MODELING COMPUTATIONAL NETWORKS BY TIME-VARYING SYSTEMS

Alle-Jan van der Veen and Patrick Dewilde

Delft University of Technology, Dept. of Electrical Engineering
Mekelweg 4, 2628 CD Delft, The Netherlands

ABSTRACT

Many computational schemes in linear algebra can be studied from the point of view of (discrete) time-varying linear systems theory. For example, the operation 'multiplication of a vector by an upper triangular matrix' can be represented by a computational scheme (or model) that acts sequentially on the entries of the vector. The number of intermediate quantities ('states') that are needed in the computations is a measure of the complexity of the model. If the matrix is large but its complexity is low, then not only multiplication, but also other operations such as inversion and factorization, can be carried out efficiently using the model rather than the original matrix. In the present paper we discuss a number of techniques in time-varying system theory that can be used to capture a given matrix into such a computational network.

Keywords: computational linear algebra models, model reduction, fast matrix algorithms.

1. INTRODUCTION

1.1. Computational linear algebra and time-varying modeling

In the intersection of linear algebra and system theory is the field of *computational linear algebra*. Its purpose is to find efficient algorithms for linear algebra problems (matrix multiplication, inversion, approximation). A useful model for matrix computations is provided by the state equations that are used in dynamical system theory. Such a state model is often quite natural: in any algorithm which computes a matrix multiplication or inversion, the global operation is decomposed into a sequence of local operations that each act on a limited number of matrix entries (ultimately two), assisted by intermediate quantities that connect the local operations. These quantities can be called the states of the algorithm, and translate to the state of the dynamical system that is the computational model of the matrix operation. Although many matrix operations can be captured this way by some linear dynamical system, our interest is in matrices that possess some kind of structure which allows for efficient ("fast") algorithms: algorithms that exploit this structure. Structure in a matrix is inherited from the origin of the linear algebra problem, and is for our purposes typically due to the modeling of some (physical) dynamical system. Many signal processing applications, inverse scattering problems and least squares estimation problems give structured matrices that can indeed be modeled by a low complexity network.

Besides sparse matrices (many zero entries), traditional structured matrices are Toeplitz and Hankel matrices, which translate to linear time-invariant (LTI) systems. Associated computational algorithms are well-known, *e.g.,* for Toeplitz systems we have Schur recursions for LU- and Cholesky factorization [1], Levinson recursions for factorization of the inverse [2], Gohberg/Semencul recursions for computing the inverse [3], and Schur-based recursions for QR factorization [4]. The resulting algorithms have computing complexity of order $\mathcal{O}(n^2)$ for matrices of size $(n\times n)$, as compared to $\mathcal{O}(n^3)$ for algorithms that do not take the Toeplitz

[0]Integration, the VLSI Journal, 16(3):267-291, December 1993.

structure into account. Generalizations of the Toeplitz structure are obtained by considering matrices which have a so-called displacement structure [5, 6, 7, 8]: matrices G for which there are (simple) matrices F_1, F_2 such that

$$G - F_1^* G F_2 \tag{1}$$

is of low rank, α say. Such a matrix occurs *e.g.*, in stochastic adaptive prediction problems as the covariance matrix of the received stochastic signal; the matrix is called α-stationary. An overview of inversion and factorization algorithms for such matrices can be found in [9].

In this paper, we pursue a complementary notion of structure which we will call a state structure. The state structure applies to upper triangular matrices and is seemingly unrelated to the Toeplitz structure mentioned above. A first purpose of the computational schemes considered in this paper is to perform a desired linear transformation T on a vector ('input sequence') u,

$$u = [u_1 \quad u_2 \quad \cdots \quad u_n]$$

with an output vector or sequence $y = uT$ as the result. The key idea is that we can associate with this matrix-vector multiplication a computational network that takes u and computes y, and that matrices with a sparse state structure have a computational network of low complexity so that using the network to compute y is more efficient than computing uT directly. To introduce this notion, consider an upper triangular matrix T along with its inverse,

$$T = \begin{bmatrix} 1 & 1/2 & 1/6 & 1/24 \\ & 1 & 1/3 & 1/12 \\ & & 1 & 1/4 \\ & & & 1 \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} 1 & -1/2 & & \\ & 1 & -1/3 & \\ & & 1 & -1/4 \\ & & & 1 \end{bmatrix}.$$

The inverse of T is sparse, which is an indication of a sparse state structure. A computational network that models multiplication by T is depicted in figure 1(a), and it is readily verified that this network does indeed compute $[y_1 \;\; y_2 \;\; y_3 \;\; y_4] = [u_1 \;\; u_2 \;\; u_3 \;\; u_4]T$ by trying vectors of the form $[1 \;\; 0 \;\; 0 \;\; 0]$ up to $[0 \;\; 0 \;\; 0 \;\; 1]$. The dependence of y_k on u_i, $(i < k)$, introduces intermediate quantities x_k called *states*. At each point k the processor in the stage at that point takes its input data u_k from the input sequence u and computes a new output data y_k which is part of the output sequence y generated by the system. The computations in the network are split into sections, which we will call *stages*, where the k-th stage consumes u_k and produces y_k. To execute the computation, the processor will use some remainder of its past history, *i.e.*, the state x_k, which has been computed by the previous stages and which was temporarily stored in registers indicated by the symbol z. The complexity of the computational network is equal to the number of states at each point. A non-trivial computational network to compute $y = uT$ which requires less states is shown in figure 1(b). The total number of multiplications required in this network that are different from 1 is 5, as compared to 6 in a direct computation using T. Although we have gained only one multiplication here, for a less moderate example, say a $(n \times n)$ upper triangular matrix with $n = 10000$ and $d \ll n$ states at each point, the number of multiplications in the network can be as low as $\mathcal{O}(8dn)$, instead of $\mathcal{O}(1/2\, n^2)$ for a direct computation using T. Note that the number of states can vary from one point to the other, depending on the nature of T. In the example above, the number of states entering the network at point 1 is zero, and the number of states leaving the network at point 4 is also zero. If we would change the value of one of the entries of the 2×2 submatrix in the upper-right corner of T to a different value, then two states would have been required to connect stage 2 to stage 3.

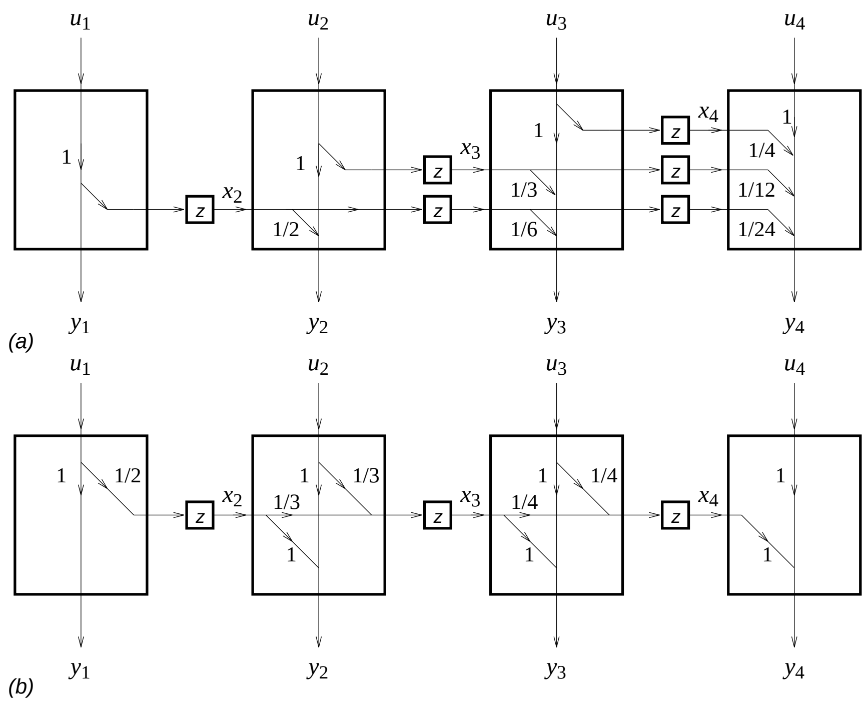

Figure 1. Computational networks corresponding to T. (*a*) Direct (trivial) realization, (*b*) minimal realization.

The computations in the network can be summarized by the following recursion, for $k = 1$ to n:

$$y = uT \qquad \Leftrightarrow \qquad \begin{array}{rcl} x_{k+1} & = & x_k A_k + u_k B_k \\ y_k & = & x_k C_k + u_k D_k \end{array} \tag{2}$$

or

$$[x_{k+1} \quad y_k] = [x_k \quad u_k]\,\mathbf{T}_k\,, \qquad \mathbf{T}_k = \begin{bmatrix} A_k & C_k \\ B_k & D_k \end{bmatrix}$$

in which x_k is the state vector at time k (taken to have d_k entries), A_k is a $d_k \times d_{k+1}$ (possibly non-square) matrix, B_k is a $1 \times d_{k+1}$ vector, C_k is a $d_k \times 1$ vector, and D_k is a scalar. More general computational networks can have the number of inputs and outputs at each stage different from one, and possibly also varying from stage to stage. In the example, we have a sequence of realization matrices

$$\mathbf{T}_1 = \begin{bmatrix} \cdot & \cdot \\ 1/2 & 1 \end{bmatrix} \qquad \mathbf{T}_2 = \begin{bmatrix} 1/3 & 1 \\ 1/3 & 1 \end{bmatrix} \qquad \mathbf{T}_3 = \begin{bmatrix} 1/4 & 1 \\ 1/4 & 1 \end{bmatrix} \qquad \mathbf{T}_4 = \begin{bmatrix} \cdot & 1 \\ \cdot & 1 \end{bmatrix}$$

where the '$\cdot$' indicates entries that actually have dimension 0 because the corresponding states do not exist. The recursion in equation (2) shows that it is a recursion for increasing values of k: the order of computations in the network is strictly from left to right, and we cannot compute y_k unless we know x_k, *i.e.*, unless we have processed $u_1 \cdots u_{k-1}$. On the other hand, y_k does not depend on $u_{k+1} \cdots u_n$. This is a direct consequence of the fact that T has been chosen upper triangular, so that such an ordering of computations is indeed possible.

1.2. Time-varying systems

A link with system theory is obtained when T is regarded as the transfer matrix of a *non-stationary* causal linear system with input u and output $y = uT$. The k-th row of T then corresponds to the impulse response of the system when excited by an impulse at time instant i, that is, the output y due to an input vector $u =$

$[0 \cdots 0\ 1\ 0 \cdots 0]$, where $u_i = 1$. The case where T has a Toeplitz structure then corresponds with a time-invariant system for which the impulse response due to an impulse at time $i+1$ is just the same as the response due to an impulse at time i, shifted over one position. The computational network is called a state realization of T, and the number of states at each point of the computational network is called the system order of the realization at that point in time. For time-invariant systems, the state realization can be chosen constant in time. Since for time-varying systems the number of state variables need not be constant in time, but can increase and shrink, it is seen that in this respect the time-varying realization theory is much richer, and that the accuracy of an approximating computational network of T can be varied in time at will.

1.3. Sparse computational models

If the number of state variables is relatively small, then the computation of the output sequence is efficient in comparison with a straight computation of $y = uT$. One example of a matrix with a small state space is the case where T is an upper triangular band-matrix: $T_{ij} = 0$ for $j - i > p$. In this case, the state dimension is equal to or smaller than $p-1$, since only $p-1$ of the previous input values need to be remembered at any point in the multiplication. However, the state space model can be much more general, *e.g.*, if a banded matrix has an inverse, then this inverse is known to have a sparse state space (of the same complexity) too, as we had in the example above. Moreover, this inversion can be easily carried out by local computations on the realization of T: if $T^{-1} = S$, then $u = yS$ can be computed via

$$\left\{ \begin{array}{lcl} x_{k+1} & = & x_k A_k + u_k B_k \\ y_k & = & x_k C_k + u_k D_k \end{array} \right. \quad \Leftrightarrow \quad \left\{ \begin{array}{lcl} x_{k+1} & = & x_k (A_k - C_k D_k^{-1} B_k) + y_k D_k^{-1} B_k \\ u_k & = & -x_k C_k D_k^{-1} + y_k D_k^{-1} \end{array} \right.$$

hence S has a computational model given by

$$\mathbf{S}_k = \begin{bmatrix} A_k - C_k D_k^{-1} B_k & -C_k D_k^{-1} \\ D_k^{-1} B_k & D_k^{-1} \end{bmatrix} \qquad (3)$$

Observe that the model for $S = T^{-1}$ is obtained in a *local* way from the model of T: $\mathbf{S}_k$ depends only on $\mathbf{T}_k$. The sum and product of matrices with sparse state structure have again a sparse state structure with number of states at each point not larger than the sum of the number of states of its component systems, and computational networks of these compositions (but not necessarily minimal ones) can be easily derived from those of its components.

At this point, one might wonder for which class of matrices T there exists a sparse computational network (or state space realization) that realizes the same multiplication operator. For an upper triangular $(n \times n)$ matrix T, define matrices H_i $(1 \le i \le n)$, which are submatrices of T, as

$$H_i = \begin{bmatrix} T_{i-1,i} & T_{i-1,i+1} & \cdots & T_{i-1,n} \\ T_{i-2,i} & T_{i-2,i+1} & & \vdots \\ \vdots & & \ddots & T_{2,n} \\ T_{1,i} & \cdots & T_{1,n-1} & T_{1,n} \end{bmatrix}$$

(see figure 2). We call the H_i (time-varying) Hankel matrices, as they will have a Hankel structure (constant along anti-diagonals) if T has a Toeplitz structure.[1] In terms of the Hankel matrices, the criterion by which

[1] Warning: in the current context (arbitrary upper triangular matrices) the H_i do not have a Hankel structure and the predicate ‘Hankel matrix’ could lead to misinterpretations. Our terminology finds its motivation in system theory, where the H_i are related to an abstract operator H_T which is commonly called the Hankel operator. For time-invariant systems, H_T reduces to an operator with a matrix representation that has indeed a Hankel structure.

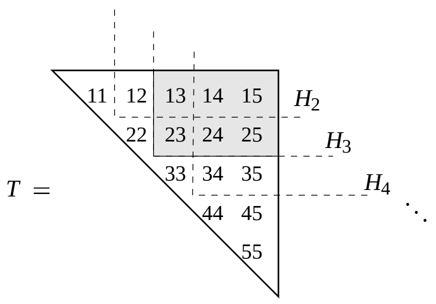

Figure 2. Hankel matrices are (mirrored) submatrices of T.

matrices with a sparse state structure can be detected is given by the following Kronecker or Ho-Kalman [10] type theorem (proven in section 3).

Theorem 1. *The number of states that are needed at stage k in a minimal computational network of an upper triangular matrix T is equal to the rank of its k-th Hankel matrix* H_k.

Let's verify this statement for our example. The Hankel matrices are

$$H_1 = [\cdot\ \cdot\ \cdot]\,, \qquad H_2 = [1/2\ \ 1/6\ \ 1/24]\,, \qquad H_3 = \begin{bmatrix} 1/3 & 1/12 \\ 1/6 & 1/24 \end{bmatrix}, \qquad H_4 = \begin{bmatrix} 1/4 \\ 1/12 \\ 1/24 \end{bmatrix}.$$

Since $\text{rank}(H_1) = 0$, no states x_1 are needed. One state is needed for x_2 and one for x_4, because $\text{rank}(H_2) = \text{rank}(H_4) = 1$. Finally, also only one state is needed for x_3, because $\text{rank}(H_3) = 1$. In fact, this is (for this example) the only non-trivial rank condition: if one of the entries in H_3 would have been different, then two states would have been needed. In general, $\text{rank}(H_i) \le \min(i-1, n-i-1)$, and for a general upper triangular matrix T without state structure, a computational model will indeed require at most $\min(i-1, n-i-1)$ states for x_i.

1.4. Example: Cholesky Factorization

Computational advantages of the time-varying system theory are not *per se* restricted to upper triangular operators. To illustrate this, we introduce the following example. Consider a (strictly) positive definite matrix G, and suppose G has been normalized such that its main diagonal is the identity matrix. It is desired to obtain a Cholesky factorization of G: a factorization $G = L^*L$, where L is an upper triangular matrix. For Toeplitz matrices, this can be done using Schur recursions [1, 11]. The Schur algorithm can be generalized in various ways to apply to triangular factorizations of general matrices [12], matrices with a displacement structure [5, 6, 7, 8], and approximate factorizations on a staircase band [13]. In the context of this paper, it can be shown that if the upper triangular part of G has a state structure, then a computational model of this part can be used to determine a computational model of the factor L.

A transition to upper matrices is obtained by an analog of the Cayley transformation, which is used to map positive functions to contractive (scattering) functions. Define $\mathbf{P}(G)$ to be the upper triangular part of G, and $G_1 = 2\mathbf{P}(G) - I$, then $S = (G_1 + I)^{-1}(G_1 - I)$ is a well-defined and contractive upper triangular matrix: $\|S\| < 1$. It has a direct relation with G:

$$\mathbf{P}(G) = (I - S)^{-1}; \qquad S = I - [\mathbf{P}(G)]^{-1}\,.$$

This shows that the state structure of G carries over to S: if $\mathbf{P}(G)$ has a computational model with d_k states at point k, then $[\mathbf{P}(G)]^{-1}$ and hence also S have computational models with at each point the same number of states, and can be directly derived using equation (3).

A computational model for $\mathbf{P}(G)$ is obtained using the realization algorithm of section 3. Thus let $\{A,B,C,D\}$ be a realization of $\mathbf{P}(G)$ (satisfying certain conditions which we omit at this point). Via the model of S, it is possible to derive a state model of the factor L satisfying $G = L^*L$. It turns out that the model of L is given by

$$\mathbf{L}_k = \begin{bmatrix} I & \\ & (D - C_k^* M_k C_k)^{1/2} \end{bmatrix} \begin{bmatrix} A_k & C_k \\ B_k' & I \end{bmatrix}$$

where $B_k' = -(D - C_k^* M_k C_k)^{-1}(B_k - C_k^* M_k A_k)$, and M_k is given by the recursion

$$\begin{array}{rcl} M_1 & = & [\cdot] \\ M_{k+1} & = & A_k^* M_k A_k + (B_k - C_k^* M_k A_k)^* (D - C_k^* M_k C_k)^{-1} (B_k - C_k^* M_k A_k)\,. \end{array}$$

2. OBJECTIVES OF COMPUTATIONAL MODELING

With the preceding section as background material, we are now in a position to identify the objectives of our computational modeling. We will assume throughout that we are dealing with upper triangular matrices. However, applications which involve other type of matrices are viable if they provide some transformation to the class of upper triangular matrices In addition, we assume that the concept of a sparse state structure is *meaningful* for the problem, in other words that a typical matrix in the application has a sequence of Hankel matrices that has low rank (relative to the size of the matrix), or that an approximation of that matrix by one whose Hankel matrices have low rank would indeed yield a useful approximation of the underlying (physical) problem that is described by the original matrix.

For such a matrix T, the generic objective is to determine a minimal computational model $\{\mathbf{T}_k\}$ for it by which multiplications of vectors by T are effectively carried out, but in a *computationally efficient* and *numerically stable* manner. This objective is divided into four subproblems: (1) realization of a given matrix T by a computational model, (2) embedding of this realization in a larger model that consists entirely of unitary (lossless) stages, (3) factorization of the stages of the embedding into a cascade of elementary (degree-1) lossless sections. It could very well be that the originally given matrix has a computational model of a very high order. Then intermediate in the above sequence of steps is (4) approximation of a given realization of T by one of lower complexity. These steps are motivated below.

Realization

The first step is, given T, to determine any minimal computational network $\mathbf{T}_k = \{A_k, B_k, C_k, D_k\}$ that models T. This problem is known as the *realization problem*. If the Hankel matrices of T have low rank, then $\mathbf{T}$ is a computationally efficient realization of the operation 'multiplication by T'.

Lossless embedding

From $\mathbf{T}$, all other minimal realizations of T can be derived by state transformations. Not all of these have the same numerical stability. This is because the computational network has introduced a recursive aspect to the multiplication: states are used to extract information from the input vector u, and a single state x_k gives a contribution both to the current output y_k and to the sequence x_{k+1}, x_{k+2} etc. In particular, a perturbation in x_k (or u_k) also carries over to this sequence. Suppose that T is bounded in norm by some number, say $\| T \| \leq 1$,[2] so that we can measure perturbation errors relative to 1. Then a realization of T is said to be error insensitive if $\| \mathbf{T}_k \| \leq 1$, too. In that case, an error in $[x_k \;\; u_k]$ is not magnified by $\mathbf{T}_k$, and the resulting error in $[x_{k+1} \;\; y_k]$ is smaller than the original perturbation. Hence the question is: is it possible to obtain a realization for which $\| \mathbf{T}_k \| \leq 1$ if T is such that $\| T \| \leq 1$? The answer is yes, and an algorithm to obtain such a realization is given by the solution of the *lossless embedding problem*. This problem is the following: for a given matrix T with $\| T \| \leq 1$, determine a computational model $\{\mathbf{\Sigma}_k\}$ such that (1) each $\mathbf{\Sigma}_k$ is a unitary matrix, and (2) T is a subsystem of the transfer matrix Σ that corresponds to $\{\mathbf{\Sigma}_k\}$. The latter requirement means that T is the transfer matrix from a subset of the inputs of Σ to a subset of its outputs: Σ can be partitioned conformably as

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}, \qquad T = \Sigma_{11}\,.$$

The fact that T is a subsystem of Σ implies that a certain submatrix of $\mathbf{\Sigma}_k$ is a realization $\mathbf{T}_k$ of T, and hence from the unitarity of $\mathbf{\Sigma}_k$ we have that $\|\mathbf{T}_k\| \leq 1$. From the construction of the solution to the embedding problem, it will follow that we can ensure that this realization is minimal, too.

[2] $\| T \|$ is the operator norm (matrix 2-norm) of T: $\| T \| = \sup_{\|u\|_2 \leq 1} \|uT\|_2$.

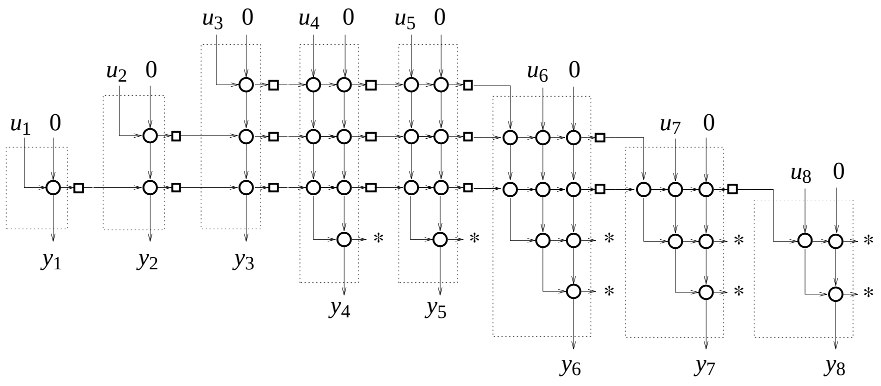

Figure 3. Cascade realization of a contractive 8×8 matrix T, with a maximum of 3 states at each point.

Cascade factorization

Assuming that we have obtained such a realization $\mathbf{\Sigma}_k$, it is possible to break down the operation 'multiplication by $\mathbf{\Sigma}_k$' on vectors $[x_k \;\; u_k]$ into a minimal number of elementary operations, each in turn acting on two entries of this vector. Because $\mathbf{\Sigma}_k$ is unitary, we can use elementary unitary operations (acting on scalars) of the form

$$[a_1 \;\; b_1] \begin{bmatrix} c & s \\ -s^* & c^* \end{bmatrix} = [a_2 \;\; b_2] \,, \qquad cc^* + ss^* = 1 \,,$$

i.e., elementary rotations. The use of such elementary operations will ensure that $\mathbf{\Sigma}_k$ is internally numerically stable, too. In order to make the number of elementary rotations minimal, the realization $\mathbf{\Sigma}$ is transformed to an equivalent realization $\mathbf{\Sigma}'$, which realizes the same system Σ, is still unitary and which still contains a realization $\mathbf{T}'$ for T. A factorization of each $\mathbf{\Sigma}'_k$ into elementary rotations is known as a *cascade realization* of Σ. A possible minimal computational model for T that corresponds to such a cascade realization is drawn in figure 3. In this figure, each circle indicates an elementary rotation. The precise form of the realization depends on whether the state dimension is constant, shrinks or grows. The realization can be divided horizontally into elementary *sections*, where each section describes how a single state entry is mapped to an entry of the 'next state' vector x_{k+1}. It has a number of interesting properties; one is that it is *pipelineable*, which is interesting if the operation 'multiplication by T' is to be carried out on a collection of vectors u on a parallel implementation of the computational network. The property is a consequence of the fact that the signal flow in the network is strictly uni-directional: from top-left to bottom-right, so that computations on a new vector u (a new u_k and a new x_k) can commence in the top-left part of the network, while computations on the previous u are still being carried out in the bottom-right part.

Approximation

In the previous items, we have assumed that the matrix T has indeed a computational model of an order that is low enough to favor a computational network over an ordinary matrix multiplication. However, if the rank of the Hankel matrices of T (the system order) is not low, then it makes often sense to *approximate* T by a new upper triangular matrix T_a that has a lower complexity. While it is fairly known in linear algebra how to obtain a (low-rank) approximant to a matrix in a certain norm (*e.g.,* by use of the singular value decomposition (SVD)), such approximations are not necessarily appropriate for our purposes, because the approximant should be upper triangular again, and have a lower system order. Because the system order at each point is given by the rank of the Hankel matrix at that point, a possible approximation scheme is to approximate each Hankel operator by one that is of lower rank (this could be done using the SVD). However, because the

Hankel matrices have many entries in common, it is not clear at once that such an approximation scheme is feasible: replacing one Hankel matrix by one of lower rank in a certain norm might make it impossible for the next Hankel matrix to find an optimal approximant. The severity of this dilemma is mitigated by a proper choice of the error criterion. In fact, it is remarkable that this dilemma has a nice solution, and that this solution can be obtained in a non-iterative manner. The error criterion for which a solution is obtained is called the Hankel norm and denoted by $\|\cdot\|_H$: it is the maximum over the operator norm (the matrix 2-norm) of each individual Hankel matrix approximation, and a generalization of the Hankel norm for time-invariant systems. In terms of the Hankel norm, the following theorem holds true and generalizes the model reduction techniques based on the Adamyan-Arov-Krein paper [14] to time-varying systems:

Theorem 2. **([15])** *Let T be a strictly upper triangular matrix and let $\Gamma = \mathrm{diag}(\gamma_i)$ be a diagonal Hermitian matrix which parametrizes the acceptable approximation tolerance ($\gamma_i > 0$). Let H_k be the Hankel matrix of $\Gamma^{-1}T$ at stage k, and suppose that, for each k, none of the singular values of H_k are equal to 1. Then there exists a strictly upper triangular matrix T_a with system order at stage k equal to the number of singular values of H_k that are larger than 1, such that*

$$\|\Gamma^{-1}(T - T_a)\|_H \leq 1\,.$$

In fact, there is an algorithm that determines a model for T_a directly from a model of T. Γ can be used to influence the local approximation error. For a uniform approximation, $\Gamma = \gamma I$, and hence $\|T - T_a\|_H \leq \gamma$: the approximant is γ-close to T in Hankel norm, which implies in particular that the approximation error in each row or column of T is less than γ. If one of the γ_i is made larger than γ, then the error at the i-th row of T can become larger also, which might result in an approximant T_a to take on less states. Hence Γ can be chosen to yield an approximant that is accurate at certain points but less tight at others, and whose complexity is minimal.

As a numerical example of the use of theorem 2, let the matrix to be approximated be

$$T = \left[\begin{array}{rrr|rrr} 0 & .800 & .200 & .050 & .013 & .003 \\ 0 & 0 & .600 & .240 & .096 & .038 \\ 0 & 0 & 0 & .500 & .250 & .125 \\ \cline{4-6} 0 & 0 & 0 & 0 & .400 & .240 \\ 0 & 0 & 0 & 0 & 0 & .300 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

We have indicated the position of the Hankel matrix H_4. Taking $\Gamma = 0.1I$, the non-zero singular values of the Hankel operators of $\Gamma^{-1}T$ are

	H_1	H_2	H_3	H_4	H_5	H_6
σ_1 :		8.26	6.85	6.31	5.53	4.06
σ_2 :			0.33	0.29	0.23	
σ_3 :				0.01		

Hence T has a state space realization which grows from zero states ($i = 1$) to a maximum of 3 states ($i = 4$), and then shrinks back to 0 states ($i > 6$). The number of Hankel singular values of T that are larger than one is 1 ($i = 2 \cdots 6$). This is to correspond to the number of states of the approximant at each point. Using the

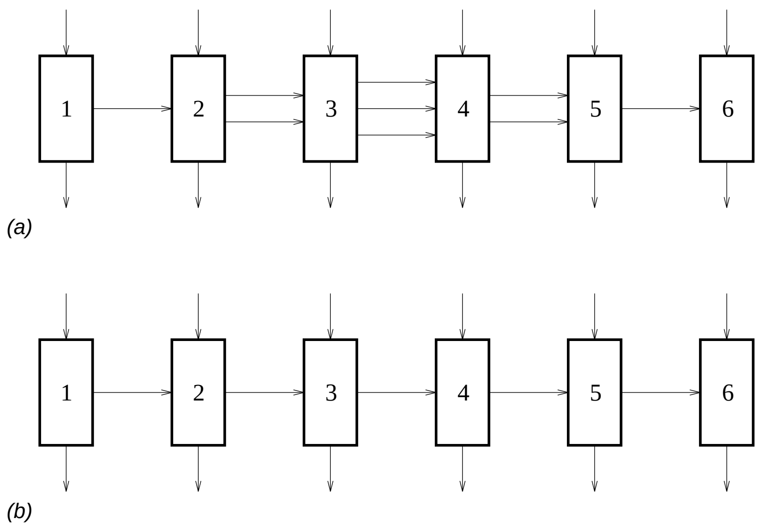

Figure 4. Computational scheme (a) of T and (b) of T_a.

techniques in [15], the approximant can be obtained as

$$T_a = \begin{bmatrix} 0 & .790 & .183 & .066 & .030 & .016 \\ 0 & 0 & .594 & .215 & .098 & .052 \\ 0 & 0 & 0 & .499 & .227 & .121 \\ 0 & 0 & 0 & 0 & .402 & .214 \\ 0 & 0 & 0 & 0 & 0 & .287 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

with non-zero Hankel singular values (scaled by Γ)

	H_1	H_2	H_3	H_4	H_5	H_6
σ_1 :		8.15	6.71	6.16	5.36	3.82

whose number indeed correspond to the number of Hankel singular values of $\Gamma^{-1}T$ that are larger than 1. Also, the modeling error is

$$\|\Gamma^{-1}(T-T_a)\|_H = \sup\{0.334,\, 0.328,\, 0.338,\, 0.351,\, 0.347\} = 0.351$$

which is indeed smaller than 1. The corresponding computational schemes of T and T_a are depicted in figure 4.

In the remainder of the paper, we will discuss an outline of the algorithms that are involved in the first three of the above items. A full treatment of item 1 was published in [16], item 2 in [17], and item 3 was part of the subject of [18]. Theory on Hankel norm approximations is available [15, 19] but is omitted here for lack of space.

3. REALIZATION OF A TIME-VARYING SYSTEM

The purpose of this section is to give a proof of the realization theorem for time-varying systems (specialized to finite matrices): theorem 1 of section 1.3. A more general and detailed discussion can be found in [16]. Recall that we are given an upper triangular matrix T, and view it as a time-varying system transfer operator. The objective is to determine a time-varying state realization for it. The approach is as in Ho-Kalman's theory for the time-invariant case [10]. Denote a certain time instant as 'current time', apply all possible inputs in the 'past' with respect to this instant, and measure the corresponding outputs in 'the future', from the current time instant on. For each time instant, we select in this way an upper-right part of T: these are its Hankel matrices as defined in the introduction. Theorem 1 claimed that the rank of H_k is equal to the order of a minimal realization at point k.

PROOF of theorem 1. The complexity criterion can be derived straightforwardly, and the derivation will give rise to a realization algorithm as well. Suppose that $\{A_k, B_k, C_k, D_k\}$ is a realization for T as in equation (2). Then a typical Hankel matrix has the following structure:

$$\begin{aligned} H_6 \;=\; \begin{bmatrix} B_5C_6 & B_5A_6C_7 & B_5A_6A_7C_8 & \cdots \\ B_4A_5C_6 & B_4A_5A_6C_7 & & \\ B_3A_4A_5C_6 & & \ddots & \\ \vdots & & & \\ B_1A_2\cdots A_5C_6 & & & \end{bmatrix} &= \begin{bmatrix} B_5 \\ B_4A_5 \\ B_3A_4A_5 \\ \vdots \\ B_1A_2\cdots A_5 \end{bmatrix} \cdot [C_6 \;\; A_6C_7 \;\; A_6A_7C_8 \;\; \cdots] \\ &= \mathcal{C}_6\mathcal{O}_6 \end{aligned} \tag{4}$$

From the decomposition $H_k = \mathcal{C}_k\mathcal{O}_k$ it is directly inferred that if A_k is of size $(d_k \times d_{k+1})$, then $\operatorname{rank}(H_k)$ is at most equal to d_k. We have to show that there exists a realization $\{A_k, B_k, C_k, D_k\}$ for which $d_k = \operatorname{rank}(H_k)$: if it does, then clearly this must be a minimal realization. To find such a minimal realization, take any minimal factorization $H_k = \mathcal{C}_k\mathcal{O}_k$ into full rank factors $\mathcal{C}_k$ and $\mathcal{O}_k$. We must show that there are matrices $\{A_k, B_k, C_k, D_k\}$ such that

$$\mathcal{C}_k = \begin{bmatrix} B_{k-1} \\ B_{k-2}A_{k-1} \\ \vdots \end{bmatrix} \qquad\qquad \mathcal{O}_k = [C_k \;\; A_kC_{k+1} \;\; A_kA_{k+1}C_{k+2} \;\; \cdots]\,.$$

To this end, we use the fact that H_k satisfies a shift-invariance property: for example, with $H_6^{\leftarrow}$ denoting H_6 without its first column, we have

$$H_6^{\leftarrow} = \begin{bmatrix} B_5 \\ B_4A_5 \\ B_3A_4A_5 \\ \vdots \\ B_1A_2\cdots A_5C_6 \end{bmatrix} \cdot A_6 \cdot [C_7 \;\; A_7C_8 \;\; A_7A_8C_9 \;\; \cdots]\,.$$

In general, $H_k^{\leftarrow} = \mathcal{C}_kA_k\mathcal{O}_{k+1}$, and in much the same way, $H_k^{\uparrow} = \mathcal{C}_{k-1}A_{k-1}\mathcal{O}_k$, where $H_k^{\uparrow}$ is H_k without its first row. The shift-invariance properties carry over to $\mathcal{C}_k$ and $\mathcal{O}_k$, *e.g.*, $\mathcal{O}_k^{\leftarrow} = A_k\mathcal{O}_{k+1}$, and we obtain that $A_k = \mathcal{O}_k^{\leftarrow}\mathcal{O}_{k+1}^*(\mathcal{O}_{k+1}\mathcal{O}_{k+1}^*)^{-1}$, where '*' denotes complex conjugate transposition. The inverse exists because $\mathcal{O}_{k+1}$ is of full rank. C_k follows as the first column of the chosen $\mathcal{O}_k$, while B_k is the first row of $\mathcal{C}_{k+1}$. It remains to verify that $\mathcal{C}_k$ and $\mathcal{O}_k$ are indeed generated by this realization. This is straightforward by a recursive use of the shift-invariance properties. □

The construction in the above proof leads to a realization algorithm (algorithm 1). In this algorithm, $A(:, 1:p)$ denotes the first p columns of A, and $A(1:p, :)$ the first p rows. The key part of the algorithm is to obtain a basis $\mathcal{O}_k$ for the rowspace of each Hankel matrix H_k of T. The singular value decomposition (SVD)[20] is a robust tool for doing this. It is a decomposition of H_k into factors U_k, Σ_k, V_k, where U_k and V_k are unitary matrices whose columns contain the left and right singular vectors of H_k, and Σ_k is a diagonal matrix with

```
In:   $T$              (an upper triangular matrix)
Out:  $\{\mathbf{T}_k\}$   (a minimal realization)

$\mathcal{O}_{n+1} = [\cdot], \mathcal{C}_{n+1} = [\cdot]$
for $k = n, \cdots, 1$
    $H_k =: U_k \Sigma_k V_k^*$
    $d_k = \mathrm{rank}(\Sigma_k)$
    $\mathcal{C}_k = (U_k \Sigma_k)(:, 1 : d_k)$
    $\mathcal{O}_k = V_k^*(1 : d_k, :)$
    $A_k = \mathcal{O}_k [0 \quad \mathcal{O}_{k+1}]^*$
    $C_k = \mathcal{O}_k(:, 1)$
    $B_k = \mathcal{C}_{k+1}(1, :)$
    $D_k = T(k, k)$
end
```

Algorithm 1. The realization algorithm.

positive entries (the singular values of H_k) on the diagonal. The integer d_k is set equal to the number of nonzero singular values of H_k, and $V_k^*(1 : d_k, :)$ contains the corresponding singular vectors. The rows of $V^*(1 : d_k, :)$ span the row space of H_k. Note that it is natural that $d_1 = 0$ and $d_{n+1} = 0$, so that the realization starts and ends with zero number of states. The rest of the realization algorithm is straightforward in view of the shift-invariance property. It is in fact very reminiscent of the Principal Component identification method in system theory[21].

The above is only an algorithmic outline. Because H_{k+1} has a large overlap with H_k, an efficient SVD updating algorithm can be devised that takes this structure into account. Note that, based on the singular values of H_k, a reduced order model can be obtained by taking a smaller basis for $\mathcal{O}_k$, a technique that is known in the time-invariant context as balanced model reduction. Although widely used for time-invariant systems, this is in fact a “heuristic” model reduction theory, as the modeling error norm is not known. A precise approximation theory results if the tolerance on the error is given in terms of the Hankel norm[15].

4. ORTHOGONAL EMBEDDING OF CONTRACTIVE TIME-VARYING SYSTEMS

This section discusses a constructive solution of the problem of the realization of a given (strictly) contractive time-varying system as the partial transfer operator of a lossless system. This problem is also known as the Darlington problem in classical network theory [22], while in control theory, a variant of it is known as the Bounded Real Lemma [23]. The construction is done in a state space context and gives rise to a time-varying Riccati-type equation. We are necessarily brief here; details can be found in [17].

The problem setting is the following. Let be given the transfer operator T of a contractive causal linear time-varying system with n_1 inputs and n_0 outputs, and let $\mathbf{T}_k = \{A_k, B_k, C_k, D_k\}$ be a given time-varying state space realization of T (as obtained in the previous section). Then determine a unitary and causal multi-port Σ (corresponding to a lossless system) such that $T = \Sigma_{11}$, along with a state realization $\mathbf{\Sigma}$, where

$$\Sigma = \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}, \qquad \mathbf{\Sigma}_k = \begin{bmatrix} A_{\Sigma,k} & C_{\Sigma,k} \\ B_{\Sigma,k} & D_{\Sigma,k} \end{bmatrix}.$$

Without loss of generality we can in addition require $\mathbf{\Sigma}$ to be a unitary realization: ($\mathbf{\Sigma}_k\mathbf{\Sigma}_k^* = I$, $\mathbf{\Sigma}_k^*\mathbf{\Sigma}_k = I$). Since $T^*T + \Sigma_{21}^*\Sigma_{21} = I$, this will be possible only if T is contractive: $I - TT^* \geq 0$. While it is clear that contractivity is a necessary condition, we will require strict contractivity of T in the sequel, which is sufficient to construct a solution to the embedding problem. (The extension to the boundary case is possible but its derivation is non-trivial.)

Theorem 3. *Let T be an upper triangular matrix, with state realization $\mathbf{T}_k = \{A_k, B_k, C_k, D_k\}$. If T is strictly contractive and $\mathbf{T}$ is controllable: $\mathcal{C}_k^*\mathcal{C}_k > 0$ for all k, then the embedding problem has a solution Σ with a lossless realization $\mathbf{\Sigma}_k = \{A_{\Sigma,k}, B_{\Sigma,k}, C_{\Sigma,k}, D_{\Sigma,k}\}$, such that $\Sigma_{11} = T$. This realization has the following properties (where T has n_1 inputs, n_0 outputs, and d_k incoming states at instant k):*

- *A_Σ is state equivalent to A by an invertible state transformation R, i.e., $A_{\Sigma,k} = R_kA_kR_{k+1}^{-1}$,*
- *The number of inputs added to T in Σ is equal to n_0,*
- *The number of added outputs is time-varying and given by $d_k - d_{k+1} + n_1 \geq 0$.*

PROOF (partly). The easy part of the proof is by construction, but the harder existence proofs are omitted. We use the property that a system is unitary if its realization is unitary, and that $T = \Sigma_{11}$ if $\mathbf{T}$ is a submatrix of $\mathbf{\Sigma}$, up to a state transformation.

Step 1. of the construction is to find, for each time instant k, a state transformation R_k and matrices $B_{2,k}$ and $D_{21,k}$ such that the columns of $\mathbf{\Sigma}_{1,k}$,

$$\mathbf{\Sigma}_{1,k} = \begin{bmatrix} R_k & & \\ & I & \\ & & I \end{bmatrix} \left[\begin{array}{cc} A_k & C_k \\ B_k & D_k \\ \hline B_{2,k} & D_{21,k} \end{array}\right] \begin{bmatrix} R_{k+1}^{-1} & \\ & I \end{bmatrix}$$

are isometric, *i.e.*, $(\mathbf{\Sigma}_{1,k})^*\mathbf{\Sigma}_{1,k} = I$. Upon writing out the equations, we obtain, by putting $M_k = R_k^*R_k$, the set of equations

$$\left\{ \begin{array}{ccccccc} M & = & A^*MA & + & B^*B & + & B_2^*B_2 \\ 0 & = & A^*MC & + & B^*D & + & B_2^*D_{21} \\ 1 & = & C^*MC & + & D^*D & + & D_{21}^*D_{21} \end{array} \right. \tag{5}$$

which by substitution lead to

$$M_{k+1} = A_k^*M_kA_k + B_k^*B_k \quad + \left[A_k^*M_kC_k + B_k^*D_k\right](I - D_k^*D_k - C_k^*M_kC_k)^{-1}\left[D_k^*B_k + C_k^*M_kA_k\right].$$

This equation can be regarded as a time-recursive Riccati-type equation with time-varying parameters. It can be shown (see [17]) that $(I - D_k^*D_k - C_k^*M_kC_k)$ is strictly positive (hence invertible) if T is strictly contractive and that M_{k+1} is strictly positive definite (hence R_{k+1} exists and is invertible) if $\mathbf{T}$ is controllable. $B_{2,k}$ and $D_{21,k}$ are determined from (5) in turn as

$$\begin{array}{lcl} D_{21,k} & = & (I - D_k^*D_k - C_k^*M_kC_k)^{1/2} \\ B_{2,k} & = & -(I - D_k^*D_k - C_k^*M_kC_k)^{-1/2}\left[D_k^*B_k + C_k^*M_kA_k\right] \end{array}$$

Step 2. Find a complementary matrix $\mathbf{\Sigma}_{2,k}$ such that $\mathbf{\Sigma}_k = \begin{bmatrix}\mathbf{\Sigma}_{1,k} & \mathbf{\Sigma}_{2,k}\end{bmatrix}$ is a square unitary matrix. This is always possible and reduces to a standard exercise in linear algebra. It can be shown that the system corresponding to $\mathbf{\Sigma}_k$ is indeed an embedding of T.

□

The embedding algorithm can be implemented along the lines of the proof of the embedding theorem. However, as is well known, the Riccati recursions on M_i can be replaced by more efficient algorithms that recursively compute the square root of M_i, *i.e.*, R_i, instead of M_i itself. These are the so-called square-root algorithms. The existence of such algorithms has been known for a long time; see *e.g.*, Morf [24] for a list of

```
In:   $\{\mathbf{T}_k\}$   (a controllable realization of T, $\|T\| < 1$)
Out:  $\{\mathbf{\Sigma}_k\}$   (a unitary realization of embedding $\Sigma$)

$R_1 = [\cdot]$
for $k = 1, \cdots, n$
    $\mathbf{T}_{e,k} = \begin{bmatrix} R_k & & \\ & I & \\ & & I \end{bmatrix} \begin{bmatrix} A_k & C_k \\ B_k & D_k \\ 0 & I \end{bmatrix}$
    $\mathbf{T}'_{e,k} := \Theta_k \mathbf{T}_{e,k}$,   $\Theta_k$ $J$-unitary, and such that $\mathbf{T}'_{e,k}(2,2) = \mathbf{T}'_{e,k}(1,2) = \mathbf{T}'_{e,k}(2,1) = 0$
    $\mathbf{T}'_{e,k} =: \begin{bmatrix} R_{k+1} & 0 \\ 0 & 0 \\ B_{2,k} & D_{21,k} \end{bmatrix}$
    $\mathbf{\Sigma}_{1,k} = \begin{bmatrix} R_k & & \\ & I & \\ & & I \end{bmatrix} \left[\begin{array}{cc} A_k & C_k \\ B_k & D_k \\ \hline B_{2,k} & D_{21,k} \end{array}\right] \begin{bmatrix} R_{k+1}^{-1} & \\ & I \end{bmatrix}$
    $\mathbf{\Sigma}_k = \begin{bmatrix} \mathbf{\Sigma}_{1,k} & \mathbf{\Sigma}_{1,k}^{\perp} \end{bmatrix}$
end
```

Algorithm 2. The embedding algorithm.

pre-1975 references. The square-root algorithm is given in algorithm 2. The algorithm acts on data known at the k-th step: the state matrices A_k, B_k, C_k, D_k, and the state transformation R_k obtained at the previous step. This data is collected in a matrix $\mathbf{T}_{e,k}$. The key of the algorithm is the construction of a J-unitary matrix Θ: $\Theta^* J \Theta = J$, where

$$\Theta = \begin{bmatrix} \Theta_{11} & \Theta_{12} & \Theta_{13} \\ \Theta_{21} & \Theta_{22} & \Theta_{23} \\ \Theta_{31} & \Theta_{32} & \Theta_{33} \end{bmatrix} \qquad J = \begin{bmatrix} I & & \\ & I & \\ & & -I \end{bmatrix},$$

such that certain entries of $\mathbf{T}'_{e,k} = \Theta \mathbf{T}_{e,k}$ are zero. We omit the necessary theory on this. It turns out that, because Θ_k is J-unitarity, we have that $\mathbf{T}'^{*}_{e,k} J \mathbf{T}_{e,k} = \mathbf{T}^{*}_{e,k} J \mathbf{T}_{e,k}$; writing these equations out and comparing with (5) it is seen that the remaining non-zero entries of $\mathbf{T}'_{e,k}$ are precisely the unknowns R_{k+1}, $B_{2,k}$ and $D_{21,k}$. It is also a standard technique to factor Θ even further down into elementary (J)-unitary operations that each act on only two scalar entries of $\mathbf{T}_e$, and zero one of them by applying an elementary J-unitary rotation of the form

$$\theta = \frac{1}{c} \begin{bmatrix} 1 & s \\ s & 1 \end{bmatrix}, \qquad c^* c + s^* s = 1\,.$$

With B_2 and D_{21} known, it is conjectured that it is not really necessary to apply the state transformation by R and to determine the orthogonal complement of $\mathbf{\Sigma}_1$, if in the end only a cascade factorization of T is required, much as in [25].

5. CASCADE FACTORIZATION OF LOSSLESS MULTI-PORTS

In the previous section, it was discussed how a strictly contractive transfer operator T can be embedded into a lossless scattering operator Σ. We will now derive minimal structural factorizations, and corresponding lossless cascade networks, for arbitrary lossless multi-ports Σ with square unitary realizations $\{\mathbf{\Sigma}_k\}$. The network synthesis is a two-stage algorithm:

1. Using unitary state transformations, bring $\mathbf{\Sigma}$ into a form that allows a minimal factorization (*i.e.*, a minimal number of factors). We choose to make the A-matrix of $\mathbf{\Sigma}$ upper triangular. This leads to a QR-iteration on the $\{A_k\}$ and is the equivalent of the Schur decomposition (eigenvalue computations) of A that would be required for time-invariant systems.

2. Using Givens rotations extended by I to the correct size, factor $\mathbf{\Sigma}$ into a product of such elementary sections. From this factorization, the lossless cascade network follows directly.

While the factorization strategy is more or less clear-cut, given a state space matrix that allows a minimal factorization, the optimal (or desired) cascade structure is not. We will present a solution based on $\mathbf{\Sigma}$ itself. However, many other solutions exist, for example based on a factorization of a J-unitary transfer operator related to Σ, yielding networks with equal structure but with different signal flow directions; this type of network is favored in the time-invariant setting for selective filter synthesis and was first derived by Deprettere and Dewilde [26] (see also [27]). To avoid eigenvalue computations, cascade factorizations based on a state transformation to Hessenberg form are also possible [28, 29]. In the time-varying setting, eigenvalue computations are in a natural way replaced by recursions consisting of QR factorizations, so this motivation seems no longer to be an issue.

5.1. Time-varying Schur decomposition

Let A_k be the A-matrix of $\mathbf{\Sigma}$ at time k. The first step in the factorization algorithm is to find square unitary state transformations Q_k such that

$$Q_k^* A_k Q_{k+1} \;=\; R_k \tag{6}$$

has R_k upper triangular. If A_k is not square, say of size $d_k \times d_{k+1}$, then R_k will be of the same size and also be rectangular. In that case, 'upper triangular' is understood as usual in QR-factorization, *i.e.*, the lower-left $d \times d$ corner ($d = \min[d_k, d_{k+1}]$) of R_k consists of zeros (figure 5). In the time-invariant case, expression (6) would read $Q^*AQ = R$, and the solution is then precisely the Schur-decomposition of A. In that context, the main diagonal of A consists of its eigenvalues, which are the (inverses of the) poles of the system. In the present context, relation (6) is effectively the (unshifted) QR-iteration algorithm that is sometimes used to compute eigenvalues if all A_k are the same [20]:

$$\begin{array}{rcl} Q_1^* A_1 & =: & R_1 Q_2^* \\ Q_2^* A_2 & =: & R_2 Q_3^* \\ Q_3^* A_3 & =: & R_3 Q_4^* \\ & \cdots & \end{array}$$

Each step in the computation amounts to a multiplication by the previously computed Q_k, followed by a QR-factorization of the result, yielding Q_{k+1} and R_k. Since we are in the context of finite upper triangular matrices whose state realization starts with 0 states at instant $k = 1$, we can take as initial transformation $Q_1 = [\,\cdot\,]$.

5.2. Elementary Givens Rotations

We say that $\hat{\Sigma}$ is an elementary orthogonal rotation if $\hat{\Sigma}$ is a 2×2 unitary matrix,

$$\hat{\Sigma} = \begin{bmatrix} c^* & s \\ -s^* & c \end{bmatrix}, \tag{7}$$

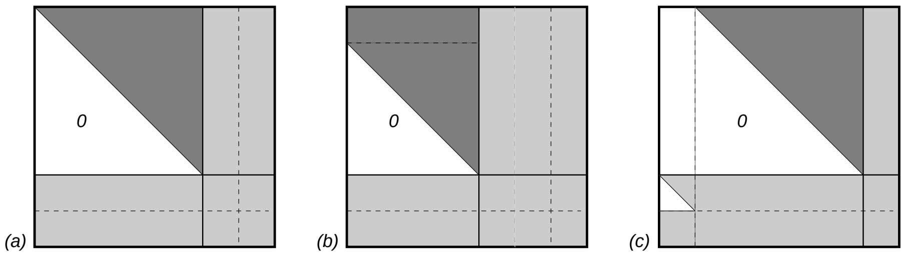

Figure 5. Schur forms of Σ. (*a*) Constant state dimension, (*b*) shrinking state dimension, (*c*) growing state dimension.

with $c^*c+s^*s=1$. An important property of elementary rotations is that they can be used to zero a selected entry of a given operator: for given a and b, there exists an elementary orthogonal rotation $\hat{\Sigma}$ such that

$$\hat{\Sigma}^* \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a' \\ 0 \end{bmatrix},$$

i.e., such that $s^*a+c^*b=0$ and $a'=(a^*a+b^*b)^{1/2}$. In this case, $\hat{\Sigma}$ is called a Givens rotation, and we write $\hat{\Sigma}=\text{givens}[a;b]$ in algorithms. Givens rotations will be used in the next section to factor a given state realization into pairs of elementary rotations, or elementary sections. The basic operation, the computation of one such pair, is merely the application of two elementary Givens rotations: let $\mathbf{T}$ be a 3×3 matrix

$$\mathbf{T} = \left[\begin{array}{c|cc} a & c_1 & c_2 \\ \hline b_1 & d_{11} & d_{12} \\ b_2 & d_{21} & d_{22} \end{array}\right]$$

such that it satisfies the orthogonality conditions $[a^* \quad b_1^* \quad b_2^*]\,\mathbf{T} = [1 \quad 0 \quad 0]$, then there exist elementary rotations $\mathbf{\Sigma}_1, \mathbf{\Sigma}_2$ such that $\mathbf{\Sigma}_2^*\mathbf{\Sigma}_1^*\mathbf{T}=\mathbf{T}'$, with

$$\mathbf{\Sigma}_1 = \begin{bmatrix} c_1^* & s_1 & \\ -s_1^* & c_1 & \\ & & 1 \end{bmatrix}, \qquad \mathbf{\Sigma}_2 = \begin{bmatrix} c_2^* & & s_2 \\ & 1 & \\ -s_2^* & & c_2 \end{bmatrix}, \qquad \mathbf{T}' = \left[\begin{array}{c|cc} 1 & 0 & 0 \\ \hline 0 & d'_{11} & d'_{12} \\ 0 & d'_{21} & d'_{22} \end{array}\right].$$

5.3. Factorization

Let be given a lossless state realization $\mathbf{\Sigma}$ of a lossless two-port Σ. For each time instant k, we will construct a cascade factorization of $\mathbf{\Sigma}_k$ by repeated use of the above generic factorization step. Assume that a preprocessing state transformation based on the Schur decomposition has been carried out, *i.e.*, that each $\mathbf{\Sigma}_k$ has its A_k upper triangular. For the sake of exposition, we specialize to the case where A_k is a square $d\times d$ matrix and Σ has a constant number of two inputs and outputs, but the method is easily generalized. Thus

$$\mathbf{\Sigma}_k = \begin{bmatrix} A_k & C_k \\ B_k & D_k \end{bmatrix} = \left[\begin{array}{cccc|cc} a & \cdot & \cdot & \cdot & c_1 & c_2 \\ & \bullet & \cdot & \cdot & \cdot & \cdot \\ & & \bullet & \cdot & \cdot & \cdot \\ & & & \bullet & \cdot & \cdot \\ \hline b_1 & \cdot_3 & \cdot_5 & \cdot & d_{11} & d_{12} \\ b_2 & \cdot_4 & \cdot_6 & \cdot & d_{21} & d_{22} \end{array}\right] \tag{8}$$

For $i=1,\cdots,d,\ j=1,2$, let $\hat{\Sigma}_{ij}$ be an elementary (Givens) rotation matrix, and denote by $\mathbf{\Sigma}_{ij}$ the extension of $\hat{\Sigma}_{ij}$ to an elementary rotation of the same size as $\mathbf{\Sigma}$, with $\mathbf{\Sigma}_{ij}=I$ except for the four entries (i,i), $(i,d+j)$, $(d+j,i)$, $(d+j,d+j)$, which together form the given $\hat{\Sigma}_{ij}$. Then $\mathbf{\Sigma}_k$ admits a (minimal) factorization

$$\mathbf{\Sigma}_k = [\mathbf{\Sigma}_{1,1}\mathbf{\Sigma}_{1,2}]\cdot[\mathbf{\Sigma}_{2,1}\mathbf{\Sigma}_{2,2}]\cdots[\mathbf{\Sigma}_{d,1}\mathbf{\Sigma}_{d,2}]\cdot\mathbf{\Sigma}'. \tag{9}$$

```
In:   $\mathbf{\Sigma}_k$                         (in Schurform; $A_k : d_k \times d_{k+1}$, $n_1$ inputs, $n_0$ outputs)
Out:  $\{\mathbf{\Sigma}_{ij}\}, \{\mathbf{\Sigma}'_{ij}\}$   (elementary rotations: factors of $\mathbf{\Sigma}_k$)

– if $d_k > d_{k+1}$ (‘shrink’), move first $d_k - d_{k+1}$ rows of $[A_k\ \ C_k]$ to $[B_k\ \ D_k]$.
– if $d_k < d_{k+1}$ (‘grow’), move first $d_{k+1} - d_k$ rows of $[B_k\ \ D_k]$ to $[A_k\ \ C_k]$.

for $i = 1, \cdots, d_k$
  for $j = 1, \cdots, n_1$
    $\hat{\Sigma}_{ij}$ = givens$[A_k(i,i); B_k(j,i)]$
    $\mathbf{\Sigma}_k$ := $\mathbf{\Sigma}^*_{ij} \mathbf{\Sigma}_k$
  end
end

$\mathbf{\Sigma}'_k = D_{\Sigma_k}$          (also factor ‘residue’)
for $i = 1, \cdots, n_0$
  for $j = 1, \cdots, n_1$
    $\hat{\Sigma}'_{ij}$ = givens$[\mathbf{\Sigma}'(i,i); \mathbf{\Sigma}'(j,i)]$
    $\mathbf{\Sigma}'$ := $\mathbf{\Sigma}'^*_{ij} \mathbf{\Sigma}'$
end
```

Algorithm 3. The factorization algorithm.

into extended elementary rotations, where the ‘residue’ $\mathbf{\Sigma}'$ is a state realization matrix of the same form as $\mathbf{\Sigma}_k$, but with $A = I$, $B = C = 0$, and D unitary. The factorization is based on the cancellation, in turn, of the entries of B_k of $\mathbf{\Sigma}_k$ in equation (8). To start, apply the generic factorization of section 5.2 to the equally-named entries a, b_1, b_2 etc. in equation (8), yielding Givens rotations $\hat{\Sigma}_{1,1}$ and $\hat{\Sigma}_{1,2}$, which are extended by I to $\mathbf{\Sigma}_{1,1}$ and $\mathbf{\Sigma}_{1,2}$. The resulting state space matrix $[\mathbf{\Sigma}^*_{1,2}\mathbf{\Sigma}^*_{1,1}]\, \mathbf{\Sigma}_k$ has $(1,1)$-entry $a' = I$, and of necessity zeros on the remainder of the first column and row. The factorization can now be continued in the same way, in the order indicated by the labeling of the entries of B_k in equation (8) by focusing on the second-till-last columns and rows of this intermediate operator. The result is the factorization of $\mathbf{\Sigma}_k$ in (9). Algorithm 3 summarizes the procedure for the general case of non-square A_k-matrices. In the case that the state dimension shrinks, *i.e.*, $d_k > d_{k+1}$, then the first $d_k - d_{k+1}$ states are treated as inputs rather than states, but the actual factorization algorithm remains the same. If the state dimension grows ($d_k < d_{k+1}$), then the states that are added can be treated as extra outputs in the factorization algorithm.

With the above factorization, it is seen that the actual operations that are carried out are pairs of rotations. The network corresponding to this factorization scheme is as depicted in figure 6, where each circle indicates an elementary section as in equation (7). This picture is obtained by considering the sequence of operations that are applied to a vector $[x_{1,k}\ \ x_{2,k}\ \cdots\ x_{d_k,k}\ ;\ u_k\ \ z_k]$ when it is multiplied by $\mathbf{\Sigma}_k$ in factored form. Each state variable interacts with each input quantity $[u_k\ \ z_k]$, after which it has become a ‘next state’ variable. The residue $\mathbf{\Sigma}'$ appears as the single rotation at the right. In the picture, we put $z_k = 0$ to obtain T as transfer $u_k \to y_k$. The secondary output of Σ is discarded.

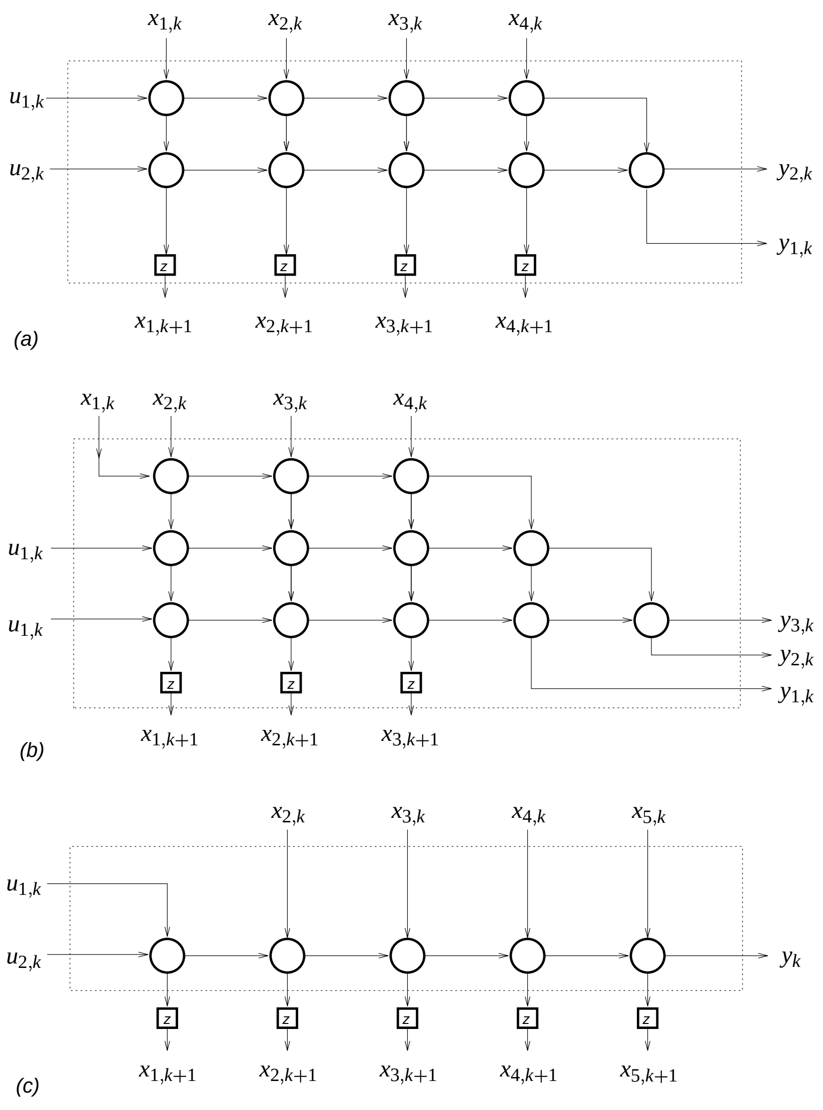

Figure 6. Lossless cascade realizations of a contractive system T, stage k. (a) Constant state dimension, (b) shrinking state dimension, (c) growing state dimension. Outputs marked by '$*$' are ignored.

6. CONCLUDING REMARKS

In the preceding sections, we have presented algorithms to compute, for a given upper triangular matrix, a computational model of lowest possible complexity. We have also derived synthesis algorithms to realize this model as a lossless cascade network in which the basic processors are elementary (Givens) rotations. This provides a numerically stable implementation in which a minimal number of parameters (the rotation angles) are used. The number of operations needed to implement a matrix-vector multiplication is $\mathcal{O}(2d_k)$ elementary rotations for a stage with d_k states, or $\mathcal{O}(8d_k)$ multiplications per stage. The total number of multiplications that are required is $\mathcal{O}(8dn)$ if $d \ll n$ is some averaged number of states, as compared to $\mathcal{O}(1/2\,n^2)$ for a direct matrix-vector multiplication. Of course, it is possible to select other structures than the lossless cascade structure to realize a given computational model. Finally, if the number of states at any point is larger than desired, then it is possible to find optimal approximating matrices: for a given error tolerance, measured in the Hankel norm, it is known how many states the approximating computational network will require at least.

The realization and approximation theory presented in this paper are consequences of time-varying systems theory. This theory can be applied in many ways to reduce matrix computations if the given matrix exhibits what was called a state structure, and can be used to determine new types of matrix approximations for enforcing such a state structure onto matrices. The computational expensive part of the presented scheme is the retrieval of the structure of the given matrix, and it is even more expensive to compute a reduced order model. This is an instance of a more general property on computations with structured matrices: algorithms which exploit the structure can be efficient only after the structure has been captured in some way, which either requires advance knowledge of this structure (for example, the fact that a matrix is Toeplitz or has zeros at specific entries), or will be computationally expensive, because all entries of the matrix must be operated upon at least once. In the case of matrices with state structure, the derivation of an (approximating) model will make sense only if one is interested in a sparse representation of the given matrix, that is, if the resulting model is heavily used in other computational procedures. The focus of the paper on matrix-vector multiplications must in this respect be regarded only as an example on how one can exploit knowledge of such sparse models.

Additional results obtained during the review process include the inversion of a general (not necessarily upper triangular) matrix using time-varying state space techniques [30]. More details on the discussed topics are available in [31].

7. ACKNOWLEDGEMENT

This research was supported in part by the commission of the EC under the ESPRIT BRA program 6632 (NANA2).

8. REFERENCES

[1] I. Schur, “Uber Potenzreihen, die im Innern des Einheitskreises Beschränkt Sind, I,” *J. Reine Angew. Math.*, **147**:205–232, 1917 Eng. Transl. *Operator Theory: Adv. Appl.*, vol. 18, pp. 31-59, Birkhäuser Verlag, 1986.

[2] N. Levinson, “The Wiener RMS Error Criterion in Filter Design and Prediction,” *J. Math. Phys.*, **25**:261–278, 1947.

[3] I. Gohberg and A. Semencul, “On the Inversion of Finite Toeplitz Matrices and their Continuous Analogs,” *Mat. Issled.*, **2**:201–233, 1972.

[4] J. Chun, T. Kailath, and H. Lev-Ari, “Fast Parallel Algorithms for *QR* and Triangular Factorizations,” *SIAM J. Sci. Stat. Comp.*, **8**(6):899–913, 1987.

[5] T. Kailath, S.Y. Kung, and M. Morf, “Displacement Ranks of Matrices and Linear Equations,” *J. Math. Anal. Appl.*, **68**(2):395–407, 1979.

[6] H. Lev-Ari and T. Kailath, “Lattice Filter Parametrization and Modeling of Non-Stationary Processes,” *IEEE Trans. Informat. Th.*, **30**(1):2–16, January 1984.

[7] H. Lev-Ari and T. Kailath, “Triangular Factorizations of Structured Hermitian Matrices,” In *Operator Theory: Advances and Applications*, volume 18, pp. 301–324. Birkhäuser Verlag, 1986.

[8] H. Lev-Ari and T. Kailath, “Lossless Arrays and Fast Algorithms for Structured Matrices,” In Ed. F. Deprettere and A.J. van der Veen, editors, *Algorithms and Parallel VLSI Architectures*, volume A, pp. 97–112. Elsevier, 1991.

[9] J. Chun, *“Fast Array Algorithms for Structured Matrices,”* PhD thesis, Stanford Univ., Stanford, CA, 1989.

[10] B.L. Ho and R.E. Kalman, “Effective Construction of Linear, State-Variable Models from Input/Output Functions,” *Regelungstechnik*, **14**:545–548, 1966.

[11] T. Kailath, “A Theorem of I. Schur and its impact on Modern Signal Processing,” In *Operator Theory: Advances and Applications*, volume 18, pp. 9–30. Birkhäuser Verlag, Basel, 1986.

[12] H. Ahmed, J. Delosme, and M. Morf, “Highly Concurrent Computing Structures for Matrix Arithmetic and Signal Processing,” *Computer*, pp. 65–82, January 1982.

[13] P. Dewilde and E. Deprettere, “The Generalized Schur Algorithm: Approximation and Hierarchy,” In *Operator Theory: Advances and Applications*, volume 29, pp. 97–116. Birkhäuser Verlag, 1988.

[14] V.M. Adamjan, D.Z. Arov, and M.G. Krein, “Analytic Properties of Schmidt Pairs for a Hankel Operator and the Generalized Schur-Takagi Problem,” *Math. USSR Sbornik*, **15**(1):31–73, 1971 (transl. of *Iz. Akad. Nauk Armjan. SSR Ser. Mat. 6 (1971))*.

[15] P.M. Dewilde and A.J. van der Veen, “On the Hankel-Norm Approximation of Upper-Triangular Operators and Matrices,” *Integral Equations and Operator Theory*, **17**(1):1–45, 1993.

[16] A.J. van der Veen and P.M. Dewilde, “Time-Varying System Theory for Computational Networks,” In P. Quinton and Y. Robert, editors, *Algorithms and Parallel VLSI Architectures, II*, pp. 103–127. Elsevier, 1991.

[17] A.J. van der Veen and P.M. Dewilde, “Orthogonal Embedding Theory for Contractive Time-Varying Systems,” In *Proc. IEEE ISCAS*, pp. 693–696, 1992.

[18] P.M. Dewilde, “A Course on the Algebraic Schur and Nevanlinna-Pick Interpolation Problems,” In Ed. F. Deprettere and A.J. van der Veen, editors, *Algorithms and Parallel VLSI Architectures*, volume A, pp. 13–69. Elsevier, 1991.

[19] A.J. van der Veen and P.M. Dewilde, “AAK Model Reduction for Time-Varying Systems,” In J. Vandewalle e.a., editor, *Proc. Eusipco’92 (Brussels)*, pp. 901–904. Elsevier Science Publishers, August 1992.

[20] G. Golub and C.F. Van Loan, *“Matrix Computations,”* The Johns Hopkins University Press, 1984.

[21] S.Y. Kung, “A New Identification and Model Reduction Algorithm via Singular Value Decomposition,” In *Twelfth Asilomar Conf. on Circuits, Systems and Comp.*, pp. 705–714, Asilomar, CA, November 1978.

[22] S. Darlington, “Synthesis of Reactance 4-Poles which Produce Prescribed Insertion Loss Characteristics,” *J. Math. Phys.*, **18**:257–355, 1939.

[23] B.D.O. Anderson and S. Vongpanitlerd, *“Network Analysis and Synthesis,”* Prentice Hall, 1973.

[24] M. Morf and T. Kailath, “Square-Root Algorithms for Least-Squares Estimation,” *IEEE Trans. Automat. Control*, **20**(4):487–497, 1975.

[25] H. Lev-Ari and T. Kailath, “State-Space Approach to Factorization of Lossless Transfer Functions and Structured Matrices,” *Lin. Alg. Appl.*, **162**:273–295, February 1992.

[26] E. Deprettere and P. Dewilde, “Orthogonal Cascade Realization of Real Multiport Digital Filters,” *Circuit Theory and Appl.*, **8**:245–272, 1980.

[27] P.M. Van Dooren and P.M. Dewilde, “Minimal Cascade Factorization of Real and Complex Rational Transfer Matrices,” *IEEE Trans. Circuits Syst.*, **28**(5):390–400, 1981.

[28] S.K. Rao and T. Kailath, “Orthogonal Digital Filters for VLSI Implementation,” *IEEE Trans. Circuits Syst.*, **31**(11):933–945, November 1984.

[29] U.B. Desai, “A State-Space Approach to Orthogonal Digital Filters,” *IEEE Trans. Circuits Syst.*, **38**(2):160–169, February 1991.

[30] A.J. van der Veen and P.M. Dewilde, “Connections of Time-Varying Systems and Computational Linear Algebra,” In H. Dedieu, editor, *Circuit Theory and Design: Proc. 11-th ECCTD*, pp. 167–172, Davos, Switzerland, August 1993. Elsevier.

[31] A.J. van der Veen, *“Time-Varying System Theory and Computational Modeling: Realization, Approximation, and Factorization,”* PhD thesis, Delft University of Technology, Delft, The Netherlands, June 1993.