THE UNIVERSITY OF TEXAS AT AUSTIN
AUSTIN, TEXAS 78712

Final Report for

NASA Grant NSG5154

A STUDY OF AUTONOMOUS SATELLITE NAVIGATION METHODS USING THE GLOBAL POSITIONING SATELLITE SYSTEM

Submitted by

Byron D. Tapley

April 20, 1980

(NASA-CR-162635) A STUDY OF AUTONOMOUS SATELLITE NAVIGATION METHCDS USING THE GLOBAL POSITIONING SATELLITE SYSTEM Final Report (Texas Univ. at Austin.) 55 p HC A04/MF A01 CSCL 17G G3/04

N80-22296

Unclas 17979

Prepared by

Department of Aerospace Engineering and Engineering Mechanics
The University of Texas at Austin
Austin, Texas 78712

Final Report on Grant NSG5154

A Study of Autonomous Satellite Navigation Methods Using the Global Positioning Satellite System

Introduction

The development of the NAVSTAR Global Positioning System (GPS) will allow satellites to perform orbit determination calculations in real time with on-board computers. Special orbit determination algorithms are being developed to accomodate the size and speed limitations of on-board computer systems. One class of these algorithms consists of square root sequential filtering methods. The purpose of the square root filters is to reduce the likelihood of filter divergence which can occur as a consequence of a small computer word length.

In this initial study, a new method for the time update of the square root covariance matrix was developed. In addition, this time update method is compared with another square root covariance propagation method to determine relative performance characteristics. Comparisons are based on the results of computer simulations of the LANDSAT-D satellite processing pseudo range and pseudo range-rate measurements from the Phase I GPS. A summary of the comparison results is contained in the following paragraphs.

Summary of Results

In square root algorithms that employ triangular square root covariance matrices, time propagation by transition matrix methods destroys the triangularity of the square root covariance matrix. Retriangularization is required after each time update. Such a procedure is required of the square root filter (the UDU^T algorithm) currently proposed for the LANDSAT-D computer. In addition to the computational burden of the retriangularization, the effects of process noise can only be approximated, if the algorithm is to be computationally efficient. As a part of this study, an algorithm has been developed which integrates the square root covariance directly in its triangular form. Retriangularizations are not necessary in this algorithm, and the effect of process noise is included exactly.

Appendix A contains a derivation of the proposed propagation algorithm for the UDU^T filter, as well as the results of a performance comparison between the proposed method and a UDU^T algorithm using a transition matrix time update. Two versions of a standard formulation of the Extended Kalman Filter (EKF) are included in the comparison, also. The results show that, for this test problem, the proposed propagation method has superior performance to the transition matrix update formulation in terms of efficiency and accuracy. Its efficiency is only marginally less than that of the EKF formulations. Detailed results are available in Appendix A.

The results of additional algorithm comparisons are contained in Appendix B. Two more square root methods, the Potter and Carlson algorithms have been included in the comparisons. A directly integrated square root covariance propagation algorithm, similar to that derived in Appendix B show, again, that the direct square root covariance updates can be competitive with the transition matrix formulations in terms of computation efficiency and estimation accuracy.

A SEQUENTIAL ESTIMATION ALGORITHM USING A CONTINUOUS UDU^T COVARIANCE FACTORIZATION

B. D. Tapley* and J. G. Peters†

The University of Texas at Austin
Austin, Texas 78712

Nomenclature

a	semi-major axis of satellite orbit
$\bar{a}_d$	atmospheric drag acceleration
b	filter model clock bias
b_ℓ	satellite clock bias
b_s	GPS clock bias
c	speed of light
d	ballistic coefficient
e	eccentricity
f	true anomaly
$\bar{g}$	gravitational acceleration
h	satellite altitude
h_o	scale height for density model
i	inclination
k	density model scaling factor
n	filter model clock drift
n_ℓ	satellite clock drift
n_s	GPS clock drift
$\bar{r}$	satellite inertial position vector
t	true time
T	satellite clock indicated time
$\bar{v}$	satellite inertial velocity vector
$\bar{v}_{rel}$	velocity vector relative to the atmosphere
v_{rel}	magnitude of $\bar{v}_{rel}$
Y_ρ	range measurement
$Y_{\dot\rho}$	range-rate measurement
β	correlation parameter for clock model
γ	atmospheric density
γ_0	atmospheric density at reference altitude
ρ	geometric range
$\dot\rho$	geometric range-rate
ω	argument of pericenter
Ω	longitude of ascending node

Abstract

A method for propagating the square root of the state error covariance matrix in lower triangular UDU^T form is described. The propagation method can be combined with the UDU^T measurement incorporation algorithm to obtain a complete square root free triangular estimation algorithm. The method is compared with (1) the UDU^T state transition matrix propagation algorithm and (2) the conventional sequential estimation algorithms on the basis of estimation accuracy, computational efficiency and storage requirements, by a simulation of LANDSAT-D processing data from the Phase 1 Global Positioning System. The numerical

The research was supported by the National Aeronautics & Space Administration, Goddard Space Flight Center under Grant No. NSG 5154.

*Professor, Department of Aerospace Engineering & Engineering Mechanics; Associate Fellow, AIAA.

†Research Assistant, Department of Aerospace Engineering & Engineering Mechanics; Student Member, AIAA.

INDEX CATEGORIES: Spacecraft navigation, guidance and flight-path control; computer communications, information processing and software; spacecraft systems.

results indicate that, while slower than the conventional methods, the method proposed here is more efficient than the previous UD factorizations with regard to computer storage and computation time, and leads to the most accurate estimate of any of the methods considered.

Introduction

Square root filter formulations have been proposed as a means of eliminating the problem of filter divergence in the real-time application of sequential estimation algorithms. In these methods, the state error covariance matrix is replaced by its square root during the propagation and update of the estimate.[1-5] The state error covariance matrix does not appear explicitly and, if it is required, it can be obtained by multiplying the square root covariance by its transpose. Consequently, it will always be semi-positive definite.

In the initial formulations,[1-4] the enhanced numerical stability was obtained at the expense of increased computation complexity, and an associated increase in computation time. In Ref. 5, a square root measurement update method is proposed which offers potential improvement in the computational efficiency of the square root filtering methods. This efficiency is created by maintaining the square root covariance matrix in triangular form. Following a procedure based on Givens's transformation,[6] an algorithm has been proposed[7] which factors the state error covariance P into the form $P \equiv UDU^T$, where U is unit upper triangular and D is diagonal. Using the UDU^T factorization eliminates the square root functions present in the algorithms discussed in Refs. 1 - 5. The measurement incorporation formulation derived for this factorization technique is summarized in the Appendix. The proposed algorithm[7] for propagating the estimate is summarized in the following paragraphs.

The discrete time propagation equation for the state error covariance matrix is

$$\bar{P}_{k+1} = \Phi(t_{k+1},t_k)\,\hat{P}_k\Phi^T(t_{k+1},t_k) + \Gamma_{k+1} \tag{1}$$

where $\bar{P}_{k+1}$ is the a priori state error covariance matrix at t_{k+1}, $\hat{P}_k$ is the a posteriori state error covariance matrix at t_k, $\Phi(t_{k+1},t_k)$ is the state transition matrix used to map the state from t_k to t_{k+1}, and Γ_{k+1} is the matrix which accounts for the effects of process noise in the interval from t_k to t_{k+1}. The matrices $\Phi(t_{k+1},t_k)$ and Γ_{k+1} satisfy the following equations:

$$\dot{\Phi}(t,t_k) = A(t)\Phi(t,t_k)\;;\;\Phi(t_k,t_k) = I \tag{2}$$

$$\Gamma_{k+1} = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1},\tau)\;Q(\tau)\Phi^T(t_{k+1},\tau)d\tau \tag{3}$$

where $A(t)$ is a known nxn time-dependent matrix and $Q(t)$ is the process noise covariance matrix.

The discrete square root time propagation algorithm, based on the UDU^T transformation, can be summarized as follows[7]: Form the two matrices

$$W_{K+1} \equiv [\Phi(t_{k+1},t_k)\hat{U}_k \;\vdots\; B_{k+1}] \tag{4}$$

$$\tilde{D}_k = \begin{bmatrix} \hat{D}_k & \vdots & 0 \\ \cdots & \vdots & \cdots \\ 0 & \vdots & Q_k/\Delta t \end{bmatrix} \tag{5}$$

where

$$B_{k+1} \equiv \int_{t_k}^{t_{k+1}} \Phi(t_{k+1},\tau)d\tau \tag{6}$$

The process noise covariance matrix, Q_k, is assumed to be constant in the integration interval $\Delta t \equiv t_{k+1} - t_k$, and Γ_{k+1}, as defined in Eq.(3), is approximated as

$$\Gamma_{k+1} \equiv B_{k+1}(Q_k/\Delta t)B_{k+1}^T \tag{7}$$

Then, the updated factors $(\bar{U}_{k+1},\bar{D}_{k+1})$ are obtained in upper triangular and diagonal forms, respectively, by performing a Modified Weighted Gram-Schmidt orthogonalization on the matrix W_{k+1}, where its columns are weighted by the diagonal matrix $\tilde{D}_k$.[7]

The calculation of $\Phi(t_{k+1},t_k)$ requires the integration of nxn equations in addition to the n-state equations. The determination of B_{k+1} necessitates an nxn quadrature. Therefore, the total number of equations to be integrated is 2(nxn) + n. The UDU^T formulation proposed in Ref. 7 approximates B_{k+1} by an analytical trapezoid-rule integration which eliminates the nxn quadrature. The error introduced by this approximation can be neglected if the propagation interval $(t_{k+1} - t_k)$ is small. The effect of error accumulated over long prediction intervals, during loss of tracking or data drop-outs, must be considered to ascertain the accuracy of this approximation.

The matrix multiplication, $\Phi\hat{U}_k$, combined with the creation of the augmented matrix W_{k+1}, destroys the triangularity of the square root covariance matrix. The application of the modified Gram-Schmidt orthogonalization procedure is required to retriangularize the UD factors at the time each measurement is processed. The added computational burden of this orthogonalization at each observation point could be eliminated if the square root covariance matrix were propagated without the loss of its triangularity.

In this investigation, a method is proposed which allows the integration of the continuous state-error covariance differential equations in square root form. The derivation follows the approach used in Ref. 8, but the results are based on the $P \equiv UDU^T$ decomposition. The new algorithm can be combined with a triangular measurement update algorithm to obtain a complete square root estimation algorithm for which square roots are avoided. In addition, the effects of state process noise are included without approximation.

The Square-Root Propagation Equations in Triangular Form

The differential equation for propagating the state error covariance matrix can be expressed as

$$\dot{\bar{P}}(t) = A(t)\bar{P}(t) + \bar{P}(t)A^T(t) + Q(t) \tag{8}$$

where $\bar{P}(t)$ is the a priori state error covariance matrix, $A(t)$ is the nxn linearized dynamics matrix, and $Q(t)$ is the process noise covariance matrix. Each of the matrices in Eq.(8) is time-dependent in the general case. However, for simplicity, the time dependence will not be noted specifically in the following discussion.

If the following definitions are used,

$$\bar{P} \equiv \bar{U}\bar{D}\bar{U}^T \quad ; \quad \bar{Q} \equiv Q/2 \tag{9}$$

and if the first part of Eq.(9) is differentiated with respect to time and substituted into Eq.(8), the results can be rearranged to form

$$(\dot{\bar{U}}\bar{D} + \frac{\bar{U}\dot{\bar{D}}}{2} - \bar{Q}\bar{U}^{-T} - A\bar{U}\bar{D})\bar{U}^T + \bar{U}(\bar{D}\dot{\bar{U}}^T + \frac{\dot{\bar{D}}\bar{U}^T}{2} - \bar{U}^{-1}\bar{Q}^T - \bar{D}\bar{U}^T A^T) = 0. \tag{10}$$

Noting that the first term of (10) is the transpose of the second term, and making the following definition:

$$C(t) \equiv (\dot{\bar{U}}\bar{D} + \frac{\bar{U}\dot{\bar{D}}}{2} - \bar{Q}\bar{U}^{-T} - A\bar{U}\bar{D})\bar{U}^T, \tag{11}$$

one obtains

$$C(t) + C^T(t) = 0 \tag{12}$$

Relation (12) requires that C(t) be eitner the null matrix or, more generally, skew symmetric.

Equation (11) can be simplified by selectively carrying out the multiplication of the $-Q\bar{U}^{-T}$ term by $\bar{U}^T$ to yield, after terms are rearranged,

$$(\dot{\bar{U}}\bar{D} + \frac{\bar{U}\dot{\bar{D}}}{2} - A\bar{U}\bar{D})\bar{U}^T = \bar{Q} + C(t) \equiv \tilde{C}(t) \tag{13}$$

Equation (13) defines the differential equations for $\bar{U}$ and $\bar{D}$ to the degree of uncertainty in C(t). Since the unknown matrix C(t) is skew symmetric, there exist $n(n-1)/2$ unknown scalar quantities in Eq.(13). The problem considered here is one of specifying the elements of C(t) so that $\dot{\bar{U}}$ is maintained in triangular form during the integration of Eq.(13). (The derivation pursued here assumes that $\bar{U}$ is lower triangular and $\bar{D}$ is diagonal, although an algorithm for an upper triangular $\bar{U}$ can be obtained as easily.) The following definitions are made to facilitate the solution to the problem posed above.

$$T \equiv A\bar{U}\bar{D} \quad ; \quad M \equiv \dot{\bar{U}}\bar{D} + \frac{\bar{U}\dot{\bar{D}}}{2} - T \tag{14}$$

With these definitions, (13) is expressed as

$$M\bar{U}^T = \tilde{C} = \bar{Q} + C(t) \tag{15}$$

Since $\dot{\bar{U}}$ and $\bar{U}$ in Eq.(13) are lower triangular, and since from (12), C(t) is skew symmetric, several observations can be made regarding Eq.(15). There are n(n-1)/2 unkown elements in $\tilde{C}$. The products $\dot{\bar{U}}\bar{D}$ and $\bar{U}\dot{\bar{D}}$ are lower triangular creating n(n+1)/2 unknowns. Therefore, the n x n system of equations (15) has [n(n-1)/2 + n(n+1)/2] = n x n unknowns which can be determined uniquely.

An expansion of Eq.(15) into matrix elements indicates the method of solution.

$$\begin{pmatrix} M_{11} & -T_{12} & \cdots & -T_{1n} \\ M_{21} & M_{22} & \cdots & -T_{2n} \\ \vdots & \vdots & & \vdots \\ M_{n1} & M_{n2} & \cdots & M_{nn} \end{pmatrix} \begin{pmatrix} 1 & \bar{U}_{21} & \cdots & \bar{U}_{n1} \\ \cdot & 1 & \cdots & \bar{U}_{n2} \\ \vdots & & & \vdots \\ 0 & & & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \bar{q}_{11} & -C_{21} & \cdots & -C_{n1} \\ C_{21} & \bar{q}_{22} & \cdots & -C_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ C_{n1} & C_{n2} & \cdots & \bar{q}_{nn} \end{pmatrix} \qquad (16)$$

In Eq.(16), $\bar{Q}$ is assumed to be a diagonal matrix with elements $\bar{q}_{ii} = q_{ii}/2$; i=1,...,n. (This assumption can be generalized to allow other non-zero terms in the $\bar{Q}$ matrix with only a slight increase in algebraic complexity.) Each row of the upper triangular portion of the $\tilde{C}$ matrix in Eq.(16) is determined as the product of the corresponding row of the M matrix with the appropriate column of the $\bar{U}^T$ matrix. After an upper triangular row of C is determined, the condition from Eq.(12) that $C_{ij} = - C_{ji}$ $(i=1,\ldots,n\ ;\ j=1,\ldots,i-1)$ is invoked to evaluate the corresponding lower triangular column of C. Then a column of the lower triangular elements of M can be evaluated. Once the elements of the M matrix are determined, the next row of the upper triangular C elements can be computed along with a column of the $\dot{\bar{U}}$ and $\dot{\bar{D}}$ elements. This process is repeated until all $\dot{\bar{U}}$ and $\dot{\bar{D}}$ values are determined. The implementation of this approach proceeds as follows. From Eqs. (13) and (14) one can write

$$M + T = \dot{\bar{U}}\bar{D} + \frac{\bar{U}\dot{\bar{D}}}{2} \quad . \tag{17}$$

The expansion of (17) in summation notation gives

$$M_{ij} + T_{ij} = \sum_{k=1}^{n} \dot{\bar{U}}_{ik}\bar{d}_{kj} + \sum_{k=1}^{n} \frac{\bar{U}_{ik}\dot{\bar{d}}_{kj}}{2}$$

$$i=1,\ldots,n\ ;\ j=1,\ldots,i \tag{18}$$

But, since $\bar{D}$ is diagonal, (18) becomes

$$M_{ij} + T_{ij} = \dot{\bar{U}}_{ij}\bar{d}_{jj} + \frac{\bar{U}_{ij}\dot{\bar{d}}_{jj}}{2}$$

$$i=1,\ldots,n\ ;\ j=1,\ldots,i \tag{19}$$

For $i=j$, $\bar{U}_{ij} \equiv 1$ and $\dot{\bar{U}}_{ij} \equiv 0$. Therefore, (19) becomes

$$\dot{\bar{d}}_{ii} = 2(M_{ii} + T_{ii}) \quad i=1,\ldots,n \tag{20}$$

For $i > j$, (19) is rearranged to obtain the differential equation

$$\dot{\bar{U}}_{ij} = (M_{ij} + T_{ij} - \frac{\bar{U}_{ij}\dot{\bar{d}}_{jj}}{2})/\bar{d}_{jj}$$

$$i=2,\ldots,n \; ; \; j=1,\ldots,i-1 \qquad (21)$$

Equations (20) and (21) are the forms of the differential equations to be employed in the derivative routine of a numerical integrator. The elements T_{ij} and M_{ij} are computed as defined in Eq.(14). The pertinent equations can be combined to obtain the following algorithm.

Triangular Square-Root Propagation Algorithm

Given the elements of the square root state error covariance in lower triangular $\bar{U}\bar{D}$ form, $\bar{Q} \doteq Q/2$, and $A(t)$, the differential equations $\dot{\bar{U}}_{ij}$ and $\dot{\bar{d}}_{ii}$ can be computed as follows:

$$T_{ij} = \sum_{k=j}^{n} A_{ik}\bar{U}_{kj}\bar{d}_{jj} \qquad i=1,\ldots,n \; ; \; j=1,\ldots,n \qquad (22)$$

$$\tilde{C}_{ij} = \sum_{k=1}^{i} M_{ik}\bar{U}_{jk} - \sum_{k=i+1}^{j} T_{ik}\bar{U}_{jk} \qquad i=1,\ldots,n \; ; \; j=i+1,\ldots,n \qquad (23)$$

$$M_{ii} = \bar{q}_{ii} - \sum_{k=1}^{i-1} M_{ik}\bar{U}_{ik} \qquad i=1,\ldots,n \qquad (24)$$

$$M_{ij} = -\tilde{C}_{ji} - \sum_{k=1}^{j-1} M_{ik}\bar{U}_{jk} \qquad i=2,\ldots,n \; ; \; j=1,\ldots,i-1 \qquad (25)$$

$$\dot{\bar{d}}_{ii} = 2(M_{ii} + T_{ii}) \qquad i=1,\ldots,n \qquad (26)$$

$$\dot{\bar{U}}_{ij} = (M_{ij} + T_{ij} - \frac{\bar{U}_{ij}\dot{\bar{d}}_{jj}}{2})/\bar{d}_{jj} \qquad i=2,\ldots,n \; ; \; j=1,\ldots,i-1 \qquad (27)$$

The propagation algorithm summarized in Eqs. (22) through (27) can be combined with the algorithm for incorporating an observation to obtain a complete sequential estimation algorithm in which the covariance matrices $\hat{P}$ and $\bar{P}$ are replaced by the factors $(\hat{U},\hat{D})$ and $(\bar{U},\bar{D})$, respectively. The algorithm given in the Appendix assumes that only a single scalar observation is processed at each observation epoch; however, the algorithm is applicable to the case of multiple observations at a given epoch, if the observation errors are assumed to be uncorrelated.

Numerical Comparison

In the following numerical example, the two methods for propagation of the UDU^T factored covariance matrix are compared to determine the relative computation speed and estimation accuracy. As a basis for determining their absolute performance, numerical results are obtained with the conventional Extended Kalman-Bucy filter using both Eqs.(1) and (8) as the bases for propagating the state error covariance matrix. The numerical comparisons are made by using each of the algorithms to process a set of simulated Global Positioning System (GPS) range and range-rate observations obtained by the LANDSAT-D spacecraft. A detailed discussion of the GPS and the associated navigation measurements is given in Ref. 12. Since the range measurement will require a precise measurement of the time interval between signal transmission from one of the GPS satellites to reception at the LANDSAT-D spacecraft, the clock error must be modeled and included as part of the overall state vector. The primary clock errors are the bias and drift.

The dynamic model used for the motion of LANDSAT and the associated model of the satellite's clock are combined to obtain a filter model which contains nine state variables. The nine components of the state vector are: position ($\bar{r}$; 3x1), velocity ($\bar{v}$; 3x1), clock bias (b), clock drift (n), and clock-drift model correlation parameter (β). The differential equations defining these parameters in the filter are:

$$\dot{\bar{r}} = \bar{v} \tag{28}$$

$$\dot{\bar{v}} = \bar{g} + \bar{a}_d + \bar{\xi}_v \tag{29}$$

$$\dot{b} = n \tag{30}$$

$$\dot{n} = \beta n + \xi_n \tag{31}$$

$$\dot{\beta} = \xi_\beta \tag{32}$$

The stochastic processes, $\bar{\xi}_v$, ξ_n and ξ_β, are white noise forcing functions which are assumed to have the following statistics:

$$E[\xi(t)]_i = 0 \ ; \ E[\xi(t)\xi^T(\tau)]_i = Q_i(t)\delta(t-\tau) \tag{33}$$

where the subscript i indicates the appropriate member of the set $\{\bar{v},n,\beta\}$ and $\delta(t-\tau)$ is the Dirac delta function.

For computational efficiency, the filter assumes simple models for the gravitational acceleration, $\bar{g}$, and for the atmospheric drag acceleration, $\bar{a}_d$. The geopotential model adopted for the filter is obtained by truncating the Goddard Earth Model (GEM7)[11] to the fourth degree and order. The drag acceleration is calculated as

ORIGINAL PAGE IS
OF POOR QUALITY

$$\bar{a}_d = - d\gamma v_{rel}\bar{v}_{rel} \qquad (34)$$

where the atmospheric density, γ , is approximated by the exponential model:

$$\gamma = \gamma_o e^{-k(h-h_o)} \qquad (35)$$

The values of the drag model parameters assumed for this investigation are: $h_o = 840{,}000$ m, $\gamma_o = 5.74 \times 10^{-14}$ kg/m^3, $k = 7.58 \times 10^{-6}$m, and $d = 1.18 \times 10^{-2}$ m^2/kg.

The estimates of the clock bias, b, and drift, n, are used to predict the true time of the user's clock, t, by the equation

$$t = T - b_o - n_o(t-t_o) \qquad (36)$$

where the subscript (o) indicates the epoch of the last estimate of the parameters. The quantity T is the time as indicated by the LANDSAT-D clock.

The observations used for this study are pseudo range and pseudo range-rate measurements as observed by the LANDSAT-D satellite using the six satellites of the Phase I GPS constellation.[12] The computer software used to simulate the observations as well as further details on the simulation procedure are discussed in Ref. 10.

The models used to generate the simulated measurements have the form:

$$Y_\rho = \rho + (b_\ell - b_s)c + \xi_\rho \qquad (37)$$

$$Y_{\dot\rho} = \dot\rho + (b_\ell - n_s)c + \xi_{\dot\rho} \qquad (38)$$

where ρ and $\dot\rho$ are the true values of range and range-rate between LANDSAT-D and a given GPS satellite, Y_ρ and $Y_{\dot\rho}$ are the measured values of the range and range-rate; b_ℓ, n_ℓ, b_s, and n_s are the biases and drifts in the satellite clock and in the GPS clocks, respectively; and c is the speed of light.[12]

The measurement Y_ρ and $Y_{\dot\rho}$ is processed by the LANDSAT-D navigation filter using the model

$$Y_\rho = \rho + cb + \xi_\rho \tag{39}$$

$$Y_{\dot\rho} = \dot\rho + cn + \xi_{\dot\rho} \tag{40}$$

That is, in the filter model the GPS clocks are assumed to be perfect and the total time error is assumed to be contained in the LANDSAT-D satellite clock. The interval between observations 's assumed to be six seconds and observations from only one GPS satellite can be obtained during any six-second interval. The observations from the visible satellites are processed sequentially.

The GPS satellites are assumed to be in circular orbits (e=0) about a point mass earth with inclinations of 63° and periods of 12 hours (43,200 sec). Three satellites are equally spaced on each of two orbital planes. The orbital elements are referenced to a coordinate system whose xy-plane is the earth's equator and whose xz-plane lies along the Greenwich meridian.

The epoch condition for LANDSAT-D was chosen so that the resulting simulated observations would accurately reflect the possible extremes of GPS satellite visibility. The epoch elements chosen are $a \equiv 7.086901 \times 10^6$m, $e \equiv .0001$, $i \equiv 98^\circ\!.181$, $\Omega \equiv 354^\circ\!.878$, $\omega \equiv 180^\circ$, and f(true anomaly) $\equiv -185°$. The elements are specified at a GPS system time t=0. The epoch elements for GPS are specified at a system time of

-7200 sec. The difference in initial epochs is included in the filter program's update of user and GPS states.

The values of the LANDSAT-D and GPS clock phase and frequency errors are simulated as the sum of three different error sources: a noise-free phase error with a polynomial form, an error due to exponentially correlated frequency noise, and a random walk bias error. The exact form of the error models and the coefficients used in the models are given in Ref. 10.

The numerical simulations of the filter performance were made with the following initial conditions:

State	State Covariance	Noise Covariance
$X(1) = 7.046 \times 10^6$ m	$P(1,1) = 6 \times 10^6$ m^2	$Q(1,1) = 0$
$X(2) = -5.433 \times 10^6$ m	$P(2,2) = 6 \times 10^6$ m^2	$Q(2,2) = 0$
$X(3) = -6.120 \times 10^5$ m	$P(3,3) = 6 \times 10^6$ m^2	$Q(3,3) = 0$
$X(4) = 5.562 \times 10^2$ m/sec	$P(4,4) = 1 \times 10^4$ (m/sec)^2	$Q(4,4) = 1 \times 10^{-6}$ m^2/sec^3
$X(5) = -1.116 \times 10^3$ m/sec	$P(5,5) = 1 \times 10^4$ (m/sec)^2	$Q(5,5) = 1 \times 10^{-6}$ m^2/sec^3
$X(6) = 7.388 \times 10^3$ m/sec	$P(6,6) = 1 \times 10^4$ (m/sec)^2	$Q(6,6) = 1 \times 10^{-6}$ m^2/sec^3
$X(7) = 2.998 \times 10^1$ m	$P(7,7) = 3.600 \times 10^3$ m^2	$Q(7,7) = 0$
$X(8) = 5.996 \times 10^{-1}$ m/sec	$P(8,8) = 6. \times 10^0$ (m/sec)^2	$Q(8,8) = 1 \times 10^{-4}$ m^2/sec^3
$X(9) = 5.550 \times 10^{-4}$	$P(9,9) = 1 \times 10^{-5}$	$Q(9,9) = 1 \times 10^{-7}$

The off-diagonal terms of the state-error covariance and noise covariance matrices are set to zero initially.

The data in Table 1 and Table 2 show the relative performance of the four algorithms when each processed a simulated observation data set with a duration of approximately 10000 sec. The five columns in each of the tables are, respectively: (1) the total CPU time required to perform measurement updates of the state and covariance, (2) the total CPU time required for propagating the state and covariance, (3) the total CPU propagation time normalized by the fastest CPU propagation time, (4) the total CPU time required for propagation and measurement updates (the sum of columns (1) and (2)), and (5) the RMS of the position magnitude errors and velocity magnitude errors over the duration of the simulation. All CPU times are listed in milliseconds. Position errors are given in meters and velocity errors are given in meters per second. The algorithms are ranked in the tables in order of increasing total computation time.

Time propagation is performed with a fixed-step modified Euler integrator, which requires two function evaluations per step. The integration step size for the data of Table 1 is six seconds, equal to the time interval between GPS observations. The data of Table 2 result from integration with a three-second step size.

The relations for the propagation of the time bias and drift as approximated by Eqs. (30), (31) and (32) have simple analytic solutions. This allows certain elements of the state transition matrix to be updated analytically. The implementation of the UDU($\dot{\Phi}$) and EKF ($\dot{\Phi}$) algorithms has taken advantage of this simplification to reduce the number of numerically integrated differential equations below the theoretical value of n^2. The high degree of coupling in the covariance differential equations for the

($\dot{P}$) and ($\dot{U}$,$\dot{D}$) algorithms does not permit a convenient reduction in the integration vector size. A total of n(n+1)/2 covariance equations has been integrated in the simulations described here.

While the programming effort required to implement the ($\dot{U}$,$\dot{D}$) formulation will be greater because of the recursive nature of Eqs. (22) through (27), fewer computer storage locations will be required to execute this algorithm since the total number of equations involved in (22) through (27) is n(n+1)/2. This value compares with the (nxn) computer memory locations required to integrate and store the $\dot{\Phi}$ equations. Since there are some zeros in the $\dot{\Phi}$ equation, one can reduce the storage requirements of this method at the expense of added programming complexity. Further comparisons of the computer storage location requirem ats for these two algorithms are given in Ref. 10.

The numerical results shown in Table 1 indicate that the ($\dot{U}$,$\dot{D}$) algorithm is competitive with both the UDU^T($\dot{\Phi}$) and conventional formulations in terms of CPU times and estimation accuracy for this filtering problem. The ($\dot{U}$,$\dot{D}$) method is faster than the UDU^T($\dot{\Phi}$) algorithm for the six-second integration interval. Its position estimation error is lower than that for any of the other algorithms. For the three-second step size results in Table 2, the ($\dot{U}$,$\dot{D}$) algorithm remains competitive in terms of estimation accuracy, but is no longer as fast as the UDU^T($\dot{\Phi}$) algorithm. With the decrease in integration step size, the number of expensive ($\dot{U}$,$\dot{D}$) function evaluations has increased, but the number of time consuming orthogonalizations in the UDU^T($\dot{\Phi}$) algorithm remains the same. This factor causes the UDU^T($\dot{\Phi}$) algorithm to have a faster computation time, in this case. Again, for the three-second results, the ($\dot{U}$,$\dot{D}$) formulation yields the most accurate position estimate.

Both UDU^T methods require higher total computation times than the EKF algorithms. This is not an unexpected result as square root filter formulations often incur computation time penalties as the price for increased numerical stability.

These numerical results were generated on a CDC6600 computer system. The relative performance of the algorithms will vary as a function of the computer system being used, the dynamic model assumed by the filter, the method and order of numerical integration, and the integration step size. The influence on performance of the latter two factors can be seen in the numerical results given in Ref. 10. The $(\dot{U},\dot{D})$ method becomes more efficient as the total number of function evaluations within a given integration interval is decreased. Therefore, the choice of the time propagation method should depend on the formulation of the specific problem under consideration and on the supporting algorithms and computer system used to perform the calculations.

Conclusions

Based on the results presented in the previous discussion, it is concluded that, for the example problem considered here, the $(\dot{U},\dot{D})$ algorithm is more efficient than the (U,D) algorithm based on the $\dot{\Phi}$ propagation. Furthermore, the estimate obtained with the $(\dot{U},\dot{D})$ formulation was more accurate than the estimate obtained by either the conventional EKF estimation algorithms or the (U,D)-$\dot{\Phi}$ algorithm. The performance of the algorithms will be dependent on the computer architecture and software, the dynamic model assumed for the filter and the method used to perform the numerical integrations, and will vary as these factors change.

Table 1. Numerical Algorithm Comparison for the Modified Euler Integrator (Step Size = 6 seconds).

Algorithm	Meas. Update,ms	Time Update,ms	Norm. Time	Total Update,ms	Accuracy	
					RSS Pos.,m	RSS Vel.,m/s
EKF($\dot{P}$)	7.246	38.608	1.000	45.855	113.1	.431
EKF($\dot{\Phi}$)	8.057	42.792	1.108	50.849	120.3	.432
UDU($\dot{U}\dot{D}$)	8.045	51.131	1.324	59.176	111.4	.433
UDU($\dot{\Phi}$)	8.216	61.296	1.588	69.512	117.6	.432

Table 2. Numerical Algorithm Comparison for the Modified Euler Integrator (Step Size = 3 seconds).

Algorithm	Meas. Update,ms	Time Update,ms	Norm. Time	Total Update,ms	Accuracy	
					RSS Pos.,m	RSS Vel.,m/s
EKF($\dot{\Phi}$)	7.981	64.688	1.000	72.669	148.5	.442
EKF($\dot{P}$)	8.341	75.224	1.163	83.565	146.8	.441
UDU($\dot{\Phi}$)	7.867	82.872	1.281	90.739	146.8	.441
UDU($\dot{U}\dot{D}$)	6.972	102.831	1.590	109.803	146.0	.440

Appendix

The measurement update algorithm for the UDU^T factorization[7] has the following form. Using the observation $Y_{k+1} = G(X_{k+1}, t_{k+1})$, calculate:

$$H_{k+1} = [\partial G(\bar{X}_{k+1}, t_{k+1})/\partial \bar{X}_{k+1}] \qquad (A.1)$$

For $i = 1 \rightarrow n$,

$$\tilde{F}_i = H_i + \sum_{k=i+1}^{n} H_k \bar{U}_{ki} \qquad (A.2)$$

$$V_i = \bar{d}_i \tilde{F}_i \qquad (A.3)$$

Set $\beta_{n+1} = R_{k+1}$ (where R_{k+1} is the measurement noise) and calculate:

$$\beta_i = \beta_{i+1} + V_i \tilde{F}_i \qquad ; \ i = n \rightarrow 1 \qquad (A.4)$$

Calculate diagonal covariance elements:

$$\hat{d}_i = \bar{d}_i + \beta_{i+1}/\beta_i \qquad ; \ i = n \rightarrow 1 \qquad (A.5)$$

$$\alpha = \beta_1 \qquad (A.6)$$

For $i = 2 \rightarrow n$, and $j = 1 \rightarrow i-1$, calculate:

$$P_j = \tilde{F}_j/\beta_{j+1} \qquad (A.7)$$

$$B_{ij} = V_i + \sum_{k=j+1}^{i-1} \bar{U}_{ik} V_k \qquad (A.8)$$

$$\hat{U}_{ij} = \bar{U}_{ij} - B_{ij}P_j \qquad ; \begin{array}{l} i = 2 \rightarrow n \\ j = 1 \rightarrow i-1 \end{array} \qquad (A.9)$$

Compute residual:

$$y_{k+1} = Y_{k+1} - G(\bar{X}_{k+1}, t_{k+1}) \qquad (A.10)$$

Calculate gain and update state:

$$\tilde{K}_i = B_{i1} + \bar{U}_{i1}V_1 \qquad ; \ i = 1 \rightarrow n \qquad (A.11)$$

$$\hat{X}_i = \bar{X}_i + \tilde{K}_i y_{k+1}/\alpha \qquad ; \ i = 1 \rightarrow n \qquad (A.12)$$

References

1. Battin, R.H., Astronautical Guidance, McGraw Hill Book Co., New York, 1964, pp. 388-389.

2. Bellantoni, J.F., and Dodge, K.W., "A Square Root Formulation of the Kalman-Schmidt Filter," AIAA Journal, Vol. 5, July 1967, pp. 1309-1314.

3. Kaminsky, P.G., Bryson, A.E., and Schmidt, S., "Discrete Square Root Filtering: A Survey of Current Techniques," IEEE Transactions on Automatic Control, Vol. AC-16, December 1971, pp. 727-736.

4. Andrews, A., "A Square Root Formulati : of the Kalman Covariance Equations," AIAA Journal, Vol. 6, June 1968, pp. 1165-1166.

5. Carlson, N.A., "Fast Triangular Formulation of the Square Root Filter," AIAA Journal, Vol. 11, September 1973, pp. 1239-1265.

6. Gentleman, W.M., "Least Squares Computations by Givens Transformations without Square Roots," Journal of the Institute of Mathematical Applications, Vol. 12, 1973, pp. 329-336.

7. Bierman, G.J., Factorization Methods for Discrete Sequential Estimation, Academic Press, New York, 1976, pp. 124-133.

8. Tapley, B.D., and Choe, C. Y., "An Algorithm for Propagating the Square Root Covariance Matrix in Triangular Form," IEEE Transactions on Automatic Control, Vol. AC-21, February 1976, pp. 122-123.

9. Wooden, W.H., and Dunham, J.B., "Simulation of Autonomous Satellite Navigation with the Global Positioning System," AIAA Paper 78-1429, Palo Alto, California, August 1978.

10. Tapley, B.D., Peters, J.G., and Schutz, B.E., "A Comparison of Square Root Estimation Algorithms for Autonomous Satellite Navigation," Institute for Advanced Study in Orbital Mechanics, Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, Texas, Report No. TR79-1, September 1979.

11. Wagner, C.A., et al, "Improvement in the Geopotential Derived From Satellite and Surface Data (GEM 7 and 8)," NASA-Goddard Space Flight Center, Greenbelt, Maryland, Report No. GSFC X-921-76-20, January 1976.

12. Kruczynski, L., "Global Positioning System Navigation Algorithms," Applied Mechanics Research Laboratory, Department of Aerospace Engineering & Engineering Mechanics, University of Texas at Austin, Texas, Report No. AMRL 1078, May 1976.

APPENDIX B

PAPER 79-150

RELATIVE PERFORMANCE OF ALGORITHMS FOR AUTONOMOUS SATELLITE ORBIT DETERMINATION

By

BYRON D. TAPLEY, G. PETERS, AND B. E. SCHUTZ

Univeristy of Texas
Austin, Texas

AAS/AIAA Astrodynamics Specialist Conference

PROVINCETOWN, MASS/JUNE 25-27, 1979

RELATIVE PERFORMANCE OF ALGORITHMS FOR AUTONOMOUS SATELLITE ORBIT DETERMINATION[1]

B. D. Tapley[2], J. G. Peters[3] and B. E. Schutz[4]

Limited word size in contemporary microprocessors causes numerical problems in autonomous satellite navigation applications. Numerical error introduced in navigation computations performed on small wordlength machines can cause divergence of sequential estimation algorithms. To insure filter reliability, square root algorithms have been adopted in many applications. The optimal navigation algorithm requires a careful match of the estimation algorithm, dynamic model, and numerical integrator. In this investigation, different representations of these elements are evaluated to determine their relative performance for satellite navigation applications. Numerical simulations are conducted using the Phase I GPS constellation to determine the orbit of the LANDSAT-D satellite. Numerical comparisons are made of various square root filter formulations, and their dependence on the order of the integrator is examined.

Nomenclature

$\bar{a}_d$	atmospheric drag acceleration	n	satellite clock drift
b	satellite clock bias	$\bar{r}$	inertial position vector
c	speed of light	t	true time
d	ballistic coefficient	T	satellite clock indicated time
$\bar{g}$	gravitational acceleration	$\bar{v}$	inertial velocity vector
G_ρ	computed range measurement	v_{rel}	magnitude of $\bar{v}_{rel}$
$G_{\dot{\rho}}$	computed range-rate meas.	$\bar{v}_{rel}$	velocity vector relative to the atmosphere
h	satellite altitude		
h_o	scale height for density model	δ	correlation parameter for clock drift model
k	density model scaling factor		

[1] The research was supported by NASA Goddard Space Flight Center under Grant No. NSG5154.

[2] Professor, Aerospace Engineering & Engineering Mechanics, The University of Texas at Austin.

[3] Research Engineer, Dept. of Aerospace Engineering & Engineering Mechanics, The University of Texas at Austin.

[4] Associate Professor, Aerospace Engineering & Engineering Mechanics, The University of Texas at Austin.

γ	variable atmospheric density	$\dot{\rho}_c$	computed geometric range-rate
γ_o	atmospheric density at reference altitude	ρ_m	range measurement
		$\dot{\rho}_m$	range-rate measurement
ρ	geometric range	$\Delta\rho_m$	range bias due to error in user satellite and GPS clocks
$\dot{\rho}$	geometric range-rate		
ρ_c	computed geometric range	$\Delta\dot{\rho}_m$	range-rate bias due to error in satellite and GPS clock

1. Introduction

The use of artificial earth satellites for accurate dissemination of time and frequency holds high potential for the development of an accurate and reliable autonomous navigation system [1]. Current satellite systems have demonstrated the ability to obtain global positioning of points, fixed on the surface of the earth, to an accuracy of two meters in three dimensions [2]. While the position error achieved by a dynamic navigator, i.e., one moving with respect to the earth's surface, would be considerably greater than two meters, this investigation indicates the potential inherent in satellite navigation methods.

The Global Positioning System (GPS)[3], which is being deployed currently, is designed to allow a user to satisfy real-time navigation requirements by the calculation of position and velocity using simultaneous pseudo range and pseudo range-rate measurements from several GPS satellites [4, 5,6]. The requirements for determining the orbits of low altitude satellites in near real-time, coupled with the need for increased accuracy, generates an interest in evaluating the GPS as a means for satisfying satellite orbit determination requirements. With the development of compact low-power computers and atomic clocks, the ability to perform the satellite navigation function on-board the spacecraft in an autonomous navigation mode is an attractive alternative to telemetering the GPS range and range-rate measurements to the ground for processing by a ground-based orbit determination program [6].

Allowable computer storage and execution times will place constraints on the model and the algorithms which can be selected to estimate the satellite's state. To minimize the storage requirements and achieve a real-time state estimate, the estimate of the satellite orbit will be performed on-board, sequentially, using a Kalman-Bucy filter [7]. One problem which must be considered if a sequential data processing method is used is the problem of filter divergence [8]. The divergence occurs due to either (1) dynamic or measurement model error, or (2) numerical errors introduced during the computation process. Since most computers for autonomous satellite navigation will have a short wordlength, this second cause of divergence will be of considerable importance.

The problem of filter divergence has led to a number of studies aimed at the development of stable estimation algorithms. The square-root measurement update algorithm proposed by Potter [9,10] has been used in a number of applications to prevent divergence caused by a computed non-positive definite covariance matrix. The algorithm proposed by Carlson [11] allows the measurement update to be accomplished in an efficient manner by maintaining a triangular square root covariance matrix. In [12], a triangular decomposition which is free of square root operations is proposed. The "square root free" algorithm is reported to be faster than the Carlson-Cholesky algorithm. The square root covariance matrix must be maintained in triangular form during the time propagation if these two algorithms are to be applicable. Usually, the square root covariance matrix time update destroys the matrix triangularity and a retriangularization at each potential observation epoch must be employed. This procedure requires an additional computation effort and an associated computation time penalty. A recent development by Tapley, et al [13,14], in which the time update of the square root covariance matrix is maintained in lower triangular form throughout the entire estimation process, can be adopted to obtain a complete triangular square root estimation algorithm.

The objective of this investigation is to evaluate the performance of the various square root algorithms in performing on-board satellite orbit determination using the Global Positioning System. The evaluation is based on a comparison of the computation time and the state estimate accuracy. The effect of the numerical integration method on the estimate accuracy is considered also.

2. Filter Model

The models of the satellite dynamics and of the observation state relation have a critical impact on the computer storage requirement and on the execution time for the navigation algorithm. If the models are too complex, unacceptably large storage requirements and computation times will occur. However, if the models are too simple, the accuracy of the navigation estimate will be degraded below an acceptable value. The following model is selected as a compromise between the requirements for accuracy and efficiency.

Dynamic Equations. The dynamic model for the motion of the user satellite and the associated model of the satellite clock behavior are combined to obtain a filter model which contains nine state variables. The nine components of the state vector are: position ($\bar{r}$; 3 × 1), velocity ($\bar{v}$; 3 × 1), clock bias (b), clock drift (n), and clock-drift model correlation parameter (β). The differential equations defining these parameters in the filter are:

$$\dot{\bar{r}} = \bar{v} \qquad (1)$$

$$\dot{\bar{v}} = \bar{g} + \bar{a}_d + \bar{\xi}_v \qquad (2)$$

$$\dot{b} = n \qquad (3)$$

$$\dot{n} = \beta n + \xi_n \qquad (4)$$

$$\dot{\beta} = \xi_\beta \qquad (5)$$

where $(\dot{\ })$ = d( )/dT, e.g., the independent variable is the satellite clock time.

The stochastic processes, ξ_v, ξ_n and ξ_β, are white noise forcing functions which are assumed to have the following statistics:

$$\begin{cases} E[\xi(t)]_i = 0 \\ E[\xi(t)\xi^T(\tau)]_i = Q_i(t)\delta(t-\tau) \end{cases} \qquad (6)$$

where the subscript i indicates the appropriate member of the set $\{\bar{v},n,\beta\}$.

The filter uses relatively simple models for gravitational force, $\bar{g}$, and the atmospheric drag forces, $\bar{a}_d$. The geopotential model adopted for the filter is obtained by truncating the Goddard Earth Model (GEM) 7 [15] to the fourth degree and order. The drag acceleration is calculated as

$$\bar{a}_d = -d\gamma v_{rel}\bar{v}_{rel} \qquad (7)$$

where the atmospheric density, γ, is approximated by the exponential model:

$$\gamma = \gamma_o e^{-k(h-h_o)} \qquad (8)$$

The values of the drag model parameters assumed for this investigation are: $h_o = 840{,}000$ m, $\gamma_o = 5.74\times10^{-14}$ kg/m^3, $k = 7.58\times10^{-6}$m, and $d = 1.18\times10^{-2}$ m^2/kg.

The clock bias, b, and drift, n, estimated by the filter, are used to predict the true time of the user's clock by the equation

$$t = T - b_o - n_o(t-t_o) \qquad (9)$$

where the subscript (o) indicates the epoch of the last evaluation of the parameters.

Variational Equations. Linear variational equations are required to implement the sequential estimation algorithm [8]. The equations are derived by the linearization of Eqs. (1) through (5). Further details can be found in [15].

Observation State Relation. The observation types processed during the study are pseudo range and pseudo range-rate. The actual measurements can be expressed, mathematically, as:

$$\rho_m = \rho + \Delta\rho_m + \xi_\rho \qquad (10)$$

$$\dot{\rho}_m = \dot{\rho} + \Delta\dot{\rho}_m + \xi_{\dot{\rho}} \qquad (11)$$

where ρ_m and $\dot{\rho}_m$ are the measured values of the range and range-rate, ρ and $\dot{\rho}$ are the actual geometric range and range-rate, $\Delta\rho_m$ and $\Delta\dot{\rho}_m$ are range and range-rate biases due to errors in the user's clock and the GPS satellite's clock, and the ξ_i are zero mean white noise sequences with known variances.

The modeled values of the measurements have the form

$$G_\rho \equiv \rho_c + (b - b_s)c \qquad (12)$$

$$G_{\dot\rho} \equiv \dot\rho_c + (n - n_s)c \qquad (13)$$

where ρ_c and $\dot\rho_c$ are the computed values of range and range-rate, b, n, b_s and n_s are the predicted biases and drifts in the satellite clock and in the GPS clocks, based on previous measurements, and c is the speed of light.

The linearized observation-state matrix H is computed by taking partial derivatives of G_ρ and $G_{\dot\rho}$ with respect to the state. The complete expressions for G_ρ and $G_{\dot\rho}$, along with the partial derivatives, are given in [15].

3. Observation Simulation Model

Observations generated for this study are pseudo range and pseudo range-rate measurements as observed by the LANDSAT-D satellite using the six satellites of the Phase I GPS constellation. The computer software used to generate the observations is a modification of the program developed by Kruczynski [5]. Further details on the software modifications, which were made to simulate the orbit of the LANDSAT satellite, are given in [15].

The simulation philosophy was to produce a physically realizable set of data points against which the filters could be tested and evaluated. Simplified models of the GPS satellites were used to reduce computer time requirements.

The description of the observation generation program can be broken into four basic areas: (1) simulation of GPS satellite motion, (2) simulation of LANDSAT-D motion, (3) simulation of clocks, and (4) simulation of the measurement process.

Simulation of GPS Satellite Motion. For simplicity, the GPS satellites are assumed to move in circular orbits around a point mass earth. The GPS satellite motion can be determined, then, by using a closed form solution and a set of Keplerian elements defined at some specified epoch. This approximation will not have a significant impact on the results presented here because of the relatively short time interval involved in the simulation.

The epoch orbital elements for the Phase I GPS constellation are the following:

Satellite	Long. of Asc. Node (Deg)	Mean Anomaly (Deg)
1	-130.	0.
2	-130.	40.
3	-130.	80.

(continued...)

Satellite	Long. of Asc. Node (Deg)	Anomaly (Deg)
4	110.	40.
5	110.	80.
6	110.	120.

The GPS satellites are assumed to be in circular orbits ($e = 0$) with inclinations of 63° and periods of 12 hours (43,200 sec). These elements are referenced to a coordinate system whose xy-plane is the earth's equator and whose xz-plane lies along the Greenwich Meridian.

Simulation of LANDSAT-D Motion. The force model used to simulate the LANDSAT-D motion contains the effects of the earth's non-spherical mass distribution and the effects of atmospheric drag. The gravitational accelerations for the observation simulations are obtained using the GEM7 geopotential model truncated at order and degree 8. The drag acceleration is computed by using Eqs. (7) and (8), with the same set of constants as specified previously.

The epoch conditions for LANDSAT-D and the GPS satellites were chosen so that the resulting simulated observations would accurately reflect the possible extremes of GPS satellite visibility. The epoch elements chosen for LANDSAT-D are:

$$a \equiv 7.086901 \times 10^6 \text{m}$$

$$e \equiv 0.001$$

$$i \equiv 98^\circ\!.181$$

$$\Omega \equiv 354^\circ\!.878$$

$$\omega \equiv 180^\circ$$

$$f \text{ (true anomaly) } \equiv -185^\circ\!.$$

These elements are specified at the GPS system time $t = 0$. The epoch elements previously listed for the GPS constellation are specified at the GPS system time $t = -7200$ sec. This difference in initial epoch is included in the simulation program's updates for the user and GPS states.

Simulation of Clocks. The values of the LANDSAT-D and GPS clock phase and frequency errors can be calculated as the sum of three different error sources: a noise-free phase error with a polynomial form (ε_1); an error due to exponentially correlated frequency noise ($\dot{\varepsilon}_2$); and a random walk bias error (ε_3).

The polynomial error term is expressed mathematically as:

$$\varepsilon_1(t) = a_2 + a_3(t-a_1) + (a_4/2)(t-a_1)^2 \qquad (14)$$

The exact form of the error models and the coefficients used in the models are given in [5].

Simulation of Measurement Process. To generate the observations, the GPS satellites are "sampled" every six seconds in ascending numerical order. Upon calculation of the geometric range vector between the LANDSAT-D and a GPS satellite, a measurement is accepted if the GPS satellite is no more than 20° below the LANDSAT-D local horizontal. If a satellite is rejected because of this geometric constraint, the next higher numbered satellite is sampled in the same manner. The procedure is repeated until a "visible" GPS satellite is found. If none of the six satellites is considered visible at a particular time, no measurement is taken. Then, all satellite vehicles are propagated forward six seconds and the procedure is repeated. A random number of measurements is rejected in an effort to simulate measurement losses due to actual system problems such as periodic failures in signal acquisition or bad data in the GPS transmission. A Gaussian error term is added to each of the geometric observations to account for purely random anomalies in the measurement process. The standard deviations for the random errors are: $\sigma_\rho = 2.\,\mathrm{m}$, and $\sigma_{\dot{\rho}} = .2\,\mathrm{m/sec}$. If no satellites are visible at a particular time, the user's clock errors and position and velocity magnitudes are recorded on a file to be used to compute navigation errors during data gaps.

4. Filter Algorithms

The filter used to process the GPS measurements will consist of two major segments. These segments are: (1) the measurement update segment, and (2) the time propagation segment. The measurement update segment receives the observations at a given time epoch and processes these observations to obtain an updated estimate of the state. The propagation segment maps the estimate and the associated state error covariance matrix forward in time to the next observation epoch. For each measurement update algorithm, there are two propagation algorithms which can be considered. The primary difference in the propagation algorithms is determined by whether one integrates the state transition matrix for propagating the state error covariance matrix or whether the differential equation for the state error covariance matrix is integrated directly. The filter algorithms compared in this investigation are:

(1) The Extended Kalman-Bucy Filter [8,10]

(2) The Carlson-Cholesky Filter [11,13]

(3) The Potter Filter [9,10]

(4) The UDU Filter [12,14]

The state error covariance matrix can be based on the following set of differential equations:

$$\dot{\bar{P}}(t) = A(t)\bar{P}(t) + \bar{P}(t)A^T(t) + Q(t) \tag{15}$$

where $A(t) = [\partial F(X,t)/\partial X]$, the $n \times 1$ state vector, $X(t)$ is defined to have the components $X^T(t) = [\vec{r}^T \vdots \vec{v}^T \vdots b,n,\varepsilon]$, $F(X,t)$ is an $n \times 1$ vector whose components are the right-hand sides of Eqs. (1) through (5), and $[\ ]^*$

indicates that the elements of the matrix are evaluated on the reference solution $X^*(t)$. Alternately, the covariance matrix can be propagated by using the state-transition matrix, $\Phi(t,t_k)$, where $\dot{\Phi}(t,t_k) = A(t)\Phi(t,t_k)$; $\Phi(t_k,t_k) = I$. The integral of Eq. (15) can be expressed, then, as:

$$\bar{P}_{k+1} = \Phi(t_{k+1},t_k)P_k\,\Phi^T(t_{k+1},t_k) + \int_{t_k}^{t_{k+1}} \Phi(t,\tau)Q(\tau)\Phi^T(t,\tau)d\tau \quad (16)$$

Rather than evaluate the integral in Eq. (16), an average value, Γ_k, is used, where

$$\Gamma_k = \text{avg}\int_{t_k}^{t_{k+1}} \Phi(t,\tau)Q(t)\Phi^T(t,\tau)d\tau \quad (17)$$

Using Eq. (17), Eq. (16) can be expressed as:

$$\bar{P}_{k+1} = \Phi(t_{k+1},t_k)P_k\,\Phi^T(t_{k+1},t_k) + \Gamma_k \quad (18)$$

The square root estimation algorithms can be based on either Eq. (15) or Eq. (18). In the numerical simulations described in the next section, methods based on both approaches are compared. The use of Eq. (15) allows a triangular square root factorization for the covariance to be maintained during the propagation interval. Propagation with Eq. (18) will destroy triangularity and, after the propagation interval, special computation techniques are required to obtain a new square root covariance matrix. For the square root propagation algorithms based on Eq. (15), the relative advantage of maintaining the covariance matrix in triangular form is offset by a more complicated form for the governing differential equation.

Further discussion of the algorithms as well as the specific implementation used for this investigation is given in [15].

5. Algorithm Comparison

The numerical performance of the algorithms discussed in Section 4 was compared by conducting a series of computer simulations in which the algorithms were used to process GPS range and range-rate observations. Observations were simulated for a GPS system time interval of 10,000 sec. from the LANDSAT-D orbit. This is approximately 1.7 revolutions of LANDSAT-D. A history of the number of GPS satellites visible from LANDSAT-D, versus time, is shown in Fig. 1. It can be seen that the number of satellites visible varies from zero to six, the number of satellites in the Phase I GPS constellation.

The numerical results obtained with each filter are very similar to those shown in Figs. 2, 3, and 4. The plots in these three figures are, respectively, RSS position error versus time, RSS velocity error versus time, and the error in the time estimate versus time. This solution was generated with the UDU(Φ) algorithm and a modified Euler integrator with an integration step size of six seconds.

Two types of error growth are visible in each of the figures. First, there are two time periods in each of the plots during which large error growth occurs. Comparison of the three figures with Fig. 1 shows a coincidence between the occurrence of the large error and the periods when fewer than four GPS satellites are visible. The large errors result from the inability of the filter model to predict the state estimate through time periods of limited observation data.

The second type of error growth occurs gradually. After each period of low satellite visibility, the navigation error is reduced to a lower level. However, the average value of the lower-level navigation error increases during each successive period of good satellite visibility. Additional simulations run with algorithms other than the one used to generate Figs. 2, 3, and 4 produced nearly identical error plots. It is likely that the long-term error growth is caused by the influence of geopotential model error on the estimate of the LANDSAT-D clock parameters. The large errors result from the periodic reduction of GPS satellite visibility. Both error types require further study; however, since the primary purpose of this investigation is the comparison of the algorithm efficiencies, in-depth study of the causes of the large error growth and methods for their removal is deferred to a later investigation.

Operation Count Comparison. As an evaluation of the theoretical computational efficiency, operation counts of the number of numeric operations have been made for the seven different time update algorithms under consideration. The operations recorded are the additions (subtractions), multiplications, divisions, and square roots required to perform a single time update of the state error covariance matrix or the square root covariance matrix. The operation counts required to incorporate the measurements are discussed in [12].

Table 1 and 2 give the number of operations for the covariance time update of a system with an n-dimension state vector, whose complete nxn-transition matrix is obtained by numerical integration. The dimension of the system's state noise covariance matrix is m. The counts are broken into three groups depending on whether the operations occur once per time update interval, once per integration step, or once per integration function evaluation.

The operation count per integration step depends on the type of numerical integrator being employed. Tables 1 and 2 give the values for a second-order Euler integrator with two function evaluations per step. The coefficients of the n^2 terms in the table will increase significantly for higher order integrators (roughly as a factor of $k(k+1)/2$, where k is the number of function evaluations).

If the assumption of an Euler integrator is maintained and the integration step is specified, then the three count groups can be combined to give the total number of operations for a covariance time update.

Assuming that a time update occurs every six seconds and the integration step size is six seconds, the total number of operation counts can be computed. The result is shown in Table 3.

The total counts show that direct covariance integration algorithms have fewer operations if all transition matrix elements are numerically integrated in the (Φ) algorithms. In this situation, the direct integration methods have fewer equations to numerically integrate, fewer operations in each function evaluation, and are not forced to retriangularize at the end of the update.

However, it is often the case that the solution to some of the elements of the transition matrix can be obtained analytically. In such a case, the number of terms to be numerically integrated can be reduced. A similar reduction in integration vector size has not proven feasible for the algorithm based on direct integration of the covariance matrix. Therefore, a full $n(n+1)/2$ set of covariance elements must be integrated numerically.

The actual numerical operation counts for each time update algorithm are shown in Table 4. These values reflect the following set of assumptions:

(1) The state vector size, n, is 11; the measurement noise covariance vector, m, has the dimension of 8. The eleven state filter is the maximum filter size used in these studies. For the 11-state filter, a two parameter model is used to estimate the drag coefficient [14].

(2) Use of all possible analytical function matrix updates (elements updated analytically are those derived from the clock parameter differential equations).

(3) The modified Euler integrator with two function evaluations per step is used.

(4) An integration step size of six seconds is adopted.

The results can be combined into an equivalent addition count by weighting the multiplications, divisions and square roots by their relative execution times. The simulations have been performed on a CDC6600 computer system which has the following operation times weights:

Add	1
Mult	2.5
Div	7.25
Sq.Root	62.5

The use of these weights with the counts of Table 4 gives the operation counts in Table 5 expressed in equivalent numbers of additions. Total value of equivalent additions in Table 5 indicates the relative efficiency of the algorithms in performing covariance time updates under the assumptions described above. It can be observed from Table 5 that the square root algorithm based on the $P=UDU^T$ transformation in combination with the Φ propagation compares quite favorably with the conventional extended Kalman-Bucy filter. Note that if the symmetric properties of Eq. (15) are used, the $\dot{P}$ operation count is substantially less than either of the other algorithms. Detailed investigations have indicated that there are numerical problems associated with integrating the nx(n+1)/2 differential equations obtained by invoking the symmetry requirements on $\dot{P}$. This fact illustrates the point that the numerical stability is as important as numerical efficiency in any autonomous satellite application.

Numerical Comparison. In addition to the operation counts described in the previous section, specific numerical simulations were performed on the algorithms to determine their actual relative performance in performing navigation computations. Tables 6 through 9 contain the results obtained in these simulations. The numerical results were obtained using the following conditions for numerical integration of the appropriate differential equations.

(1) Variable step (2)4 Runge-Kutta with the absolute single-step error tolerance of 10^{-2}, and a relative single-step tolerance of 10^{-6},

(2) Fixed step fourth order Runge-Kutta with a six-second step size,

(3) Fixed step, second-order Euler integrator with a three-second step size, and

(4) Fixed step, second-order Euler integrator with a step size of six seconds.

The four different integrators were used to show how the relative performance changes as a function of the order and method of integration, the integration step size, and the method by which the integration step is selected.

The data are presented in each of the tables in the following manner. The columns are, from left to right:

1. The algorithm used for covariance matrix propagation,
2. The total measurement update time,
3. The total propagation time,
4. The propagation time normalized by the lowest propagation time,
5. The total computation time for time and measurement updates,
6. The RSS of the position magnitude error,
7. The RSS of the velocity magnitude error.

The times are given in seconds; the position errors are expressed in meters, and velocity errors are given in meters per second. The algorithms are listed in order of increasing total computation times.

The results indicate that the algorithm performance has a strong dependence on the characteristics of the numerical integration algorithm. In general, the $(\dot{\Phi})$ algorithms have lower propagation times than their directly integrated counterparts. The significant deviation from this rule is shown in the results of Table 9, where the direct integration methods shows computation time advantages.

The other variations in performance of the algorithms are quite unexpected. Although the computation times of all algorithms decrease as the order of integration is lowered and the step size is increased, the estimation accuracies of the algorithms increase with thesc same changes in integration characteristics. In one case, the $(\dot{U}\dot{D})$ algorithm fails to complete a simulation with the fourth order integrator and a six-second integration step but runs successfully and competitively with the second-order integrator at the same step size.

To rule out the possibility of computer roundoff error as a cause of this anomaly, tests were conducted on the Euler integrator. With step sizes as low as 0.5 sec., the Euler algorithm approached the results of a well validated high order multistep integration code [16]. These results ruled out the liklihood of roundoff error or coding error as the cause for the anomalous characteristics in the estimation accuracy. The effect is believed to be caused by a complex interaction between the stability characteristics of the estimator and the integrator.

The relative performance of the algorithms in Table 9 leads to a second unexpected trend in the numerical results. The predicted relative performance based on the operation counts in Table 5 does not agree with the actual results obtained in Table 9. Specifically, the numerical performance of the $\dot{U}\dot{D}$ algorithm is better than the relative performance predicted by the operation counts while the $\dot{W}$ is not as good. The cause for this discrepancy is thought to be coding overhead not accounted for in the operation count or the possible parallel multiplication capability of the CDC6600. This question requires further investigation.

6. Conclusions

Based on the results given in the previous section, several general conclusions can be drawn. First, the estimation algorithm performance has a strong dependence on the order and method used for integrating the differential equations involved in propagating the state estimate and the state estimate covariance matrix between observation epochs. This dependence has not been fully understood and requires further consideration. Based on the results presented here, the square root method based on the UD transformation in combination with the state transition matrix propagation approach appears to be the best overall square root method. However, for the Euler integrator using a six-second integration step, the best overall results were obtained with the UD algorithm. Finally, the single-step Euler numerical integration algorithm yields a more accurate and stable estimate than the fourth order Runge-Kutta algorithms.

In relating the results presented in this paper to the microprocessor environment, several factors should be remembered. Of greatest importance is the fact that the timing comparisons have been obtained under conditions

which are different than those existing for on-board computer implementations. Table 3 and 4 are of most significance for the on-board application in which a high-level language, such as FORTRAN, will probably not be used. The overhead associated with FORTRAN is the most probable cause of the discrepancies between the performance comparisons in terms of operation count versus execution times. The on-board implementation will be in assembly language and should approach the operation count performance given in Tables 3 and 4, although other factors must be considered also.

References

1. Ehrilich, E., "The Role of Time-Frequency in Satellite Position Determination Systems," Proceedings of the IEEE, Vol. 60, No. 5, May 1972, pp. 564-569.

2. Anderle, R.J., Transformation of Terrestrial Survey Data to Doppler Satellite Datum", Jr. of Geophysical Research, Vol. 79, No. 35, Dec. 1974, pp. 5319-5332.

3. Duiven, Edward N., and A. Richard LeShack, "Global Positioning System," Simulation Development and Analysis," The Analytic Sciences Corporation, Tech. Report No. TR0428-2, December 15, 1974.

4. Bogen, A. H., "Geometric Performance of the Global Positioning System, The Aerospace Corporation, El Segundo, California, Report No. SAMSO-TR-74-169, June 21, 1974.

5. Kruczynski, L. R., "Global Positioning System Navigation Algorithms," Applied Mechanics Research Laboratory, Report No. ARML 1079, Department of Aerospace Engineering & Engineering Mechanics, The University of Texas at Austin, Texas, May 1977.

6. Fuchs, A.J., W. H. Wooden, II and A. C. Long, "Autonomous Satellite Navigation with the Global Positioning System," AAS/AIAA Astrodynamics Conference, Jackson Hole, Wyoming, September, 1977.

7. Kalman, R. E., and R. S. Bucy, "New Results in Linear Filtering and Prediction Theory," Journal of Basic Engineering, March 1961, pp. 95-108.

8. Tapley, B. D., "Statistical Orbit Determination Theory," Recent Advances in Dynamical Astronomy, B. D. Tapley and V. Szebehely, Eds., D. Reidel Publishing Company, Dordrecth, Holland, 1973, pp. 396-425.

9. Battin, R. H., Astronautical Guidance, McGraw-Hill Book Co., New York, 1964, pp. 338-389.

10. Kaminsky, P. G., A. E. Bryson, and S. Schmidt, "Discrete Square Root Filtering: A Survey of Current Techniques," IEEE Trans. on Automatic Control, Vol. AC-16, No. 6, December 1971, pp. 727-736.

11. Carlson, N.A., "Fast Triangular Formulation of the Square Root Filter," AIAA Jr., Vol. 11, No. 9, September 1973, pp. 1239-1265.

12. Thornton, C.L., "Triangular Covariance Factorizations for Kalman Filtering," Tech. Memo 33-798, Jet Propulsion Laboratory, Pasadena, California, October 1976.

13. Tapley, B.D., and C.Y. Choe, "An Algorithm for Propagating the Square Root Covariance Matrix in Triangular Form," IEEE Trans, on Automatic Control, Vol. 21, No. 1, 1976.

14. Tapley, B. D., and G. Peters, "A Triangular Covariance Factorization for Sequential Filtering Algorithms," AIAA/AAS Astrodynamics Conference, Palo Alto, California, August 1978/

15. Tapley, B. D., G. Peters, and B. E. Schutz, "A Comparison of Square-Root Estimation Algorithms for Autonomous Satellite navigation," Institute for Advanced Study in Orbital Mechanics, Report No. TR79-1, The University of Texas at Austin, December 1978,

16. Shampine, L. F., and M. K. Gordon, Computer Solution of Ordinary Differential Equations, W. H. Freeman and Company, San Francisco, 1975.

Table 1. Operation Counts for ($\dot{\Phi}$) Algorithms Broken Down by Frequency.

Algorithm	Adds	Mults	Divs	Sq.Rts.
EKF($\dot{\Phi}$) per update	$2n^3+2m$	$2n^3+m+1$	-	-
per step*	$3n^2$	$2n^2$	-	-
per fcn.eval.	n^3	n^3	-	-
UDU($\dot{\Phi}$) per update	$1.5n^3+.5n^2+n$ $+5m+n^2m$	$1.5n^3+2n^2+.5n$ $+m+(n^2+n)m+1$	$n-1$	-
per step	$3n^2$	$2n^2$	-	-
per fcn.eval.	n^3	n^3	-	-
CARL($\dot{\Phi}$) per update	$1.5n^3+n^2+n^2m$ $+m$	$1.5n^3+n^2-.5n$ $+n^2m+m$	$n-1+m$	-
per step	$3n^2$	$2n^2$	-	-
per fcn.eval.	n^3	n^3	-	-
POTT($\dot{\Phi}$) per update	$2n^3+n^2m+m$	$2n^3+.5n^2-.5n$ $+n^2m+m$	$n-1+m$	-
per step	$3n^2$	$2n^2$	-	-
per fcn.eval.	n^3	n^3	-	-

*per step calculations assume second-order Euler integrator with two function evaluations.

Table 2. Operation Counts for ($\dot{P}$) Algorithms Broken Down by Frequency.

Algorithm	Adds	Mults	Divs	Sq.Rts.
$\dot{P}$ per step	$3(n^2+n)/2$	(n^2+n)	-	-
per fcn. eval.	$n^3+n^2+m^2$	n^3	-	-
$\dot{U}\dot{D}$ per step	$3(n^2+n)/2$	n^2+n	-	-
per fcn. eval.	$n +4n +2n-3$	$n +3n +n-3$	n^2+n	
$\dot{W}$ per step	$3(n^2+n)/2$	n^2+n	-	-
per fcn. eval.	n^3+n^2	$n^3-.5n^2+.5n$	$.5(n^2+n)$	-

Table 3. Total Operation Per Time Update Weighted by Frequency

Algorithm	Additions	Multiplications	Divs.	Sq.Rts.
EKF($\dot{\Phi}$)	$4n^3+3n^2+2m$	$4n^3+2n^2+m+1$	-	-
UDU($\dot{\Phi}$)	$3.5n^3+3.5n^2+n+5m+n^2m$	$3.5n^3+4n^2+.5n+m(n^2+n)+1$	$n-1$	-
CARLSON($\dot{\Phi}$)	$3.5n^3+4n^2+n^2m+m$	$3.5n^3+3n^2-.5n+n^2m+m$	$n-1+m$	$n+m$
POTTER($\dot{\Phi}$)	$4n^3+3n^2+n^2m+m$	$4n^3+2.5n^2-.5n+n^2m+m$	$n-1+m$	$n+m$
$\dot{P}$	$2n^3+5n^2+3n+m^2$	$2n^3+2n^2+2n$	-	-
$\dot{U}\dot{D}$	$2n^3+9.5n^2+5.5n-6$	$2n^3+7n^2+3n-6$	$2(n^2+n)$	
$\dot{W}$	$2n^3+3.5n^2+1.5n$	$2n^3+2n$	(n^2+n)	

Table 4. Numerical Operation Counts, GPS Problem.

Algorithm	Additions	Multiplications	Divisions	Sq.Roots
EKF($\dot{\Phi}$)	3434	3455	12	-
UDU($\dot{\Phi}$)	3880	4035	22	-
CARLSON($\dot{\Phi}$)	3837	3824	30	19
POTTER($\dot{\Phi}$)	4442	4429	30	19
$\dot{P}$	3230	2792	-	-
$\dot{U}\dot{D}$	3866	3536	264	-
$\dot{W}$	3102	2684	66	-

Table 5. Numerical Operation Counts (GPS Problem). Equivalent Additions (CDC6600) for Modified Euler, Step Size = 6.

Algorithm	Additions	Wghtd.Mults.	Wghtd.Divs.	Wghtd. Sq.Rts.	Σ
EKF($\dot{\Phi}$)	3434.	8637.5	87.	-	12156.5
UDU($\dot{\Phi}$)	3880.	10085.0	159.5	-	14124.5
CARLSON($\dot{\Phi}$)	3837.	9560.0	217.5	1187.5	14802.0
POTTER($\dot{\Phi}$)	4442.	11072.5	217.5	1187.5	16919.5
$\dot{P}$	3230.	6980.0	-	-	10210.0
$\dot{U}\dot{D}$	3866.	8840.	264.	-	14620.0
$\dot{W}$	3102.	6710.	478.5	-	10290.5

Table 6. Numerical Algorithm Comparison. Variable Step (2)4 Runge-Kutta. Error Tolerance: REL = 10^{-2}, ABS = 10^{-6}

Algorithm	Meas. Update	Time Update	Norm Time	Total Update	Accuracy	
					RSS Pos.	RSS Vel.
EKF($\dot{\Phi}$)	8.172	67.377	1.000	75.549	163.4	.452
EKF($\dot{P}$)	8.255	83.325	1.237	91.580	163.3	.452
UDU($\dot{\Phi}$)	8.110	86.241	1.280	94.351	163.2	.452
CARL($\dot{\Phi}$)	18.459	87.820	1.303	106.279	163.2	.452
POTT($\dot{\Phi}$)	9.981	97.158	1.442	107.139	163.4	.452
CARL($\dot{W}$)	20.198	98.321	1.459	118.519	163.4	.452
UDU($\dot{U}\dot{D}$)	7.729	119.456	1.773	127.185	163.3	.452

Table 7. Numerical Algorithm Comparison.
Fourth Order Runge-Kutta, Step Size = 6 sec.

Algorithm	Meas. Update	Time Update	Norm Time	Total Update	Accuracy	
					RSS Pos.	RSS Vel.
EKF($\dot{\Phi}$)	7.909	65.340	1.000	73.249	163.4	.452
EKF($\dot{P}$)	9.144	77.024	1.179	86.168	163.3	.452
UDU($\dot{\Phi}$)	8.015	84.362	1.291	92.377	163.2	.452
CARL($\dot{\Phi}$)	19.361	84.784	1.298	104.145	163.2	.452
POTT($\dot{\Phi}$)	9.379	95.122	1.456	104.501	163.2	.452
CARL($\dot{W}$)	19.045	96.933	1.484	115.978	163.4	.452
UDU($\dot{U}\dot{D}$)	-	-	-	-	-	-

Table 8. Numerical Algorithm Comparison.
Modified Euler, Step Size = 3 sec.

Algorithm	Meas. Update	Time Update	Norm Time	Total Update	Accuracy	
					RSS Pos.	RSS Vel.
EKF($\dot{\Phi}$)	7.981	64.688	1.000	72.079	148.5	.441
EKF($\dot{P}$)	8.341	75.224	1.163	83.565	146.8	.441
UDU($\dot{\Phi}$)	7.867	82.872	1.281	90.739	146.8	.441
CARL($\dot{\Phi}$)	19.110	83.625	1.293	102.735	146.8	.441
POTT($\dot{\Phi}$)	9.893	93.319	1.443	103.212	146.8	.441
UDU($\dot{U}\dot{D}$)	6.972	102.631	1.590	109.803	146.0	.440
CARL($\dot{W}$)	18.818	95.112	1.470	113.930	146.5	.441

Table 9. Numerical Algorithm Comparison.
Modified Euler, Step Size = 6 sec.

Algorithm	Meas. Update	Time Update	Norm Time	Total Update	Accuracy	
					RSS Pos.	RSS Vel.
EKF($\dot{P}$)	7.246	38.609	1.000	45.555	113.1	.431
EKF($\dot{\Phi}$)	8.057	42.792	1.108	50.849	120.3	.432
UDU($\dot{U}\dot{D}$)	8.045	51.131	1.324	59.176	111.4	.433
CARL($\dot{W}$)	19.402	42.886	1.111	62.288	126.3	.438
UDU($\dot{\Phi}$)	8.216	61.296	1.588	69.508	117.6	.432
CARL($\dot{\Phi}$)	19.122	62.404	1.616	81.526	117.6	.432
POTT($\dot{\Phi}$)	10.186	71.897	1.862	82.083	117.6	.432

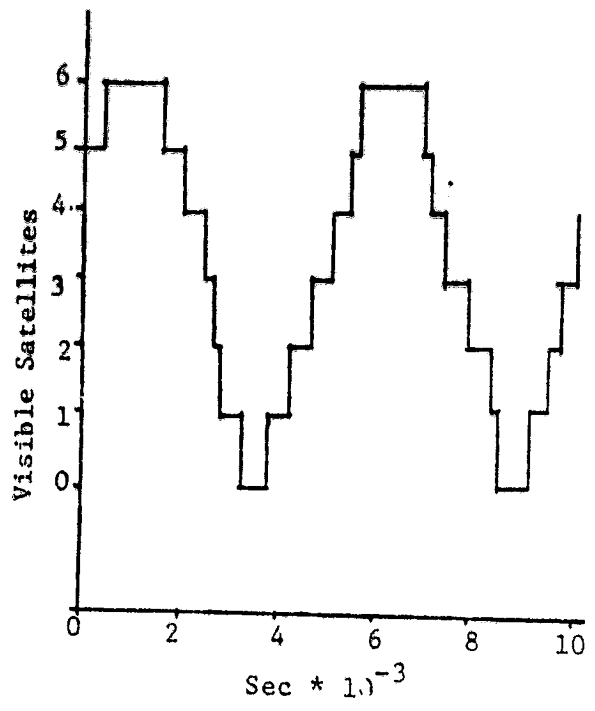
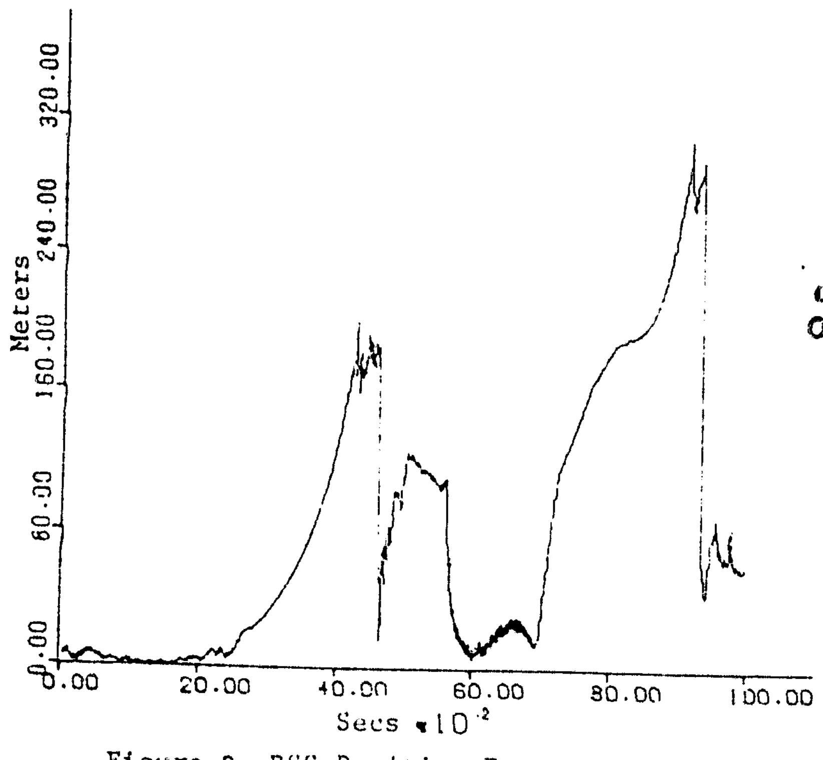


Figure 1. GPS Satellite Visibility vs. Time



ORIGINAL PAGE IS OF POOR QUALITY

Figure 2. RSS Position Error vs. Time

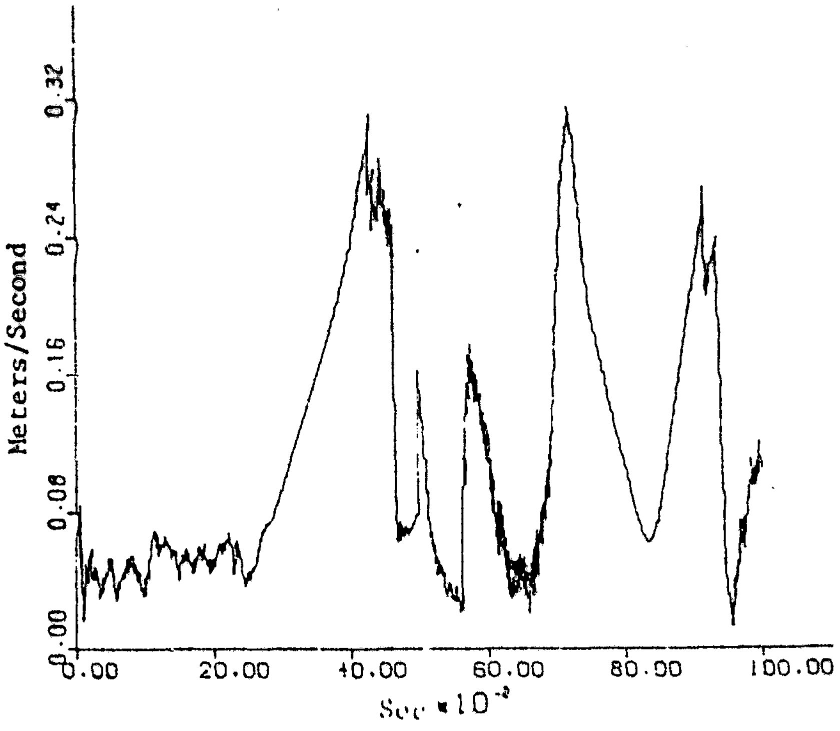


Figure 3. RSS Velocity Error vs. Time

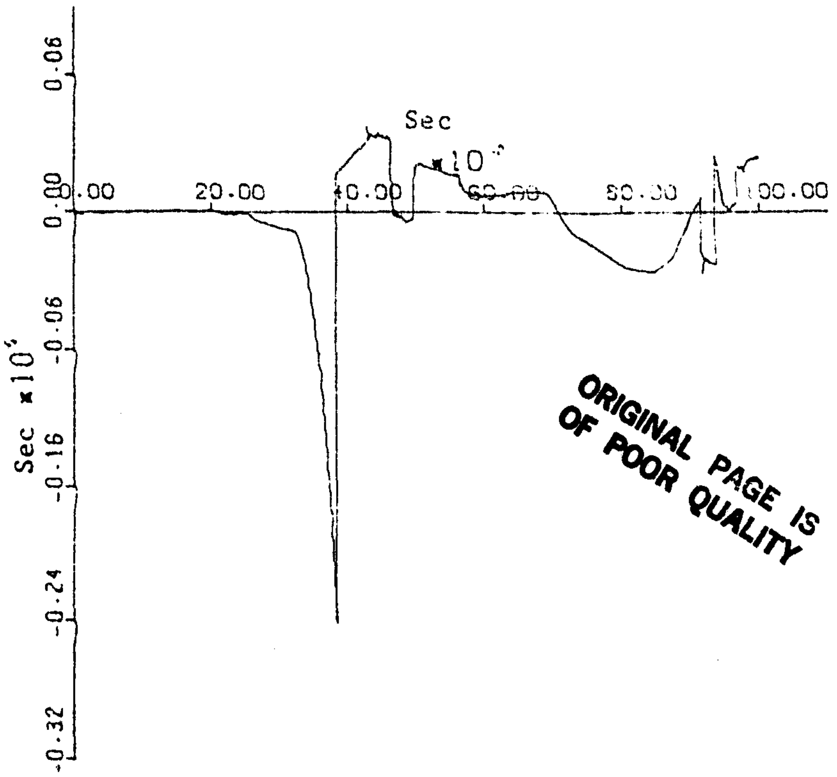


Figure 4. System Time Error vs. Time

ORIGINAL PAGE IS
OF POOR QUALITY

APPENDIX C

Sponsored by—
American Institute of Aeronautics and Astronautics (AIAA)
American Astronautical Society (AAS)

78-1428

A Triangular Covariance Factorization for Sequential Filtering Algorithms

B. D. Tapley and J. G. Peters, *University of Texas, Austin, Texas*

AIAA/AAS ASTRODYNAMICS CONFERENCE

Palo Alto, Calif./August 7-9, 1978

analytical trapezoid rule integration. The error introduced by this approximation can be neglected if the time interval $(t_{k+1}-t_k)$ is small enough. The effect of error accumulated over long prediction intervals, during loss of tracking or data drop-outs, must be considered to ascertain the accuracy of this approximation. The approximation does eliminate the computation of the n × n quadrature, however.

The matrix multiplication $\Phi\hat{W}_k$ combined with the creation of the augmented matrix W_{k+1}, destroys the triangularity of the square-root covariance matrix. The modified Gram-Schmidt orthogonalization is required to retriangularize the UD factors. The orthogonalization procedure is required at the time of each measurement update. The added computational burden of the orthogonalization would be eliminated if the square-root covariance matrix were propagated without the destruction of its triangularity.

In this investigation, a modification is proposed which allows the integration of the continuous state-error covariance differential equations in square-root form. The new algorithm can be combined with a triangular measurement update algorithm to obtain a complete square-root estimation algorithm for which matrix retriangularizations are avoided. Simultaneously, the effects of state process noise are included without approximation. The derivation follows the approach found in [8], but the $P=UDU^T$ decomposition is used rather than $P=WW^T$.

Derivation of the Square-Root Propagation <u>Equations in Triangular Form</u>

The differential equation for propagating the state error covariance matrix can be expressed as

$$\dot{\bar{P}} = A\bar{P} + \bar{P}A^T + Q\ , \qquad (1)$$

where

$\bar{P}$ ≡ state error covariance matrix
A ≡ linearized dynamics matrix
Q ≡ state noise covariance matrix.

If the following definitions are used:

$$\bar{P} \equiv \bar{U}\bar{D}\bar{U}^T \quad ; \quad \tilde{Q} \equiv Q/2 \qquad (2)$$

and, if the first part of (2) is differentiated with respect to time and substituted into (1), the results can be rearranged to form

$$(\dot{\bar{U}}\bar{D} + \frac{\bar{U}\dot{\bar{D}}}{2} - \bar{Q}\bar{U}^{-T} - A\bar{U}\bar{D})\bar{U}^T$$
$$+ \bar{U}(\bar{D}\dot{\bar{U}}^T + \frac{\dot{\bar{D}}\bar{U}^T}{2} - \bar{U}^{-1}\bar{Q}^T - \bar{D}\bar{U}^TA^T) = 0 \qquad (3)$$

Noting that the first term of (3) is the transpose of the second term, the following definition is made:

$$C(t) \equiv (\dot{\bar{U}}\bar{D} + \frac{\bar{U}\dot{\bar{D}}}{2} - \tilde{Q}\bar{U}^{-T} - A\bar{U}\bar{D}) \qquad (4)$$

With this definition, (3) can be expressed as

$$C(t) + C^T(t) = 0 \qquad (5)$$

Relation (5) is true if C(t) is the null matrix or, more generally, if it is skew symmetric. The skew symmetric property will become important later in the derivation.

At this point, (4) can be simplified by selectively carrying out the multiplication of the $-\tilde{Q}\bar{U}^{-T}$ term by $\bar{U}^T$ yielding, after terms are rearranged:

$$(\dot{\bar{U}}\bar{D} + \frac{\bar{U}\dot{\bar{D}}}{2} - A\bar{U}\bar{D})\bar{U}^T = \tilde{Q} + C(t) \equiv \tilde{C}(t) \qquad (6)$$

The problem now is to find the elements of C(t) to maintain $\dot{\bar{U}}$ in triangular form. (This derivation assumes $\bar{U}$ is lower triangular and $\bar{D}$ is diagonal, although an algorithm for an upper triangular $\bar{U}$ can be obtained as easily.) The following definitions are made to facilitate analysis of the solution to (6):

$$T \equiv A\bar{U}\bar{D} \quad ; \quad M \equiv \dot{\bar{U}}\bar{D} + \frac{\bar{U}\dot{\bar{D}}}{2} - T \qquad (7)$$

With these definitions, (6) is expressed as

$$M\bar{U}^T = \tilde{C} = \tilde{Q} + C(t) \qquad (8)$$

Since $\dot{\bar{U}}$ and $\bar{U}$ in (6) are lower triangular, and since from (5), C(t) is skew symmetric, the following observations can be made regarding (8): There are n(n-1)/2 unknown elements in $\tilde{C}$. The products $\dot{\bar{U}}\bar{D}$ and $\bar{U}\dot{\bar{D}}$ are lower triangular creating n(n+1)/2 unknowns. Therefore, the n × n system of equations (8) have [n(n-1)/2 + n(n+1)/2] = n × n unknowns which can be determined uniquely.

An expansion of (8) into matrix elements indicates the method of solution. (In this expansion $\tilde{Q}$ is assumed to be a diagonal matrix with elements $\tilde{q}_{ii} = q_{ii}/2$, i=1,...,n.)

$$\begin{bmatrix} M_{11} & -T_{12} & \cdots & -T_{1n} \\ M_{21} & M_{22} & \cdots & -T_{2n} \\ \vdots & \vdots & & \vdots \\ M_{n1} & M_{n2} & \cdots & M_{nn} \end{bmatrix} \begin{bmatrix} 1 & \bar{U}_{21} & \cdots & \bar{U}_{n1} \\ \cdot & 1 & \cdots & \bar{U}_{n2} \\ \vdots & & & \vdots \\ 0 & & & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \tilde{q}_{11} & -C_{21} & \cdots & -C_{n1} \\ C_{21} & \tilde{q}_{22} & \cdots & -C_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ C_{n1} & C_{n2} & \cdots & \tilde{q}_{nn} \end{bmatrix} \qquad (9)$$

Each row of the upper triangular portion of the C matrix is determined as the product of the same numbered row of the M matrix with the columns of the $\bar{U}^T$ matrix. After an upper triangular row of $\tilde{C}$ is computed, the condition that $C_{ij} = -C_{ji}$ ($i=1,\ldots,n$; $j=1,\ldots,i-1$) is invoked to evaluate the corresponding lower triangular column of C. Then a column of the lower triangular elements of M can be evaluated. Once the M matrix elements are determined, the next row of the upper triangular $\tilde{C}$ elements is computed as is a column of $\dot{\bar{U}}$ and $\dot{\bar{D}}$ elements. This process is repeated until all $\dot{\bar{U}}$ and $\dot{\bar{D}}$ values are determined. The $\dot{\bar{U}}$ and $\dot{\bar{D}}$ elements are determined in the following manner. From (6) and (7) define

$$N \equiv M + T = \dot{\bar{U}}\bar{D} + \frac{\bar{U}\dot{\bar{D}}}{2} \tag{10}$$

The expansion of N in summation notation gives

$$M_{ij} + T_{ij} = \sum_{k=1}^{n} \dot{\bar{U}}_{ik}\bar{d}_{kj} + \sum_{k=1}^{n} \frac{\bar{U}_{ik}\dot{\bar{d}}_{kj}}{2}$$

$$i=1,\ldots,n \; ; \; j=1,\ldots,i \tag{11}$$

But, since $\bar{D}$ is diagonal, (11) becomes

$$M_{ij} + T_{ij} = \dot{\bar{U}}_{ij}\bar{d}_{jj} + \frac{\bar{U}_{ij}\dot{\bar{d}}_{jj}}{2}$$

$$i=1,\ldots,n \; ; \; j=1,\ldots,i \tag{12}$$

For $i=j$, $\dot{\bar{U}}_{ij} \equiv 0$ and $\bar{U}_{ij} \equiv 1$. Therefore, (12) becomes

$$\dot{\bar{d}}_{ii} = 2(M_{ii} + T_{ii}) \qquad i=1,\ldots,n \tag{13}$$

For $i > j$, (12) is rearranged to obtain the differential equation

$$\dot{\bar{U}}_{ij} = (M_{ij} + T_{ij} - \frac{\bar{U}_{ij}\dot{\bar{d}}_{jj}}{2})/\bar{d}_{jj}$$

$$i=1,\ldots,n \; ; \; j=1,\ldots,i-1 \tag{14}$$

Equations (13) and (14) are the forms of the differential equations to be employed in the derivative routine of a numerical integrator. The elements T_{ij} and M_{ij} are computed as outlined previously, and formalized in the following algorithm.

Triangular Square-Root Propagation Algorithm

Given the elements of the square-root state error covariance in lower triangular $\bar{U}\bar{D}$ form, $\bar{Q} \equiv Q/2$, and A(t), the differential equations $\dot{\bar{U}}_{ij}$ and $\dot{\bar{d}}_{ii}$ can be computed as follows:

(1) $T_{ij} = \sum_{k=1}^{n} A_{ik}\bar{U}_{kj}\bar{d}_{jj} \qquad i=1,\ldots,n \; ; \; j=1,\ldots,n$

(2) $C_{ij} = \sum_{k=1}^{i} M_{ik}\bar{U}_{jk} - \sum_{k=i+1}^{j} T_{ik}\bar{U}_{jk}$

$i=1,\ldots,n \; ; \; j=i+1,\ldots,n$

(3) $M_{ii} = \bar{q}_{ii} - \sum_{k=1}^{i-1} M_{ik}\bar{U}_{ik} \qquad i=1,\ldots,n$

(4) $M_{ij} = -C_{ji} - \sum_{k=1}^{j-1} M_{ik}\bar{U}_{jk}$

$i=1,\ldots,n \; ; \; j=1,\ldots,i-1$

(5) $\dot{\bar{d}}_{ii} = 2(M_{ii} + T_{ii}) \qquad i=1,\ldots,n$

(6) $\dot{\bar{U}}_{ij} = (M_{ij} + T_{ij} - \frac{\bar{U}_{ij}\dot{\bar{d}}_{jj}}{2})/d_{jj}$

$i=1,\ldots,n \; ; \; j=1,\ldots,i-1$

ORIGINAL PAGE IS
OF POOR QUALITY

Preliminary Numerical Results

The two methods for time propagation of the square-root covariance matrix in the UDU algorithm were evaluated to determine the relative computational speed and integration accuracy. The test problem chosen was a planar Keplerian orbit at an altitude of 800 km above the earth. This problem was chosen for the following reasons:

1. The problem's simplicity allows for quick implementation and computation time.
2. The existence of an analytical solution gives a standard for measuring the accuracy of the numerical integration.
3. The orbit chosen approximates those of proposed operational satellites (such as LANDSAT-D) which will use square-root algorithms in onboard navigation computers.

The two methods were evaluated by integrating the state and square-root covariance matrix equations for one revolution.* Position-velocity components, state covariance, and integration times were tabulated at the end of the revolution as well as at 1/4, 1/2, and 3/4 points in the orbit. The integration was performed with a Runge-Kutta algorithm of fourth order (RK2(4)). The time comparisons were made with variable stepsize integration at relative error tolerances of 10^{-7}, $10^{-[illegible]}$, and 10^{-10}. Comparisons were also made for three fixed step sizes.

The standard for measuring the accuracy of the numerical integration of the covariance matrix was generated by integrating the state and square-root covariance equations at a tight tolerance with a high-order multi-step integrator. Covariance accuracy was measured as the relative error in the magnitude of the diagonal position and velocity covariance elements. The error was measured relative to the standard solution described at the beginning of the paragraph.

* Numerical comparisons were performed on a CDC 6600/6400 computer using single precision arithmetic. Integration times and accuracies will vary with different machines.

In the actual operation of the time propagation algorithm with measurement updates, the frequency of measurements has an effect on the computation time required for propagation. For the algorithms tested here, this is true for two reasons. First, a high measurement rate may reduce the allowable stepsize that the integrator may take. Additionaly, whenever the update algorithm requires a re triangularization at the time of each measurement, a high measurement rate requires that this procedure be performed more often, increasing the computation time. For reasons of simplicity, the measurement rate effect was simulated by the time interval value specified for each call to the numerical integrator. The simulations were run with different integration intervals to simulate the effects of different measurement rates.

The initial conditions for numerical integrations were:

State:

$x_1 = 0.$ m ; $\dot{x}_1 = x_3 = 7.4518$[illegible] $\times 10^{3}$ mps

$x_2 = 7178.165$ m ; $\dot{x}_2 = x_4 = 0.$ mps

Covariance (initially diagonal):

$P_{x_1x_1} = 9. \times 10^{6}$; $P_{x_3x_3} = 1.2 \times 10^{1}$

$P_{x_2x_2} = 9. \times 10^{6}$; $P_{x_4x_4} = 1.2 \times 10^{1}$

where $UDU^T(t_0) = P(t_0)$.

The nominal state covariance noise was:

$Q_{x_1x_1} = Q_{x_2x_2} = 0.$

$Q_{x_3x_3} = Q_{x_4x_4} = 5. \times 10^{-}$[illegible]

Tables 1 through 4 show test results at times at 4 points in the orbit and error in the total variance for the two time update techniques. These quantities are tabulated as functions of integration tolerance and simulated measurement rate. The transition-matrix propagation method is designated $(\dot{\Phi})$, and the square-root covariance integration method is denoted by $(\dot{U},\dot{D})$. All integration times are tabulated in milliseconds.

Discussion of Numerical Results

Based on the tabulated data shown in Tables 1 through 4, the $(\dot{\Phi})$ time update algorithm is superior to the $(\dot{U},\dot{D})$ method in computation efficiency for this particular test problem. Specifically, the following trends in relative algorithm performance have been deduced from the results:

1. For all measurement rates tested, the $(\dot{\Phi})$ algorithm operates faster for a given integration tolerance. The difference in computation speed increases nonlinearly as integration tolerance becomes tighter.

2. For the fixed step sizes and for the measurement rates tested, the $(\dot{\Phi})$ algorithm offers faster integration time and higher accuracy. Each derivative routine call is cheaper for the $(\dot{\Phi})$ code (approximately 1.[illegible] versus 2.[illegible] ms).

3. The $(\dot{U},\dot{D})$ algorithm gives a higher integration accuracy at tight tolerances, but with a severe penalty in integration time. The most accurate solution results with the $(\dot{U},\dot{D})$ equation being integrated at a single step relative error tolerance of 10^{-}[illegible].

4. As measurement rates increase, the difference in relative performance of the two update methods becomes smaller. The major remaining differences in integration time occur at tight tolerances, and differences in accuracy occur at loose tolerances, and in fixed-step cases.

The margin of difference in the performance of the two algorithms was unexpected. The cause of the difference is, evidently, the significantly smaller step size required in the $(\dot{U},\dot{D})$ integration — a step-size so small that the method's advantage of integrating fewer equations is lost. This indicates that the $(\dot{U},\dot{D})$ update would be more competitive in a problem where an external factor, such as high data rate, constrains the integration step size. Such a constraint would not allow the $(\dot{\Phi})$ method to use its ability to take larger integration steps at a given tolerance, and could force it to take steps of the order of magnitude of those used by the $(\dot{U},\dot{D})$ code. Also of note is that the higher the measurement rate, the more often the $(\dot{\Phi})$ algorithm is forced to perform matrix orthogonalizations, which increase computation time. The numerical results in the tables indicate that the large disparity in comparative performance decreases as the measurement rate increases. It seems, therefore, that one direction of search for an application where the $(\dot{U},\dot{D})$ method would be more efficient is toward systems with measurement rates high enough to constrain integration step size. When integration step size can be freely chosen by accuracy criteria, the $(\dot{\Phi})$ method is the more efficient.

Although small integration steps appear to be the major deterrent to efficient operation of the $(\dot{U},\dot{D})$ method in this problem, three other factors affect integration speed and should be considered in further comparative efficiency studies.

1. A perceived advantage of the $(\dot{U},\dot{D})$ update is its requirement for numerically integrating $n(n+1)/2$ equations only, as opposed to the $n \times n$ equations integrated in the $(\dot{\Phi})$ algorithm. For the test problem where $n=4$, the numbers of equations to be integrated are 10 and 16, respectively. Derivative routine calls still require less computation time for the $(\dot{\Phi})$ method. Perhaps a larger state vector would be necessary for the $(\dot{U},\dot{D})$ to show a time savings due to its smaller integration vector size. This result will be highly problem-dependent, but tests with larger state vectors would be enlightening.

2. The $(\dot{\Phi})$ algorithm presented here approximates the process noise with an analytic trapezoid rule integration. This approximation removes an $n \times n$ quadrature from the $(\dot{\Phi})$ algorithm. The $(\dot{U},\dot{D})$

ORIGINAL PAGE IS
OF POOR QUALITY

method accounts for process noise exactly. In the simulations, the process noise approximation had no discernable affect on covariance accuracy. Prediction intervals were small. Intuitively, the process noise approximation will become less accurate as the measurement rate decreases, or if there is a long prediction interval with no measurements. The degree of accuracy degradation that would occur in an actual filtering problem with long prediction intervals has not been addressed here, but merits study. If the approximation were not to give the desired accuracy, an n × n quadrature would have to be added to the ($\dot{\Phi}$) algorithm. The associated increase in computational burden might improve the competitiveness of the ($\dot{U},\dot{D}$) update.

3. For orbit problems with higher order forcing functions than that tested, the acceptable stepsize will become smaller as the forcing function becomes less smooth. The percentage reduction in step size to accomodate the more irregular forcing function may not be the same for the two algorithms, changing their relative efficiencies. This problem is coupled with that of the effect of measurement rate on step size. The combination of the two factors determines allowable step size and, therefore, integration speed.

Table 1
Integration Times & Covariance Error
25-Second Data Rate

Rev		10^{-10}	10^{-6}	10^{-2}	Fixed
		TOLERANCE			
1/4	$\dot{U},\dot{D}$	58858	6240	955	-
	$\dot{\Phi}$	9548	1246	751	-
1/2	$\dot{U},\dot{D}$	73211	7698	1726	-
	$\dot{\Phi}$	19063	2468	1496	-
3/4	$\dot{U},\dot{D}$	106970	11535	2495	-
	$\dot{\Phi}$	28590	3671	2229	-
1	$\dot{U},\dot{D}$	147757	15503	3260	-
	$\dot{\Phi}$	38078	4895	2942	-
COVARIANCE ERROR (1 REVOLUTION)					
Pos	$\dot{U},\dot{D}$	0	9.74×10^{-8}	5.14×10^{-3}	-
Vel	$\dot{U},\dot{D}$	2.32×10^{-11}	9.27×10^{-8}	5.34×10^{-3}	-
Pos	$\dot{\Phi}$	1.62×10^{-6}	1.61×10^{-6}	1.51×10^{-6}	-
Vel	$\dot{\Phi}$	1.46×10^{-6}	1.46×10^{-6}	1.36×10^{-6}	-

Table 3
Integration Time & Covariance Error
5-Second Data Rate

Rev		10^{-10}	10^{-6}	10^{-2}	Fixed
		TOLERANCE			
1/4	$\dot{U},\dot{D}$	61281	7923	3812	3688
	$\dot{\Phi}$	11237	3619	3599	3410
1/2	$\dot{U},\dot{D}$	76846	11728	7576	7409
	$\dot{\Phi}$	22512	7251	7203	6860
3/4	$\dot{U},\dot{D}$	112596	16886	11404	11103
	$\dot{\Phi}$	34125	10871	10836	10327
1	$\dot{U},\dot{D}$	153250	22182	15224	14810
	$\dot{\Phi}$	45457	14487	14428	13668
COVARIANCE ERROR (1 REVOLUTION)					
Pos	$\dot{U},\dot{D}$	0	7.25×10^{-8}	2.73×10^{-3}	4.29×10^{-1}
Vel	$\dot{U},\dot{D}$	2.32×10^{-11}	7.64×10^{-8}	2.73×10^{-3}	4.29×10^{-1}
Pos	$\dot{\Phi}$	6.25×10^{-8}	6.25×10^{-8}	6.25×10^{-8}	6.25×10^{-8}
Vel	$\dot{\Phi}$	5.79×10^{-8}	5.79×10^{-8}	5.70×10^{-8}	5.79×10^{-8}

Table 2
Integration Times & Covariance Error
10-Second Data Rate

Rev		10^{-10}	10^{-6}	10^{-2}	Fixed
		TOLERANCE			
1/4	$\dot{U},\dot{D}$	59640	6831	2075	-
	$\dot{\Phi}$	10232	1817	1820	1709
1/2	$\dot{U},\dot{D}$	74635	8723	3980	-
	$\dot{\Phi}$	20515	3594	3462	3441
3/4	$\dot{U},\dot{D}$	109016	12917	5904	-
	$\dot{\Phi}$	30854	5370	5391	5130
1	$\dot{U},\dot{D}$	148035	17488	7816	-
	$\dot{\Phi}$	41090	7146	7170	6778
COVARIANCE ERROR (1 REVOLUTION)					
Pos	$\dot{U},\dot{D}$	0	8.49×10^{-8}	2.99×10^{-3}	-
Vel	$\dot{U},\dot{D}$	0	8.34×10^{-8}	2.99×10^{-3}	-
Pos	$\dot{\Phi}$	2.57×10^{-7}	2.55×10^{-7}	2.55×10^{-7}	2.55×10^{-7}
Vel	$\dot{\Phi}$	2.34×10^{-7}	2.32×10^{-7}	2.32×10^{-7}	2.32×10^{-7}

Table 4
Integration Times & Covariance Error
2-Second Data Rate

Rev		10^{-10}	10^{-6}	10^{-2}	Fixed
		TOLERANCE			
1/4	$\dot{U},\dot{D}$	63370	12638	9534	9238
	$\dot{\Phi}$	15439	8857	8916	8403
1/2	$\dot{U},\dot{D}$	82088	22013	19025	18553
	$\dot{\Phi}$	30918	17701	17872	17083
3/4	$\dot{U},\dot{D}$	119973	31768	28456	27711
	$\dot{\Phi}$	46532	26648	26883	25633
1	$\dot{U},\dot{D}$	163553	41182	37966	36741
	$\dot{\Phi}$	62087	35562	35876	34187
COVARIANCE ERROR (1 REVOLUTION)					
Pos	$\dot{U},\dot{D}$	0	5.25×10^{-8}	3.75×10^{-4}	1.87×10^{-3}
Vel	$\dot{U},\dot{D}$	2.32×10^{-11}	5.79×10^{-8}	3.74×10^{-4}	1.87×10^{-3}
Pos	$\dot{\Phi}$	1.00×10^{-8}	1.00×10^{-8}	1.00×10^{-8}	1.00×10^{-8}
Vel	$\dot{\Phi}$	9.27×10^{-9}	9.27×10^{-9}	9.27×10^{-9}	9.27×10^{-9}

Conclusions

Based on the numerical results obtained for the example problem considered in the investigation, it is concluded that:

- the ($\dot{\Phi}$) algorithm is the more efficient in terms of the integration time required to achieve a specified computation accuracy
- the ($\dot{U}, \dot{D}$) method can achieve the highest computation accuracy but with a heavy penalty in integration time
- the ($\dot{U}, \dot{D}$) algorithm has an advantage in terms of core storage, as it requires an integration vector of n(n+1)/2 elements, as opposed to the $n \times n$ elements required in the ($\dot{\Phi}$) method
- the disparity in efficiency between the two methods becomes less as the simulated measurement rate is increased.

The performance of the ($\dot{U}, \dot{D}$) propagation algorithm for this test problem was not what had been desired. Yet, results indicate there are at least two situations where the ($\dot{U}, \dot{D}$) method may offer improved efficiencies. These are in systems with measurement rates high enough to constrain integration stepsize, and during large prediction intervals without measurements, where analytic approximation of the process noise may not be adequate. These are the areas where future research will continue.

Acknowledgements

The authors wish to thank John Lundberg and Paul Wing of the Department of Aerospace Engineering and Engineering Mechanics for their respective efforts in preparing the simulation software and in typing manuscripts of this paper.

APPENDIX

The measurement update algorithm for the UDU factorization [7] has the following form. Using the observation $Y_{k+1} = G(X_{k+1}, t_{k+1})$ calculate:

$$\left.\begin{aligned} H_{k+1} &= [\partial G(\bar{X}_{k+1}, t_{k+1})/\partial \bar{X}_{k+1}] \\ \tilde{F}_i &= H_i + \sum_{k=i+1}^{n} H_k \bar{U}_{ki} \\ V_i &= \bar{d}_i \tilde{F}_i \end{aligned}\right\} \quad i = 1 \rightarrow n$$

Set $\beta_{n+1} = R_{k+1}$ (where R_{k+1} is the measurement noise) and calculate:

$$\beta_i = \beta_{i+1} + V_i F_i \qquad i = n \rightarrow 1$$

$$\left.\begin{aligned} \hat{d}_i &= \bar{d}_i \beta_{i+1}/\beta_i \\ P_j &= \tilde{F}_j/\beta_{j+1} \\ B_{ij} &= V_i + \sum_{k=j+1}^{i-1} \bar{U}_{ik} V_k \\ \alpha &= \beta_1 \end{aligned}\right\} \quad i = 1 \rightarrow n$$

Update square-root covariance matrix:

$$\hat{U}_{ij} = \bar{U}_{ij} - B_{ij} P_j \qquad , i = 2 \rightarrow n \; ; \; j = 1 \rightarrow i-1$$

Calculate:

$$K_i = V_i + \sum_{j=1}^{i-1} \bar{U}_{ij} V_j \qquad , i = 1 \rightarrow n$$

$$v_{k+1} = (Y_{k+1} - G(\bar{X}_{k+1}, t_{k+1})) \text{ [illegible] } \bar{X}_{k+1}$$

Update state vector:

$$\hat{X}_i = \bar{X}_i + K_i v_{k+1}/\alpha \qquad i = 1 \rightarrow n$$

References

1. R. H. Battin, Astronautical Guidance, McGraw-Hill, 1964, pp 338-339.
2. J.F. Bellantoni and K.W. Dodge, "A Square Root Formulation of the Kalman-Schmidt Filter," AIAA J, Vol 5 pp 1309-1314, July 1967.
3. P.G. Kaminsky, A.E. Bryson, and S. Schmidt, "Discrete Square Root Filtering: A Survey of Current Techniques," IEEE Trans. Automat. Contr. Vol AC-16, pp 727-736, December 1971.
4. A. Andrews, "A Square Root Formulation of the Kalman Covariance Equations," AIAA J., Vol 11, pp 1165-1166, June 1968.
5. N.A. Carlson, "Fast Triangular Formulation of the Square Root Filter," AIAA J., Vol 11, pp 1259-1265, September 1973.
6. W.M. Gentleman, "Least Squares Computations by Givens Transformations without Square Roots," J. Inst. Math. Appl., Vol 12, pp 329-336, 1973.
7. G.J. Bierman, Factorization Methods for Discrete Sequential Estimation, Academic Press, 1976.
8. B.D. Tapley and C.Y. Choe, "An Algorithm for Propagating the Square-Root Covariance Matrix in Triangular Form," IEEE Trans. Automat. Contr., Vol 21, No 1, pp 122-123, February 1976.