Array Algorithms for H^2 and H^∞ Estimation *

Babak Hassibi, Thomas Kailath
Information Systems Laboratory
Stanford University

and Ali H. Sayed
Department of Electrical Engineering
University of California, Los Angeles

July 11, 1997

ABSTRACT Currently, the preferred method for implementing H^2 estimation algorithms is what is called the *array* form, and includes two main families: square-root array algorithms which are typically more stable than conventional ones, and fast array algorithms which, when the system is time-invariant, typically offer an order of magnitude reduction in the computational effort. Using our recent observation that H^∞ filtering coincides with Kalman filtering in Krein space, in this paper we develop array algorithms for H^∞ filtering. These can be regarded as natural generalizations of their H^2 counterparts, and involve propagating the indefinite square-roots of the quantities of interest. The H^∞ square-root and fast array algorithms both have the interesting feature that one does not need to explicitly check for the positivity conditions required for the existence of H^∞ filters. These conditions are built into the algorithms themselves so that an H^∞ estimator of the desired level exists if, and only if, the algorithms can be executed. However, since H^∞ square-root algorithms predominantly use J-unitary transformations, rather than the unitary transformations required in the H^2 case, further investigation is needed to determine the numerical behaviour of such algorithms.

*This work was supported in part by DARPA through the Department of Air Force under contract F49620-95-1-0525-P00001 and by the Joint Service Electronics Program at Stanford under contract DAAH04-94-G-0058-P00003.

1 Introduction

Ever since its inception in 1960, the celebrated Kalman filter has played a central role in estimation. The Kalman filter was first expressed as a recursive algorithm which required the propagation of a certain Riccati recursion. However, for several reasons, current implementations of the Kalman filter are most often expressed in (what is called) an *array* form, and do not propagate this Riccati recursion directly.

The first array forms for the Kalman filter are called the square-root array algorithms and were devised in the late 1960's [1, 2, 3]. These algorithms are closely related to the (so-called) QR method for solving systems of linear equations [4, 5, 6, 7], and have the properties of better conditioning, reduced dynamical range, and the use of orthogonal transformations, which typically lead to more stable algorithms.

Furthermore for constant systems, or in fact for systems where the time-variation is structured in a certain way, the Riccati recursions and the square-root recursions, both of which take $O(n^3)$ elementary computations (flops) per iteration (where n is the dimension of the state-space), can be replaced by more efficient fast recursions, which require only $O(n^2)$ flops per iteration [8, 9, 10]. These recursions are analogous to certain equations invented in 1943-47 by the astrophysicists Ambartsumian [11] and Chandrasekhar (see [8]). The resemblance is much more remote in the discrete-time case, though the name was carried over (see [9, 12, 10]). These algorithms are also closely related to the concept of displacement structure [13, 14].

The conventional Kalman filter (and its variants) are H^2-optimal estimators in the sense that they minimize the least-mean-square estimation errors. Recently, on the other hand, there has been growing interest in worst-case, or H^∞, estimation where the goal is to minimize (or, in the suboptimal case, bound) the worst-case energy gain from the disturbances to the estimation errors (see *e.g.*, [15, 16, 17, 18]). The rationale for considering H^∞ estimation is that, unlike the H^2 case, the resulting estimators will have more robust performance in the face of model uncertainty and lack of statistical knowledge on the exogenous signals.

The resulting H^∞ estimators involve propagating a Riccati recursion and bear a striking resemblance to the conventional Kalman filter. In a series of papers [19, 20], we have recently shown that H^∞ filters are indeed Kalman filters, provided we set up estimation problems, not in the usual Hilbert space of random variables, but in an indefinite-metric (or so-called Krein) space. This observation leads to a unified approach to H^2 and H^∞ theory and has various further ramifications. One major bonus of this unification is that it shows a way to apply to the H^∞ setting many of the results developed for Kalman filtering and LQG control over the last three decades.

One immediate fall-out is that it allows one to generalize the square-root and fast array algorithms of H^2 estimation to the H^∞ setting. This is the topic of the current paper. The hope is that the resulting H^∞ array algorithms will be more attractive for actual implementations of H^∞ filters and controllers. As we shall see, the H^∞ array algorithms have several interesting features. They involve propagating (indefinite) square-roots of the quantities of interest and guarantee that the proper inertia of these quantities is preserved. Furthermore, the condition required for the existence of the H^∞ filters is built into the algorithms — if the algorithms can be carried out, then an H^∞ filter of the desired level exists, and if they cannot be executed then such H^∞ filters do not exist. This can be a significant simplification of the existing algorithms.

The remainder of the paper is organized as follows. The conventional square-root array algorithms are introduced in Sec. 2 along with some of their properties. In Sec. 3 we begin the development of the H^∞ square-root array algorithms and mention why they are natural extensions of their conventional counterparts. We initially encounter some difficulties in generalizing these arrays to the Krein space setting, and in order to alleviate them we then invoke the concept of indefinite square-roots, and study the inertia properties of the Gramian matrices in the H^∞ filtering problem in some detail. These inertia properties are related to the triangularization of matrices via J-unitary transformations and are crucial for the development of the H^∞ array algorithms. Finally, the general form of the H^∞ a posteriori and a priori filters are given in Sec. 3.4 and the (so-called) central filters in Sec. 3.5.

The conventional fast recursions, along with several of their properties, are given in Sec. 4. Sec. 5.1 extends these recursions to the H^∞ setting, and Sec. 5.2 gives the corresponding central H^∞ filters.

The Appendix deals with the elementary unitary and hyperbolic transformations that are needed for the implementation of H^2 and H^∞ array algorithms. In particular, the three families of Householder, Givens, and fast Givens transformations are studied.

In closing this introduction we should note that there are many variations to the conventional square-root and fast array algorithms, of which only a few have been considered here. In particular, there are those that arise in finite-horizon control problems. However, the approach adopted here is of sufficient generality that it should allow a reader to extend any other variation of these algorithms to the H^∞ setting.

A brief remark on the notation. To avoid confusion between the various gain matrices used in this paper, we shall employ the following convention: $K_{p,i}$ will denote the gain matrix in the usual Krein space, or conventional, Kalman filter, $K_{f,i}$ will denote the gain matrix in the corresponding filtered form of the Kalman filter, and $K_{s,i}$ and $K_{a,i}$ will denote the gain matrices in the H^∞ a posteriori and a priori filters, respectively.

2 H^2 Square-Root Array Algorithms

In state-space estimation problems we begin with a (possibly) time-varying state-space model of the form

$$\begin{cases} x_{j+1} &= F_j x_j + G_j u_j, \qquad x_0 \\ y_j &= H_j x_j + v_j \end{cases} \tag{2.1}$$

where $F_j \in \mathcal{C}^{n\times n}$, $G_j \in \mathcal{C}^{n\times m}$ and $H_j \in \mathcal{C}^{p\times n}$, and where the $\{u_j, v_j\}$ are disturbances whose nature depends on the criterion being used, and where the $\{y_j\}$ are the observed outputs. We are typically interested in obtaining estimates of some given linear combination of the states, say $s_i \triangleq L_i x_i$ ($L_i \in \mathcal{C}^{q\times n}$), and, most frequently, filtered and predicted estimates, denoted by $\hat{s}_{j|j}$ and $\hat{s}_j$, that use the observations $\{y_k, k \le j\}$ and $\{y_k, k < j\}$, respectively.

2.1 Kalman Filtering

In conventional Kalman filtering the $\{x_0, u_j, v_j\}$ are assumed to be zero-mean random variables with

$$E\begin{bmatrix} x_0 \\ u_i \\ v_i \end{bmatrix}\begin{bmatrix} x_0^* & u_j^* & v_j^* \end{bmatrix} = \begin{bmatrix} \Pi_0 & 0 & 0 \\ 0 & Q_i\delta_{ij} & \\ 0 & 0 & R_i\delta_{ij} \end{bmatrix} \ge 0.$$

Moreover, the output covariance of (2.1) is assumed to be positive-definite, *i.e.* $R_y > 0$, where $[R_y]_{ij} = E y_i y_j^*$.[1]

Once the $\{x_0, \{u_i, v_i\}\}$ are random, the same will be true of the states, outputs and desired signals, $\{x_i, y_i, s_i\}$. In this setting, the H^2 criterion states that the <u>linear</u> estimates $\hat{s}_{j|j}$ and $\hat{s}_j$ should be found so as to minimize the expected estimation error energies,

$$\sum_{j=0}^{i}(s_j - \hat{s}_{j|j})^*(s_j - \hat{s}_{j|j}) \quad \text{and} \quad \sum_{j=0}^{i}(s_j - \hat{s}_j)^*(s_j - \hat{s}_j), \tag{2.2}$$

respectively.

Using this criterion, the predicted and filtered estimates are given by $\hat{s}_j = L_j\hat{x}_j$ and $\hat{s}_{j|j} = L_j\hat{x}_{j|j}$, respectively, where $\hat{x}_j$ satisfies the predicted form of the conventional Kalman filter recursions

$$\hat{x}_{j+1} = F_j\hat{x}_j + K_{p,j}(y_j - H_j\hat{x}_j), \qquad \hat{x}_0 = 0 \tag{2.3}$$

[1] One way to ensure the positive definiteness of the output covariance, R_y, is to assume that the measurement noise covariance matrix is full rank, *i.e.*, $R_i > 0$. This is often a very reasonable assumption.

and $\hat{x}_{j|j}$ satisfies its filtered form,

$$\hat{x}_{j+1|j+1} = F_j \hat{x}_{j|j} + K_{f,j+1}(y_{j+1} - H_{j+1} F_j \hat{x}_{j|j}), \qquad \hat{x}_{-1|-1} = 0. \tag{2.4}$$

Here $\hat{x}_j$ denotes the predicted estimate of x_j, given $\{y_0, \ldots, y_{j-1}\}$, and $\hat{x}_{j|j}$ denotes its filtered estimate, given $\{y_0, \ldots, y_j\}$. The gain matrices $K_{p,j}$ and $K_{f,j}$ can be computed in several ways. The most common method uses a certain Riccati recursion, viz.,

$$K_{f,j} = P_j H_j R_{e,j}^{-1} \quad , \quad K_{p,j} = F_j K_{f,j} \quad , \quad R_{e,j} = R_j + H_j P_j H_j^* \tag{2.5}$$

where P_j satisfies the Riccati recursion,

$$P_{j+1} = F_j P_j F_j^* + G_j Q_j G_j^* - K_{p,j} R_{e,j} K_{p,j}^*, \qquad P_0 = \Pi_0. \tag{2.6}$$

We should also mention that the invertibility of the $R_{e,j}$ is guaranteed by the positivity assumption on R_y.

2.2 Square-Root Arrays

The matrix P_j appearing in the Riccati recursion (2.6) has the physical meaning of being the variance of the state prediction error, $\tilde{x}_j = x_j - \hat{x}_j$, and therefore has to be positive (semi)definite. Round-off errors can cause a loss of positive-definiteness, thus throwing all the obtained results into doubt. For this, and other reasons (reduced dynamic range, better conditioning, more stable algorithms, etc.) attention has moved in the Kalman filtering community to the so-called square-root array (or factorized) estimation algorithms [1, 2] that propagate square-root factors of P_j, *i.e.* a matrix, $P_j^{1/2}$ say, with positive diagonal entries, and such that

$$P_j = P_j^{1/2}(P_j^{1/2})^* = P_j^{1/2} P_j^{*/2}.$$

Square roots can be similarly defined for the system covariances $\{Q_j, R_j\}$. Then it is in fact not hard to show the following.

Find any orthogonal transformation, say Θ_j,[2] that triangularizes the pre-array shown below

$$\begin{bmatrix} R_j^{1/2} & H_j P_j^{1/2} & 0 \\ 0 & F_j P_j^{1/2} & G_j Q_j^{1/2} \end{bmatrix} \Theta_j = \begin{bmatrix} X & 0 & 0 \\ Y & Z & 0 \end{bmatrix}. \tag{2.7}$$

The resulting post-array entries can be determined, by taking squares and using the orthogonality of Θ_j, to obey

$$XX^* \quad = \quad R_j + H_j P_j H_j^* = R_{e,j}$$

[2] By an orthogonal transformation, Θ, we mean one for which, $\Theta\Theta^* = \Theta^*\Theta = I$.

$$\begin{aligned} YX^* &= F_jP_jH_j^* \\ ZZ^* &= F_jP_jF_j^* + G_jQ_jG_j^* - YY^* \\ &= F_jP_jF_j^* + G_jQ_jG_j^* - YX^*(XX^*)^{-1}XY^* \\ &= F_jP_jF_j^* + G_jQ_jG_j^* - F_jP_jH_j^*R_{e,j}^{-1}H_jP_jF_j^* \\ &= P_{j+1}. \end{aligned}$$

Therefore we can identify

$$Z = P_{j+1}^{1/2} \quad , \quad X = R_{e,j}^{1/2} \tag{2.8}$$

and also

$$Y = F_jP_jH_j^*R_{e,j}^{-*/2} = K_{p,j}R_{e,j}^{1/2}. \tag{2.9}$$

Thus the square-root algorithm not only propagates the square-roots of the Riccati variable, P_j, but also gives us quantities useful for the state estimation recursion

$$\hat{x}_{j+1} = F_j\hat{x}_j + K_{p,j}(y_j - H_j\hat{x}_j). \tag{2.10}$$

The unitary transformation Θ_j is highly nonunique and can be computed in many ways, the simplest ones being to construct it as a sequence of elementary (Givens or plane) rotations nulling one entry at a time in the pre-array, or as a sequence of elementary (Householder) reflections nulling out a block of entries in each row. We refer to [21, 5, 7] and the Appendix for more details.[3] The numerical advantages of the square-root transformations arise from the length preserving properties of unitary transformations, and from the fact that the dynamic range of the entries in $P_j^{1/2}$ is roughly the square-root of the dynamic range of those in P_j. Moreover, regular computational (systolic) arrays can be designed to implement sequences of elementary unitary transformations [22].

A final result will be useful before we summarize the above discussion in a theorem. Any unitary transformation Θ_j that triangularizes the pre-array in (2.7) also gives the, readily checked, identity,

$$\begin{bmatrix} -R_j^{-1/2}y_j & P_j^{-1/2}\hat{x}_j & 0 \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j}^{-1/2}e_j & P_{j+1}^{-1/2}\hat{x}_{j+1} & \times \end{bmatrix} \tag{2.11}$$

where $\times$ denotes an entry whose exact form is not relevant at the moment. This gives us an alternative way of computing the state estimates via $\hat{x}_{j+1} = P_{j+1}^{1/2}P_{j+1}^{-1/2}\hat{x}_{j+1}$, rather than the state estimate equation (2.10).

We can summarize the above discussion as follows.

---

[3]The above square-root method is closely related to the QR (factorization) method for solving systems of linear equations.

Algorithm 1 (Conventional Square-Root Algorithm) *The gain matrix $K_{p,j}$ necessary to obtain the state estimates in the conventional Kalman filter (2.3)-(2.5)*

$$\hat{x}_{j+1} = F_j \hat{x}_j + K_{p,j}(y_j - H_j \hat{x}_j), \qquad \hat{x}_0 = 0,$$

can be updated as follows

$$\begin{bmatrix} R_j^{1/2} & H_j P_j^{1/2} & 0 \\ 0 & F_j P_j^{1/2} & G_j Q_j^{1/2} \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j}^{1/2} & 0 & 0 \\ K_{p,j} R_{e,j}^{1/2} & P_{j+1}^{1/2} & 0 \end{bmatrix}, \tag{2.12}$$

where Θ_j is any unitary matrix that triangularizes the above pre-array. The algorithm is initialized with $P_0 = \Pi_0$.

Note that the quantities necessary to update the square-root array, and to calculate the state estimates, may all be found from the triangularized post-array.

It will also be useful to quote the filtered form of the square-root array algorithm, that can be verified in a fashion similar to what was done above.

Algorithm 2 (Conventional Square-Root Alg. - Filtered Form) . *The gain matrix $K_{f,j}$ necessary to obtain the state estimates in the filtered form of the conventional Kalman filter*

$$\hat{x}_{j+1|j+1} = F_j \hat{x}_{j|j} + K_{f,j+1}(y_{j+1} - H_{j+1} F_j \hat{x}_{j|j}), \qquad \hat{x}_{-1|-1} = 0,$$

can be updated as follows

$$\begin{bmatrix} R_j^{1/2} & H_j P_j^{1/2} \\ 0 & P_j^{1/2} \end{bmatrix} \Theta_j^{(1)} = \begin{bmatrix} R_{e,j}^{1/2} & 0 \\ K_{f,j} R_{e,j}^{1/2} & P_{j|j}^{1/2} \end{bmatrix}, \tag{2.13}$$

$$\begin{bmatrix} F_j P_{j|j}^{1/2} & G_j Q_j^{1/2} \end{bmatrix} \Theta_j^{(2)} = \begin{bmatrix} P_{j+1}^{1/2} & 0 \end{bmatrix} \tag{2.14}$$

where $\Theta_j^{(1)}$ and $\Theta_j^{(2)}$ are any unitary matrices that triangularize the above pre-arrays. The algorithm is initialized with $P_0 = \Pi_0$.

3 H^∞ Square-Root Array Algorithms

We now turn our attention to H^∞ filtering. Our goal here is to show that it is possible to construct square-root array implementations of H^∞ filters, similar to what was done in the aforementioned H^2 case.

3.1 H^∞ Filtering

Consider once more the state-space model (2.1). In the H^∞ approach it is assumed that disturbances $\{x_0, \{u_i\}, \{v_i\}\}$ are unknown, but *nonrandom*. One can therefore not speak of expected values, or attempt to minimize the average estimation error energy. Instead one can look at the energy gain from the unknown disturbances $\{\Pi_0^{-1/2}x_0, \{u_j\}_{j=0}^i, \{v_j\}_{j=0}^i\}$ to the filtered and predicted errors $\{s_j - \hat{s}_{j|j}\}_{j=0}^i$ and $\{s_j - \hat{s}_j\}_{j=0}^i$, respectively, *i.e.*,

$$\frac{\sum_{j=0}^i (s_j - \hat{s}_{j|j})^*(s_j - \hat{s}_{j|j})}{x_0^*\Pi_0{}^{-1}x_0 + \sum_{j=0}^i u_j^* u_j + \sum_{j=0}^i v_j^* v_j}\ , \tag{3.1}$$

and

$$\frac{\sum_{j=0}^i (s_j - \hat{s}_j)^*(s_j - \hat{s}_j)}{x_0^*\Pi_0{}^{-1}x_0 + \sum_{j=0}^i u_j^* u_j + \sum_{j=0}^i v_j^* v_j}\ . \tag{3.2}$$

Here Π_0 is a positive definite weighting matrix. It is quite clear that if the ratios in (3.1-3.2) are small then the estimators perform well, and vice-versa. However, the problem with these ratios is that they depend on the disturbances $\{x_0, \{u_i\}, \{v_i\}\}$, which are not known. To overcome this problem, we can consider their worst-cases, *i.e.*,

$$\sup_{x_0,\{u_j\},\{v_j\}} \frac{\sum_{j=0}^i (s_j - \hat{s}_{j|j})^*(s_j - \hat{s}_{j|j})}{x_0^*\Pi_0{}^{-1}x_0 + \sum_{j=0}^i u_j^* u_j + \sum_{j=0}^i v_j^* v_j}\ , \tag{3.3}$$

and

$$\sup_{x_0,\{u_j\},\{v_j\}} \frac{\sum_{j=0}^i (s_j - \hat{s}_{j|j})^*(s_j - \hat{s}_{j|j})}{x_0^*\Pi_0{}^{-1}x_0 + \sum_{j=0}^i u_j^* u_j + \sum_{j=0}^i v_j^* v_j}\ . \tag{3.4}$$

The goal in H^∞ estimation is to bound this worst-case energy gain, and the claim is that the resulting estimators will be robust with respect to disturbance variation, since no statistical assumptions are being made about the disturbances, and since we are safe-guarding against the worst-case scenario. However, the resulting estimators may be over-conservative.

Here we quote the standard solution to the so-called suboptimal H^∞ estimation problems using the notation of [19, 20]. (See also [16, 23, 17].)

Theorem 1 (Suboptimal H^∞ A Posteriori Filter) *Given $\gamma > 0$, an H^∞ a posteriori filter that achieves*

$$\sup_{x_0,\{u_j\},\{v_j\}} \frac{\sum_{j=0}^i (s_j - \hat{s}_{j|j})^*(s_j - \hat{s}_{j|j})}{x_0^*\Pi_0{}^{-1}x_0 + \sum_{j=0}^i u_j^* u_j + \sum_{j=0}^i v_j^* v_j} < \gamma^2, \tag{3.5}$$

exists if, and only if, the matrices

$$R_j = \begin{bmatrix} I_p & 0 \\ 0 & -\gamma^2 I_q \end{bmatrix} \ \textit{and}\ R_{e,j} = \begin{bmatrix} I_p & 0 \\ 0 & -\gamma^2 I_q \end{bmatrix} + \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} \tag{3.6}$$

have the same inertia[4] *for all* $0 \leq j \leq i$*, where* $P_0 = \Pi_0$ *and* P_j *satisfies the Riccati recursion*

$$P_{j+1} = F_j P_j F_j^* + G_j G_j^* - F_j P_j \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} R_{e,j}^{-1} \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j F_j^*. \quad (3.7)$$

If this is the case, then all possible H^∞ *a posteriori estimators that achieve (3.5) are given by those* $\hat{s}_{j|j} = \mathcal{F}_{f,j}(y_0, \ldots y_j)$ *that satisfy*

$$\sum_{j=0}^{k} \begin{bmatrix} y_j - H_j\hat{x}_j \\ \hat{s}_{j|j} - L_j\hat{x}_j \end{bmatrix}^* R_{e,j}^{-1} \begin{bmatrix} y_j - H_j\hat{x}_j \\ \hat{s}_{j|j} - L_j\hat{x}_j \end{bmatrix} > 0, \quad 0 \leq k \leq i \quad (3.8)$$

where $\hat{x}_j$ *is given by the recursion*

$$\hat{x}_{j+1} = F_j\hat{x}_j + K_{p,j} \begin{bmatrix} y_j - H_j\hat{x}_j \\ \hat{s}_{j|j} - L_j\hat{x}_j \end{bmatrix}, \quad \hat{x}_0 = 0 \quad (3.9)$$

and $K_{p,j} = F_j P_j \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} R_{e,j}^{-1}$.

One important special choice that guarantees (3.8) is the so-called *central* a posteriori filter,

$$\hat{s}_{j|j} = L_j \underbrace{\left(\hat{x}_j + P_j H_j^* (I_p + H_j P_j H_j^*)^{-1}(y_j - H_j\hat{x}_j)\right)}_{\hat{x}_{j|j}}. \quad (3.10)$$

With this choice of estimate, the recursion (3.9) can be used to obtain the following recursion for $\hat{x}_{j|j}$

$$\hat{x}_{j+1|j+1} = F_j\hat{x}_{j|j} + K_{s,j+1}(y_{j+1} - H_{j+1}F_j\hat{x}_{j|j}), \quad \hat{x}_{-1|-1} = 0 \quad (3.11)$$

with $K_{s,j} = P_j H_j^*(I_p + H_j P_j H_j^*)^{-1}$. Moreover, the central estimate is simply $\hat{s}_{j|j} = L_j\hat{x}_{j|j}$.

Theorem 2 (Suboptimal H^∞ A Priori Filter) *Given* $\gamma > 0$*, an* H^∞ *a priori filter that achieves*

$$\sup_{x_0, \{u_j\}, \{v_j\}} \frac{\sum_{j=0}^{i}(s_j - \check{s}_j)^*(s_j - \check{s}_j)}{x_0^*\Pi_0^{-1}x_0 + \sum_{j=0}^{i} u_j^* u_j + \sum_{j=0}^{i} v_j^* v_j} < \gamma^2, \quad (3.12)$$

exists if, and only if, <u>*all leading submatrices*</u> *of the matrices*

$$R_j = \begin{bmatrix} -\gamma^2 I_q & 0 \\ 0 & I_p \end{bmatrix} \text{ and } R_{e,j} = \begin{bmatrix} -\gamma^2 I_q & 0 \\ 0 & I_p \end{bmatrix} + \begin{bmatrix} L_j \\ H_j \end{bmatrix} P_j \begin{bmatrix} L_j^* & H_j^* \end{bmatrix} \quad (3.13)$$

---

[4] By the inertia of a Hermitian matrix, we mean the number of its positive, negative and zero eigenvalues.

have the same inertia for all $0 \le j \le i$, *where* P_j *is the same as in Theorem 1.*

If this is the case, then all possible H^∞ *a priori estimators that achieve (3.12) are given by those* $\hat{s}_j = \mathcal{F}_{p,j}(y_0, \ldots y_{j-1})$ *that satisfy*

$$\begin{array}{l}\sum_{j=0}^{k-1} \begin{bmatrix} \hat{s}_{j|j} - L_j\hat{x}_j \\ y_j - H_j\hat{x}_j \end{bmatrix}^* R_{e,j}^{-1} \begin{bmatrix} \hat{s}_{j|j} - L_j\hat{x}_j \\ y_j - H_j\hat{x}_j \end{bmatrix} \\ -(\hat{s}_{k|k} - L_k\hat{x}_k)^*(\gamma^2 I_q - L_k P_k L_k^*)^{-1} > 0,\end{array} \quad 0 \le k \le i \tag{3.14}$$

where $\hat{x}_j$ *is given by the recursion*

$$\hat{x}_{j+1} = F_j\hat{x}_j + K_{p,j}\begin{bmatrix} \hat{s}_j - L_j\hat{x}_j \\ y_j - H_j\hat{x}_j \end{bmatrix}, \quad \hat{x}_0 = 0 \tag{3.15}$$

and $K_{p,j} = F_j P_j \begin{bmatrix} L_j^* & H_j^* \end{bmatrix} R_{e,j}^{-1}$.

Once more, an important special choice that guarantees (3.14) is the so-called *central* a priori filter,

$$\hat{s}_j = L_j\hat{x}_j. \tag{3.16}$$

With this choice of estimate, the recursion (3.15) can be rewritten as

$$\hat{x}_{j+1} = F_j\hat{x}_j + K_{a,j}(y_j - F_j\hat{x}_j), \quad \hat{x}_0 = 0 \tag{3.17}$$

with $K_{a,j} = \tilde{P}_j H_j^*(I_p + H_j\tilde{P}_j H_j^*)^{-1}$ and $\tilde{P}_j = \left[P_j^{-1} - \gamma^{-2}L_j^*L_j\right]^{-1}$.

Remark: Note that the existence condition for a priori level-γ H^∞ filters is more stringent than the existence condition for a posteriori filters, since the latter requires that the matrices R_j and $R_{e,j}$ have the same inertia, whereas the former requires that all leading submatrices of R_j and $R_{e,j}$ have the same inertia. This, of course, makes perfect sense, since a posteriori filters have access to one additional measurement and should therefore outperform a priori ones.

We can also more explicitly show that one condition is more stringent than the other. To this end, let us remark that, when $\Pi_0 > 0$, and when a solution to the level-γ H^∞ estimation problem exists, it can be shown that $P_j > 0$ for all j (see *e.g.*, [19]). In this case, using a simple Schur complement argument, the inertia condition for the a posteriori problem of Theorem 1 (*cf.*, Eq. (3.6)) becomes,

$$I_p + H_j P_j H_j^* > 0 \quad \text{and} \quad -\gamma^2 I_q + L_j(P_j^{-1} + H_j^* H_j)^{-1} L_j^* < 0. \tag{3.18}$$

Since $P_j > 0$, the first condition is superfluous so that the existence condition becomes

$$\gamma^2 I_q > L_j(P_j^{-1} + H_j^* H_j)^{-1} L_j^*. \tag{3.19}$$

Likewise, using a similar Schur complement argument, the inertia condition for the a priori problem of Theorem 2 (*cf.*, Eq. (3.13)) becomes,

$$-\gamma^2 I_q + L_j P_j L_j^* < 0 \quad \text{and} \quad I_p + H_j(P_j^{-1} - \gamma^{-2} L_j^* L_j)^{-1} H_j^* > 0. \quad (3.20)$$

Now when $P_j > 0$ and $-\gamma^2 I_q + L_j P_j L_j^* < 0$, it follows that $P_j^{-1} - \gamma^{-2} L_j^* L_j > 0$. Therefore the second condition is superfluous and the existence condition becomes

$$\gamma^2 I_q > L_j P_j L_j^*. \quad (3.21)$$

Since $(P_j^{-1} + H_j^* H_j)^{-1} \leq P_j$, it readily follows that condition (3.21) is more stringent than condition (3.19)

3.2 A Krein Space Formulation

The central H^∞ a priori and a posteriori filters of Eqs. (3.17) and (3.11) look very similar to their Kalman filter counterparts (2.3) and (2.4). Indeed the only difference is that the H^∞ Riccati recursion (3.7) differs from the Kalman filter Riccati recursion (2.6), since:

- We have indefinite "covariance" matrices, $\begin{bmatrix} I & 0 \\ 0 & -\gamma_f^2 I \end{bmatrix}$.
- The L_i (of the quantity to be estimated) enters the Riccati equation, (3.7).
- We have an additional condition, (3.6), that must be satisfied for the filter to exist; in the Kalman filter problem the L_i would not appear, and the P_i would be positive semidefinite, so that (3.6) is immediate.

Despite these differences, we have recently shown that these H^∞ filters can in fact be obtained as certain Kalman filters, not in an H^2 (Hilbert) space, but in a certain indefinite vector space, called a Krein space [19, 20, 24]. The indefinite "covariances" and the appearance of L_i in the Riccati recursions are all easily explained in this framework. The additional condition (3.6) will be seen to arise from the fact that in Krein space, unlike the usual Hilbert space context, quadratic forms need not always have minima or maxima, unless certain additional conditions are met.

We shall not go into the details of estimation in indefinite-metric spaces here. Instead, we shall use the observation that H^∞ filtering coincides with Kalman filtering in Krein space as a guideline to generalize the square-root array algorithms of Sec. 2 to the H^∞ setting.

To this end, recall the conventional square-root array algorithm of Sec. 2,

$$\begin{bmatrix} R_j^{1/2} & H_j P_j^{1/2} & 0 \\ 0 & F_j P_j^{1/2} & G_j Q_j^{1/2} \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j}^{1/2} & 0 & 0 \\ K_{p,j} R_{e,j}^{1/2} & P_{j+1}^{1/2} & 0 \end{bmatrix}. \quad (3.22)$$

The first problem that occurs if one wants to extend the square-root array algorithm to the Krein space setting (of which the H^∞ filtering problem is a special case) is that the matrices R_i, Q_i, P_i and $R_{e.i}$ are in general indefinite and square-roots may not exist. To overcome this problem we employ the notion of an indefinite square-root, as defined below.

Definition 1 (Indefinite Square-Roots) *Suppose A is an arbitrary Hermitian matrix. Let the signature matrix S (*i.e.* a diagonal matrix with diagonal elements either $+1$ or -1) represent the number of positive and negative eigenvalues of A. Then $A^{1/2}$ will be called an indefinite square-root of A if, and only, if*

$$A = A^{1/2} S A^{*/2}.$$

Note that when A is non-negative definite, then $S = I$ and $A^{1/2}$ is the conventional square-root.

In the Krein space case, however, R_i, Q_i, P_i and $R_{e.i}$ may all have arbitrary inertia, *i.e.*

$$\begin{cases} R_i = R_i^{1/2} S_i^{(1)} R_i^{*/2} & , \quad Q_i = Q_i^{1/2} S_i^{(2)} Q_i^{*/2} \\ P_i = P_i^{1/2} S_i^{(3)} P_i^{*/2} & , \quad R_{e,i} = R_{e,i}^{1/2} S_i^{(4)} R_{e,i}^{*/2} \end{cases}$$

for arbitrary signature matrices $S_i^{(k)}$, $k = 1, 2, 3, 4$. It is thus not obvious how to incorporate all these different time-variant signature matrices into a square-root array algorithm of the type (3.22). Although this can be done in the general case, by either introducing non-Hermitian factorizations of the Gramians, or by keeping track of the inertia, we do not need to pursue such generalities here. The reason is that, as it turns out, in the special case of the H^∞ estimation problems that we have been studying, the Gramians satisfy certain inertia properties that allow us to extend the algorithm of (3.22) in a very natural way.

Indeed when a solution to the H^∞ filtering problem exists, we know from Theorems 1 and 2 that $P_i \geq 0$, and that $R_{e,i}$ and R_i have the same inertia. Moreover, $Q_i = I_m > 0$ and $R_i = \begin{bmatrix} I_p & 0 \\ 0 & -\gamma^2 I_q \end{bmatrix}$ have constant inertia (and thus so does $R_{e,i}$) so that we may write,

$$R_i = R_i^{1/2} J R_i^{*/2}, \; Q_i = Q_i^{1/2} Q_i^{*/2}, \; P_i = P_i^{1/2} P_i^{*/2}, \; R_{e,i} = R_{e,i}^{1/2} J R_{e,i}^{*/2} \tag{3.23}$$

with

$$R_i^{1/2} = \begin{bmatrix} I_p & 0 \\ 0 & \gamma I_q \end{bmatrix} \quad \text{and} \quad J = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}. \tag{3.24}$$

This suggests that in the H^∞ filtering problem, the pre-array in (3.22) should be replaced by,

$$\left[\begin{array}{cc} \left[\begin{array}{cc} I_p & 0 \\ 0 & \gamma I_q \end{array}\right] & \left[\begin{array}{c} H_j \\ L_j \end{array}\right] P_j^{1/2} \quad 0 \\ 0 & F_j P_j^{1/2} \quad\quad G_j \end{array}\right]. \tag{3.25}$$

[Recall that, compared to the H^2 Riccati, in H^∞ Riccati recursion H_j is replaced by $\left[\begin{array}{c} H_j \\ L_j \end{array}\right]$.] Now in the H^2 case the pre-array in (3.22) was triangularized by a unitary transformation (or simply by a rotation). Since the H^2 estimation problem can be formulated in a Hilbert space, whereas the H^∞ estimation problem is most naturally formulated in a Krein space, it seems plausible that we should attempt to triangularize (3.25), not by an ordinary rotation, but by a hyperbolic rotation. To be more specific, we need to use a J-unitary transformation, as discussed below.

3.3 J-Unitary Transformations

Let us begin with the definition.

Definition 2 (J-unitary Matrices) *For any signature matrix, J, (a diagonal matrix with $+1$ and -1 diagonal elements) the matrix Θ will be called J-unitary if*

$$\Theta J \Theta^* = J. \tag{3.26}$$

Recall that unitary transformations (or ordinary rotations) preserve the length (or ordinary norm) of vectors. J-unitary transformations, on the other hand, preserve the (indefinite) J-norm of vectors. Indeed, if $b = a\Theta$, with Θ J-unitary, then

$$bJb^* = a\Theta J \Theta^* a^* = aJa^*.$$

The above discussions suggest that we should attempt to triangularize (3.25) via a J-unitary transformation, with

$$J = \left[\begin{array}{ccc} \left[\begin{array}{cc} I_p & 0 \\ 0 & -I_q \end{array}\right] & & \\ & I_n & \\ & & I_m \end{array}\right]. \tag{3.27}$$

Now it is well known that it is always possible to triangularize arrays using unitary transformations. But is this also true of J-unitary transformations? To see if this is the case, consider a simple example where we are given the (two-element) row vector

$$\left[\begin{array}{cc} a & b \end{array}\right],$$

and are asked to hyperbolically rotate it so that the resulting vector lies along the direction of the x-axis.[5] For the time being, assume that such a transformation can be found. Then we can write,

$$\begin{bmatrix} a & b \end{bmatrix} \Theta = \begin{bmatrix} c & 0 \end{bmatrix}, \tag{3.28}$$

where

$$\Theta J \Theta^* = J \quad \text{and} \quad J = \begin{bmatrix} 1 & \\ & -1 \end{bmatrix}. \tag{3.29}$$

Since Θ is J-unitary this implies that,

$$\begin{bmatrix} a & b \end{bmatrix} J \begin{bmatrix} a^* \\ b^* \end{bmatrix} = \begin{bmatrix} c & 0 \end{bmatrix} J \begin{bmatrix} c^* \\ 0 \end{bmatrix}, \tag{3.30}$$

or more explicitly,

$$|a|^2 - |b|^2 = |c|^2 \geq 0. \tag{3.31}$$

Thus, $\begin{bmatrix} a & b \end{bmatrix}$ must have nonnegative J-norm. In other words, if the given $\begin{bmatrix} a & b \end{bmatrix}$ has negative J-norm (*i.e.*, $|a|^2 - |b|^2 < 0$) then it is impossible to hyperbolically rotate it to lie along the x-axis. [This fact is shown in Fig. 1. As can be seen, standard rotations move the vector $\begin{bmatrix} a & b \end{bmatrix}$ along the circle, $a^2 + b^2 = constant$, whereas hyperbolic rotations move it along the hyperbola, $a^2 - b^2 = constant$. Thus while it is always possible to rotate $\begin{bmatrix} a & b \end{bmatrix}$ to lie along the x-axis, if $|a|^2 - |b|^2 < 0$ then it is impossible to do so with a *hyperbolic* rotation. Indeed hyperbolic rotations cannot move vectors from the positive to negative subspaces of a Krein space, or vice versa.]

Thus it is quite obvious that it is not always possible to triangularize arrays using J-unitary transformations. The precise condition follows.

Lemma 1 (J-unitary Matrices and Triangularization) *Let A and B be arbitrary $n \times n$ and $n \times m$ matrices, respectively, and suppose $J = \begin{bmatrix} S_1 & \\ & S_2 \end{bmatrix}$ where S_1 and S_2 are $n \times n$ and $m \times m$ signature matrices. Then $\begin{bmatrix} A & B \end{bmatrix}$ can be triangularized by a J-unitary transformation Θ as*

$$\begin{bmatrix} A & B \end{bmatrix} \Theta = \begin{bmatrix} L & 0 \end{bmatrix}$$

with L lower triangular, if and only if, all leading submatrices of

$$S_1 \quad \textit{and of} \quad AS_1A^* + BS_2B^*$$

have the same inertia.

[5]Note in standard (two-dimensional) Euclidean space this can always be done.

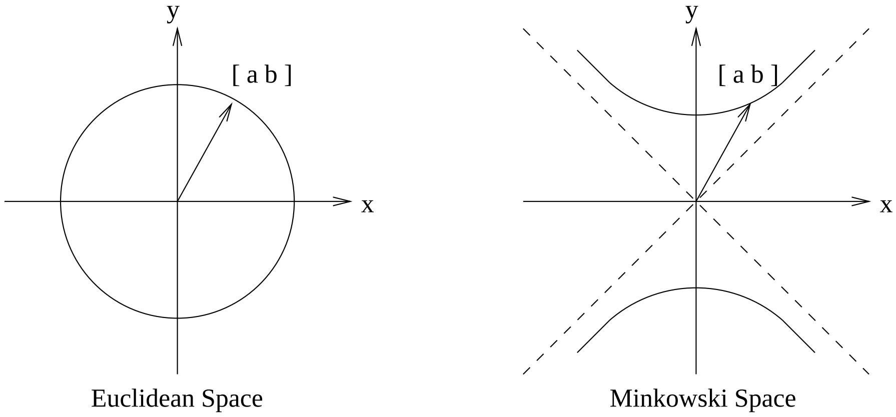

FIGURE 1. Standard rotations vs. hyperbolic rotations.

Proof: To prove one direction suppose there exists a J-unitary transformation Θ that triangularizes $\begin{bmatrix} A & B \end{bmatrix}$. Consider an arbitrary partitioning of A, B and L, *i.e.*

$$\begin{bmatrix} A^{(1)} & B^{(1)} \\ A^{(2)} & B^{(2)} \end{bmatrix} \quad , \quad \begin{bmatrix} L^{(1)} & 0 \\ L^{(2)} & 0 \end{bmatrix}$$

where $A^{(1)}$, $B^{(1)}$ and $L^{(1)}$ have r rows. Now

$$\begin{bmatrix} A^{(1)} & B^{(1)} \\ A^{(2)} & B^{(2)} \end{bmatrix} \underbrace{\Theta J \Theta^*}_{=J} \begin{bmatrix} A^{(1)*} & A^{(2)*} \\ B^{(1)*} & B^{(2)*} \end{bmatrix} = \begin{bmatrix} L^{(1)} & 0 \\ L^{(2)} & 0 \end{bmatrix} J \begin{bmatrix} L^{(1)*} & L^{(2)*} \\ 0 & 0 \end{bmatrix}$$

so that

$$\begin{bmatrix} A^{(1)} S_1 A^{(1)*} + B^{(1)} S_2 B^{(1)*} & \times \\ \times & \times \end{bmatrix} = \begin{bmatrix} L^{(1)} S_1 L^{(1)*} & \times \\ \times & \times \end{bmatrix} \qquad (3.32)$$

where $\times$ denotes irrelevant entries. Moreover, since L is lower triangular we have $L^{(1)} = \begin{bmatrix} L^{(11)} & 0 \end{bmatrix}$, where $L^{(11)}$ is lower triangular and $r \times r$. Thus if we denote by $S_1^{(1)}$ the leading $r \times r$ submatrix of S_1, equating the $(1,1)$ block entries in (3.32) yields

$$A^{(1)} S_1 A^{(1)*} + B^{(1)} S_2 B^{(1)*} = L^{(11)} S_1^{(1)} L^{(11)*} \qquad (3.33)$$

The LHS of the above equation is the leading $r \times r$ submatrix of $AS_1A^* + BS_2B^*$. Thus (3.33) shows that the leading $r \times r$ submatrices of $AS_1A^* + BS_2B^*$ and S_1 have the same inertia. Since r was arbitrary the same is true for all leading submatrices.

To prove the other direction, we assume that all leading submatrices of $AS_1A^* + BS_2B^*$ and S_1 have the same inertia. In particular the leading $(1,1)$ submatrices, so that

$$aS_1a^* + bS_2b^* = l_{11}^2 s \tag{3.34}$$

where a and b are the leading rows of A and B, s is the leading diagonal of S_1 and l_{11} is a scalar. Now define the vector

$$v = \begin{bmatrix} a & b \end{bmatrix} + l_{11}e_1$$

where $e_1 = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}$ is the first unit *row* vector. Consider the matrix[6]

$$\Theta_1 = I - 2\frac{Jv^*v}{vJv^*}.$$

A straightforward calculation shows that $\Theta_1 J \Theta_1^* = J$ so that Θ_1 is J-unitary. Moreover, another direct calculation shows that

$$\begin{bmatrix} a & b \end{bmatrix} \Theta_1 = l_{11}e_1 = \begin{bmatrix} l_{11} & 0 & \dots & 0 \end{bmatrix}.$$

We thus far have

$$\begin{bmatrix} A & B \end{bmatrix} \Theta_1 = \begin{bmatrix} \begin{bmatrix} l_{11} & 0 \end{bmatrix} & 0 \\ \begin{bmatrix} A_{21} & A_2 \end{bmatrix} & B_2 \end{bmatrix}. \tag{3.35}$$

Now if all leading submatrices of two given matrices have the same inertia, then their $(1,1)$ entries should have the same inertia and all leading submatrices of the *Schur complement* of their $(1,1)$ entries should have the same inertia. Now partition S_1 as

$$S_1 = \begin{bmatrix} s & \\ & S^{(1)} \end{bmatrix}$$

so that $S^{(1)}$ is the Schur complement of s in S_1. Likewise the Schur complement of the $(1,1)$ entry of $AS_1A^* + BS_2B^*$ is $A_2S^{(1)}A_2^* + B_2S_2B_2^*$ where A_2 and B_2 are defined in (3.35). Therefore all leading submatrices of $A_2S^{(1)}A_2^* + B_2S_2B_2^*$ and $S^{(1)}$ have the same inertia. We may now proceed as before to find a J-unitary matrix Θ_2 that rotates the first row of $\begin{bmatrix} A_{21} & A_2 & B_2 \end{bmatrix}$ to lie along the *second* unit vector. Continuing in a similar fashion will result in a J-unitary matrix $\Theta = \Theta_1\Theta_2 \dots \Theta_{n-1}$ that triangularizes $\begin{bmatrix} A & B \end{bmatrix}$.

■

[6] Θ_1 may be recognized as an elementary Householder reflection in the J-metric. See Appendix A.

3.4 Square-Root Array Algorithms

We are now in a position to apply the result of Lemma 1 to the triangularization of the pre-array (3.25) using a J-unitary transformation with J given by (3.27). In fact, we need only consider the condition for the triangularization of the first block row (since setting the block $(2,3)$ entry of the post array to be zero can always be done via a standard unitary transformation). Thus we need only consider triangularizing

$$\begin{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & \gamma I_q \end{bmatrix} & \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j^{1/2} \end{bmatrix}, \tag{3.36}$$

using a J-unitary transformation with

$$J = \begin{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} & \\ & I_n \end{bmatrix}. \tag{3.37}$$

From Lemma 1, the condition obviously is that all leading submatrices of J and

$$\underbrace{\begin{bmatrix} I_p & 0 \\ 0 & \gamma I_q \end{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & \gamma I_q \end{bmatrix}}_{R_j} + \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j^{1/2} P_j^{*/2} \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} = R_{e,j}, \tag{3.38}$$

have the same inertia. But this is precisely the condition required for the existence of an H^∞ a posteriori filter! (See Theorem 1). This result is quite useful — it states that an H^∞ (a posteriori) filter exists if, and only if, the pre-array can be triangularized, *i.e.*, if, and only if, the square-root algorithm can be performed and does not break down.

Now that we have settled the existence question, let us return to triangularizing the pre-array (3.25), so that we can write

$$\begin{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & \gamma I_q \end{bmatrix} & \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j^{1/2} & 0 \\ 0 & F_j P_j^{1/2} & G_j \end{bmatrix} \Theta_j = \begin{bmatrix} A & 0 & 0 \\ B & C & 0 \end{bmatrix}, \tag{3.39}$$

where A and C are lower triangular, and where Θ_j is J-unitary with J as in (3.27). The array on the left hand side of (3.39) is referred to as the *pre-array* and the array on the right hand side as the *post-array*. To identify the elements A, B and C in the post array let us square both sides of (3.39) and use the fact that Θ_j is J-unitary. Therefore

$$\begin{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & \gamma I_q \end{bmatrix} & \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j^{1/2} & 0 \\ 0 & F_j P_j^{1/2} & G_j \end{bmatrix} \underbrace{\Theta_j J \Theta_j^*}_{=J} \begin{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & \gamma I_q \end{bmatrix} & 0 \\ P_j^{*/2} \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} & P_j^{*/2} F_j^* \\ 0 & G_j^* \end{bmatrix}$$

$$= \begin{bmatrix} A & 0 & 0 \\ B & C & 0 \end{bmatrix} J \begin{bmatrix} A^* & B^* \\ 0 & C^* \\ 0 & 0 \end{bmatrix}. \tag{3.40}$$

Equating the $(1,1)$ blocks on the left hand and right hand sides of (3.40) yields

$$\begin{bmatrix} I_p & 0 \\ 0 & -\gamma^2 I_q \end{bmatrix} + \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} = A \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} A^*.$$

The left hand side of the above relation is simply $R_{e,j}$. Therefore A is an indefinite square-root of $R_{e,j}$, and we can write

$$A = R_{e,j}^{1/2} \quad , \quad R_{e,j}^{1/2} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} R_{e,j}^{*/2} = R_{e,j}. \tag{3.41}$$

Equating the $(2,1)$ blocks on the left hand and right hand sides of (3.40) yields

$$F_j P_j \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} = B \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} A^*.$$

Therefore

$$B = F_j P_j \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} R_{e,j}^{-*/2} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix},$$

so that we can write

$$B = K_{p,j} R_{e,j}^{1/2}. \tag{3.42}$$

Equating the $(2,2)$ blocks on the left hand and right hand sides of (3.40) yields

$$\begin{aligned} F_j P_j F_j^* + G_j G_j^* &= B \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} B^* + CC^* \\ &= K_{p,j} R_{e,j}^{1/2} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} R_{e,j}^{*/2} K_{p,j}^* + CC^* \\ &= K_{p,j} R_{e,j} K_{p,j}^* + CC^*. \end{aligned}$$

Therefore

$$CC^* = F_j P_j F_j^* + G_j G_j^* - K_{p,j} R_{e,j} K_{p,j}^* = P_{j+1},$$

so that we may write

$$C = P_{j+1}^{1/2} \quad , \quad P_{j+1}^{1/2} P_{j+1}^{*/2} = P_{j+1}. \tag{3.43}$$

We can now summarize our results as follows.

Theorem 3 **(H^∞ A Posteriori Square-Root Algorithm)** *The H^∞ a posteriori filtering problem with level γ has a solution if, and only if, for all $j = 0, \ldots, i$ there exist J-unitary matrices (with J given by (3.27)), Θ_j, such that*

$$\begin{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & \gamma I_q \end{bmatrix} & \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j^{1/2} & 0 \\ 0 & F_j P_j^{1/2} & G_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j}^{1/2} & 0 & 0 \\ K_{p,j} R_{e,j}^{1/2} & P_{j+1}^{1/2} & 0 \end{bmatrix} \tag{3.44}$$

where the algorithm is initialized with, $P_0 = \Pi_0$. If this is the case, then all possible H^∞ a posteriori filters, $\check{s}_{j|j} = \mathcal{F}_{f,j}(y_0, \ldots y_j)$, are given by any choices that yield,

$$\sum_{j=0}^{k} \begin{bmatrix} y_j - H_j \hat{x}_j \\ \check{s}_{j|j} - L_j \hat{x}_j \end{bmatrix}^* R_{e,j}^{-1} \begin{bmatrix} y_j - H_j \hat{x}_j \\ \check{s}_{j|j} - L_j \hat{x}_j \end{bmatrix} \geq 0, \quad 0 \leq k \leq i$$

where $\hat{x}_j$ satisfies the recursion,

$$\hat{x}_{j+1} = F_j \hat{x}_j + K_{p,j} \begin{bmatrix} y_j - H_j \hat{x}_j \\ \check{s}_{j|j} - L_j \hat{x}_j \end{bmatrix}, \quad \hat{x}_0 = 0.$$

In the H^∞ a priori filtering problem we need to begin with the pre-array

$$\begin{bmatrix} \begin{bmatrix} \gamma I_q & 0 \\ 0 & I_p \end{bmatrix} & \begin{bmatrix} L_j \\ H_j \end{bmatrix} P_j^{1/2} & 0 \\ 0 & F_j P_j^{1/2} & G_j \end{bmatrix}, \tag{3.45}$$

and with

$$J = \begin{bmatrix} \begin{bmatrix} -I_q & 0 \\ 0 & I_p \end{bmatrix} & & \\ & I_n & \\ & & I_m \end{bmatrix}. \tag{3.46}$$

[Note the reversal of the order of the $\{H_j, L_j\}$ as compared to the a posteriori case.]

Using similar arguments we may prove the following result.

Theorem 4 **(H^∞ A Priori Square-Root Algorithm)** *The H^∞ a priori filtering problem with level γ has a solution if, and only if, for all $j = 0, \ldots, i$ there exist J-unitary matrices (with J given by (3.46)), Θ_j, such that*

$$\begin{bmatrix} \begin{bmatrix} \gamma I_q & 0 \\ 0 & I_p \end{bmatrix} & \begin{bmatrix} L_j \\ H_j \end{bmatrix} P_j^{1/2} & 0 \\ 0 & F_j P_j^{1/2} & G_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j}^{1/2} & 0 & 0 \\ K_{p,j} R_{e,j}^{1/2} & P_{j+1}^{1/2} & 0 \end{bmatrix} \tag{3.47}$$

where the algorithm is initialized with, $P_0 = \Pi_0$. If this is the case, then all possible H^∞ a priori filters, $\check{s}_j = \mathcal{F}_{f,j}(y_0, \ldots y_{j-1})$, are given by any choices that yield,

$$\sum_{j=0}^{k} \begin{bmatrix} \check{s}_j - L_j\hat{x}_j \\ y_j - H_j\hat{x}_j \end{bmatrix}^* R_{e,j}^{-1} \begin{bmatrix} \check{s}_j - L_j\hat{x}_j \\ y_j - H_j\hat{x}_j \end{bmatrix} \geq 0, \quad 0 \leq k \leq i$$

where $\hat{x}_j$ satisfies the recursion,

$$\hat{x}_{j+1} = F_j\hat{x}_j + K_{p,j} \begin{bmatrix} \check{s}_j - L_j\hat{x}_j \\ y_j - H_j\hat{x}_j \end{bmatrix}, \qquad \hat{x}_0 = 0.$$

Note that, as in the H^2 case, the quantities necessary to update the square-root array and to calculate the desired estimates may all be found from the triangularized post-array.

In conventional Kalman filtering, square-root arrays are preferred since the positive-definiteness of the matrices is guaranteed, and since the Θ_j are unitary, which improves the numerical stability of the algorithm. In the H^∞ setting, the square-root arrays guarantee that the various matrices have their appropriate inertia; however, the Θ_j are no longer unitary but J-unitary. Therefore the numerical aspects need further investigation.

An interesting aspect of Theorems 3 and 4 is that there is no need to explicitly check for the existence conditions required of H^∞ filters (see Theorems 1 and 2). These conditions are built into the square-root algorithms themselves: if the algorithms can be performed then an H^∞ estimator of the desired level exists, and if they cannot be performed such an estimator does not exist.

3.5 The Central Filters

In the previous section we obtained a square-root version of the parametrization of all H^∞ a posteriori and a priori filters. Perhaps the most important filters in these classes are the so-called central filters, which we described in Eqs. (3.11) and (3.17). These filters have the additional properties of being maximum-entropy and risk-sensitive-optimal filters [25, 26], as well as being the solution to certain quadratic dynamic games [27]. In this section we shall develop square-root algorithms for such central filters. As expected, the observer gains for the central filters turn out to be readily obtainable from the square-root array algorithms of Theorems 3 and 4.

Let us begin by recalling the central H^∞ a posteriori filter recursions, *cf.*, Eq. (3.11),

$$\hat{x}_{j+1|j+1} = F_j\hat{x}_{j|j} + K_{s,j+1}(y_{j+1} - H_{j+1}F_j\hat{x}_{j|j}), \tag{3.48}$$

where the desired estimate is given by $\hat{s}_{j|j} = L_j\hat{x}_{j|j}$, and where the gain matrix is given by

$$K_{s,j} = P_j H_j^*(I_p + H_j P_j H_j^*)^{-1}. \tag{3.49}$$

We will now show how to obtain the above gain matrix from the a posteriori square-root recursions.

To this end, let us first note that we can rewrite the a posteriori square-root algorithm of Theorem 3 via the following two-step procedure,

$$\begin{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & \gamma I_q \end{bmatrix} & \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j^{1/2} \\ 0 & P_j^{1/2} \end{bmatrix} \Theta_j^{(1)} = \begin{bmatrix} R_{e,j}^{1/2} & 0 \\ K_{f,j}R_{e,j}^{1/2} & P_{j|j}^{1/2} \end{bmatrix}, \tag{3.50}$$

$$\begin{bmatrix} F_j P_{j|j}^{1/2} & G_j \end{bmatrix} \Theta_j^{(2)} = \begin{bmatrix} P_{j+1}^{1/2} & 0 \end{bmatrix} \tag{3.51}$$

where $\Theta_j^{(1)}$ is J-unitary, with $J = I_p \oplus (-I_q) \oplus I_n$, and $\Theta_j^{(2)}$ is unitary. [Note that the above two-step procedure is the H^∞ analog of Algorithm 2.] In the above recursions, we of course have

$$K_{f,j} = P_j \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} R_{e,j}^{-1}, \tag{3.52}$$

with

$$R_{e,j} = \begin{bmatrix} I_p & 0 \\ 0 & -\gamma^2 I_q \end{bmatrix} + \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j \begin{bmatrix} H_j^* & L_j^* \end{bmatrix}. \tag{3.53}$$

Now in (3.50), $R_{e,j}^{1/2}$ can be any square-root of $R_{e,j}$. Let us study the consequences of choosing a lower triangular square-root. To do so, consider the following triangular factorization of $R_{e,j}$,

$$\begin{aligned} R_{e,j} &= \begin{bmatrix} I_p & 0 \\ L_j P_j H_j^*(I_p + H_j P_j H_j^*)^{-1} & I_q \end{bmatrix} \begin{bmatrix} I_p + H_j P_j H_j^* & 0 \\ 0 & -\Delta_j \end{bmatrix} \times \\ &\quad \begin{bmatrix} I_p & (I_p + H_j P_j H_j^*)^{-1} H_j P_j L_j^* \\ 0 & I_q \end{bmatrix}, \end{aligned}$$

where we have defined the Schur complement,

$$-\Delta_j \triangleq -\gamma^2 I_q + L_j P_j L_j^* - L_j P_j H_j^*(I_p + H_j P_j H_j^*)^{-1} H_j P_j L_j^*.$$

Note that the inertia condition on $R_{e,j}$ implies that $\Delta_j > 0$, so that we may write

$$R_{e,j} = R_{e,j}^{1/2} S R_{e,j}^{*/2}, \tag{3.54}$$

with

$$\begin{aligned} R_{e,j}^{1/2} &= \begin{bmatrix} I_p & 0 \\ L_j P_j H_j^*(I_p + H_j P_j H_j^*)^{-1} & I_q \end{bmatrix} \begin{bmatrix} (I_p + H_j P_j H_j^*)^{1/2} & 0 \\ 0 & \Delta_j^{1/2} \end{bmatrix} \\ &= \begin{bmatrix} (I_p + H_j P_j H_j^*)^{1/2} & 0 \\ L_j P_j H_j^*(I_p + H_j P_j H_j^*)^{-*/2} & \Delta_j^{1/2} \end{bmatrix}, \end{aligned} \tag{3.55}$$

and $S = \begin{bmatrix} I_p & \\ & -I_q \end{bmatrix}$. Now the $(2,1)$ block entry in the post-array of (3.50) is given by

$$\begin{aligned} K_{f,j}R_{e,j}^{1/2} &= P_j \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} R_{e,j}^{-*/2} \\ &= P_j \begin{bmatrix} H_j^* & L_j^* \end{bmatrix} \begin{bmatrix} (I_p + H_jP_jH_j^*)^{-*/2} & \times \\ 0 & \times \end{bmatrix} \\ &= P_j \begin{bmatrix} H_j^*(I_p + H_jP_jH_j^*)^{-*/2} & \times \end{bmatrix}, \end{aligned} \tag{3.56}$$

where $\times$ denotes irrelevant entries.

Eqs. (3.55) and (3.56) now suggest how to compute the desired gain matrix $K_{s,i}$. Indeed,

$$\begin{gathered} \left(\text{first block column of } K_{f,j}R_{e,j}^{1/2}\right) \cdot \left((1,1) \text{ block entry of } R_{e,j}^{1/2}\right)^{-1} = \\ P_jH_j^*(I_p + H_jP_jH_j^*)^{-*/2} \cdot (I_p + H_jP_jH_j^*)^{-1/2} = \\ K_{s,j}. \end{gathered} \tag{3.57}$$

We are thus led to the following result.

Algorithm 3 (Central H^∞ A Posteriori Square-Root Algorithm) *The H^∞ a posteriori filtering problem with level γ has a solution if, and only if, for all $j = 0, \ldots, i$ there exist J-unitary (with $J = I_p \oplus (-I_q) \oplus I_n$) matrices, $\Theta_j^{(1)}$, such that*

$$\begin{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & \gamma I_q \end{bmatrix} & \begin{bmatrix} H_j \\ L_j \end{bmatrix} P_j^{1/2} \\ 0 & P_j^{1/2} \end{bmatrix} \Theta_j^{(1)} = \begin{bmatrix} R_{e,j}^{1/2} & 0 \\ K_{f,j}R_{e,j}^{1/2} & P_{j|j}^{1/2} \end{bmatrix}, \tag{3.58}$$

$$\begin{bmatrix} F_jP_{j|j}^{1/2} & G_j \end{bmatrix} \Theta_j^{(2)} = \begin{bmatrix} P_{j+1}^{1/2} & 0 \end{bmatrix} \tag{3.59}$$

with $R_{e,j}^{1/2}$ lower block triangular, and with $\Theta_j^{(2)}$ unitary. The gain matrix $K_{s,j}$ needed to update the estimates in the central filter recursions

$$\hat{x}_{j|j} = F_{j-1}\hat{x}_{j-1|j-1} + K_{s,j}(y_j - H_jF_{j-1}\hat{x}_{j-1|j-1}), \qquad \hat{x}_{-1|-1} = 0,$$

is equal to

$$K_{s,j} = \bar{K}_{s,j}(I + H_jP_jH_j^*)^{-1/2},$$

where $\bar{K}_{s,j}$ is given by the first block column of $\bar{K}_{f,j} = K_{f,j}R_{e,j}^{1/2}$, and $(I + H_jP_jH_j^*)^{1/2}$ is given by the $(1,1)$ block entry of $R_{e,j}^{1/2}$. The algorithm is initialized with $P_0 = \Pi_0$.

We can now proceed with a similar argument to find square-root recursions for the central H^∞ a priori filters. Let us first recall from Eq. (3.17) that the central H^∞ a priori filter recursions are

$$\hat{x}_{j+1} = F_j\hat{x}_j + K_{a,j}(y_j - H_j\hat{x}_j), \tag{3.60}$$

where the desired estimate is given by $\hat{s}_j = L_j\hat{x}_j$, and where the gain matrix is given by

$$K_{a,j} = F_j\tilde{P}_jH_j^*(I_p + H_j\tilde{P}_jH_j^*)^{-1}, \tag{3.61}$$

with

$$\tilde{P}_j = P_j - P_jL_j^*(-\gamma^2 I_q + L_jP_jL_j^*)^{-1}L_jP_j. \tag{3.62}$$

We now show how to obtain the above gain matrix from the a priori square-root recursions,

$$\begin{bmatrix} \begin{bmatrix} \gamma I_q & 0 \\ 0 & I_p \end{bmatrix} & \begin{bmatrix} L_j \\ H_j \end{bmatrix} P_j^{1/2} & 0 \\ 0 & F_jP_j^{1/2} & G_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j}^{1/2} & 0 & 0 \\ K_{p,j}R_{e,j}^{1/2} & P_{j+1}^{1/2} & 0 \end{bmatrix} \tag{3.63}$$

Note now that,

$$K_{p,j} = F_jP_j \begin{bmatrix} L_j^* & H_j^* \end{bmatrix} R_{e,j}^{-1}, \tag{3.64}$$

with

$$R_{e,j} = \begin{bmatrix} -\gamma^2 I_q & 0 \\ 0 & I_p \end{bmatrix} + \begin{bmatrix} L_j \\ H_j \end{bmatrix} P_j \begin{bmatrix} L_j^* & H_j^* \end{bmatrix}. \tag{3.65}$$

As mentioned earlier, $R_{e,j}^{1/2}$ in (3.63) can be any indefinite square-root of R_e. Let us, once more, study the consequences of choosing a lower triangular square-root. To do so, consider the following block lower-diagonal-upper triangular factorization of the matrix, $R_{e,j}$,

$$\begin{bmatrix} I_q & 0 \\ -\gamma^{-2}H_j\tilde{P}_jL_j^* & I_p \end{bmatrix} \begin{bmatrix} -\gamma^2 I_q + L_jP_jL_j^* & 0 \\ 0 & I_p + H_j\tilde{P}_jH_j^* \end{bmatrix} \begin{bmatrix} I_q & -\gamma^{-2}L_j\tilde{P}_jH_j^* \\ 0 & I_p \end{bmatrix}, \tag{3.66}$$

where we have used the facts that,

$$H_jP_jL_j^*(-\gamma^2 I_q + L_jP_jL_j^*)^{-1} = -\gamma^{-2}H_j\tilde{P}_jL_j^*,$$

and, for the Schur complement,

$$I_p + H_jP_jH_j^* - H_jP_jL_j^*(-\gamma^2 I_q + L_jP_jL_j^*)^{-1}L_jP_jH_j^* = I_p + H_j\tilde{P}_jH_j^*.$$

Now the inertia conditions on $R_{e,j}$ require that $-\gamma^{-2}I_q + L_jP_jL_j^* < 0$ and $I_p + H_j\tilde{P}_jH_j^* > 0$, so that we may write,

$$R_{e,j} = R_{e,j}^{1/2}SR_{e,j}^{*/2}, \tag{3.67}$$

with

$$R_{e,j}^{1/2} = \begin{bmatrix} (\gamma^2 I_q - L_jP_jL_j^*)^{1/2} & 0 \\ -\gamma^{-2}H_j\tilde{P}_jL_j^*(\gamma^2 I_q - L_jP_jL_j^*)^{1/2} & (I_p + H_j\tilde{P}_jH_j^*)^{1/2} \end{bmatrix}, \tag{3.68}$$

and $S = \begin{bmatrix} -I_q & \\ & I_p \end{bmatrix}$. Now the $(2,1)$ block entry in the post-array of (3.63) is given by

$$\begin{aligned}
K_{p,j}R_{e,j}^{1/2} &= F_jP_j \begin{bmatrix} L_j^* & H_j^* \end{bmatrix} R_{e,j}^{-*/2} \\
&= F_jP_j \begin{bmatrix} L_j^* & H_j^* \end{bmatrix} \begin{bmatrix} \times & \gamma^{-2}L_j\tilde{P}_jH_j^*(I_p + H_j\tilde{P}_jH_j^*)^{-*/2} \\ 0 & (I_p + H_j\tilde{P}_jH_j^*)^{-*/2} \end{bmatrix} \\
&= F_jP_j \begin{bmatrix} \times & (\gamma^{-2}L_j^*L_j\tilde{P}_j + I_n)H_j^*(I_p + H_j\tilde{P}_jH_j^*)^{-*/2} \end{bmatrix} \\
&= F_j \begin{bmatrix} \times & \tilde{P}_jH_j^*(I_p + H_j\tilde{P}_jH_j^*)^{-*/2} \end{bmatrix}, \qquad (3.69)
\end{aligned}$$

where in the last step we have used the (readily verified) identity,

$$P_j(\gamma^{-2}L_j^*L_j\tilde{P}_j + I_n) = \tilde{P}_j,$$

and where $\times$ denotes irrelevant entries.

Eqs. (3.68) and (3.69) now suggest how to compute the desired gain matrix $K_{a,i}$. Indeed,

$$\begin{gathered}
\left(\text{second block column of } K_{p,j}R_{e,j}^{1/2}\right) \cdot \left((2,2) \text{ block entry of } R_{e,j}^{1/2}\right)^{-1} = \\
F_j\tilde{P}_jH_j^*(I_p + H_j\tilde{P}_jH_j^*)^{-*/2} \cdot (I_p + H_j\tilde{P}_jH_j^*)^{-1/2} = \\
K_{a,j}.
\end{gathered} \qquad (3.70)$$

We are thus led to the following result.

Algorithm 4 (Central H^∞ A Priori Square-Root Alg.) *The H^∞ a priori filtering problem with level γ has a solution if, and only if, for all $j = 0, \ldots, i$ there exist J-unitary matrices (with $J = (-I_q) \oplus I_p \oplus I_n \oplus I_m$), Θ_j, such that*

$$\begin{bmatrix} \begin{bmatrix} \gamma I_q & 0 \\ 0 & I_p \end{bmatrix} & \begin{bmatrix} L_j \\ H_j \end{bmatrix} P_j^{1/2} & 0 \\ 0 & F_jP_j^{1/2} & G_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j}^{1/2} & 0 & 0 \\ K_{p,j}R_{e,j}^{1/2} & P_{j+1}^{1/2} & 0 \end{bmatrix} \qquad (3.71)$$

with $R_{e,j}^{1/2}$ lower block triangular. The gain matrix $K_{a,i}$ needed to update the estimates in

$$\hat{x}_{j+1} = F_j\hat{x}_j + K_{a,j}(y_j - F_j\hat{x}_j), \qquad \hat{x}_0 = 0,$$

is equal to

$$K_{a,j} = \bar{K}_{a,j}(I + H_j\tilde{P}_jH_j^*)^{-1/2},$$

where $\bar{K}_{a,j}$ is given by the second block column of $\bar{K}_{p,j} = K_{p,j}R_{e,j}^{1/2}$, and $(I + H_j\tilde{P}_jH_j^*)^{1/2}$ is given by the $(2,2)$ block entry of $R_{e,j}^{1/2}$. The algorithm is initialized with $P_0 = \Pi_0$.

4 H^2 Fast Array Algorithms

The conventional Kalman filter and square-root array recursions of Sec. 2 both require $O(n^3)$ operations per iteration (where n is the number of states in the state-space model). However, when the state-space model is time-invariant (or if the time-variation is structured in a certain way), there exist fast recursions that require only $O(n^2)$ operations per iteration [8, 9, 10, 28].

In what follows we shall assume a time-invariant state-space model of the form

$$\left\{ \begin{array}{rcl} x_{j+1} & = & Fx_j + Gu_j, \quad x_0 \\ y_j & = & Hx_j + v_j \end{array} \right. . \tag{4.1}$$

In the H^2 case, where the $\{u_j, v_j\}$ are zero mean independent random variables, we shall also assume that the covariances of the $\{u_j, v_j\}$ are constant, *i.e.* $Q_j = Q \geq 0$ and $R_j = R > 0$, for all j. As before, we are interested in obtaining estimates of some linear combinations of the states, $s_j = L_j x_j$, and, in particular, the filtered estimates, $\hat{s}_{j|j} = L_j \hat{x}_{j|j}$, and predicted estimates, $\hat{s}_j = L_j \hat{x}_j$, that use the observations $\{y_k\}_{k=0}^{j}$ and $\{y_k\}_{k=0}^{j-1}$, respectively.

Under the aforementioned assumptions, it turns out that we can write

$$P_{j+1} - P_j = M_j S M_j, \qquad \text{for all } j, \tag{4.2}$$

where M_j is a $n \times d$ matrix and S is a $d \times d$ signature matrix (*i.e.* a diagonal matrix with $+1$ and -1 on the diagonal). Thus, for time-invariant state-space models, $P_{j+1} - P_j$ has rank d for all j and in addition has constant inertia. In several important cases, d can be much less than n.

One such case is when $P_0 = 0$, where, using the Riccati recursion (2.6), we have

$$P_1 = GQG^*, \tag{4.3}$$

so that $M_0 = GQ^{1/2}$ and $S = I_m$. Thus, here, we have $d = m$, with m typically less than n.

Another case is when $P_0 = \Pi$, the solution to the (steady-state) Lyapunov equation,

$$\Pi = F\Pi F^* + GQG^*, \tag{4.4}$$

in which case from (2.6) it follows that

$$P_1 - P_0 = -F\Pi(R + H\Pi H^*)^{-1}\Pi F^*, \tag{4.5}$$

so that $M_0 = F\Pi(R + H\Pi H^*)^{-*/2}$ and $S = -I_p$. Thus, here, we have $d = p$, with p typically less than n.

In anycase, when $d < n$, propagating the smaller matrices M_j, which is equivalent to propagating the P_j, can offer computational reductions. This what is done by the following (so-called Chandrasekhar) recursions (see [29], App. II).

In the conventional fast recursions, one begins with the pre-array

$$\begin{bmatrix} R_{e,j}^{1/2} & HM_j \\ \bar{K}_{p,j} & FM_j \end{bmatrix}, \tag{4.6}$$

where $R_{e,j}^{1/2}R_{e,j}^{*/2} = R_{e,j} = R + HP_jH^*$ and $\bar{K}_{p,j} = K_{p,j}R_{e,j}^{-*/2}$, and triangularizes the array using a J-unitary matrix Θ_j where J is given by $\begin{bmatrix} I_p & \\ & S \end{bmatrix}$. The result of this triangularization gives us the various quantities of interest for propagating the Kalman filter recursions.

Algorithm 5 (Fast H^2 Recursions) *The gain matrix $K_{p,j} = \bar{K}_{p,j}R_{e,j}^{-1/2}$ necessary to obtain the state estimates in the conventional Kalman filter*

$$\hat{x}_{j+1} = F\hat{x}_j + K_{p,j}(y_j - H\hat{x}_j), \qquad \hat{x}_0 = 0,$$

can be computed using

$$\begin{bmatrix} R_{e,j}^{1/2} & HM_j \\ \bar{K}_{p,j} & FM_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j+1}^{1/2} & 0 \\ \bar{K}_{p,j+1} & M_{j+1} \end{bmatrix}, \tag{4.7}$$

where Θ_j is any J-unitary matrix (with $J = I_p \oplus S$) that triangularizes the above pre-array. The algorithm is initialized with

$$R_{e,0} = R + H\Pi_0H^*, \qquad \bar{K}_{p,0} = F\Pi_0H^*R_{e,0}^{1/2},$$

and

$$P_1 - \Pi_0 = F\Pi_0F^* + GQG^* - K_{p,0}R_{e,0}K_{p,0}^* - \Pi_0 = M_0SM_0^*.$$

Thus, once more, the quantities necessary to update the arrays and to calculate the state estimates are all found from the triangularized post array.

The validity of the above algorithm can be readily verified by squaring both sides of the equation,

$$\begin{bmatrix} R_{e,j}^{1/2} & HM_j \\ \bar{K}_{p,j} & FM_j \end{bmatrix} \Theta_j = \begin{bmatrix} A & 0 \\ B & C \end{bmatrix}, \tag{4.8}$$

and using the J-unitarity of Θ_j, to find the entries of the post array. This leads to,

$$\underbrace{\underbrace{R_{e,j}^{1/2}R_{e,j}^{*/2}}_{R_{e,j}} + H\underbrace{M_jSM_j^*}_{P_{i+1}-P_i}H^*}_{R_{e,j+1}} = AA^*, \tag{4.9}$$

from which we conclude that $A = R_{e,j+1}^{1/2}$, and

$$\underbrace{\underbrace{\bar{K}_{p,j} R_{e,j}^{*/2}}_{FP_j H^*} + F \underbrace{M_j S M_j^*}_{P_{i+1}-P_i} H^*}_{FP_{i+1}H^*} = BA^*, \tag{4.10}$$

from which we conclude that $B = FP_{i+1}H^* R_{e,j+1}^{-*/2} = \bar{K}_{p,j+1}$. Finally, we have

$$\begin{aligned} CSC^* &= FM_j SM_j^* F^* + \bar{K}_{p,j}\bar{K}_{p,j}^* - BB^* \\ &= F(P_{i+1} - P_i)F^* + \bar{K}_{p,j}\bar{K}_{p,j}^* - \bar{K}_{p,j+1}\bar{K}_{p,j+1}^* \\ &= FP_{i+1}F^* + GQG^* - \bar{K}_{p,j+1}\bar{K}_{p,j+1}^* - \left[FP_iF^* + GQG^* - \bar{K}_{p,j}\bar{K}_{p,j}^*\right] \\ &= P_{i+2} - P_{i+1}, \end{aligned}$$

from which we infer that, $C = M_{j+1}$.

If, instead of defining $M_j S M_j^* = P_{j+1} - P_j$, we had defined

$$N_j S_f N_j^* = P_{j|j} - P_{j-1|j-1}, \tag{4.11}$$

where $P_{j|j} = E\tilde{x}_{j|j}\tilde{x}_{j|j}^*$ is the filtered state error variance, which satisfies the recursion,

$$P_{j|j} = FP_{j-1|j-1}F^* + GQG^* - K_{f,j}R_{e,j}K_{f,j}^*, \tag{4.12}$$

with $K_{f,j} = P_j H R_{e,j}^{-1}$, then it is also possible to obtain the following (so-called) filtered form of the fast recursions.[7]

Algorithm 6 (Fast H^2 Recursions- Filtered Form) *If F is invertible, the gain matrix $K_{f,j} = \bar{K}_{f,j} R_{e,j}^{-1/2}$ necessary to obtain the state estimates in the filtered form of the conventional Kalman filter*

$$\hat{x}_{j|j} = F\hat{x}_{j-1|j-1} + K_{f,j}(y_j - HF\hat{x}_{j-1|j-1}), \qquad \hat{x}_{-1|-1} = 0,$$

can be computed using

$$\begin{bmatrix} R_{e,j}^{1/2} & HFN_j \\ \bar{K}_{f,j} & FN_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j+1}^{1/2} & 0 \\ \bar{K}_{f,j+1} & N_{j+1} \end{bmatrix}, \tag{4.13}$$

where Θ_j is any J-unitary matrix (with $J = I_p \oplus S_f$) that triangularizes the above pre-array. The algorithm is initialized with

$$R_{e,0} = R + H\Pi_0 H^*, \qquad \bar{K}_{f,0} = \Pi_0 H^* R_{e,0}^{1/2},$$

[7] Note that in this case *if F is nonsingular* it can also be shown that $P_{j+1|j+1} - P_{j|j}$ has constant inertia, given by the inertia of the signature matrix, S_f, for all j. This follows from the fact that $P_{j+1} = FP_{j|j}F^* + GQG^*$, so that $P_{j+1} - P_j = F(P_{j|j} - P_{j-1|j-1})F^*$.

and

$$F^{-1}(P_1-\Pi_0)F^{-*} = \Pi_0+F^{-1}GQG^*F^{-*}-K_{f,0}R_{e,0}K_{f,0}^*-F^{-1}\Pi_0F^{-*} = N_0S_fN_0^*.$$

Note that compared to the square-root formulas, the size of the pre-array in the fast recursions has been reduced from $(p+n)\times(p+n+m)$ to $(p+n)\times(p+d)$ where m and p are the dimensions of the driving disturbance and output, respectively, and where n is the number of the states. Thus the number of operations for each iteration has been reduced from $O(n^3)$ to $O(n^2d)$, with d typically much less than n.

5 H^∞ Fast Array Algorithms

In this section we shall derive the H^∞ counterparts of the fast H^2 Chandrasekhar recursions of the previous section. We shall essentially see that, when the underlying state-space model is time-invariant, all the arguments necessary for the development of these algorithms go through, provided that we consider the geometry of indefinite spaces. We first give the general recursions, and then specialize them to obtain the central filters.

5.1 The General Case

Consider, once more, the time-invariant state-space model (4.1), and the corresponding H^∞ Riccati recursion (3.7). Suppose that the matrix Π_0 can be chosen such that $P_1-\Pi_0$ has low rank. In other words,

$$P_1-\Pi_0 = F\Pi_0F^*+GG^*-K_{p,0}R_{e,0}K_{p,0}^*-\Pi_0 = M_0SM_0^*, \tag{5.1}$$

where M_0 is a $n\times d$ matrix (typically $d\ll n$) and S is a $d\times d$ signature matrix, and, of course,

$$K_{p,j} = FP_j\begin{bmatrix} H^* & L^*\end{bmatrix}R_{e,j}^{-1},\ R_{e,j} = \begin{bmatrix} I_p & 0 \\ 0 & -\gamma^2 I_q\end{bmatrix}+\begin{bmatrix} H \\ L\end{bmatrix}P_j\begin{bmatrix} H^* & L^*\end{bmatrix}. \tag{5.2}$$

We shall presently show by induction that under the assumptions of a time-invariant state-space model, if the a posteriori H^∞ filtering problem has a solution for all j, then $P_{j+1}-P_j$ has rank d and constant inertia for all j and that we can actually write $P_{j+1}-P_j = M_jSM_j^*$.

Consider the following pre-array

$$\begin{bmatrix} R_{e,j}^{1/2} & \begin{bmatrix} H \\ L\end{bmatrix}M_j \\ \bar{K}_{p,j} & FM_j\end{bmatrix}, \tag{5.3}$$

which is the obvious extension of the pre-array in (4.7) to the indefinite-metric setting of the H^∞ a posteriori filtering problem. Now the H^∞ a posteriori filtering problem will have a solution if, and only if, all leading submatrices of R and $R_{e,j}$ (or $R_{e,j+1}$, for that matter) have the same inertia. In view of Lemma 1, this implies that the H^∞ a posteriori filtering problem with level γ will have a solution if, and only if, there exists a J-unitary matrix Θ_j that triangularizes (5.3) where

$$J = \begin{bmatrix} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} & \\ & S \end{bmatrix}. \tag{5.4}$$

Therefore we can write

$$\begin{bmatrix} R_{e,j}^{1/2} & \begin{bmatrix} H \\ L \end{bmatrix} M_j \\ \bar{K}_{p,j} & FM_j \end{bmatrix} \Theta_j = \begin{bmatrix} A & 0 \\ B & C \end{bmatrix}. \tag{5.5}$$

To identify the elements A, B and C in the post-array we square both sides of (5.5) and use the fact that Θ_j is J-unitary. Therefore

$$\begin{aligned} \begin{bmatrix} R_{e,j}^{1/2} & \begin{bmatrix} H \\ L \end{bmatrix} M_j \\ \bar{K}_{p,j} & FM_j \end{bmatrix} \underbrace{\Theta_j J \Theta_j^*}_{=J} \begin{bmatrix} R_{e,j}^{*/2} & \bar{K}_{p,j}^* \\ M_j^* \begin{bmatrix} H^* & L^* \end{bmatrix} & M_j F^* \end{bmatrix} \\ = \begin{bmatrix} A & 0 \\ B & C \end{bmatrix} J \begin{bmatrix} A^* & B^* \\ 0 & C^* \end{bmatrix}. \end{aligned} \tag{5.6}$$

Equating the $(1,1)$ blocks in (5.6) yields,

$$\begin{aligned} A \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} A^* &= R_{e,j}^{1/2} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} R_{e,j}^{*/2} + \begin{bmatrix} H \\ L \end{bmatrix} M_j S M_j^* \begin{bmatrix} H^* & L^* \end{bmatrix} \\ &= R_{e,j} + \begin{bmatrix} H \\ L \end{bmatrix} (P_{j+1} - P_j) \begin{bmatrix} H^* & L^* \end{bmatrix} \\ &= R_j + \begin{bmatrix} H \\ L \end{bmatrix} P_{j+1} \begin{bmatrix} H^* & L^* \end{bmatrix} \\ &= R_{e,j+1}. \end{aligned}$$

Therefore A is an indefinite square-root of $R_{e,j+1}$,

$$A = R_{e,j+1}^{1/2}. \tag{5.7}$$

Equating the $(2,1)$ blocks in (5.6) yields,

$$\begin{aligned} B \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} A^* &= \bar{K}_{p,j} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} R_{e,j}^{*/2} + FM_j S M_j^* \begin{bmatrix} H^* & L^* \end{bmatrix} \\ &= K_{p,j} R_{e,j} + F(P_{j+1} - P_j) \begin{bmatrix} H^* & L^* \end{bmatrix} \\ &= FP_j \begin{bmatrix} H^* & L^* \end{bmatrix} + F(P_{j+1} - P_j) \begin{bmatrix} H^* & L^* \end{bmatrix} \\ &= FP_{j+1} \begin{bmatrix} H^* & L^* \end{bmatrix}. \end{aligned}$$

Therefore

$$\begin{aligned} B &= FP_{j+1} \begin{bmatrix} H^* & L^* \end{bmatrix} A^{-*} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} \\ &= FP_{j+1} \begin{bmatrix} H^* & L^* \end{bmatrix} R_{e,j+1}^{-*/2} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} = \bar{K}_{p,j+1}. \end{aligned} \tag{5.8}$$

Equating the $(2,2)$ blocks in (5.6) yields

$$CSC^* + B \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} B^* = \bar{K}_{p,j} \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} \bar{K}_{p,j}^* + FM_jSM_j^*F^*.$$

Therefore

$$CSC^* + K_{p,j+1}R_{e,j+1}K_{p,j+1}^* = K_{p,j}R_{e,j}K_{p,j}^* + F(P_{j+1} - P_j)F^*.$$

We can now write

$$\begin{aligned} CSC^* &= FP_{j+1}F^* - K_{p,j+1}R_{e,j+1}K_{p,j+1}^* + GG^* \\ &\quad -(FP_jF^* - K_{p,j}R_{e,j}K_{p,j}^* + GG^*) \\ &= P_{j+2} - P_{j+1}, \end{aligned}$$

and finally

$$C = M_{j+1}. \tag{5.9}$$

Note that our derivation of C also shows that if $P_1 - P_0 = M_0SM_0^*$ then $P_{j+1} - P_j = M_jSM_j^*$ for all j.

We have thus established

$$\begin{bmatrix} R_{e,j}^{1/2} & \begin{bmatrix} H \\ L \end{bmatrix} M_j \\ \bar{K}_{p,j} & FM_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j+1}^{1/2} & 0 \\ \bar{K}_{p,j+1} & M_{j+1} \end{bmatrix},$$

from which we can now give the following fast Chandrasekhar version of the parametrization of all H^∞ a posteriori filters.

Theorem 5 (Fast H^∞ A Posteriori Recursions) *The H^∞ a posteriori filtering problem with level γ has a solution if, and only if, all leading submatrices of*

$$R = \begin{bmatrix} I_p & 0 \\ 0 & -\gamma^2 I_q \end{bmatrix} \quad \textit{and} \quad R_{e,0} = \begin{bmatrix} I_p & 0 \\ 0 & -\gamma^2 I_q \end{bmatrix} + \begin{bmatrix} H \\ L \end{bmatrix} \Pi_0 \begin{bmatrix} H^* & L^* \end{bmatrix}$$

have the same inertia, and if for all $j = 0, \ldots, i$ there exist J-unitary matrices (with $J = I_p \oplus (-I_q) \oplus S$), Θ_j, such that

$$\begin{bmatrix} R_{e,j}^{1/2} & \begin{bmatrix} H \\ L \end{bmatrix} M_j \\ K_{p,j}R_{e,j}^{1/2} & FM_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j+1}^{1/2} & 0 \\ K_{p,j+1}R_{e,j+1}^{1/2} & M_{j+1} \end{bmatrix} \tag{5.10}$$

where the algorithm is initialized with, $R_{e,0}$, $K_{p,0} = F\Pi_0 \begin{bmatrix} H^* & L^* \end{bmatrix} R_{e,0}^{-1}$, *and*

$$P_1 - \Pi_0 = F\Pi_0 F^* + GQG^* - K_{p,0}R_{e,0}K_{p,0}^* - \Pi_0 = M_0 S M_0^*. \qquad (5.11)$$

If this is the case, then all possible H^∞ a posteriori filters, $\check{s}_{j|j} = \mathcal{F}_{f,j}(y_0, \ldots y_j)$, *are given by any choices that yield,*

$$\sum_{j=0}^{k} \begin{bmatrix} y_j - H_j\hat{x}_j \\ \check{s}_{j|j} - L_j\hat{x}_j \end{bmatrix}^* R_{e,j}^{-1} \begin{bmatrix} y_j - H_j\hat{x}_j \\ \check{s}_{j|j} - L_j\hat{x}_j \end{bmatrix} \geq 0, \quad 0 \leq k \leq i$$

where $\hat{x}_j$ *satisfies the recursion,*

$$\hat{x}_{j+1} = F_j\hat{x}_j + K_{p,j} \begin{bmatrix} y_j - H_j\hat{x}_j \\ \check{s}_{j|j} - L_j\hat{x}_j \end{bmatrix}, \qquad \hat{x}_0 = 0.$$

In the H^∞ a priori filtering problem, we need instead to start with the pre-array,

$$\begin{bmatrix} R_{e,j}^{1/2} & \begin{bmatrix} L \\ H \end{bmatrix} M_j \\ K_{p,j}R_{e,j}^{1/2} & FM_j \end{bmatrix}, \qquad (5.12)$$

where now

$$K_{p,j} = FP_j \begin{bmatrix} L^* & H^* \end{bmatrix} R_{e,j}^{-1}, \; R_{e,j} = \begin{bmatrix} -\gamma^2 I_q & 0 \\ 0 & I_p \end{bmatrix} + \begin{bmatrix} L \\ H \end{bmatrix} P_j \begin{bmatrix} L^* & H^* \end{bmatrix}. \qquad (5.13)$$

Note that the only difference with the a posteriori case is in the order of the matrices $\{H, L\}$.

Proceeding with an argument similar to what was done in the a posteriori case, we can show the following result.

Theorem 6 (Fast H^∞ A Priori Recursions) *The H^∞ a priori filtering problem with level γ has a solution if, and only if, all leading submatrices of*

$$R = \begin{bmatrix} -\gamma^2 I_q & 0 \\ 0 & I_p \end{bmatrix} \quad \text{and} \quad R_{e,0} = \begin{bmatrix} -\gamma^2 I_q & 0 \\ 0 & I_p \end{bmatrix} + \begin{bmatrix} L \\ H \end{bmatrix} \Pi_0 \begin{bmatrix} L^* & H^* \end{bmatrix}$$

have the same inertia, and if for all $j = 0, \ldots, i$ there exist J-unitary matrices (with $J = (-I_q) \oplus I_p \oplus S$), Θ_j, such that

$$\begin{bmatrix} R_{e,j}^{1/2} & \begin{bmatrix} L \\ H \end{bmatrix} M_j \\ K_{p,j}R_{e,j}^{1/2} & FM_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j+1}^{1/2} & 0 \\ K_{p,j+1}R_{e,j+1}^{1/2} & M_{j+1} \end{bmatrix} \qquad (5.14)$$

where the algorithm is initialized with, $R_{e,0}$, $K_{p,0} = F\Pi_0 \begin{bmatrix} L^* & H^* \end{bmatrix} R_{e,0}^{-1}$, *and*

$$P_1 - \Pi_0 = F\Pi_0 F^* + GQG^* - K_{p,0} R_{e,0} K_{p,0}^* - \Pi_0 = M_0 S M_0^*. \qquad (5.15)$$

If this is the case, then all possible H^∞ *a priori filters,* $\check{s}_j = \mathcal{F}_{f,j}(y_0, \ldots y_{j-1})$, *are given by any choices that yield,*

$$\sum_{j=0}^{k} \begin{bmatrix} \check{s}_j - L_j \hat{x}_j \\ y_j - H_j \hat{x}_j \end{bmatrix}^* R_{e,j}^{-1} \begin{bmatrix} \check{s}_j - L_j \hat{x}_j \\ y_j - H_j \hat{x}_j \end{bmatrix} \geq 0, \quad 0 \leq k \leq i$$

where $\hat{x}_j$ *satisfies the recursion,*

$$\hat{x}_{j+1} = F_j \hat{x}_j + K_{p,j} \begin{bmatrix} \check{s}_j - L_j \hat{x}_j \\ y_j - H_j \hat{x}_j \end{bmatrix}, \qquad \hat{x}_0 = 0.$$

Note that compared to the H^∞ square-root formulas, the size of the pre-array in the H^∞ fast recursions has been reduced from $(p+q+n) \times (p+q+n+m)$ to $(p+q+n) \times (p+q+d)$ where m, p and q are the dimensions of the driving disturbance, output and states to be estimated, respectively, and where n is the number of the states. Thus the number of operations for each iteration has been reduced from $O(n^3)$ to $O(n^2 d)$ with d typically much less than n.

As in the square-root case, the fast recursions do not require explicitly checking the positivity conditions of Theorems 1 and 2 — if the recursions can be carried out then an H^∞ estimator of the desired level exists, and if not, such an estimator does not exist.

5.2 The Central Filters

The preceding section gave fast versions of all possible H^∞ a posteriori and a priori filters. Here we shall specialize these recursions to the central a posteriori and a priori filters. We shall show that the observer gains for these filters can be obtained from the post-arrays of Theorems 5 and 6, provided we insist on (block) lower triangular square-roots for $R_{e,j}^{1/2}$. The development closely follows that of Sec. 3.5 and uses the important facts (established in Sec. 3.5) that if $R_{e,j}^{1/2}$ is lower triangular, then $K_{s,j}$, the gain matrix for the central a posteriori filter, is given by,

$$K_{s,j} = \left(\text{first block column of } K_{f,j} R_{e,j}^{1/2}\right) \cdot \left((1,1) \text{ block entry of } R_{e,j}^{1/2}\right)^{-1}, \qquad (5.16)$$

and $K_{a,j}$, the gain matrix for the central a priori filter, is given by,

$$K_{a,j} = \left(\text{second block column of } K_{p,j} R_{e,j}^{1/2}\right) \cdot \left((2,2) \text{ block entry of } R_{e,j}^{1/2}\right)^{-1}. \qquad (5.17)$$

The following results are now readily establishable. (The proofs are straightforward and will be omitted for brevity.)

Algorithm 7 (Fast Central H^∞ A Posteriori Recursions) *If F is invertible, the H^∞ a posteriori filtering problem with level γ has a solution if, and only if, all leading submatrices of*

$$R = \begin{bmatrix} I_p & 0 \\ 0 & -\gamma^2 I_q \end{bmatrix} \quad \text{and} \quad R_{e,0} = \begin{bmatrix} I_p & 0 \\ 0 & -\gamma^2 I_q \end{bmatrix} + \begin{bmatrix} H \\ L \end{bmatrix} \Pi_0 \begin{bmatrix} H^* & L^* \end{bmatrix}$$

have the same inertia, and if for all $j = 0, \ldots, i$ there exist J-unitary matrices, Θ_j, (where $J = I_p \oplus (-I_q) \oplus S_f$) such that

$$\begin{bmatrix} R_{e,j}^{1/2} & \begin{bmatrix} H \\ L \end{bmatrix} F N_j \\ K_{f,j} R_{e,j}^{1/2} & F N_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j+1}^{1/2} & 0 \\ K_{f,j} R_{e,j+1}^{1/2} & N_{j+1} \end{bmatrix} \tag{5.18}$$

with $R_{e,j}^{1/2}$ and $R_{e,j+1}^{1/2}$ block lower triangular. The algorithm is initialized with, $R_{e,0}$, $K_{f,0} = \Pi_0 \begin{bmatrix} H^* & L^* \end{bmatrix} R_{e,0}^{-1}$, and

$$F^{-1}(P_1 - \Pi_0)F^{-*} = \Pi_0 + F^{-1}GQG^*F^{-*} - K_{f,0}R_{e,0}K_{f,0}^* - F^{-1}\Pi_0 F^{-*} = N_0 S_f N_0^*.$$

The gain matrix $K_{s,j}$ needed to update the estimates in the central filter recursions

$$\hat{x}_{j|j} = F_{j-1}\hat{x}_{j-1|j-1} + K_{s,j}(y_j - H_j F_{j-1}\hat{x}_{j-1|j-1}), \qquad \hat{x}_{-1|-1} = 0,$$

is equal to

$$K_{s,j} = \bar{K}_{s,j}(I + H_j P_j H_j^*)^{-1/2},$$

where $\bar{K}_{s,j}$ is given by the first block column of $\bar{K}_{f,j} = K_{f,j} R_{e,j}^{1/2}$, and $(I + H_j P_j H_j^*)^{1/2}$ is given by the $(1,1)$ block entry of $R_{e,j}^{1/2}$.

Algorithm 8 (Fast Central H^∞ A Priori Recursions) *The H^∞ a priori filtering problem with level γ has a solution if, and only if, all leading submatrices of*

$$R = \begin{bmatrix} -\gamma^2 I_q & 0 \\ 0 & I_p \end{bmatrix} \quad \text{and} \quad R_{e,0} = \begin{bmatrix} -\gamma^2 I_q & 0 \\ 0 & I_p \end{bmatrix} + \begin{bmatrix} L \\ H \end{bmatrix} \Pi_0 \begin{bmatrix} L^* & H^* \end{bmatrix}$$

have the same inertia, and if for all $j = 0, \ldots, i$ there exist J-unitary matrices, Θ_j, (where $J = (-I_q) \oplus I_p \oplus S$) such that

$$\begin{bmatrix} R_{e,j}^{1/2} & \begin{bmatrix} L \\ H \end{bmatrix} M_j \\ K_{p,j} R_{e,j}^{1/2} & F M_j \end{bmatrix} \Theta_j = \begin{bmatrix} R_{e,j+1}^{1/2} & 0 \\ K_{p,j+1} R_{e,j+1}^{1/2} & M_{j+1} \end{bmatrix} \tag{5.19}$$

with $R_{e,j}^{1/2}$ and $R_{e,j+1}^{1/2}$ block lower triangular. The algorithm is initialized with, $R_{e,0}$, $K_{p,0} = F\Pi_0 \begin{bmatrix} L^* & H^* \end{bmatrix} R_{e,0}^{-1}$ and

$$P_1 - \Pi_0 = F\Pi_0 F^* + GQG^* - K_{p,0}R_{e,0}K_{p,0}^* - \Pi_0 = M_0 S M_0^*.$$

The gain matrix $K_{a,i}$ needed to update the estimates in

$$\hat{x}_{j+1} = F_j\hat{x}_j + K_{a,j}(y_j - F_j\hat{x}_j), \qquad \hat{x}_0 = 0,$$

is equal to

$$K_{a,j} = \bar{K}_{a,j}(I + H_j\tilde{P}_jH_j^*)^{-1/2},$$

where $\bar{K}_{a,j}$ is given by the second block column of $\bar{K}_{p,j} = K_{p,j}R_{e,j}^{1/2}$, and $(I + H_j\tilde{P}_jH_j^*)^{1/2}$ is given by the $(2,2)$ block entry of $R_{e,j}^{1/2}$.

6 Conclusion

In this paper, we developed square-root and fast array algorithms for the H^∞ a posteriori and a priori filtering problems. These algorithms involve propagating the indefinite square-roots of the quantities of interest, and have the interesting property that the appropriate inertia of these quantities is preserved. Moreover, the conditions for the existence of the H^∞ filters are built into the algorithms, so that filter solutions will exist if,and only if, the algorithms can be executed.

The conventional square-root and fast array algorithms are preferred because of their better numerical behaviour (in the case of square-root arrays) and their reduced computational complexity (in the case of the fast recursions). Since the H^∞ square-root and fast array algorithms are the direct analogs of their conventional counterparts, they may be more attractive for numerical implementations of H^∞ filters. However, since J-unitary rather than unitary operations are involved, further numerical investigation is needed.

Our derivation of the H^∞ square-root and fast array algorithms demonstrates a virtue of the Krein space approach to H^∞ estimation and control; the results appear to be more difficult to conceive and prove in the traditional H^∞ approaches. We should also mention that there are many variations of the conventional square-root and fast array algorithms, *e.g.* for control problems, and the methods given here are directly applicable to extending these variations to the H^∞ setting as well. Finally, the algorithms presented here are equally applicable to risk-sensitive estimation and control problems, and to quadratic dynamic games.

7 References

[1] P. Dyer and S. McReynolds. Extension of square-root filtering to include process noise. *J. Optimiz. Theory Appl.*, 3:444–459, 1969.

[2] P.G. Kaminski, A.E. Bryson, and S.F. Schmidt. Discrete square-root filtering: A survey of current techniques. *IEEE Transactions on Automatic Control*, 16:727–735, Dec. 1971.

[3] H.L. Harter. The method of least-squares and some alternatives. Technical Report ARL 72-0129, Aerospace Res. Lab., Air Force Systems Command, Wright-Patterson AFB, Ohio, Sept. 1972.

[4] P. Businger and G.H. Golub. Linear least-squares solutions by Householder transformation. *Math. Comput.*, 20:325–328, 1966.

[5] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.

[6] R.A. Horn and C.R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, 1990.

[7] G. Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, Wellesley, MA, 1993.

[8] T. Kailath. Some new algorithms for recursive estimation in constant linear systems. *IEEE Transactions on Information Theory*, 19(6):750–760, Nov. 1973.

[9] M. Morf, G.S. Sidhu, and T. Kailath. Some new algorithms for recursive estimation in constant, linear, discrete-time systems. *IEEE Transactions on Automatic Control*, 19:315–323, 1974.

[10] A.H. Sayed and T. Kailath. Extended Chandrasekhar recursions. *IEEE Transactions on Automatic Control*, 39(3):619–623, March 1994.

[11] A.V. Ambartsumian. Diffuse reflection of light by a foggy medium. *Dokl. Akad. Sci. SSSR*, 38:229–322, 1943.

[12] M. Morf and T. Kailath. Square-root algorithms for linear least squares estimation. *IEEE Transactions on Automatic Control*, 20(4):487–497, 1975.

[13] T. Kailath, S.Y. Kung, and M. Morf. Displacment ranks of matrices and linear equations. *J. Math. Analysis & Appls.*, 68(2):395–407, April 1979.

[14] T. Kailath and A.H. Sayed. Displacment structure: theory and applications. *SIAM Review*, 37(3):297–386, Sep. 1995.

[15] T. Basar. Optimum performance levels for minimax filters, predictors and smoothers. *Systems and Control Letters*, 16:309–317, 1991.

[16] P.P. Khargonekar and K.M. Nagpal. Filtering and smoothing in an H^{∞} setting. *IEEE Trans. on Automatic Control*, 36:151–166, 1991.

[17] U. Shaked and Y. Theodor. $H^\infty-$optimal estimation: A tutorial. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2278–2286, Tucson, AZ, 1992.

[18] M. Green and D.J.N. Limebeer. *Linear Robust Control.* Prentice-Hall, Englewood Cliffs, NJ, 1995.

[19] B. Hassibi, A.H. Sayed, and T. Kailath. Linear estimation in Krein spaces - Part I: Theory. *IEEE Transactions on Automatic Control*, 41(1):18–33, Jan. 1996.

[20] B. Hassibi, A.H. Sayed, and T. Kailath. Linear estimation in Krein spaces - Part II: Applications. *IEEE Transactions on Automatic Control*, 41(1):34–49, Jan. 1996.

[21] A.S. Householder. *Principles of Numerical Analysis.* McGraw-Hill, New York, 1953.

[22] J.H. Moreno and T. Lang. *Matrix Computations on Systolic-Type Arrays.* Kluwer Academic Publishers, Boston, MA, 1992.

[23] I. Yaesh and U. Shaked. $H_\infty-$optimal estimation - The discrete time case. In *Proc. of the Mathematical Theory of Networks and Systems*, pages 261–267, Kobe, Japan, June 1991.

[24] B. Hassibi. *Indefinite Metric Spaces in Estimation, Control and Adaptive Filtering.* PhD thesis, Stanford University, 1996.

[25] D. Mustafa and K. Glover. *Minimum Entropy H^∞ Control.* Springer-Verlag, New York, 1990.

[26] P. Whittle. *Risk-sensitive Optimal Control.* John Wiley & Sons, New York, 1990.

[27] T. Basar and P. Bernhard. *H^∞-optimal Control and Related Minimax Design Problems: A Dynamic Games Approach.* Birkhauser, Boston, 1995.

[28] T. Kailath, A.H. Sayed, and B. Hassibi. *State Space Estimation.* Prentice-Hall, Englewood Cliffs, NJ, 1997.

[29] T. Kailath. *Lectures on Wiener and Kalman Filtering.* Springer-Verlag, New York, 1981.

[30] N.J. Higham. *Accuracy and Stability of Numerical Algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[31] G.W. Stewart. *Introduction to Matrix Computations.* Academic Press, New York, 1973.

[32] W.M. Gentleman. Least-squares computations by Givens transformations without square-roots. *J. Inst. Math. Appl.*, 12:329–336, 1973.

[33] S.F. Hsieh, K.J.R. Liu, and K. Yao. A unified square-root-free approach for QRD-based recursive-least-squares estimation. *IEEE Transactions on Signal Processing*, 41(3):1405–1409, March 1993.

[34] A.W. Bojanczyk and A.O. Steinhardt. Stability analysis of a Householder-based algorithm for downdating the Cholesky factorization. *SIAM Journal on Scientific and Statistical Computing*, 12(6):1255–1265, 1991.

[35] S. Chandrasekaran and A.H. Sayed. Stabilizing the generalized Schur algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):950–983, 1996.

A Unitary and Hyperbolic Rotations

In this appendix we review three families of elementary (unitary and hyperbolic) transformations that can be used to annihilate selected entries in a vector: the Householder, Givens, and fast Givens transformations. For more details, and historical background, see [30] (Ch. 18). Special care needs to be taken when dealing with complex-valued data as compared to real-valued data, as we show in the sequel.

A.1 Elementary Householder Transformations

Suppose we wish to simultaneously annihilate several entries in a row vector via a unitary transformation, say to transform an $n-$dimensional vector

$$x = \begin{bmatrix} x_1 & x_2 & \dots & x_{n-1} \end{bmatrix}$$

to the form

$$\begin{bmatrix} \alpha & 0 & 0 & 0 \end{bmatrix} ,$$

where, for general complex data, the resulting α may be complex as well.

One way to achieve this transformation is to employ a so-called *Householder reflection* Θ, which takes a row vector x and aligns it along the direction of the basis vector

$$e_0 = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} .$$

More precisely, it performs the transformation

$$\begin{bmatrix} x_1 & x_2 & \dots & x_{n-1} \end{bmatrix} \Theta = \alpha \, e_0. \tag{A.1}$$

Since, as we shall promptly verify, the transformation Θ that we shall employ is not only unitary but also Hermitian, we can be more specific about the resulting α. In particular, it follows from (A.1) that the magnitude of α must be equal to $\|x\|$, *i.e.,* $|\alpha| = \|x\|$. This is because

$$x \underbrace{\Theta\Theta^*}_{I} x^* \;=\; \|x\|^2 \;=\; |\alpha|^2 \; .$$

Moreover, it also follows from (A.1) that

$$x\Theta x^* = \alpha x_1^* \; .$$

But since Θ will be Hermitian, we conclude that $x\Theta x^*$ is a real number and, hence, αx_1^* must be real as well.

This means that by rotating a vector x with a unitary and Hermitian transformation Θ we can achieve a post-array of the form $\begin{bmatrix} \alpha & 0 & \ldots & 0 \end{bmatrix}$, where α will in general be a complex number whose magnitude is the norm of $\|x\|$ and whose phase is such that αx_1^* is real. For example, $\alpha = \pm\|x\|e^{j\phi_{x_1}}$ are possible values for α, where ϕ_{x_1} denotes the phase of x_1. [For real data, $\alpha = \pm\|x\|$ are the possible values for α.]

Now, assume we define

$$\Theta = I - 2\,\frac{g^*g}{gg^*} \quad \text{where} \quad g = x + \alpha\, e_0 \; , \tag{A.2}$$

and α is the complex number chosen above, *i.e.*, $\alpha = \pm\|x\|e^{j\phi_{x_1}}$. It can be verified by direct calculation that, for any g, Θ is a unitary matrix, *i.e.,*

$$\Theta\Theta^* = I = \Theta^*\Theta \; .$$

It is also clearly Hermitian.

Lemma 2 (Complex Householder Transformation) *Given a row vector x with leading entry x_1, define Θ and g as in (A.2) where α is any complex number that satisfies the following two requirements: $|\alpha| = \|x\|$ and αx_1^* is real. Then it holds that*

$$x\Theta = -\alpha e_0 \; .$$

That is, x is rotated and aligned with e_0; the leading entry of the post-array is equal to $-\alpha$.

(**Algebraic**) **proof**: We shall provide a geometric proof below. Here we verify our claim algebraically. Indeed, direct calculation shows that

$$\begin{aligned} gg^* &= \|x\|^2 + \alpha^* x_1 + \alpha x_1^* + |\alpha|^2 \; , \\ &= 2\|x\|^2 + 2\alpha x_1^* \; , \qquad \text{since } |\alpha|^2 = \|x\|^2 \text{ and } \alpha^* x_1 = \alpha x_1^* . \end{aligned}$$

Likewise,

$$\begin{aligned} xg^*g &= x\|x\|^2 + \alpha\|x\|^2 e_0 + \alpha^* x_1 x + |\alpha|^2 x_1 e_0 \ , \\ &= x\|x\|^2 + \alpha\|x\|^2 e_0 + \alpha^* x_1 x + \alpha(\alpha^* x_1) e_0 \ , \\ &= x\|x\|^2 + \alpha\|x\|^2 e_0 + \alpha^* x_1 x + \alpha(\alpha x_1^*) e_0 \ , \\ xgg^* &= 2x\|x\|^2 + 2\alpha x_1^* x \ , \end{aligned}$$

and we obtain

$$x\Theta = \frac{xgg^* - 2xg^*g}{gg^*} = \frac{-(2\|x\|^2 + 2\alpha x_1^*)\alpha e_0}{2\|x\|^2 + 2\alpha x_1^*} \ = \ -\alpha e_0 \ .$$

■

A choice for α, and hence g, is

$$g = x \pm e^{j\phi_{x_1}} \|x\| \ e_0 \ . \tag{A.3}$$

It leads to

$$x\Theta = \mp e^{j\phi_{x_1}} \|x\| e_0. \tag{A.4}$$

The choice of the sign in (A.4) depends on the choice of the sign in the expression for g. Usually, the sign in the expression for g is chosen so as to avoid a vector g of small Euclidean norm, since this norm appears in the denominator of the expression defining Θ. [In the real case, this can be guaranteed by choosing the sign in the expression for g to be the same as the sign of the leading entry of the row vector x, viz., the sign of x_1.]

A Geometric Derivation

The result of the above lemma has a simple geometric interpretation. Given a vector x, we would like to rotate it and align it with the vector e_0.This rotation should keep the norm of x unchanged. Hence, the tip of the vector x should be rotated along a circular trajectory until it becomes aligned with e_0. The vector aligned with e_0 is equal to αe_0. Here, as indicated in Figure 2, we are assuming that the rotation is performed in the clockwise direction. The triangle with sides x and αe_0 and base $g = x - \alpha e_0$ is then an isosceles triangle.

We thus have $\alpha e_0 = x - g$. But we can also express this in an alternative form. If we drop a perpendicular (denoted by $g^{\perp}$) from the origin of x to the vector g, it will divide g into two equal parts. In fact, the upper part is nothing but the projection of the vector x onto the vector g and is thus equal to $\langle x, g \rangle \ \|g\|^{-2} \ g$. Therefore,

$$\alpha e_0 = x - 2\langle x, g\rangle \|g\|^{-2} g = x - 2xg^*(gg^*)^{-1} g = x \underbrace{\left[I - 2\frac{g^*g}{gg^*} \right]}_{\Theta},$$

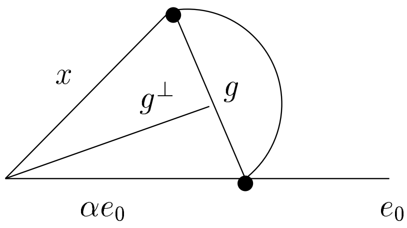

FIGURE 2. *Geometric interpretation of the Householder transformation.*

which leads to the Householder transformation of (A.4).

Note that a similar argument holds for the choice $g = \alpha e_0 + x$. The Householder transformation *reflects* the vector x across the line $g^{\perp}$ to the vector αe_0. So it is often called a Householder reflection.

Triangularizing a Matrix

A sequence of Householder transformations of this type can be used to triangularize a given $m \times n$ matrix, say A. For this we first find a transformation Θ_0 to rotate the first row to lie along e_0, so that we have $A\Theta_0$ of the form (where $\times$ denotes entries whose exact values are not of current interest):

$$A\Theta_0 = \begin{bmatrix} \sigma_1 & \vdots & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \times & \vdots & & & & \\ \times & \vdots & & & A_1 & \\ \times & \vdots & & & & \end{bmatrix}.$$

Now apply a transformation of the type

$$\begin{bmatrix} 1 & \vdots & \\ \cdots & \cdots & \cdots \\ & \vdots & \Theta_1 \\ & \vdots & \end{bmatrix},$$

where Θ_1 rotates the first row of A_1 so that it lies along e_0 in an $(n-1)$-dimensional space, and so on. This so-called Householder reduction of matrices has been found to be an efficient and stable tool for displaying rank information via matrix triangularization and it is widely used in numerical analysis (see, *e.g.*, [31, 5, 30]).

A.2 Elementary Circular or Givens Rotations

An elementary 2×2 unitary rotation Θ (also known as Givens or circular rotation) takes a 1×2 row vector $x = \begin{bmatrix} a & b \end{bmatrix}$ and rotates it to lie along the basis vector $e_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}$. More precisely, it performs the transformation

$$\begin{bmatrix} a & b \end{bmatrix} \Theta = \begin{bmatrix} \alpha & 0 \end{bmatrix} , \tag{A.5}$$

where, for general complex data, α may be complex as well. Furthermore, its magnitude needs to be consistent with the fact that the pre-array, $\begin{bmatrix} a & b \end{bmatrix}$, and the post-array, $\begin{bmatrix} \alpha & 0 \end{bmatrix}$, must have equal Euclidean norms since. In other words, α must satisfy

$$|\alpha| = \sqrt{|a|^2 + |b|^2} \, .$$

An expression for Θ that achieves the transformation (A.5) is given by

$$\Theta = \frac{1}{\sqrt{1+|\rho|^2}} \begin{bmatrix} 1 & -\rho \\ \rho^* & 1 \end{bmatrix} \quad \text{where} \quad \rho = \frac{b}{a} \, , \quad a \neq 0. \tag{A.6}$$

In the trivial case $a = 0$, we simply choose Θ to be the permutation matrix,

$$\Theta = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} .$$

We finally note that, in the special case of real data, a general unitary rotation as in (A.6) can be expressed in the alternative form:

$$\Theta = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

where the so-called cosine and sine parameters, c and s, respectively, are defined by

$$c = \frac{1}{\sqrt{1+|\rho|^2}} \quad , \quad s = \frac{\rho}{\sqrt{1+|\rho|^2}} \, .$$

This justifies the name *circular rotation* for Θ, since the effect of Θ is to rotate a vector x along a circle of radius $\|x\|$, by an angle θ that is determined by the inverse of the above cosine and/or sine parameters, $\theta = \tan^{-1} \rho$, in order to align it with the basis vector $\begin{bmatrix} 1 & 0 \end{bmatrix}$. The trivial case $a = 0$ corresponds to a 90 degrees rotation in an appropriate clockwise (if $b \geq 0$) or anti-clockwise (if $b < 0$) direction.

Triangularizing a Matrix

Matrix triangularization can also be effected by a product of Givens transformations, each of which introduces a zero in a particular location. For example, suppose that

$$x = [\ldots, x_i, \ldots, x_j, \ldots] \, ,$$

and we wish to null out x_j using x_i. Then let $\rho = x_j/x_i$, and define

$$\Theta = \begin{bmatrix} 1 &&&&&&& \\ & 1 &&&&&& \\ && \frac{1}{\sqrt{1+|\rho|^2}} &&& \frac{-\rho}{\sqrt{1+|\rho|^2}} && \\ &&& 1 &&&& \\ &&&& 1 &&& \\ && \frac{\rho^*}{\sqrt{1+|\rho|^2}} &&& \frac{1}{\sqrt{1+|\rho|^2}} && \\ &&&&&& 1 & \\ &&&&&&& 1 \end{bmatrix},$$

Except for the ρ terms, all off-diagonal entries are 0. Then we can see that

$$x\Theta = [\ldots, \alpha, \ldots, 0, \ldots] .$$

where the entries indicated by ... are arbitrary and unchanged by Θ, while the resulting α will be of the general form

$$\alpha \;=\; \pm\, e^{j\phi_{x_i}}\, \sqrt{x_i^2 + x_j^2}\; .$$

To systematically triangularize a $p \times p$ matrix A, apply a sequence of $p-1$ such transformations to zero all entries in the first row of A expect for the first element. Proceed to the second row of the thus transformed A matrix and apply a sequence of $p-2$ transformations to zero all entries after the second one. The first row has a zero in every column affected by this sequence of transformations, so it will be undisturbed. Continuing in this fashion for all rows except the last one, we will transform A to lower triangular form.

In general, the Givens method of triangularization requires more computations than the Householder method. It requires about 30% more multiplications, and it requires one scalar square root per zero produced as opposed to one per column for the Householder method. However, the Givens method is more flexible in preserving zeros already present in the A matrix and can require fewer computations than the Householder method when A is nearly triangular to begin with (see [32]). Moreover, there is a fast version, presented next, that uses 50% fewer multiplications.

A.3 Fast Givens Transformations

In [32] it is also shown how proper prescaling can be used to avoid arithmetic square roots in the Givens algorithm. Moreover, the resulting algorithm uses half as many multiplications. The idea is to introduce weighted norms, so that the rotational invariance is expressed as

$$pD_p p^* = qD_q q^* \;, \tag{A.7}$$

where $\{p, q\}$ are given 1×2 row vectors and $\{D_p, D_q\}$ are diagonal weighting matrices with positive elements. D_p can be arbitrary and D_q is selected so that there are no arithmetic square-roots in the actual transformation.

We will not present the details of the fast Givens transformation here. Interested readers can refer to [32]. We should remark that there are several variants of such fast rotations, including some that avoid both arithmetic square-roots and divisions (see, *e.g.,* [33]). These modified Givens transformations tend to suffer from possible overflow/underflow problems — so called self-scaling fast Givens rotations have also been studied.

A.4 Hyperbolic Transformations

In H^∞ array algorithms, as well as in fast (Chandrasekhar) array algorithms, it is necessary to use hyperbolic transformations, rather than unitary transformations. We therefore exhibit here the necessary modifications.

Elementary Hyperbolic Rotations

An elementary 2×2 hyperbolic rotation Θ takes a row vector $x = \begin{bmatrix} a & b \end{bmatrix}$ and rotates it to lie either along the basis vector $e_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}$ (if $|a| > |b|$) or along the basis vector $e_1 = \begin{bmatrix} 0 & 1 \end{bmatrix}$ (if $|a| < |b|$). More precisely, it performs either of the transformations:

$$\begin{bmatrix} a & b \end{bmatrix} \Theta = \begin{bmatrix} \alpha & 0 \end{bmatrix} \qquad \text{if } |a| > |b| \tag{A.8}$$

$$\begin{bmatrix} a & b \end{bmatrix} \Theta = \begin{bmatrix} 0 & \alpha \end{bmatrix} \qquad \text{if } |a| < |b| \tag{A.9}$$

where, for general complex data, α may be complex as well. Furthermore, its magnitude needs to be consistent with the fact that the pre-array, $\begin{bmatrix} a & b \end{bmatrix}$, and the post-array, $\begin{bmatrix} \alpha & 0 \end{bmatrix}$, must have equal Euclidean $J-$norms, *e.g.,* when $|a| > |b|$ we get

$$\begin{bmatrix} a & b \end{bmatrix} \underbrace{\Theta J \Theta^*}_{J} \begin{bmatrix} a^* \\ b^* \end{bmatrix} = \begin{bmatrix} \alpha & 0 \end{bmatrix} J \begin{bmatrix} \alpha^* \\ 0 \end{bmatrix} ,$$

where

$$J = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = (1 \oplus -1) .$$

By the $J-$norm of a row vector x we mean the indefinite quantity xJx^*, which can be positive, negative, or even zero. Hence, for $|a| > |b|$, α must satisfy

$$|\alpha|^2 \; = \; |a|^2 - |b|^2 ,$$

and its magnitude should therefore be equal to

$$|\alpha| = \sqrt{|a|^2 - |b|^2} .$$

When $|b| > |a|$ we should get

$$|\alpha| = \sqrt{|b|^2 - |a|^2}\ .$$

An expression for a $J-$unitary hyperbolic rotation Θ that achieves (A.8) or (A.9) is given by

$$\Theta = \frac{1}{\sqrt{1-|\rho|^2}} \begin{bmatrix} 1 & -\rho \\ -\rho^* & 1 \end{bmatrix} \quad \text{where} \quad \rho = \frac{b}{a}\ , \ \text{if} \ |a| > |b| \tag{A.10}$$

$$\Theta = \frac{1}{\sqrt{1-|\rho|^2}} \begin{bmatrix} 1 & -\rho \\ -\rho^* & 1 \end{bmatrix} \quad \text{where} \quad \rho^* = \frac{a}{b}\ , \ \text{if} \ |a| < |b| \tag{A.11}$$

For real data, a general hyperbolic rotation as in (A.10) or (A.11) can be expressed in the alternative form:

$$\Theta = \begin{bmatrix} ch & -sh \\ -sh & ch \end{bmatrix}$$

where the so-called hyperbolic cosine and sine parameters, ch and sh, respectively, are defined by

$$ch = \frac{1}{\sqrt{1-|\rho|^2}}\ , \quad sh = \frac{\rho}{\sqrt{1-|\rho|^2}}\ .$$

This justifies the name *hyperbolic rotation* for Θ, since the effect of Θ is to rotate a vector x along the *hyperbola* of equation

$$x^2 - y^2 = |a|^2 - |b|^2,$$

by an angle θ that is determined by the inverse of the above hyperbolic cosine and/or sine parameters, $\theta = \tanh^{-1}\rho$, in order to align it with the appropriate basis vector. Note also that the special case $|a| = |b|$ corresponds to a row vector $x = \begin{bmatrix} a & b \end{bmatrix}$ with zero hyperbolic norm since $|a|^2 - |b|^2 = 0$. It is then easy to see that there does not exist a hyperbolic rotation that will rotate x to lie along the direction of one basis vector or the other.

There exist alternative implementations of the hyperbolic rotation Θ that exhibit better numerical properties. Here we briefly mention two modifications.

Mixed Downdating

Assume we apply a hyperbolic rotation Θ to a row vector $\begin{bmatrix} x & y \end{bmatrix}$, say

$$\begin{bmatrix} x_1 & y_1 \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \frac{1}{\sqrt{1-|\rho|^2}} \begin{bmatrix} 1 & -\rho \\ -\rho^* & 1 \end{bmatrix}\ . \tag{A.12}$$

Then, more explicitly,

$$x_1 = \frac{1}{\sqrt{1-|\rho|^2}}\left[x - \rho^* y\right] , \tag{A.13}$$

$$y_1 = \frac{1}{\sqrt{1-|\rho|^2}}\left[-\rho x + y\right] . \tag{A.14}$$

Solving for x in terms of x_1 from the first equation and substituting into the second equation we obtain

$$y_1 = -\rho x_1 + \sqrt{1-|\rho|^2}\, y . \tag{A.15}$$

An implementation that is based on (A.13) and (A.15) is said to be in mixed downdating form. It has better numerical stability properties than a direct implementation of Θ as in (A.12) – see [34].

In the above mixed form, we first evaluate x_1 and then use it to compute y_1. We can obtain a similar procedure that first evaluates y_1 and then uses it to compute x_1. For this purpose, we solve for y in terms of y_1 from (A.14) and substitute into (A.13) to obtain

$$x_1 = -\rho^* y_1 + \sqrt{1-|\rho|^2}\, x . \tag{A.16}$$

Eqs. (A.14) and (A.16) represent the second mixed form.

The OD Method

The OD (Orthogonal-Diagonal) procedure is based on using the SVD of the hyperbolic rotation Θ. Assume ρ is real and write $\rho = b/a$, where $|a| > |b|$. Then it is straightforward to verify that any hyperbolic rotation of this form admits the following eigen-decomposition:

$$\Theta = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} \sqrt{\frac{a+b}{a-b}} & 0 \\ 0 & \sqrt{\frac{a-b}{a+b}} \end{bmatrix}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}\frac{1}{\sqrt{2}} \triangleq QDQ^T, \tag{A.17}$$

where the matrix

$$Q = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

is orthogonal ($QQ^T = I$), and T denotes transposition.

Due to the special form of the factors (Q, D), a real hyperbolic rotation, with $|\rho| < 1$ can then be applied to a row vector $\begin{bmatrix} x & y \end{bmatrix}$ to yield $\begin{bmatrix} x_1 & y_1 \end{bmatrix}$ as follows (note that the first and last steps involve simple additions and subtractions):

$$\begin{bmatrix} x' & y' \end{bmatrix} \leftarrow \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x'' & y'' \end{bmatrix} \leftarrow \begin{bmatrix} x' & y' \end{bmatrix} \begin{bmatrix} \frac{1}{2}\sqrt{\frac{a+b}{a-b}} & 0 \\ 0 & \frac{1}{2}\sqrt{\frac{a-b}{a+b}} \end{bmatrix}$$

$$\begin{bmatrix} x_1 & y_1 \end{bmatrix} \leftarrow \begin{bmatrix} x'' & y'' \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

This procedure is numerically stable, as shown in [35]. An alternative so-called H-procedure is also described in the same reference. It is costlier than the OD method, but is more accurate and can be shown to be "forward" stable, which is a very desirable property for finite precision implementations.

When ρ is a complex number, the unitary matrix Q becomes complex. If we now write $\rho = b/a$, then $|\rho| = |b|/|a|$ and $|a| > |b|$, and we obtain

$$Q = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -\frac{|a|}{a}\frac{b}{|b|} & \frac{|a|}{a}\frac{b}{|b|} \end{bmatrix} , \quad D = \begin{bmatrix} \frac{\sqrt{|a|+|b|}}{\sqrt{|a|-|b|}} & \\ & \frac{\sqrt{|a|-|b|}}{\sqrt{|a|+|b|}} \end{bmatrix} .$$

Hyperbolic Householder Transformations

Let J be an $n \times n$ signature matrix such as

$$J = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} , \quad p + q = n.$$

We are now interested in a $J-$unitary Householder transformation Θ that takes a $1 \times n$ row vector x and aligns it either along the basis vector $e_0 = \begin{bmatrix} 1 & 0 \end{bmatrix}$ (if $xJx^* > 0$) or along the basis vector $e_{n-1} = \begin{bmatrix} 0 & 1 \end{bmatrix}$ (if $xJx^* < 0$).

Hence, we require Θ to perform either of the transformations

$$x\Theta = \pm\alpha \, e_0 \quad \text{if} \quad xJx^* > 0 \tag{A.18}$$

$$x\Theta = \pm\alpha \, e_{n-1} \quad \text{if} \quad xJx^* < 0 \, , \tag{A.19}$$

where, for general complex data, the resulting α may be complex as well.

When $xJx^* > 0$, we define

$$\Theta = I - 2\,\frac{Jg^*g}{gJg^*} \quad \text{where} \quad g = x + \alpha \, e_0 \, , \tag{A.20}$$

and α is a complex number that satisfies $|\alpha|^2 = xJx^*$ and αx_1^* is real. It can be verified by direct calculation that Θ is a $J-$unitary matrix, *i.e.,*

$$\Theta J \Theta^* = J = \Theta^* J \Theta \, .$$

When $xJx^* < 0$, we use the same expression for Θ but with

$$g = x + \alpha\, e_{n-1}\ , \tag{A.21}$$

and α a complex number that satisfies $|\alpha|^2 = -xJx^*$ and αx^*_{n-1} is real.

Lemma 3 (Complex Hyperbolic Householder Transformation) . *Given a row vector x with leading entry x_1 and $xJx^* > 0$, define Θ and g as in (A.20) where α is any complex number that satisfies the following two requirements (see below): $|\alpha| = \sqrt{xJx^*}$ and αx^*_1 is real. Then it holds that*

$$x\Theta = -\alpha e_0\ .$$

That is, x is rotated and aligned with e_0; the leading entry of the post-array is equal to $-\alpha$.

For a vector x that satisfies instead $xJx^* < 0$, and with trailing entry x_{n-1}, we choose g as in (A.21) where α is any complex number that satisfies: $|\alpha| = \sqrt{|xJx^*|}$ and αx^*_{n-1} is real. Then it holds that

$$x\Theta = -\alpha e_{n-1}\ .$$

(**Algebraic**) **proof**: We prove the first statement only since the second one follows from a similar argument. Direct calculation shows that

$$\begin{aligned} gJg^* &= 2xJx^* + 2\alpha x^*_1\ , \\ xJg^*g &= x(xJx^*) + \alpha(xJx^*)e_0 + \alpha^* x_1 x + \alpha(\alpha x^*_1)e_0\ , \\ xgJg^* &= 2x(xJx^*) + 2\alpha x^*_1 x\ . \end{aligned}$$

Therefore,

$$x\Theta = \frac{xgJg^* - 2xJg^*g}{gJg^*} = -\alpha e_0\ .$$

■

Geometric Derivation

The geometric derivation presented earlier for Householder transformations still applies provided we use "J-inner products", *i.e.*, provided we interpret

$$\langle x, g\rangle_J = xJg^*.$$

Then we can write, for example, when $\|x\|_J = \sqrt{xJx^*}\ >\ 0$,

$$\mp\alpha e_0 = x - 2\langle x, g\rangle_J\ \|g\|_J^{-2}\ g\ =\ x\left(I - 2\frac{Jg^*g}{gJg^*}\right)\ ,$$

where

$$g \overset{\Delta}{=}\ x \pm \alpha\, e_0\ .$$

Table 1 collects the expressions for the several rotations that we have considered in the earlier discussion.

TABLE 1. *Unitary and hyperbolic rotations.*

Rotation	Expression	Effect
Circular or Givens	$\Theta = \frac{1}{\sqrt{1+\|\rho\|^2}} \begin{bmatrix} 1 & -\rho \\ \rho^* & 1 \end{bmatrix}$, $\rho = \frac{b}{a}$, $a \neq 0$.	$\begin{bmatrix} a & b \end{bmatrix} \Theta = \begin{bmatrix} \pm e^{j\phi_a}\sqrt{\|a\|^2 + \|b\|^2} & 0 \end{bmatrix}$.
Permutation	$\Theta = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $a = 0$.	$\begin{bmatrix} 0 & b \end{bmatrix} \Theta = \begin{bmatrix} b & 0 \end{bmatrix}$.
Hyperbolic I	$\Theta = \frac{1}{\sqrt{1-\|\rho\|^2}} \begin{bmatrix} 1 & -\rho \\ -\rho^* & 1 \end{bmatrix}$, $\rho = \frac{b}{a}$, $a \neq 0$, $\|a\| > \|b\|$.	$\begin{bmatrix} a & b \end{bmatrix} \Theta = \begin{bmatrix} \pm e^{j\phi_a}\sqrt{\|a\|^2 - \|b\|^2} & 0 \end{bmatrix}$.
Hyperbolic II	$\Theta = \frac{1}{\sqrt{1-\|\rho\|^2}} \begin{bmatrix} 1 & -\rho \\ -\rho^* & 1 \end{bmatrix}$, $\rho^* = \frac{a}{b}$, $b \neq 0$, $\|a\| < \|b\|$.	$\begin{bmatrix} a & b \end{bmatrix} \Theta = \begin{bmatrix} 0 & \pm e^{j\phi_b}\sqrt{\|b\|^2 - \|a\|^2} \end{bmatrix}$.
Unitary Householder	$\Theta = I_n - 2\,\frac{g^* g}{g g^*}$, $g = x \pm e^{j\phi_{x_1}} \\|x\\|\, e_0$.	$\begin{bmatrix} x_1 & \dots & x_{n-1} \end{bmatrix} \Theta = \mp e^{j\phi_{x_1}} \\|x\\| e_0$.
Hyperbolic Householder I	$\Theta = I - 2\,\frac{Jg^* g}{gJg^*}$, $g = x \pm e^{j\phi_{x_1}} \sqrt{\|xJx^*\|}\, e_0$, $xJx^* > 0$.	$\begin{bmatrix} x_1 & \dots & x_{n-1} \end{bmatrix} \Theta = \mp e^{j\phi_{x_1}} \sqrt{\|xJx^*\|} e_0$, $e_0 = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}$.
Hyperbolic Householder II	$\Theta = I - 2\,\frac{Jg^* g}{gJg^*}$, $g = x \pm e^{j\phi_{x_{n-1}}} \sqrt{\|xJx^*\|}\, e_{n-1}$, $xJx^* < 0$.	$\begin{bmatrix} x_1 & \dots & x_{n-1} \end{bmatrix} \Theta = \mp e^{j\phi_{x_{n-1}}} \sqrt{\|xJx^*\|} e_{n-1}$, $e_{n-1} = \begin{bmatrix} 0 & \dots & 0 & 1 \end{bmatrix}$.