Chapter 10

MINIMAL COMPLEXITY REALIZATION OF STRUCTURED MATRICES

Patrick Dewilde

10.1 INTRODUCTION

The earlier chapters considered the class of matrices that satisfy displacement equations (cf. Ch. 1) and, hence, have small displacement ranks. There are also other kinds of structured matrices. As a general working definition, we propose "matrices whose entries satisfy class generic constraints that reduce the number of algebraically free parameters."

Class generic constraints on the entries can be of several kinds:

- Linear constraints between entries. Examples are the following:
  - (i) Toeplitz, Hankel, or even Cauchy matrices.
  - (ii) Their generalizations to matrices of *low displacement rank*, as studied extensively by the Kailath school and its many ramifications (see Ch. 1). This is the class of matrices studied in the earlier chapters.
- Hard value constraints on entries. Examples are the following:
  - (i) Banded or multibanded matrices.
  - (ii) Inverses of banded matrices and (possibly continuous) products of banded matrices with inverses of banded matrices.
- Matrices described by a low-complexity time-varying state-space model.
- Nonlinear algebraic constraints. (Unitary matrices may seem to be of this type, but they can be brought into the class with linear constraints via the transformation $U = e^{iH}$ in which H is a Hermitian matrix and $i = \sqrt{-1}$.)

There are connections among the types described above. A banded matrix or its inverse has a low-complexity state-space realization; the collection of matrices described by a low-order state-space model is a generalization of the "banded" case. Products of these may also have low-complexity state-space realizations. An upper triangular Toeplitz matrix can be interpreted as a partial transfer operator of a time-varying linear

system and has a state-space model derived from it. Some generic constraints do not reduce the number of parameters involved. Positivity in one form or the other is one; the fact that a parameter or an algebraic expression involving parameters is restricted to an interval does not decrease the "algebraic freedom," at least not in infinite precision arithmetic. We shall not be concerned with such cases.

10.2 MOTIVATION OF MINIMAL COMPLEXITY REPRESENTATIONS

Why are we interested in structured matrices? We are interested for at least two reasons:

1. Structured matrices may be represented by (often many) fewer parameters.

2. Computations involving structured matrices may be much more efficient, sometimes at the cost of numerical accuracy or stability, but in important cases even improving on these two factors (see, e.g., the discussions in Sec. 1.13 and in Chs. 2, 3, and 4 in this book).

Two important additional reasons may be the following:

3. The reduced complexity is indicative of an underlying physical structure which is interesting in its own right.

4. The reduced complexity may lead to approximations and "model reduction," which reduce the number of necessary parameters even further.

The present chapter treats the combination of two entirely different methods of using matrix structure for parameter reduction: low displacement rank on the one hand and representation by low-order time-varying state models on the other. The first type has been pioneered by Kailath and his coworkers (Ch. 1 of this book gives a recent survey), while the second type was the subject of [VD94]. In the case of low-displacement-rank matrices, one computational advantage derives from the fact that in many cases the matrix or its inverse can be represented by a small sum of by-products of Toeplitz matrices. If the FFT algorithm is used to execute the product of a Toeplitz matrix with a vector, then the overall computational complexity of the matrix-vector multiplication is reduced to $\alpha nk \log n$, where α is the "displacement rank" and k is a small number depending on the type of FFT algorithm chosen. (See [Bla84] for detailed information—see also the discussion in Secs. 1.4 and 8.3.1 of this book.) On the other hand, matrices with low state representations (e.g., single band matrices) also give rise to reduced matrix-vector computations either with the original matrix or with its inverse, the computational complexity now being $2\delta n$, where δ is the maximal state complexity. We shall see further in the theory that matrices with low state representations may aptly be called "matrices with low Hankel rank," a term which we shall use in this chapter. (Another expression would be "with low local degree".)

Matrices with low displacement rank are not the same as matrices with low Hankel rank. One can easily exhibit examples of matrices which score high for one and low for

the other characteristic. For example, the Toeplitz matrix defined by

$$H \triangleq \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots \\ & 1 & \frac{1}{2} & \frac{1}{3} & \ddots \\ & & 1 & \frac{1}{2} & \ddots \\ & & & 1 & \ddots \\ & & & & \ddots \end{bmatrix}$$

has low displacement rank but does not have a useful low-order system representation, since each such representation would either decay exponentially or not decay at all with increasing index, while the original decays as $1/n$, just like the Maclaurin series for $\log(1+z)$. For matrices that score high for one of the criteria, it just pays to use the corresponding representation method. However, for matrices in the middle category, it may be advantageous to combine the two techniques. This may be reason enough to study the combination; there are more reasons, however, which we shall consider in the concluding section.

10.3 DISPLACEMENT STRUCTURE

Let R be an $n \times n$ positive-definite matrix (the entries of R may actually be $N \times N$ blocks; R is then a positive-definite $nN \times nN$ matrix overall), and let us define the lower triangular shift matrix Z with ones (or unit matrices) on the first subdiagonal and zeros elsewhere:

$$Z \triangleq \begin{bmatrix} 0 & & & \\ I & \ddots & & \\ & \ddots & \ddots & \\ & & I & 0 \end{bmatrix}.$$

We consider the displacement of R with respect to Z (cf. [KKM79a]—see also Ch. 1),

$$\nabla_Z R \triangleq R - ZRZ^*,$$

where $\cdot^*$ denotes Hermitian transpose, and assume that it has inertia $(p, 0, q)$. This means that there exist matrices (the a_i are the rows of the matrix G)

$$G = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}, \quad J = \begin{bmatrix} I_p & \\ & -I_q \end{bmatrix}$$

of dimensions $n \times (p+q)$ and $(p+q) \times (p+q)$, respectively, such that

$$R - ZRZ^* = GJG^* = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} J \begin{bmatrix} a_0^* & a_1^* & \cdots & a_{n-1}^* \end{bmatrix}, \tag{10.3.1}$$

in which G has full (column) rank. It is convenient to split each entry a_k according to the inertia formula

$$a_k = \left[\overbrace{a_{k1}}^{p} \quad \overbrace{a_{k2}}^{q} \right].$$

The integer $\alpha \triangleq p+q$ is called the *displacement rank* of R. If it is small compared with n, then R is said to be of low displacement rank and one can expect great simplifications of calculus with such matrices.

Another notion leading to low complexity representations of matrices is that of "low Hankel rank," defined in [DV93]. It leads to reduced state-space models for matrices. The derivation of such a model from the matrix viewed as an input-output operator is called a "realization theory."

10.4 REALIZATION THEORY FOR MATRICES

Let $T = [t_{ij}]$ be an upper triangular matrix. Such a matrix often represents a linear computation, say, of the type $y = uT$. The matrix T is applied linearly to an input row vector u to produce an output row vector y. Any computation, including the linear computation under consideration, will happen in stages. A major, but often hidden, assumption is that the input vector is presented in sequence; it takes the character of a signal that flows in the computation and is used sequentially. As soon as sufficient data are present, a partial calculation can start, produce an intermediate result, store it, output whatever data it has been able to generate (also in sequence), and then move to input new data, engage in a new partial calculation, and so on. Viewed in this way, a linear computation becomes what we would traditionally call a linear system.

Let us analyze such a computing system. We assume that at time k ($k = 1, \ldots, n$) the component u_k of the input vector together with some stored information gleaned from previous samples is used to compute the component y_k of the output vector. If we collect the relevant information from stage k as a vector x_k, which we call the *state*, then a data flow scheme of our computation would appear as in Fig. 10.1.

For example, we can write the ordinary vector-matrix multiplication $y = uT$ with scalar series u and y as

$$\begin{aligned} y_0 &= u_0 t_{00}, \\ y_1 &= u_0 t_{01} + u_1 t_{11}, \\ y_2 &= u_0 t_{02} + u_1 t_{12} + u_2 t_{22}, \\ \vdots &= \ddots \end{aligned} \tag{10.4.1}$$

As soon as u_0 is available, y_0 can be computed and outputed. For the computation of u_1 one needs to remember u_0, so the state at $t = 1$ must contain u_0, but then, as soon as u_1 becomes available, y_1 can be computed and outputed; for the computation of y_2, u_0 and u_1 may be needed, etc. If this continues, then the dimension of the state will quickly "explode." In many cases (as we shall see soon), a more clever choice of state leads to a more efficient choice of state vector at each time point.

The linear scheme of computations at each stage is shown in Fig. 10.2.

In the case of the straight vector-matrix multiplication shown in (10.4.1), we have

1. $x_1 = u_0$, $y_0 = u_0 t_{00}$; hence

$$B_0 = [1], \; D_0 = [t_{00}],$$

while A_0 and C_0 are empty since there is no initial state.

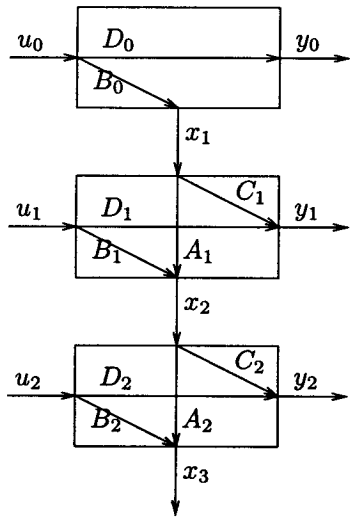

Figure 10.1. The realization of a linear computation.

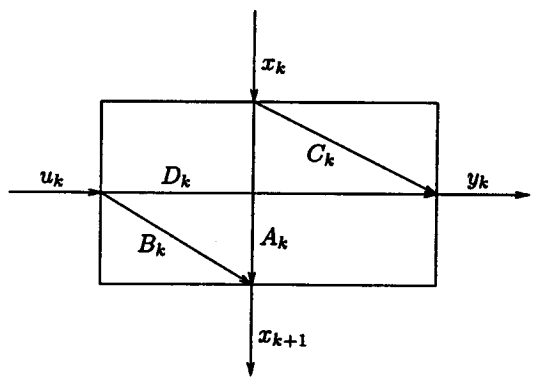

Figure 10.2. The local linear computing scheme.

2. $x_2 = [u_0\ u_1] = x_1[1\ 0] + u_1[0\ 1]$ and $y_1 = x_1 t_{01} + u_1 t_{11}$; hence

$$A_1 = [1\ 0],\ B_1 = [0\ 1],\ C_1 = [t_{01}],\ D_1 = [t_{11}].$$

3. $x_3 = [u_0\ u_1\ u_2] = x_2 \left[\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{smallmatrix}\right] + u_2[0\ 0\ 1]$ and $y_2 = x_2 \left[\begin{smallmatrix} t_{02} \\ t_{12} \end{smallmatrix}\right] + u_2 t_{22}$; hence

$$A_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},\ B_2 = [0\ 0\ 1],\ C_2 = \begin{bmatrix} t_{02} \\ t_{12} \end{bmatrix},\ D_2 = [t_{22}].$$

The principle should be clear! In any case, at each stage of the procedure linear computations take place connecting the present state x_k and input u_k with the next state x_{k+1} and output y_k:

$$\begin{cases} x_{k+1} &= x_k A_k + u_k B_k, \\ y_k &= x_k C_k + u_k D_k. \end{cases} \tag{10.4.2}$$

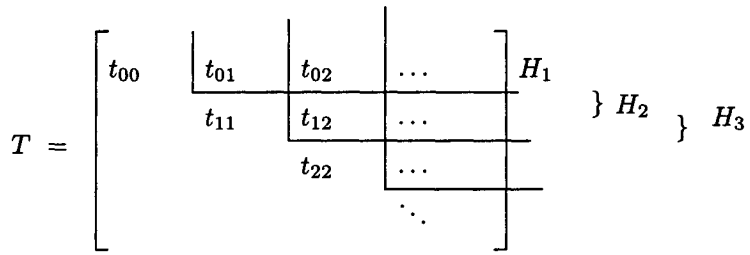

Figure 10.3. Collection of Hankel matrices of T.

However, there is no reason why the state cannot be a linear combination of past inputs, rather than the inputs themselves. There is also no reason why the calculations should be restricted to scalar inputs and outputs. In principle, all dimensions may vary; the input dimensions form a sequence $j_0, j_1, j_2, \ldots$ and likewise the output dimensions, $o_0, o_1, o_2, \ldots$, as well as the state dimensions, $\delta_0 = 0, \delta_1, \delta_2, \ldots$. Zero dimensions means that the entry is empty. (We agree, by definition, that the matrix multiplication of an empty matrix of dimension $m \times 0$ with one of dimension $0 \times n$ yields a zero matrix of dimension $m \times n$.) In the case of interest here, however, we work in a framework in which the input dimensions and output dimensions are the same for all k ($j_k = o_k = N$), in which N is a fixed number. (We keep all formulas as general as possible.)

The purpose of realization theory is to find a scheme of minimal state dimensions which computes $y = uT$. It turns out that such a scheme exists and that the minimal dimension of the state at stage k is given by the rank of a certain submatrix of T called the Hankel matrix H_k [VD94], which we now define. Our utilization of the term "Hankel matrix" here is not the traditional one (a matrix with second diagonals consisting of equal entries), but it is in the sense of mathematical system theory: a matrix that maps past inputs to future outputs. We define the *collection of Hankel matrices of* T as the set of matrices H_i represented by the scheme of Fig. 10.3.

More explicitly, each H_i is $(j_0 + j_1 + \cdots j_{i-1}) \times (o_i + o_{i+1} + \cdots + o_n)$ and given by

$$H_i = \begin{bmatrix} t_{0,i} & t_{0,i+1} & \cdots & t_{0,n-1} \\ \vdots & \vdots & \vdots & \vdots \\ t_{i-1,i} & \cdots & \cdots & t_{i-1,n-1} \end{bmatrix}. \tag{10.4.3}$$

We make the diagonal an exception in the definition. In most cases the diagonal is special or plays a special role, and it could be included in the calculation by moving all diagonals one notch up and introducing a new main diagonal consisting exclusively of zeros.

The important role that Hankel matrices play in realization theory can be deduced from a reconstruction of the matrix T, which is actually the transfer operator of the computational scheme of Fig. 10.1, based on the computation model

$$T = \begin{bmatrix} D_0 & B_0C_1 & B_0A_1C_2 & B_0A_1A_2C_3 & \cdots \\ & D_1 & B_1C_2 & B_1A_2C_3 & \cdots \\ & & D_2 & B_2C_3 & \cdots \\ & & & \ddots & \vdots \\ & & & & \vdots \end{bmatrix},$$

and hence

$$H_k = \begin{bmatrix} \vdots \\ B_{k-3}A_{k-2}A_{k-1} \\ B_{k-2}A_{k-1} \\ B_{k-1} \end{bmatrix} \begin{bmatrix} C_k & A_kC_{k+1} & \cdots \end{bmatrix}.$$

We see that the realization induces a factorization of each Hankel operator. A direct conclusion is that their ranks are at most equal to the dimensions of the respective local state spaces, $\text{rank}(H_k) = \delta_k$, where δ_k is the minimal dimension of x_k. The converse appears true as well, and it is also true that *any* minimal factorization of each of the Hankel matrices induces a specific realization, so that the corresponding $\{A_k, B_k, C_k, D_k\}$ can be recovered from them. A "physical interpretation" of this fact goes back to the celebrated Nerode state equivalence theory, which is briefly reviewed next.

10.4.1 Nerode Equivalence and Natural State Spaces

At each time point k, the Hankel operator maps input signals with support on time points up to and including $k-1$ to the restriction of output signals on the interval $[k, \infty)$. If T is the transfer operator concerned and $\mathbf{P}_{[k,\infty)}$ indicates projection on the ℓ_2 space based on the interval $[k, \infty)$ (we consider the general case), then H_k is given by

$$H_k = \mathbf{P}_{[k,\infty)}(T) \mid_{\ell_2(-\infty,k-1]}. \tag{10.4.4}$$

The image $\mathcal{H}_{ok}$ of H_k is the set of natural responses that the system is able to generate at time k, while the image $\mathcal{H}_k$ of H_k^* is the orthogonal complement of the nullspace at time k—the space of strict past inputs that generate the zero state. It is a space of equivalent classes, called *Nerode equivalent classes*, each of which represents a class of strictly past input signals that generate the same state, while the output space of natural responses is actually isomorphic to a natural state space. (See [KFA70] for an account of the original theory.) Hence, the state dimension is given by the dimension δ_k of H_k. To find an $\{A, C\}$ realization pair, we choose a basis for each $\mathcal{H}_{ok}$ and collect all those base vectors in one observability operator, as shown in Fig. 10.4.

If each block row in Fig. 10.4 is indeed a basis, then it is also left invertible and we see that the choice determines each A_k and C_k. The corresponding B_k and D_k then follow straightforwardly from knowledge of the transfer map T. The choice of a basis either in $\mathcal{H}_0$ or dually $\mathcal{H}$ determines the realization.

10.4.2 Algorithm for Finding a Realization

An algorithm for finding a minimal realization for T then simply proceeds as follows.

Algorithm 10.4.1 (Time Varying System Realization of T). *Consider an upper triangular matrix $T = [t_{ij}]$ and define its Hankel matrices as in* (10.4.3). *A realization (A_k, B_k, C_k, D_k) can be found as follows:*

1. *The D_k are obtained from the diagonal entries of T.*
2. *Find a minimal factorization of H_k as $H_k = \mathcal{R}_k\mathcal{O}_k$.*
3. *Put $B_{k-1} = [\mathcal{R}_k]_{k-1}$ (last block entry of $\mathcal{R}_k$).*

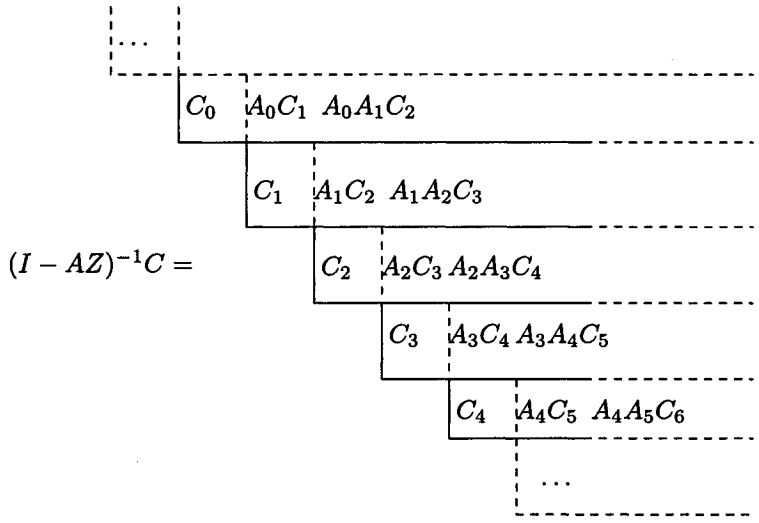

Figure 10.4. Choosing and representing a basis for the space of natural responses at each time point.

4. *Put* $C_k = [\mathcal{O}_k]_1$ *(first block entry of* $\mathcal{O}_k$*).*

5. *Find* A_k *so that*

$$\begin{bmatrix} [\mathcal{O}_k]_2 & [\mathcal{O}_k]_3 & \dots \end{bmatrix} = A_k \mathcal{O}_{k+1}. \tag{10.4.5}$$

◇

The matrix A_k is uniquely determined by the condition (10.4.5), since $\mathcal{O}_{k+1}$ is right invertible by the assumption of minimal factorization. The matrices $\mathcal{R}_k$ and $\mathcal{O}_k$ play a central role in system theory and are called the *reachability* and *observability* matrices of the realization. (For discussion on controllability and observability, see [KFA70], [Kai80], [DV93].)

The proof that the algorithm works requires us to show that if the realization is given by the algorithm, then it reproduces the entries $T_{k\ell} = B_k A_{k+1} \cdots A_{\ell-1} C_\ell$ exactly for all $0 \le k < \ell$ (in which expression some of the A_i factors can disappear when $\ell - k \le 2$). This can (fairly simply) be done as a recursion on $\ell - k$. Note first that the algorithm defines the entries of all the observability matrices $\mathcal{O}_k$: the $[\mathcal{O}_k]_1$ are given directly, while $[\mathcal{O}_k]_i$ follows from the definition of A_{i-1} and the value just below it, $[\mathcal{O}_{k+1}]_{i-1}$. Since all the values of the B_{k-1} also are known, we now have all the bottom rows of the H_k specified and hence all the upper off diagonal entries of T. (As stated above, the diagonal entries are assumed known, while the lower entries are all zero.) It is now easy to check that all the reachability matrices are well defined (just put $[\mathcal{R}]_i = B_{k-i} A_{k-i+1} \cdots A_{k-1}$ for $i > 1$) and that the H_k factor is as expected.

We can also obtain nonminimal realizations through nonminimal factorizations of the H_k, but then we have to be a little more careful. Examination of the proof in the previous paragraph shows that one way could be by producing a sequence of well-defined observability matrices and that this can be done by ensuring that for all k, the rows of $[[\mathcal{O}_k]_2, [\mathcal{O}_k]_3, \dots]$ lie in the row space of the subsequent $\mathcal{O}_{k+1}$. Then an A_k can be found for each k (although now it is not necessarily unique), and all the entries of the observability matrices follow recursively.

Dually, a realization could be based on the reachability matrices rather than on the observability matrices. It should be clear that once the choice for one has been made, the other follows automatically. Clearly, even minimal realizations are not unique. Each different factorization will produce a different realization: there is a one-to-one correspondence between the choice of bases for the observability (or dually reachability) spaces and to be minimal realizations. We say that a realization is in *output normal form* if the bases of all the observability spaces have been chosen to be orthonormal. In that case, the corresponding $\{A_k, C_k\}$ will be isometric, i.e., they will satisfy (for all $k \geq 0$)

$$A_k A_k^* + C_k C_k^* = I, \tag{10.4.6}$$

and vice versa. If (10.4.6) is satisfied, then the corresponding basis for the observability spaces is orthonormal. (The recursive proof is not difficult; we skip it for the sake of brevity.) The transformation of one minimal realization to another is accomplished via a transformation of the state at each point k. If we write, for each point k, $x'_k = T_k x_k$, in which T_k is an invertible matrix, then the state equation in the primed quantities becomes

$$\begin{cases} x'_{k+1} &= x'_k T_k A_k T_{k+1}^{-1} + u_k B_k T_{k+1}^{-1}, \\ y_k &= x'_k T_k C_k + u_k D_k, \end{cases} \tag{10.4.7}$$

and the transformed realization is given by $\{T_k A_k T_{k+1}^{-1}, B_k T_{k+1}^{-1}, T_k C_k, D_k\}$. In particular, suppose that we are given a realization and that we wish to bring it to normal form. Then we should find a collection $\{T_k\}$ such that the transformed $\{A'_k, C'_k\}$ satisfy (10.4.6). If we define $\Lambda_k = T_k^{-1}(T_k^*)^{-1}$, then this amounts to satisfying the recursive equation

$$A_k \Lambda_{k+1} A_k^* + C_k C_k^* = \Lambda_k. \tag{10.4.8}$$

In the remainder of the chapter we shall treat cases where this equation can indeed be satisfied uniquely by a (uniformily) invertible collection of $\{\Lambda_k\}$. It is known that the existence of such a solution to (10.4.8) requires uniform exponential stability of the sequence $\{A_k\}$. This is known as the Lyapunov condition, and we shall put ourselves in a situation where this condition is automatically satisfied.

Once a realization for an upper triangular operator T is obtained, it is easy to derive a realization for its inverse. If T is invertible, then that will also be the case for its main diagonal. If $\{A_k, B_k, C_k, D_k\}$ is a realization for T, then each $D_k = T_{kk}$ will also be invertible, and from the state equations we obtain

$$\begin{cases} x_{k+1} &= x_k(A_k - C_k D_k^{-1} B_k) + y_k D_k^{-1} B_k, \\ u_k &= -x_k C_k D_k^{-1} + y_k D_k^{-1}; \end{cases} \tag{10.4.9}$$

hence $\{A_k - C_k D_k^{-1} B_k, D_k^{-1} B_k, -C_k D_k^{-1}, D_k^{-1}\}$ provides a realization for T^{-1} of the same state complexity as the original.

Generalizing our framework, suppose that the original operator T is not upper triangular in the traditional sense. Then we have a number of options at our disposal to make it upper triangular in a generalized sense:

- We can shift the $(0,0)$th position to the bottom left corner. This strategy requires the introduction of a more general numbering scheme than that used so far; see the next section.
- We can additively decompose T as the sum of a lower triangular and an upper triangular component and realize each of them separately. Needless to say, this strategy will not be very useful when our actual purpose is to compute the inverse of T, but it may be very useful in other circumstances.

- We can also try to do a multiplicative decomposition of T into a lower triangular matrix that multiplies an upper triangular one (or vice versa); this strategy would yield good results when we wish to invert T subsequently.

The important point is that the low complexity representation technique remains valid for the components.

In the next section we shall discuss realizations for the additive and the multiplicative decompositions of a positive-definite matrix R, which presumably is of low displacement rank.

10.5 REALIZATION OF LOW DISPLACEMENT RANK MATRICES

The goal of the next two sections is the derivation of low-complexity representations for the additive and the multiplicative decomposition of a positive-definite matrix R. Let us write $R = [r_{ij}]$ as

$$R = U^*U = \frac{1}{2}(F + F^*), \tag{10.5.1}$$

in which U and F are upper triangular matrices. Then

$$F = \begin{bmatrix} r_{00} & 2r_{01} & 2r_{02} & \cdots \\ & r_{11} & 2r_{12} & \cdots \\ & & r_{22} & \cdots \\ & & & \ddots \end{bmatrix}.$$

If a low state dimension realization exists for F, then one can be found for U as well. Although this fact can be derived in general (it is a form of a spectral factorization result), we shall rederive it and specialize the calculation for the case at hand.

Let us write for simplicity $R - ZRZ^* \triangleq X$; then it is easy to see that R can be recovered from X via the formula

$$R = X + ZXZ^* + \cdots + Z^{n-1}X(Z^*)^{n-1}.$$

The contribution of each term to the Hankel operators for F is easy to evaluate. Indeed, consider the Hankel operator H_k for F. Then the contributions of the individual terms to H_k are

$$H_k(X) = \begin{bmatrix} a_0 \\ \vdots \\ a_{k-1} \end{bmatrix} J \begin{bmatrix} a_k^* & \cdots & a_{n-1}^* \end{bmatrix},$$

$$H_k(ZXZ^*) = \begin{bmatrix} 0 \\ a_0 \\ \vdots \\ a_{k-2} \end{bmatrix} J \begin{bmatrix} a_{k-1}^* & \cdots & a_{n-2}^* \end{bmatrix},$$

$$H_k(Z^{k-1}X(Z^*)^{k-1}) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ a_0 \end{bmatrix} J \begin{bmatrix} a_1^* & \cdots & a_{n-k}^* \end{bmatrix}.$$

Putting these terms together and using the outer product representation of a matrix, we get

$$H_k(F) = 2 \begin{bmatrix} a_0 J & 0 & \dots & 0 \\ a_1 J & a_0 J & \ddots & \\ \vdots & \ddots & \ddots & 0 \\ a_{k-1} J & \dots & \dots & a_0 J \end{bmatrix} \begin{bmatrix} a_k^* & \dots & \dots & a_{n-1}^* \\ \vdots & & & \vdots \\ a_2^* & \dots & \dots & a_{n-k+1}^* \\ a_1^* & a_2^* & \dots & a_{n-k}^* \end{bmatrix},$$

which is of the form (traditional) Toeplitz matrix times (traditional) Hankel matrix. (The second matrix seems to be like a traditional Toeplitz matrix also, but it is actually of the classical Hankel type since it maps a past input sequence to a past output sequence. We recover the classical Hankel type when we reverse the order of the rows. The difference between the two types is essential for the discussion here; see the explanation in Sec. 10.4.1 for added emphasis.) The rank of the Hankel matrix $H_k(F)$ (the term now refers to its more general meaning) is the essential parameter in the derivation of the state realization, and we obtain

$$\operatorname{rank}(H_k(F)) \leq \operatorname{rank} \begin{bmatrix} a_1^* & a_2^* & \dots & a_{n-1}^* \\ a_2^* & a_3^* & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_k^* & \ddots & \ddots & a_{n-k}^* \end{bmatrix},$$

which is a submatrix of the global (linear time invariant (LTI)) Hankel operator for the system

$$a_0^* + a_1^* z + a_2^* z^2 + a_3^* z^3 + \dots,$$

which is given by

$$\begin{bmatrix} a_1^* & a_2^* & \dots & \dots \\ a_2^* & a_3^* & \ddots & \dots \\ \vdots & \ddots & \ddots & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

A (standard) realization for that LTI system can be used as a starting point for the realization of F. Assuming that the dimension of the state space needed is δ, we find matrices α, β, γ of dimensions $\delta \times \delta, (p+q) \times \delta, \delta \times N$ such that, for $i \geq 1$,

$$a_i^* = \beta \alpha^{i-1} \gamma. \tag{10.5.2}$$

We choose the realization in output normal form, which means that the matrix $[\alpha\ \gamma]$ is isometric, i.e., $\alpha\alpha^* + \gamma\gamma^* = 1$. The realization $\{\alpha, \beta, \gamma\}$ may even be an adequate approximation to the system of a_k^*'s. Use your favorite approximation theory, based either on Hankel approximation theory in the style of [AAK71] or on balanced realizations [Kun78]. Notice that in the case of scalar inputs or outputs, the a_k^*'s just form a single z-dependent column. The relevant Hankel matrix for the series $\{a_i^*\}$ is now

$$\begin{bmatrix} a_k^* & \dots & a_{n-1}^* \\ \vdots & & \vdots \\ a_1^* & \dots & a_{n-k}^* \end{bmatrix} = \begin{bmatrix} \beta\alpha^{k-1} \\ \vdots \\ \beta\alpha \\ \beta \end{bmatrix} [\gamma\ \alpha\gamma\ \dots] \begin{bmatrix} I_{n-k} \\ 0 \end{bmatrix}, \tag{10.5.3}$$

in which the last matrix cuts out the first $n-k$ components from an otherwise infinite series.

The kth Hankel matrix for F is now

$$H_k(F) = 2\begin{bmatrix} a_0 J & & 0 \\ \vdots & \ddots & \\ a_{k-1}J & \cdots & a_0 J \end{bmatrix}\begin{bmatrix} \beta\alpha^{k-1} \\ \vdots \\ \beta\alpha \\ \beta \end{bmatrix}[\gamma\ \alpha\gamma\ \cdots]\begin{bmatrix} I_{n-k} \\ 0 \end{bmatrix}$$

and we find a realization for F with $A = \alpha$ and $C = \gamma$ already determined and, from the last row,

$$\begin{aligned} B_{k-1} &= 2a_{k-1}J\beta\alpha^{k-1} + \cdots + 2a_0 J\beta \\ &= 2\gamma^*[(\alpha^*)^{k-2}\beta^* J\beta\alpha^{k-1} + \cdots + \beta^* J\beta\alpha] + 2a_0 J\beta. \end{aligned} \tag{10.5.4}$$

Let us define

$$M_k \triangleq (\alpha^*)^{k-1}\beta^* J\beta\alpha^{k-1} + \cdots + \beta^* J\beta; \tag{10.5.5}$$

then M_k satisfies the recursive Lyapunov equation

$$M_k = \alpha^* M_{k-1}\alpha + \beta^* J\beta, \tag{10.5.6}$$

and B_k can easily be computed from M_k via

$$B_k = 2\gamma^* M_k\alpha + 2a_0 J\beta. \tag{10.5.7}$$

Similarly,

$$\begin{aligned} D_k &= a_k J a_k^* + \cdots + a_0 J a_0^* \\ &= a_k J a_k^* + D_{k-1} \quad \text{for } k \geq 1, \end{aligned} \tag{10.5.8}$$

and we have found a low rank recursive realization for F.

Algorithm 10.5.1 (Realization of F). *Given a symmetric positive-definite matrix R with displacement structure* (10.3.1)*, a state-space realization (A, B_k, C, D_k) for the upper triangular additive component F in* (10.5.1) *can be found as follows:*

1. $A = \alpha$, $C = \gamma$ *from the LTI system* (10.5.2).
2. $B_k = 2C^* M_k\alpha + 2a_0 J\beta$, *where* $M_1 = \beta^* J\beta$ *and* $M_k = \alpha^* M_{k-1}\alpha + \beta^* J\beta$ *when* $k \geq 1$.
3. $D_k = D_{k-1} + a_k J a_k^*$ *for* $k \geq 1$ *and* $D_0 = a_0 J a_0^*$.

◇

This realization is not necessarily (locally) minimal; it can be trimmed at the borders if needed. (The detailed procedure may be worthwhile but would lead us too far astray here—see [VD94].) A final observation is that the scheme gradually converges to a time-invariant realization when the operator R, now viewed as semi-infinite for positive indices, is bounded, since M_k, D_k, B_k converge as $k \to \infty$.

10.6 A REALIZATION FOR THE CHOLESKY FACTOR

We had before

$$R = \frac{1}{2}(F + F^*) = U^*U.$$

To find a realization for the Cholesky factor U, we keep $\{A, C\}$ and compute new b_k, d_k from the realization $\{A, B_k, C, D_k\}$ of F of the preceding section. First we show how this can be done in a general way; we deduce the algorithm and we show its correctness.

At this point it is advantageous to introduce block diagonal matrices to represent time-varying state-space representations. Let

$$\begin{aligned}
\hat{A} &\triangleq \operatorname{diag}\{\boxed{A_0}, A_1, A_2, \ldots\}, \\
\hat{B} &\triangleq \operatorname{diag}\{\boxed{B_0}, B_1, B_2, \ldots\}, \\
\hat{C} &\triangleq \operatorname{diag}\{\boxed{C_0}, C_1, C_2, \ldots\}, \\
\hat{D} &\triangleq \operatorname{diag}\{\boxed{D_0}, D_1, D_2, \ldots\}.
\end{aligned}$$

The first transition matrix A_0 is usually empty because there is no incoming state in the calculation—see Fig. 10.3. Let us assume that the sequence of state dimensions is $\{\Delta_1, \Delta_2, \ldots\}$ (we take $\Delta_0 = 0$!); then $\hat{A}$ consists of a sequence of diagonal blocks of dimensions $0 \times \Delta_1, \Delta_1 \times \Delta_2, \Delta_2 \times \Delta_3, \ldots$. It is a block diagonal matrix for which the first block is empty. (As indicated before, we write matrix-vector multiplication usually as row vector $\times$ matrix: $y = uA$.)

In addition to block diagonal matrices we introduce the (causal) shift matrix as

$$\mathcal{Z} \triangleq \begin{bmatrix} \ddots & \ddots & & & & \\ & 0 & I & & & \\ & & 0 & I & & \\ & & & 0 & I & \\ & & & & \ddots & \ddots \end{bmatrix}.$$

$\mathcal{Z}$ will shift rows if applied to the left of a matrix and will shift columns when applied to the right. It is also a block matrix where the first off diagonal is filled with unit matrices of possibly varying dimensions. $\mathcal{Z}$ actually stands for a collection of block matrices, because the dimensions of the various blocks may be different. For example, if $\mathcal{Z}$ is applied to the right of matrix $\hat{A}$ with block column dimensions $\Delta_1 + \Delta_2 + \Delta_3 + \cdots$, it will also have that sequence of (block) rows, while its sequence of (block) columns will be $\Delta_0 + \Delta_1 + \Delta_2 + \cdots$. It is understood that $\Delta_0 = 0$ in this case but that an empty column is there as a placeholder (and actually defines the location of the main diagonal). The underlying matrix when the block structure is stripped is just a unit matrix (which would not be the case if T had been doubly infinite to start with). We find in this way that the computing scheme "realizes" the operator

$$T = \hat{D} + \hat{B}\mathcal{Z}(I - \hat{A}\mathcal{Z})^{-1}\hat{C}. \tag{10.6.1}$$

We show in Fig. 10.5 a semigraphical illustration of how the block matrix mechanics of (10.6.1) work starting at $t = 0$. Because of the start-up condition, the state dimension δ_0 at $t = 0$ is zero. The various matrices of importance to the reasoning are also shown

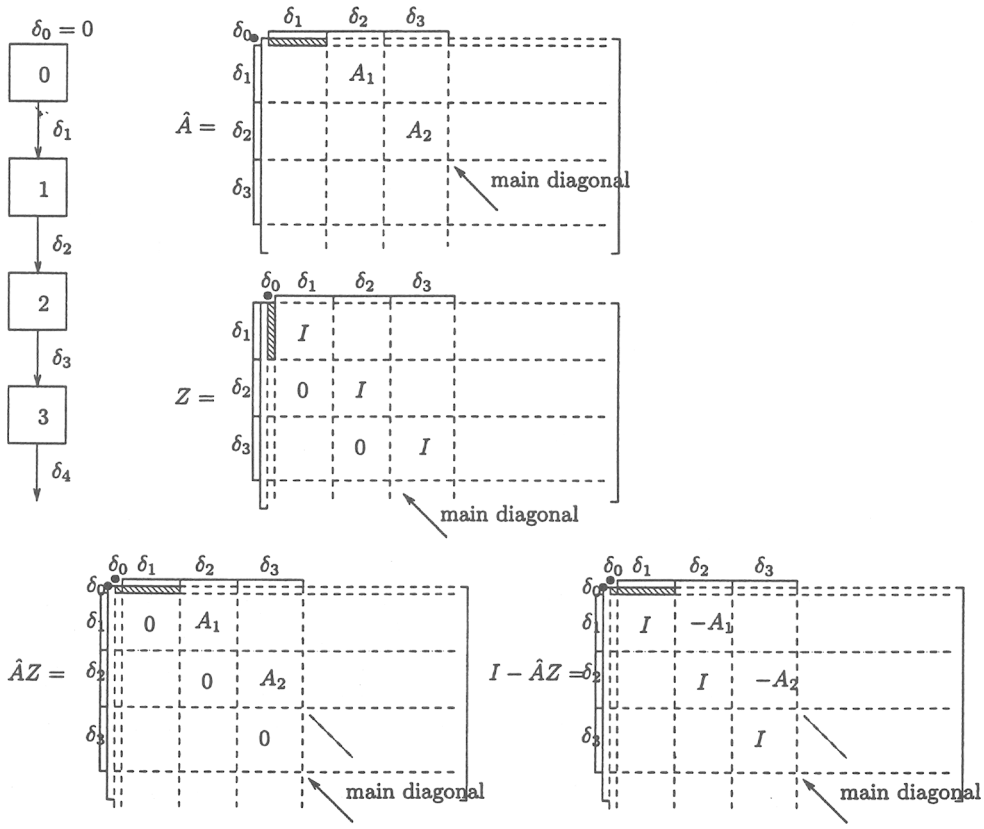

Figure 10.5. How the operator $I - \hat{A}Z$ originates.

in the figure. We remark further that

$$I - \hat{A}Z = \begin{bmatrix} I & -A_1 & & 0 \\ & I & -A_2 & \\ & & \ddots & \ddots \\ 0 & & & \ddots \end{bmatrix}.$$

Let us now try to find a realization for U with the same $\hat{A}$ and $\hat{C}$ as before but a new $\hat{b}$ and $\hat{d}$. Then we should have

$$\begin{aligned} U^*U = \hat{d}^*\hat{d} \;&+\; \hat{C}^*(I - Z^*\hat{A}^*)^{-1}Z^*\hat{b}^*\hat{d} + \hat{d}^*\hat{b}Z(I - \hat{A}Z)^{-1}\hat{C} \\ &+\; \hat{C}^*(I - Z^*\hat{A}^*)^{-1}Z^*\hat{b}^*\hat{b}Z(I - \hat{A}Z)^{-1}\hat{C}. \end{aligned}$$

The last term in this expression is quadratic. It can be subjected to a partial fraction expansion, which in this generalized context works as follows.

Let us define the diagonal shift on any block matrix $\hat{M}$ (a shift of one block down the main diagonals in the southeast direction):

$$\hat{M}^{(1)} \triangleq Z^*\hat{M}Z.$$

Then

$$\begin{aligned}(I - \mathcal{Z}^*\hat{A}^*)^{-1}(\hat{b}^*\hat{b})^{(1)}(I - \hat{A}\mathcal{Z})^{-1} &= (I - \mathcal{Z}^*\hat{A}^*)^{-1}\hat{m} + \hat{m}(I - \hat{A}\mathcal{Z})^{-1} - \hat{m} \\ &= \hat{m} + \hat{m}\hat{A}\mathcal{Z}(I - \hat{A}\mathcal{Z})^{-1} \\ &\quad + (I - \mathcal{Z}^*\hat{A}^*)^{-1}\mathcal{Z}^*\hat{A}^*\hat{m}\end{aligned}$$

if the equation for the block diagonal matrix $\hat{m}$,

$$\hat{m} = (\hat{b}^*\hat{b} + \hat{A}^*\hat{m}\hat{A})^{(1)}, \tag{10.6.2}$$

has a solution. Checking is immediate by pre- and postmultiplication of the expression with $(I - \mathcal{Z}^*\hat{A}^*)$ and $(I - \hat{A}\mathcal{Z})$, respectively.

This equation is known as a recursive Lyapunov–Stein equation, and in the present context it has a (unique) solution which can be computed recursively, provided that $b_k^*b_k$ is known at each step. Moreover, $\hat{m}$ is a matrix with square diagonal blocks of dimensions $\Delta_0 \times \Delta_0$, $\Delta_1 \times \Delta_1$, $\Delta_2 \times \Delta_2$, ..., in which the Δ_i are the block row dimensions of $\hat{A}$ and the first block $\Delta_0 \times \Delta_0$ is empty (row and column dimensions are zero). Let us write $\hat{m} = \operatorname{diag}(m_0 \dot{+} m_1 \dot{+} m_2 \cdots)$ in which m_0 is empty; then the recursion (10.6.3) says

$$\begin{aligned}(1)\quad m_1 &= b_0^*b_0, \\ (2)\quad m_2 &= b_1^*b_1 + A_1^*m_1A_1, \\ (3)\quad m_3 &= b_2^*b_2 + A_2^*m_2A_2, \\ \vdots &= \ddots\end{aligned}$$

When the partial fraction decomposition is now introduced in the equation for U^*U above, and we identify the strictly upper triangular, diagonal, and strictly lower triangular parts, then we see that the block diagonal matrices $\hat{b}$ and $\hat{d}$ must now satisfy the set of equations

$$\begin{cases} \frac{1}{2}(\hat{D} + \hat{D}^*) &= \hat{d}^*\hat{d} + \hat{C}^*\hat{m}\hat{C}, \\ \frac{1}{2}\hat{B} &= \hat{d}^*\hat{b} + \hat{C}^*\hat{m}\hat{A}, \\ \hat{m}^{(-1)} &= \hat{b}^*\hat{b} + \hat{A}^*\hat{m}\hat{A}. \end{cases} \tag{10.6.3}$$

This set of equations clearly leads to a recursive algorithm if consistent.

Algorithm 10.6.1 (Realization of the Cholesky Factor). *Given a symmetric positive-definite matrix R with displacement structure* (10.3.1), *a state-space realization (A, b_k, C, d_k) for the Cholesky factor U in* (10.5.1) *can be found as follows:*

1. *Keep the (A, C) from Alg.* 10.5.1 *and consider the (B_k, D_k) from the same algorithm.*
2. *Step* 0. *Set $d_0 = [\frac{1}{2}(D_0 + D_0^*)]^{1/2}$, $b_0 = \frac{1}{2}d_0^{-*}B_0$, and $m_1 = b_0^*b_0$.*
3. *Step* i. *Now assume m_i is known! Then set $d_i = [\frac{1}{2}(D_i + D_i^*) - C^*m_iC]^{1/2}$ (we shall show that the positive square root exists), $b_i = d_i^{-*}[\frac{1}{2}B_i - C^*m_iA]$, and $m_{i+1} = b_i^*b_i + A^*m_iA$.*

◇

One may think that a solution must exist, almost by construction (since the starting point of the recursion is well known—$m_0 = \emptyset$, the empty set) or by the theory of

spectral factorization for time-varying systems, but because the realization for F is not guaranteed minimal, there is reasonable doubt that at the kth step the equation for d_k,

$$d_k^* d_k = \frac{1}{2}(D_k + D_k^*) - C_k^* m_k C_k,$$

cannot be satisfied because the second member is possibly not positive definite. It is instructive to show that this cannot happen. In doing so we also give an alternative proof of the spectral factorization theorem for a time-varying R, and we show that it amounts to a Cholesky factorization. We construct the proof by looking at the Cholesky factorization of R in a Crout–Doolittle fashion—the classical method for solving a system of linear equations; see [Ste73]. The general Crout–Doolittle method consists of a recursive construction of a tableau for the lower or upper factorization of a general matrix $T = LU$ (in which we take U upper with unit diagonal entries and L lower). It has the form (taking u_{ij} and l_{ij} as the entries of U and L, respectively, and assuming no pivoting is necessary)

$$\begin{bmatrix} u_{00} & u_{01} & u_{02} & \cdots \\ (l_{10}) & u_{11} & u_{12} & \cdots \\ (l_{20}) & (l_{21}) & u_{22} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

It turns out that an entry, say, u_{ij}, of this tableau can be computed from the entries of the original matrix and the entries of the tableau with lower indices k, ℓ with either $k < i$ and $\ell \leq j$ or vice versa. In the next paragraph we give the details for our case.

The Cholesky modification of the Crout–Doolittle tableau for R looks as follows (in which the u_{ij} are the entries of U):

$$\begin{bmatrix} r_{00} & r_{01} & r_{02} & \cdots \\ r_{10} & r_{11} & r_{12} & \cdots \\ r_{20} & r_{21} & r_{22} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \Rightarrow \begin{bmatrix} u_{00} & u_{01} & u_{02} & \cdots \\ (u_{01}^*) & u_{11} & u_{12} & \cdots \\ (u_{02}^*) & (u_{12}^*) & u_{22} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{10.6.4}$$

The right-hand side is not really a useable matrix since its strictly lower part belongs to the matrix U^* and not to U, but it is customary and useful to include it in the tableau. Moreover, we remark that the entries can be matrices themselves. Filling the Cholesky tableau recursively then proceeds as follows:

- Step 0. $u_{00} = r_{00}^{1/2}$, $u_{0i} = u_{00}^{-*} r_{0i}$.

- Step i.

$$\begin{array}{lll} \text{[pivot]} & u_{ii}^* u_{ii} & = r_{ii} - \sum_{k=0}^{i-1} u_{ki}^* u_{ki}, \\ (j > i) & u_{ij} & = u_{ii}^{-*}[r_{ij} - \sum_{k=0}^{i-1} u_{ki}^* u_{kj}]. \end{array}$$

The central property in the algorithm is that the pivot is positive definite when R is, so that its square root can be taken in the case of scalar as well as matrix block entries. A proof for this fact is found by looking at a partial factorization of R up to the ith step and is of course classical.

In our case we have, thanks to the realization for F,

$$R = \begin{bmatrix} \frac{1}{2}[D_0 + D_0^*] & \frac{1}{2}B_0\gamma & \frac{1}{2}B_0 A\gamma & \frac{1}{2}B_0 A^2\gamma & \cdots \\ \frac{1}{2}\gamma^* B_0^* & \frac{1}{2}[D_1 + D_1^*] & \frac{1}{2}B_1\gamma & \frac{1}{2}B_1 A\gamma & \cdots \\ \frac{1}{2}\gamma^* A^* B_0^* & \frac{1}{2}\gamma^* B_1^* & \frac{1}{2}[D_2 + D_2^*] & \frac{1}{2}B_2\gamma & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

We now show that the recursion (10.6.2) in effect generates the (modified) Crout–Doolittle recursion.

- Step 0. $m_0 = \emptyset$, $\frac{1}{2}[D_0 + D_0^*] = d_0^* d_0$, $\frac{1}{2}B_0 = d_0^* b_0$ are of course solvable and produce the first row of U as

$$d_0 \quad b_0\gamma \quad b_0 A\gamma \quad b_0 A^2\gamma \quad \cdots.$$

- Step i. Let us assume that the first i rows (i.e., with indices $0, \ldots, i-1$) are computed. We have to show that $d_i^* d_i$ is well defined (i.e., the expression for it is positive definite) and also that the rest of the row with index i is correct. The Crout–Doolittle scheme applies and, thanks to the induction hypothesis, says that

$$\frac{1}{2}[D_i + D_i^*] - \sum_{k=0}^{i-1} \gamma^* [A^*]^{i-1-k} b_k^* b_k A^{i-k-1}\gamma$$

is positive definite. The recursion for $\hat{m}$ on the other hand gives an expression for the sum, so that the formula is in fact

$$\frac{1}{2}[D_i + D_i^*] - \gamma^* m_i \gamma,$$

which is hence positive definite and can be factored as $d_i^* d_i$. A further identification with the Crout–Doolittle scheme produces for $j > i$,

$$\begin{aligned} u_{ij} &= d_i^{-*}\{\tfrac{1}{2}B_i A^{j-i-1}\gamma - \textstyle\sum_{k=0}^{i-1}(b_k A^{i-1-k}\gamma)^*(b_k A^{j-1-k}\gamma)\} \\ &= d_i^{-*}[\tfrac{1}{2}B_i - \gamma^* m_i A]A^{j-i-1}\gamma, \end{aligned}$$

an equation that will be satisfied if (the more demanding) equation

$$\frac{1}{2}B_i = d_i^* b_i + \gamma^* m_i A$$

is. We may conclude that the scheme given by (10.6.2) always produces a positive-definite expression for

$$\frac{1}{2}[D_i + D_i^*] - \gamma^* m_i \gamma$$

when the original R is positive definite.

This concludes the proof of the existence of the realization for U as given by the algorithm, and we see that it is of the same complexity as the realization for F. In the following section we shall see that this realization also leads to an attractive computational scheme for U^{-1}, again of the same complexity.

10.7 DISCUSSION

The theory presented here is of course a specialization of a theory that can handle more general types of operators beyond matrices, e.g., general time-varying, time-discrete systems. For a reasonably complete account see [DV98]. The situation considered here is much simpler for two reasons: (1) matrices have a starting index, and equations become recursive with a well-defined initial condition; (2) there is an underlying time-invariant system that can be handled by classical techniques, especially concerning approximation. The more general case, however, does have some interest even here. Conceivably, the

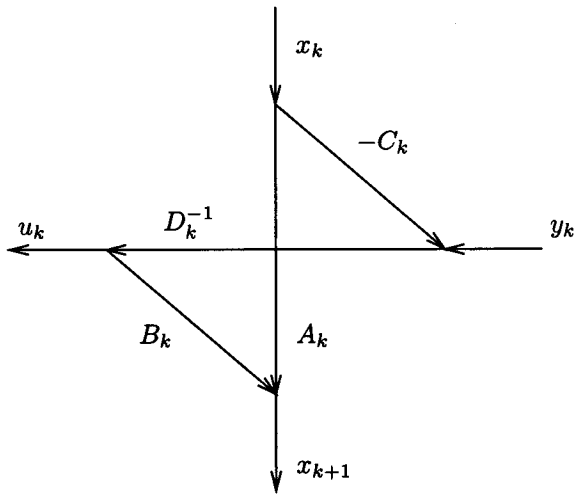

Figure 10.6. Simple local computation scheme for the inverse of a system.

matrices that we handle can be very large, but even more interesting, they may consist of subsequent "slices" of different low displacement rank systems. In that case we have a global time-varying but local low displacement behavior, but the approximation and complexity reduction theory will work roughly along the same lines as set out above, now with the use of the general theory.

A legitimate question, already announced in the introductory sections of this chapter, is, What do we gain in complexity reduction of the calculations if we apply the realization theory detailed in this chapter? A meaningful answer to such a question requires an understanding of what we mean by "the calculations." We consider two cases: calculations aiming at the construction of a model for the system or its inverse and calculations that aim at applying the model to an input. Both the low displacement rank and the low Hankel degree will contribute in both cases and in the expected manner. The low displacement rank allows for realizations of F and U in which only the matrices B_k and D_k vary from one point to the next using simple update equations, depending only on the actual a_k, which is itself dependent only on time-invariant data. If the Hankel rank is δ and the original system has scalar inputs and outputs, then the complexity of the realization $\{\alpha, \beta, \gamma\}$ is of the order of $(p+q)\delta$. Hence we end up with a parameter update scheme which can be very efficient depending on the precise values of the three parameters. The computational efficiency of vector-matrix products realized by a computational scheme as shown in Fig. 10.1 is directly proportional to the size of the state δ. As for the inverse (say, of U), we also face two types of computation: updates and the application of the computing scheme to inputs. Again, the update of the realization matrices for the inverse is a purely local matter dependent only on the actual a_k or their realization, via the formulas given by (10.4.9), but the computation can be restricted to the computation of D_k^{-1} since the other realization matrices can be used directly by graph inversion, as shown in Fig. 10.6. (More sophisticated schemes in which only δ scalar entries must be inverted exist; see [Vee93].)

Of course, the usage of a model for computation as shown in Fig. 10.1 precludes the utilization of the FFT as a complexity reducing engine. An FFT scheme, however, requires a complete shuffle of the data, either at the input side, or in the course of

the computations. It is (in some variations) the computational scheme that uses the smallest number of multiplications and additions possible but at the cost of maximal shuffling of data. It also does not utilize the fact that relevant impulse responses can have a lot of structure or can be approximated with very efficient data. In selective applications, accuracy will suffer. This is the reason why, in many signal processing applications, filtering algorithms are the preferred mode of implementation, although they coexist with the FFT. Even intermediate forms are possible, utilized in subband or multiresolution coding schemes, in which some shuffling of data takes place, combined with classical filtering. Therefore, a clear-cut statement concerning the advantage of one or the other is hard to make outside a specific application domain. This holds true even for the Toeplitz case. Here, Hankel realization theory reduces to the classical LTI realization theory, and the displacement rank may be just 1, so that vector-matrix multiplication reduces to a single FFT. The relative computational efficiency then pitches the system's degree δ against the logarithm of the time sequence, $\ln n$, which might appear to be to the advantage of the latter. But then, not all items in the complexity calculation have been included! For example, the "pipeline" (space) complexity of the FFT is again n against δ, which may be very disadvantageous in concrete cases. And if selective accuracy is included in the considerations, then the length of the FFT and the wordlength to be used may be impractical.