Math. Control Signals Systems (1989) 2: 109–136

Mathematics of Control, Signals, and Systems

A Systolic Algorithm for Riccati and Lyapunov Equations*

J.-P. Charlier† and P. Van Dooren†

Abstract. Riccati and Lyapunov equations can be solved using the recursive matrix sign method applied to symmetric matrices constructed from the corresponding Hamiltonian matrices. In this paper we derive an efficient systolic implementation of that algorithm where the LDL^T and UDU^T decompositions of those symmetric matrices are propagated. As a result the solution of a class of Riccati and Lyapunov equations can be obtained in $O(n)$ time steps on a bidimensional (triangular) grid of $O(n^2)$ processors, leading to an optimal speedup.

Key words. Riccati equation, Lyapunov equation, Parallel algorithm, Matrix sign function, Numerical methods.

1. Introduction

Matrix Lyapunov equations and matrix Riccati equations continue to receive attention in the control literature [BS], [B1], [B3], [B4], [L1], [PLS], [H1], [V], mainly because of their key role in various control and filtering problems. In this paper we give a new numerical technique for solving a class of such equations. The method nicely combines symmetric factorization techniques with the matrix sign algorithm. The resulting algorithm has the particular advantage of being easily implemented on parallel architectures of distributed type. To our knowledge it is the first parallel implementation developed specifically for these equations and it results in an $O(n)$-time algorithm when using a triangular array of $O(n^2)$ processors. The basic operations of the iterative algorithm amount to solving triangular systems of equations and updating (skew-)orthogonal decompositions of nearly triangular matrices. The interleaving of these recursive operations on a triangular array of processors is optimized in order to yield a minimum time delay between two consecutive recursions. On pipeline machines (supercomputers such as CRAY, CYBER, etc) the algorithm should also perform well [DS] since the basic operations are typically of vector type and allow efficient "chaining."

In the next section we first recall the matrix sign algorithm and its use for solving various *symmetric* quadratic matrix equations. We follow the development of Gardiner and Laub [GL] from which this paper is strongly inspired. Then we present a factored version of the matrix sign algorithm as applied to these symmetric

* Date received: June 17, 1987. Date revised: December 12, 1987.

† Philips Research Laboratory, Av. Van Becelaere 2, Box 8, B-1170 Brussels, Belgium.

equations, which is in a sense the main idea of this paper. In Section 3 we give some basic results on symmetric recursions which are then used in Section 4 for tackling a class of more general algebraic Riccati equations using factored recursions. Our factored version can be rewritten in a combined *forward* and *backward* factorization algorithm which is crucial to obtain a parallel implementation with a minimum of transmission lags, as pointed out in Section 5. The actual implementation details of this parallel algorithm are then presented in Section 6. There we show how the combination of the forward and backward recursions can be done on the same processors without any additional time steps. The last section briefly presents some numerical aspects of the algorithm.

2. The Matrix Sign Algorithm and Its Variants

2.1. *The Sign Algorithm Applied to Hamiltonian Matrices*

The matrix sign algorithm has been used [A], [B1], [B4], [GL] for solving both the Lyapunov equation

$$PA + A^T P + Q = 0 \tag{1}$$

and the more general algebraic Riccati equation (ARE)

$$PA + A^T P - PBR^{-1}B^T P + Q = 0 \tag{2}$$

which arise in design problems for continuous-time systems. The key technique here is to convert those problems to an eigenvalue problem involving the *Hamiltonian* matrix

$$H := \begin{pmatrix} A & -BR^{-1}B^T \\ -Q & -A^T \end{pmatrix} \tag{3}$$

(where we take $B = 0$ for the Lyapunov equation). Standard assumptions which are made in this context are $Q > 0$, $R > 0$, and H has no pure imaginary eigenvalues, and these will also be assumed in the sequel. We do not consider the case of singular or indefinite Q, except for a few remarks in the text. In general, an order $2n$ matrix H is said to be *Hamiltonian* if and only if $JH^TJ = H$, with

$$J := \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}. \tag{4}$$

Note that $J^{-1} = J^T = -J$. It is known that the stable subspace $\mathcal{H}_s$ of the Hamiltonian matrix H, written as

$$\mathcal{H}_s = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}, \tag{5}$$

in an arbitrary basis, yields a solution to the above equations in the form

$$P = X_2 X_1^{-1}. \tag{6}$$

The matrix sign algorithm is a simple way to compute this subspace. Consider the following recursion for an $N \times N$ matrix H ($N = 2n$):

$$H_{k+1} = \frac{1}{2c_k} \cdot [H_k + c_k^2 H_k^{-1}], \qquad c_k = |\det(H_k)|^{1/N} \tag{7}$$

started with $H_0 = H$. Notice here that this implies that H_k is Hamiltonian for every k (it is easily seen that the property $JH_k^T J = H_k$ is maintained at each step of the recursion). This recursion is known to converge (rather rapidly) to a semisimple matrix [B1]

$$\operatorname{sign}(H) := H_\infty = \lim_{k\to\infty} H_k \tag{8}$$

with all eigenvalues equal to ± 1 and with the *same* stable and unstable invariant subspaces as H_0. In other words, H and H_∞ have a joint decomposition of the type

$$H = T^{-1}\begin{pmatrix} H_s & 0 \\ 0 & H_u \end{pmatrix} T, \qquad H_\infty = T^{-1}\begin{pmatrix} -I_n & 0 \\ 0 & I_n \end{pmatrix} T. \tag{9}$$

The null space of $H_\infty + I_{2n}$ is then also the requested stable invariant subspace $\mathscr{H}_s$ and P can be computed via [B1], [B4]

$$W := \operatorname{sign}(H) + I_{2n} = \begin{pmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{pmatrix}, \qquad P = -\begin{pmatrix} W_{12} \\ W_{22} \end{pmatrix}^+ \begin{pmatrix} W_{11} \\ W_{21} \end{pmatrix},$$

where $(\cdot)^+$ denotes the pseudoinverse. This specializes to

$$P = -W_{12}^{-1} W_{11} = -W_{22}^{-1} W_{21} \tag{10}$$

when the relevant matrices are invertible.

2.2. *Generalized and Symmetric Recursions*

In [GL] a modified recursion is given which exploits some of the structure of the matrix H in order to save some computations. We first recall the *generalized* matrix sign algorithm presented there:

$$Z_{k+1} = \frac{1}{2c_k} \cdot [Z_k + c_k^2 Y Z_k^{-1} Y], \qquad c_k = |\det(Z_k)/\det(Y)|^{1/N}. \tag{11}$$

When applying this recursion to a matrix $Z_0 = Y_1 H_0 Y_2$ with $Y = Y_1 Y_2$, then we also have the relation $Z_k = Y_1 H_k Y_2$ in each step of the recursions (7) and (11). This is easily seen by multiplying (11) left and right by Y_1^{-1} and Y_2^{-1}, respectively, giving

$$Y_1^{-1} Z_{k+1} Y_2^{-1} = \frac{1}{2c_k} \cdot [Y_1^{-1} Z_k Y_2^{-1} + c_k^2 (Y_1^{-1} Z_k Y_2^{-1})^{-1}], \qquad c_k = |\det(Y_1^{-1} Z_k Y_2^{-1})|^{1/N}, \tag{12}$$

which shows the relation with (7). We thus have $Z_\infty = Y_1 H_\infty Y_2 = Y_1 \operatorname{sign}(H) Y_2$.

While H_k is Hamiltonian for every k this is not really exploited in (7). On the other hand, when using $Z_0 = JH_0$ (i.e., $Y = Y_1 = J$, $Y_2 = I$) the recursion (11) becomes

$$JH_{k+1} = \frac{1}{2c_k}(JH_k + c_k^2 JH_k^{-1}) = \frac{1}{2c_k}(JH_k + c_k^2 J(JH_k)^{-1} J), \qquad c_k = |\det(JH_k)|^{1/N} \tag{13}$$

and each matrix $Z_k = JH_k$ is *symmetric*, which is much easier to exploit algorithmi-

cally. Similar comments also hold when using $Z_0 = H_0 J$ (i.e., $Y_1 = I$, $Y = Y_2 = J$):

$$H_{k+1}J = \frac{1}{2c_k}(H_k J + c_k^2 H_k^{-1} J) = \frac{1}{2c_k}(H_k J + c_k^2 J(H_k J)^{-1} J), \qquad c_k = |\det(H_k J)|^{1/N}. \tag{14}$$

Obviously these two recursions are very similar and most results derived for one of them will also apply to the other. In the sequel we therefore focus on (13) only.

In Section 3 we show that such generalized symmetric recursions could in fact be constructed for computing sign(M) of *any* nonsingular matrix M. The construction of the corresponding Z_0, Y_1, and Y_2 matrices, though, are not as simple then as for a Hamiltonian matrix.

2.3. *Factored Recursions*

Since in the recursion (13) only symmetric matrices are involved, we would like to exploit these properties when computing the inverse of JH_k as well as the updated matrix JH_{k+1}. This leads to the idea of using factorizations of the type $JH_k = L_k D_k L_k^T$, where L_k is lower triangular and D_k is diagonal with nonzero diagonal elements (equal to ± 1 when normalized). Such factorizations exist *provided* all the leading principal submatrices of JH_k are nonsingular [GV, pp. 82–84]. In Section 3 it is shown that this in fact follows *automatically* from the standard assumptions $Q > 0$, $R > 0$, and $\det(H) \neq 0$; moreover, all *normalized* D_k matrices are equal to the matrix

$$\Sigma := \begin{pmatrix} -I_n & 0 \\ 0 & I_n \end{pmatrix}. \tag{15}$$

Let us thus assume that for each matrix JH_k of the recursion (13), we have $JH_k = L_k \Sigma L_k^T$. Then we can rewrite (13) as

$$L_{k+1}\Sigma L_{k+1}^T = \frac{1}{2c_k}(L_k \quad c_k J L_k^{-T}) \begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix} \begin{pmatrix} L_k^T \\ c_k L_k^{-1} J^T \end{pmatrix}. \tag{16}$$

This suggests that we look for updating formulas for L_k instead of JH_k. Such an updating formula amounts to determining a $\begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix}$-unitary matrix R_k:

$$R_k \begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix} R_k^T = \begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix}, \tag{17}$$

such that

$$\frac{1}{\sqrt{2c_k}}(L_k \quad c_k J L_k^{-T}) R_k = (L_{k+1} \quad 0), \tag{18}$$

where L_{k+1} is lower triangular. From (17), (18) we indeed derive (16) and it follows that $JH_{k+1} = L_{k+1}\Sigma L_{k+1}^T$. The converse, namely that (16) also implies the existence of an R_k satisfying (17), (18), is the following theorem.

Theorem 1. *Let the matrices JH_k and JH_{k+1} of the recursion* (13) *both possess a decomposition $L_i \Sigma L_i^T$ with L_i lower triangular (for $i = k, k+1$) and $\Sigma = \operatorname{diag}\{-I_n, I_n\}$.*

Then there exists a $\operatorname{diag}\{\Sigma, -\Sigma\}$*-unitary transformation* R_k *such that*

$$\frac{1}{\sqrt{2c_k}}(L_k \quad c_k J L_k^{-T}) R_k = (L_{k+1} \quad 0). \tag{19}$$

Proof. Let $\hat{\Sigma}$ be an arbitrary sign matrix of dimension $\hat{n} \times \hat{n}$ and let $\hat{A}$ and $\hat{B}$ be two $\hat{m} \times \hat{n}$ matrices satisfying

$$\hat{A} \cdot \hat{\Sigma} \cdot \hat{A}^T = \hat{B} \cdot \hat{\Sigma} \cdot \hat{B}^T. \tag{20}$$

Then there always exists a $\hat{\Sigma}$-unitary matrix $\hat{S}$ such that

$$\hat{A} \cdot \hat{S} = \hat{B} \tag{21}$$

if $\hat{A} \cdot \hat{\Sigma} \cdot \hat{A}^T = \hat{B} \cdot \hat{\Sigma} \cdot \hat{B}^T$ has full rank $\hat{m}$ [B2] (note that this condition implies $\hat{m} \leq \hat{n}$). Since L_{k+1} was assumed to be nonsingular, the theorem follows directly from this result. ■

In Section 5, where we look at parallel implementations of such factored recursions, we show that it has some advantages to propagate *two* types of factorizations at once, namely

$$JH_k = L_k \Sigma L_k^T \tag{22}$$

and

$$J(JH_k)J = -H_k J = U_k \Sigma U_k^T, \tag{23}$$

where U_k is chosen to be *upper triangular*. That such a decomposition is again possible for all k is proved in the next section. This then leads to the *coupled* recursions (derived from (13) and (14), respectively)

$$L_{k+1} \Sigma L_{k+1}^T = \frac{1}{2c_k}(L_k \quad c_k U_k^{-T}) \begin{pmatrix} \Sigma & 0 \\ 0 & \Sigma \end{pmatrix} \begin{pmatrix} L_k^T \\ c_k U_k^{-1} \end{pmatrix}, \tag{24}$$

$$U_{k+1} \Sigma U_{k+1}^T = \frac{1}{2c_k}(U_k \quad c_k L_k^{-T}) \begin{pmatrix} \Sigma & 0 \\ 0 & \Sigma \end{pmatrix} \begin{pmatrix} U_k^T \\ c_k L_k^{-1} \end{pmatrix}. \tag{25}$$

Obtaining L_{k+1} and U_{k+1} from L_k and U_k then amounts to determining $\begin{pmatrix} \Sigma & 0 \\ 0 & \Sigma \end{pmatrix}$-unitary matrices S_k and T_k such that

$$\frac{1}{\sqrt{2c_k}}(L_k \quad c_k U_k^{-T}) S_k = (L_{k+1} \quad 0) \tag{26}$$

and

$$\frac{1}{\sqrt{2c_k}}(U_k \quad c_k L_k^{-T}) T_k = (U_{k+1} \quad 0). \tag{27}$$

When comparing the *single* recursion (19) with the combined system (26), (27), it appears that we are doing too much work in the latter. But the system (26), (27) has a number of important advantages. First there is no need to multiply with the matrix J as is done in (19). At first sight, this is not a true improvement because of the form of J. However, for the general symmetric recursion analyzed in the next

two sections, multiplication with J is replaced by multiplication with a full matrix Y and savings can then be significant. Secondly, all matrices in (26), (27) are triangular which leads to a faster triangularization than for (19). Last, but not least, the computations in (26), (27) are completely analogous and the algorithm for both reductions to triangular form is better suited for implementation on a parallel machine than the one for (19). These different aspects are further discussed in later sections.

3. General Results on Symmetric Recursions

In this section we look at general symmetric recursions of type (11) and analyze their properties in view of deriving factored forms for them. We especially look at properties inspired from the special cases (13), (14) related to Hamiltonian matrices. The idea is to use these general recursions for solving Lyapunov and Riccati equations, for example, those encountered in implicit and/or discrete-time systems to which the classical sign algorithm cannot be directly applied.

Below we show that for any *nonsingular* matrix M we can derive a generalized recursion (11) which involves symmetric matrices only. The necessary and sufficient conditions for the recursion (11) to yield symmetric Z_k for all k are in fact easily given in terms of Z_0 and Y *only*.

Lemma 1. *Let Y and Z_0 be nonsingular matrices. Then all matrices Z_k of the recursion* (11) *are symmetric and nonsingular if and only if Z_0 and $YZ_0^{-1}Y$ are symmetric.*

Proof. The symmetry of all Z_k clearly implies that of Z_0 and $YZ_0^{-1}Y$. We use induction, showing that Z_{k+1} and $YZ_{k+1}^{-1}Y$ are symmetric when Z_k and $YZ_k^{-1}Y$ are. This follows from the identities

$$Z_{k+1} = \frac{1}{2c_k} \cdot [Z_k + c_k^2 YZ_k^{-1}Y], \tag{28}$$

$$(YZ_{k+1}^{-1}Y)^{-1} = \frac{1}{2c_k} \cdot [(YZ_k^{-1}Y)^{-1} + c_k^2 Z_k^{-1}], \tag{29}$$

which can be derived from one another. The nonsingularity of Z_0 and Y is clearly required for the existence of the above recursions. That this is also preserved for each k follows easily from (12): the *standard* sign recursion (7) (with the matrices $Y_1^{-1}Z_kY_2^{-1}$) is known to yield invertible matrices at each step [B1], whence it is clear that the matrices Z_k are all invertible as well. ∎

In order to find a symmetric recursion of type (11) for M, we have thus only to find matrices Y_1 and Y_2 (with $Y = Y_1Y_2$) such that $Z_0 := Y_1MY_2$ and $YZ_0^{-1}Y := Y_1M^{-1}Y_2$ are both symmetric. This is always possible, since any matrix M possesses a factorization $M = S_1^{-1}S_2$ where both S_1 and S_2 are symmetric [U]; one solution is then given by $Y_1 = S_1$ and $Y_2 = I$, which yields the symmetric matrices $Z_0 = S_2$ and $YZ_0^{-1}Y = S_1S_2^{-1}S_1$. We do not claim that such matrices are *easy* to find (in general this involves an eigenvalue problem [U], but it turns out that for our

example, $M = H$, they are trivial to find as is shown in (13), (14). It is worth mentioning here, on the other hand, that if *one* symmetric form is known then it is easy to derive others by merely replacing Y_1 and Y_2 by UY_1 and Y_2U^T, respectively, where U is an arbitrary invertible matrix. This in fact amounts to a congruence transformation of the recursion (11), which of course preserves symmetry.

Remark 1. Notice that while Z_0 and $YZ_0^{-1}Y$ are required to be symmetric, no such condition is imposed on Y itself. In practice, though, if Z_0 is symmetric, it is sufficient to take Y symmetric or antisymmetric (i.e., $Y = \pm Y^T$) in order to guarantee the symmetry of $YZ_0^{-1}Y$. A simple example of this is the choice $Y = J$ in the previous section. Other possible choices of Y are functions of Z_0 and are therefore less tractable.

We now want to find *sufficient* conditions on the matrices Z_k of our recursion such that they possess a factorization $L_k\Sigma L_k^T$ with L_k lower triangular and invertible and Σ some diagonal sign matrix, possibly different from the one in (15). Such conditions are now derived using the following general Lemmas about the *inertia* $\mathrm{In}(M)$ of a symmetric matrix M (i.e., the number of its negative eigenvalues, zero eigenvalues, and positive eigenvalues), denoted in the sequel by $\mathrm{In}(M) = [n_-, n_0, n_+]$.

Lemma 2. *Let $\hat{M}$ be a principal submatrix of a symmetric matrix M, with respective inertias $[\hat{n}_-, \hat{n}_0, \hat{n}_+]$ and $[n_-, n_0, n_+]$. Then*

$$\begin{cases} n_- + n_0 \geq \hat{n}_- + \hat{n}_0, \\ n_0 + n_+ \geq \hat{n}_0 + \hat{n}_+. \end{cases} \tag{30}$$

Proof. This follows from the interlacing property of eigenvalues of symmetric matrices [GV, p. 269]. ∎

Lemma 3. *Let M be an $N \times N$ nonsingular symmetric matrix. If M has two principal submatrices of dimensions $p \times p$ and $q \times q$ with $p + q = N$, which are, respectively, negative and positive semidefinite, then $\mathrm{In}(M) = [p, 0, q]$.*

Proof. Let $\mathrm{In}(M) = [n_-, n_0, n_+]$. Then Lemma 2 implies that $n_- + n_0 \geq p$ and $n_0 + n_+ \geq q$. Since M is nonsingular we have $n_0 = 0$ which completes the proof. ∎

Remark 2. Notice that applying this result to JH and HJ where H is the matrix described in (3), we immediately find that $\mathrm{In}(JH) = \mathrm{In}(HJ) = [n, 0, n]$.

Lemma 4. *Let M be a symmetric matrix with inertia $[p, 0, q]$. Then it has a decomposition $L\Sigma L^T$ with L lower triangular and invertible and with Σ equal to*

$$\Sigma = \begin{pmatrix} -I_p & 0 \\ 0 & I_q \end{pmatrix} \tag{31}$$

if and only if the $p \times p$ leading principal submatrix M_{11} is negative definite.

Proof. From the triangularity of L it follows that $M_{11} = -L_{11} I_p L_{11}^T$, where L_{11} is the $p \times p$ leading principal submatrix of L. This thus proves necessity.

For sufficiency, we first recall that a decomposition $M = L\Sigma L^T$ with L lower triangular and invertible is known to exist if all the leading principal submatrices of M are nonsingular [GV]. (The inertias of the corresponding leading principal submatrices of M and Σ are then also equal.) The $k \times k$ leading principal submatrices $\hat{M}$ of M with $k \leq p$ are also those of M_{11} and hence are nonsingular since $M_{11} < 0$ [GV]. For the $k \times k$ leading principal submatrices $\hat{M}$ of M with $k > p$ we give a proof by contradiction. Let $\text{In}(\hat{M}) = [\hat{n}_-, \hat{n}_0, \hat{n}_+]$ for such a $k \times k$ singular submatrix $\hat{M}$. Then $\hat{n}_- + \hat{n}_0 \geq p + 1$ by Lemma 2 (since M_{11} is its submatrix). Using again Lemma 2 this would also imply that the nonsingular matrix M—with inertia $[n_-, 0, n_+]$—has $n_- \geq p + 1$ negative eigenvalues (since $\hat{M}$ is a submatrix of M). But this contradicts Lemma 3, which says that $\text{In}(M) = [p, 0, q]$.

Finally, for the sign pattern of the normalized Σ we need to prove that the inertias of the leading principal submatrices of M are also those of the leading principal submatrices of Σ as given in (31). This follows from the above arguments on inertias of $\hat{M}$. ∎

Lemma 5. *Let M be a symmetric matrix with $\text{In}(M) = [p, 0, q]$. Then the leading $p \times p$ principal submatrix of M is negative (semi)definite if and only if the $q \times q$ trailing principal submatrix of M^{-1} is positive (semi)definite.*

Proof. Let us first assume that the leading principal submatrix M_{11} of M is negative definite. Then it follows from Lemma 4 that there exists a decomposition of M of the type

$$M = \begin{pmatrix} M_{11} & M_{21}^T \\ M_{21} & M_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} -I_p & 0 \\ 0 & I_q \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{pmatrix} := L\Sigma L^T, \quad (32)$$

where L is lower triangular. Now define $N := M^{-1}$ and $U := L^{-T}$ (U is thus upper triangular). Then it follows that

$$N = \begin{pmatrix} N_{11} & N_{21}^T \\ N_{21} & N_{22} \end{pmatrix} = \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} \begin{pmatrix} -I_p & 0 \\ 0 & I_q \end{pmatrix} \begin{pmatrix} U_{11}^T & 0 \\ U_{12}^T & U_{22}^T \end{pmatrix} = U\Sigma U^T. \quad (33)$$

From $N_{22} = U_{22} U_{22}^T$ we derive that N_{22} is positive definite. To prove the converse we follow similar arguments.

For the case of semidefinite matrices we cannot use the above arguments, but the result easily follows by continuity arguments. This completes the proof. ∎

Remark 3. Notice that we can only relate properties of M_{11} and N_{22}, *not* of M_{11} and M_{22}, as is easily seen from the following example with inertia [1, 0, 1]:

$$M = \begin{pmatrix} -3 & 2 \\ 2 & -1 \end{pmatrix}, \qquad N = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}.$$

Using this result we now derive simple *sufficient* conditions (in terms of Z_0 and Y *only*) for all Z_k to have a decomposition $Z_k = L_k \Sigma L_k^T$ with Σ independent of k.

Theorem 2. *Let the $N \times N$ matrices Y and Z_0 be nonsingular and let Z_0 and $YZ_0^{-1}Y$ be symmetric. Then if Z_0 and $YZ_0^{-1}Y$ both have a leading $p \times p$ principal submatrix which is negative definite and a trailing $q \times q$ principal submatrix which is positive semidefinite (with $p + q = N$), then this property is preserved for all matrices Z_k and $YZ_k^{-1}Y$. The matrices Z_k then also possess a decomposition $L_k \Sigma L_k^T$ with L_k lower triangular and $\Sigma = \operatorname{diag}\{-I_p, I_q\}$.*

Proof. We prove the theorem by induction. Let the property hold for k. Then from (28) it readily follows that the linear combination Z_{k+1} also has a leading $p \times p$ principal submatrix which is negative definite and a trailing principal submatrix which is positive semidefinite. By Lemma 5, both Z_k^{-1} and $(YZ_k^{-1}Y)^{-1}$ have a trailing $q \times q$ principal submatrix which is positive definite and a leading principal submatrix which is negative semidefinite, and from (29) it again follows that this also holds for the linear combination $(YZ_{k+1}^{-1}Y)^{-1}$. By again applying Lemma 5, we conclude that $YZ_{k+1}^{-1}Y$ has a leading $p \times p$ principal submatrix which is negative definite and a trailing $q \times q$ principal submatrix which is positive semidefinite. Using Lemmas 3 and 4, the existence of the desired decomposition for Z_k follows. ∎

Using similar arguments we easily derive the following corollary.

Corollary 1. *Under the conditions of Theorem 2, a factorization*

$$(YZ_k^{-1}Y)^{-1} = U_k \Sigma U_k^T$$

always exists at each step k, where U_k is upper triangular.

Remark 4. The above only proves sufficiency of the conditions. It follows from Lemmas 4 and 5 that the negativity of the $p \times p$ leading principal submatrices of Z_0 and $YZ_0^{-1}Y$ is *necessary* for the existence of L_k and U_k. (The other conditions could perhaps be relaxed.) For our example from optimal control, though, all conditions are met if we take $Z_0 = JH$, $Y = J$, and hence $(YZ_0^{-1}Y)^{-1} = -HJ$ and if we make the standard assumptions $Q > 0$ and H nonsingular. Indeed we then have

$$Z_0 = \begin{pmatrix} -Q & -A^T \\ -A & BR^{-1}B^T \end{pmatrix}, \qquad YZ_0^{-1}Y = -\begin{pmatrix} BR^{-1}B^T & A \\ A^T & -Q \end{pmatrix}^{-1}, \tag{34}$$

which satisfy the conditions of Theorem 2 because of Lemma 5.

4. Symmetric recursions and Hamiltonian Pencils

The conditions of Theorem 2 on Z_0 and $YZ_0^{-1}Y$ are met when we deal with the special case of the ARE encountered in optimal control of continuous-time systems; see Remark 4. In this section we investigate how this can be extended to other Lyapunov and Riccati equations encountered in discrete-time and continuous-time systems—possibly also defined with generalized state space models. These extensions are strongly inspired by the work of Laub and coworkers [GL], [L2],

[PLS] and are based on Hamiltonian pencils, i.e., pencils $Z_H - \lambda Y_H$ for which $Z_H J Y_H^T = -Y_H J Z_H^T$.

We first recall that these extensions can be reduced to one involving a pencil of the type

$$Z_0 - \lambda Y = T(JH - \lambda J)T^T, \tag{35}$$

where H is a Hamiltonian matrix. Note that $JH - \lambda J$ is a Hamiltonian pencil, while the pencil $Z_0 - \lambda Y$ is not Hamiltonian in general. From the discussion of Section 2 it is clear that the matrices Z_k corresponding to both pencils $Z_0 - \lambda Y$ and $JH - \lambda J$ are related by the same congruence transformation T and that eventually they both lead to the sign matrix H_∞ of the underlying Hamiltonian matrix H. Moreover, the Z_k corresponding to (35) are all symmetric by application of Lemma 1.

Consider the generalized ARE encountered in continuous-time systems modeled by generalized state-space systems:

$$E^T P A + A^T P E - E^T P B R^{-1} B^T P E + Q = 0, \tag{36}$$

where E is nonsingular. In [GL] it is shown that the solution P of this equation boils down to the computation of the generalized sign recursion applied to the pencil

$$Z_0 - \lambda Y = \begin{pmatrix} -Q & -A^T \\ -A & BR^{-1}B^T \end{pmatrix} - \lambda \begin{pmatrix} 0 & E^T \\ -E & 0 \end{pmatrix}. \tag{37}$$

This pencil is congruent to a Hamiltonian pencil as in (35), using the transformation

$$T = \begin{pmatrix} I & 0 \\ 0 & E \end{pmatrix}. \tag{38}$$

Lyapunov equations of the type

$$E^T P A + A^T P E + Q = 0 \tag{39}$$

are encountered in stability analysis of generalized state space systems and can be viewed as a special case of (37) by putting $B = 0$. Other extensions considered in [GL] are the corresponding Lyapunov and Riccati equations encountered in discrete-time systems using generalized state-space models. These can both be represented by the equation (with $B = 0$ for the Lyapunov equation)

$$A^T P A - E^T P E - A^T P B (R + B^T P B)^{-1} B^T P A + Q = 0. \tag{40}$$

In [GL] it is shown that the matrix M defined as

$$M := \begin{pmatrix} I & BR^{-1}B^T \\ 0 & E^{-T}A^T \end{pmatrix}^{-1} \begin{pmatrix} AE^{-1} & 0 \\ -E^{-T}QE^{-1} & I \end{pmatrix} \tag{41}$$

is *symplectic* (i.e., $MJM^T = J$) and that the matrix H defined as

$$H := (M + I)^{-1}(M - I) \tag{42}$$

is Hamiltonian (i.e., $HJ = -JH^T$). Here a classical assumption is that M has no eigenvalues on the unit circle, since by (42) they are mapped to eigenvalues of H located on the imaginary axis. The solution P of (40) can thus be found from the

generalized sign recursion applied to the pencil

$$Z_H - \lambda Y_H := J(M + I)^{-1}(M - I) - \lambda J. \tag{43}$$

In order to simplify this expression in terms of the matrices occurring in (40), we first use the congruence transformation $T_1 = (M + I)J^T$ to obtain the following pencil:

$$T_1(Z_H - \lambda Y_H)T_1^T = (M - I)J(M + I)^T - \lambda(M + I)J(M + I)^T \tag{44}$$

to which the generalized sign recursion can also be applied. Then a second congruence transformation

$$T_2 := \begin{pmatrix} I & BR^{-1}B^T \\ 0 & A^T \end{pmatrix} \tag{45}$$

further reduces this pencil to the more tractable form

$$\begin{aligned} Z_0 - \lambda Y &:= T_2 T_1 (Z_H - \lambda Y_H) T_1^T T_2^T \\ &= \begin{pmatrix} A - E & -BR^{-1}B^T \\ -Q & E^T - A^T \end{pmatrix} \begin{pmatrix} 0 & -E^T \\ E & 0 \end{pmatrix}^{-1} \begin{pmatrix} A + E & BR^{-1}B^T \\ -Q & E^T + A^T \end{pmatrix}^T \\ &\quad - \lambda \begin{pmatrix} A + E & BR^{-1}B^T \\ -Q & E^T + A^T \end{pmatrix} \begin{pmatrix} 0 & -E^T \\ E & 0 \end{pmatrix}^{-1} \begin{pmatrix} A + E & BR^{-1}B^T \\ -Q & E^T + A^T \end{pmatrix}^T. \end{aligned} \tag{46}$$

We now prove that pencils $Z_0 - \lambda Y$ such as defined in (37) and (46) can always be used as a starting point for a factorized version of the generalized sign recursion. For this we prove the following theorem.

Theorem 3. *Let $Z_0 - \lambda Y$ be a pencil congruent to the Hamiltonian pencil $JH - \lambda J$ (i.e., $Z_0 - \lambda Y = T(JH - \lambda J)T^T$ for some T). Then there always exists an updating congruence $\hat{T}$ such that $\hat{T}(Z_0 - \lambda Y)\hat{T}^T$ satisfies the conditions of Theorem 2, i.e., both $\hat{T}Z_0\hat{T}^T$ and $\hat{T}YZ_0^{-1}Y\hat{T}^T$ have an $n \times n$ leading principal submatrix which is negative definite and an $n \times n$ trailing principal submatrix which is positive semidefinite.*

Proof. We prove the result by reducing it to simpler ones in successive steps. First, it is clear that it suffices to prove it for the pencil $JH - \lambda J$ itself, since the congruence transformation T^{-1} can always be absorbed in $\hat{T}$. Secondly, it suffices to prove it for the pencil

$$JH_1 - \lambda J := T_1^T(JH - \lambda J)T_1 = \begin{pmatrix} 0 & -A_S^T \\ -A_S & -X \end{pmatrix} - \lambda J, \tag{47}$$

where T_1 is the *symplectic unitary* matrix reducing H to its Schur form with A_S stable (see [PV]), i.e.,

$$HT_1 = T_1 \begin{pmatrix} A_S & X \\ 0 & -A_S^T \end{pmatrix} \quad \text{with} \quad T_1^T J T_1 = J. \tag{48}$$

This follows again from the fact that T_1 can also be absorbed in $\hat{T}$. For (48) we show

below that there is an updating *symplectic* transformation T_2

$$T_2 := \begin{pmatrix} P & -I_n \\ I_n & 0 \end{pmatrix} \quad \text{with} \quad T_2^T J T_2 = J, \tag{49}$$

such that

$$JH_2 = T_2^T J H_1 T_2 = \begin{pmatrix} -I_n & A_S \\ A_S^T & 0 \end{pmatrix}. \tag{50}$$

If this holds then both JH_2 and

$$J(JH_2)^{-1}J = -(H_2 J)^{-1} := \begin{pmatrix} 0 & A_S^T \\ A_s & I_n \end{pmatrix}^{-1} = \begin{pmatrix} -A_S^{-1}A_S^{-T} & A_S^{-1} \\ A_S^{-T} & 0 \end{pmatrix} \tag{51}$$

have an $n \times n$ leading principal submatrix which is negative definite. A T_2 satisfying (49) is in fact obtained from the solution P of the Lyapunov equation

$$PA_S^T + A_S P = I - X. \tag{52}$$

Since A_S is stable, (52) always has a symmetric solution P. ■

Note that this theorem only proves the existence of a desired updating congruence transformation $\hat{T}$, albeit by constructing it. This construction is indeed more involved than the actual generalized sign recursion since it uses the Schur form of H and hence implicitly solves the underlying Riccati equation. There is definitely a need for a simpler construction of an updating transformation $\hat{T}$ such that $\hat{T}(Z_0 - \lambda Y)\hat{T}^T$ satisfies the conditions of Theorem 2. It was shown at the end of Section 3 that for the standard algebraic Riccati equation (2) with $Q > 0$ such an updating transformation is superfluous but for the extensions considered in this section, $\hat{T}$ is crucial for the existence of factored recursions.

5. Parallelism Enhancement

When developing factored versions of a certain matrix recursion, we are usually thinking of lowering the complexity of the recursion or improving its numerical performance. In our case we are also guided by the possibility of improving the inherent parallelism of the resulting version. In this section we use that to compare the variants of the factored recursions described earlier. We first recall these two variants as applied to pencils of the type $Z_0 - \lambda Y$ with Z_0 and $YZ_0^{-1}Y$ symmetric and where Z_0 and Y satisfy the conditions of Theorem 2. In this case the matrices Z_k and $YZ_k^{-1}Y$ can be factorized for all k as, respectively, $Z_k = L_k \Sigma L_k^T$ and $Y^{-1}Z_k Y^{-1} = U_k \Sigma U_k^T$ with L_k lower triangular, U_k upper triangular, and Σ as in (15). The following recurrences for the factors L_k and U_k, involving skew-unitary matrices R_k, S_k, and T_k, can then be derived (see Section 2):

$$\frac{1}{\sqrt{2c_k}}(L_k \quad c_k Y L_k^{-T})R_k = (L_{k+1} \quad 0) \tag{53}$$

involving L_k only, if Y is antisymmetric (i.e., $Y = -Y^T$), and

$$\frac{1}{\sqrt{2c_k}}(L_k \quad c_k U_k^{-T})S_k = (L_{k+1} \quad 0), \tag{54}$$

$$\frac{1}{\sqrt{2c_k}} T_k^T \begin{pmatrix} U_k^T \\ c_k L_k^{-1} \end{pmatrix} = \begin{pmatrix} U_{k+1}^T \\ 0 \end{pmatrix} \tag{55}$$

involving both L_k and U_k, in the general case.

When implementing the factored version (53), we perform at each step the triangularization of a $2n \times 4n$ matrix with the following structure (with $n = 3$ and where x denotes a nonzero element):

$$\begin{pmatrix} x & & & & & & x & x & x & x & x & x \\ x & x & & & & & x & x & x & x & x & x \\ x & x & x & & & & x & x & x & x & x & x \\ x & x & x & x & & & x & x & x & x & x & x \\ x & x & x & x & x & & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x & x \end{pmatrix}.$$

The computational complexity of this reduction is $16n^3$ flops[1] when using skew Givens rotations and $8n^3$ when using skew Householder or modified skew Givens transformations (see the next section for an explanation of the latter). We also have to add to that the $4n^3$ flops needed to form YL_k^{-T}. The total complexity of one iteration step is thus $20n^3$ or $12n^3$ depending on the choice of transformations.

When specializing to the case $Z_0 = JH$, $Y = J$ discussed earlier, things become cheaper since then the $2n \times 4n$ matrices involved look like

$$\begin{pmatrix} x & & & & & & & & & x & x & x \\ x & x & & & & & & & & & x & x \\ x & x & x & & & & & & & & & x \\ x & x & x & x & & & x & x & x & x & x & x \\ x & x & x & x & x & & & x & x & x & x & x \\ x & x & x & x & x & x & & & x & x & x & x \end{pmatrix}.$$

Whether using skew Givens or skew Householder transformations, it is not possible to avoid partial fill in this triangularization, and in fact it is as if we were treating a $2n \times 4n$ matrix with the following structure:

$$\begin{pmatrix} x & & & & & & & & & x & x & x \\ x & x & & & & & & & & x & x & x \\ x & x & x & & & & & & & x & x & x \\ x & x & x & x & & & x & x & x & x & x & x \\ x & x & x & x & x & & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x & x \end{pmatrix}.$$

[1] One "flop," or floating-point operation, is the work needed to execute $y := a * x + b$ where a, b, and x are real scalars.

The computational complexity of this reduction is $10n^3$ when using skew Givens rotations and $5n^3$ when using skew Householders or modified skew Givens. Adding to this the $\frac{4}{3}n^3$ flops needed to construct JL^{-T} we come to a total of $11\frac{1}{3}n^3$ or $6\frac{1}{3}n^3$ flops, depending on the choice of transformations.

When, on the other hand, we use the combined recursions (54), (55), then the $2n \times 4n$ matrices involved are of the form

$$\begin{pmatrix} x & & & & & & x & & & & & \\ x & x & & & & & x & x & & & & \\ x & x & x & & & & x & x & x & & & \\ x & x & x & x & & & x & x & x & x & & \\ x & x & x & x & x & & x & x & x & x & x & \\ x & x & x & x & x & x & x & x & x & x & x & x \end{pmatrix}$$

and

$$\begin{bmatrix} x & & & & & & \\ x & x & & & & & \\ x & x & x & & & & \\ x & x & x & x & & & \\ x & x & x & x & x & & \\ x & x & x & x & x & x & \\ x & & & & & & \\ x & x & & & & & \\ x & x & x & & & & \\ x & x & x & x & & & \\ x & x & x & x & x & & \\ x & x & x & x & x & x & \end{bmatrix}.$$

Here we have thus to perform two triangularizations and two inversions which together require $13\frac{1}{3}n^3$ and $8n^3$ flops depending on the choice of transformations.

For a sequential machine we would of course choose skew Householder transformations because of their speed. In the case $Y = J$ the fastest factored form is (53) (which takes only $6\frac{1}{3}n^3$ flops per iteration) but in fact this is still slower than the nonfactored forms (13) or (14). These indeed require only $4n^3$ flops per iteration when symmetry is exploited. In the case of a general Y, the most interesting factored recursion is (54), (55), which requires $8n^3$ flops per iteration and is thereby even faster than the nonfactored form (11). The latter requires $10\frac{2}{3}n^3$ flops even when exploiting symmetry.

When using parallel machines, completely different criteria have to be considered. Preference goes then to the use of Givens-like rotations, and we consider not operation counts but "time steps." This is explained in the next section.

6. Mapping on a Triangular Array of Processors

In this section we deal with the above algorithms implemented on "systolic-like" arrays [K], [KL]. The aim is not to build up the "best" parallel algorithm, but to

show that an execution time proportional to the problem size can be achieved on simple array processors which exhibit fine grain parallelism. Since the combined factored recursions (54), (55) seem to be the most appealing, we focus here on those. However, parts of the analysis are also applicable to the single factored recursion (53). Finally, we assume that the coefficients c_k are all equal to 1, for simplicity. The algorithm considered in this section is thus:

1. Initialization. LDL^T decomposition of Z_0 and UDU^T decomposition of $Y^{-1}Z_0Y^{-1}$:

$$Z_0 = L_0 \Sigma L_0^T \quad \text{and} \quad Y^{-1}Z_0Y^{-1} = U_0 \Sigma U_0^T.$$

2. Iteration. For $k = 0, 1, \ldots$:
   (a) Invert L_k and U_k^T.
   (b) Determine L_{k+1} by the triangularization (54) and U_{k+1}^T by the triangularization (55).
   Stop the iteration when converged or after a fixed number of steps.
3. Solve for P.

The core of the above algorithm is the iteration process. Due to the lower triangular structure of the matrices to be inverted and of the blocks of those to be triangularized, it is natural to look for an implementation of that iteration on a corresponding lower triangular two-dimensional array consisting of $N(N+1)/2$ processors p_{ij} $(i = 1, \ldots, N; j = 1, \ldots, i)$. As sketched in Fig. 1, each processor is assumed to be connected (at most) to its four nearest neighbors: north, east, south, and west. Similar implementations are often suggested for classical matrix problems, e.g., [L3] and [L4]. We take into account the constraint that systolic algorithms are to be compute-bound, for technological reasons [HB, p. 769]: any pair of neighboring processors will be allowed to exchange only a limited number (two in our case) of data at a given time.

In the hope of clarifying the exposition, inversion and triangularization are first handled separately and a way to link (and overlap) them together is presented afterward. Then, the realization of the initial and solution stages of the algorithm on the same array is discussed.

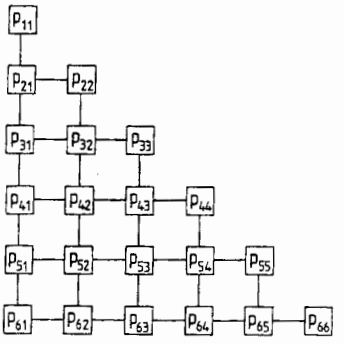

Fig. 1. The array of processors taken as a reference in the text, here shown for $N = 6$.

6.1. *Inversion*

Examining the simultaneous inversion of L_k and U_k^T, we assume that at the starting time ($t = 0$), elements (i, j) of L_k and U_k^T are in processor p_{ij}. In order to simplify notation we drop the subscript k referring to the recursion step and only consider explicitly the inversion of L_k. The inversion of U_k^T follows exactly the same process. In Fig. 2 the systolic inversion of $L \equiv (l_{ij})$ is detailed. At most one number is transmitted on a given branch per time step. With the notation $L^{-1} \equiv (x_{ij})$, the

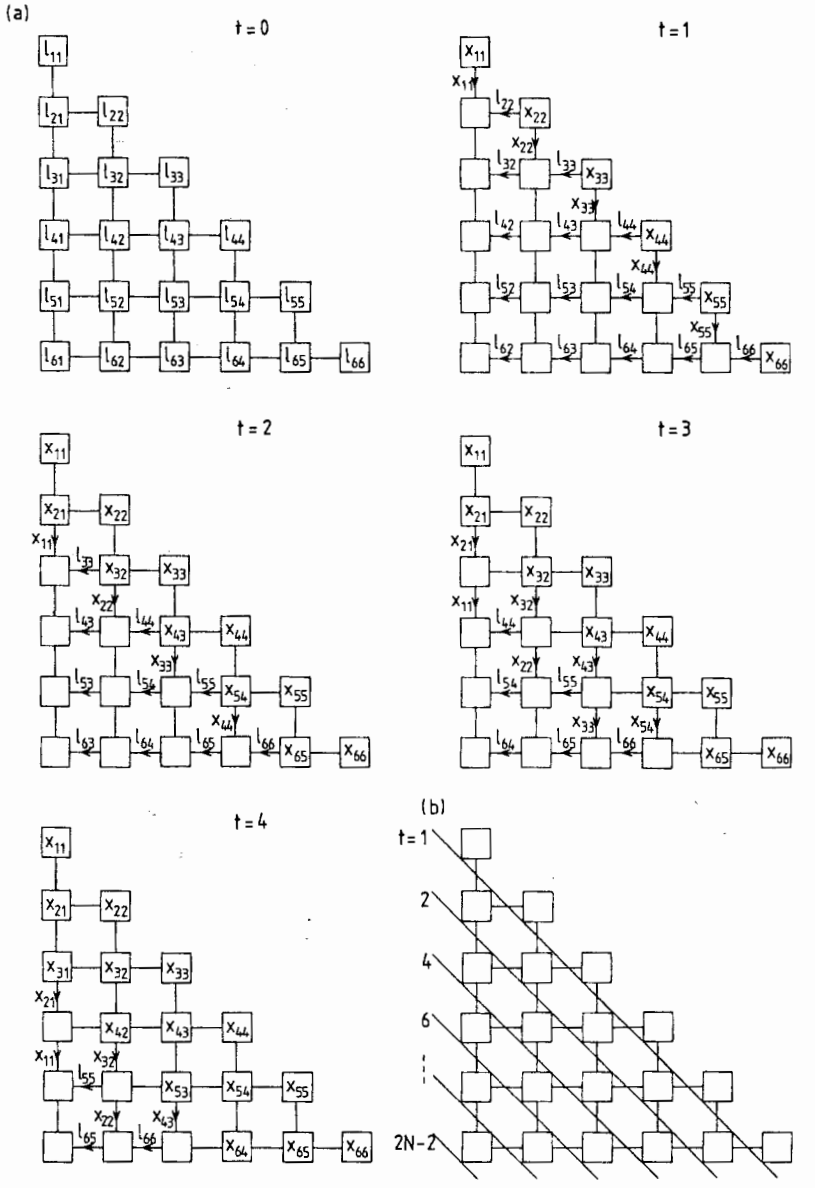

Fig. 2. Inversion of $L \equiv (l_{ij})$ on the array of Fig. 1, giving $L^{-1} \equiv (x_{ij})$, for $N = 6$.

triangular array computes

$$x_{ii} = \frac{1}{l_{ii}},$$
$$x_{ij} = -\frac{1}{l_{ii}} \sum_{m=j}^{i-1} l_{im} x_{mj} \qquad \text{if} \quad i > j, \tag{56}$$

leaving x_{ij} in p_{ij} as follows. At time 1, all the diagonal elements x_{ii} are computed and their value is communicated to the south processor; at the same time all the l_{ij}'s are shifted one processor to the west. At time 2, the subdiagonal $x_{i+1,i}$'s are obtained, while the x_{ii}'s and the l_{ij}'s are propagated one processor further along, respectively, the columns and the rows of the array. At time 3, no new x_{ij} are computed: the values of $x_{i+1,i}$ first have to be passed to the processors $p_{i+2,i}$, due to the communication constraint; propagation still goes on in rows and columns. At time 4, the $x_{i+2,i}$'s can be computed. The rest of the inversion is then a succession of steps similar to the two previous ones; propagation and propagation plus computation of a subdiagonal of x_{ij} alternate until $x_{N,1}$ is obtained in the processor $p_{N,1}$. In Fig. 2(b) this is summarized by an indication of the time at which the x_{ij}'s are obtained. In this way, the inversion of L and U^T is carried out in $2N-2$ time steps. It is also apparent that, prior to the computation of x_{ij}, $(i-j)$ values are stored in the processor p_{ij}. This possibly imposes excessive needs for local memory, unless other propagation patterns than those of Fig. 2 can be considered. We do not discuss that here, confining ourselves to basic principles, except for a remark in Subsection 6.3.

6.2. *Triangularization*

We now deal with the triangularizations (54) and (55). They will be performed simultaneously on the triangular array of Fig. 1. As they differ mainly by the fact that (54) is achieved by means of column transformations and (55) by means of row transformations, the following description focuses on one of them, say (54). Basically, the method proposed here is characterized by the use of "elementary" operations acting "locally" on the matrices to be triangularized, in a way similar to the systolic algorithms developed for classical matrix computations; to a certain extent, this can be related to the systolic implementation of the QR-decomposition by Luk [L3], [L4].

The $\begin{pmatrix} \Sigma & 0 \\ 0 & \Sigma \end{pmatrix}$-unitary matrix in (54) is seen as the product of $\begin{pmatrix} \Sigma & 0 \\ 0 & \Sigma \end{pmatrix}$-unitary rank-two corrections to the identity I_{2N}, hereafter denoted R_{ij} $(i = 1, \ldots, N; j = 1, \ldots, i)$, of the form

$$R_{ij} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & r_{ii} & \cdots & r_{ij} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & r_{ji} & \cdots & r_{jj} & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow i \\ \\ \leftarrow j+N. \\ \\ \\ \end{matrix} \tag{57}$$
$$\qquad\qquad\qquad\quad \uparrow \qquad\qquad \uparrow$$
$$\qquad\qquad\qquad\quad i \qquad j+N$$

R_{ij} modifies the ith column of L_k (possibly transformed already) and the jth column of U_k^{-T} (same remark), so that the element (i, j) of the current U_k^{-T} is annihilated. Those annihilations will be performed as follows. A first "wave" W_1 of transformations set the diagonal elements of U_k^{-T} to zero: $W_1 = R_{11}R_{22}\cdots R_{NN}$. Then the first subdiagonal of U_k^{-T}, already modified by W_1, is affected by a second wave $W_2 = R_{21}R_{32}\cdots R_{N,N-1}$. Other waves succeed one another until the last element, in position $(N, 1)$, is annihilated by the last wave $W_N = R_{N1}$. The waves W_i have to be applied in order, for increasing i (more generally, the element (i, j) is to be annihilated prior to any other with larger index i and same index j). But in a given wave the R_{ij}'s can be ordered arbitrarily; in particular, they may be applied in parallel.

Before detailing the parallel implementation of that process, let us consider more precisely the form of the R_{ij}'s as well as the question of their existence. It is useful to write the $N/2 \times N/2$ blocks in the triangularization:

$$\begin{matrix} \begin{pmatrix} (L_k)_{11} & 0 & (U_k^{-T})_{11} & 0 \\ (L_k)_{21} & (L_k)_{22} & (U_k^{-T})_{21} & (U_k^{-T})_{22} \end{pmatrix} \rightarrow \begin{pmatrix} (L_{k+1})_{11} & 0 \\ (L_{k+1})_{21} & (L_{k+1})_{22} \end{pmatrix}. \\ \textit{labels:} \quad -1 \quad +1 \quad -1 \quad +1 \end{matrix} \tag{58}$$

"Labels" are added here to mark the columns of L_k and U_k^{-T} according to the diagonal elements of $\begin{pmatrix} \Sigma & 0 \\ 0 & \Sigma \end{pmatrix}$. There are two types of R_{ij}'s. First, if R_{ij} modifies columns with the same label, then $\begin{pmatrix} r_{ii} & r_{ij} \\ r_{ji} & r_{jj} \end{pmatrix}$ is unitary, and we can choose

$$\begin{pmatrix} r_{ii} & r_{ij} \\ r_{ji} & r_{jj} \end{pmatrix} = \begin{pmatrix} \cos\theta_{ij} & -\sin\theta_{ij} \\ \sin\theta_{ij} & \cos\theta_{ij} \end{pmatrix},$$

where θ_{ij} is determined such that

$$\tan\theta_{ij} = \frac{[(L_k \quad U_k^{-T})W_1 \cdots W_{i-j}](i, j+N)}{[(L_k \quad U_k^{-T})W_1 \cdots W_{i-j}](i, i)}.$$

Such transformations always exist and are numerically stable [W]. The second type occurs when the labels are different, i.e., when zeroing the block (2, 1) of U_k^{-T}; therefore $\begin{pmatrix} r_{ii} & r_{ij} \\ r_{ji} & r_{jj} \end{pmatrix}$ is $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$-unitary, which leads to

$$\begin{pmatrix} r_{ii} & r_{ij} \\ r_{ji} & r_{jj} \end{pmatrix} = \begin{pmatrix} \sec\theta_{ij} & -\tan\theta_{ij} \\ -\tan\theta_{ij} & \sec\theta_{ij} \end{pmatrix},$$

where θ_{ij} is such that

$$\sin\theta_{ij} = \frac{[(L_k \quad U_k^{-T})W_1 \cdots W_{i-j}](i, j+N)}{[(L_k \quad U_k^{-T})W_1 \cdots W_{i-j}](i, i)}.$$

This expression can be greater than 1, in general. We now show that this is never the case here, and thus that the associated R_{ij}'s always exist. Let us consider for a moment the situation where the diagonal blocks of U_k^{-T} would have been

eliminated by "first type" R_{ij}'s, denoting it as

$$\begin{pmatrix} L_{11} & 0 & 0 & 0 \\ L_{21} & L_{22} & X & 0 \end{pmatrix} \to \begin{pmatrix} (L_{k+1})_{11} & 0 \\ (L_{k+1})_{21} & (L_{k+1})_{22} \end{pmatrix}.$$

The blocks L_{11} and L_{21} will no longer be modified. Annihilation of X results from transformations which involve only L_{22}. Because of (24), we can write

$$\tfrac{1}{2}\begin{pmatrix} -L_{11}L_{11}^T & -L_{11}L_{21}^T \\ -L_{21}L_{11}^T & -L_{21}L_{21}^T + L_{22}L_{22}^T - X^TX \end{pmatrix}$$
$$= \begin{pmatrix} -(L_{k+1})_{11}(L_{k+1})_{11}^T & -(L_{k+1})_{11}(L_{k+1})_{21}^T \\ -(L_{k+1})_{21}(L_{k+1})_{11}^T & -(L_{k+1})_{21}(L_{k+1})_{21}^T + (L_{k+1})_{22}(L_{k+1})_{22}^T \end{pmatrix}.$$

It then follows that $\frac{1}{2}L_{21}L_{21}^T = (L_{k+1})_{21}(L_{k+1})_{21}^T$, from which we deduce that $L_{22}L_{22}^T - X^TX$ is positive definite. With x_i the ith row of X, $L_{22}L_{22}^T - x_i^Tx_i$ is positive definite for any value of i. This implies that the R_{ij}'s of the "second type" exist, as has been shown in the study of algorithms for downdating the Cholesky factorization of a positive definite matrix [GS], [LH]. Moreover, those transformations are stable provided that care is taken with computational ordering (see [BBVH]).

A systolic realization of the triangularization (54) is suggested in Fig. 3. At the starting time ($t = 0$), the elements (i, j) of $L_k \equiv (l_{ij})$ and $U_k^{-T} \equiv (x_{ij})$ are assumed to be in the processor p_{ij}. For convenience, l_{ij} and x_{ij} represent initial as well as modified values. At the end of the process, p_{ij} contains the element (i, j) of $L_{k+1} \equiv (l_{ij}^+)$. No more than one item of data is exchanged per branch per time step. The waves $W_i = \prod_{j=i}^{N} R_{j,j-i+1}$ are successively initiated in the main diagonal processors at odd time steps, and then propagated down the columns of processors. For instance, at time 1, W_1 starts in the processors p_{ii} $(i = 1, \ldots, N)$, each determining the angle θ_{ii}, annihilating x_{ii}, and passing it to the south processor (see Fig. 3(a)). At time 2, the θ_{ii}'s are used to compute modified values of the $x_{i+1,i}$'s and of the $l_{i+1,i}$'s, and are passed further on; also, these new $x_{i+1,i}$ are shifted one processor to the east. This is repeated at time 3 in the next diagonal of processors, while W_2 is started in the main diagonal. And so it continues. The l_{ij}'s are stationary. The shifting to the right of the elements x_{ij} follows the rhythm of annihilation of the diagonals of U_k^{-T}, leaving l_{ij}^+ in p_{ij} from the northwest corner of the array. The times of obtaining the l_{ij}^+'s are summarized in Fig. 3(b).

A similar process can be used for the triangularization (55). In this case, elementary transformations are propagated from the main diagonal along the rows of processors. At $t = 0$, p_{ij} contains the elements (i, j) of U_k^T and L_k^{-1}; the element (i, j) of U_{k+1}^T stays in p_{ij}. U_{k+1}^T is produced in the array from the southeast corner in a timing similar to that of Fig. 3(b).

As a consequence, the triangularizations (54) and (55) are carried out in $2N - 1$ time steps.

6.3. *Merging*

The previous systolic inversions and triangularizations are now linked together to form one iteration step of the matrix sign algorithm. The implementation of such

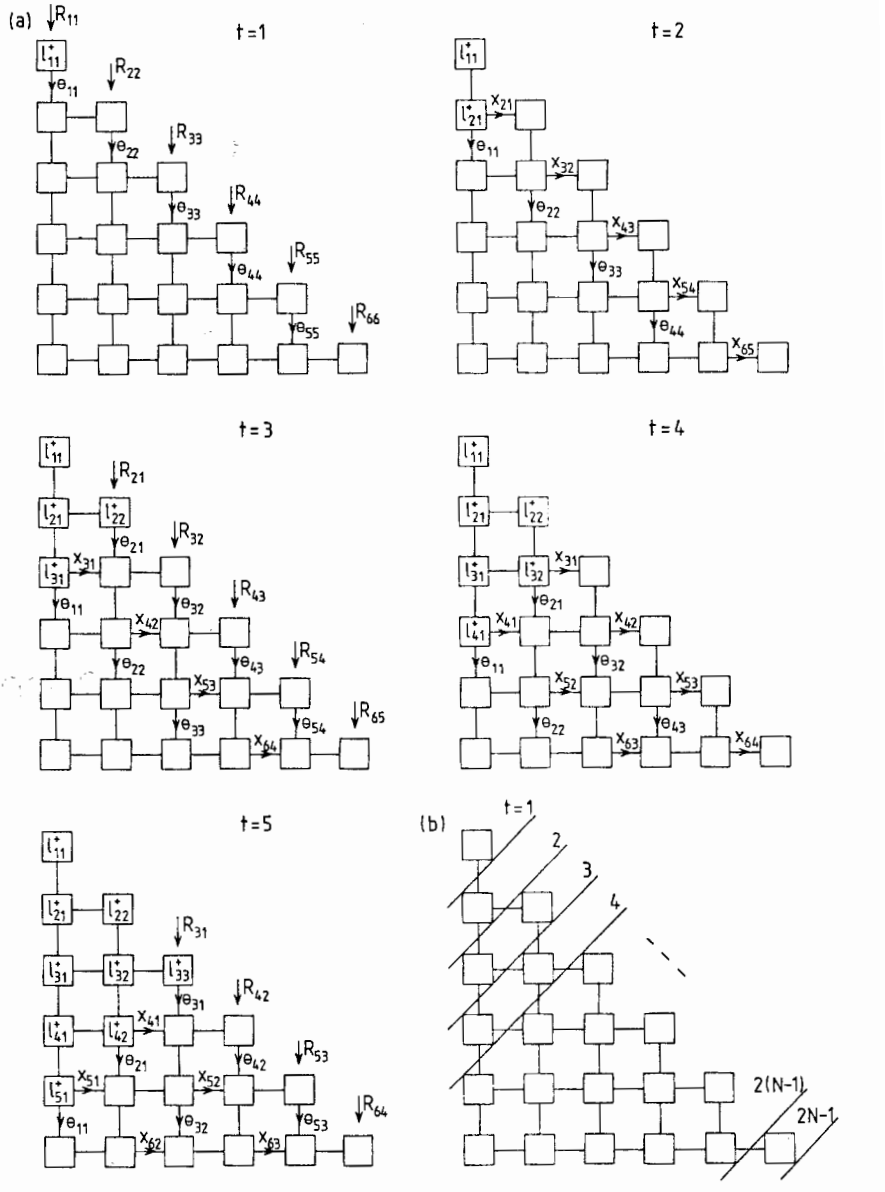

Fig. 3. Triangularization (54) with $L_{k+1} \equiv (l_{ij}^{+})$ and $U_k^{-T} \equiv (x_{ij})$ (initial or modified values), on the array of Fig. 1 for $N = 6$.

a step is illustrated in Fig. 4. Here we only show the times at which we obtain the elements of the inverses (continuous lines) and of the triangular factors (large-dashed lines for the triangularization (54), small-dashed lines for (55)), in the same manner as in Fig. 2(b) and 3(b). It is observed that the elements of L_k and U_k^T, present in the processors at $t = 0$, are replaced by the corresponding elements of L_{k+1} and U_{k+1}^T at $t = 3N - 2$, which is thus the time needed to complete one step of iteration on the triangular array.

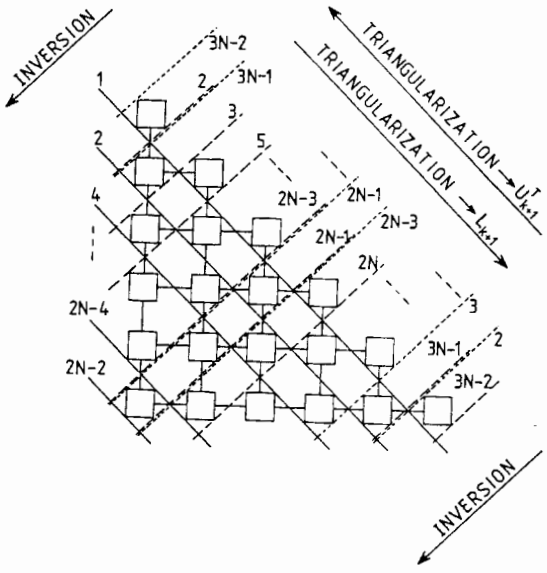

Fig. 4. One iteration of the matrix sign algorithm.

The scheme of Fig. 4 involves local memory, as stressed above in the examination of the inversion phase. In this respect, it is worthwhile to mention an alternative scheme which consists of representing one iteration step as triangularizations (Fig. 3) *followed* by inversions (this is not illustrated here, for conciseness). We find that the total time is unchanged and that the data propagation is such that the two operands of the products in (56) reach the corresponding processor simultaneously; only the intermediate result is thus to be stored in that processor until the arrival of the next operands. However, the implementation is less "natural" and implies, for instance, that the first inversion would be merged with the initial decompositions.

6.4. *Initial Decompositions*

Before executing the iteration, it is first necessary to determine the LDL^T (resp. UDU^T) decomposition of Z_0 (resp. $(Y^{-1}Z_0\,Y^{-1})$). Here again, due to the similarities of both computations, the description is restricted to one of them, say $Z_0 = L_0 \Sigma L_0^T$. Let $Z_0 \equiv (x_{ij})$ and $L_0 \equiv (l_{ij})$. The equations to be inverted are

$$\begin{aligned} x_{ij} &= -\sum_{k=1}^{j} l_{ik} l_{jk} \qquad \text{if} \quad j \leq N/2. \\ x_{ij} &= -\sum_{k=1}^{N/2} l_{ik} l_{jk} + \sum_{k=N/2+1}^{j} l_{ik} l_{jk} \qquad \text{if} \quad j > N/2. \end{aligned} \tag{59}$$

Systolic algorithms have been proposed for closely related factorizations. For instance, Brent and Luk [BL] dealt with the Cholesky decomposition of a positive definite matrix on a hex-connected array of processors where data enter and leave in a similar rhythmic fashion. Here, instead, we want to solve (59) on the array of Fig. 1, in such a way that l_{ij} stays in p_{ij} at the outcome. A possible algorithm,

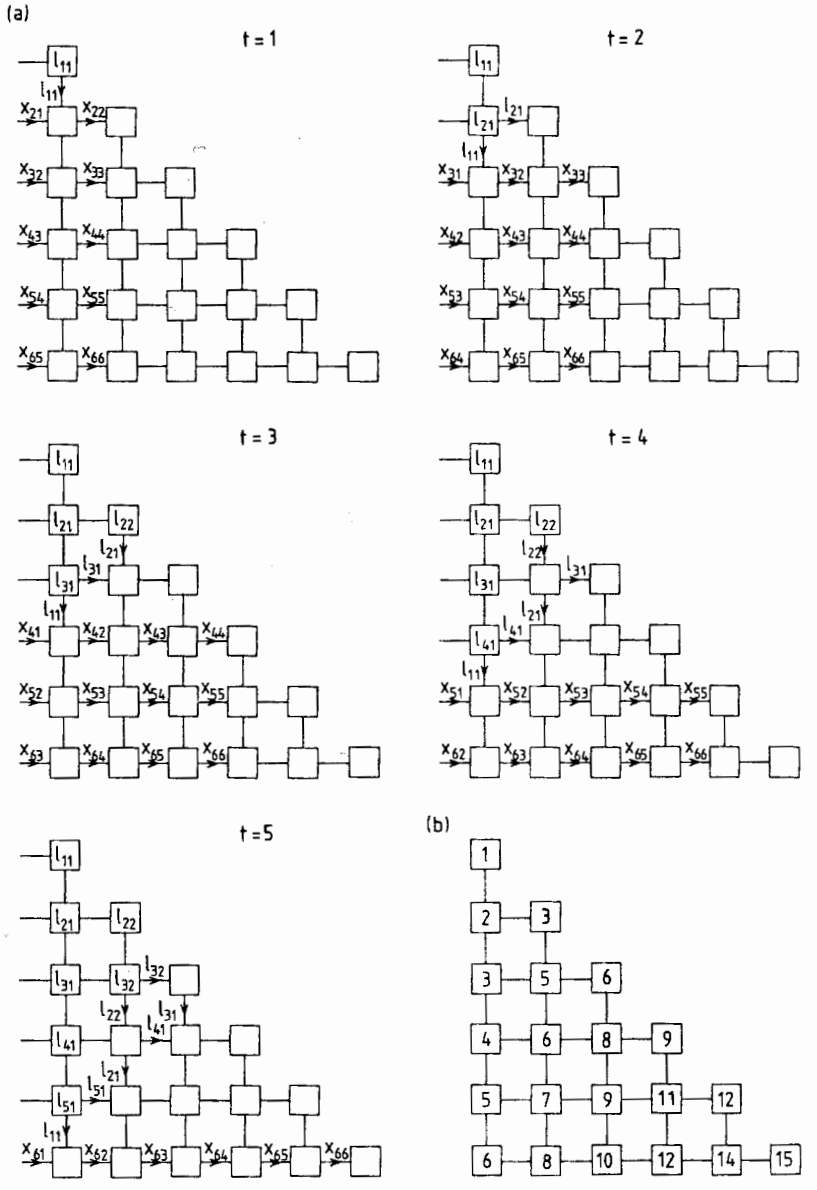

Fig. 5. $L_0 \Sigma L_0^T$ decomposition of Z_0, with $Z_0 \equiv (x_{ij})$ and $L_0 \equiv (l_{ij})$, on the array of Fig. 1 ($N = 6$).

directly related to the study of O'Leary and Stewart [OS], is shown in Fig. 5, where at most one item of data is exchanged per branch per time step.

Given the lower (or upper) part of Z_0, the factor L_0 is obtained as follows (Fig. 5(a)). The diagonals of Z_0 are fed into the array at its western boundary, the main diagonal first (i.e., p_{i1} receives x_{ii} at time $t = 1$). Then they are transmitted along the rows of processors. The element l_{ij} is computed in p_{ij}, propagated in the ith

row up to the diagonal processor, then down the ith column of processors. Product operands reach appropriate processors simultaneously; thus only intermediate results need to be stored. Times at which the l_{ij}'s are obtained are summarized in Fig. 5(b).

The other decomposition can progress jointly. The diagonals of the lower (or upper) part of $(Y^{-1}Z_0Y^{-1})$ enter the array along the bottom boundary, main diagonal first (i.e., p_{Ni} receives the element (i, i) at $t = 1$), so that the element (i, j) of U_0^T stays in p_{ij} at the outcome. Therefore, both decompositions are achieved, or equivalently the initial state of the iteration phase is attained, in $3(N-1)$ time steps.

6.5. *Solution*

We now assume that the iteration process has converged; the question of a stopping criterion is briefly considered at the end of this section. Only the extraction of the solution P is still to be made systolic. A simple and neat implementation is not apparent, due to the form of P (10). It is not our aim to suggest a definitive scheme, nor even to recommend a convenient one, but we outline here a possible scheme in order to estimate the number of time steps required.

A priori, the inverse of P is easier to compute than P itself. Upon convergence, the recursions become

$$JH_\infty = \tfrac{1}{2}(L_\infty \Sigma L_\infty^T + U_\infty^{-T}\Sigma U_\infty^{-1}) = L_\infty \Sigma L_\infty^T,$$

$$-H_\infty J = \tfrac{1}{2}(U_\infty \Sigma U_\infty^T + L_\infty^{-T}\Sigma L_\infty^{-1}) = U_\infty \Sigma U_\infty^T,$$

from which $L_\infty = U_\infty^{-T}$ follows. Denoting the triangular matrices L_∞ and U_∞^T by

$$\begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} U_{11}^T & 0 \\ U_{12}^T & U_{22}^T \end{pmatrix}, \tag{60}$$

we thus have $L_{11} = U_{11}^{-T}$ and $L_{22} = U_{22}^{-T}$. Expressions for P are obtained from (10):

$$P = (-L_{11}L_{21}^T + I)^{-1}L_{11}L_{11}^T = (U_{22}U_{12}^T + I)^{-1}U_{22}U_{22}^T.$$

Therefore, the inverse of P is given by

$$P^{-1} = L_{22}(U_{12}^T + L_{22}^T) = (U_{12} + L_{22})L_{22}^T = U_{11}(U_{11}^T - L_{21}^T) = (U_{11} - L_{21})U_{11}^T, \tag{61}$$

where the second and the fourth equalities are due to the symmetry of P. With the additional notation $Pr_{11}, Pr_{21}, Pr_{22}$, for the blocks of processors corresponding to the partitions (60), the following operations determine P^{-1} (or, more precisely, the (lower) Cholesky factor of P^{-1}) and, if necessary, P (i.e., the (upper) Cholesky factor of P), from the first expression for P^{-1} in (61):

1. With U_{12}^T distributed over Pr_{21} and L_{22} over Pr_{22}, the sum $(U_{12}^T + L_{22}^T)$ is easily realized by simultaneously sliding the contents of each column i of Pr_{22} (row i of L_{22}^T) in row i of Pr_{21}, up to the diagonal processor. This requires $N/2$ time steps.
2. Starting with $(U_{12}^T + L_{22}^T)$ and L_{22}^T superposed in Pr_{21}, the upper part of the

symmetric product $L_{22}(U_{12}^T + L_{22}^T)$ can be obtained by shifting L_{22}^T to the right and propagating down the products of its elements by those of $(U_{12}^T + L_{22}^T)$. This is accomplished in $N/2$ time steps provided that the ith row of L_{22}^T shifts one time step ahead of the $(i+1)$st row, and that partial sums in scalar products are accumulated in each processor before being passed to the next one in the same column. Then the upper part of Pr_{21} contains the upper part of P^{-1}. To some extent, this process can be seen as a generalization of the algorithm proposed in [KL] for matrix–vector multiplication.

3. The lower Cholesky triangle of P^{-1} is derived in a manner similar to the one adopted for the initial decompositions (see Subsection 6.4 above and Fig. 5). Here computations are performed in Pr_{22} (rather than in the whole array) and hence require $3[(N/2) - 1]$ time steps (for $N > 2$); in addition, $N/2$ time steps are necessary for moving each column i (with diagonal element first) of the upper part of P^{-1} up to the left end of the ith row of Pr_{22}.
4. The inversion of the last result on Pr_{22}, by analogy with the process of Fig. 2, requires $2[(N/2) - 1]$ time steps (for $N > 2$), leaving in Pr_{22} the transposed upper Cholesky factor of P.

To sum up:

(i) P^{-1} can be computed in N time steps,
(ii) its lower Cholesky factor in $3(N-1)$ time steps,
(iii) and the upper Cholesky factor of P in $4N - 5$ time steps.

The whole algorithm then takes $(7N - 8) + x \cdot (3N - 2)$ time steps if x iterations are needed for convergence. (x is typically less than 10 for matrices of reasonable size.) The execution time on a parallel machine with $O(N^2)$ processors is thus essentially *linear* in N, which corresponds to an optimal speedup.

A simple stopping criterion would be to execute a predetermined number of iterations (depending on empirical knowledge). A more sophisticated and satisfactory criterion consists of checking the following property. Upon convergence, it was indicated above (Subsection 6.5) that $L_\infty = U_\infty^{-T}$. In terms of waves (as introduced in Subsection 6.2), each triangularization is then equivalent to a first wave

$$W_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} I_N & -I_N \\ I_N & I_N \end{pmatrix}$$

followed by waves $W_i = I_{2N}$. Therefore, since the elementary transformations which compose these waves are computed in the diagonal processors, it is sufficient to check there if the angles are close to $\pi/4$ for W_1 and to 0 for the other W_i's. The largest difference from these ideal angles can thus be used as a valid stopping criterion, and it can easily be propagated through the diagonal processors during the determination of the next iterates. The implementation of this criterion on our array of processors (Fig. 1) could be facilitated by the addition of direct links between successive diagonal processors. At the same time, the evaluation and propagation of the coefficients c_k, as required in an efficient implementation of the algorithm, could be performed without prohibitive costs.

7. Numerical Aspects

In this section we briefly investigate two numerical properties of the algorithm:

(i) the stability of the factorizations, and
(ii) the stability of the relationship between the factors L_k and U_k.

(i) At each iteration step, two triangular factorizations are to be computed. With the notation and the hypotheses of Theorem 2 and Corollary 1, these are

$$Z_0 = L_0 \Sigma L_0^T \quad \text{and} \quad Y^{-1} Z_0 Y^{-1} = U_0 \Sigma U_0^T \tag{62}$$

at the first step, and their existence at subsequent steps is then also guaranteed. Without pivoting, numerical instabilities possibly occur due to rounding errors [BP], as illustrated in the following simple example. The matrix

$$Z_0 = \begin{pmatrix} -\varepsilon & -1 \\ -1 & 1 \end{pmatrix}, \qquad \varepsilon > 0,$$

gives

$$L_0 = \begin{pmatrix} \varepsilon^{1/2} & 0 \\ \varepsilon^{-1/2} & (1 + 1/\varepsilon)^{1/2} \end{pmatrix}.$$

In finite precision, if ε is so small that $(1 + 1/\varepsilon)$ is rounded to $1/\varepsilon$, then the computed factor $\hat{L}_0$ is such that

$$\hat{L}_0 \Sigma \hat{L}_0^T = \hat{Z}_0 = \begin{pmatrix} -\varepsilon & -1 \\ -1 & 0 \end{pmatrix}.$$

Such instabilities reflect the bad conditioning of L_0. Theorem 2 ensures that the condition number of the L_k's remains finite if it is so initially; also we have observed empirically that this condition number usually improves with k. Essentially, we thus have to guarantee a well-conditioned solution L_0 and U_0 to (62). In theory this is possible by first applying a well-conditioned (ideally unitary) congruence simultaneously to Z_0 and $Y^{-1} Z_0 Y^{-1}$ in order to improve the (positive or negative) definiteness of their diagonal blocks. If only one matrix had to be handled, a practical approach would be to use techniques developed in [BP]. However, here as for the preprocessing of a pencil congruent to a Hamiltonian one (see Theorem 3 and thereafter), two symmetric matrices are involved, and a practical and effective construction of the congruences is an open question.

Similarly for the case of a Hamiltonian matrix H (see Section 2.1 and Remark 4), we could perform an initial symplectic transformation in order to improve the positivity of Q. In [PV] a class of unitary symplectic transformations is described which could be used for this purpose, but here again a practical constructive algorithm is still lacking. Hence, our algorithm in its present form is of real interest only for matrices (pencils) with definite diagonal blocks which, moreover, are far enough from singularity.

(ii) When analyzing the stability of the relationship between L_k and U_k, stated via (22) and (23) for a Hamiltonian matrix (extension of what follows to more general

situations is direct), some parallels can be drawn with the stability of the basic sign recursion (7) for H, as implicitly contained in Higham [H2]. We recall this briefly. Let the current computed iterate be $\hat{H} = H + \Delta$ and its *exact* update be $\hat{H}_+ = H_+ + \Delta_+$. With $c_k = 1$ for simplicity, we have, by (7),

$$H_+ + \Delta_+ = \tfrac{1}{2}[H + \Delta + (H + \Delta)^{-1}] = \tfrac{1}{2}[H + \Delta + H^{-1} - H^{-1}\Delta H^{-1}] + O(\Delta^2),$$

which gives

$$\Delta_+ \approx \tfrac{1}{2}[\Delta - H^{-1}\Delta H^{-1}], \tag{63}$$

up to the first order. Let T be the similarity transformation which diagonalizes H_∞: $T^{-1}H_\infty T = \Sigma$ (see (9)). Then

$$T^{-1}\Delta_+ T \approx \tfrac{1}{2}[T^{-1}\Delta T - (T^{-1}H^{-1}T)(T^{-1}\Delta T)(T^{-1}H^{-1}T)]$$

follows from (63), again up to the first order. Defining a norm by $\|X\|_T \equiv \|T^{-1}XT\|_2$, we finally have

$$\|\Delta_+\|_T \approx \tfrac{1}{2}\|\Delta\|_T(1 + \|H^{-1}\|_T^2), \tag{64}$$

which tends to $\|\Delta\|_T$ as H converges to H_∞. Therefore, at least near convergence, the error at a given step is propagated in a stable way.

We now turn to the relationship between the triangular factors L and U in our coupled recursions algorithm. Let the computed values be $\hat{L}$ and $\hat{U}$ at some iterate; the corresponding Hamiltonian matrices are denoted as

$$JH_L = \hat{L}\Sigma\hat{L}^T \quad \text{and} \quad -H_U J = \hat{U}\Sigma\hat{U}^T,$$

according to (22) and (23). The deviation Δ of H_U with respect to H_L ($\Delta \equiv H_U - H_L$) is used to characterize how much $\hat{L}$ and $\hat{U}$ are unrelated at this iterate. The propagation of Δ in *exact* arithmetic can be estimated via the following rewriting of (24) and (25) ($c_k = 1$):

$$(H_L)_+ = \tfrac{1}{2}[H_L + H_U^{-1}] \quad \text{and} \quad (H_U)_+ = \tfrac{1}{2}[H_U + H_L^{-1}].$$

With $\Delta_+ \equiv (H_U)_+ - (H_L)_+$, we have

$$\Delta_+ = \tfrac{1}{2}[\Delta + H_L^{-1} - (H_L + \Delta)^{-1}] = \tfrac{1}{2}[\Delta + H_L^{-1}\Delta H_L^{-1}] + O(\Delta^2).$$

Then,

$$\|\Delta_+\|_T \approx \tfrac{1}{2}\|\Delta\|_T(1 + \|H_L^{-1}\|_T^2)$$

is obtained in a way similar to that leading to (64). Therefore, at least near convergence, the deviation occurring at a given step is not really amplified by the following ones.

In addition to Δ_+, the total (in finite precision arithmetic) deviation at the next iterate also consists of the rounding error and of the interaction between this one and Δ_+. The rounding error comes from the computation of (26) and (27); each of those involve an inversion of a triangular factor and a triangularization. The error due to the inversion depends on the conditioning of the triangular factors (see (i) above), while the error due to the triangularization behaves well (Subsection 6.2) and tends ultimately to the round-off level of the machine, since the transformations involved

tend to be unitary when the process approaches convergence. On the other hand, the dominant term of the interaction part is of a higher order than that of the rounding error and Δ_+. Summing up, we conclude that the numerical evolution of the relationship between L and U is satisfactory *ultimately*, provided that the initial decompositions (62) are stable. However, nothing precise is known about its global numerical behavior over the whole process, to the same extent that the global numerical stability of the basic sign recursion for H is not clearly understood.

8. Conclusion

In this paper a new algorithm was presented for implementing the generalized sign recursion of Gardiner and Laub in a factored form. This idea was inspired from the so-called square root algorithms as, e.g., surveyed in [A]. Our implementation differs from earlier work by propagating two coupled factorizations simultaneously. This approach has the following advantages:

(i) The complexity (i.e., number of flops) is lowered for the generalized sign recursion with dense Y (Section 5).
(ii) It can be efficiently implemented on a triangular grid of $O(N^2)$ processors leading to an optimal speedup (Section 6).

The method directly applies to the solution of standard AREs with a positive definite weighting matrix Q. Its extension to a semidefinite Q (Section 7) or to the more general case of pencils congruent a Hamiltonian one (Section 4), requires the derivation of an appropriate congruence (Theorem 3 and Section 7) for starting the factored sign recursion. This is yet an open problem.

Acknowledgment. We wish to thank Judith Gardiner and Alan Laub for drawing our attention to this problem and for several useful discussions. We also thank an anonymous reviewer whose remarks initiated Section 7 and who provided the example at the beginning of that section.

References

[A] B. D. O. Anderson, Second-order convergent algorithms for the steady-state Riccati equation, *Internat. J. Control*, **28** (1978), 295–306.
[B1] A. Y. Barraud, Investigations autour de la fonction signe d'une matrice. Application a l'équation de Riccati, *RAIRO Automat.*, **13** (1979), 335–368.
[BS] R. Bartels and G. Stewart, Solution of the matrix equation $AX + XB = C$, *Comm. ACM*, **15** (1972), 820–826.
[B2] J. Bognar, *Indefinite Inner Product Spaces*, Springer-Verlag, New York, 1974.
[BBVH] A. Bojanczyk, R. Brent, P. Van Dooren, and F. de Hoog, Downdating the Cholesky factorization, *SIAM J. Sci. Statist. Comput.*, **8** (1987), 210–221.
[BL] R. P. Brent and F. T. Luk, Computing the Cholesky Factorization Using a Systolic Architecture, Report TR-CS-82-08, Department of Computer Science, Australian National University, 1982; also *Austral. Comput. Sci. Commun.*, **5** (1983), 295–302.
[BP] J. R. Bunch, and B. N. Parlett, Direct methods for solving symmetric indefinite systems of linear equations, *SIAM J. Numer. Anal.*, **8** (1971), 639–655.

[B3] R. Byers, Hamiltonian and Symplectic Algorithms for the Algebraic Riccati Equation, Ph.D. thesis, Cornell University, Ithaca, NY, 1983.

[B4] R. Byers, Solving the algebraic Riccati equation with the matrix sign function, *Linear Algebra Appl.*, **85** (1987), 267–279.

[DS] J. J. Dongarra and D. C. Sorensen, Linear algebra on high performance computers, *Appl. Math. Comput.*, **20** (1986), 57–88.

[GL] J. Gardiner and A. Laub, A generalization of the matrix-sign-function solution for algebraic Riccati equations, *Internat. J. Control*, **44** (1986), 823–832.

[GS] G. H. Golub and M. A. Saunders, Linear least squares and quadratic programming, in *Integer and Nonlinear Programming II* (J. Abadie ed.), pp. 229–256, North Holland, Amsterdam, 1970.

[GV] G. H. Golub and C. F. Van Loan, *Matrix Computations*, North Oxford Academic, Oxford, 1983.

[H1] S. Hammarling, Numerical solution of the stable non-negative definite Lyapunov equation, *IMA J. Numer. Anal.*, **2** (1982), 303–323.

[H2] N. J. Higham, Newton's method for the matrix square root, *Math. Comp.*, **46** (1986) 537–549.

[HB] K. Hwang and F. A. Briggs, *Computer Architecture and Parallel Processing*, McGraw-Hill, New York, 1984.

[K] H. T. Kung, Why systolic architectures?, *IEEE Trans. Comput.*, **15** (1983), 37–46.

[KL] H. T. Kung and C. E. Leiserson, Algorithms for VLSI processor arrays, in *Introduction to VLSI Systems* (C. Mead and L. Conway, eds.), pp. 271–292, Addison-Wesley, Reading, MA, 1980.

[L1] A. Laub, A Schur method for solving the algebraic Riccati equation, *IEEE Trans. Automat. Control*, **24** (1979), 913–921.

[L2] A. Laub, Schur techniques in invariant embedding methods for solving two-point boundary value problems, *Proceedings of the 21st IEEE Conference on Decision and Control*, Orlando, Florida, 1982, pp. 56–61.

[LH] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[L3] F. T. Luk, A triangular processor array for computing singular values, *Linear Algebra Appl.*, **77** (1986), 259–273.

[L4] F. T. Luk, Architectures for computing eigenvalues and SVDs, in *Highly Parallel Signal Processing Architectures*, pp. 24–33, Proceedings of the Society of Photo-optical Instrumentation Engineers, Vol. 614, 1986.

[OS] D. P. O'Leary and G. W. Stewart, Data-flow algorithms for parallel matrix computations, *Comm. ACM*, **28** (1985), 840–853.

[PV] C. C. Paige and C. Van Loan, A Schur decomposition for Hamiltonian matrices, *Linear Algebra Appl.*, **41** (1981), 11–32.

[PLS] T. Pappas, A. Laub, and N. Sandell, On the discrete-time algebraic Riccati equation, *IEEE Trans. Automat. Control*, **25** (1980), 631–641.

[U] F. Uhlig, Inertia and eigenvalue relations between symmetrized and symmetrizing matrices for the real and the general field case, *Linear Algebra Appl.*, **35** (1981), 203–226.

[V] P. Van Dooren, A generalized eigenvalue approach for solving Riccati equations, *SIAM J. Sci. Statist. Comput.*, **2** (1981), 121–135.

[W] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.