CHAPTER 7

Square root filtering

7.1 INTRODUCTION

The two previous chapters discussed the Kalman filter in substantial detail. Although this is the optimal solution to the filtering problem posed in Section 5.2 (with respect to essentially all viable optimality criteria), the algorithm itself is prone to serious numerical difficulties. As noted in Section 5.6, measurement updating of the covariance matrix requires rather long wordlength to maintain acceptable numerical accuracy: for onboard computers, double precision computations are usually required. In fact, without double precision arithmetic, these numerical characteristics can easily become the dominant error source corrupting estimation precision, and unfortunately an error source usually not included in designers' error budgets.

The difficulties encountered in converting a tuned Kalman filter on a long wordlength, large computer system used for engineering design to an effective algorithm on a smaller wordlength online computer are well documented [7, 22]. For instance, although it is theoretically impossible for the covariance matrix to have negative eigenvalues, such a condition can, and often does, result due to numerical computation using finite wordlength, especially when (1) the measurements are very accurate [eigenvalues of $\mathbf{R}(t_i)$ are small relative to those of $\mathbf{P}(t_i^-)$, this being accentuated by large eigenvalues in $\mathbf{P}_0$] or (2) a linear combination of state vector components is known with great precision while other combinations are nearly unobservable (i.e., there is a large range of magnitudes of state covariance eigenvalues). Such a condition can lead to subsequent divergence or total failure of the recursion. On close inspection, even Kalman filters that maintain adequate estimation accuracy exhibit instances of negative covariance diagonal terms [7].

To circumvent these problems in numerics inherent to the Kalman filter algorithm, alternate recursion relationships [24] have been developed to propagate and update a state estimate and error covariance square root or inverse covariance square root instead of the covariance or its inverse themselves. Although equivalent algebraically to the conventional Kalman filter recursion, these *square root filters* exhibit improved numerical precision and stability, particularly in ill-conditioned problems (i.e., the cases described that yield erroneous results due to finite wordlength). The square root approach can yield twice the effective precision of the conventional filter in ill-conditioned problems. In other words, the same precision can be achieved with approximately half the wordlength. Moreover, this method is completely successful in maintaining the positive semidefiniteness of the error covariance.

These excellent numerical characteristics, combined with modest additional computation cycle time and memory storage requirements, make the square root filter approach a viable alternative to the conventional filter in many applications, especially when computer wordlength is limited or the estimation problem is ill conditioned. The formulation of the square root filter for the case of no dynamic noise is especially attractive because of its computational simplicity, and its outstanding numerical characteristics led to its implementation in the Apollo spacecraft navigation filters.

A number of practitioners have argued, with considerable logic, that square root filters should *always* be adopted in preference to the standard Kalman filter recursion, rather than switching to these more accurate and stable algorithms when and if numerical problems occur [7]. Even though Kalman algorithms can be made to work in most applications, by using double precision mathematics or scaling variables to reduce dynamic range or employing ad hoc modifications, numerics degrade performance from that achievable by numerically better conditioned recursions. Recent investigations tend to support an approach of designing and tuning an optimal filter by the methods of the two previous chapters, ignoring the errors caused by numerics, but then implementing the square root equivalent for online operation. Nevertheless, one can expect conventional Kalman algorithms to be applied rather extensively as well.

Section 7.2 introduces the concept of matrix square roots, and then Section 7.3 develops the initially designed and simplest covariance square root filter, applicable to the case of no dynamic driving noise and scalar measurements. The succeeding two sections generalize these results, first incorporating vector-valued measurements and then allowing dynamic driving noise. In Section 7.6, the square root counterpart to the inverse covariance formulation of the optimal filter is considered. Although it is not actually a square root filter, the U–D covariance factorization filter is very closely related to square root filtering, and it is depicted in Section 7.7. Finally, Section 7.8 presents the tradeoff of

numerical advantages and increased computational burden of the square root filters.

7.2 MATRIX SQUARE ROOTS

Let $\mathbf{A}$ be an n-by-n, symmetric, positive semidefinite matrix. Then there exists at least one n-by-n "square root" matrix, denoted as $\sqrt{\mathbf{A}}$, such that

$$\sqrt{\mathbf{A}}\sqrt{\mathbf{A}}^{\mathrm{T}} = \mathbf{A} \tag{7-1}$$

In fact, there are many matrices $\sqrt{\mathbf{A}}$ which satisfy (7-1) in general. The essential idea of square root filters is to replace the recursion for the error covariance $\mathbf{P}$ with a recursion for its square root, $\sqrt{\mathbf{P}}$, and to compute the state estimate using an optimal gain calculated in terms of $\sqrt{\mathbf{P}}$ instead of $\mathbf{P}$ itself. To motivate this, consider the scalar case: if dynamic range numerical precision problems are encountered in a filter that propagates the variance $P = \sigma^2$, the problem can be reduced by expressing the result in terms of the standard deviation σ since the dynamic range will be effectively reduced to half the original range. This basic idea can be generalized to the vector case by defining the state error covariance square roots, before and after measurement incorporation at time t_i, as $\mathbf{S}(t_i^-)$ and $\mathbf{S}(t_i^+)$ respectively, via:

$$\mathbf{S}(t_i^-)\mathbf{S}^{\mathrm{T}}(t_i^-) \triangleq \mathbf{P}(t_i^-) \tag{7-2}$$

$$\mathbf{S}(t_i^+)\mathbf{S}^{\mathrm{T}}(t_i^+) \triangleq \mathbf{P}(t_i^+) \tag{7-3}$$

Similarly, define the square root of the covariances depicting the strengths of discrete-time white Gaussian noises $\mathbf{w}_d(\cdot,\cdot)$ and $\mathbf{v}(\cdot,\cdot)$ as

$$\mathbf{W}_d(t_i)\mathbf{W}_d^{\mathrm{T}}(t_i) \triangleq \mathbf{Q}_d(t_i) \triangleq E\{\mathbf{w}_d(t_i)\mathbf{w}_d^{\mathrm{T}}(t_i)\} \tag{7-4}$$

$$\mathbf{V}(t_i)\mathbf{V}^{\mathrm{T}}(t_i) \triangleq \mathbf{R}(t_i) \triangleq E\{\mathbf{v}(t_i)\mathbf{v}^{\mathrm{T}}(t_i)\} \tag{7-5}$$

The covariance square roots are *not uniquely* defined by (7-2)–(7-5), and square root filters can be formulated in terms of general matrix square roots. One means of exploiting this fact is to develop algorithms which maintain a particularly attractive square root form, namely an upper or lower triangular matrix (with all zeros below or above the main diagonal, respectively), thereby requiring computation and storage of only $n(n+1)/2$ instead of n^2 scalar variables.

This lack of uniqueness does not cause difficulties in converting from a problem description in terms of initial $\mathbf{P}_0$ and time histories of $\mathbf{Q}_d(t_i)$ and $\mathbf{R}(t_i)$ to corresponding $\mathbf{S}_0$, $\mathbf{W}_d(t_i)$, and $\mathbf{V}(t_i)$ values, as might first appear to be the case. The reason is that any positive semidefinite matrix can be factored into the product of a lower triangular matrix and its transpose by the *Cholesky decomposition* [13] algorithm. Although (7-1) does not uniquely define $\sqrt{\mathbf{A}}$, a *unique* Cholesky lower triangular square root $\sqrt[c]{\mathbf{A}}$ can be defined such that

$\sqrt[c]{\mathbf{A}}\sqrt[c]{\mathbf{A}}^{\mathrm{T}} = \mathbf{A}$:

$$\begin{bmatrix} \sqrt[c]{A_{11}} & 0 & \cdots & 0 \\ \sqrt[c]{A_{21}} & \sqrt[c]{A_{22}} & & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sqrt[c]{A_{n1}} & \sqrt[c]{A_{n2}} & \cdots & \sqrt[c]{A_{nn}} \end{bmatrix} \begin{bmatrix} \sqrt[c]{A_{11}} & \sqrt[c]{A_{21}} & \cdots & \sqrt[c]{A_{n1}} \\ 0 & \sqrt[c]{A_{22}} & \cdots & \sqrt[c]{A_{n2}} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt[c]{A_{nn}} \end{bmatrix}$$

$$= \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{12} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{bmatrix}$$

The elements of the Cholesky square root matrix can be generated sequentially, row by row, from the recursion: for $i = 1, 2, \ldots, n$, compute

$$\sqrt[c]{A_{ij}} = \begin{cases} (1/\sqrt[c]{A_{jj}})[A_{ij} - \sum_{k=1}^{j-1} \sqrt[c]{A_{ik}}\sqrt[c]{A_{jk}}] & j = 1, 2, \ldots, i-1 \\ (A_{ii} - \sum_{k=1}^{i-1} \sqrt[c]{A_{ik}^2})^{1/2} & j = i \\ 0 & j > i \end{cases} \tag{7-6}$$

Thus, $\mathbf{A}$ is scanned and $\sqrt[c]{\mathbf{A}}$ is generated in the order depicted in Fig. 7.1.

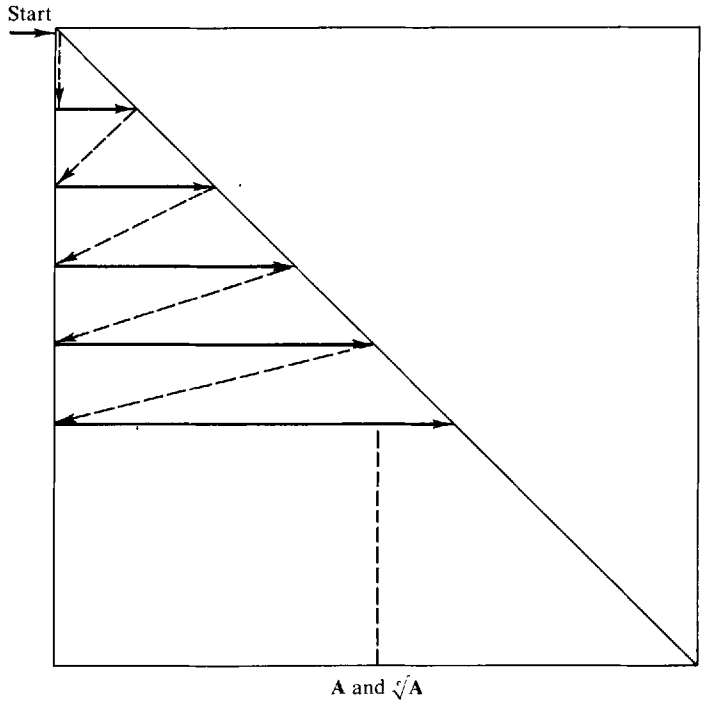

FIG. 7.1 Scanning of $\mathbf{A}$ and generation of $\sqrt[c]{\mathbf{A}}$.

EXAMPLE 7.1 Let $\mathbf{A}$ be given as

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 8 & 2 \\ 3 & 2 & 14 \end{bmatrix}$$

Then the elements of $\sqrt[c]{\mathbf{A}}$ are generated row by row as

$$\sqrt[c]{A_{11}} = \sqrt{1} = 1, \quad \sqrt[c]{A_{12}} = 0, \quad \sqrt[c]{A_{13}} = 0$$

$$\sqrt[c]{A_{21}} = (1/1)[2 - 0] = 2, \quad \sqrt[c]{A_{22}} = \sqrt{8 - 2^2} = 2, \quad \sqrt[c]{A_{23}} = 0$$

$$\sqrt[c]{A_{31}} = (1/1)[3 - 0] = 3, \quad \sqrt[c]{A_{32}} = (1/2)[2 - (3)(2)] = -2, \quad \sqrt[c]{A_{33}} = [14 - (3)^2 - (-2)^2]^{1/2} = 1$$

Note that the summation term in (7-6) for $j = i$ becomes effective only for $i > 1$ and involves the sum of squares of previously generated $\sqrt[c]{\mathbf{A}}$ elements in that row. Furthermore, the sum term for $j < i$ is effective only for $i > 1$, and involves terms from the jth and ith rows. From above, $\sqrt[c]{\mathbf{A}}$ is

$$\sqrt[c]{\mathbf{A}} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ 3 & -2 & 1 \end{bmatrix}$$

and it is readily seen that $\sqrt[c]{\mathbf{A}}\sqrt[c]{\mathbf{A}}^{\mathrm{T}} = \mathbf{A}$. ■

Later in the Carlson filter [11] of Section 7.5, we will have occasion to seek an *upper* triangular Cholesky square root $\sqrt[c']{\mathbf{A}}$ such that $\sqrt[c']{\mathbf{A}}\sqrt[c']{\mathbf{A}}^{\mathrm{T}} = \mathbf{A}$. Such

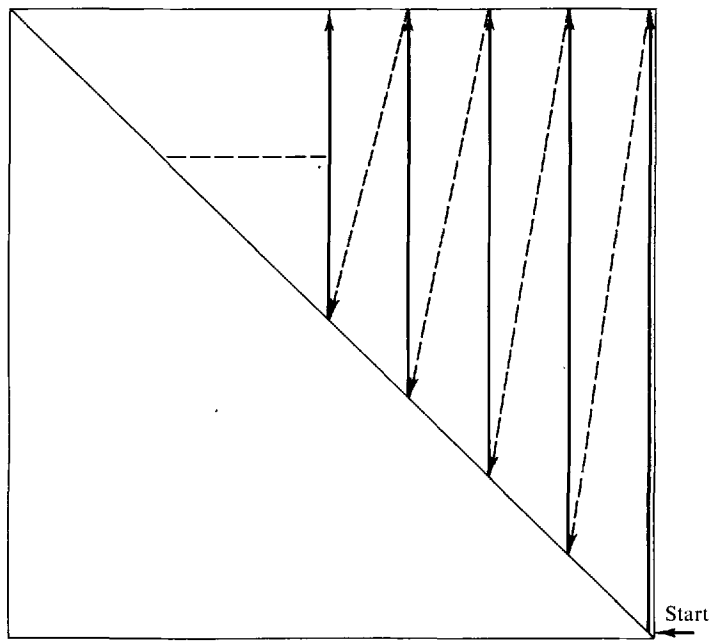

FIG. 7.2 Scanning of $\mathbf{A}$ and generation of $\sqrt[c]{\mathbf{A}}$.

a matrix can be found by operating (7-6) backwards, or specifically, for $j = n$, $n-1, \ldots, 1$, perform the following computations:

$$\sqrt[c]{A_{ij}} = \begin{cases} 0 & i > j \\ (A_{jj} - \sum_{k=j+1}^{n} \sqrt[c]{A_{jk}^2})^{1/2} & i = j \\ (1/\sqrt[c]{A_{jj}})[A_{ij} - \sum_{k=j+1}^{n} \sqrt[c]{A_{ik}}\sqrt[c]{A_{jk}}] & i = j-1, j-2, \ldots, 1 \end{cases} \tag{7-7}$$

$\sqrt[c]{\mathbf{A}}$ is thus generated column by column, from the nth column to the first and from the bottom to top within each column, as in Fig. 7.2.

7.3 COVARIANCE SQUARE ROOT FILTER FOR $\mathbf{Q}_d \equiv \mathbf{0}$

In 1964, Potter [26] developed a square root filter implementation for space navigation applications in which there was no dynamic driving noise in the system model, i.e., $\mathbf{Q}_d(t_i) \equiv \mathbf{0}$ for all time, motivated by restricted wordlength in the Apollo guidance computer. For this case, the time propagation in a conventional Kalman filter would be (neglecting control inputs):

$$\hat{\mathbf{x}}(t_{i+1}^-) = \mathbf{\Phi}(t_{i+1}, t_i)\hat{\mathbf{x}}(t_i^+) \tag{7-8a}$$

$$\mathbf{P}(t_{i+1}^-) = \mathbf{\Phi}(t_{i+1}, t_i)\mathbf{P}(t_i^+)\mathbf{\Phi}^{\mathrm{T}}(t_{i+1}, t_i) \tag{7-8b}$$

By letting $\mathbf{P}(t_i^+) = \mathbf{S}(t_i^+)\mathbf{S}^{\mathrm{T}}(t_i^+)$ and $\mathbf{P}(t_{i+1}^-) = \mathbf{S}(t_{i+1}^-)\mathbf{S}^{\mathrm{T}}(t_{i+1}^-)$, (7-8b) can be rewritten as

$$[\mathbf{S}(t_{i+1}^-)][\mathbf{S}^{\mathrm{T}}(t_{i+1}^-)] = [\mathbf{\Phi}(t_{i+1}, t_i)\mathbf{S}(t_i^+)][\mathbf{S}^{\mathrm{T}}(t_i^+)\mathbf{\Phi}^{\mathrm{T}}(t_{i+1}, t_i)]$$

From this it is evident that the appropriate time propagation relations for the square root filter would be

$$\hat{\mathbf{x}}(t_{i+1}^-) = \mathbf{\Phi}(t_{i+1}, t_i)\hat{\mathbf{x}}(t_i^+) \tag{7-9a}$$

$$\mathbf{S}(t_{i+1}^-) = \mathbf{\Phi}(t_{i+1}, t_i)\mathbf{S}(t_i^+) \tag{7-9b}$$

Because of his particular application, Potter confined his attention to scalar measurements. The covariance measurement update for this case is, since $\mathbf{H}(t_i)$ is a row vector,

$$\mathbf{P}(t_i^+) = \mathbf{P}(t_i^-) - \mathbf{P}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i)\frac{1}{[\mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i) + R(t_i)]}\mathbf{H}(t_i)\mathbf{P}(t_i^-) \tag{7-10}$$

Therefore, one can write this as

$$\mathbf{S}(t_i^+)\mathbf{S}^{\mathrm{T}}(t_i^+) = \mathbf{S}(t_i^-)[\mathbf{I} - b(t_i)\mathbf{a}(t_i)\mathbf{a}^{\mathrm{T}}(t_i)]\mathbf{S}^{\mathrm{T}}(t_i^-) \tag{7-11}$$

where by n-by-1 $\mathbf{a}(t_i)$ and scalar $b(t_i)$ are defined by

$$\mathbf{a}(t_i) = \mathbf{S}^{\mathrm{T}}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i) \tag{7-12a}$$

$$1/b(t_i) = \mathbf{a}^{\mathrm{T}}(t_i)\mathbf{a}(t_i) + R(t_i) \tag{7-12b}$$

Potter showed that the bracketed term in (7-11) can be factored into

$$[\mathbf{I} - b\mathbf{a}\mathbf{a}^{\mathrm{T}}] = [\mathbf{I} - b\gamma\mathbf{a}\mathbf{a}^{\mathrm{T}}][\mathbf{I} - b\gamma\mathbf{a}\mathbf{a}^{\mathrm{T}}]^{\mathrm{T}} \tag{7-13}$$

where γ is a scalar defined by

$$\gamma = 1/(1 + \sqrt{bR}) \tag{7-14}$$

Substituting this into (7-11) yields the covariance update as

$$\begin{aligned}\mathbf{S}(t_i^+) &= \mathbf{S}(t_i^-)[\mathbf{I} - b(t_i)\gamma(t_i)\mathbf{a}(t_i)\mathbf{a}^{\mathrm{T}}(t_i)] \\ &= \mathbf{S}(t_i^-) - b(t_i)\gamma(t_i)\mathbf{S}(t_i^-)\mathbf{a}(t_i)\mathbf{a}^{\mathrm{T}}(t_i)\end{aligned} \tag{7-15}$$

The state estimate measurement update is of the conventional form, but with the Kalman gain evaluated as $[b(t_i)\mathbf{S}(t_i^-)\mathbf{a}(t_i)]$. Thus, the measurement update becomes

$$\begin{aligned}\mathbf{a}(t_i) &= \mathbf{S}^{\mathrm{T}}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i) \\ b(t_i) &= 1/[\mathbf{a}^{\mathrm{T}}(t_i)\mathbf{a}(t_i) + R(t_i)] \\ \gamma(t_i) &= 1/[1 + \{b(t_i)R(t_i)\}^{1/2}] \\ \mathbf{K}(t_i) &= b(t_i)\mathbf{S}(t_i^-)\mathbf{a}(t_i) \\ \hat{\mathbf{x}}(t_i^+) &= \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)[z_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)] \\ \mathbf{S}(t_i^+) &= \mathbf{S}(t_i^-) - \gamma(t_i)\mathbf{K}(t_i)\mathbf{a}^{\mathrm{T}}(t_i)\end{aligned} \tag{7-16}$$

An equivalent form that is often employed is

$$\begin{aligned}\mathbf{a}(t_i) &= \mathbf{S}^{\mathrm{T}}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i) \\ \sigma(t_i) &= [\mathbf{a}^{\mathrm{T}}(t_i)\mathbf{a}(t_i) + R(t_i)]^{1/2} \\ \alpha(t_i) &= \sigma(t_i) + V(t_i) \\ \beta(t_i) &= 1/[\sigma(t_i)\alpha(t_i)] \\ \mathbf{g}(t_i) &= \beta(t_i)[\mathbf{S}(t_i^-)\mathbf{a}(t_i)] \\ \hat{\mathbf{x}}(t_i^+) &= \hat{\mathbf{x}}(t_i^-) + \mathbf{g}(t_i)\{[\alpha(t_i)/\sigma(t_i)][z_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)]\} \\ \mathbf{S}(t_i^+) &= \mathbf{S}(t_i^-) - \mathbf{g}(t_i)\mathbf{a}^{\mathrm{T}}(t_i)\end{aligned} \tag{7-17}$$

An example using both (7-16) and (7-17) will be presented in Section 7.5.

Note that, even if $\mathbf{S}(t_i^-)$ is lower triangular, $\mathbf{S}(t_i^+)$ will generally not be lower triangular when this update form is used. A method that preserves the lower triangular nature of the covariance square root will be discussed later.

7.4 VECTOR-VALUED MEASUREMENTS

The preceding section considered scalar measurement updates. Bellantoni and Dodge [3] extended these results to the vector measurement case by using eigenvalue decompositions, but their algorithm is inefficient for the typical

case in which the measurement vector dimension m is significantly less than the state dimension n. Andrews [2] also developed an update that processed an m-dimensional measurement vector in a single update, without requiring diagonalization:

$$\begin{aligned}
\mathbf{A}(t_i) &= \mathbf{S}^{\mathrm{T}}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i) \\
\boldsymbol{\Sigma}(t_i) &= \sqrt[c]{\mathbf{A}^{\mathrm{T}}(t_i)\mathbf{A}(t_i) + \mathbf{R}(t_i)} \\
\hat{\mathbf{x}}(t_i^+) &= \hat{\mathbf{x}}(t_i^-) + \mathbf{S}(t_i^-)\mathbf{A}(t_i)[\boldsymbol{\Sigma}^{-1}(t_i)]^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(t_i)[\mathbf{z}_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)] \\
\mathbf{S}(t_i^+) &= \mathbf{S}(t_i^-) - \mathbf{S}(t_i^-)\mathbf{A}(t_i)[\boldsymbol{\Sigma}^{-1}(t_i)]^{\mathrm{T}}[\boldsymbol{\Sigma}(t_i) + \mathbf{V}(t_i)]^{-1}\mathbf{A}^{\mathrm{T}}(t_i)
\end{aligned} \tag{7-18}$$

This can be seen to be a direct extension of (7-17), and it is more efficient computationally than the Bellantoni and Dodge algorithm. Processing a measurement entails a Cholesky decomposition of an m-by-m matrix to generate $\boldsymbol{\Sigma}(t_i)$, [the extension of $\sigma(t_i)$] and inversion of two triangular m-by-m matrices, $\boldsymbol{\Sigma}(t_i)$ and $[\boldsymbol{\Sigma}(t_i) + \mathbf{V}(t_i)]$.

For the covariance square root filter, the most efficient means of performing a vector measurement update is to employ the Potter scalar update, (7-16) or (7-17), repeatedly m times. An m-dimensional measurement vector $\mathbf{z}_i$ can *always* be processed equivalently as m scalar measurements. If $\mathbf{R}(t_i)$ is diagonal, the m components can be treated as independent measurements and processed sequentially. If $\mathbf{R}(t_i)$ is not diagonal, the procedure is somewhat more complicated. First the Cholesky decomposition of $\mathbf{R}(t_i)$ is computed, yielding $\sqrt[c]{\mathbf{R}(t_i)}$ as a lower triangular matrix. Then a transformation of variables is used to convert

$$\mathbf{z}(t_i) = \mathbf{H}(t_i)\mathbf{x}(t_i) + \mathbf{v}(t_i) \tag{7-19}$$

into

$$\mathbf{z}^*(t_i) = \mathbf{H}^*(t_i)\mathbf{x}(t_i) + \mathbf{v}^*(t_i) \tag{7-20}$$

where

$$\sqrt[c]{\mathbf{R}(t_i)}\mathbf{z}^*(t_i) = \mathbf{z}(t_i) \tag{7-21a}$$

$$\sqrt[c]{\mathbf{R}(t_i)}\mathbf{H}^*(t_i) = \mathbf{H}(t_i) \tag{7-21b}$$

$$\sqrt[c]{\mathbf{R}(t_i)}\mathbf{v}^*(t_i) = \mathbf{v}(t_i) \tag{7-21c}$$

Note that (7-21c) implies that $\mathbf{v}^*(\cdot,\cdot)$ is a unit power white Gaussian noise, i.e., $E\{\mathbf{v}^*(t_i)\mathbf{v}^{*\mathrm{T}}(t_i)\} = \mathbf{I}$, since

$$\begin{aligned}
E\{\mathbf{v}(t_i)\mathbf{v}^{\mathrm{T}}(t_i)\} &= \mathbf{R}(t_i) = E\{\sqrt[c]{\mathbf{R}(t_i)}\mathbf{v}^*(t_i)\mathbf{v}^{*\mathrm{T}}(t_i)\sqrt[c]{\mathbf{R}(t_i)}^{\mathrm{T}}\} \\
&= \sqrt[c]{\mathbf{R}(t_i)}E\{\mathbf{v}^*(t_i)\mathbf{v}^{*\mathrm{T}}(t_i)\}\sqrt[c]{\mathbf{R}(t_i)}^{\mathrm{T}}
\end{aligned}$$

Thus, the components of $\mathbf{z}^*(t_i, \omega_j) = \mathbf{z}_i^*$ can be processed one at a time sequentially. Moreover, (7-21a) and (7-21b) can be solved to yield $\mathbf{z}_i^*$ and $\mathbf{H}^*(t_i)$ by simple back substitution rather than matrix inversion, because $\sqrt[c]{\mathbf{R}(t_i)}$ is

lower triangular and thus the jth component of $\mathbf{z}_i^*$ is a linear combination of the first j components of $\mathbf{z}_i$.

EXAMPLE 7.2 Consider a four-state estimation problem with a three-dimensional vector measurement, with

$$\mathbf{H}(t_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}, \qquad \mathbf{R}(t_i) = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 8 & 2 \\ 3 & 2 & 14 \end{bmatrix}$$

Let the realized value of the measurement (7-19) be

$$\mathbf{z}(t_i, \omega_j) = \mathbf{z}_i = \begin{bmatrix} z_{i1} \\ z_{i2} \\ z_{i3} \end{bmatrix}$$

From Example 7.1, the Cholesky square root of $\mathbf{R}(t_i)$ is

$$\sqrt{\mathbf{R}(t_i)} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ 3 & -2 & 1 \end{bmatrix}$$

The problem is equivalent to one in which a measurement of the form of (7-20) is made available, in which $\mathbf{v}^*(t_i)$ is a unit power noise. Equation (7-21a) yields

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ 3 & -2 & 1 \end{bmatrix} \begin{bmatrix} z_{i1}^* \\ z_{i2}^* \\ z_{i3}^* \end{bmatrix} = \begin{bmatrix} z_{i1} \\ z_{i2} \\ z_{i3} \end{bmatrix} \Rightarrow \begin{aligned} z_{i1}^* &= z_{i1} \\ 2z_{i1}^* + 2z_{i2}^* &= z_{i2} \\ 3z_{i1}^* - 2z_{i2}^* + z_{i3}^* &= z_{i3} \end{aligned}$$

Back substitution yields, sequentially:

$$z_{i1}^* = z_{i1}, \qquad z_{i2}^* = \tfrac{1}{2}[z_{i2} - 2z_{i1}^*], \qquad z_{i3}^* = z_{i3} - 3z_{i1}^* + 2z_{i2}^*$$

Back substitution can also be used to solve

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ 3 & -2 & 1 \end{bmatrix} \begin{bmatrix} H_{11}^* & H_{12}^* & H_{13}^* & H_{14}^* \\ H_{21}^* & H_{22}^* & H_{23}^* & H_{24}^* \\ H_{31}^* & H_{32}^* & H_{33}^* & H_{34}^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

row by row as

$$\begin{aligned} H_{1j}^* &= H_{1j} && j = 1, 2, 3, 4 \\ H_{2j}^* &= \tfrac{1}{2}[H_{2j} - 2H_{1j}^*] && j = 1, 2, 3, 4 \\ H_{3j}^* &= H_{3j} - 3H_{1j}^* + 2H_{2j}^* && j = 1, 2, 3, 4 \end{aligned}$$

or

$$\mathbf{H}^* = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & \frac{1}{2} & 0 & 0 \\ -5 & 1 & 1 & -1 \end{bmatrix}$$

Using the transformed measurements, (7-16) or (7-17) can be applied iteratively three times to perform the update. ■

As noted in the previous section, the $\mathbf{S}(t_i^+)$ matrix generated by these update forms is generally not lower triangular, even if $\mathbf{S}(t_i^-)$ is.

7.5 COVARIANCE SQUARE ROOT FILTER FOR $\mathbf{Q}_d \neq \mathbf{0}$

If dynamic driving noise enters the system model, the conventional Kalman filter propagates the covariance matrix from one measurement time to the next by means of

$$\mathbf{P}(t_{i+1}^-) = \mathbf{\Phi}(t_{i+1}, t_i)\mathbf{P}(t_i^+)\mathbf{\Phi}^T(t_{i+1}, t_i) + \mathbf{G}_d(t_i)\mathbf{Q}_d(t_i)\mathbf{G}_d^T(t_i) \tag{7-22}$$

where this might be an equivalent discrete-time representation of a continuous-time system with sampled output (in which case $\mathbf{G}_d(t_i) \equiv \mathbf{I}$). Now we wish to develop an analogous recursion to yield $\mathbf{S}(t_{i+1}^-)$ in terms of $\mathbf{S}(t_i^+)$. It would be desirable to generate a lower triangular $\mathbf{S}(t_{i+1}^-)$ since then only $\frac{1}{2}n(n+1)$ elements would require computation rather than n^2.

One means of achieving the desired result is called the *matrix* RSS (*root-sum-square*) *method* [11]:

$$\begin{aligned}
\mathbf{X}(t_{i+1}) &= \mathbf{\Phi}(t_{i+1}, t_i)\mathbf{S}(t_i^+) \\
\mathbf{P}(t_{i+1}^-) &= \mathbf{X}(t_{i+1})\mathbf{X}^T(t_{i+1}) + [\mathbf{G}_d(t_i)\mathbf{Q}_d(t_i)\mathbf{G}_d^T(t_i)] \\
\mathbf{S}(t_{i+1}^-) &= \sqrt{\mathbf{P}(t_{i+1}^-)}
\end{aligned} \tag{7-23}$$

This method actually computes $\mathbf{P}(t_{i+1}^-)$ and then generates $\mathbf{S}(t_{i+1}^-)$ as its lower triangular Cholesky square root. Although this is a rapid method, it does suffer in having only the same numerical precision as the conventional filter time propagation. Nevertheless, since it is the measurement update and not the time propagation that causes the critical numerical problems in the filter, (7-23) may well be acceptable for many applications.

EXAMPLE 7.3 Let

$$\mathbf{\Phi}\mathbf{S}(t_i^+) = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}, \qquad \mathbf{G}_d\mathbf{Q}_d\mathbf{G}_d^T = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}$$

Note that $[\mathbf{\Phi}\mathbf{S}(t_i^+)]$ has purposely been chosen as nontriangular. Equation (7-23) yields, evaluating the Cholesky square root by (7-6),

$$\mathbf{X}(t_{i+1}) = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}$$

$$\mathbf{P}(t_{i+1}^-) = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 2 & 3 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 8 & 8 \\ 8 & 10 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 9 & 9 \\ 9 & 13 \end{bmatrix}$$

$$S_{11} = \sqrt{9} = 3, \qquad S_{12} = 0, \qquad S_{21} = \tfrac{1}{3} \cdot 9 = 3, \qquad S_{22} = \sqrt{13 - 3^2} = \sqrt{4} = 2$$

Thus

$$\mathbf{S}(t_{i+1}^-) = \begin{bmatrix} 3 & 0 \\ 3 & 2 \end{bmatrix} \qquad \blacksquare$$

The other means of establishing the time propagation relations is called the *triangularization method* [18]. In Section 7.3, the desired result (7-9b) was established by writing both sides of the covariance propagation (7-8b) in terms of a factor times its own transpose, and then equating the individual factors. Let us attempt to apply the same logic to (7-22). Assume that the square roots of $\mathbf{P}(t_i^+)$ and $\mathbf{Q}_d(t_i)$ are available: for $\mathbf{P}_0$ and $\mathbf{Q}_d(t_i)$ for all t_i, a Cholesky decomposition could be used, and for $\mathbf{P}(t_i^+)$ in general, assume the square root has been propagated and updated by the filter algorithm. Thus, we have

$$\mathbf{P}(t_i^+) = \mathbf{S}(t_i^+)\mathbf{S}^T(t_i^+) \tag{7-24a}$$

$$\mathbf{Q}_d(t_i) = \mathbf{W}_d(t_i)\mathbf{W}_d{}^T(t_i) \tag{7-24b}$$

Note that $\mathbf{S}(t_i^+)$ need not be lower triangular (important in view of the preceding section). Equation (7-22) can therefore be written as

$$\begin{aligned}\mathbf{P}(t_{i+1}^-) = {} & \mathbf{\Phi}(t_{i+1}, t_i)\mathbf{S}(t_i^+)\mathbf{S}^T(t_i^+)\mathbf{\Phi}^T(t_{i+1}, t_i) \\ & + \mathbf{G}_d(t_i)\mathbf{W}_d(t_i)\mathbf{W}_d{}^T(t_i)\mathbf{G}_d{}^T(t_i)\end{aligned} \tag{7-25}$$

Now it is desired to find the propagation equation for the square root of $\mathbf{P}(t_{i+1}^-)$: to find the relation to yield $\mathbf{S}(t_{i+1}^-)$ such that $\mathbf{S}(t_{i+1}^-)\mathbf{S}^T(t_{i+1}^-)$ is equal to the right hand side of (7-25).

One such matrix would be $\tilde{\mathbf{S}}(t_{i+1}^-)$ defined by

$$\tilde{\mathbf{S}}(t_{i+1}^-) = [\mathbf{\Phi}(t_{i+1}, t_i)\mathbf{S}(t_i^+) \mid \mathbf{G}_d(t_i)\mathbf{W}_d(t_i)] \tag{7-26}$$

However, if $\mathbf{S}(t_i^+)$ is n-by-n, then $[\mathbf{\Phi}(t_{i+1}, t_i)\mathbf{S}(t_i^+)]$ is n-by-n and $[\mathbf{G}_d(t_i)\mathbf{W}_d(t_i)]$ is n-by-s, so $\tilde{\mathbf{S}}(t_{i+1}^-)$ would be an n-by-$(n+s)$ square root of $\mathbf{P}(t_{i+1}^-)$. Since this type of square root increases the dimension of the covariance square root matrix for each propagation interval, it must be rejected as computationally impractical.

However, this does in fact provide a fruitful insight. If $\tilde{\mathbf{S}}(t_{i+1}^-)$ is a square root of $\mathbf{P}(t_{i+1}^-)$, then so is $[\tilde{\mathbf{S}}(t_{i+1}^-)\mathbf{T}]$ if $\mathbf{T}$ is an orthogonal $(n+s)$-by-$(n+s)$ matrix, i.e., $\mathbf{T}\mathbf{T}^T = \mathbf{I}$, since

$$\tilde{\mathbf{S}}(t_{i+1}^-)\mathbf{T}\mathbf{T}^T\tilde{\mathbf{S}}^T(t_{i+1}^-) = \tilde{\mathbf{S}}(t_{i+1}^-)\tilde{\mathbf{S}}^T(t_{i+1}^-) \tag{7-27}$$

Therefore, if an orthogonal matrix $\mathbf{T}$ can be found such that

$$\tilde{\mathbf{S}}(t_{i+1}^-)\mathbf{T} = [\underbrace{\mathbf{S}(t_{i+1}^-)}_{n \text{ columns}} \mid \underbrace{\mathbf{0}}_{s \text{ columns}}] \Big\}^{n \text{ rows}} \tag{7-28}$$

then in fact an n-by-n square root matrix $\mathbf{S}(t_{i+1}^-)$ will have been found which satisfies the desired relationship. If, in addition, this $\mathbf{S}(t_{i+1}^-)$ were lower triangular, the result would be especially advantageous. Two methods [22] of generating such a $\mathbf{S}(t_{i+1}^-)$, known as triangularization algorithms, are the *Gram–Schmidt orthogonalization* [13, 28] procedure and the *Householder transformation* [13, 20] technique. Note that the same procedure could also

be applied to

$$\mathbf{P}(t_i^+) = [\mathbf{I} - \mathbf{K}(t_i)\mathbf{H}(t_i)]\mathbf{P}(t_i^-)[\mathbf{I} - \mathbf{K}(t_i)\mathbf{H}(t_i)]^{\mathrm{T}} + \mathbf{K}(t_i)\mathbf{R}(t_i)\mathbf{K}^{\mathrm{T}}(t_i)$$

or

$$\mathbf{P}^{-1}(t_i^+) = \mathbf{P}^{-1}(t_i^-) + \mathbf{H}^{\mathrm{T}}(t_i)\mathbf{R}^{-1}(t_i)\mathbf{H}(t_i)$$

for vector measurement updates; the latter of these will be discussed subsequently.

First let us demonstrate that the Gram–Schmidt procedure yields the desired result. Let $\mathbf{e}^k$ denote the n-dimensional vector composed of all zeros except for a one as the kth component, so that $\mathbf{e}^1, \mathbf{e}^2, \ldots, \mathbf{e}^n$ form the standard basis for n-dimensional space. Then $[\tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^-)\mathbf{e}^k]$ is just the kth column of $\tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^-)$, of dimension $(n + s)$:

$$\tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^-) = \left.\underbrace{\begin{bmatrix} | & | & & | \\ \tilde{\mathbf{s}}^1 & \tilde{\mathbf{s}}^2 & \cdots & \tilde{\mathbf{s}}^n \\ | & | & & | \end{bmatrix}}_{n \text{ columns}}\right\}_{(n+s) \text{ rows}} \tag{7-29}$$

where

$$\begin{aligned} \tilde{\mathbf{s}}^1 &= \tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^-)\mathbf{e}^1 \\ \tilde{\mathbf{s}}^2 &= \tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^-)\mathbf{e}^2 \\ &\vdots \\ \tilde{\mathbf{s}}^n &= \tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^-)\mathbf{e}^n \end{aligned} \tag{7-30}$$

Construct the orthonormal basis vectors $\mathbf{b}^1, \mathbf{b}^2, \ldots, \mathbf{b}^n$ [each of dimension $(n + s)$] by the Gram–Schmidt procedure as:

$$\begin{aligned} \mathbf{b}^1 &= \text{unit}\,(\tilde{\mathbf{s}}^1) \\ \mathbf{b}^2 &= \text{unit}\,(\tilde{\mathbf{s}}^2 - [\tilde{\mathbf{s}}^{2\mathrm{T}}\mathbf{b}^1]\mathbf{b}^1) \\ \mathbf{b}^3 &= \text{unit}\,(\tilde{\mathbf{s}}^3 - [\tilde{\mathbf{s}}^{3\mathrm{T}}\mathbf{b}^1]\mathbf{b}^1 - [\tilde{\mathbf{s}}^{3\mathrm{T}}\mathbf{b}^2]\mathbf{b}^2) \\ &\vdots \end{aligned} \tag{7-31}$$

If $\mathbf{P}(t_i^+)$ is positive definite, then $\tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^-)$ is of rank n, so n such orthogonal unit basis vectors can be obtained. Now the desired orthogonal transformation matrix $\mathbf{T}$ can be defined as the $(n + s)$-by-$(n + s)$ matrix

$$\mathbf{T} = [\mathbf{b}^1, \mathbf{b}^2, \ldots, \mathbf{b}^n, \mathbf{b}^{n+1}, \ldots, \mathbf{b}^{n+s}] \tag{7-32}$$

where $\mathbf{b}^1, \mathbf{b}^2, \ldots, \mathbf{b}^n$ have been computed as in (7-31) and the remaining s columns, $\mathbf{b}^{n+1}, \ldots, \mathbf{b}^{n+s}$, are additional orthogonal unit basis vectors of $(n + s)$-dimensional space. However, it will be shown that they do not have to be computed to obtain $\mathbf{S}(t_{i+1}^-)$, so their generation will not be specified explicitly.

At this point, $\mathbf{T}^{\mathrm{T}}\tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^{-})$ can be written as

$$\mathbf{T}^{\mathrm{T}}\tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^{-}) = \begin{bmatrix} — & \mathbf{b}^{1\mathrm{T}} & — \\ — & \mathbf{b}^{2\mathrm{T}} & — \\ & \vdots & \\ — & \mathbf{b}^{(n+s)\mathrm{T}} & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \tilde{\mathbf{s}}^1 & \tilde{\mathbf{s}}^2 & \cdots & \tilde{\mathbf{s}}^n \\ | & | & & | \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{b}^{1\mathrm{T}}\tilde{\mathbf{s}}^1 & \mathbf{b}^{1\mathrm{T}}\tilde{\mathbf{s}}^2 & & \mathbf{b}^{1\mathrm{T}}\mathbf{s}^n \\ \mathbf{b}^{2\mathrm{T}}\tilde{\mathbf{s}}^1 & \mathbf{b}^{2\mathrm{T}}\tilde{\mathbf{s}}^2 & \cdots & \mathbf{b}^{2\mathrm{T}}\mathbf{s}^n \\ \vdots & \vdots & & \vdots \\ \mathbf{b}^{n\mathrm{T}}\tilde{\mathbf{s}}^1 & \mathbf{b}^{n\mathrm{T}}\tilde{\mathbf{s}}^2 & \cdots & \mathbf{b}^{n\mathrm{T}}\tilde{\mathbf{s}}^n \\ \vdots & \vdots & & \vdots \\ \mathbf{b}^{(n+s)\mathrm{T}}\tilde{\mathbf{s}}^1 & \mathbf{b}^{(n+s)\mathrm{T}}\tilde{\mathbf{s}}^2 & \cdots & \mathbf{b}^{(n+s)\mathrm{T}}\tilde{\mathbf{s}}^n \end{bmatrix} \tag{7-33}$$

However, since the rank of $\tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^{-})$ is n and $\{\mathbf{b}^1, \mathbf{b}^2, \ldots, \mathbf{b}^n\}$ span its range space while $\{\mathbf{b}^{n+1}, \ldots, \mathbf{b}^{n+s}\}$ are orthogonal to this spanning set, it follows that the last s rows in (7-33) are all zeros. By the manner in which the basis vectors were chosen, it is also true that

$$\mathbf{b}^{k\mathrm{T}}\tilde{\mathbf{s}}^j = 0, \qquad k > j$$

by the same reasoning. Thus, (7-33) becomes

$$\mathbf{T}^{\mathrm{T}}\tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^{-}) = \begin{bmatrix} \mathbf{b}^{1\mathrm{T}}\tilde{\mathbf{s}}^1 & \mathbf{b}^{1\mathrm{T}}\tilde{\mathbf{s}}^2 & \cdots & \mathbf{b}^{1\mathrm{T}}\tilde{\mathbf{s}}^n \\ & \mathbf{b}^{2\mathrm{T}}\tilde{\mathbf{s}}^2 & \cdots & \mathbf{b}^{2\mathrm{T}}\tilde{\mathbf{s}}^n \\ \mathbf{0} & & \ddots & \vdots \\ & & & \mathbf{b}^{n\mathrm{T}}\tilde{\mathbf{s}}^n \\ \hline & \mathbf{0} & & \end{bmatrix} \begin{matrix} \} n \text{ rows} \\ \} s \text{ rows} \end{matrix} \tag{7-34}$$

$$\underbrace{\qquad\qquad}_{n \text{ columns}}$$

The upper n-by-n partition of this matrix is just the $\mathbf{S}^{\mathrm{T}}(t_{i+1}^{-})$ we have been seeking, so that $\mathbf{S}(t_{i+1}^{-})$ is in fact an n-by-n lower triangular matrix:

$$\mathbf{S}(t_{i+1}^{-}) = \begin{bmatrix} \mathbf{b}^{1\mathrm{T}}\tilde{\mathbf{s}}^1 & & & \mathbf{0} \\ \mathbf{b}^{1\mathrm{T}}\tilde{\mathbf{s}}^2 & \mathbf{b}^{2\mathrm{T}}\tilde{\mathbf{s}}^2 & & \\ \vdots & \vdots & \ddots & \\ \mathbf{b}^{1\mathrm{T}}\tilde{\mathbf{s}}^n & \mathbf{b}^{2\mathrm{T}}\tilde{\mathbf{s}}^n & \cdots & \mathbf{b}^{n\mathrm{T}}\tilde{\mathbf{s}}^n \end{bmatrix} \tag{7-35}$$

An efficient computational form of the Gram–Schmidt orthogonalization called the *modified Gram–Schmidt* (MGS) [22, 28] algorithm has been shown to be numerically superior to the straightforward classical procedure, i.e., less susceptible to roundoff errors [9, 27]. Moreover, it requires no more arithmetic operations than the conventional Gram–Schmidt procedure, uses less storage, and has been shown [21] to have numerical accuracy comparable to Householder [19, 23] and Givens [15] transformations. Generation of $\mathbf{S}(t_{i+1}^{-})$ through this recursion proceeds as follows. Define the initial condition on $\mathbf{A}^k$, an $(n + s)$-

by-n matrix, as

$$\mathbf{A}^1 = \tilde{\mathbf{S}}^T(t_{i+1}^-) = [\boldsymbol{\Phi}(t_{i+1}, t_i)\mathbf{S}(t_i^+) \mid \mathbf{G}_d(t_i)\mathbf{W}_d(t_i)]^T \tag{7-36}$$

Notationally let $\mathbf{A}_j^k$ denote the jth column of $\mathbf{A}^k$. Perform the n-step recursion, for $k = 1, 2, \ldots, n$:

$$\begin{aligned} a^k &= \sqrt{\mathbf{A}_k^{kT}\mathbf{A}_k^k} \\ C_{kj} &= \begin{cases} 0 & j = 1, \ldots, k-1 \\ a^k & j = k \\ [(1/a^k)\mathbf{A}_k^{kT}]\mathbf{A}_j^k & j = k+1, \ldots, n \end{cases} \\ \mathbf{A}_j^{k+1} &= \mathbf{A}_j^k - C_{kj}[(1/a^k)\mathbf{A}_k^k] \qquad j = k+1, \ldots, n \end{aligned} \tag{7-37}$$

Note that on successive iterations, the new $\mathbf{A}_j^{k+1}$ vectors can be "written over" the $\mathbf{A}_j^k$ vectors to conserve memory. At the end of this recursion,

$$\mathbf{S}(t_{i+1}^-) = \mathbf{C}^T \tag{7-38}$$

Notice that the computational algorithm never calculates or stores $\mathbf{T}$ explicitly in generating $\mathbf{S}(t_{i+1}^-)$.

EXAMPLE 7.4 As in Example 7.3, let

$$\boldsymbol{\Phi}\mathbf{S}(t_i^+) = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}, \qquad \mathbf{G}_d\mathbf{Q}_d\mathbf{G}_d^T = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}, \qquad \mathbf{G}_d\mathbf{W}_d = \begin{bmatrix} 1 & 0 \\ 1 & \sqrt{2} \end{bmatrix}$$

so that

$$\tilde{\mathbf{S}}(t_{i+1}^-) = [\boldsymbol{\Phi}\mathbf{S}(t_i^+) \mid \mathbf{G}_d\mathbf{W}_d] = \begin{bmatrix} 2 & 2 & 1 & 0 \\ 1 & 3 & 1 & \sqrt{2} \end{bmatrix}$$

By (7-36), the initial condition is

$$\mathbf{A}^1 = \tilde{\mathbf{S}}^T(t_{i+1}^-) = \begin{bmatrix} 2 & 1 \\ 2 & 3 \\ 1 & 1 \\ 0 & \sqrt{2} \end{bmatrix}$$

The first pass through the recursion (7-37) yields:

$$a^1 = [(2)^2 + (2)^2 + (1)^2 + (0)^2]^{1/2} = \sqrt{9} = 3$$

$$C_{11} = a^1 = 3, \qquad C_{12} = \tfrac{1}{3}[2 \cdot 1 + 2 \cdot 3 + 1 \cdot 1 + 0 \cdot \sqrt{2}] = \tfrac{1}{3}[9] = 3$$

$$\mathbf{A}_2^2 = \mathbf{A}_2^1 - C_{12}(1/a^1)\mathbf{A}_1^1 = \begin{bmatrix} 1 \\ 3 \\ 1 \\ \sqrt{2} \end{bmatrix} - (3)\tfrac{1}{3}\begin{bmatrix} 2 \\ 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 0 \\ \sqrt{2} \end{bmatrix}$$

The second iteration of (7-37) produces:

$$a^2 = [(-1)^2 + (1)^2 + (0)^2 + (\sqrt{2})^2]^{1/2} = \sqrt{4} = 2$$

$$C_{21} = 0, \qquad C_{22} = 2$$

$$\mathbf{A}_3^3 = \text{not computed}$$

Finally, (7-38) generates $\mathbf{S}(t_{i+1}^-)$ as

$$\mathbf{S}(t_{i+1}^-) = \begin{bmatrix} C_{11} & C_{21} \\ C_{12} & C_{22} \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 3 & 2 \end{bmatrix}$$

which agrees with the result of Example 7.3. Moreover

$$\mathbf{S}(t_{i+1}^-)\mathbf{S}^{\mathrm{T}}(t_{i+1}^-) = \begin{bmatrix} 3 & 0 \\ 3 & 2 \end{bmatrix}\begin{bmatrix} 3 & 3 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 9 & 9 \\ 9 & 13 \end{bmatrix}$$

which agrees with a conventional Kalman filter covariance time propagation computation for this problem. ■

A *Householder transformation* [20] can also be used to solve (7-28) for the square root matrix $\mathbf{S}(t_{i+1}^-)$. Conceptually, it generates $\mathbf{T}$ as

$$\mathbf{T} = \mathbf{T}^n\mathbf{T}^{(n-1)} \cdots \mathbf{T}^1$$

where $\mathbf{T}^k$ is generated recursively as

$$\mathbf{T}^k = \mathbf{I} - d^k\mathbf{u}^k\mathbf{u}^{k\mathrm{T}}$$

with the scalar d^k and the $(n+s)$-vector $\mathbf{u}^k$ defined in the following. However, the computational algorithm never calculates these $\mathbf{T}^k$'s or $\mathbf{T}$ explicitly. The initial condition on the $(n+s)$-by-n $\mathbf{A}^k$ is

$$\mathbf{A}^1 = \tilde{\mathbf{S}}^{\mathrm{T}}(t_{i+1}^-) = [\boldsymbol{\Phi}(t_{i+1}, t_i)\mathbf{S}(t_i^+) \mid \mathbf{G}_{\mathrm{d}}(t_i)\mathbf{W}_{\mathrm{d}}(t_i)]^{\mathrm{T}} \tag{7-39}$$

Again, letting $\mathbf{A}_j^k$ represent the jth column of $\mathbf{A}^k$, perform the n-step recursion, for $k = 1, 2, \ldots, n$:

$$\begin{aligned}
a^k &= \sqrt{\textstyle\sum_{j=k}^{n+s} [A_{jk}^k]^2} \cdot \operatorname{sgn}\{A_{kk}^k\} \\
d^k &= \frac{1}{a^k(a^k + A_{kk}^k)} \\
u_j^k &= \begin{cases} 0 & j < k \\ a^k + A_{kk}^k & j = k \\ A_{jk}^k & j = (k+1), \ldots, (n+s) \end{cases} \\
y_j^k &= \begin{cases} 0 & j < k \\ 1 & j = k \\ d^k\mathbf{u}^{k\mathrm{T}}\mathbf{A}_j^k & j = (k+1), \ldots, n \end{cases} \\
\mathbf{A}^{k+1} &= \mathbf{A}^k - \mathbf{u}^k\mathbf{y}^{k\mathrm{T}}
\end{aligned} \tag{7-40}$$

At stage k, the first $(k-1)$ columns of $\mathbf{A}^k$ are zero below the diagonal of the upper square partition, and $\mathbf{u}^k$ has been chosen so that the subdiagonal elements of $\mathbf{A}_k^{k+1}$ will be zero. After the n iterations of (7-40),

$$\mathbf{A}^{n+1} = \begin{bmatrix} \mathbf{C} \\ \mathbf{0} \end{bmatrix} \begin{matrix} \} n \\ \} s \end{matrix} \tag{7-41}$$

$$\qquad\qquad\quad n$$

and then $\mathbf{S}(t_{i+1}^-)$ is generated as

$$\mathbf{S}(t_{i+1}^-) = \mathbf{C}^T \tag{7-42}$$

EXAMPLE 7.5 Consider the same problem as in Example 7.4. By (7-39),

$$\mathbf{A}^1 = \begin{bmatrix} 2 & 1 \\ 2 & 3 \\ 1 & 1 \\ 0 & \sqrt{2} \end{bmatrix}$$

The first iteration of (7-40) yields

$$a^1 = (2^2 + 2^2 + 1^2 + 0^2)^{1/2} \cdot \operatorname{sgn}\{2\} = \sqrt{9} = 3, \qquad d^1 = 1/[3(3+2)] = 1\ 15$$
$$u_1{}^1 = (3+2) = 5, \qquad u_2{}^1 = 2, \qquad u_3{}^1 = 1, \qquad u_4{}^1 = 0$$
$$y_1{}^1 = 1, \qquad y_2{}^1 = (1/15)[5 \cdot 1 + 2 \cdot 3 + 1 \cdot 1 + 0 \cdot \sqrt{2}] = 4/5$$

$$\mathbf{A}^2 = \begin{bmatrix} 2 & 1 \\ 2 & 3 \\ 1 & 1 \\ 0 & \sqrt{2} \end{bmatrix} - \begin{bmatrix} 5 \\ 2 \\ 1 \\ 0 \end{bmatrix} [1 \quad 4/5] = \begin{bmatrix} -3 & -3 \\ 0 & 7/5 \\ 0 & 1/5 \\ 0 & \sqrt{2} \end{bmatrix}$$

The second iteration of (7-40) produces

$$a^2 = [(7/5)^2 + (1/5)^2 + \sqrt{2}^2]^{1/2} \cdot \operatorname{sgn}\{7/5\} = \sqrt{4} = 2, \qquad d^2 = 1/\{2[2 + (7/5)]\} = 5/34$$
$$u_1{}^2 = 0, \qquad u_2{}^2 = 2 + (7/5) = 17/5, \qquad u_3{}^2 = 1/5, \qquad u_4{}^2 = \sqrt{2}$$
$$y_1{}^2 = 0, \qquad y_2{}^2 = 1$$

$$\mathbf{A}^3 = \begin{bmatrix} -3 & -3 \\ 0 & 7/5 \\ 0 & 1/5 \\ 0 & \sqrt{2} \end{bmatrix} - \begin{bmatrix} 0 \\ 17/5 \\ 1/5 \\ \sqrt{2} \end{bmatrix} [0 \quad 1] = \begin{bmatrix} -3 & -3 \\ 0 & -2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Therefore, from (7-41) and (7-42), $\mathbf{S}(t_{i+1}^-)$ is identified as the transpose of the upper 2-by-2 partition of $\mathbf{A}^3$:

$$\mathbf{S}(t_{i+1}^-) = \begin{bmatrix} -3 & 0 \\ -3 & -2 \end{bmatrix}$$

This is just the negative of the previous results, and thus is also a valid covariance square root. ■

The Householder triangularization requires $[4n^3 + 6n^2(s+1) + 2n]/6$ multiplies, $[4n^3 + 6sn^2 + 8n]/6$ adds, n divides, and n square roots. This is $[2n^3 - 8n]/6$ fewer multiples and $[2n^3 + 3n^2 - 11n]/6$ fewer adds than required by the modified Gram–Schmidt algorithm. However, the MGS algorithm becomes slightly more precise numerically as the residual size increases [22], and thus is a viable alternative.

A Householder transformation method has also been proposed for performing measurement updates [12, 22]. However, this has been shown to be

equivalent to the Potter method described previously, but not as efficient computationally [4].

Thus, the *covariance square root filter* (*Potter filter*) algorithm can be specified as follows. The propagation of the state estimate from one sample time to the next is given by (7-9a). Covariance square root time propagations are calculated by means of the matrix RSS method (7-23), the MGS algorithm given by (7-36)–(7-38), or the Householder transformation as in (7-39)–(7-42). Of these, the latter two are preferable since they are more accurate numerically than the computationally efficient first method, and numerics are the basic motivation for square root forms. Measurement updates would be processed through m iterations of the Potter algorithm (7-16) or (7-17). If the $\mathbf{R}(t_i)$ matrix is not diagonal, the transformation of variables given by (7-19)–(7-21) must first be performed.

EXAMPLE 7.6 This example illustrates one complete recursion of the Potter filter. Let $\mathbf{S}(t_i^-)$ and the corresponding $\mathbf{P}(t_i^-) = \mathbf{S}(t_i^-)\mathbf{S}^{\mathrm{T}}(t_i^-)$ be

$$\mathbf{S}(t_i^-) = \begin{bmatrix} 3 & 0 \\ 3 & 2 \end{bmatrix}, \qquad \mathbf{P}(t_i^-) = \begin{bmatrix} 9 & 9 \\ 9 & 13 \end{bmatrix}$$

as computed by the time propagation of Examples 7.3, 7.4, or 7.5. Now let a scalar measurement be taken such that $\mathbf{H}(t_i) = [1/3 \quad 1]$ and $R(t_i) = 4$. A conventional Kalman update would yield

$$\begin{aligned}
\mathbf{K}(t_i) &= \mathbf{P}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i)[\mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i) + R(t_i)]^{-1} \\
&= \frac{1}{20+4}\begin{bmatrix} 12 \\ 16 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 2/3 \end{bmatrix} \\
\mathbf{P}(t_i^+) &= \mathbf{P}(t_i^-) - \mathbf{K}(t_i)\mathbf{H}(t_i)\mathbf{P}(t_i^-) \\
&= \begin{bmatrix} 9 & 9 \\ 9 & 13 \end{bmatrix} - \begin{bmatrix} 1/2 \\ 2/3 \end{bmatrix}[12 \quad 16] = \begin{bmatrix} 3 & 1 \\ 1 & 7/3 \end{bmatrix} \\
\hat{\mathbf{x}}(t_i^+) &= \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)[z_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)]
\end{aligned}$$

The corresponding result given by (7-16) is:

$$\begin{aligned}
\mathbf{a}(t_i) &= \begin{bmatrix} 3 & 3 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 1/3 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} \\
b(t_i) &= 1/[4^2 + 2^2 + 4] = 1/24 \\
\gamma(t_i) &= 1/[1 + \sqrt{(1/24)(4)}] = 1/[1 + \sqrt{1/6}] \\
\mathbf{K}(t_i) &= \frac{1}{24}\begin{bmatrix} 3 & 0 \\ 3 & 2 \end{bmatrix}\begin{bmatrix} 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 12/24 \\ 16/24 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 2/3 \end{bmatrix} \\
\hat{\mathbf{x}}(t_i^+) &= \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)[z_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)] \\
\mathbf{S}(t_i^+) &= \begin{bmatrix} 3 & 0 \\ 3 & 2 \end{bmatrix} - \frac{1}{1+\sqrt{1/6}}\begin{bmatrix} 1/2 \\ 2/3 \end{bmatrix}[4 \quad 2] \\
&= \frac{1}{1+\sqrt{1/6}}\begin{bmatrix} 1 + 3\sqrt{1/6} & -1 \\ (1/3) + 3\sqrt{1/6} & (2/3) + 2\sqrt{1/6} \end{bmatrix}
\end{aligned}$$

Note that the computed gains $\mathbf{K}(t_i)$ agree and that

$$\mathbf{S}(t_i^+)\mathbf{S}^T(t_i^+) = \begin{bmatrix} 3 & 1 \\ 1 & 7/3 \end{bmatrix}$$

which agrees with $\mathbf{P}(t_i^+)$.

By comparison, (7-17) yields:

$$\mathbf{a}(t_i) = \begin{bmatrix} 3 & 3 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 1/3 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

$$\sigma(t_i) = (4^2 + 2^2 + 4)^{1/2} = \sqrt{24} = 2\sqrt{6}$$

$$\alpha(t_i) = 2\sqrt{6} + 2$$

$$\beta(t_i) = 1/[(2\sqrt{6})(2\sqrt{6} + 2)] = 1/[24 + 4\sqrt{6}]$$

$$\mathbf{g}(t_i) = \frac{1}{[24 + 4\sqrt{6}]}\begin{bmatrix} 12 \\ 16 \end{bmatrix} = \frac{1}{6 + \sqrt{6}}\begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \frac{1}{6 + \sqrt{6}}\begin{bmatrix} 3 \\ 4 \end{bmatrix}\frac{2\sqrt{6} + 2}{2\sqrt{6}}[z_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)]$$

$$\mathbf{S}(t_i^+) = \begin{bmatrix} 3 & 0 \\ 3 & 2 \end{bmatrix} - \frac{1}{6 + \sqrt{6}}\begin{bmatrix} 3 \\ 4 \end{bmatrix}[4 \quad 2]$$

$$= \frac{1}{1 + \sqrt{1/6}}\begin{bmatrix} 1 + 3\sqrt{1/6} & -1 \\ (1/3) + 3\sqrt{1/6} & (2/3) + 2\sqrt{1/6} \end{bmatrix}$$

The $\mathbf{S}(t_i^+)$ agrees with that just obtained. Moreover, if $\mathbf{g}(t_i)[\alpha(t_i)/\sigma(t_i)]$ were computed instead of the more efficient multiplication of the residual by the scalar $[\alpha(t_i)/\sigma(t_i)]$ followed by multiplication by $\mathbf{g}(t_i)$, the result would be identical to the $\mathbf{K}(t_i)$ previously computed. ■

One significant drawback of the covariance square root filter just described is that the triangularity of the square root matrix is generally destroyed during the measurement updating. Consequently, all n^2 elements must be computed and stored. A more recent algorithm, the *Carlson filter* [11], provides substantial improvement in both computational speed and required storage by maintaining the covariance square root matrix in triangular form. By doing so, only $n(n + 1)/2$ memory locations need be allocated for $\mathbf{S}(t_i^+)$, and the product $[\mathbf{\Phi}(t_{i+1}, t_i)\mathbf{S}(t_i^+)]$ for the subsequent time propagation requires only half the usual number of computations.

Like the Potter measurement update, the Carlson algorithm processes vector measurements iteratively as scalars. Therefore, consider the general square root solution to (7-11):

$$\begin{aligned} \mathbf{S}(t_i^+) &= \mathbf{S}(t_i^-)[\mathbf{I} - b(t_i)\mathbf{a}(t_i)\mathbf{a}^T(t_i)]^{1/2} \\ &= \mathbf{S}(t_i^-)[\mathbf{I} - \mathbf{a}(t_i)\mathbf{a}^T(t_i)/d(t_i)]^{1/2} \end{aligned} \tag{7-43}$$

where, for convenience, $d(t_i)$ has been defined as $1/b(t_i)$. Assuming $\mathbf{S}(t_i^-)$ to be upper triangular, we seek a matrix $[\mathbf{I} - \mathbf{a}(t_i)\mathbf{a}^T(t_i)/d(t_i)]^{1/2}$ such that the $\mathbf{S}(t_i^+)$ computed in (7-43) is also upper triangular. The choice between upper and

lower triangular form is arbitrary, governed by selecting either forward or backward recursion algorithms for the Cholesky, Householder, and Gram–Schmidt procedures. Upper triangular forms are motivated to some extent by state vector partitioning, discussed in Problem 7.13.

The desired square root matrix is in fact derived by means of an analytic Cholesky decomposition, and can be expressed as

$$\begin{bmatrix} b_1 & & & \\ & b_2 & & \\ & & \ddots & \\ & & & b_n \end{bmatrix} - \begin{bmatrix} 0 & a_1 & a_1 & \cdots & a_1 \\ & 0 & a_2 & \cdots & a_2 \\ & & 0 & & \vdots \\ & & & \ddots & a_{n-1} \\ & & & & 0 \end{bmatrix} \begin{bmatrix} c_1 & & & \\ & c_2 & & \\ & & \ddots & \\ & & & c_n \end{bmatrix}$$

where a_k is the kth component of $\mathbf{a}(t_i)$, and b_k and c_k for $k = 1, 2, \ldots, n$ will be described presently. However, the computational algorithm neither computes this square root explicitly nor requires a matrix multiplication as in (7-43) to generate $\mathbf{S}(t_i^+)$. The algorithm is initialized by setting the scalar d_0 and n-vectors $\mathbf{e}_0$ and $\mathbf{a}$ as

$$d_0 = R(t_i), \qquad \mathbf{e}_0 = \mathbf{0}, \qquad \mathbf{a} = \mathbf{S}^{\mathrm{T}}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i) \tag{7-44}$$

and iterating for $k = 1, 2, \ldots, n$ on

$$\begin{aligned} d_k &= d_{k-1} + a_k^2 \\ b_k &= (d_{k-1}/d_k)^{1/2} \\ c_k &= a_k/(d_{k-1}d_k)^{1/2} \\ \mathbf{e}_k &\doteq \mathbf{e}_{k-1} + \mathbf{S}_k^- a_k \\ \mathbf{S}_k^+ &= \mathbf{S}_k^- b_k - \mathbf{e}_{k-1} c_k \end{aligned} \tag{7-45}$$

In the recursion, $\mathbf{S}_k^-$ denotes the kth column of $\mathbf{S}(t_i^-)$, and both it and $\mathbf{e}_k$ consist of zeros below the kth element. After the n iterations, $\mathbf{S}(t_i^+)$ is produced as

$$\mathbf{S}(t_i^+) = [\mathbf{S}_1^+ \; \mathbf{S}_2^+ \cdots \mathbf{S}_n^+] \tag{7-46}$$

which is an upper triangular matrix. The state vector update is then given by

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{e}_n\{[z_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)]/d_n\} \tag{7-47}$$

EXAMPLE 7.7 Consider the same problem as in Example 7.6, but now assume $\mathbf{S}(t_i^-)$ to be upper triangular and such that $\mathbf{S}(t_i^-)\mathbf{S}^{\mathrm{T}}(t_i^-)$ is equal to

$$\mathbf{P}(t_i^-) = \begin{bmatrix} 9 & 9 \\ 9 & 13 \end{bmatrix}$$

The upper triangular Cholesky square root is found through (7-7) as:

$$\mathbf{S}(t_i^-) = \begin{bmatrix} 6/\sqrt{13} & 9/\sqrt{13} \\ 0 & \sqrt{13} \end{bmatrix}$$

The initialization of (7-44) yields

$$d_0 = 4, \qquad \mathbf{e}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad \mathbf{a} = \begin{bmatrix} 6/\sqrt{13} & 0 \\ 9/\sqrt{13} & \sqrt{13} \end{bmatrix} \begin{bmatrix} 1/3 \\ 1 \end{bmatrix} = \begin{bmatrix} 2/\sqrt{13} \\ 16/\sqrt{13} \end{bmatrix}$$

The first recursion of (7-45) yields

$$d_1 = 4 + \frac{4}{13} = \frac{56}{13}, \qquad b_1 = \left(4 \Big/ \frac{56}{13}\right)^{1/2} = \sqrt{\frac{13}{14}}$$

$$c_1 = \begin{bmatrix} \frac{2}{\sqrt{13}} \end{bmatrix} \Big/ \left[4 \cdot \frac{56}{13}\right]^{1/2} = \frac{1}{2\sqrt{14}}$$

$$\mathbf{e}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 6/\sqrt{13} \\ 0 \end{bmatrix} \frac{2}{\sqrt{13}} = \begin{bmatrix} 12/13 \\ 0 \end{bmatrix}$$

$$\mathbf{S}_1^+ = \begin{bmatrix} 6/\sqrt{13} \\ 0 \end{bmatrix} \sqrt{\frac{13}{14}} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 6/\sqrt{14} \\ 0 \end{bmatrix}$$

The second iteration yields

$$d_2 = \frac{56}{13} + \frac{256}{13} = \frac{312}{13}, \qquad b_2 = \left(\frac{56}{13} \Big/ \frac{312}{13}\right)^{1/2} = \sqrt{\frac{56}{312}} = \sqrt{\frac{7}{39}}$$

$$c_2 = \begin{bmatrix} \frac{16}{\sqrt{13}} \end{bmatrix} \Big/ \left[\frac{56}{13} \cdot \frac{312}{13}\right]^{1/2} = \frac{2}{\sqrt{21}}$$

$$\mathbf{e}_2 = \begin{bmatrix} 12/13 \\ 0 \end{bmatrix} + \begin{bmatrix} 9/\sqrt{13} \\ \sqrt{13} \end{bmatrix} \frac{16}{\sqrt{13}} = \begin{bmatrix} 156/13 \\ 16 \end{bmatrix}$$

$$\mathbf{S}_2^+ = \begin{bmatrix} 9/\sqrt{13} \\ \sqrt{13} \end{bmatrix} \frac{\sqrt{7}}{\sqrt{39}} - \begin{bmatrix} 12/13 \\ 0 \end{bmatrix} \frac{2}{\sqrt{21}} = \begin{bmatrix} \sqrt{3/7} \\ \sqrt{7/3} \end{bmatrix}$$

Thus, $\mathbf{S}(t_i^+)$ and $\hat{\mathbf{x}}(t_i^+)$ are given by (7-46) and (7-47) as

$$\mathbf{S}(t_i^+) = \begin{bmatrix} 6/\sqrt{14} & \sqrt{3/7} \\ 0 & \sqrt{7/3} \end{bmatrix}$$

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \begin{bmatrix} 156/13 \\ 16 \end{bmatrix} \left\{ [z_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)] \Big/ \frac{312}{13} \right\}$$

Note that the value of $[\mathbf{e}_2/d_2]$, not calculated explicitly above, agrees with $\mathbf{K}(t_i)$ of Example 7.6. Moreover, $\mathbf{S}(t_i^+)\mathbf{S}^T(t_i^+)$ is equal to the $\mathbf{P}(t_i^+)$ in that example. ■

For time propagations, Carlson suggested the matrix RSS method, (7-23), but with an upper triangular Cholesky square root as generated in (7-7) replacing the lower triangular form in (7-23). However, the triangularization methods could also be employed, thereby sacrificing some computational speed for increased numerical precision. A modified Gram–Schmidt algorithm [29] can

be written by initializing $\mathbf{A}$ according to (7-36) and then iterating for $k = n$, $n-1, \ldots, 1$ on

$$\left.\begin{aligned} S_{kk}(t_{i+1}^-) &= \sqrt{\mathbf{A}_k^{\mathrm{T}}\mathbf{A}_k} \\ \mathbf{v}_k &= \mathbf{A}_k / S_{kk}(t_{i+1}^-) \\ S_{jk}(t_{i+1}^-) &= \mathbf{A}_j^{\mathrm{T}}\mathbf{v}_k \\ \mathbf{A}_j &\leftarrow \mathbf{A}_j - S_{jk}(t_{i+1}^-)\mathbf{v}_k \end{aligned}\right\} \quad j = 1, 2, \ldots, (k-1) \tag{7-48}$$

where $\leftarrow$ denotes replacement by means of "writing over" old variables.

7.6 INVERSE COVARIANCE SQUARE ROOT FILTER

In Section 5.7, the inverse covariance formulation of the optimal filter was presented, an algorithm which is algebraically equivalent to the Kalman filter but has substantially different characteristics, such as being able to incorporate unknown initial conditions and being more efficient if the measurement vector is very large in dimension ($m > n$). Now we consider the square root filter analog of such a formulation.

If we define the covariance square root matrix $\mathbf{S}(t_i^+)$ through

$$\mathbf{P}(t_i^+) \triangleq \mathbf{S}(t_i^+)\mathbf{S}^{\mathrm{T}}(t_i^+) \tag{7-49}$$

then it is consistent that an inverse covariance square root $\mathbf{S}^{-1}(t_i^+)$ be defined through

$$\mathbf{P}^{-1}(t_i^+) \triangleq \mathbf{S}^{-\mathrm{T}}(t_i^+)\mathbf{S}^{-1}(t_i^+) \tag{7-50a}$$

where $\mathbf{S}^{-\mathrm{T}}$ denotes $[\mathbf{S}^{-1}]^{\mathrm{T}} = [\mathbf{S}^{\mathrm{T}}]^{-1}$. Similarly, $\mathbf{S}^{-1}(t_i^-)$ would be defined through

$$\mathbf{P}^{-1}(t_i^-) \triangleq \mathbf{S}^{-\mathrm{T}}(t_i^-)\mathbf{S}^{-1}(t_i^-) \tag{7-50b}$$

To develop the inverse covariance square root filter [5, 22], first consider the measurement update equation in the inverse covariance filter:

$$\mathbf{P}^{-1}(t_i^+) = \mathbf{P}^{-1}(t_i^-) + \mathbf{H}^{\mathrm{T}}(t_i)\mathbf{R}^{-1}(t_i)\mathbf{H}(t_i) \tag{7-51}$$

Using (7-5) and (7-50), this can be written as

$$\mathbf{P}^{-1}(t_i^+) = \mathbf{S}^{-\mathrm{T}}(t_i^-)\mathbf{S}^{-1}(t_i^-) + \mathbf{H}^{\mathrm{T}}(t_i)\mathbf{V}^{-\mathrm{T}}(t_i)\mathbf{V}^{-1}(t_i)\mathbf{H}(t_i) \tag{7-52}$$

We now seek an update relation for $\mathbf{S}^{-1}(t_i^+)$ such that $\{[\mathbf{S}^{-1}(t_i^+)]^{\mathrm{T}}[\mathbf{S}^{-1}(t_i^+)]\}$ is equivalent to the right hand side of (7-52). One such matrix would be the $(n+m)$-by-n matrix

$$\tilde{\mathbf{S}}^{-1}(t_i^+) = \begin{bmatrix} \mathbf{S}^{-1}(t_i^-) \\ \hline \mathbf{V}^{-1}(t_i)\mathbf{H}(t_i) \end{bmatrix} \tag{7-53}$$

As in the previous section, such an $\tilde{\mathbf{S}}^{-1}(t_i^+)$ would be unacceptable due to the increasing matrix dimensions it would cause. However, if an orthogonal matrix $\mathbf{T}$ can be constructed such that

$$\mathbf{T}^{\mathrm{T}}\tilde{\mathbf{S}}^{-1}(t_i^+) = \begin{bmatrix} \mathbf{S}^{-1}(t_i^+) \\ \hline \mathbf{0} \end{bmatrix} \begin{matrix} \} \ n \text{ rows} \\ \} \ m \text{ rows} \end{matrix} \tag{7-54}$$

or

$$\tilde{\mathbf{S}}^{-\mathrm{T}}(t_i^+)\mathbf{T} = [\mathbf{S}^{-\mathrm{T}}(t_i^+) \mid \mathbf{0}] \tag{7-54'}$$

then the resulting n-by-n $\mathbf{S}^{-1}(t_i^+)$ is the desired square root matrix. In analogy to the previous development, it would be especially beneficial if the $\mathbf{S}^{-1}(t_i^+)$ so generated were upper triangular. Either the modified Gram–Schmidt orthogonalization procedure or the Householder transformation algorithm can be employed to solve for the desired $\mathbf{S}^{-1}(t_i^+)$, and this will be developed in detail after the state estimate is discussed.

Recall from Section 5.7 that the inverse covariance filter did not compute a state estimate directly, but rather

$$\hat{\mathbf{y}}(t_i^-) \triangleq \mathbf{P}^{-1}(t_i^-)\hat{\mathbf{x}}(t_i^-) \tag{7-55a}$$

$$\hat{\mathbf{y}}(t_i^+) \triangleq \mathbf{P}^{-1}(t_i^+)\hat{\mathbf{x}}(t_i^+) \tag{7-55b}$$

which were related by

$$\hat{\mathbf{y}}(t_i^+) = \hat{\mathbf{y}}(t_i^-) + \mathbf{H}^{\mathrm{T}}(t_i)\mathbf{R}^{-1}(t_i)\mathbf{z}_i \tag{7-56}$$

Analogously, the inverse covariance square root filter does not generate an estimate of the state explicitly, but instead calculates

$$\hat{\boldsymbol{\alpha}}(t_i^-) \triangleq \mathbf{S}^{-1}(t_i^-)\hat{\mathbf{x}}(t_i^-) \tag{7-57a}$$

and

$$\hat{\boldsymbol{\alpha}}(t_i^+) \triangleq \mathbf{S}^{-1}(t_i^+)\hat{\mathbf{x}}(t_i^+) \tag{7-57b}$$

The update relationship between these estimates can be shown to be

$$\begin{matrix} n \text{ rows} \\ m \text{ rows} \end{matrix} \left\{ \begin{bmatrix} \hat{\boldsymbol{\alpha}}(t_i^+) \\ \hline \boldsymbol{\beta}(t_i) \end{bmatrix} \right. = \mathbf{T}^{\mathrm{T}} \begin{bmatrix} \hat{\boldsymbol{\alpha}}(t_i^-) \\ \hline \mathbf{V}^{-1}(t_i)\mathbf{z}_i \end{bmatrix} \begin{matrix} \} \ n \text{ rows} \\ \} \ m \text{ rows} \end{matrix} \tag{7-58}$$

where $\mathbf{T}$ is the same orthogonal matrix as in (7-54), and $\boldsymbol{\beta}(t_i)$ is an m-dimensional vector (the residual after processing the measurement, $[\mathbf{z}_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^+)]$) that need not be calculated. Since the first n rows of $\mathbf{T}^{\mathrm{T}}$ are the result of an n-step Gram–Schmidt or Householder process, $\hat{\boldsymbol{\alpha}}(t_i^+)$ can be computed without knowledge of any additional portion of $\mathbf{T}^{\mathrm{T}}$ than that generated by either of the triangularization algorithms discussed previously.

The *modified Gram–Schmidt* (MGS) *measurement update* initializes an $(n + m)$-by-n matrix $\mathbf{A}^k$ and an n-vector $\mathbf{b}^k$ as

$$\mathbf{A}^1 = \tilde{\mathbf{S}}^{-1}(t_i^+) = \begin{bmatrix} \mathbf{S}^{-1}(t_i^-) \\ \hline \mathbf{V}^{-1}(t_i)\mathbf{H}(t_i) \end{bmatrix} \tag{7-59a}$$

$$\mathbf{b}^1 = \mathbf{0} \tag{7-59b}$$

Then an n-step recursion is performed that is identical to (7-37) except for two additional equations for eventual generation of $\hat{\boldsymbol{\alpha}}(t_i^+)$; for $k = 1, 2, \ldots, n$,

$$\begin{aligned} a^k &= \sqrt{\mathbf{A}_k^{k\mathrm{T}}\mathbf{A}_k{}^k} \\ C_{kj} &= \begin{cases} 0 & j = 1, \ldots, k-1 \\ a^k & j = k \\ [(1/a^k)\mathbf{A}_k^{k\mathrm{T}}]\mathbf{A}_j{}^k & j = k+1, \ldots, n \end{cases} \\ e^k &= [(1/a^k)\mathbf{A}_k^{k\mathrm{T}}]\mathbf{b}^k \\ \mathbf{A}_j^{k+1} &= \mathbf{A}_j{}^k - C_{kj}[(1/a^k)\mathbf{A}_k{}^k] \qquad j = k+1, \ldots, n \\ \mathbf{b}^{k+1} &= \mathbf{b}^k - e^k[(1/a^k)\mathbf{A}_k{}^k] \end{aligned} \tag{7-60}$$

At the end of this recursion,

$$\mathbf{S}^{-1}(t_i^+) = \mathbf{C}, \qquad \hat{\boldsymbol{\alpha}}(t_i^+) = \mathbf{b}^{n+1} \tag{7-61}$$

A *Householder measurement update* [10, 18] can also be employed. The $(n + m)$-by-n matrix $\mathbf{A}^k$ and $(n + m)$-vector $\mathbf{b}^k$ (note the different dimension on $\mathbf{b}^k$) are initialized as in (7-59). Subsequently an n-step recursion identical in form to (7-40) except for auxiliary steps to calculate $\hat{\boldsymbol{\alpha}}(t_i^+)$ is performed [22]; for $k = 1, 2, \ldots, n$,

$$\begin{aligned} a^k &= \sqrt{\textstyle\sum_{j=k}^{n+m} [A_{jk}^k]^2} \cdot \operatorname{sgn}\{A_{kk}^k\} \\ d^k &= \frac{1}{a^k(a^k + A_{kk}^k)} \\ u_j^k &= \begin{cases} 0 & j < k \\ a^k + A_{kk}^k & j = k \\ A_{jk}^k & j = k+1, \ldots, (n+m) \end{cases} \\ y_j^k &= \begin{cases} 0 & j < k \\ 1 & j = k \\ d^k\mathbf{u}^{k\mathrm{T}}\mathbf{A}_j{}^k & j = k+1, \ldots, n \end{cases} \\ e^k &= a^k\mathbf{u}^{k\mathrm{T}}\mathbf{b}^k \\ \mathbf{A}^{k+1} &= \mathbf{A}^k - \mathbf{u}^k\mathbf{y}^{k\mathrm{T}} \\ \mathbf{b}^{k+1} &= \mathbf{b}^k - \mathbf{u}^k e^k \end{aligned} \tag{7-62}$$

After the n iterations of (7-62), $\mathbf{S}^{-1}(t_i^+)$ and $\hat{\boldsymbol{\alpha}}(t_i^+)$ are obtained from

$$\mathbf{A}^{n+1} = \begin{bmatrix} \mathbf{S}^{-1}(t_i^+) \\ \hline \mathbf{0} \end{bmatrix}, \qquad \mathbf{b}^{n+1} = \begin{bmatrix} \hat{\boldsymbol{\alpha}}(t_i^+) \\ \hline \boldsymbol{\beta}(t_i) \end{bmatrix} \tag{7-63}$$

For *time propagations* in which the dynamic driving noise is s dimensional, s scalar recursions analogous to the Potter measurement update in the covariance square root filter are performed. Thus $\mathbf{Q}_d(t_i)$ is assumed diagonal, perhaps after a change of variables (as explicitly described in the next section), and the effects of $\mathbf{w}_d(\cdot,\cdot)$ are incorporated component by component. Letting $\mathbf{G}_{dk}$ be the kth column of $\mathbf{G}_d(t_i)$ and Q_{dk} be the kth diagonal element of $\mathbf{Q}_d(t_i)$, the algorithm becomes, for $k = 1, 2, \ldots, s$,

$$\begin{aligned}
\mathbf{a}(t_i) &= \mathbf{S}^{-1}(t_i^+)\boldsymbol{\Phi}(t_i, t_{i+1})\mathbf{G}_{dk} \\
b(t_i) &= 1/[\mathbf{a}^T(t_i)\mathbf{a}(t_i) + \{1/Q_{dk}\}] \\
\gamma(t_i) &= 1/[1 + \sqrt{b(t_i)\{1/Q_{dk}\}}] \\
\mathbf{l}^T(t_i) &= b(t_i)\mathbf{a}^T(t_i)\mathbf{S}^{-1}(t_i^+)\boldsymbol{\Phi}(t_i, t_{i+1}) \\
\hat{\boldsymbol{\alpha}}(t_{i+1}^-) &= \boldsymbol{\alpha}(t_i^+) - b(t_i)\gamma(t_i)\mathbf{a}(t_i)\mathbf{a}^T(t_i)\boldsymbol{\alpha}(t_i^+) \\
\mathbf{S}^{-1}(t_{i+1}^-) &= \mathbf{S}^{-1}(t_i^+)\boldsymbol{\Phi}(t_i, t_{i+1}) - \gamma(t_i)\mathbf{a}(t_i)\mathbf{l}^T(t_i)
\end{aligned} \tag{7-64}$$

Note the order of the time indices on the state transition matrix and that $\boldsymbol{\Phi}(t_i, t_{i+1}) = \boldsymbol{\Phi}^{-1}(t_{i+1}, t_i)$. After the first of the s iterations of (7-64), $\boldsymbol{\Phi}(t_i, t_{i+1})$ is replaced by the identity matrix and $\mathbf{S}^{-1}(t_i^+)$ and $\hat{\boldsymbol{\alpha}}(t_i^+)$ are replaced by the $\mathbf{S}^{-1}$ and $\hat{\boldsymbol{\alpha}}$ computed in the previous iteration. In analogy to the covariance square root filter, a Householder transformation has also been proposed for performing the time propagation, but it has been shown to be equivalent to, but less efficient than, the Potter-type algorithm given in (7-64) [4].

Thus, in the inverse covariance square root filter, measurement updates are conducted in vector form through a triangularization procedure, and time propagations involve iterative applications of a Potter-type scalar incorporation algorithm. This is in direct opposition to the covariance square root filter. As a result, its time propagations are more efficient than those of the covariance square root filter for the typical case in which the state dimension is much greater than the dynamic noise dimension s. On the other hand, its measurement update is more efficient only when the measurement dimension m is considerably greater than n. Alternative, efficient forms of this filter, also known as square root information filters, have been developed and used extensively for certain applications [5, 8]. Although most applications have shown the covariance square root filter to be more efficient computationally, there are circumstances ($m \gg n \gg s$) under which the inverse covariance square root formulation is preferable. Section 7.8 will compare the various forms explicitly.

7.7 U–D COVARIANCE FACTORIZATION FILTER

Another approach to enhancing the numerical characteristics of the optimal filter algorithm is known as "**U–D** covariance factorization," developed by Bierman and Thornton [1, 6–8, 14–17, 29, 30]. Rather than decomposing the covariance into its square root factors as in (7-2) and (7-3), this method expresses the covariances before and after measurement incoporation as

$$\mathbf{P}(t_i^-) = \mathbf{U}(t_i^-)\mathbf{D}(t_i^-)\mathbf{U}^{\mathrm{T}}(t_i^-) \tag{7-65}$$

$$\mathbf{P}(t_i^+) = \mathbf{U}(t_i^+)\mathbf{D}(t_i^+)\mathbf{U}^{\mathrm{T}}(t_i^+) \tag{7-66}$$

where the **U** matrices are upper triangular and unitary (with ones along the diagonal) and the **D** matrices are diagonal. Although covariance square roots are never explicitly evaluated in this method, this filter algorithm is included in this chapter because (1) $\mathbf{UD}^{1/2}$ corresponds directly to the covariance square root of the Carlson filter in Section 7.5, and the Carlson filter in fact partially motivated this filter development, and (2) the **U–D** covariance factorization filter shares the advantages of the square root filters discussed previously: guaranteeing nonnegativity of the computed covariance and being numerically accurate and stable. (Merely being a square root filter is not a sufficient condition for numerical accuracy and stability, but the algorithms discussed previously do have these attributes.) Like the Carlson filter, triangular forms are maintained so that this algorithm is considerably more efficient in terms of computations and storage than the Potter filter. Though similar in concept and computation to the Carlson filter, this algorithm does not require any of the $(nm + s)$ computationally expensive scalar square roots as processed in the former.

Before considering the filter algorithm itself, let us demonstrate that, given some **P** as an n-by-n symmetric, positive semidefinite matrix, a unit upper triangular factor **U** and diagonal factor **D** such that $\mathbf{P} = \mathbf{UDU}^{\mathrm{T}}$ can always be generated. Although such **U** and **D** matrices are not unique, a uniquely defined pair can in fact be generated through an algorithm closely related to the backward running Cholesky decomposition algorithm, (7-7). This will be shown by explicitly displaying the result. First, for the nth column

$$\begin{aligned} D_{nn} &= P_{nn} \\ U_{in} &= \begin{cases} 1 & i = n \\ P_{in}/D_{nn} & i = n-1, n-2, \ldots, 1 \end{cases} \end{aligned} \tag{7-67a}$$

Then for the remaining columns, for $j = n-1, n-2, \ldots, 1$, compute

$$\begin{aligned} D_{jj} &= P_{jj} - \textstyle\sum_{k=j+1}^{n} D_{kk}U_{jk}^2 \\ U_{ij} &= \begin{cases} 0 & i > j \\ 1 & i = j \\ [P_{ij} - \sum_{k=j+1}^{n} D_{kk}U_{ik}U_{jk}]/D_{jj} & i = j-1, j-2, \ldots, 1 \end{cases} \end{aligned} \tag{7-67b}$$

This is useful for defining the required U–D factors of $\mathbf{P}_0$ and the $\mathbf{Q}_d$ time history for a given application.

To develop the filter algorithm itself, first consider a scalar measurement update, for which $\mathbf{H}(t_i)$ is 1-by-n. For convenience, we drop the time index and let $\mathbf{P}(t_i^-) = \mathbf{P}^-$, $\mathbf{P}(t_i^+) = \mathbf{P}^+$, and so forth. The Kalman update

$$\mathbf{P}^+ = \mathbf{P}^- - (\mathbf{P}^-\mathbf{H}^T)(1/a)(\mathbf{H}\mathbf{P}^-), \qquad a = \mathbf{H}\mathbf{P}^-\mathbf{H}^T + R \tag{7-68}$$

can be factored as

$$\begin{aligned}\mathbf{U}^+\mathbf{D}^+\mathbf{U}^{+T} &= \mathbf{U}^-\mathbf{D}^-\mathbf{U}^{-T} - (1/a)(\mathbf{U}^-\mathbf{D}^-\mathbf{U}^{-T}\mathbf{H}^T)\mathbf{H}\mathbf{U}^-\mathbf{D}^-\mathbf{U}^{-T} \\ &= \mathbf{U}^-[\mathbf{D}^- - (1/a)(\mathbf{D}^-\mathbf{U}^{-T}\mathbf{H}^T)(\mathbf{D}^-\mathbf{U}^{-T}\mathbf{H}^T)^T]\mathbf{U}^{-T}\end{aligned} \tag{7-69}$$

Note that $\mathbf{U}^{-T}$ is $(\mathbf{U}^-)^T$, as distinct from $\mathbf{S}^{-T} = (\mathbf{S}^{-1})^T$ in the previous section. Defining the n-vectors $\mathbf{f}$ and $\mathbf{v}$ as

$$\mathbf{f} = \mathbf{U}^{-T}\mathbf{H}^T \tag{7-70a}$$

$$\mathbf{v} = \mathbf{D}^-\mathbf{f}; \qquad \text{i.e.,} \quad v_j = D_{jj}^- f_j, \qquad j = 1, 2, \ldots, n \tag{7-70b}$$

and substituting into (7-69) yields

$$\mathbf{U}^+\mathbf{D}^+\mathbf{U}^{+T} = \mathbf{U}^-[\mathbf{D}^- - (1/a)\mathbf{v}\mathbf{v}^T]\mathbf{U}^{-T} \tag{7-71}$$

Now let $\bar{\mathbf{U}}$ and $\bar{\mathbf{D}}$ be the U–D factors of $[\mathbf{D}^- - (1/a)\mathbf{v}\mathbf{v}^T]$:

$$\bar{\mathbf{U}}\bar{\mathbf{D}}\bar{\mathbf{U}}^T = [\mathbf{D}^- - (1/a)\mathbf{v}\mathbf{v}^T] \tag{7-72}$$

so that (7-71) can be written as

$$\mathbf{U}^+\mathbf{D}^+\mathbf{U}^{+T} = [\mathbf{U}^-\bar{\mathbf{U}}]\bar{\mathbf{D}}[\mathbf{U}^-\bar{\mathbf{U}}]^T \tag{7-73}$$

Since $\mathbf{U}^-$ and $\bar{\mathbf{U}}$ are unit upper triangular, this then yields

$$\mathbf{U}^+ = \mathbf{U}^-\bar{\mathbf{U}} \tag{7-74a}$$

$$\mathbf{D}^+ = \bar{\mathbf{D}} \tag{7-74b}$$

In this manner, the problem of factoring the Kalman filter measurement update has been reduced to the problem of factoring a symmetric matrix, $[\mathbf{D} - (1/a)\mathbf{v}\mathbf{v}^T]$ into $\bar{\mathbf{U}}$ and $\bar{\mathbf{D}} = \mathbf{D}^+$. These factors can be generated [6, 14–16] recursively by letting $a_0 = R$ and computing, for $j = 1, 2, \ldots, n$,

$$\begin{aligned} a_j &= \sum_{k=1}^{j} D_{kk} f_k^2 + R \\ \bar{D}_{jj} &= D_{jj} a_{j-1}/a_j \\ \bar{U}_{ij} &= \begin{cases} -D_{ii} f_i f_j / a_{j-1} & i = 1, 2, \ldots, j-1 \\ 1 & i = j \\ 0 & i = j+1, j+2, \ldots, n \end{cases} \end{aligned} \tag{7-75}$$

Thus, $[\mathbf{D} - (1/a)\mathbf{v}\mathbf{v}^{\mathrm{T}}]$ is scanned and $\bar{\mathbf{U}}$ is generated column by column, as depicted in Fig. 7.3. The validity of the terms generated in (7-75) can be demonstrated by substituting them into (7-67) and showing that the resulting $[P_{ij}]$ matrix is in fact $[\mathbf{D} - (1/a)\mathbf{v}\mathbf{v}^{\mathrm{T}}]$.

The *scalar measurement update for the* **U–D** *covariance factorization filter* can now be specified. At time t_i, $\mathbf{U}(t_i^-)$ and $\mathbf{D}(t_i^-)$ are available from a previous time propagation (to be discussed). Using the measurement value z_i and the known 1-by-n $\mathbf{H}(t_i)$ and scalar $R(t_i)$, one computes

$$\begin{aligned} \mathbf{f} &= \mathbf{U}^{\mathrm{T}}(t_i^-)\mathbf{H}^{\mathrm{T}}(t_i) \\ v_j &= D_{jj}(t_i^-)f_j \qquad j = 1, 2, \ldots, n \\ a_0 &= R \end{aligned} \tag{7-76}$$

Then, for $k = 1, 2, \ldots, n$, calculate the results of (7-75), but in a more efficient manner as

$$\begin{aligned} a_k &= a_{k-1} + f_k v_k \\ D_{kk}(t_i^+) &= D_{kk}(t_i^-)\, a_{k-1}/a_k \\ b_k &\leftarrow v_k \\ p_k &= -f_k/a_{k-1} \\ U_{jk}(t_i^+) &= U_{jk}(t_i^-) + b_j p_k \\ b_j &\leftarrow b_j + U_{jk}(t_i^-)v_k \end{aligned} \qquad j = 1, 2, \ldots, (k-1) \tag{7-77}$$

In (7-77), $\leftarrow$ denotes replacement, exploiting the technique of "writing over" old variables for efficiency. For $k = 1$, only the first three equations need be processed. After the n iterations of (7-77), $\mathbf{U}(t_i^+)$ and $\mathbf{D}(t_i^+)$ have been computed, and the filter gain $\mathbf{K}(t_i)$ can be calculated in terms of the n-vector $\mathbf{b}$ made up of components $b_1, b_2, \ldots, b_n$ computed in the last iteration of (7-77), and the state updated as

$$\begin{aligned} \mathbf{K}(t_i) &= \mathbf{b}/a_n \\ \hat{\mathbf{x}}(t_i^+) &= \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)[z_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)] \end{aligned} \tag{7-78}$$

Vector measurement updates would be performed component by component, requiring a transformation of variables as in Section 7.4 if $\mathbf{R}(t_i)$ is not originally diagonal.

EXAMPLE 7.8 Consider the same problem treated previously, such that

$$\mathbf{P}(t_i^-) = \begin{bmatrix} 9 & 9 \\ 9 & 13 \end{bmatrix}, \qquad \mathbf{H}(t_i) = [\tfrac{1}{3} \quad 1], \qquad R(t_i) = 4$$

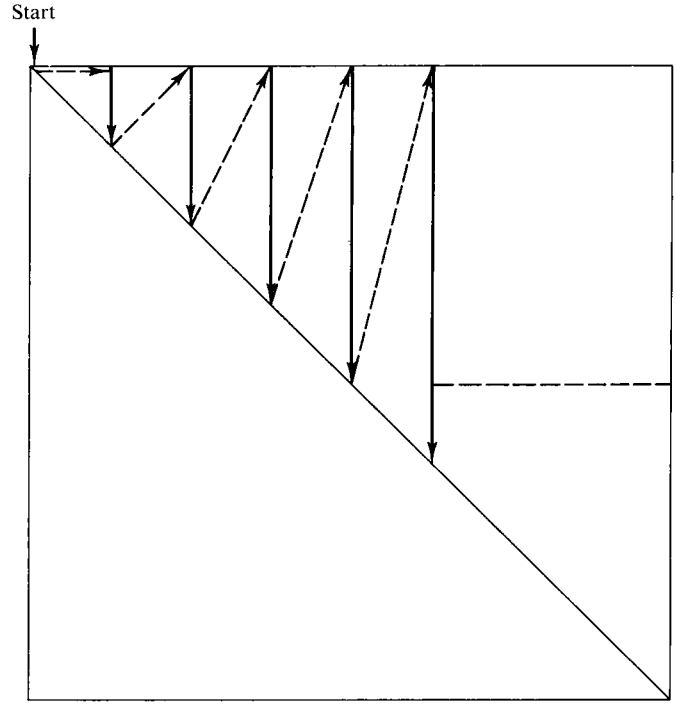

FIG. 7.3 Scanning of $[\mathbf{D} - (1/a)\mathbf{v}\mathbf{v}^{\mathrm{T}}]$ and generation of $\bar{\mathbf{U}}$.

The factors of $\mathbf{P}(t_i^-)$ are obtained from (7-67) as

$$D_{22}^- = P_{22}^- = 13$$
$$U_{22}^- = 1, \qquad U_{12}^- = P_{12}^-/D_{22}^- = 9/13$$
$$D_{11}^- = P_{11}^- - D_{22}^- U_{12}^{-2} = 9 - 13(9^2/13^2) = 36/13$$
$$U_{21}^- = 0, \qquad U_{11}^- = 1$$

Thus

$$\mathbf{U}(t_i^-) = \begin{bmatrix} 1 & 9/13 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{D}(t_i^-) = \begin{bmatrix} 36/13 & 0 \\ 0 & 13 \end{bmatrix}$$

Initialization by (7-76) yields

$$\mathbf{f} = \begin{bmatrix} 1 & 0 \\ 9/13 & 1 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 16/13 \end{bmatrix}$$
$$v_1 = [36/13][1/3] = 12/13$$
$$v_2 = [13][16/13] = 16$$
$$a_0 = 4$$

The first iteration of (7-77) produces

$$a_1 = 4 + [1/3][12/13] = 56/13$$
$$D_{11}(t_i^+) = [36/13][4]/[56/13] = 18/7$$
$$b_1 \leftarrow 12/13$$

The second iteration yields

$$a_2 = [56/13] + [16/13][16] = 24$$
$$D_{22}(t_i^+) = [13][56/13]/[24] = 7/3$$
$$b_2 \leftarrow 16$$
$$p_2 = -[16/13]/[56/13] = -2/7$$
$$U_{12}(t_i^+) = [9/13] + [12/13][-2/7] = 3/7$$
$$b_1 \leftarrow [12/13] + [9/13][16] = 12$$

Finally, (7-78) generates

$$\mathbf{K}(t_i) = \begin{bmatrix} 12 \\ 16 \end{bmatrix} \Big/ 24 = \begin{bmatrix} 1/2 \\ 2/3 \end{bmatrix}$$

$$\hat{\mathbf{x}}(t_i^+) = \hat{\mathbf{x}}(t_i^-) + \mathbf{K}(t_i)[z_i - \mathbf{H}(t_i)\hat{\mathbf{x}}(t_i^-)]$$

Note that the gain $\mathbf{K}(t_i)$ agrees with previous results and that

$$\mathbf{U}(t_i^+)\mathbf{D}(t_i^+)\mathbf{U}^{\mathrm{T}}(t_i^+) = \begin{bmatrix} 1 & 3/7 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 18/7 & 0 \\ 0 & 7/3 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 3/7 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 7/3 \end{bmatrix}$$

which is also consistent with the earlier computations. ■

The time propagation of the U–D factors employs a generalized Gram–Schmidt orthogonalization to preserve numerical accuracy while attaining computational efficiency [29]. Given the covariance time propagation relation

$$\mathbf{P}(t_{i+1}^-) = \mathbf{\Phi}(t_{i+1}, t_i)\mathbf{P}(t_i^+)\mathbf{\Phi}^{\mathrm{T}}(t_{i+1}, t_i) + \mathbf{G}_{\mathrm{d}}(t_i)\mathbf{Q}_{\mathrm{d}}(t_i)\mathbf{G}_{\mathrm{d}}{}^{\mathrm{T}}(t_i) \qquad (7\text{-}79)$$

and the U–D factors of $\mathbf{P}(t_i^+)$, we desire the factors $\mathbf{U}(t_{i+1}^-)$ and $\mathbf{D}(t_{i+1}^-)$ such that $[\mathbf{U}(t_{i+1}^-)\mathbf{D}(t_{i+1}^-)\mathbf{U}^{\mathrm{T}}(t_{i+1}^-)]$ equals the right hand side of (7-79). Without loss of generality, $\mathbf{Q}_{\mathrm{d}}(t_i)$ is assumed diagonal, since, given the n-by-n matrix $[\mathbf{G}_{\mathrm{d}}(t_i)\mathbf{Q}_{\mathrm{d}}(t_i)\mathbf{G}_{\mathrm{d}}{}^{\mathrm{T}}(t_i)]$, (7-67) can be used to generate $\mathbf{G}_{\mathrm{d}}(t_i)$ as its U-factor and $\mathbf{Q}_{\mathrm{d}}(t_i)$ as its D-factor.

If an n-by-$(n+s)$ matrix $\mathbf{Y}(t_{i+1}^-)$ and an $(n+s)$-by-$(n+s)$ diagonal matrix $\tilde{\mathbf{D}}(t_{i+1}^-)$ are defined as

$$\mathbf{Y}(t_{i+1}^-) = [\mathbf{\Phi}(t_{i+1}, t_i)\mathbf{U}(t_i^+) \mid \mathbf{G}_{\mathrm{d}}(t_i)] \qquad (7\text{-}80a)$$

$$\tilde{\mathbf{D}}(t_{i+1}^-) = \begin{bmatrix} \mathbf{D}(t_i^+) & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{Q}_{\mathrm{d}}(t_i) \end{bmatrix} \qquad (7\text{-}80b)$$

then it can be seen that $[\mathbf{Y}(t_{i+1}^-)\tilde{\mathbf{D}}(t_{i+1}^-)\mathbf{Y}^{\mathrm{T}}(t_{i+1}^-)]$ satisfies (7-79). Similar to the development of (7-29)–(7-55) of Section 7.5, the desired result can be generated through a Gram–Schmidt procedure applied to $\mathbf{Y}(t_{i+1}^-)$. The only significant

modification is that the inner products used in the procedure are weighted inner products: whereas in (7-31) the inner product of $\tilde{\mathbf{s}}^j$ [a column of $\tilde{\mathbf{S}}^T(t_{i+1}^-)$] and a basis vector $\mathbf{b}^k$ was written as $[\tilde{\mathbf{s}}^{jT}\mathbf{b}^k]$, here the inner product of $\mathbf{y}^j$ [a column of $\mathbf{Y}^T(t_{i+1}^-)$] and a basis vector $\mathbf{b}^k$ would be written as $[\mathbf{y}^{jT}\tilde{\mathbf{D}}(t_{i+1}^-)\mathbf{b}^k]$. When an analogous development is made, $\mathbf{D}(t_i^+)$ and $\mathbf{U}(t_i^+)$ can be identified as, for $j = 1, 2, \ldots, n$ and $k = j, j+1, \ldots, n$,

$$D_{jj}(t_{i+1}^-) = [\mathbf{b}^j]^T\tilde{\mathbf{D}}(t_{i+1}^-)\mathbf{b}^j \tag{7-81a}$$

$$U_{jk}(t_{i+1}^-) = \frac{1}{D_{kk}(t_{i+1}^-)}\{[\mathbf{y}^j]^T\tilde{\mathbf{D}}(t_{i+1}^-)\mathbf{b}^k\} \tag{7-81b}$$

As in Section 7.5, the actual computational algorithm is the efficient, numerically superior modified weighted Gram–Schmidt (MWGS) method. Thus, the *time propagation relations* are to compute $\mathbf{Y}(t_{i+1}^-)$ and $\tilde{\mathbf{D}}(t_{i+1}^-)$ as in (7-80), and initialize n vectors, each of dimension $(n+s)$, through

$$[\mathbf{a}_1 \quad \mathbf{a}_2 \mid \cdots \mid \mathbf{a}_n] = \mathbf{Y}^T(t_{i+1}^-) \tag{7-82}$$

and then to iterate on the following relations for $k = n, n-1, \ldots, 1$:

$$\begin{aligned}
\mathbf{c}_k &= \tilde{\mathbf{D}}(t_{i+1}^-)\mathbf{a}_k \qquad (c_{kj} = \tilde{D}_{jj}(t_{i+1}^-)a_{kj}, \quad j = 1, 2, \ldots, n) \\
D_{kk}(t_{i+1}^-) &= \mathbf{a}_k^T\mathbf{c}_k \\
\mathbf{d}_k &= \mathbf{c}_k/D_{kk}(t_{i+1}^-) \\
U_{jk}(t_{i+1}^-) &= \mathbf{a}_j^T\mathbf{d}_k \\
\mathbf{a}_j &\leftarrow \mathbf{a}_j - U_{jk}(t_{i+1}^-)\mathbf{a}_k
\end{aligned} \Bigg\} \quad j = 1, 2, \ldots, k-1 \tag{7-83}$$

As before, $\leftarrow$ denotes replacement, or "writing over" old variables to reduce storage requirements. On the last iteration, for $k = 1$, only the first two relations need be computed. The state estimate is given by

$$\hat{\mathbf{x}}(t_{i+1}^-) = \boldsymbol{\Phi}(t_{i+1}, t_i)\hat{\mathbf{x}}(t_i^+) \tag{7-84}$$

EXAMPLE 7.9 Consider the same time propagation as in Examples 7.3, 7.4, and 7.5; let

$$[\boldsymbol{\Phi}\mathbf{P}(t_i^+)\boldsymbol{\Phi}^T] = [\boldsymbol{\Phi}\mathbf{S}(t_i^+)][\mathbf{S}^T(t_i^+)\boldsymbol{\Phi}^T] = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}\begin{bmatrix} 2 & 1 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 8 & 8 \\ 8 & 10 \end{bmatrix}$$

$$\mathbf{G}_d\mathbf{Q}_d\mathbf{G}_d^T = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}$$

For the sake of this example, let

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & 1 \end{bmatrix}, \qquad \mathbf{P}(t_i^+) = \begin{bmatrix} 8 & 4 \\ 4 & 4 \end{bmatrix}$$

so that $[\boldsymbol{\Phi}\mathbf{P}(t_i^+)\boldsymbol{\Phi}^T]$ is as given above. The U–D factors of $\mathbf{P}(t_i^+)$ would be given by a previous measurement update; for this problem, they can be computed from (7-67) as

$$\mathbf{U}(t_i^+) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{D}(t_i^+) = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

Finally, since $\mathbf{Q}_d$ is assumed to be diagonal, $[\mathbf{G}_d\mathbf{Q}_d\mathbf{G}_d^T]$ can be factored by (7-67) into

$$\mathbf{G}_d = \begin{bmatrix} 1 & \frac{1}{3} \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{Q}_d = \begin{bmatrix} \frac{2}{3} & 0 \\ 0 & 3 \end{bmatrix}$$

The time propagation computations are initialized by (7-80) as

$$\mathbf{Y}(t_{i+1}^-) = \left[\begin{bmatrix} 1 & 0 \\ \frac{1}{2} & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \vdots \begin{bmatrix} 1 & \frac{1}{3} \\ 0 & 1 \end{bmatrix}\right] = \begin{bmatrix} 1 & 1 & 1 & \frac{1}{3} \\ \frac{1}{2} & \frac{3}{2} & 0 & 1 \end{bmatrix}$$

$$\tilde{\mathbf{D}}(t_{i+1}^-) = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & \frac{2}{3} & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

so that (7-82) yields

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \frac{1}{3} \end{bmatrix}, \qquad \mathbf{a}_2 = \begin{bmatrix} \frac{1}{2} \\ \frac{3}{2} \\ 0 \\ 1 \end{bmatrix}$$

The first iteration of (7-83), for $k = n = 2$, produces

$$\mathbf{c}_2 = \begin{bmatrix} 4 \cdot \frac{1}{2} \\ 4 \cdot \frac{3}{2} \\ \frac{2}{3} \cdot 0 \\ 3 \cdot 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 0 \\ 3 \end{bmatrix}$$

$$D_{22}(t_{i+1}^-) = \begin{bmatrix} \frac{1}{2} & \frac{3}{2} & 0 & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 6 \\ 0 \\ 3 \end{bmatrix} = 13$$

$$\mathbf{d}_2 = \begin{bmatrix} 2 \\ 6 \\ 0 \\ 3 \end{bmatrix} \cdot \frac{1}{13} = \begin{bmatrix} 2/13 \\ 6/13 \\ 0 \\ 3/13 \end{bmatrix}$$

$$U_{12}(t_{i+1}^-) = \begin{bmatrix} 1 & 1 & 1 & 1/3 \end{bmatrix}\begin{bmatrix} 2/13 \\ 6/13 \\ 0 \\ 3/13 \end{bmatrix} = 9/13$$

$$\mathbf{a}_1 \leftarrow \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1/3 \end{bmatrix} - \frac{9}{13}\begin{bmatrix} 1/2 \\ 3/2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 17/26 \\ -1/26 \\ 1 \\ -14/39 \end{bmatrix}$$

The second iteration, for $k = 1$, generates

$$\mathbf{c}_1 = \begin{bmatrix} 4 \cdot 17/26 \\ 4 \cdot -1/26 \\ 2/3 \cdot 1 \\ 3 \cdot -14/39 \end{bmatrix} = \begin{bmatrix} 34/13 \\ -2/13 \\ 2/3 \\ -14/13 \end{bmatrix}$$

$$D_{11}(t_{i+1}^-) = [17/26 \quad -1/26 \quad 1 \quad -14/39] \begin{bmatrix} 34/13 \\ -2/13 \\ 2/3 \\ -14/13 \end{bmatrix} = 36/13$$

Thus, $\mathbf{U}(t_{i+1}^-)$ and $\mathbf{D}(t_{i+1}^-)$ have been generated as

$$\mathbf{U}(t_{i+1}^-) = \begin{bmatrix} 1 & 9/13 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{D}(t_{i+1}^-) = \begin{bmatrix} 36/13 & 0 \\ 0 & 13 \end{bmatrix}$$

Note that

$$\mathbf{U}(t_{i+1}^-)\mathbf{D}(t_{i+1}^-)\mathbf{U}^{\mathrm{T}}(t_{i+1}^-) = \begin{bmatrix} 9 & 9 \\ 9 & 13 \end{bmatrix}$$

as found in the earlier examples or by adding $[\mathbf{\Phi P}(t_i^+)\mathbf{\Phi}^{\mathrm{T}}]$ and $[\mathbf{G}_d\mathbf{Q}_d\mathbf{G}_d^{\mathrm{T}}]$ directly. ■

7.8 FILTER PERFORMANCE AND REQUIREMENTS

The algorithms of this chapter have been investigated in order to implement the optimal filtering solution given by the Kalman filter, but without the numerical instability and inaccuracy of that algorithm when processed with finite wordlength. In this section, both the numerical advantages and the increased computational burden of these filters will be delineated.

An algorithm can be said to be numerically stable if the computed result of the algorithm corresponds to an exactly computed solution to a problem that is only slightly perturbed from the original one [31]. By this criterion, the *Kalman filter is numerically unstable* [6], in both the conventional and Joseph formulations. In contrast, *all of the filters described in this chapter can be shown to be numerically stable.*

The numerical conditioning of a set of computations can be described in part by what is called a "condition number," a concept which is often used to analyze the effects of perturbations in linear equations. If $\mathbf{A}$ is a matrix, not necessarily square, then the condition number $k(\mathbf{A})$ associated with $\mathbf{A}$ is defined by [22]:

$$k(\mathbf{A}) = \sigma_{\max}/\sigma_{\min} \tag{7-85}$$

where $\sigma_{\max}^2$ and $\sigma_{\min}^2$ are the maximum and minimum eigenvalues of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$, respectively. When computing in base 10 (or base 2) arithmetic with N significant digits (or bits), numerical difficulties may be expected as $k(\mathbf{A})$ approaches 10^N

(or 2^N). For instance, if the maximum and minimum numbers of interest, σ_{max} and σ_{min}, were 100000 and 000001 (in base 10 or 2), then to add these values together and obtain 100001 without numerical difficulties would require at least six significant figures (digits or bits). But,

$$k(\mathbf{P}) = k(\mathbf{S}\mathbf{S}^T) = [k(\mathbf{S})]^2 \tag{7-86}$$

Therefore, while numerical operations on the covariance $\mathbf{P}$ may encounter difficulties when $k(\mathbf{P}) = 10^N$ (or 2^N), those same numerical problems would arise when $k(\mathbf{S}) = 10^{N/2}$ (or $2^{N/2}$) according to (7-86): *the same numerical precision is achieved with half the wordlength.*

EXAMPLE 7.10 This example and the next illustrate the improved numerical characteristics of the square root filters. To simulate roundoff, let $e \ll 1$ be such that

$$1 + e \overset{r}{\neq} 1$$
$$1 + e^2 \overset{r}{=} 1$$

where $\overset{r}{=}$ means equal due to rounding. Consider a scalar measurement update of a two-state problem, with

$$\mathbf{P}(t_i^-) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{H}(t_i) = [1 \quad 0], \qquad R(t_i) = e^2$$

and compare the computed results of the filters of Chapter 5 and of this chapter. Note that

$$\mathbf{P}(t_i^-) = \mathbf{P}^{-1}(t_i^-) = \mathbf{S}(t_i^-) = \mathbf{S}^{-1}(t_i^-) = \mathbf{U}(t_i^-) = \mathbf{D}(t_i^-) = \mathbf{I}$$

and that the exact covariance $\mathbf{P}(t_i^+)$ for this example is:

$$\mathbf{P}(t_i^+) = \begin{bmatrix} e^2/(1+e^2) & 0 \\ 0 & 1 \end{bmatrix}$$

The computed results are

(a) conventional Kalman

$$\mathbf{P}(t_i^+) \overset{r}{=} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

(b) Joseph form, Kalman

$$\mathbf{P}(t_i^+) \overset{r}{=} \begin{bmatrix} e^2 & 0 \\ 0 & 1 \end{bmatrix}$$

(c) Potter covariance square root

$$\mathbf{S}(t_i^+) \overset{r}{=} \begin{bmatrix} e & 0 \\ 0 & 1 \end{bmatrix}$$

(d) Carlson covariance square root

$$\mathbf{S}(t_i^+) \overset{r}{=} \begin{bmatrix} e & 0 \\ 0 & 1 \end{bmatrix}$$

(e) inverse covariance

$$\mathbf{P}^{-1}(t_i^+) \triangleq \begin{bmatrix} 1/e^2 & 0 \\ 0 & 1 \end{bmatrix}$$

(f) inverse covariance square root

$$\mathbf{S}^{-1}(t_i^+) \triangleq \begin{bmatrix} 1/e & 0 \\ 0 & 1 \end{bmatrix}$$

(g) **U–D** factor

$$\mathbf{U}(t_i^+) \triangleq \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{D}(t_i^+) \triangleq \begin{bmatrix} e^2 & 0 \\ 0 & 1 \end{bmatrix}$$

For this example, all but the conventional Kalman filter yield nonsingular and nearly exact answers. Although the difference between 0 and e^2 in the upper left element of $\mathbf{P}(t_i^+)$ may seem insignificant, it can have grave consequences. For instance, assume no dynamics and let a second measurement of the same form be taken. The gain $\mathbf{K}$ computed by the conventional Kalman filter would be

$$\begin{aligned} \mathbf{K}(t_i^{++}) &= \mathbf{P}(t_i^+)\mathbf{H}(t_i)/[\mathbf{H}(t_i)\mathbf{P}(t_i^+)\mathbf{H}^{\mathrm{T}}(t_i) + R(t_i)] \\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Big/ [0 + 1] = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned}$$

whereas the correct value is

$$\mathbf{K}(t_i^{++}) = \frac{e^2}{1 + e^2} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Big/ \left\{ \frac{e^2}{1 + e^2} + e^2 \right\} \approxeq \frac{1}{2} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

as would be calculated correctly by the Joseph form in this case. ■

EXAMPLE 7.11 Consider the same problem as in Example 7.10, but let $\mathbf{H}(t_i)$ now be $[1\ 1]$ instead of $[1\ 0]$. In this case, the exact answer is

$$\mathbf{P}(t_i^+) = \frac{1}{2 + e^2} \begin{bmatrix} 1 + e^2 & -1 \\ -1 & 1 + e^2 \end{bmatrix}$$

The computed results are

(a) conventional Kalman

$$\mathbf{P}(t_i^+) \triangleq \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

(b) Joseph form Kalman

$$\mathbf{P}(t_i^+) \triangleq \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

(c) Potter covariance square root

$$\mathbf{S}(t_i^+) \triangleq \begin{bmatrix} 1 + e/\sqrt{2} & 0 \\ -1 + e/\sqrt{2} & 1 + e/\sqrt{2} \end{bmatrix}$$

(d) Carlson covariance square root

$$\mathbf{S}(t_i^+) \triangleq \begin{bmatrix} e & -1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{bmatrix}$$

(e) inverse covariance

$$\mathbf{P}^{-1}(t_i^+) \triangleq \frac{1}{e^2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

(f) inverse covariance square root

$$\mathbf{S}^{-1}(t_i^+) \triangleq \frac{1}{e}\begin{bmatrix} -1 & -1 \\ 0 & e\sqrt{2} \end{bmatrix}$$

(g) **U**–**D** factor

$$\mathbf{U}(t_i^+) \triangleq \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}, \qquad \mathbf{D}(t_i^+) \triangleq \begin{bmatrix} e^2 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

In this case, only the square root and **U**–**D** implementations yield nonsingular results. Such singular $\mathbf{P}(t_i^+)$ or $\mathbf{P}^{-1}(t_i^+)$ matrices would again yield a zero gain **K** if a second measurement of the same form were processed, while the square root and **U**–**D** factor filters compute a gain which is nearly exact. Moreover, even though **S**, $\mathbf{S}^{-1}$, **U**, and **D** are nonsingular, the associated value of $\mathbf{P}(t_i^+)$ or $\mathbf{P}^{-1}(t_i^+)$ found by multiplication would be rounded to a singular matrix; thus it is better not to perform such computations explicitly, and the time propagations based on triangularization are to be preferred over the RSS method which performs such multiplication. ■

The improved numerical characteristics of the square root and **U**–**D** factorization filters are achieved at the expense of increased computational burden. Letting n be the dimension of the state vector **x**, s be the dimension of the dynamic driving noise **w**, and m be the dimension of the measurement **z** and its corruptive noise **v**, we now determine the number of mathematical operations required by the various filters, assuming that

(1) all implementations take advantage of symmetry and zeros as they appear in general forms,
(2) $\mathbf{R}(t_i)$ and $\mathbf{Q}_d(t_i)$ are diagonal, and
(3) the inverse covariance and inverse covariance square root filters generate explicit state estimates, $\hat{\mathbf{x}}(t_i^-)$ and $\hat{\mathbf{x}}(t_i^+)$.

Table 7.1 presents the number of operations for one time propagation and one measurement update required by

(1) Kalman filter—with conventional and Joseph form measurement update,
(2) Potter covariance square root filter—with MGS and Householder time propagations,
(3) Carlson covariance square root filter—with matrix RSS and MGS time propagations,
(4) inverse covariance filter [using (5-91) for time propagation],

TABLE 7.1

Operations Required for One Time Propagation and One Measurement Update

Filter	Adds (all times $\frac{1}{6}$)	Multiplies (all times $\frac{1}{6}$)	Divides	Square roots
Conventional Kalman	$9n^3 + 3n^2(3m + s - 1) + n(15m + 3s - 6)$	$9n^3 + 3n^2(3m + s + 3) + n(27m + 9s)$	m	0
Joseph form Kalman	$18n^3 + 3n^2(5m + s - 10) + n(9m^2 + 6m + 3s) + 3m^3 - 6m^2 + 3m$	$18n^3 + 3n^2(5m + s + 4) + n(9m^2 + 24m + 9s) + 3m^3 + 9m^2 - 6m$	$2m - 1$	0
Potter covariance square root (MGS)	$12n^3 + 3n^2(6m + 2s) + n(6m - 6) + 6m$	$12n^3 + 3n^2(6m + 2s + 2) + n(24m + 6s) + 12m$	$n + 2m$	$n + m$
Potter covariance square root (Householder)	$10n^3 + 3n^2(6m + 2s - 1) + n(6m + 5) + 6m$	$10n^3 + 3n^2(6m + 2s + 2) + n(24m + 6s + 8) + 12m$	$n + 2m$	$n + m$
Carlson covariance square root (RSS)	$5n^3 + 3n^2(3m + s + 1) + n(9m + 3s - 14) + 2s^3 + 4s$	$5n^3 + 3n^2(4m + s + 3) + n(30m + 9s - 2) + 2s^3 + 6s^2 - 2s$	$2mn + s$	$mn + s$
Carlson covariance square root (MGS)	$9n^3 + 3n^2(3m + s - 1) + 3n(3m + 3s - 8) + 2s^3 + 6s^2 + 4s$	$9n^3 + 3n^2(4m + s + 2) + 3n(10m + 5s - 7) + 2s^3 + 12s^2 + 4s$	$2mn + s$	$mn + s$
Inverse covariance	$10n^3 + 3n^2(m + 3s + 2) + n(9m + 9s - 16)$	$10n^3 + 3n^2(m + 3s + 6) + n(15m + 21s - 10)$	$2s - 1$	0
Inverse covariance square root	$9n^3 + 3n^2(2m + 6s + 5) + n(12m + 6s - 6)$	$9n^3 + 3n^2(2m + 6s + 6) + n(12m + 24s + 3) + 6s$	$2n + 2s$	$n + s$
U–D factor	$9n^3 + 3n^2(3m + 2s + 2) + 3n(3m + 1)$	$9n^3 + 3n^2(3m - 2s + 7) + 3n(m + 4s - 4) - 6s$	$n(m + 1) - 1$	0

(5) inverse covariance square root filter (MGS update), and
(6) **U–D** covariance factorization filter.

This table can be used to project the computation time required by each filter formulation for a given application. Note that if, instead of assuming $\mathbf{Q}_d(t_i)$ to be diagonal, we were to assume that the n-by-n $[\mathbf{G}_d(t_i)\mathbf{Q}_d(t_i)\mathbf{G}_d^T(t_i)]$ or the n-by-s $[\mathbf{G}_d(t_i)\mathbf{W}_d(t_i)]$ were known, there would be $\frac{1}{2}ns(n + 3)$ fewer multiplies and $\frac{1}{2}n(n + 1)(s - 1)$ fewer adds in filter forms 1, 3 with RSS time propagation, and 4, or ns fewer multiplies in forms 2 and 3 with MGS time propagation. Problem 7.13 extends this table to account for taking advantage of matrix sparsity and structure in typical estimation problems.

EXAMPLE 7.12 To put the algebraic expressions of Table 7.1 into perspective, Table 7.2 presents the number of operations required for one time propagation and one measurement update for the case of $n = 10$, $s = 10$, and $m = 2$. The noise dimension s was intentionally set equal

TABLE 7.2

Operations for One Total Filter Recursion[a]

Filter	Adds	Multiplies	Divides	Square roots	Time (msec)
Conventional Kalman	2340	2690	2	0	17.36
Joseph form Kalman	3631	4498	3	0	28.27
Potter covariance square root (MGS)	3612	3884	14	12	26.49
Potter covariance square root (Householder)	3247	3564	14	12	24.19
Carlson covariance square root (RSS)	2080	2560	50	30	18.24
Carlson covariance square root (MGS)	2830	3355	50	30	23.53
Inverse covariance	3520	3950	19	0	25.82
Inverse covariance square root	5080	5455	40	20	37.55
U–D factor	2935	3330	29	0	21.77

[a] $n = s = 10$ and $m = 2$.

to n to correspond to the n-by-n $[\mathbf{G}_d(t_i)\,\mathbf{Q}_d(t_i)\mathbf{G}_d^T(t_i)]$ being of full rank, typical of an equivalent discrete-time model. The last column in Table 7.2 portrays computer time required for one total filter recursion, neglecting the computations associated with the various subscripting and storage operations for each filter (roughly the same for each), and using single precision instruction times typical of the IBM 360 and some smaller state-of-the-art computers:

$$\text{time for addition} = 2.7\ \mu\text{sec}$$
$$\text{time for multiplication} = 4.1\ \mu\text{sec}$$
$$\text{time for division} = 6.6\ \mu\text{sec}$$
$$\text{time for square root} = 60.0\ \mu\text{sec}$$

As can be seen from Table 7.2, the covariance square root filters and the U–D covariance factorization filter involve a computational load greater than the conventional Kalman filter, but not so great as to be prohibitive. In fact, the increase is less than that caused by employing the Joseph form of the update equation, which is inferior to these filters in performance. Moreover, since the Kalman filter would probably require double precision operations instead of the single precision assumed to establish Table 7.2, these filters are even more competitive with the Kalman filter than indicated in the table.

Of the square root type filters, the Carlson covariance square root and the U–D covariance factorization filters are the most efficient computationally. The Carlson filter with matrix RSS time propagations requires the least computer time, but this is offset by the degraded numerical accuracy of the matrix RSS method. Thus, the U–D covariance factorization filter would appear to be an exceptionally efficient and numerically advantageous alternative to the conventional Kalman filter for this particular application. ■

7.9 SUMMARY

This chapter presented the concept of square root filters and the closely related **U–D** covariance factorization filter as viable alternatives to conventional Kalman filters. For a modest increase in computational loading, one obtains optimal filter algorithms equivalent to the Kalman filter if infinite wordlength is assumed, but with vastly superior numerical characteristics with finite wordlength. From a numerical analysis standpoint, this is at least as good a solution to troublesome measurement update computations as implementing a Kalman filter in double precision, since the Kalman filter inherently involves unstable numerics.

Of the covariance square root forms, the Carlson filter is more efficient than the Potter form computationally, and it also maintains triangularity of the square root matrices. The **U–D** covariance factorization filter is comparable to the Carlson filter and does not require square root computations. In comparison, the inverse covariance square root filter is often considerably more burdensome computationally, although it too becomes competitive if the measurement dimension m is very large.

Chandrasekhar-type square root algorithms have also been reported in the literature [25]. However, these have been omitted because they do not appear to be computationally competitive with algorithms presented herein for the nonstationary linear discrete-time estimation problem.

REFERENCES

1. Agee, W. S., and Turner, R. H., "Triangular Decomposition of a Positive Definite Matrix Plus a Symmetric Dyad with Applications to Kalman Filtering," Mathematical Services Branch, Analysis and Computation Division, Tech. Rep. 38, White Sands Missile Range, 1972.
2. Andrews, A., "A Square Root Formulation of the Kalman Covariance Equations," *AIAA J.* **6**(6), 1165–1166 (1968).
3. Bellantoni, J. F., and Dodge, K. W., "A Square Root Formulation of the Kalman–Schmidt Filter," *AIAA J.* **5** (7), 1309–1314 (1967).
4. Bierman, G. J., "A Comparison of Discrete Linear Filtering Algorithms," *IEEE Trans. Aerospace and Electron. Systems* **AES-9** (1) 28–37 (1973).
5. Bierman, G. J., "Sequential Square Root Filtering and Smoothing of Discrete Linear Systems," *Automatica* **10**, 147–158 (1974).
6. Bierman, G. J., "Measurement Updating Using the U-D Factorization," *Proc. IEEE Control and Decision Conf., Houston, Texas* 337–346 (1975).
7. Bierman, G. J., and Thornton, C. L., "Numerical Comparison of Kalman Filter Algorithms: Orbit Determination Case Study," *Automatica* **13**, 23–35 (1977).
8. Bierman, G. J., *Factorization Methods for Discrete Sequential Estimation.* Academic Press, New York, 1977.
9. Björck, A., "Solving Linear Least Squares Problems by Gram–Schmidt Orthogonalization," *BIT* **7**, 1–21 (1967).
10. Businger, P., and Golub, G. H., "Linear Least Squares Solution by Householder Transformations," *Numer. Math.* **7**, 269–276 (1965).

11. Carlson, N. A., "Fast Triangular Formulation of the Square Root Filter," *AIAA J.* **11** (9), 1259–1265 (1973).
12. Dyer, P., and McReynolds, S., "Extension of Square-Root Filtering to Include Process Noise," *J. Optimization Theory Appl.* **3** (6), 444–459 (1969).
13. Fadeeva, V. N., *Computational Methods of Linear Algebra.* Dover, New York, 1959.
14. Fletcher, R., and Powell, M. J. D., "On the Modification of LDL^T Factorizations," *Math. Comp.* **28** (128), 1067–1087 (1974).
15. Gentleman, W. M., "Least Squares Computations by Givens Transformations without Square Roots," *J. Inst. Math. Appl.* **12**, 329–336 (1973).
16. Gill, P. E., Golub, G. H., Murray, W., and Saunders, M. A., "Methods for Modifying Matrix Factorizations," *Math. Comp.* **28** (126), 505–535 (1974).
17. Gill, P. E., Murray, W., and Saunders, M. A., "Methods for Computing and Modifying the LDV Factors of a Matrix," *Math. Comp.* **29** (132), 1051–1077 (1975).
18. Golub, G. H., "Numerical Methods for Solving Linear Least Squares Problems," *Numer. Math.* **7**, 206–216 (1965).
19. Hanson, R. J., and Lawson, C. L., "Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems," *Math. Comp.* **23** (108), 787–812 (1969).
20. Householder, A. S., *The Theory of Matrices in Numerical Analysis.* Blaisdell, Waltham, Massachusetts, 1964.
21. Jordan, T., "Experiments on Error Growth Associated with Some Linear Least Squares Procedures," *Math. Comp.* **22**, 579–588 (1968).
22. Kaminski, P. G., Bryson, A. E. Jr., and Schmidt, S. F., "Discrete Square Root Filtering: A Survey of Current Techniques," *IEEE Trans. Automatic Control* **AC-16** (6), 727–735 (1971).
23. Lawson, C. L., and Hanson, R. J., *Solving Linear Least Squares Problems.* Prentice-Hall, Englewood Cliffs, New Jersey, 1974.
24. Maybeck, P. S., "Solutions to the Kalman Filter Wordlength Problem: Square Root and U–D Covariance Factorizations," Tech. Rep. AFIT-TR-77-6, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, September 1977.
25. Morf, M., and Kailath, T., "Square-Root Algorithms for Least-Squares Estimation," *IEEE Trans. Automatic Control* **AC-20** (4), 487–497 (1975).
26. Potter, J. E., "W Matrix Augmentation," M.I.T. Instrumentation Laboratory Memo SGA 5-64, Cambridge, Massachusetts, January 1964.
27. Rice, J. R., "Experiments on Gram–Schmidt Orthogonalization," *Math. Comp.* **20**, 325–328 (1966).
28. Schmidt, S. F., "Computational Techniques in Kalman Filtering," *in Theory and Applications of Kalman Filtering*, AGARDograph 139, Chapter 3. London, 1970.
29. Thornton, C. L., and Bierman, G. J., "Gram–Schmidt Algorithms for Covariance Propagation," *Proc. IEEE Control and Decision Conf., Houston, Texas* 489–498 (1975).
30. Wampler, R. H., "A Report on the Accuracy of Some Widely Used Least Squares Computer Programs," *J. Amer. Statist. Assoc.* **65** (330), 549–565 (1970).
31. Wilkinson, J. H., *Rounding Errors in Algebraic Processes.* Prentice-Hall, Englewood Cliffs, New Jersey, 1963.

PROBLEMS

7.1 Generate both the lower and upper Cholesky square root matrices of

$$\text{(a)}\quad \mathbf{A} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 0.4 \end{bmatrix} \qquad \text{(b)}\quad \mathbf{A} = \begin{bmatrix} 3 & 1 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 2 & 2 & 3 & 2 \\ 1 & 1 & 2 & 4 \end{bmatrix}$$

7.2 Show the equivalence of Eqs. (7-16) and (7-17) for the Potter measurement update.

7.3 Let $\mathbf{S}(t_i^-)$ be given as in Example 7.6, but let the measurement at time t_i be described as

$$\begin{bmatrix} z_1(t_i) \\ z_2(t_i) \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1(t_i) \\ x_2(t_i) \end{bmatrix} + \begin{bmatrix} v_1(t_i) \\ v_2(t_i) \end{bmatrix}$$

with

$$\mathbf{R}(t_i) = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$

Show that incorporating $z_2(t_i, \omega_j)$ by a second iteration of the Potter algorithm upon the result of Example 7.6 yields a solution equivalent to the Andrews vector update given by (7-18).

7.4 Consider an application in which a three-state filter is to be updated with two measurements each sample time. Let

$$\mathbf{H} = \begin{bmatrix} 1 & -2 & -1 \\ -1 & -1 & 1 \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}, \qquad \mathbf{z}_i = \begin{bmatrix} z_{i1} \\ z_{i2} \end{bmatrix}$$

Explicitly convert this into a form compatible with iterative scalar measurement updating in a Potter covariance square root filter.

7.5 Let a system of interest be described by

$$\begin{bmatrix} x_1(t_{i+1}) \\ x_2(t_{i+1}) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t_i) \\ x_2(t_i) \end{bmatrix} + \begin{bmatrix} w_{d1}(t_i) \\ w_{d2}(t_i) \end{bmatrix}$$

$$\begin{bmatrix} z_1(t_i) \\ z_2(t_i) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1(t_i) \\ x_2(t_i) \end{bmatrix} + \begin{bmatrix} v_1(t_i) \\ v_2(t_i) \end{bmatrix}$$

where the a priori knowledge of $\mathbf{x}(t_0)$ is that it can be modeled as a Gaussian random vector with mean $\hat{\mathbf{x}}_0$ and covariance $\mathbf{P}_0$ given by

$$\hat{\mathbf{x}}_0 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, \qquad \mathbf{P}_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Let $\mathbf{w}_d(\cdot, \cdot)$ and $\mathbf{v}(\cdot, \cdot)$ be independent white Gaussian noises, each independent of $\mathbf{x}(t_0)$ and of mean zero, and having covariances

$$\mathbf{Q}_d = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} 2 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

Let $z_1(t_i) = 6$ and $z_2(t_i) = 4$.

Perform the first time propagation from t_0 to t_1, and the update at time t_1, for the Potter covariance square root filter. For the time propagation, use

(a) the matrix root sum squared (RSS) method,
(b) the modified Gram–Schmidt (MGS) technique,
(c) the Householder transformation algorithm.

7.6 Repeat the previous problem, but generate the Carlson filter (using both matrix RSS and MGS time propagations).

7.7 Generate the inverse covariance square root filter for the application given in Problem 7.5. Use both the MGS and Householder measurement updates.

7.8 Generate the U–D covariance factorization filter for the application given in Problem 7.5.

7.9 Generate the U–D factors of the matrices in Problem 7.1.

7.10 You are confronted with an engineer who tells you that he has been investigating square root filters as an alternative to conventional filters. The general problem requires a Gram–Schmidt orthogonalization or a Householder transformation, but the square root covariance filter does not

require such computation for the case of $\mathbf{Q} = \mathbf{0}$, or no dynamic noise. He suggests using very accurate measurements, modeled as essentially perfect, and then using a square root inverse covariance filter. This, he claims, will similarly not require a Gram–Schmidt or Householder algorithm in the filter computations. What is your response to him?

7.11 Explicitly derive the numerical results depicted in Examples 7.10 and 7.11.

7.12 Repeat the calculations of Example 7.12 for

(a) $n = 10, s = 10, m = 5$;
(b) $n = 15, s = 15, m = 2$;
(c) $n = 15, s = 10, m = 2$;
(d) $n = 20, s = 20, m = 2$;
(e) $n = 20, s = 20, m = 5$.

7.13 Often the state variables estimated by a filter can be classified in two general categories: primary system states (as position, velocity, and misalignment error states in navigation filters) and secondary states—for shaping filters to generate outputs affecting either state dynamics or measured outputs. Thus, the state can be partitioned as ($n_1 + n_2 = n$):

$$\mathbf{x}(\cdot,\cdot) = \begin{bmatrix} \mathbf{x}_1(\cdot,\cdot) \\ \hline \mathbf{x}_2(\cdot,\cdot) \end{bmatrix} \begin{matrix} \} n_1 \text{ primary states} \\ \} n_2 \text{ secondary states} \end{matrix}$$

Usually, the propagation of $\mathbf{x}_2(\cdot,\cdot)$ is independent of $\mathbf{x}_1$ so that the state transition matrix $\boldsymbol{\Phi}$ can be partitioned as

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\Phi}_{11} & \boldsymbol{\Phi}_{12} \\ \hline \mathbf{0} & \boldsymbol{\Phi}_{22} \end{bmatrix} \begin{matrix} \} n_1 \\ \} n_2 \end{matrix}$$

where $\boldsymbol{\Phi}_{11}$ is typically dense, $\boldsymbol{\Phi}_{22}$ is often diagonal, and $\boldsymbol{\Phi}_{12}$ typically contains one or two nonzero elements per column.

(a) Show that if $\mathbf{S}(t_i^+)$ is upper triangular, only the upper left n_1-by-n_1 partition of $[\boldsymbol{\Phi}\mathbf{S}(t_i^+)]$ is nontriangular and requires retriangularization in a Carlson-type filter. (This in fact motivated the choice of *upper* triangular forms for this filter.)

(b) Recalculate the entries of Table 7.1 as a function of n_1 and n_2 instead of n, assuming $\boldsymbol{\Phi}$ to be the form just described.

(c) Repeat the calculations of Example 7.12 for $n_1 = n_2 = 5$, and for the case of $n_1 = 3, n_2 = 7$.

7.14 In Monte Carlo analyses and other types of system simulations, it is often desired to generate samples of a discrete-time white Gaussian noise vector process, described by mean of zero and covariance

$$E\{\mathbf{w}_d(t_i)\mathbf{w}_d^{\mathrm{T}}(t_i)\} = \mathbf{Q}_d(t_i)$$

with $\mathbf{Q}_d(t_i)$ nondiagonal. Independent scalar white Gaussian noises can be simulated readily through use of pseudorandom codes, but the question remains, how does one properly provide for cross-correlations of the scalar noises?

(a) Let $\mathbf{w}_1(\cdot,\cdot)$ be a vector process composed of independent scalar white Gaussian noises of zero mean and unit variance:

$$E\{\mathrm{w}_{1k}(t_i)\} = 0, \qquad E\{\mathrm{w}_{1k}^2(t_i)\} = 1, \qquad k = 1, 2, \ldots, s$$

Show that

$$\mathbf{w}_d(t_i,\cdot) = \begin{cases} \sqrt{\mathbf{Q}_d(t_i)}\,\mathbf{w}_1(t_i,\cdot) \\ \text{or} \\ -\sqrt{\mathbf{Q}_d(t_i)}\,\mathbf{w}_1(t_i,\cdot) \end{cases} \quad \text{for all } i$$

properly models the desired characteristics.

(b) If $\mathbf{U}_Q(t_i)$ and $\mathbf{D}_Q(t_i)$ are the U–D factors of $\mathbf{Q}_d(t_i)$, show that

$$\mathbf{w}_d(t_i, \cdot) = \mathbf{U}_Q(t_i)\mathbf{w}_2(t_i, \cdot) \qquad \text{for all } i$$

also provides the desired model if $\mathbf{w}_2(\cdot, \cdot)$ is a vector process composed of independent scalar white Gaussian noises of mean zero and variance

$$E\{\mathbf{w}_{2k}^2(t_i)\} = D_{Qkk}$$

(c) Show the means of simulating $\mathbf{w}_d(\cdot, \cdot)$ if $\mathbf{Q}_d$ is given by

$$\mathbf{Q}_d = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}$$