A Symplectic Acceleration Method for the Solution of the Algebraic Riccati Equation on a Parallel Computer

W. W. Lin
Institute of Applied Mathematics
National Tsing-Hua University
Hsin-Chu, Taiwan, R.O.C.

and

S. S. You
Telecommunication Laboratories
Ministry of Transportation and Communications
Chung-Li, Taiwan, R.O.C.

Submitted by Volker Mehrmann

ABSTRACT

We give a cubic acceleration method for improving the current symplectic Jacobi-like algorithm for computing the Hamiltonian-Schur decomposition of a Hamiltonian matrix and finding the positive semidefinite solution of the Riccati equation. The acceleration method can speed up the rate of convergence at the end of the symplectic Jacobi-like process when the norm of the current strictly J-lower triangle has become sufficiently small; it has high parallelism and takes $O(n)$ computational time when implemented on a mesh-connected $n \times n$ array processor system. A quantitative analysis of convergence and numerical comparisons of one Jacobi sweep versus one correction step are presented.

1. INTRODUCTION

The problem of solving the algebraic Riccati equation

$$-XNX + XA + A^*X + K = 0 \tag{1.1}$$

(here A, K, N, and X are complex $n \times n$ matrices, $N = N^* \geqslant 0$, and $K = K^* \geqslant 0$) arises for instance in linear-quadratic optimal control problems. It is assumed that (A, B) is stabilizable and (C, A) is detectable, where B and C are full-rank factorizations of N and K, respectively [9]. Under these assumptions, the equation (1.1) has a unique positive semidefinite solution which is equivalent to the problem of finding an n-dimensional invariant subspace $\begin{bmatrix} Y \\ Z \end{bmatrix}$ corresponding to the stable eigenvalues of the Hamiltonian matrix

$$M = \begin{bmatrix} A & N \\ K & -A^* \end{bmatrix}. \tag{1.2}$$

The solution of Equation (1.1) is then obtained by $X = -ZY^{-1}$. A matrix M is called Hamiltonian if $(JM)^* = JM$, where J is the $2n \times 2n$ matrix

$$\begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}$$

and I is the $n \times n$ identity matrix. It is well known that a Hamiltonian matrix is invariant under symplectic similarity transformations (a matrix $S \in \mathbf{C}^{2n \times 2n}$ is symplectic if $S^*JS = J$). In 1981, Paige and Van Loan [11] proved that, if the eigenvalues of M have nonzero real parts, then M has a Schur-Hamiltonian decomposition, i.e., there exists a unitary symplectic matrix

$$Q = \begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix}, \qquad Q_1, Q_2 \in \mathbf{C}^{n \times n},$$

such that

$$Q^*MQ = \begin{bmatrix} T & G \\ 0 & -T^* \end{bmatrix} \equiv R, \tag{1.3}$$

where $G^* = G \in \mathbf{C}^{n \times n}$, $T \in \mathbf{C}^{n \times n}$ is upper triangular, and the eigenvalues of T are in the left half plane. The matrix R in (1.3) is called a J-upper triangular matrix. Since

$$M \begin{bmatrix} Q_1 \\ -Q_2 \end{bmatrix} = \begin{bmatrix} Q_1 \\ -Q_2 \end{bmatrix} T,$$

the desired nonnegative Hermitian solution of the Riccati equation (1.1) is given by $X = Q_2 Q_1^{-1}$ [11].

Bunse-Gerstner and Mehrmann [2] and Ammar and Mehrmann [1] proposed the SR algorithm and SISHC algorithm, respectively, for solving the equation (1.1) on a sequential machine. Both methods preserve the Hamiltonian structure. In the former, some intermediate transformation may fail to exist or may become very ill conditioned. But the latter uses only unitary symplectic transformations. The QR algorithm using unitary transformations, proposed earlier by Laub [9], unfortunately destroys the Hamiltonian structure. However, the above three algorithms are not suitable for parallel processing.

In 1989, Byers [4] first proposed the symplectic Jacobi-like algorithm for the computation of the Hamiltonian-Schur decomposition of a Hamiltonian matrix. This algorithm requires $O(n)$ computational time for a sweep when implemented on a mesh-connected $n \times n$ array processor system. It uses only unitary symplectic transformations and is close to the Jacobi-like method for a non-Hermitian matrix [12]. However, the convergence of Byers's method [4] can be very slow if the Hamiltonian matrix is not near to normality; and very often the method does not converge at all for problems of dimension greater than 20. Recently, Bunse-Gerstner [3] developed a symplectic Jacobi-like algorithm for the computation of the Hamiltonian-Schur decomposition (1.3) based on the technique of Eberlein [7]. Each iterate in [3] needs only local information about the current matrix, thus admitting efficient parallel implementations on certain parallel architectures. The numerical experiments show that the convergence seems to be between linear and quadratic, which is much faster than the method in [4] when the matrices are far from normality.

The purpose of this paper is to describe an effective acceleration method (correction method) which can be used to speed up the rate of convergence at the end of the symplectic Jacobi-like algorithm in [3] or [4], when the norm of the strictly J-lower triangle of the current Hamiltonian matrix has become sufficiently small. The new method can be implemented on a mesh-connected $n \times n$ array processor system in $O(n)$ time.

In Section 2, we first briefly introduce the symplectic Jacobi-like (SJL) method (see [3, 4] for more details). Then, we propose a reordering technique for eigenvalues and its parallel implementation. In Section 3, we derive some special matrix equations and then use these equations to develop a cubic symplectic acceleration method (SAM). This method is regarded as a corrector after some sweeps of the SJL method. A parallel processing for the SAM method and a Hermitian updating of an approximate solution of (1.1) are also given. In Section 4 we present a quantitative convergence analysis for the acceleration method. Theoretically, we prove that the SAM method convergences cubically. In Section 5, we compare the flop counts of the SAM method and the SJL method. The comparison shows that the SAM method needs fewer flops. Finally, we give some examples first computed by the SJL method and then corrected by the SAM method. Those results show that the accuracy of the solutions is almost twice more than that by the SJL method only.

We denote by I_n (or I) the $n \times n$ unit matrix, by A^* the complex conjugate transpose of an $n \times n$ matrix, and by $B \oplus C$ the direct sum of matrices B and C.

2. SYMPLECTIC JACOBI-LIKE ALGORITHM

Byers [4] developed a symplectic Jacobi-like method for reducing the Hamiltonian matrix

$$M = \begin{bmatrix} A & N \\ K & -A^* \end{bmatrix}$$

to a J-upper triangular matrix (1.3) by Householder [$H(k, c, s)$-rotation] and Jacobi [$J(n, c, s)$-rotation] symplectic similarity transformations. These transformations, based on plane rotations, are used to annihilate the elements of K and the strictly lower triangular elements of A. We briefly describe these basic unitary symplectic rotations (see also [4]).

2.1. *Householder Symplectic Rotation $H(k, c, s)$*

Let

$$A_k = \begin{bmatrix} a_{k,k} & a_{k,k+1} \\ a_{k+1,k} & a_{k+1,k+1} \end{bmatrix}$$

be a 2×2 submatrix of A, and let $[c, s]^T$ be a unit eigenvector of A_k associated with the eigenvalue λ. Then the matrix

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

is unitary and satisfies

$$U^*A_kU = \begin{bmatrix} \lambda & \times \\ 0 & \times \end{bmatrix}.$$

Let $P = I_{k-1} \oplus U \oplus I_{n-k-1}$. We then call the unitary symplectic matrix of the form

$$H(k) := H(k, c, s) = \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix}$$

the Householder symplectic rotation.

2.2. *Jacobi symplectic rotation $J(n, c, s)$*
We now consider the 2×2 submatrix

$$M_n = \begin{bmatrix} a_{nn} & n_{nn} \\ k_{nn} & -\bar{a}_{nn} \end{bmatrix}$$

of M as in (1.2). Let $[c, s]^T$ be a unit eigenvector of M_n associated with λ. It is easily seen tht if $\operatorname{Re}\lambda \neq 0$, then $\bar{c}s \in \mathbf{R}$. Hence the matrix

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

is unitary and symplectic, and satisfies

$$V^*M_nV = \begin{bmatrix} \lambda & \times \\ 0 & -\bar{\lambda} \end{bmatrix}.$$

But when $\operatorname{Re}\lambda = 0$, i.e. $k_{nn}n_{nn} + (\operatorname{Re} a_{nn})^2 \leqslant 0$, then $\bar{c}s$ need not be real. We want to find a unitary symplectic rotation

$$V = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \quad \text{(say)}$$

with $\bar{c}s \in \mathbf{R}$ such that the element $e_2^TV^*M_nVe_1$ is as small as possible. This can be done by solving the following minimization:

$$\min_{c^2+s^2=1} \left|c^2 k_{nn} - cs(a_{nn} + \bar{a}_{nn}) - s^2 n_{nn}\right|.$$

This gives

$$c = \frac{1}{\sqrt{1+t^2}} \quad \text{and} \quad s = ct,$$

where

$$t = \frac{(\operatorname{sign} u)\left(|u| \pm \sqrt{u^2 + v^2}\right)}{v},$$

$$u = \frac{n_{nn} + k_{nn}}{2} \quad \text{and} \quad v = \operatorname{Re} a_{nn}.$$

We call a unitary symplectic matrix which has the structure

$$J(n) := J(n,c,s) = \begin{bmatrix} \Gamma & -\Sigma \\ \Sigma & \Gamma \end{bmatrix} \quad \text{with} \quad \Gamma, \Sigma \in \mathbf{C}^{n\times n},$$

where $\Gamma = \operatorname{diag}(1, \dots 1, c)$ and $\Sigma = \operatorname{diag}(0, \dots, 0, s)$, a *Jacobi symplectic rotation.*

2.3. *Choice of Rotation*

In general, there are two choices of the vector $[c, s]^T$ for the matrix A_k in Section 2.1 (for M_n in Section 2.2). We call the one for which $|c|$ in U as in Section 2.1 ($|c|$ in V as in Section 2.2) is the smallest the *outer H-rotation* (*outer J-rotation*), and the other the *inner H-rotation* (*inner G-rotation*); here H stands for Householder and J for Jacobi. To help insure convergence, the outer H- and J-rotations are preferred [12].

The symplectic Jacobi-like algorithm consists of a sequence of unitary symplectic similarity transformations $M := S^*MS$, where S is one of the two basic symplectic transformations described above. Each similarity transformation reduces the Frobenius norm $\sigma(\cdot)$ of the strictly J-lower triangle of M, where $\sigma(M)$ is defined by

$$\left(\sum_{j=1}^{n} \sum_{i=1}^{n} |k_{ij}|^2 + \sum_{i=j+1}^{n} |a_{ij}|^2 \right)^{1/2}.$$

Although the matrix M will converge to a J-upper triangular matrix by the symplectic Jacobi-like algorithm [4], it is not guaranteed that the diagonal elements of M with negative real parts appear in T [as in (1.3)], which is necessary to get the nonnegative Hermitian solution of (1.1). Therefore, a reordering process is needed to arrange all diagonal elements of M with negative real parts appearing on the diagonal of T. A sequential reordering by unitary symplectic similarity transformations of this second pass was described in [2]. However, a parallel implementation of this reordering process is not trivial at all. The next subsection and Figure 1 describe this parallel implementation.

2.4. *Reordering of the Eigenvalues and Parallel Implementation*

We now consider the case when the SJL method converges. That is, for $k = 1, 2, \dots,$ the sequence $M_{k+1} = Q_k^* M_k Q_k$ converges to a J-upper triangular matrix,

$$R := \begin{bmatrix} T & G \\ 0 & -T^* \end{bmatrix}$$

as in (1.3) with $M_1 := M$, where Q_k is the product of the H- and J-rotations.

Now, let $\operatorname{Re} t_{pp} > 0$, where t_{pp} is the pth diagonal entry of T. We first apply the Householder rotations $H(p), \dots, H(n-1)$ to move the element t_{pp} to the (n, n) position of T, and then apply the Jacobi rotation $J(n)$ to interchange t_{pp} with $-\bar{t}_{pp}$ [the (n, n) entry of the current $-T^*$]. Here the H- and J-rotations are chosen so that the corresponding eigenvalues interchanged. The following process formulates the interchange of the entry t_{pp} of T with $-\bar{t}_{pp}$ of $-T^*$:

$$\hat{R} := J(n)^* H(n-1)^* \cdots H(p)^* R H(p) \cdots H(n-1) J(n). \quad (2.1)$$

We exhibit in Figure 1 a useful strategy for devising a parallel implementation of the interchange algorithm on the matrix T $(n = 6)$ (the other matrices $-T^*$ and G are the same) for the worst case for reordering. That is, we consider the case that all diagonal elements of T having positive real parts.

1
$$\begin{array}{cccccc}\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathit{1}\end{array}$$

2
$$\begin{array}{cccccc}\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathit{1}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathit{1}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathit{1}&\mathit{1}&\Delta\end{array}$$

3
$$\begin{array}{cccccc}\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathit{1}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathit{1}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&2&2\\\mathrm{X}&\mathit{1}&\mathit{1}&\mathrm{X}&2&2\end{array}$$

4
$$\begin{array}{cccccc}\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&1\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&2&2\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&2&2\\\mathrm{X}&\mathrm{X}&2&2&\Delta&\mathrm{X}\\\mathit{1}&\mathrm{X}&2&2&\mathrm{X}&\mathit{3}\end{array}$$

5
$$\begin{array}{cccccc}\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&2&2\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&2&2\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&4&4&\mathit{3}\\2&2&\mathrm{X}&4&4&\mathit{3}\\2&2&\mathrm{X}&\mathit{3}&\mathit{3}&\Delta\end{array}$$

6
$$\begin{array}{cccccc}\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&4&4&\mathit{3}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&4&4&\mathit{3}\\\mathrm{X}&4&4&\Delta&\mathrm{X}&4\\\mathrm{X}&4&4&\mathrm{X}&\mathrm{X}&4\\\mathrm{X}&\mathit{3}&\mathit{3}&4&4&\Delta\end{array}$$

7
$$\begin{array}{cccccc}\mathrm{X}&\mathrm{X}&\mathrm{X}&4&4&\mathit{3}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&5&5&\mathrm{X}&\mathrm{X}\\4&\mathrm{X}&5&5&\mathrm{X}&\mathrm{X}\\4&\mathrm{X}&\mathrm{X}&\mathrm{X}&5&5\\\mathit{3}&\mathrm{X}&\mathrm{X}&\mathrm{X}&5&5\end{array}$$

8
$$\begin{array}{cccccc}\mathrm{X}&\mathrm{X}&5&5&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&5&5&\mathrm{X}&\mathrm{X}\\5&5&\Delta&\mathrm{X}&5&5\\5&5&\mathrm{X}&\mathrm{X}&5&5\\\mathrm{X}&\mathrm{X}&5&5&\Delta&\mathrm{X}\\\mathrm{X}&\mathrm{X}&5&5&\mathrm{X}&\mathit{6}\end{array}$$

9
$$\begin{array}{cccccc}\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&5&5\\\mathrm{X}&7&7&\mathrm{X}&5&5\\\mathrm{X}&7&7&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&7&7&\mathit{6}\\5&5&\mathrm{X}&7&7&\mathit{6}\\5&5&\mathrm{X}&\mathit{6}&\mathit{6}&\Delta\end{array}$$

10
$$\begin{array}{cccccc}\mathrm{X}&7&7&\mathrm{X}&\mathrm{X}&\mathrm{X}\\7&\Delta&\mathrm{X}&7&7&\mathit{6}\\7&\mathrm{X}&\mathrm{X}&7&7&\mathit{6}\\\mathrm{X}&7&7&\Delta&\mathrm{X}&7\\\mathrm{X}&7&7&\mathrm{X}&\mathrm{X}&7\\\mathrm{X}&\mathit{6}&\mathit{6}&7&7&\Delta\end{array}$$

11
$$\begin{array}{cccccc}8&8&\mathrm{X}&7&7&\mathit{6}\\8&8&\mathrm{X}&\mathrm{X}&\mathrm{X}&7\\\mathrm{X}&\mathrm{X}&8&8&\mathrm{X}&7\\7&\mathrm{X}&8&8&\mathrm{X}&\mathrm{X}\\7&\mathrm{X}&\mathrm{X}&\mathrm{X}&8&8\\\mathit{6}&7&7&\mathrm{X}&8&8\end{array}$$

12
$$\begin{array}{cccccc}\Delta&\mathrm{X}&8&8&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&8&8&\mathrm{X}&\mathrm{X}\\8&8&\Delta&\mathrm{X}&8&8\\8&8&\mathrm{X}&\mathrm{X}&8&8\\\mathrm{X}&\mathrm{X}&8&8&\Delta&\mathrm{X}\\\mathrm{X}&\mathrm{X}&8&8&\mathrm{X}&\mathit{9}\end{array}$$

13
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&8&8\\\mathrm{X}&\mathrm{A}&\mathrm{A}&\mathrm{X}&8&8\\\mathrm{X}&\mathrm{A}&\mathrm{A}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{A}&\mathrm{A}&\mathit{9}\\8&8&\mathrm{X}&\mathrm{A}&\mathrm{A}&\mathit{9}\\8&8&\mathrm{X}&\mathit{9}&\mathit{9}&\Delta\end{array}$$

14
$$\begin{array}{cccccc}\Delta&\mathrm{A}&\mathrm{A}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{A}&\Delta&\mathrm{X}&\mathrm{A}&\mathrm{A}&\mathit{9}\\\mathrm{A}&\mathrm{X}&\mathrm{X}&\mathrm{A}&\mathrm{A}&\mathit{9}\\\mathrm{X}&\mathrm{A}&\mathrm{A}&\Delta&\mathrm{X}&\mathrm{A}\\\mathrm{X}&\mathrm{A}&\mathrm{A}&\mathrm{X}&\mathrm{X}&\mathrm{A}\\\mathrm{X}&\mathit{9}&\mathit{9}&\mathrm{A}&\mathrm{A}&\Delta\end{array}$$

15
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{X}&\mathrm{A}&\mathrm{A}&\mathit{9}\\\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{A}\\\mathrm{X}&\mathrm{X}&\mathrm{B}&\mathrm{B}&\mathrm{X}&\mathrm{A}\\\mathrm{A}&\mathrm{X}&\mathrm{B}&\mathrm{B}&\mathrm{X}&\mathrm{X}\\\mathrm{A}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{B}&\mathrm{B}\\\mathit{9}&\mathrm{A}&\mathrm{A}&\mathrm{X}&\mathrm{B}&\mathrm{B}\end{array}$$

16
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{B}&\mathrm{B}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\Delta&\mathrm{B}&\mathrm{B}&\mathrm{X}&\mathrm{X}\\\mathrm{B}&\mathrm{B}&\Delta&\mathrm{X}&\mathrm{B}&\mathrm{B}\\\mathrm{B}&\mathrm{B}&\mathrm{X}&\mathrm{X}&\mathrm{B}&\mathrm{B}\\\mathrm{X}&\mathrm{X}&\mathrm{B}&\mathrm{B}&\Delta&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{B}&\mathrm{B}&\mathrm{X}&C\end{array}$$

17
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{B}&\mathrm{B}\\\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{B}&\mathrm{B}\\\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{D}&\mathrm{D}&C\\\mathrm{B}&\mathrm{B}&\mathrm{X}&\mathrm{D}&\mathrm{D}&C\\\mathrm{B}&\mathrm{B}&\mathrm{X}&C&C&\Delta\end{array}$$

18
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\Delta&\mathrm{X}&\mathrm{D}&\mathrm{D}&C\\\mathrm{X}&\mathrm{X}&\Delta&\mathrm{D}&\mathrm{D}&C\\\mathrm{X}&\mathrm{D}&\mathrm{D}&\Delta&\mathrm{X}&\mathrm{D}\\\mathrm{X}&\mathrm{D}&\mathrm{D}&\mathrm{X}&\mathrm{X}&\mathrm{D}\\\mathrm{X}&C&C&\mathrm{D}&\mathrm{D}&\Delta\end{array}$$

19
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{X}&\mathrm{D}&\mathrm{D}&C\\\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{D}&\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}\\\mathrm{D}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{E}&\mathrm{E}\\C&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{E}&\mathrm{E}\end{array}$$

20
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{E}&\mathrm{E}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\Delta&\mathrm{E}&\mathrm{E}\\\mathrm{X}&\mathrm{X}&\mathrm{E}&\mathrm{E}&\Delta&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{E}&\mathrm{E}&\mathrm{X}&F\end{array}$$

21
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{D}&\mathrm{D}\\\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{D}&\mathrm{D}\\\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&F\\\mathrm{E}&\mathrm{E}&\mathrm{X}&\mathrm{X}&\Delta&F\\\mathrm{E}&\mathrm{E}&\mathrm{X}&F&F&\Delta\end{array}$$

22
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&F\\\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&F\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}\\\mathrm{X}&F&F&\mathrm{X}&\mathrm{X}&\Delta\end{array}$$

23
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&F\\\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}\\F&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\Delta\end{array}$$

24
$$\begin{array}{cccccc}\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\Delta&\mathrm{X}\\\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\mathrm{X}&\Delta\end{array}$$

FIG. 1. The worst case for reordering.

We use the same notation as i [4] and express the numbers in hexadecimal. We label the H-rotations generated in time step k with the upright numeral for k, and the J-rotation generated in time step k with the italic numeral. The triangle Δ denotes an eigenvalue with negative real part just interchanged in the northwestern block T. Each $\times$ represents an entry in T along with the superimposed associated entry of T or G.

In step 1, a J-rotation represented by italic *1* is generated from t_{nn}, $-\bar{t}_{nn}$, and g_{nn} and applied as the similarity transformation

$$\begin{bmatrix} -\bar{t}_{nn} & \times \\ 0 & t_{nn} \end{bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}^* \begin{bmatrix} t_{nn} & g_{nn} \\ 0 & -\bar{t}_{nn} \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix}.$$

In step 2, the first desired eigenvalue $-\bar{t}_{nn}$, denoted by Δ, is exchanged into the (n, n) position of T, and the J-rotation information is passed horizontally and vertically along the last row and column of T, respectively.

In step 3, an H-rotation represented by 2 is generated from $t_{n-1,n-1}$, $-\bar{t}_{nn}$, and the $(n-1, n)$ entry of the current T and applied as the similarity transformation

$$\begin{bmatrix} -\bar{t}_{nn} & \times \\ 0 & t_{n-1,n-1} \end{bmatrix} = \begin{bmatrix} c & -\bar{s} \\ s & \bar{c} \end{bmatrix}^* \begin{bmatrix} t_{n-1,n-1} & \times \\ 0 & -\bar{t}_{nn} \end{bmatrix} \begin{bmatrix} c & -\bar{s} \\ s & \bar{c} \end{bmatrix}.$$

The J-rotation information generated in step 1 is continuously passed horizontally and vertically along the last row and column of T, respectively.

In step 4, the first desired eigenvalue $-\bar{t}_{nn}$ is exchanged into the $(n-1, n-1)$ position of T, and a J-rotation represented by italic *3* is generated to move the second desired eigenvalue $-\bar{t}_{n-1,n-1}$ to the (n, n) position of T. The H-rotation information (2-by-2 block) denoted by 2 generated in step 3 is passed horizontally and vertically outward, respectively.

In step 5, the second desired eigenvalue $-\bar{t}_{n-1,n-1}$ is exchanged into the (n, n) position of T, and an H-rotation represented by 4 is generated to move the eigenvalue $-\bar{t}_{nn}$ to the $(n-2, n-2)$ position of T. The J-rotation information denoted by *3* is passed horizontally and vertically along the last row and column of T, respectively.

In step 6, the eigenvalues $-\bar{t}_{nn}$ and $-\bar{t}_{n-1,n-1}$ are exchanged into the $(n-2, n-2)$ and (n, n) positions of T, respectively. The H-rotation information denoted by 4 generated in step 5 is passed horizontally and vertically outward, respectively.

In step 7, two H-rotations represented by 5 are generated to move the eigenvalues $-\bar{t}_{nn}$ and $-\bar{t}_{n-1,n-1}$ to the $(n-3, n-3)$ and $(n-1, n-1)$

positions of T, respectively. The H-rotation denoted by 4 and the J-rotation denoted by *3* are continuously passed outward.

In step 8, $-\bar{t}_{nn}$ and $-\bar{t}_{n-1,n-1}$ are exchanged into the $(n-3, n-3)$ and $(n-1, n-1)$ positions, respectively, and a J-rotation represented by italic *6* is generated to move the eigenvalue $-\bar{t}_{n-2,n-2}$ to the (n, n) position of T. The H-rotations denoted by 5 are passed horizontally and vertically outward, respectively.

Steps 9 to 12, 13 to 16, 17 to 20, and 21 to 24 are essentially similar to the procedure of steps 5 to 8, respectively. The H-rotations represented by 7, 8, A, B, D and E move the eigenvalues with negative real parts upward along the diagonal of T by exchanging the desired eigenvalues with the adjacent diagonal elements stepwise. The J-rotations represented by italic *9*, *C*, and *F* are generated to move the desired eigenvalues with negative real parts to the (n, n) position of T, respectively.

It can be shown that the above parallel interchange algorithm requires at most $4n$ computational time for the general case. In practice, for large k, the matrix

$$M_k = \begin{bmatrix} A_k & N_k \\ K_k & -A_k^* \end{bmatrix}$$

approaches a J-upper triangular matrix. Hence, the diagonal elements of A_k and $-A_k^*$ are close to the eigenvalues of M and pairwise well separated. Therefore, the interchange process can be applied if some diagonal elements of A_k have positive real parts.

3. THE SYMPLECTIC ACCELERATION METHOD AND THE NONNEGATIVE HERMITIAN SOLUTION FOR THE RICCATI EQUATION

In this section we develop an acceleration technique, the so-called symplectic acceleration method (SAM), for reducing the arithmetic cost for the case when the eigenvalues of the Hamiltonian M as in (1.2) are distinct. This method can speed up the rate of convergence at the end of the SJL method in Section 2 when the norm of the strictly J-lower triangle of M has become sufficiently small. In other words, M can be regarded as a perturbation of a J-upper triangular matrix. The main justification for this method is that the operations are very well suited to the mesh-connected system and the total computational cost is only $O(n)$. Although our method, in its present form, is applicable only when the eigenvalues of M are distinct, this is in fact the case for the matrices arising from a number of engineering problems of significant practical importance.

3.1. *The Derivation of the Equations*

Let

$$S = \begin{bmatrix} P & Q \\ -Q & P \end{bmatrix}$$

be a $2n \times 2n$ unitary symplectic matrix. We suppose that

$$P = I + X_1 + X_2 \tag{3.1a}$$

and

$$Q = Y_1 + Y_2 \tag{3.1b}$$

with $\|X_1\| = \|Y_1\| = O(\varepsilon)$ and $\|X_2\| = \|Y_2\| = O(\varepsilon^2)$. (Here ε is small in magnitude, and $\|\cdot\|$ is an arbitrary matrix norm.) We will indicate how to choose X_i and Y_i $(i = 1, 2)$ so that S is fairly close to a unitary symplectic matrix (see also [6]).

Since S is symplectic and unitary, it follows that

$$P^*P + Q^*Q = I \tag{3.2a}$$

and

$$P^*Q - Q^*P = 0. \tag{3.2b}$$

If we ignore the terms in (3.2a) and (3.2b) of order higher than two, we then have

$$X_1^* + X_1 = 0, \tag{3.3a}$$

$$X_2^* + X_2 + X_1^*X_1 + Y_1^*Y_1 = 0, \tag{3.3b}$$

and

$$Y_1^* - Y_1 = 0, \tag{3.4a}$$

$$Y_2^* + Y_1^*X_1 - Y_2 - X_1^*Y_1 = 0. \tag{3.4b}$$

It follows that

$$X_1 = -X_1^* \equiv V_1 \text{ is skew-Hermitian [from (3.3a)]}, \quad (3.5a)$$

$$Y_1 = Y_1^* \equiv W_1 \text{ is Hermitian [from (3.4a)]}, \quad (3.5b)$$

$$X_2 - \frac{V_1^2}{2} + \frac{W_1^2}{2} \equiv V_2 \text{ is skew-Hermitian [from (3.3b)]}, \quad (3.5c)$$

$$Y_2 - \frac{W_1V_1 + V_1W_1}{2} \equiv W_2 \text{ is Hermitian [from (3.4b)]}. \quad (3.5d)$$

Hence our original version of the proposed approximations becomes

$$P = I + V_1 + \left(V_2 + \frac{V_1^2}{2} - \frac{W_1^2}{2} \right) \equiv I + O(\varepsilon) + O(\varepsilon^2) \quad (3.6a)$$

and

$$Q = W_1 + \left(W_2 + \frac{W_1V_1 + V_1W_1}{2} \right) \equiv O(\varepsilon) + O(\varepsilon^2). \quad (3.6b)$$

Suppose that

$$M = \begin{bmatrix} A & N \\ K & -A^* \end{bmatrix}$$

is close to a $2n \times 2n$ J-upper triangular matrix. That is, the elements of the strictly lower triangle of A and the elements of K are sufficiently small [with magnitude $O(\varepsilon)$, say]. By assumptions at the beginning of this section the diagonal elements of A, which are close to the eigenvalues of M, are also distinct. We now choose

$$S = \begin{bmatrix} P & Q \\ -Q & P \end{bmatrix}$$

with P and Q as in (3.6) so that the norm $\sigma(S^*MS)$ is as small as possible.

Let

$$S^*MS = \tilde{M} \equiv \begin{bmatrix} \tilde{A} & \tilde{N} \\ \tilde{K} & -\tilde{A}^* \end{bmatrix}. \tag{3.7}$$

Expanding the left-hand side of (3.7), we get

$$\tilde{A} \equiv P^*AP - Q^*KP - P^*NQ - Q^*A^*Q \tag{3.8a}$$

and

$$\tilde{K} \equiv Q^*AP + P^*KP - Q^*NQ + P^*A^*Q. \tag{3.8b}$$

Write $A = A_0 + A_1$, where A_0 is upper triangular and A_1 is strictly lower triangular. Substitute P and Q as in (3.6a, b) into (3.8a, b), respectively, and collect the matrices with the same order. We then have

$$\begin{aligned} \tilde{A} \equiv{}& A_0 + [A_1 - V_1A_0 + A_0V_1 - NW_1] \\ &+ \Bigg[A_0\left(V_1 + \frac{V_1^2}{2} - \frac{W_1^2}{2}\right) + \left(\frac{V_1^2}{2} - \frac{W_1^2}{2} - V_2\right)A_0 \\ &\qquad -V_1A_0V_1 - V_1A_1 + A_1V_1 - W_1K + V_1NW_1 \\ &\qquad -N\left(W_2 + \frac{W_1V_1 + V_1W_1}{2}\right) - W_1A_0^*W_1 \Bigg] + O(\varepsilon^3) \\ \equiv{}& A_0 + E_1 + E_2 + O(\varepsilon^3). \end{aligned} \tag{3.9a}$$

Similarly, we also have

$$\begin{aligned} \tilde{K} \equiv{}& [W_1A_0 + K + A_0^*W_1] \\ &+ \Bigg[W_1A_0V_1 + W_1A_1 + \left(W_2 - \frac{W_1V_1 + V_1W_1}{2}\right)A_0 + KV_1 - V_1K \\ &\qquad -W_1NW_1 + A_0^*\left(W_2 + \frac{W_1V_1 + V_1W_1}{2}\right) \\ &\qquad -V_1A_0^*W_1 + A_1^*W_1 \Bigg] + O(\varepsilon^3) \\ \equiv{}& F_1 + F_2 + O(\varepsilon^3). \end{aligned} \tag{3.9b}$$

If we set the strictly lower triangles of E_i and the matrices F_i $(i = 1, 2)$ to zero, then S^*MS is close to J-upper triangular. Hence from (3.9a, b) we solve the matrix equations

$$W_1 A_0 + A_0^* W_1 = -K, \tag{3.10a}$$

$$\begin{aligned} A_0 V_1 - V_1 A_0 &= NW_1 - A_1 + T_1 \\ &\equiv B, \end{aligned} \tag{3.10b}$$

$$\begin{aligned} W_2 A_0 + A_0^* W_2 &= \frac{V_1 K - K V_1}{2} - \frac{W_1 A_1 + A_1^* W_1}{2} + \frac{W_1 T_1 + T_1^* W_1}{2} \\ &\equiv C \qquad \text{(by computation)}, \end{aligned} \tag{3.10c}$$

$$\begin{aligned} A_0 V_2 - V_2 A_0 &= \frac{V_1 A_1 - A_1 V_1}{2} + \frac{A_0 W_1^2 - W_1^2 A_0}{2} \\ &\quad + \frac{(NV_1 - V_1 N) W_1}{2} + \frac{T_1 V_1 - V_1 T_1}{2} + NW_2 + T_2 \\ &\equiv D, \end{aligned} \tag{3.10d}$$

for Hermitian matrices W_1 and W_2, as well as for skew-Hermitian V_1 and V_2, where T_1 and T_2 are two suitable upper triangular matrices so that the strictly lower triangles of $A_0 V_1 - V_1 A_0$ and $A_0 V_2 - V_2 A_0$ are equal to those of B and D, respectively.

3.2. *Parallel Implementation*

We first consider the equation (3.10a) componentwise. Since W_1 and K are Hermitian and A_0 is upper triangular with distinct diagonal elements having negative real parts, the matrix W_1 can be computed recursively as follows: For $j \geqslant i$,

$$w_{ij} = \frac{-k_{ij} - \sum_{k=1}^{j-1} a_{kj} w_{ik} - \sum_{k=1}^{i-1} \bar{a}_{ik} w_{kj}}{\bar{a}_{ii} + a_{jj}}. \tag{3.11}$$

The values a_{ij} and w_{ij} $(j \geqslant i)$ which have been computed propagate themselves simultaneously rightward and downward at each time step as a wavefront propagation shown in Figure 2. Hence the elements w_{ij} on the

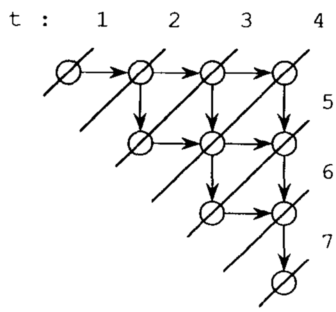

FIG. 2. Data flow of the matrix A_0.

same antidiagonal can be computed independently, and it needs $O(n)$ computational time to solve for W_1.

Next, we solve the equation (3.10b) for the skew-Hermitian V_1. That is, we find a skew-Hermitian matrix V_1 so that the strictly lower triangle of $A_0V_1 - V_1A_0$ is equal to that of B. It is easily seen that the matrix V_1 can be recursively solved by the following formula: For $i \geqslant j$,

$$v_{ij} = \frac{b_{ij} - \sum_{k=i+1}^{n} a_{ik}v_{kj} + \sum_{k=1}^{j-1} v_{ik}a_{kj}}{a_{ii} + a_{jj}}. \tag{3.12}$$

If we suitably arrange the data flow of the upper triangular matrix A_0 as shown in Figure 3, then the values a_{ij} propagate themselves simultaneously rightward and upward at each time step, and so do the values v_{ij} just computed. Therefore, the strictly lower triangular elements v_{ij} of V_1 on the same subdiagonal can be found independently, and it needs $O(n)$ computational time to solve for V_1.

Similarly, we can also solve for W_2 of (3.10c) in the same way. Here, we need to compute two matrix multiplications V_1K and W_1A_1 by using a data-flow algorithm [10] in $O(n)$ computational time, where V_1 and W_1 have been computed by (3.10b) and (3.10a) respectively. Finally, V_2 in the equation (3.10d) can also be found in the same way as above.

3.3. *Hermitian Updating of the Nonnegative Solution for the Riccati Equation in the SAM Method*

Suppose that

$$U = \begin{bmatrix} S_1 & S_2 \\ -S_2 & S_1 \end{bmatrix}$$

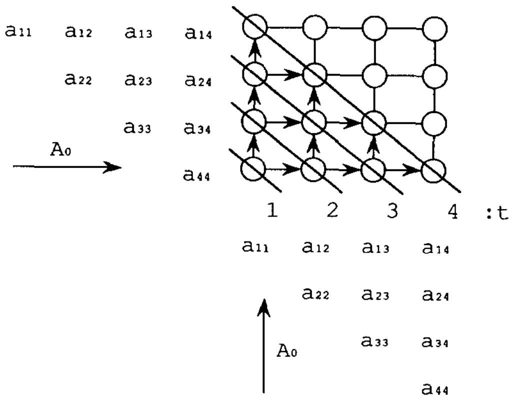

FIG. 3. Wavefront propagation.

is the $2n \times 2n$ unitary symplectic matrix accumulated by applying the SJL method and the reordering algorithm to the Hamiltonian matrix M as in Section 2. For the Hermitian updating of the nonnegative solution of the Riccati equation we are to compute the matrices

$$L := S_1^{-1}, \qquad Y := S_1^{-1}S_2, \quad \text{and} \quad X := S_2 S_1^{-1} \tag{3.13}$$

on the mesh-connected $n \times n$ processor system in $O(n)$ time. We first compute the QR factorization of $S_1 := Q_1 R_1$ and $S_1^* = \tilde{Q}_1 \tilde{R}_1$ in $O(n)$ time by using the mesh factorization algorithm in [8, Chapter 6]. Simultaneously, we compute $T_1 \equiv Q_1^* S_2$ and $\tilde{T}_1 \equiv S_2 \tilde{Q}_1$ in parallel by applying the Given rotations which produce Q_1 and $\tilde{Q}_1$, respectively, with the same data flow in the mesh factorization algorithm. Then we compute the inverse of the upper triangular R_1 and the lower triangular $\tilde{R}_1^*$ using the same wavefront as in (3.11). Finally, we compute $L = R_1^{-1}Q_1^*$, $Y = R_1^{-1}T_1$ and $X = \tilde{T}_1 \tilde{R}_1^{-*}$, by the data-flow algorithm [10] in $O(n)$ time. Furthermore, from the symplectic SVD of the unitary symplectic matrix U one can see that X and Y in (3.13) are Hermitian. In practice, we can symmetrize X and Y by the following simple Hermitian updating formulas:

$$X := \frac{X + X^*}{2} \quad \text{and} \quad Y := \frac{Y + Y^*}{2}.$$

Now, let

$$S=\begin{bmatrix} P & Q \\ -Q & P \end{bmatrix}$$

be the $2n \times 2n$ unitary symplectic matrix computed by the symplectic acceleration method, where P and Q are defined in (3.6). Since the set of the first n columns of the product of the matrices U and S,

$$US=\begin{bmatrix} S_1 & S_2 \\ -S_2 & S_1 \end{bmatrix}\begin{bmatrix} P & Q \\ -Q & P \end{bmatrix}=\begin{bmatrix} S_1P-S_2Q & S_2P+S_1Q \\ -S_2P-S_1Q & -S_2Q+S_1P \end{bmatrix}, \tag{3.14}$$

is close to a basis of the invariant subspace corresponding to the stable spectrum of M, the matrix

$$\begin{aligned}\tilde{X} &= (S_2P+S_1Q)(S_1P-S_2Q)^{-1} \\ &= (S_2P+S_1Q)P^{-1}(S_1-S_2QP^{-1})^{-1} \\ &= (S_2+S_1QP^{-1})S_1^{-1}(I-S_2QP^{-1}S_1^{-1})^{-1} \\ &= (S_2S_1^{-1}+S_1QP^{-1}S_1^{-1})(I-S_2QP^{-1}S_1^{-1})^{-1} \qquad (3.15)\end{aligned}$$

is then an approximation to the stable solution for the Riccati equation (1.1). We ignore the terms of order higher than two of the matrices P^{-1} and QP^{-1}, and get

$$\begin{aligned}QP^{-1} &= \left(W_1+W_2+\frac{W_1V_1+V_1W_1}{2}\right)(I-V_1) \\ &= W_1+W_2+\frac{V_1W_1-W_1V_1}{2}. \qquad (3.16)\end{aligned}$$

Next, substituting (3.16) into (3.15) and ignoring the terms of order higher than two of $\tilde{X}$ again, we get

$$\begin{aligned}\tilde{X} &= \left[S_2 S_1^{-1} + S_1(QP^{-1})S_1^{-1}\right] \\ &\quad\times\left[I + S_2(QP^{-1})S_1^{-1} + S_2W_1(S_1^{-}S_2)W_1S_1^{-1}\right] \\ &= S_2S_1^{-1} + S_1(QP^{-1})S_1^{-1} + S_2S_1^{-1}S_2(QP^{-1})S_1^{-1} \\ &\quad + S_2S_1^{-1}S_2W_1(S_1^{-1}S_2)W_1S_1^{-1} + S_1W_1(S_1^{-1}S_2)W_1S_1^{-1} \\ &= S_2S_1^{-1} + S_1^{-*}(QP^{-1})S_1^{-1} + S_1^{-*}W_1(S_1^{-1}S_2)W_1S_1^{-1}\end{aligned}$$

because of the fact that

$$S_1 = S_1^{-*} - S_1^{-*}S_2^*S_2 = S_1^{-*} - S_2S_1^{-1}S_2 \tag{3.17}$$

(since $S_2S_1^{-1}$ is Hermitian). From (3.13) we obtain the Hermitian updating formula of the approximate solution for the Riccati equation (1.1):

$$\tilde{X} = X - L^*\left(W_1 + W_2 + \frac{V_1W_1 - W_1V_1}{2}\right)L + L^*W_1YW_1L. \tag{3.18}$$

REMARK 3.1. If we are only interested in the nonnegative definite solution for (1.1), a Hamiltonian-Schur decomposition of M as in (1.3) is not necessary. That is, the matrix $\tilde{A}$ in (3.9a) can be arbitrary. For convenience of computations we choose $V_1 \equiv V_2 \equiv 0$ in (3.9b); then (3.9b) becomes

$$\begin{aligned}\tilde{K} &= [W_1A_0 + K + A_0^*W_1] \\ &\quad + [W_1A_1 + W_2A_0 - W_1NW_1 + A_0^*W_2 + A_1^*W_1] \\ &\quad + \Bigg[W_2A_1 + A_1^*W_2 - (W_2NW_1 + W_1NW_2) \\ &\qquad -(W_1 + W_2)(A_0 + A_1)\frac{W_1^2}{2}\end{aligned}$$

$$-W_1^2(A_0^* + A_1^*)\frac{W_1 + W_2}{2} - W_2 N W_2$$

$$\left. - \frac{W_1^2 K + K W_1^2}{2} + \frac{W_1^2 K W_1^2}{4} \right]$$

$$\equiv F_1 + F_2 + F_3. \tag{3.19b}$$

Ignoring the terms of order higher than two of $\tilde{K}$ in (3.19b), we have the matrix equations

$$W_1 A_0 + A_0^* W_1 = -K \tag{3.20a}$$

and

$$W_2 A_0 + A_0^* W_2 = -(W_1 A_1 + A_1^* W_1) + W_1 N W_1 \tag{3.20c}$$

for the Hermitian matrices W_1 and W_2, respectively. The Hermitian updating formula (3.18) can be reduced as follow:

$$\tilde{X} = X - L^*(W_1 + W_2)L + L^* W_1 Y W_1 L. \tag{3.21}$$

4. QUANTITATIVE ANALYSIS OF THE LOCAL CONVERGENCE

To use the symplectic acceleration method for finding the stable solution of Riccati equation, we have to solve the equations (3.20a) and (3.20c). Suppose the norms of A_1 and K as in (3.9) are $O(\varepsilon)$. (Here ε is small in magnitude.) From (3.20a) and (3.20c) it is clearly seen that the norms of W_1 and W_2 are of order ε and ε^2, respectively. Therefore, the norm of $\tilde{K}$ in (3.19b) is of order ε^3, which is close to zero. Thus, the SAM method improves the result.

Now, we shall estimate the upper bounds for W_1 and W_2. We define the separation of A_0 and $-A_0^*$ [13] by

$$\begin{aligned}\operatorname{Sep}_F(A_0, -A_0^*) &= \operatorname{Inf}\{\|TW\|_F : \|W\|_F = 1\} \\ &\equiv \operatorname{Inf}\{\|WA_0 + A_0^* W\|_F : \|W\|_F = 1\},\end{aligned} \tag{4.1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Let

$$d \equiv \min\Big\{|a_{ii} + \bar{a}_{jj}| : i, j = 1, \dots, n,$$

$$a_{ii} \text{ and } a_{jj} \text{ are diagonal elements of } A_0\Big\}. \tag{4.2}$$

From (3.11) it is easily seen that the operator T is nonsingular if and only if $d > 0$. However, in our case the matrix A_0 is upper triangular with Re $a_{ii} < 0$. Hence, we always have $d > 0$. Furthermore, it also holds that $\operatorname{Sep}_F(A_0, -A_0^*) \leqslant d$ [13]. We now define the quantities

$$\varepsilon := \max\{\|K\|_F, \|A_1\|_F\} < 1,$$

$$p := \max\{\|A_0\|_F, \|N\|_F\} \geqslant 1,$$

$$\delta := \operatorname{Sep}_F(A_0, -A_0^*) > 0.$$

Filling in (4.1) with (3.20a) and (3.20c), respectively, we get

$$\|W_1\|_F \leqslant \varepsilon/\delta \tag{4.3}$$

and

$$\|W_2\|_F \leqslant \frac{2\varepsilon^2}{\delta^2} + \frac{\varepsilon^2 p}{\delta^3}. \tag{4.4}$$

From (3.19b), (4.3), and (4.4) follows immediately the error of F_3 in (3.19b):

$$\|F_3\|_F \leqslant \left(\frac{2\varepsilon^2}{\delta^2} + \frac{p\varepsilon^2}{\delta^3}\right)\left[2\varepsilon + \frac{2p\varepsilon}{\delta} + \frac{(p+\varepsilon)\varepsilon^2}{\delta^2} + p\left(\frac{2\varepsilon^2}{\delta^2} + \frac{p\varepsilon^2}{\delta^3}\right)\right]$$

$$+ \frac{\varepsilon^3}{\delta^2} + \frac{(p+\varepsilon)\varepsilon^3}{\delta^3} + \frac{1}{4}\frac{\varepsilon^5}{\delta^4}. \tag{4.5}$$

Suppose that

$$\max\left\{\frac{\varepsilon^3}{\delta^2}, \frac{p\varepsilon^3}{\delta^3}, \frac{p^2\varepsilon^3}{\delta^4}, \frac{p^2\varepsilon^4}{\delta^5}, \frac{p^3\varepsilon^4}{\delta^6}\right\} = \eta \ll 1. \tag{4.6}$$

Then from (4.5) we conclude that

$$\|F_3\|_F \leqslant 28.25\eta. \tag{4.7}$$

REMARK 4.1.

(1) From (4.7) we have that if $\eta \ll \varepsilon$, then the matrix $\tilde{K}$ in (3.19b) is sufficiently close to the zero matrix. For a given small magnitude η, one can easily estimate the convergence region $[-\varepsilon, \varepsilon]$ of our acceleration method by (4.6). The smaller the quantities p and δ^{-1} are, the larger the convergence region is.

(2) Actually, the quantity $\delta := \mathrm{Sep}_F(A_0, -A_0^*)$ is difficult to compute. Since d in (4.2) is an upper bound of δ, in practice d can fairly well estimate the quantity δ.

(3) From (4.5) we see that the symplectic acceleration method converges cubically. In practice, the symplectic Jacobi-like algorithm developed by Bunse-Gerstner [3] can be used as a predictor to diminish the norms of the matrix K and the strictly lower triangular matrix A_1. If the speed of convergence is linear or slower, then the SAM method can be called as a corrector to accelerate the rate of convergence.

5. COMPARISON OF FLOP COUNTS AND NUMERICAL EXAMPLES

We first compare the flop counts of the SAM method with those of one full sweep in the SJL method on a single computer:

(1) Flop counts of the SAM method:
- (i) Solving the equation (3.10a):
  - By (3.11), computing w_{ij} requires $i + j - 1$ flops. Solving for W_1, $\approx \frac{1}{2}n^3$.

  Total flop count $\approx \frac{1}{2}n^3$.
- (ii) Solving the equation (3.10b):
  - Computing NW_1 requires n^3 flops.
  - By (3.12), computing v_{ij} requires $n - i + j$ flops. Solving for V_1, $\approx \frac{1}{3}n^3$.

  Total flop count $\approx 1\frac{1}{3}n^3$.
- (iii) Solving the equation (3.10c):
  - Computing T_1, $\approx n^3$.
  - Computing V_1K, W_1A_1, W_1T_1, $\approx 2n^3$.
  - Solving for W_2, $\approx \frac{1}{2}n^3$.

Total flop count $\approx 3\frac{1}{2}n^3$.

(iv) Solving the equation (3.10d):
- Computing the matrices W_1^2, NV_1, $\approx 1\frac{1}{2}n^3$.
- Computing the lower triangular parts of the matrices V_1A_1, A_1V_1, $A_0W_1^2$, $W_1^2A_0$, NV_1W_1, T_1V_1, V_1T_1, NW_2, $\approx 2\frac{1}{3}n^3$.
- Solving for V_2, $\approx \frac{1}{3}n^3$.

Total flop count $\approx 4\frac{1}{6}n^3$.

(v) Additional computation for Hermitian updating:
- Computing $L^*W_1YW_1L$, $\approx 2\frac{1}{2}n^3$.
- Computing $L^*(W_1 + W_2 + (V_1W_1 - W_1V_1)/2)L$, $\approx 2\frac{1}{2}n^3$.

Total flop count $\approx 5n^3$.

The total number of flops of the SAM method on a single computer is about $14\frac{1}{2}n^3$.

(2) Flop counts of the SJL-method:
- Here we ignore the flops of the J-rotation, because the H-rotation is the dominant computation.
- Each H-rotation requires $16n$. A full sweep requires $n(n-1)$ J-rotations. Therefore a full sweep on a single computer requires about $16n^3$.

Next, we compare the flop counts of these two methods on a parallel computer with $n \times n$ processors. In this paper, "mesh-connected" means each processor P_{ij} $(i, j \in \{1, 2, \ldots, n\})$ is connected to its four neighbors

$$P_{i,j-1},\ P_{i,j+1},\ P_{i-1,j},\ P_{i+1,j}.$$

Here $i \pm 1$ and $j \pm 1$ are taken modulo n. In the following estimation, the major computation of the SAM method is matrix multiplication, which can be executed in n flops on an $n \times n$ mesh-connected parallel computer:

(1) Flop counts of the SAM method:

(i) Solving the equation (3.10a), $\approx 2n$.

(ii) Solving the equation (3.10b):
- Computing NW_1 requires n flops.
- Solving V_1, $\approx 2n$.

Total flop count $\approx 3n$.

(iii) Solving the equation (3.10c):
- Computing T_1, $\approx 2n$.
- Computing V_1K, W_1A_1, W_1T_1, $\approx 3n$.

- Solving W_2, $\approx 2n$.

Total flop count $\approx 7n$.

(iv) Solving the equation (3.10d):

- Computing matrices W_1^2, NV_1, $\approx 2n$.
- Computing the matrices V_1A_1, A_1V_1, $A_0W_1^2$, $W_1^2A_0$, NV_1W_1, T_1V_1, V_1T_1, NW_2, $\approx 8n$.
- Solving V_2, $\approx 2n$.

Total flop count $\approx 12n$.

(v) Additional computation for Hermitian updating:

- Computing $L^*W_1YW_1L$, $\approx 3n$.
- Computing $L^*(W_1 + W_2 + (V_1W_1 - W_1V_1)/2)L$, $\approx 3n$.

Total flop count $\approx 6n$.

The total number of flops of the SAM method on a parallel computer is about $30n$.

(2) Flop counts of the SJL method:

- Each 2×2 rotation requires 8 flops on a 2×2 subarray processor. Therefore, each rotation can propagate to the neighbor processors for every 16 flops.
- A full sweep requires $16 \times 2n = 32n$.

We summarize the above flop counts in Table 1. From them we see that the SAM method is cheaper than one full sweep of the SJL-method. In the previous section, we showed that the asymptotic convergence of the SAM method is cubic. Therefore, the SAM method should be an efficient method to accelerate the convergence of the SJL method.

Since there exist examples which cannot converge while using the asymplectic Jacobi-like method, a global convergence theorem is not possible to establish. We will now construct some special examples, essentially due to [12], which illustrate the asymptotic properties of the algorithm.

In our numerical tests, we first iterate the matrix M in (1.2) by the symplectic Jacobi-like method until the strictly J-lower triangular elements of M are sufficiently small. Then we either speed up the rate of convergence by the symplectic acceleration method or continue the Jacobi process. All

TABLE 1

FLOP COUNTS OF SJL METHOD AND SAM METHOD

	Flop count	
Computer	SAM	SJL
Single	$14\frac{1}{2}\,n^3$	$16\,n^3$
Parallel	$30\,n$	$32\,n$

numerical results are computed on an IBM PC in FORTRAN 77 with double precision. We use the following notation for Examples 4.1 and 4.2:

s := the number of Jacobi sweeps;
σ := the F-norm of the strictly J-lower triangle of the current M;
γ := the F-norm of the error matrix by applying the solution X to the Riccati equation;
i(IC) := after i Jacobi sweeps we perform the reordering (interchanging) algorithm of eigenvalues of M;
iSAM := after the step $i-1$(IC) we perform the symplectic acceleration method.

EXAMPLE 5.1 ($n = 5$). Let

$$M_\alpha = S^*\begin{bmatrix} D+\alpha U & \alpha N \\ 0 & -D^* - \bar{\alpha} U^* \end{bmatrix} S,$$

where $D = \operatorname{diag}(-1, -2, -3, -4, -5)$; N and U are, respectively, Hermitian and strictly upper triangular with entries randomly generated between ± 1; and S is unitary symplectic. For different α ($\alpha = 0.01, 0.1, 1, 10$), we have the numerical results given in Table 2.

EXAMPLE 5.2. Let

$$A = \begin{bmatrix} A_{11} & A_{12} & & 0 & \\ & A_{22} & A_{23} & & \\ & & A_{33} & A_{34} & \\ & 0 & & A_{44} & -1 \\ & & & & -1 \end{bmatrix},$$

where

$$A_{ii} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}, \qquad A_{i,i+1} = \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix},$$

and $N = \operatorname{diag}(1,0,1,\dots,0,1)$, $K = \operatorname{diag}(0,10,0,\dots,10,0)$. The numerical result is given in Table 3.

From these two examples, we see that the symplectic acceleration method at least doubles the accuracy of the eigenvalues and significantly improves the

TABLE 2
JACOBI ALGORITHM + SYMPLECTIC ACCELERATION METHOD

α	s	σ	γ
0.01	0	5.3E + 00	3.8E + 00
	1	2.0E + 00	1.5E + 01
	2	1.8E − 01	2.3E − 01
	3	3.7E − 04	2.7E − 03
	3(IC)	3.2E − 04	3.4E − 05
	4	8.2E − 08	4.5E − 07
	5	5.7E − 15	1.1E − 14
	5(IC)	5.4E − 15	9.3E − 15
	4SAM	1.2E − 10	1.9E − 10
0.1	0	5.6E + 00	3.2E + 00
	1	2.0E + 00	1.8E − 01
	2	2.0E − 01	2.3E − 01
	3	1.6E − 03	2.0E − 02
	4	1.2E − 05	1.4E − 06
	4(IC)	1.2E − 05	5.4E − 05
	5	1.9E − 08	1.8E − 08
	6	6.4E − 12	8.3E − 12
	7	5.8E − 17	4.4E − 17
	7(IC)	5.6E − 17	6.2E − 17
	5SAM	2.3E − 12	2.0E − 11
1	0	5.7E + 00	4.2E + 00
	1	3.2E + 00	5.3E + 01
	2	5.7E − 01	6.1E − 01
	3	3.9E − 02	2.4E − 01
	4	2.4E − 04	1.2E − 03
	4(IC)	1.7E − 04	8.8E − 04
	5	4.2E − 06	1.2E − 06
	6	3.0E − 08	2.3E − 08
	7	2.9E − 11	1.2E − 11
	7(IC)	2.5E − 11	5.8E − 11
	5SAM	4.0E − 10	1.7E − 09
10	0	2.1E + 01	1.4E + 01
	1	7.1E + 00	1.3E + 01
	5	4.2E − 01	1.8E + 00
	7	5.0E − 02	7.2E − 02
	8	5.6E − 03	1.6E − 03
	9	2.2E − 05	9.3E − 06
	9(IC)	1.7E − 05	1.6E − 05

TABLE 2
(CONTINUED)

α	s	σ	γ
	10	2.3E − 07	4.4E − 08
	11	6.9E − 10	7.5E − 10
	12	1.2E − 12	4.4E − 12
	13	3.2E − 16	8.2E − 16
	13(IC)	2.8E − 16	1.7E − 16
	10SAM	2.7E − 14	1.4E − 11

TABLE 3
JACOBI ALGORITHM + SYMPLECTIC ACCELERATION METHOD

s	σ	γ
0	2.030E + 01	2.018E + 01
1	1.322E + 01	1.128E + 02
2	2.913E + 00	3.937E + 02
3	4.124E + 00	3.652E + 01
4	1.865E + 00	3.259E + 01
5	8.501E − 01	1.123E + 01
6	4.770E − 01	2.024E + 00
7	1.184E − 01	2.007E + 00
8	1.226E − 02	2.545E − 02
9	1.816E − 04	2.606E − 04
9(IC)	9.252E − 05	2.188E − 04
10	2.344E − 07	1.677E − 08
11	5.357E − 12	5.448E − 12
11(IC)	3.718E − 12	5.372E − 12
10SAM	6.325E − 11	1.303E − 10

convergence speed of the Jacobi algorithm. The algorithm of Bunse-Gerstner [2] has a better convergence rate (between linear and quadratic) than the others. The combination of that algorithm and our acceleration method would be the most powerful strategy.

REFERENCES

1 G. S. Ammar and V. Mehrmann, On Hamiltonian and symplectic Hessenberg forms, *Linear Algebra Appl.* 149:55–72 (1991).

2 A. Bunse-Gerstner and V. Mehrmann, A symplectic *QR* like algorithm for the solution of the real algebraic Riccati equation, *IEEE Trans. Automat. Control* AC-31(12):1104–1113 (1986).
3 A. Bunse-Gerstner, On the Hamiltonian-Schur decomposition of a Hamiltonian matrix, to appear.
4 R. Byers, A Hamiltonian-Jacobi algorithm, presented at SIAM Conference on Control in the '90s, May 1989.
5 J. L. Casti, *Dynamical Systems and their Applications: Linear Theory*, Academic, New York, 1977.
6 Roy O. Davies and J. J. Modi, A direct method for computing eigenproblem solutions on a parallel computer, *Linear Algebra Appl.* 77:61–74 (1986).
7 P. J. Eberlein, On the Schur decomposition of a matrix for parallel computation, *IEEE Trans. Comput.* 36:167–174 (1987).
8 G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins U.P., Baltimore, 1989.
9 A. J. Laub, A Schur method for solving algebraic Riccati equations, *IEEE Trans. Automat. Control* AC-24(13):913–925 (1979).
10 D. P. O'Leary and G. W. Stewart, Data-Flow Algorithms for Parallel Matrix Computations, Computer Science Tech. Rep. 1366, Univ. of Maryland, 1984.
11 C. Paige and C. F. Van Loan, A Schur decomposition for Hamiltonian matrices, *Linear Algebra Appl.* 41:11–32 (1981).
12 G. W. Stewart, A Jacobi-like algorithm for computing the Schur decomposition of a non-Hermitian matrix, *SIAM J. Statist. Comput.* 6(4):853–864 (1985).
13 G. W. Stewart, On the sensitivity of the eigenvalue problem $Ax = \lambda Bx$, *SIAM J. Numer. Anal.* 9(4):669–686 (1972).

Received 11 October 1990; final manuscript accepted 25 September 1992