat{R}_y \end{bmatrix} \begin{bmatrix} p_j \\ p_{j+1} \\ \bar{x}_{j+1} \\ y \end{bmatrix} = \begin{bmatrix} \hat{z}_w \\ \hat{z}_p \\ \hat{z}_x \\ \hat{z}_y \end{bmatrix} - \begin{bmatrix} \nu_w \\ \hat{\nu}_p \\ \hat{\nu}_x \\ \nu_y \end{bmatrix} \tag{3-9}$$

To obtain the desired data equations from this result one has only to orthogonally eliminate p_j from the bottom equation; and, in information array form, the result is:

$$
\begin{array}{c}
\begin{array}{ccccc} p_j & p_{j+1} & \bar{x}_{j+1} & y & z \end{array} \\
\left[\begin{array}{c|c|c|c|c}
R_p^* & R_{pp}^* & R_{px}^* & R_{py}^* & z_p^* \\
0 & \tilde{S}_p & \tilde{S}_x & \tilde{S}_y & \tilde{S}_z
\end{array}\right]
\begin{array}{l} \}n_p \\ \}n_p + n_x \end{array} \\
= \tilde{Q}\left[\begin{array}{c|c|c|c|c}
-R_w M & R_w & 0 & 0 & z_w \\
\hat{R}_p - \hat{R}_{px} V_p & 0 & \hat{R}_{px} & \hat{R}_{py} & \hat{z}_p \\
\hat{R}_{xp} - \hat{R}_x V_p & 0 & \hat{R}_x & \hat{R}_{xy} & \hat{z}_x
\end{array}\right]
\begin{array}{l} \}n_p \\ \}n_p \\ \}n_x \end{array}
\end{array}
\qquad (3\text{-}10)
$$

where $\tilde{Q}$, orthogonal, is chosen to partially triangularize the information array and the $\tilde{S}$ notation represents the time updated information array

$$
\begin{array}{l}
\begin{array}{ccccc} \overbrace{}^{n_p} & & \overbrace{}^{n_x} & \overbrace{}^{n_y} & \overbrace{}^{1} \end{array} \\
n_p + n_x \left\{ \left[\begin{array}{c|c|c|c} \tilde{S}_p & \tilde{S}_x & \tilde{S}_y & \tilde{S}_z \end{array}\right] \right. = \left[\begin{array}{cccc} \tilde{R}_p & \tilde{R}_{px} & \tilde{R}_{py} & \tilde{z}_p \\ \tilde{R}_{xp} & \tilde{R}_x & \tilde{R}_{xy} & \tilde{z}_x \end{array}\right]
\end{array}
\qquad (3\text{-}11)
$$

NOTES:

(1) The top row of Eq. 3-10 corresponds to a data equation

$$R_p^* p_j + R_{pp}^* p_{j+1} + R_{px}^* x_{j+1} + R_{py}^* y = z_p^* - \nu_p^*, \qquad (3\text{-}12)$$

and this equation is of major importance in smoothing, cf Section IV.

(2) The y information array $\left[\hat{R}_y \mid \hat{z}_y\right]$ is unchanged by the mapping. (This is as it should be since the y parameters are time-invariant.)

(3) Appendix B contains an efficient and compact FORTRAN mechanization of the mapping (Eq. 3-10). This mechanization makes maximal use of the special structure of our problem formulation.

(4) Dyer and McReynolds (Ref. 6) describe how to time update an information array corresponding to the general dynamic model (Eq. 3-1).

IV. SMOOTHING

Most orbit determination involves nonreal time or postflight data reduction and, as a result, there is an interest in smoothing, i.e., trajectory estimation based on the entire data history. In this section we show that smoothed estimates can be computed with but a modicum of computational cost. We also present smoothed analogues of the computed estimate, computed covariance, the sensitivity matrix, and the consider covariance. These generalizations are important because the formulae are consistent with the constant parameter results of Section II; and thus the software developed for that problem can be applied to this problem as well. Furthermore, OD engineers familiar with constant parameter estimation can better relate to the smoothing problem when it is couched in a familiar framework.

To see how the smoothed state estimates come about, suppose that smoothed estimates of p_{j+1}, $\bar{x}_{j+1}$ and y have been computed and that these are labeled with a "*". Smoothed estimates for p_j and $\bar{x}_j$ are a direct result of the mapping, Eq. 3-4, and the key portion of the data equation, Eq. 3-12. Define, cf Eq. 3-12,

$$\left[L_p \;\vdots\; L_x \;\vdots\; L_y \;\vdots\; L_z\right] = R_p^{*-1}\left[R_{pp}^{*} \;\vdots\; R_{px}^{*} \;\vdots\; R_{py}^{*} \;\vdots\; z_p^{*}\right] \qquad (4\text{-}1)$$

and note that the smoothed estimate satisfies the data equation with the ν^* term set to zero, i.e.,

$$p_j^* = L_z - L_p\, p_{j+1}^* - L_x\, x_{j+1}^* - L_y\, y^* \qquad (4\text{-}2)$$

$$x_j^* = x_{j+1}^* - V_p\, p_j^* \qquad \text{(cf Eq. 3-4)} \qquad (4\text{-}3)$$

Equations 4-2 and 4-3 establish the desired smoothing recursions.

NOTES:

(1) The reader is reminded that the L terms and V_p are j dependent.

(2) The smoothing recursion, which is backward in time, is initialized with the filter estimates of the terminal time point. Smoothed estimate error covariance recursions could be obtained in the usual fashion (Ref. 11) by differencing Eqs. 4-2 and 4-3 with Eqs. 3-4 and 3-12, squaring (multiplying by the transpose), and applying the expectation operator.

An alternative and better formulation of the smoothing solution results if the algorithm is arranged in a form that is analogous to the SRIF solution for the filtering problem with a computed estimate and error covariance and a sensitivity matrix. With this in mind, we decompose the p-x vector as

$$\begin{bmatrix} p \\ x \end{bmatrix} = \begin{bmatrix} p_c \\ x_c \end{bmatrix} + \begin{bmatrix} SEN_p^* \\ SEN_x^* \end{bmatrix} y \tag{4-4}$$

where the "c" subscripted variables are independent of y. Substituting this decomposition into Eqs. 3-4 and 3-12 leads to

$$\left.\begin{aligned} p_c(j) &= L_z - L_p\, p_c(j+1) - L_x\, x_c(j+1) - R_p^{*-1}\, \nu_p^* \\ x_c(j) &= x_c(j+1) - V_p\, p_c(j) \end{aligned}\right\} \tag{4-5}$$

$$\left.\begin{aligned} SEN_p^*(j) &= -L_p SEN_p^*(j+1) - L_x SEN_x^*(j+1) - L_y \\ SEN_x^*(j) &= SEN_x^*(j+1) - V_p\, SEN_p^*(j+1) \end{aligned}\right\} \tag{4-6}$$

The smoothed state recursion for p_c and x_c follow from Eq. 4-5 by setting ν_p^* equal to zero:

$$\left.\begin{aligned} p_c^*(j) &= L_z - L_p\, p_c^*(j+1) - L_x\, x_c^*(j+1) \\ x_c^*(j) &= x_c^*(j+1) - V_p\, p_c^*(j) \end{aligned}\right\} \tag{4-7}$$

To obtain a covariance recursion for the p_c and x_c estimate errors, we difference Eqs. 4-5 and 4-7:

$$\begin{bmatrix} \Delta p_c \\ \Delta x_c \end{bmatrix}_j = \begin{bmatrix} -L_p & -L_x \\ V_p L_p & I+V_p L_x \end{bmatrix} \begin{bmatrix} \Delta p_c \\ \Delta x_c \end{bmatrix}_{j+1} - \begin{bmatrix} I \\ -V_p \end{bmatrix} \begin{bmatrix} R_p^{*-1}\, \nu_p^* \end{bmatrix} \tag{4-8}$$

where $\Delta p_c = p_c - p_c^*$ and $\Delta x_c = x_c - x_c^*$. From this, the covariance recursion is seen to be:

$$\begin{bmatrix} P_p^{c*} & P_{px}^{c*} \\ \left(P_{px}^{c*}\right)^T & P_x^{c*} \end{bmatrix}_j = \begin{bmatrix} -L_p & -L_x \\ V_p L_p & I+V_p L_x \end{bmatrix} \begin{bmatrix} P_p^{c*} & P_{px}^{c*} \\ \left(P_{px}^{c*}\right)^T & P_x^{c*} \end{bmatrix}_{j+1} \begin{bmatrix} -L_p & -L_x \\ V_p L_p & I+V_p L_x \end{bmatrix}^T$$

$$+ \begin{bmatrix} \mathfrak{P}_p^* & -\mathfrak{P}_p^* V_p^T \\ V_p \mathfrak{P}_p^* & V_p \mathfrak{P}_p^* V_p^T \end{bmatrix} \; ; \quad \mathfrak{P}_p = R_p^{-1} R_p^{-T} \tag{4-9}$$

From the way that x_c^*, P_x^{c*} and SEN_x^* have been defined, it follows that the formulae of Section II are applicable, i.e.,

$$P_{x_c}^* = P_x^{c*} + SEN_x^*\, \tilde{P}_y\, (SEN_x^*)^T \quad \text{(cf Eq. 2-20)} \tag{4-10}$$

$$x^* = x_c^* + SEN_x^*\, y^* \quad \text{(cf Eq. 2-21)} \tag{4-11}$$

$$P_{x^*} = P_x^{c*} + SEN_x^*\, \hat{P}_y\, (SEN_x^*)^T \quad \text{(cf Eq. 2-22)} \tag{4-12}$$

$$SEN_x^* = \frac{\partial(x - x^*)}{\partial(y - y^*)} \qquad \text{(cf Eq. 2-23)} \qquad (4\text{-}13)$$

NOTE: When p is a scalar, the filtering and smoothing computations reduce considerably. This observation is exploited in the FORTRAN mechanizations given in Appendix B, where the algorithms are arranged so that the effects of a vector p are included one component at a time.

V. ANALYZING THE EFFECTS OF THE A PRIORI STATISTICS

Engineers and scientists who choose to apply linear estimation technology are often perplexed about what values to use for the a priori statistics that appear in the estimation algorithms. One generally has only an order of magnitude knowledge of the a priori uncertainties and, because of this, there may be doubts about the reliability of the results. Examples of the kinds of questions that should be answered in order to intelligently apply estimation software to meaningful problems are:

(1) How would doubling or halving the a priori state vector variances effect the estimates and estimate error covariances?

(2) Suppose that the measurement errors were actually larger (or smaller) than assumed. How would this affect the estimates and covariances?

(3) Process noise and correlation times are mathematical fictions designed to compensate for various modeling errors and physical phenomena. How sensitive is the estimation algorithm to these terms?

Answers to questions of this nature aid in filter design and create an atmosphere of confidence. The effects of bias parameters are best analyzed in terms of the sensitivity matrix and the consider covariance, and these have been discussed in Section II. In this section, we concentrate on the process noise effects and, to keep the notation from becoming cluttered, we omit the constant y parameters from this discussion (note that their omission is only for expositional simplicity; the mathematics is not at all complicated by their inclusion).

To analyze the effects of incorrect a priori statistics in the data equation estimation formulation (Eq. 2-9) one need only maintain a knowledge of the covariance of the ν error term. To better understand what is meant by this statement, and to initialize the filter error algorithm, suppose that $X = \begin{bmatrix} p \\ x \end{bmatrix}_1$, is assumed to have an a priori estimate error covariance, $P_X = R_X^{-1} R_X^{-T}$, but that the actual error covariance is $P_X^a = \left(R_X^a\right)^{-1} \left(R_X^a\right)^{-T}$. If we now write

$$z_X = R_X X + \nu_X, \qquad z_X = R_X \hat{X} \tag{5-1}$$

then ν_X will not have an identity covariance. Instead,

$$E\left(\nu_X \nu_X^T\right) = \Lambda_X \Lambda_X^T, \qquad \Lambda_X = R_X \left(R_X^a\right)^{-1} \tag{5-2}$$

because the true data equation for the estimate $\hat{X}$ is

$$R_X^a \hat{X} = R_X^a X + \nu_X^a, \qquad E\left[\nu_X^a \left(\nu_X^a\right)^T\right] = I \tag{5-3}$$

and comparing Eq. 5-1 with Eq. 5-3 gives $\nu_X = \Lambda_X \nu_X^a$. Thus, Λ_X is a square-root of $E\left(\nu_X \nu_X^T\right)$. Note that one can recover the actual covariance P_X^a from a knowledge of Λ_X and R_X, i.e.,

$$P_X^a = \left(R_X^{-1}\Lambda_X\right)\left(R_X^{-1}\Lambda_X\right)^T \tag{5-4}$$

The error analysis technique for the filtering algorithm is to sequentially update the augmented array

$$\left[R_X \;\vdots\; z_{\xi} \;\vdots\; \Lambda_X\right] \tag{5-5}$$

To see how Λ_X is updated when data is addended suppose that

$$z = A X + \nu \tag{5-6}$$

is to be included with the a priori:

$$E(\nu \nu^T) = \begin{cases} R_\nu^{-1} R_\nu^{-T} & : \quad \text{assumed} \\ \left(R_\nu^a\right)^{-1} \left(R_\nu^a\right)^{-T} & : \quad \text{actual} \end{cases} \tag{5-7}$$

then the SRIF algorithm would compute $\left[\hat{R}_X \mid \hat{z}_X\right]$ from

$$\left.\begin{aligned} \hat{Q}\begin{bmatrix} R_X \\ R_\nu A \end{bmatrix} X &= \hat{Q}\begin{bmatrix} z_X \\ R_\nu z \end{bmatrix} - \hat{Q}\begin{bmatrix} \nu_X \\ R_\nu \nu \end{bmatrix} \\ \begin{bmatrix} \hat{R}_X \\ \hline 0 \end{bmatrix} X &= \begin{bmatrix} \hat{z}_X \\ \hline * \end{bmatrix} - \begin{bmatrix} \hat{\nu}_X \\ \hline * \end{bmatrix} \end{aligned}\right\} \tag{5-8}$$

It is easy to show that $\hat{\Lambda}_X$, the updated ν_X square root covariance, can be obtained from

$$\hat{Q}\left[\begin{array}{c|c} \Lambda_X & 0 \\ \hline 0 & R_\nu\left(R_\nu^a\right)^{-1} \end{array}\right] = \left[\begin{array}{c|c} \Gamma_X & \Gamma_{X\nu} \\ \hline * & * \end{array}\right] \tag{5-9}$$

The Γ array is a square root of $E\left(\nu_X \nu_X^T\right)$, but it is rectangular. For reasons of storage, it is preferable to compress Γ with a triangular form:

$$\left[\hat{\Lambda}_X \mid 0\right] = \left[\Gamma_X \mid \Gamma_{X\nu}\right] Q_\nu \tag{5-10}$$

where Q_ν orthogonal is chosen to triangularize the Γ array. It is clear that $\hat{\Lambda}_X$ obtained this way is such that $\hat{\Lambda}_X \hat{\Lambda}_X^T = \Gamma_X \Gamma_X^T + \Gamma_{X\nu} \Gamma_{X\nu}^T$.

The importance of Eq. 5-9 is that it shows that the Γ terms can be obtained as addended columns to the filter algorithm, i.e.,

$$\hat{Q}\begin{bmatrix} R_X & z_X & \Lambda_X & 0 \\ \hline R_\nu A & R_\nu z & 0 & R_\nu \left(R_\nu^a\right)^{-1} \end{bmatrix} = \begin{bmatrix} \hat{R}_X & \hat{z}_X & \Gamma_X & \Gamma_{X\nu} \\ \hline 0 & * & * & * \end{bmatrix} \tag{5-11}$$

The time propagation portion of the SRIF algorithm is handled in the same way as was depicted in Eqs. 5-10 and 5-11, i.e.,

$$E(ww^T) = \begin{cases} R_w^{-1} R_w^{-T} & : \quad \text{assumed} \\ \left(R_w^a\right)^{-1} \left(R_w^a\right)^{-T} & : \quad \text{actual} \end{cases} \tag{5-12}$$

and the combined time propagation algorithm is (cf Eq. 3-10)

$$\begin{bmatrix} R_p^* & R_{pp}^* & R_{px}^* & z_p^* & * & * \\ \hline 0 & \tilde{S}_p & \tilde{S}_x & \tilde{S}_y & \tilde{\Gamma}_X & \tilde{\Gamma}_{Xw} \end{bmatrix} = \tilde{Q}\begin{bmatrix} -R_w M & R_w & 0 & z_w & 0 & \Lambda_w \\ \hline \hat{S}_p - \hat{S}_x V_p & 0 & \hat{S}_x & \hat{S}_z & \hat{\Lambda}_X & 0 \end{bmatrix} \begin{matrix} \} n_p \\ \} n_p + n_x \end{matrix} \tag{5-13}$$

where $\Lambda_w = R_w\left(R_w^a\right)^{-1}$, and

$$\left[\tilde{\Lambda}_X \mid 0\right] = \left[\tilde{\Gamma}_X \mid \tilde{\Gamma}_{Xw}\right] \tilde{Q}_w, \tag{5-14}$$

with $\tilde{Q}_w$ orthogonal.

NOTE: The entries designated with a "*" represent square root error covariance terms that contribute to the actual smoothing covariance.

One need not derive additional equations to evaluate the effects of using the wrong colored noise, i.e., using colored noise terms in the filter model which have different time constants from those of the actual model. One merely includes the variables in question twice, labeled say as p_f and p_a. The p_f variables assume the filter values and have zero variances* in the actual model; the p_a variables are assumed to have zero variances* in the filter formulation. This simple state augmentation procedure avoids burdensome additional software.

Λ_X included in the filter algorithm as just described includes the effects of using an incorrect initial covariance, incorrect measurement a priori, incorrect process noise a priori, and mismodeling of the colored noise in the filter model. Studying these kinds of effects promotes insight into the filtering process, reveals limitations of the algorithms, and most importantly it influences filter design.

*Information filter formulations can not accomodate zero variances. In practice, small values (based on engineering judgement and computer word-length considerations) are used; the results have been quite satisfactory.

APPENDIX A

PROPERTIES OF HOUSEHOLDER ORTHOGONAL TRANSFORMATIONS

One could use a variety of orthogonal transformations to effect the various matrix triangularizations that have been referred to in this paper. In this appendix we discuss a class of orthogonal transformations (Householder reflections) which have been quite adequate for our purposes. Householder transformations have the form $Q(u) = I - 2\beta uu^T$; $\beta^{-1} = u^T u$ where the components of u are generally chosen to null out certain components of some predetermined vector; this point is amplified in properties A-1 and A-2, below.

The matrix Q(u) plays no explicit role in our estimation work. However, the following list of properties are key to successful application of these transformations.

(A-1) If u is chosen to zero all but the first component, a_1, of vector "a", then one can take $u_i = a_i$ for $i \geq 2$, and $u_1 = a_1 + \text{sgn}(a_1)\sqrt{a^T a}$;

In this case $Q(u)a = (-\text{sgn}(a_1)\sqrt{a^T a},\ 0,\ \ldots,\ 0)^T$.

(A-2) If $u_j = 0$, then Q(u) acting on any vector b (i.e. Q(u)b) leaves b_j unchanged.

(A-3) The matrix Q(u) is an implicit quantity; Q(u)b is expressed directly in terms of b and u as $Q(u)b = b - \lambda u$, where $\lambda = 2\beta(u^T b)$.

(A-4) If b is orthogonal to u then Q(u)b = b.

(A-5) $Q(u) = Q^T(u)$ and $Q^2(u) = I$; i.e. Q(u) is a symmetric and orthogonal matrix.

These properties enter into matrix triangularization of a matrix A by applying a sequence of these elementary transformations. At the jth stage, one chooses a vector $u^{(j)}$ whose first j-1 components are zero and which zeros out all elements of column j (of the (j-1)st transformed matrix $A^{(j-1)}$) that are below j. The jth transformed matrix is $A^{(j)} = Q(u^{(j)})\, A^{(j-1)}$. Observe that $Q(u^{(j)})$ has no effect on the first j-1 columns of $A^{(j-1)}$, because they are orthogonal to $u^{(j)}$. Also, the first j-1 rows of $A^{(j-1)}$ are left unchanged because of property A-2. Thus, $Q(u^{(j)})$ acts only on the lower matrix partition of $A^{(j-1)}$

composed of the last m-j rows and n-j columns, (A(m,n)), and it zeroes out all elements but the first in the first column of this subarray. This method is illustrated in the FORTRAN mechanizations given in Appendix B.

The triangularization procedure applied to

$$Ax = z - \nu$$

where ν has unit covariance, is analogous to a Gaussian elimination process that preserves the unit covariance characteristic of ν.

APPENDIX B

EPOCH STATE FILTER/SMOOTHER FORTRAN MECHANIZATION

The filter/smoother algorithms described in Sections II - IV are FORTRAN mechanized in this appendix. The main reason for including FORTRAN mechanizations is that in this way we can best show how compact and efficient our algorithms are.

Essentially all the filtering computations take place within an array $S\left(n_p + n_x + m,\ n_p + n_x + n_y + 1\right)$ and a vector $R_y\left[n_y\left(n_y + 3\right)/2\right]$ where n_p, n_x, n_y and m are the respective dimensions of p, x, y and the number of measurements.

NOTE: The FORTRAN descriptions to follow do not correspond to exact programming conventions. Greek symbols, lower case letters, square root and inequality symbols, etc., were purposely used in an endeavor to make the code more readable.

I. FILTER: MEASUREMENT UPDATING

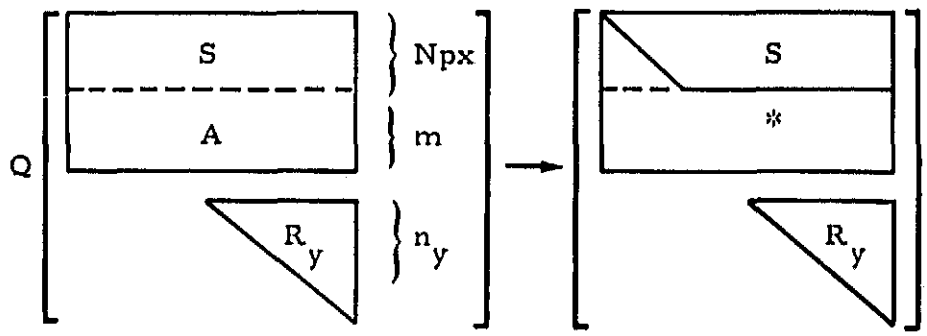
Inputs

S(Npx + m, Npxy1); where Npx = $n_p + n_x$ and Npxy1 = Npx + n_y + 1. The upper Npx rows of S contain the a priori array corresponding to Eq. 3-11; and the bottom m rows correspond to the m measurements, each with unit variance.

$R_y\left[n_y\left(n_y + 3\right)/2\right]$, a vector stored upper triangular matrix, which contains the a priori y information array.

Outputs

The updated SRIF information array occupies the upper Npx rows of S, and the vector R_y contains the updated SRIF information array for the y parameters.




```
      DO 40 J=1, Npx
      σ = z                          @z = zero
        DO 10 I=J, L                 @L = Npx + m
        w(I) = S(I, J)               @w(Npx): work vector
        S(I, J) = z
10      σ = σ + w(I)**2
      If (σ ≤ z) go to 40
c     If σ equals zero, col. J is zero and this
c     step of the reduction is omitted
      σ = √σ
      If [w(J) > z]  σ = -σ
      S(J, J) = σ
      w(J) = w(J)-σ
      σ = one/[σ* w(J)]              @one = 1.
      JP1 = J + 1
        DO 30 K = JP1, Ntot          @Ntot = Npx + n_y + 1
        δ = z
        DO 20 I=J, L
20      δ = δ + S(I, K)*w(I)
      δ = δ*σ
```

```
      DO 30 I=J, L
30    S(I, K) = S(I, K) + δ*w(I)
40    Continue
c
c     This completes the S portion of measurement processing
c
c     Begin y-related portion of measurement processing
c
      KK = 1
      DO 90 J = 1, Ny                    @Ny = n_y
        σ = Ry(KK)**2                    @Ry = R_y
        DO 50 I = 1, m
50      σ = σ + S(Npx + I, Npx + J)**2
      If(σ = z) Go to 90
      σ = √σ
      If [Ry(KK) > z] σ = -σ
      δ = Ry(KK)-σ
      Ry(KK) = σ
      β = one/(σ*δ)
      JJ = KK
      L = J
      JP1 = J + 1
```

```
      DO 80 K = JP1, Ny1              @Ny1 = Ny + 1
      JJ = JJ + L
      L = L + 1
        σ = δ*Ry(JJ)
        DO 60 I = 1, m
60      σ = σ + S(Npx + I, Npx + J)*S(Npx + I, Npx + K)
      If (σ = z) Go to 80
      σ = σ*Beta
      Ry(JJ) = Ry(JJ) + σ*δ
        DO 70 I = 1, m
70      S(Npx + I, Npx + K) = S(Npx + I, Npx + K) + σ*S(Npx + I, Npx + J)
80    Continue
90  KK = KK + J + 1
```

Both segments of the computer code are of interest. The first part, with $n_y = 0$ can be applied to triangularize a system of Npx + m equations in Npx variables; the latter part can be used, with Npx = 0, for recursive triangularization of an overdetermined system where the results are stored in vector form to economize on storage.

II. FILTER: TIME UPDATING (FROM T TO T + DT)

The time updating uses an efficient and compact algorithm that utilizes p only one component at a time; i.e., by considering n_p separate mappings

$$\left.\begin{aligned} \chi_{k+1} &= \chi_k + V_p(k)p_n(k) \\ p_{n+1}(k) &= M_k p_n(k) + w_n(k) \end{aligned}\right\} \qquad k = 1, \cdots, n_p \qquad \text{(B-1)}$$

where

$$\chi_1 = x_n \text{ and } \chi_{n_p+1} = x_{n+1}; \text{ see Eq. 3-4.}$$

The data equation for smoothing corresponding to Eq. 3-12 turns out to be

$$\left.\begin{aligned} &\sigma^*(k)p_n(k) + \sum_{j=1}^{n_p-k} S^*(k,\ j)\ p_n(k+j) \\ &+ \sum_{j=n_p-k+1}^{n_p} S^*(k,\ j)\ p_{n+1}(k+j-n_p) \\ &+ \sum_{j=Np+1}^{N_{px}} S^*(k,\ j)\ \chi_{k+1}(j-n_p) \\ &+ \sum_{j=1}^{n_y} R^*_{py}(k,\ j)\ y_j = z^*_p(k) - \nu^*_p(k) \end{aligned}\right\} \quad k = 1, \cdots, n_p \qquad \text{(B-2)}$$

Inputs

n_p, n_x, n_y, n_d: are the dimensions of p, x, y and the number of dynamically occurring p components.

$\tau(n_p)$: Time constants for the colored noise parameters

$V_p(n_x,\ n_d)$: The first n_d columns of the V_p matrix correspond to the dynamic parameters. The last $n_p - n_d$ columns are omitted because they are in theory zero.

$R_w(n_p)$: Process noise standard deviation reciprocals.

$S(Npx + n_p,\ Ntot)$: $Ntot = Npx + n_y + 1$; the bottom n_p rows of S will be used to store the smoothing related data equation terms.

Outputs

S(Npx + n_p, Ntot): The time updated S information array occupies the upper Npx rows of S, and the lower n_p rows involve data equation information for smoothing. $\sigma^*(n_p)$: Vector of smoothing related output terms.

```
      DO 100 J = 1, Np                                  @Np = n_p
      If [τ(J) ≠ z]    D(J) = EXP[-DT/τ(J)]             @z = zero
100   Continue
  c   Colored noise transition terms are set

      DO 106 J1 = 1, Np
      If (J1 > Nd) Go to 102                            @Nd = n_d
        DO 101 I = 1, Npx
        DO 101 K = 1, Nx                                @Nx = n_x
101     S(I, 1) = S(I, 1) - S(I, Np + K)* V_p(K, J1)
102   λ = -R_w(J1)*D(J1)
        σ = λ**2
        DO 103 K = 1, Npx
        w(K) = S(K, 1)                                  @w(Npx): work vector
103     σ = σ + w(K)**2
      σ = √σ
      λ = λ - σ
      σ*(J1) = σ
      σ = one/(σ*λ)                                     @one = 1.
      DO 105 J2 = 2, Ntot
      δ = z
```

```
c   If (J2 = Ntot)  δ = λ*z_w(J1)
c   Colored noise is generally zero mean; thus z_w is zero
c   in most cases.
      DO 104  I = 1, Npx
104     δ = δ + S(I, J2)*W(I)
      δ = δ*σ
      L = J2 - 1
      If (J2 > Np) L = J2
      S(Npx + J1, L) = δ*λ                          @Used for smoothing
        DO 105  I = 1, Npx
105     S(I, L) = S(I, J2) + δ*w(I)
c   S(Npx + J1, Ntot) = S(Npx + J1, Ntot) + δ*z_w(J1)   @Used for smoothing
c
      δ = λ*R_w(J1)*σ
      S(Npx + J1, Np) = R_w(J1) + δ*λ                @Used for smoothing
        DO 106  I = 1, Npx
106     S(I, Np) = δ*w(I)
```

III. SMOOTHING

Suppose that smoothed values for the computed estimate, X_c^*, computed covariance, P_c^*, and the sensitivity, SEN^*, corresponding to the augmented vector $X = \begin{bmatrix} p \\ x \end{bmatrix}$ have been obtained at time $T + DT$. Smoothed values for these terms, at time T, are obtained in the following way. Start with the previously stored smoothed array $\left[\sigma^*(Np) \mid S^*(Np, Npxy) \mid z_p^*(Np)\right]$, (cf Eq. B-2), where $Npxy = n_p + n_x + n_y$, and the transition elements $V_p\ (n_x, n_d)$. To facilitate algorithm implementation the columns of Eq. (B-2) are rearranged so that the terms p_n (n referring to time T) displace those for p_{n+1}; i. e.,

$$\left[S^*(k, n_p - k + 1), \cdots, S^*(k, n_p),\ S^*(k,1), \cdots, S^*(k, n_p - k)\right]$$

$$k = 1, \cdots, n_p \qquad \text{(B-3)}$$

This arrangement is better suited for the backward computation of $p^*(n_p), \cdots, p^*(1)$. The backward smoother computations which are documented more thoroughly in Ref. 1 are summarized as follows:

For $k = n_p, \cdots, 1$ recursively cycle through Eqs. B-4 through B-15:

$$\gamma := 1/\sigma^*(k) \qquad \text{(B-4)}$$

$$v(j) := S^*(k, j)\gamma, \qquad j = 1, \cdots, Npx \qquad \text{(B-5)}$$

$$\delta := \sum_{j=1}^{N_{px}} v(j) X_c^*(j) \qquad \text{(B-6)}$$

$$X_c^*(k) = z_p^*(k)\gamma - \delta \qquad \text{(B-7)}$$

if $(k \le n_d)$

$$X_c^*(n_p + j) := X_c^*(n_p + j) - V_p(j, k)\, X_c^*(k) \qquad j = 1, \cdots, Nx \qquad \text{(B-8)}$$

At the end of this sequence of n_p steps, X_c^* at time $T + DT$ has been replaced by X_c^* at time T. Compare this with Eq. 4-7.

$$v_y(j) := R_{py}^*(k, j)\gamma, \qquad j = 1, \cdots, n_y \qquad \text{(B-9)}$$

$$\left.\begin{aligned} \delta: &= \sum_{j=1}^{N_{px}} v(j)SEN^{*}(j,m) + v_y(m) \\ SEN^{*}(k,m): &= -\delta \\ SEN^{*}(n_p + j, m): &= SEN^{*}(n_p + j, m) \\ &\quad + V_p(j,k)\delta\,, \qquad j = 1, \cdots, n_x \end{aligned}\right\} \quad m = 1, \cdots, n_y \tag{B-10}$$

Compare this with Eq. 4-6.

$$z(j): = -\sum_{\alpha=1}^{N_{px}} P_c^{*}(j,\alpha)v(\alpha)\,, \qquad j = 1, Npx \tag{B-11}$$

Of the equations involving P_c^{*} only this one involves the lower part of P_c^{*}. In the computer implementation only the upper part of P_c^{*} is used.

$$\left.\begin{aligned} P_c^{*}(m,k): &= z(m) \\ P_c^{*}(m, n_p + j): &= \left.\begin{aligned} &P_c^{*}(m, n_p + j) \\ &- z(m)V_p(j,k) \end{aligned}\right\} \quad \begin{aligned} &j = 1, n_x \\ &(\text{when } k \le n_d) \end{aligned} \end{aligned}\right\} \quad m = 1, \cdots, k-1 \tag{B-12}$$

$$P_c^{*}(k,k): = \gamma^2 - \sum_{j=1}^{N_{px}} v(j)z(j) \tag{B-13}$$

$$P_c^*(k, j) := z(j) \qquad j = k+1, \cdots, n_p \tag{B-14}$$

$$\left.\begin{aligned}
P_c^*(k, n_p + \alpha) &:= z(n_p + \alpha) - P_c^*(k, k) V_p(\alpha, k) \\
P_c^*(j, n_p + \alpha) &:= P_c(j, n_p + \alpha) - z(j) V_p(\alpha, k), \\
&\qquad j = k+1, \cdots, n_p \\
P_c^*(n_p + j, n_p + \alpha) &:= P_c^*(n_p + j, n_p + \alpha) \\
&\quad - V_p(j, k) P_c^*(k, n_p + \alpha) \\
&\quad - z(n_p + j) V_p(\alpha, k) \\
&\qquad j = 1, \cdots, \alpha
\end{aligned}\right\} \quad \alpha = 1, \cdots, Nx \tag{B-15}$$

(Of course, in the computer implementation, Eq. B-15 must be modified when $k > n_d$, because V_p is only implicitly zero for these values of k.)

The FORTRAN mechanization which follows is, in fact, the mechanization currently employed in the JPL Orbit Determination Program. The apparent complication of the algorithm Eqs. (B-4) - (B-16) belies the compactness and efficiency of the FORTRAN realization.

Inputs

X_c^*(Npx): smoothed computed estimates of p and x (independent of y).

P_c^*(Npx, Npx): smoothed computed error covariance for the augmented vector X_c^*. The lower triangular portion of P_c^* is not explicitly used in any of the computations.

$SEN^*(Npx, Ny)$: smoothed sensitivity matrix ($Ny = n_y$).

$V_p(Nx, Nd)$: Transition matrix elements corresponding to the Nd dynamically occurring colored noise variables ($Nx = n_x$, $Nd = n_d$).

$\sigma^*(Np)$, $S^*(Np, Npxy1)$: Data equation coefficients that were stored during the time propagation portion of the filter algorithm. $Npxy1 = Npx + Ny + 1$; the last column of S^* is z_p^*.

X_c^*, P_c^* and SEN^* came from the previous smoothing step (or the filter values if this is the first step). V_p, σ^* and S^* were stored during the time propagation portion of the filter algorithm and are now read back.

Smoothing Outputs: P_c^*, SEN^*, X_c^*

```
      DO 200 K = Np, 1, -1
c     ********************************************************
      L = Np - K                                          @*
        If (L = 0) Go to 110                              @**
        DO 102 J = 1, Np                                  @***
102     v(J) = S*(K, J)                                   @****
        DO 104 J = 1, K                                   @**** Eq. B-3
104     S*(K, J) = v(J + L)                               @***
        DO 106 J = 1, L                                   @**
106     S*(K, J + K) = v(J)                               @*
c     ********************************************************
110   γ = 1./σ*(k)                                        @Eq. B-4
      SUM = 0.
      DO 115 J = 1, Npx
      S*(K, J) = S*(K, J)*γ                               @Eq. B-5
```

```
115  SUM = SUM + S*(K, J)*X*c(J)                          @Eq. B-6
     X*c(K) = S*(K, Npxy1)*γ - SUM                        @Eq. B-7
       If (K > Nd) Go to 125
       DO 120 J = 1, Nx
120    X*c(Np + J) = X*c(Np + J) - Vp(J, K)*X*c(K)        @Eq. B-8
125  If (Ny = 0) Go to 150
     DO 130 J = Npx1, Npxy                                @Npx1 = Npx + 1
130  S*(K, J) = S*(K, J)*γ                                @Eq. B-9
       If (K > Nd) Go to 150
c    ***************************************************
       DO 140 L = 1, Ny                                   @*
       SUM = S*(K, Npx + L)                               @**
       DO 135 J = 1, Npx                                  @****
135    SUM = SUM + S*(K, J)*SEN*(J, L)                    @***** Eq. B-10
       SEN*(K, L) = -SUM                                  @****
       DO 140 J = 1, Nx                                   @**
       SEN*(Np + J, L) = SEN*(Np + J, L) + Vp(J, K)*SUM   @*
c    ***************************************************
150    If (KS = FALSE) Go to 200                          @KS = True (nominally)
c      KS = TRUE if P*c is to be computed
     DO 160 J = 1, Npx
       v(J) = 0.
       DO 155 L = 1, J
155    v(J) = v(J) - P*c(L, J)*S*(K, L)
     I = J + 1
     DO 160 L = I, Npx
```

```
160  v(J) = v(J) - P*c(J, L)*S*(K, L)
  c
  c  160 loop in Eq. B-11, using only the top part of P*c
  c
     JJ = K - 1
     If (JJ = 0) Go to 180
     DO 170 L = 1, JJ                                   @*
     P*c(L,K) = v(L)                                    @**
     If (K > Nd) Go to 170                              @*** Eq. B-12
       DO 165 J = 1, Nx                                 @***
165    P*c(L, Np + J) = P*c(L, Np + J) - v(L)*Vp(J, K)  @**
170  Continue                                           @*
180  P*c(K, K) = γ**2
       DO 182 J = 1, Npx
182    P*c(K, K) = P*c(K, K) - S*(K, J)*v(J)            @Eq. B-13
     If (K = Np) Go to 186
     JJ = K + 1
     DO 184 J = JJ, Np
184  P*c(K, J) = v(J)                                   @Eq. B-14
  c  ************************************************************
186  DO 198 L = 1, Nx
     IJ = Np + L
     P*c(K, IJ) = v(J)
     If (K > Nd) Go to 198
```

```
      If (K = Np) Go to 194
        DO 192 J = JJ, Np
192     P*c(J, IJ) = P*c(J, IJ) - v(J)*Vp(L, K)
194   P*c(K, IJ) = P*c(K, IJ) - P*c(K, K)*Vp(L, K)
      DO 196 J = 1, L
196   P*c(Np + J, IJ) = P*c(Np + J, IJ) - Vp(J, K)*P*c(K, IJ) - v(Np + J)*Vp(L, K)
198   Continue
c     **********************************************************************
c     The 198 loop is Eq. B-15
200   Continue
```

Smoothed estimates and covariances and smoothed consider covariances can be obtained by applying Eqs. 4-10 through 4-12.

REFERENCES

1. Bierman, G. J., "Computational Aspects of Discrete Sequential Estimation," Report 900-661, Jet Propulsion Laboratory, Pasadena, Calif., May 1974 (JPL internal document).

2. Bierman, G. J., "The Treatment of Bias in the Square Root Information Filter/Smoother," J. Optim. Theory and Applic., Vol. 16, Nos. 1/2, 1975, 165-178.

3. Bierman, G. J., "On The Application of Discrete Square-Root Information Filtering," Int. J. Control, Vol. 20, No. 3, 1974, 465-477.

4. Bierman, G. J., "Sequential Square Root Filtering and Smoothing of Discrete Linear Systems," Automatica, Vol. 10, 147-158.

5. Curkendall, D. W., Leondes, C. T., "Sequential Filter Design For Precision Orbit Determination and Physical Constant Refinement," Celestial Mechanics Vol. 8, 1974, 481-494.

6. Dyer, P., McReynolds S., "Extension of Square Root Filtering to Include Process Noise," J. Opt. Theory and Applic., Vol. 3, No. 6, 444-458.

7. Golub, G. H., "Numerical Methods For Solving Linear Least-Squares Problems," Num. Math. 7, 1965, 206-216.

8. Kailath, T., "A View of Three Decades of Linear Filter Theory," IEEE Trans. Inf. Th., Vol. IT-20, No. 2, 1974, 146-181.

9. Kaminski, P. G., Bryson, A. E., Schmidt, S. F., "Discrete Square Root Filtering: A Survey of Current Techniques," IEEE Trans. Aut. Control, Vol. AC-16, 1971, 727-736.

10. Lawson, C. L., Hanson, R. J., Solving Least-Squares Problems, Prentice Hall, Englewood Cliffs, N. J., 1974.

11. Sage, A. P., Melsa, J. L., Estimation Theory With Applications to Communications and Control, McGraw Hill Book Co., New York, 1971.

12. Sorenson, H. W., "Least-Squares Estimation: From Gauss to Kalman," IEEE Spectrum, Vol. 7, July 1970, 63-68.

NASA — JPL — Comf., L.A., Calif.