Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2007, Article ID 21850, 10 pages
doi:10.1155/2007/21850

Research Article

On the Solution of the Rational Matrix Equation $X = Q + LX^{-1}L^T$

Peter Benner[1] and Heike Faßbender[2]

[1] *Fakultät für Mathematik, Technische Universität Chemnitz, 09107 Chemnitz, Germany*
[2] *Institut Computational Mathematics, Technische Universität Braunschweig, 38106 Braunschweig, Germany*

Received 30 September 2006; Revised 9 February 2007; Accepted 22 February 2007

Recommended by Paul Van Dooren

We study numerical methods for finding the maximal symmetric positive definite solution of the nonlinear matrix equation $X = Q + LX^{-1}L^T$, where Q is symmetric positive definite and L is nonsingular. Such equations arise for instance in the analysis of stationary Gaussian reciprocal processes over a finite interval. Its unique largest positive definite solution coincides with the unique positive definite solution of a related discrete-time algebraic Riccati equation (DARE). We discuss how to use the butterfly *SZ* algorithm to solve the DARE. This approach is compared to several fixed-point and doubling-type iterative methods suggested in the literature.

1. INTRODUCTION

The nonlinear matrix equation

$$X = f(X) \quad \text{with } f(X) = Q + LX^{-1}L^T, \tag{1}$$

where $Q = Q^T \in \mathbb{R}^{n\times n}$ is positive definite and $L \in \mathbb{R}^{n\times n}$ is nonsingular, arises in the analysis of stationary Gaussian reciprocal processes over a finite interval. The solutions of certain 1D stochastic boundary value problems are reciprocal processes. For instance, the steady state distribution of the temperature along a heated ring or beam subjected to random loads along its length can be modeled in terms of such reciprocal processes. A different example is a ship surveillance problem: given a Gauss-Markov state-space model of the ship's trajectory, it is desired to assign a probability distribution not only to the initial state, but also to the final state, corresponding to some predictive information about the ship's destination. This has the effect of modeling the trajectory as a reciprocal process. For references to these examples see, for example, [1].

The problem considered here is to find the (unique) largest positive definite symmetric solution X_+ of (1). This equation has been considered, for example, in [2–7]. In [2], the set of Hermitian solutions of (1) is characterized in terms of the spectral factors of the matrix Laurent polynomial $\mathcal{L}(z) = Q + Lz - L^T z^{-1}$. These factors are related to the Lagrangian deflating subspace of the matrix pencil

$$G - \lambda H = \begin{bmatrix} L^T & 0 \\ -Q & I \end{bmatrix} - \lambda \begin{bmatrix} 0 & I \\ L & 0 \end{bmatrix}. \tag{2}$$

In particular, one can conclude from the results in [2, Section 2] that this matrix pencil does not have any eigenvalues on the unit circle and that the spectral radius $\rho(X_+^{-1}L^T)$ is less than 1 as $\left[\begin{smallmatrix} I \\ X_+ \end{smallmatrix}\right]$ spans the stable Lagrangian deflating subspace of $G - \lambda H$. Alternatively, one could rewrite (1) as the discrete Lyapunov equation $X_+ - (X_+^{-1}L^T)^T X_+ (X_+^{-1}L^T) = Q$. As Q and X_+ are positive definite, we get $\rho(X_+^{-1}L^T) < 1$ from the discrete version of the Lyapunov stability theorem (see, e.g., [8, page 451]). Moreover, it is shown in [2] that the unique largest positive definite solution of (1) coincides with the unique positive definite solution of a related Riccati equation. For this, it is noted in [2] that if X solves (1), then it also obeys the equation

$$X = f(f(X)) = Q + F\left(R^{-1} + X^{-1}\right)^{-1} F^T \tag{3}$$

with $F = LL^{-T}$ and $R = L^T Q^{-1} L = R^T$ positive definite. Using the Sherman-Morrison-Woodbury formula to derive an expression for $(R^{-1} + X^{-1})^{-1}$, we obtain

$$\mathcal{DR}(X) = Q + FXF^T - FX(X+R)^{-1}XF^T - X, \tag{4}$$

$$0 = Q + FX\left(I + R^{-1}X\right)^{-1}F^T - X, \tag{5}$$

a discrete-time algebraic Riccati equation (DARE). Because (F, I) is controllable and (F, Q) is observable, a unique stabilizing positive definite solution X_* exists [9, Theorem 13.1.3]. This unique solution coincides with that solution of (1) which one is interested in. DAREs appear not only in the context presented, but also in numerous procedures for analysis, synthesis, and design of control and estimation systems with H_2 or H_∞ performance criteria, as well as in other branches of applied mathematics and engineering, see, for example, [9–13].

In [2], essentially three ideas for solving (1) have been proposed. The straightforward one is a basic iterative algorithm that converges to the desired positive definite solution X_+ of (1). Essentially, the algorithm interprets (1) as a fixed-point equation and iterates $X_{i+1} = f(X_i)$; see Section 2.1 for more details.

The second idea is to compute the desired solution from the stable Lagrangian deflating subspace of $G - \lambda H$. If we can compute $Y_1, Y_2 \in \mathbb{R}^{n\times n}$ such that the columns of $\left[\begin{smallmatrix} Y_1 \\ Y_2 \end{smallmatrix}\right]$ span the desired deflating subspace of $G - \lambda H$, then $X_\diamond = -Y_2 Y_1^{-1}$ is the desired solution of (1). (In order to distinguish the unique largest positive definite symmetric solution of (1) obtained by the different algorithms discussed, we will use different subscripts for each approach.)

The third idea is to compute the desired solution via the unique solution X_* of the DARE. The solution X_* can be found by direct application of Newton's method for DAREs [3, 9, 14, 15]. However, comparison with the basic fixed-point iteration is not favorable [3, Section 5]. Therefore, this approach of solving the DARE is not considered here. Instead we will compute its solution via the stable deflating subspace of an associated matrix pencil. As R is positive definite, we can define

$$M - \lambda N = \begin{bmatrix} F^T & 0 \\ Q & I \end{bmatrix} - \lambda \begin{bmatrix} I & -R^{-1} \\ 0 & F \end{bmatrix}. \tag{6}$$

As (F, I) is controllable, (F, Q) is observable, and Q and R^{-1} are positive definite, $M - \lambda N$ has no eigenvalues on the unit circle; see, for example, [9]. It is then easily seen that $M - \lambda N$ has precisely n eigenvalues in the open unit disk and n outside. Moreover, the Riccati solution X_* can be given in terms of the deflating subspace of $M - \lambda N$ corresponding to the n eigenvalues $\lambda_1, \ldots, \lambda_n$ inside the unit disk using the relation

$$\begin{bmatrix} F^T & 0 \\ Q & I \end{bmatrix} \begin{bmatrix} I \\ -X \end{bmatrix} = \begin{bmatrix} I & -R^{-1} \\ 0 & F \end{bmatrix} \begin{bmatrix} I \\ -X \end{bmatrix} \Lambda, \tag{7}$$

where $\Lambda \in \mathbb{R}^{n\times n}$ with the spectrum $\sigma(\Lambda) = \{\lambda_1, \ldots, \lambda_n\}$. Therefore, if we can compute $Y_1, Y_2 \in \mathbb{R}^{n\times n}$ such that the columns of $\left[\begin{smallmatrix} Y_1 \\ Y_2 \end{smallmatrix}\right]$ span the desired deflating subspace of $M - \lambda N$, then $X_* = -Y_2 Y_1^{-1}$ is the desired solution of the DARE (4). See, for example, [9, 15, 16] and the references therein.

Hence, two of the ideas stated in [2] how to solve (1) can be interpreted as the numerical computation of a deflating subspace of a matrix pencil $A - \lambda B$. This is usually carried out by a procedure like the QZ algorithm. Applying the numerically backward stable QZ algorithm to a matrix pencil results in a general $2n \times 2n$ matrix pencil in generalized Schur form from which the eigenvalues and deflating subspaces can be determined.

Both matrix pencils to be considered here ($G - \lambda H$ and $M - \lambda N$) have a symplectic spectrum, that is, their eigenvalues appear in reciprocal pairs λ, λ^{-1}. They have exactly n eigenvalues inside the unit disk, and n outside. Sorting the eigenvalues in the generalized Schur form such that the eigenvalues inside the unit disk are contained in the upper left $n \times n$ block, the desired deflating subspace can easily be read off and the solution $X_\diamond$, respectively X_*, can be computed. (This method results in the popular generalized Schur vector method for solving DAREs [17].) Due to roundoff errors unavoidable in finite-precision arithmetic, the computed eigenvalues will not in general come in pairs $\{\lambda, \lambda^{-1}\}$, although the exact eigenvalues have this property. Even worse, small perturbations may cause eigenvalues close to the unit circle to cross the unit circle such that the number of true and computed eigenvalues inside the open unit disk may differ. Moreover, the application of the QZ algorithm to a $2n \times 2n$ matrix pencil is computationally quite expensive. The usual initial reduction to Hessenberg-triangular form requires about $70n^3$ flops plus $24n^3$ for accumulating the Z matrix; each iteration step requires about $88n^2$ flops for the transformations and $136n^2$ flops for accumulating Z; see, for example, [18]. An estimated $40n^3$ flops are necessary for ordering the generalized Schur form. This results in a total cost of roughly $415n^3$ flops, employing standard assumptions about convergence of the QZ iteration (see, e.g., [19, Section 7.7]).

The use of the QZ algorithm is prohibitive here not only due to the fact that it does not preserve the symplectic spectra, but also due to the costly computation. More efficient methods have been proposed which make use of the following observation: $M - \lambda N$ of the form (6) is a symplectic matrix pencil. A symplectic matrix pencil $M - \lambda N$, $M, N \in \mathbb{R}^{2n\times 2n}$, is defined by the property

$$MJM^T = NJN^T, \tag{8}$$

where

$$J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix} \tag{9}$$

and I_n is the $n \times n$ identity matrix. The nonzero eigenvalues of a symplectic matrix pencil occur in reciprocal pairs: if λ is an eigenvalue of $M - \lambda N$ with left eigenvector x, then $\overline{\lambda^{-1}}$ is an eigenvalue of $M - \lambda N$ with right eigenvector $(Jx)^H$. Hence, as we are dealing with real symplectic pencils, the finite generalized eigenvalues always occur in pairs if they are real or purely imaginary or in quadruples otherwise. Although $G - \lambda H$ as in (2) is not a symplectic matrix pencil, it can be transformed into a very special symplectic pencil $\hat{G} - \lambda \hat{H}$ as noted in [5]. This symplectic pencil $\hat{G} - \lambda \hat{H}$ allows the use of a doubling algorithm to compute the solution $X_\diamond$. These methods originate from the fixed-point iteration derived from the DARE. Instead of generating the usual sequence $\{X_k\}$, doubling algorithms generate $\{X_{2^k}\}$. This class of methods attracted much interest in the 1970s and 1980s,

see [18] and the references therein. After having been abandoned for the past decade, they have recently been revived by a series of papers, for example, [5, 20]. To be more specific, define

$$\mathcal{N}(\hat{G},\hat{H}) = \left\{ [G_\star, H_\star] : G_\star, H_\star \in \mathbb{R}^{2n\times 2n}, \right. \\ \left. \operatorname{rank}[G_\star, H_\star] = 2n, [G_\star, H_\star] \begin{bmatrix} \hat{H} \\ -\hat{G} \end{bmatrix} = 0 \right\}. \tag{10}$$

Since $\operatorname{rank}\left[\begin{smallmatrix} \hat{H} \\ -\hat{G} \end{smallmatrix}\right] \le 2n$, it follows that $\mathcal{N}(\hat{G},\hat{H}) \neq \emptyset$. For any given $[G_\star, H_\star] \in \mathcal{N}(\hat{G},\hat{H})$, define

$$\check{G} = G_\star \hat{G}, \qquad \check{H} = H_\star \hat{H}. \tag{11}$$

The transformation

$$\hat{G} - \lambda\hat{H} \longrightarrow \check{G} - \lambda\check{H} \tag{12}$$

is called a doubling transformation. The doubling algorithm consists of applying the doubling transformation repeatedly. An important feature of this kind of transformation is that it is structure preserving [21], eigenspace preserving [21–23], and eigenvalue squaring. In [5], an appropriate doubling transformation for the symplectic pencil $\hat{G}-\lambda\hat{H}$ is given. The resulting algorithm has very nice numerical behavior, with a quadratic convergence rate, low computational cost, and good numerical stability. Essentially, the same algorithm was proposed in [6] using a different motivation. See Section 2.2 for more details. Alternatively, a doubling algorithm could be applied directly to the DARE (5). This is discussed in Section 2.2.1.

Here we propose to compute the desired solution X_* via an approximate solution of the DARE (4) by the (butterfly) *SZ* algorithm applied to the corresponding symplectic pencil [24–26]. This algorithm is a fast, reliable, and structure-preserving algorithm for computing the stable deflating subspace of the symplectic matrix pencil $M - \lambda N$ (6) associated with the DARE. The matrix pencil $M - \lambda N$ is first reduced to the so-called symplectic butterfly form, which is determined by only $4n - 1$ parameters. By exploiting this special reduced form and the symplecticity, the *SZ* algorithm is fast and efficient; in each iteration step only $O(n)$ arithmetic operations are required instead of $O(n^2)$ arithmetic operations for a *QZ* step. We thus save a significant amount of work. Of course, the accumulation of the Z matrix requires $O(n^2)$ arithmetic operations as in the *QZ* step. Moreover, by forcing the symplectic structure, the above-mentioned problems of the *QZ* algorithm are avoided. See Section 3 for more details.

Any approximate solution $\tilde{X}$ computed, for example, with one of the methods described above, can be improved via defect correction. This is considered in Section 4. Finally, in Section 5 we compare the different algorithms for solving (1) discussed here.

2. ITERATIVE ALGORITHMS FOR (1)

2.1. The fixed-point iteration

As suggested in [2], (1) can be solved directly by turning it into a fixed-point iteration

$$X_{i+1} = f(X_i) = Q + LX_i^{-1}L^T \tag{13}$$

with initial condition $X_0 = Q$. In [2], it is shown that the sequence $\{X_i\}$ converges to the unique positive definite solution X_+ of (1). This convergence is robust as for any positive ϵ there exists a neighborhood Υ of X_+ such that for any initial condition $X_0 \in \Upsilon$, the sequence generated by (13) remains in a ball of radius ϵ centered in X_+ and converges to X_+. Moreover, the sequence generated by (13) converges to X_+ for any positive definite initial condition X_0 as well as for any initial condition such that $X_0 \le -LQ^{-1}L^T$. The convergence rate is related to the spectral radius $\rho(X_+^{-1}L^T)$. The convergence is linear, but, if $\rho(X_+^{-1}L^T)$ is close to 1, the convergence may be very slow. See also [3, Section 2].

An inverse free variant of the fixed-point iteration is possible. However, the algorithm is not always convergent, [3, last paragraph, Section 3].

Our implementation of the fixed-point iteration first computes the Cholesky decomposition $X_i = C_iC_i^T$, next the linear system $L = C_iB_i$ is solved (i.e., $B_i = LC_i^{-1}$) and finally $X_{i+1} = Q + B_iB_i^T$ is computed. The total flop count for one iteration step is therefore $(7/3)n^3$ flops, as the first step involves about $n^3/3$, the second one n^3, and the last one n^3 flops.

In many applications, rather than the solutions of matrix equations themselves, their factors (such as Cholesky or full-rank factors) are needed; see, for example, [18, 27]. Moreover, subsequent calculations can often be performed using the factors which usually have a much better condition number. Therefore, it may be desirable to have such a method that computes such a factor directly also for (1) without ever forming the solution explicitly. Such a method can also easily be derived based on the fixed point iteration (1). As all iterates are positive definite, it is natural here to use their Cholesky factors. Assuming we have a Cholesky factorization $X_i = Y_iY_i^T$, then the Cholesky factor of

$$\begin{aligned} X_{i+1} &= Q + LX_i^{-1}L^T = CC^T + L\left(Y_iY_i^T\right)^{-1}L^T \\ &= [C, LY_i^{-T}][C, LY_i^{-T}]^T \end{aligned} \tag{14}$$

can be obtained from the leading $n \times n$ submatrix of the L-factor of the LQ factorization of

$$[C,\ LY_i^{-T}] = \left[\begin{array}{cc} \diagdown & \square \end{array}\right]. \tag{15}$$

Note that the Q-factor is not needed as it cancels:

$$X_{i+1} = Y_{i+1}Y_{i+1}^T = L_iQ_iQ_i^TL_i^T = [\hat{L}_i, 0][\hat{L}_i, 0]^T = \hat{L}_i\hat{L}_i^T. \tag{16}$$

An LQ factorization for the specially structured matrix in (15) is implemented in the SLICOT[1] subroutine `MB04JD`. Employing this, the factorized fixed-point iteration yielding the sequence Y_i of Cholesky factors of X_i requires $3n^3$ flops per iteration and is thus slightly more expensive than the fixed-point iteration itself. Additionally, $n^3/3$ flops for the initial Cholesky factorization of Q are needed.

2.2. The doubling algorithm

As already observed in [2], the solution X of (1),

$$X = Q + LX^{-1}L^T, \tag{17}$$

satisfies

$$G\begin{bmatrix} I \\ X \end{bmatrix} = H\begin{bmatrix} I \\ X \end{bmatrix} W \tag{18}$$

for some matrix $W \in \mathbb{R}^{n\times n}$, where

$$G = \begin{bmatrix} L^T & 0 \\ -Q & I \end{bmatrix}, \qquad H = \begin{bmatrix} 0 & I \\ L & 0 \end{bmatrix}. \tag{19}$$

Hence, the desired solution X can be computed via an appropriate deflating subspace of $G - \lambda H$. This could be done by employing the QZ algorithm. But the following idea suggested in [5] achieves a much faster algorithm.

Assume that X is the unique symmetric positive definite solution of (1). Then it satisfies (18) with $W = X^{-1}L^T$. Let

$$\hat{L} = LQ^{-1}L, \qquad \hat{Q} = Q + LQ^{-1}L^T, \qquad \hat{P} = L^TQ^{-1}L, \qquad \hat{X} = X + \hat{P}. \tag{20}$$

Then it follows that

$$\hat{G}\begin{bmatrix} I \\ \hat{X} \end{bmatrix} = \hat{H}\begin{bmatrix} I \\ \hat{X} \end{bmatrix} W^2, \tag{21}$$

where

$$\hat{G} = \begin{bmatrix} \hat{L}^T & 0 \\ \hat{Q} + \hat{P} & -I \end{bmatrix}, \qquad \hat{H} = \begin{bmatrix} 0 & I \\ \hat{L} & 0 \end{bmatrix}. \tag{22}$$

The pencil $\hat{G} - \lambda\hat{H}$ is symplectic as $\hat{G}J\hat{G}^T = \hat{H}J\hat{H}^T$. (As G and H are not symplectic themselves, the butterfly SZ algorithm described in the next section cannot be employed directly in order to compute the desired deflating subspace of $\hat{G} - \lambda\hat{H}$.) It is easy to see that $\hat{X}$ satisfies (21) if and only if the equation

$$\hat{X} = (\hat{Q} + \hat{P}) - \hat{L}\hat{X}^{-1}\hat{L}^T \tag{23}$$

has a symmetric positive definite solution $\hat{X}$.

[1] See http://www.slicot.org.

In [5], it is suggested to use a doubling algorithm to compute the solution $\hat{X}$ of (21). An appropriate doubling transformation for the symplectic pencil (21) is given. Applying this special doubling transformation repeatedly, the following structure-preserving doubling algorithm (SDA) arises:

for $i = 0, 1, 2, \ldots$

$$\begin{aligned} L_{i+1} &= L_i^T(Q_i - P_i)^{-1}L_i^T, \\ Q_{i+1} &= Q_i - L_i(Q_i - P_i)^{-1}L_i^T, \\ P_{i+1} &= P_i + L_i^T(Q_i - P_i)^{-1}L_i, \end{aligned} \tag{24}$$

until convergence

with

$$L_0 = \hat{L}, \qquad Q_0 = \hat{Q} + \hat{P}, \qquad P_0 = 0. \tag{25}$$

As the matrix $Q_i - P_i$ is positive definite for all i [5], the iterations above are all well defined. The sequence Q_{i+1} will converge to $\hat{X}$. Thus, the unique symmetric positive definite solution to (1) can be obtained by computing

$$X_\diamond = \hat{X} - \hat{P}. \tag{26}$$

Essentially, the same algorithm was proposed in [6] using a different motivation.

Both papers [5, 6] point out that this algorithm has very nice numerical behavior, with a quadratic convergence rate, low computational cost, and good numerical stability. The algorithm requires about $6.3n^3$ arithmetic operations per iteration step when implemented as follows: first a Cholesky decomposition of $Q_i - P_i = C_i^TC_i$ is computed ($n^3/3$ arithmetic operations), then $L_i^TC_i^{-1}$ and $C_i^{-T}L_i^T$ are computed (both steps require n^3 arithmetic operations), finally L_{i+1}, Q_{i+1}, P_{i+1} are computed using these products ($4n^3$ arithmetic operations if the symmetry of Q_{i+1} and P_{i+1} is exploited). Hence, one iteration step requires $(19/3)n^3$ arithmetic operations.

Despite the fact that a factorized version of the doubling iteration for DAREs has been around for about 30 years, see [18] and the references therein, the SDA (24) for (1) cannot easily be rewritten to work on a Cholesky factor of Q_i due to the minus sign in the definition of the Q_i's.

2.2.1. A doubling algorithm for (6)

As explained in the introduction, the solution X of (1) can also be obtained from the deflating subspace of the pencil (6). In [28], a doubling algorithm for computing this solution has been developed as an acceleration scheme for the fixed-point iteration from (5),

$$\begin{aligned} X_{k+1} &= Q + FX_k(I + R^{-1}X_k)^{-1}F^T \\ &= Q + LL^{-T}X_k(I + L^{-1}QL^{-T}X_k)^{-1}L^{-1}L^T. \end{aligned} \tag{27}$$

Using the notation introduced here, that algorithm (here called SDA-DARE) can be stated as follows (see [20]):

initialize $A_0 = LL^{-T} = F,\ G_0 = L^{-1}QL^{-T} = R^{-1},\ X_0 = Q$

for $i = 0, 1, 2, \ldots$

$$W = I + G_i X_i \tag{28}$$

solve for $V_1 : WV_1 = A_i$, (29)

solve for $V_2 : V_2 W^T = G_i$, (30)

$$G_{i+1} = G_i + A_i V_2 A_i^T, \tag{31}$$

$$X_{i+1} = X_i + V_1^T X_i A_i, \tag{32}$$

$$A_{i+1} = A_i V_1 \tag{33}$$

until convergence.

The algorithm requires $(44/3)n^3$ flops: the matrix multiplications in (28) and (33) require about $2n^3$ flops each, the computation of the symmetric matrices in (31) and (32) comes at about $3n^3$ flops, the decomposition of W costs $(2/3)n^3$ flops, and the computations in (29) and (30) require $2n^3$ flops each. Its quadratic convergence properties are analyzed in [20]. Compared to the doubling algorithm discussed in the previous section, this algorithm is more costly: $(19/3)n^3$ flops versus $(44/3)n^3$ flops, but it avoids using the inverse of Q. The inverse of L is used instead.

Like the fixed-point iteration, the doubling algorithm for DAREs can be rewritten in terms of (Cholesky) factors so that the iterates resulting from (32) in factorized form converge to a (Cholesky) factor of the solution. This has been known for decades (see [18] and the references therein), a slightly refined variant that computes a low-rank factor of the solution in case of rank deficiency of X has recently been proposed in [29]. In contrast to the usual situation for DAREs where G and Q are often of low rank, no efficiency gain can be expected from such an implementation in our situation as G, Q, and X are all full-rank matrices.

3. THE BUTTERFLY *SZ* ALGORITHM

As shown in [2], instead of solving (1) one can solve the related DARE (4),

$$\mathcal{DR}(X) = Q + FXF^T - FX(X+R)^{-1}XF^T - X. \tag{34}$$

One approach to solve this equation is via computing the stable deflating subspace of the matrix pencil from (6), that is,

$$M - \lambda N = \begin{bmatrix} F^T & 0 \\ Q & I \end{bmatrix} - \lambda \begin{bmatrix} I & -R^{-1} \\ 0 & F \end{bmatrix}. \tag{35}$$

Here we propose to use the butterfly *SZ* algorithm for computing the deflating subspace of $M - \lambda N$. The butterfly *SZ* algorithm [25, 26] is a fast, reliable, and efficient algorithm especially designed for solving the symplectic eigenproblem for a symplectic matrix pencil $\widetilde{M} - \lambda\widetilde{N}$ in which both matrices are symplectic; that is, $\widetilde{M}J\widetilde{M}^T = \widetilde{N}J\widetilde{N}^T = J$. The above symplectic matrix pencil

$$\begin{bmatrix} F^T & 0 \\ Q & I \end{bmatrix} - \lambda \begin{bmatrix} I & -R^{-1} \\ 0 & F \end{bmatrix} = \begin{bmatrix} L^{-1}L^T & 0 \\ Q & I \end{bmatrix} - \lambda \begin{bmatrix} I & -L^{-1}QL^{-T} \\ 0 & LL^{-T} \end{bmatrix} \tag{36}$$

can be rewritten (after premultiplying by $\left[\begin{smallmatrix} L & 0 \\ 0 & L^{-1} \end{smallmatrix}\right]$) as

$$\widetilde{M} - \lambda\widetilde{N} = \begin{bmatrix} L^T & 0 \\ L^{-1}Q & L^{-1} \end{bmatrix} - \lambda \begin{bmatrix} L & -QL^{-T} \\ 0 & L^{-T} \end{bmatrix}, \tag{37}$$

where both matrices $\widetilde{M} = \widetilde{N}^T$ are symplectic. In [25, 26] it is shown that for the symplectic matrix pencil $\widetilde{M} - \lambda\widetilde{N}$ there exist numerous symplectic matrices Z and nonsingular matrices S which reduce $\widetilde{M} - \lambda\widetilde{N}$ to a symplectic butterfly pencil $A - \lambda B$:

$$S(\widetilde{M} - \lambda\widetilde{N})Z = A - \lambda B = \begin{bmatrix} C & D \\ 0 & C^{-1} \end{bmatrix} - \lambda \begin{bmatrix} 0 & -I \\ I & T \end{bmatrix}, \tag{38}$$

where C and D are diagonal matrices, and T is a symmetric tridiagonal matrix. (More generally, not only the symplectic matrix pencil in (37), but any symplectic matrix pencil $\widetilde{M} - \lambda\widetilde{N}$ with symplectic matrices $\widetilde{M}$, $\widetilde{N}$ can be reduced to a symplectic butterfly pencil). This form is determined by just $4n - 1$ parameters. The symplectic matrix pencil $A - \lambda B$ is called a symplectic butterfly pencil. If T is an unreduced tridiagonal matrix, then the butterfly pencil is called unreduced. If any of the $n - 1$ subdiagonal elements of T are zero, the problem can be split into at least two problems of smaller dimension, but with the same symplectic butterfly structure.

Once the reduction to a symplectic butterfly pencil is achieved, the *SZ* algorithm is a suitable tool for computing the eigenvalues/deflating subspaces of the symplectic pencil $A - \lambda B$ [25, 26]. The *SZ* algorithm preserves the symplectic butterfly form in its iterations. It is the analogue of the *SR* algorithm (see [24, 26]) for the generalized eigenproblem, just as the *QZ* algorithm is the analogue of the *QR* algorithm for the generalized eigenproblem. Both are instances of the *GZ* algorithm [30].

Each iteration step begins with an unreduced butterfly pencil $A - \lambda B$. Choose a spectral transformation function q and compute a symplectic matrix $\check{Z}$ such that

$$\check{Z}^{-1} q(A^{-1}B) e_1 = \alpha e_1 \tag{39}$$

for some scalar α. Then transform the pencil to

$$\widetilde{A} - \lambda\widetilde{B} = (A - \lambda B)\check{Z}. \tag{40}$$

This introduces a bulge into the matrices $\widetilde{A}$ and $\widetilde{B}$. Now transform the pencil to

$$\widehat{A} - \lambda\widehat{B} = S^{-1}(\widetilde{A} - \lambda\widetilde{B})\widetilde{Z}, \tag{41}$$

where $\widehat{A} - \lambda\widehat{B}$ is again of symplectic butterfly form. S and $\widetilde{Z}$ are symplectic, and $\widetilde{Z}e_1 = e_1$. This concludes the iteration. Under certain assumptions, it can be shown that the butterfly *SZ* algorithm converges cubically. The needed assumptions are technically involved and follow from the *GZ* convergence theory developed in [30]. The convergence theorem says roughly that if the eigenvalues are separated, and the shifts converge, and the condition numbers of the accumulated transformation matrices remain bounded, then the

SZ algorithm converges. For a detailed discussion of the butterfly *SZ* algorithm see [25, 26].

Hence, in order to compute an approximate solution of the DARE (4) by the butterfly *SZ* algorithm, first the symplectic matrix pencil $\widetilde{M} - \lambda\widetilde{N}$ as in (37) has to be formed, then the symplectic matrix pencil $A - \lambda B$ as in (38) is computed. That is, symplectic matrices Z_0 and S_0 are computed such that

$$A - \lambda B := S_0^{-1}\widetilde{M}Z_0 - \lambda S_0^{-1}\widetilde{N}Z_0 \tag{42}$$

is a symplectic butterfly pencil. Using the butterfly *SZ* algorithm, symplectic matrices Z_1 and S_1 are computed such that

$$S_1^{-1}AZ_1 - \lambda S_1^{-1}BZ_1 \tag{43}$$

is a symplectic butterfly pencil and the symmetric tridiagonal matrix T in the lower right block of $S_1^{-1}BZ_1$ is reduced to quasidiagonal form with 1×1 and 2×2 blocks on the diagonal. The eigenproblem decouples into a number of simple 2×2 or 4×4 generalized symplectic eigenproblems. Solving these subproblems, finally symplectic matrices Z_2, S_2 are computed such that

$$\begin{aligned} \breve{A} &= S_2^{-1}S_1^{-1}AZ_1Z_2 = \begin{bmatrix} \phi_{11} & \phi_{12} \\ 0 & \phi_{22} \end{bmatrix}, \\ \breve{B} &= S_2^{-1}S_1^{-1}BZ_1Z_2 = \begin{bmatrix} \psi_{11} & \psi_{12} \\ 0 & \psi_{22} \end{bmatrix}, \end{aligned} \tag{44}$$

where the eigenvalues of the matrix pencil $\phi_{11} - \lambda\psi_{11}$ are precisely the n stable generalized eigenvalues. Let $Z = Z_0Z_1Z_2$. Partitioning Z conformably,

$$Z = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}, \tag{45}$$

the Riccati solution X_* is found by solving a system of linear equations:

$$X_* = -Z_{21}Z_{11}^{-1}. \tag{46}$$

This algorithm requires about $195n^3$ arithmetic operations in order to compute the solution of the Riccati equation (and is therefore cheaper than the *QZ* algorithm which requires about $422n^3$ arithmetic operations). The cost of the different steps of the approach described above are given as follows. The computation of $L^{-1}Q$ and L^{-1} using an LU decomposition of L requires about $(14/3)n^3$ arithmetic operations. A careful flop count reveals that the initial reduction of $\widetilde{M} - \lambda\widetilde{N}$ to butterfly form $A - \lambda B$ requires about $75n^3$ arithmetic operations. For computing Z_0, additional $28n^3$ arithmetic operations are needed. The butterfly *SZ* algorithm requires about $O(n^2)$ arithmetic operations for the computation of $\breve{A} - \lambda\breve{B}$ and additional $85n^3$ arithmetic operations for the computation of Z (this estimate is based on the assumption that 2/3 iterations per eigenvalue are necessary as observed in [25]). The solution of the final linear system requires $(14/3)n^3$ arithmetic operations. Hence, the entire algorithm described above requires about $(586/3)n^3$ arithmetic operations.

However, it should be noted that in the *SZ* algorithm nonorthogonal equivalence transformations have to be used. These are not as numerically stable as the orthogonal transformations used by the *QZ* algorithm. Therefore, the approximate DARE solution computed by the *SZ* algorithm is sometimes less accurate than the one obtained from using the *QZ* algorithm. A possibility to improve the computed solution is defect correction as discussed in the next section.

4. DEFECT CORRECTION

Any approximate solution $\widetilde{X}$ computed, for example, with one of the methods described above, can be improved via defect correction. Let

$$\widetilde{X} = X + E, \tag{47}$$

where X is the exact solution of (1), $X = Q + LX^{-1}L^T$. Then

$$\begin{aligned} \widetilde{X} &= E + Q + LX^{-1}L^T \\ &= E + Q + L(\widetilde{X} - E)^{-1}L^T \\ &= E + Q + L\big((I - E\widetilde{X}^{-1})\widetilde{X}\big)^{-1}L^T \\ &= E + Q + L\widetilde{X}^{-1}\big(I - E\widetilde{X}^{-1}\big)^{-1}L^T. \end{aligned} \tag{48}$$

Assume that $\|E\| < 1/\|\widetilde{X}^{-1}\|$. Then we have $\|E\widetilde{X}^{-1}\| < 1$. Using the Neumann series [19, Lemma 2.3.3] yields

$$\begin{aligned} \widetilde{X} &= E + Q + L\widetilde{X}^{-1}\big(I + E\widetilde{X}^{-1} + \big(E\widetilde{X}^{-1}\big)^2 + \cdots\big)L^T \\ &= E + Q + L\widetilde{X}^{-1}L^T + L\widetilde{X}^{-1}E\widetilde{X}^{-1}L^T \\ &\quad + L\widetilde{X}^{-1}\big(E\widetilde{X}^{-1}\big)^2L^T + \cdots \\ &= E + Q + L\widetilde{X}^{-1}L^T + L\widetilde{X}^{-1}E\widetilde{X}^{-1}L^T \\ &\quad + L\widetilde{X}^{-1}E\widetilde{X}^{-1}E\widetilde{X}^{-1}L^T + \cdots \\ &= E + Q + L\widetilde{X}^{-1}L^T + \widetilde{L}E\widetilde{L}^T + \widetilde{L}E\widetilde{X}^{-1}E\widetilde{L}^T + \cdots, \end{aligned} \tag{49}$$

where

$$\widetilde{L} = L\widetilde{X}^{-1}. \tag{50}$$

With the residual

$$R(\widetilde{X}) = \widetilde{X} - Q - L\widetilde{X}^{-1}L^T, \tag{51}$$

we thus have $R(\widetilde{X}) \approx E + \widetilde{L}E\widetilde{L}^T$. By dropping terms of order $O(\|E\|^2)$, we obtain the defect correction equation

$$R(\widetilde{X}) = \widetilde{E} + \widetilde{L}\widetilde{E}\widetilde{L}^T. \tag{52}$$

Hence, the approximate solution $\widetilde{X}$ can be improved by solving (52) for $\widetilde{E}$. The improved $\hat{X}$ is then given by $\hat{X} = \widetilde{X} - \widetilde{E}$.

Lemma 1. *Equation* (52) *has a unique solution if* $\rho(\widetilde{L}) = \rho(L\widetilde{X}^{-1}) < 1$.

Proof. Note that (52) is equivalent to the linear system of equations

$$\big(I_{n^2} + \widetilde{L}^T \otimes \widetilde{L}^T\big)\,\mathrm{vec}(\widetilde{E}) = \mathrm{vec}\,(R(\widetilde{X})), \tag{53}$$

where $\otimes$ denotes the Kronecker product and $\mathrm{vec}(A) = [a_{11}, \ldots, a_{n1}, a_{12}, \ldots, a_{n2}, \ldots, a_{1n}, \ldots, a_{nn}]^T$ is the vector that consists of the columns of $A = [a_{ij}]_{i,j=1}^{n}$ stacked on top of each other from left to right [31, Section 4.2]. As $\rho(\widetilde{L}) < 1$, the assertion follows from $\sigma(I_{n^2} + \widetilde{L}^T \otimes \widetilde{L}^T) = \{1 + \lambda_i\lambda_j \mid \lambda_i, \lambda_j \in \sigma(\widetilde{L})\}$. □

Note that Lemma 1 also follows from a more general existence result for linear matrix equations given in [7, Proposition 3.1].

In [3], essentially the same defect correction was derived by applying Newton's method to (1). Written in the notation used here, the defect correction equation derived in [3] reads

$$\widetilde{X} - Q + \widetilde{L}\widetilde{X}\widetilde{L}^T = E + \widetilde{L}E\widetilde{L}^T + 2\widetilde{L}L^T. \tag{54}$$

It is easy to see that this is equivalent to (52). In [3], it is suggested to solve the defect correction equation with a general Sylvester equation solver as in [32]. In that case, the computational work for solving the defect correction equation would be roughly 18 times that for the basic fixed point iteration. But a more efficient algorithm which makes use of the special structure of (52) can be easily devised: first, note that (52) looks very similar to a Stein (discrete Lyapunov) equation. The only difference is the sign in front of $\widetilde{E}$. With this observation and a careful inspection of the Bartels-Stewart-type algorithm for Stein equations suggested in [33] and implemented in the SLICOT basic control toolbox[2] function `slstei` (see also [34]), (52) can be solved efficiently with this algorithm when only a few signs are changed. This method requires about 14 times the cost for one fixed-point iteration as the Bartels-Stewart-type algorithm requires $32n^3$ flops [18].

5. NUMERICAL EXPERIMENTS

Numerical experiments were performed in order to compare the four different approaches for solving (1) discussed here. All algorithms were implemented in Matlab Version 7.2.0.232 (R2006a) and run on an Intel Pentium M processor.

In particular, we implemented the following:

(i) the fixed-point iteration as described in Section 2.1 which requires $(7/3)n^3$ arithmetic operations per iteration;

(ii) the doubling algorithm SDA as described in Section 2.2 which requires $(19/3)n^3$ arithmetic operations per iteration and uses the inverse of Q;

(iii) the doubling algorithm SDA-DARE as described in Section 2.2.1 which requires $(44/3)n^3$ arithmetic operations per iteration and uses the inverse of L;

(iv) the *SZ* algorithm as described in Section 3 which requires $(586/3)n^3$ arithmetic operations and uses the inverse of L.

[2] See http://www.slicot.org.

Slow convergence of the fixed-point iteration has been observed in, for example, [2, 3]. The convergence rate depends on the spectral radius $\rho(X_+L^{-T})$. One iteration of the doubling algorithm SDA costs as many as 2.7 iterations of the fixed-point iteration. In [5], no numerical examples are presented, in [6] only one example is given (see Example 3) in which the doubling algorithm is much faster than the fixed-point iteration. Our numerical experiments confirm that this is so in general. The *SZ* algorithm costs as many as 84 iterations of the fixed-point iteration, as many as 31 iterations of the doubling algorithm SDA, and as many as 13 iterations of the doubling algorithm SDA-DARE.

Example 2. First, the fixed-point equation approach as described in Section 2.1 was compared to the *SZ* approach as described in Section 3. For this, each example was first solved via the *SZ* approach. The so-computed solution X_* was used to determine the tolerance *tol*

$$tol = \frac{\left|\left|X_* - Q - LX_*^{-1}L^T\right|\right|_F}{\left|\left|X_*\right|\right|_F} \tag{55}$$

to which the fixed point iteration is run. That is, the fixed-point iteration was stopped as soon as

$$\frac{\left|\left|X_{i+1} - X_i\right|\right|_F}{\left|\left|X_{i+1}\right|\right|_F} = \frac{\left|\left|X_i - Q - LX_i^{-1}L^T\right|\right|_F}{\left|\left|X_{i+1}\right|\right|_F} < tol. \tag{56}$$

For the first set of examples Q and L were constructed as follows (using Matlab notation):

```
[Q,R]= qr(rand(n));
Q = Q'*diag(rand(n,1))*Q;
L = rand(n);
```

100 examples of size $n = 5, 6, 7, \ldots, 20$ and $n = 30, 40, 50, \ldots, 100$ were generated and solved as described above. The fixed-point iteration was never run for more than 300 steps. Table 1 reports how many examples of each size needed more than 84 iteration steps as well as how many examples of each size needed more than 300 iteration steps; here *it* denotes the number of iteration steps. Moreover, an average number *av* of iterations is determined, where only those examples of each size were counted which needed less than 300 iteration steps to converge. It can be clearly seen, that the larger n is chosen, the more iteration steps are required for the fixed-point iteration. Starting with $n = 40$ almost all examples needed more than 84 iteration steps. Hence the *SZ* approach is cheaper than the fixed-point approach. But even for smaller n, most examples needed more than 84 iterations, the average number of iterations needed clearly exceeds 84 for all $n \geq 5$. Hence, overall, it is cheaper to use the *SZ* approach.

The accuracy of the residual (55) achieved by the *SZ* approach was in general of the order of 10^{-12} for smaller n and 10^{-8} for larger n. But, as nonorthogonal transformations have to be used, occasionally, the accuracy can deteriorate to 10^{-3}. In that case, defect correction as described in Section 4 or the fixed-point iteration with starting matrix $X_0 = X_*$ can be used to increase the accuracy of the computed solution.

Table 1: First set of test examples.

n	Fixed-point iteration			SDA	SDA-DARE
	av	*it* > 84	*it* > 300	*av*	*av*
5	86.02	41	1	6.01	5.82
6	89.52	43	2	6.06	5.91
7	92.28	47	1	6.08	5.97
8	84.25	44	0	5.97	5.73
9	100.15	53	0	6.17	6.03
10	101.51	57	2	6.34	6.14
11	110.31	56	0	6.30	6.02
12	108.76	64	1	6.35	6.20
13	100.59	61	0	6.38	6.20
14	111.42	64	1	6.35	6.10
15	117.01	71	3	6.58	6.21
16	117.40	65	1	6.56	6.25
17	111.33	70	1	6.59	6.29
18	122.62	68	0	6.53	6.15
19	102.92	82	0	6.65	6.36
20	118.40	74	0	6.69	6.35
30	125.37	76	2	6.74	6.36
40	154.33	90	2	7.10	6.64
50	158.60	90	0	7.21	6.69
60	165.62	92	1	7.40	6.84
70	159.71	97	1	7.45	6.91
80	167.62	98	3	7.46	6.81
90	175.44	98	4	7.60	6.83
100	186.52	99	5	7.67	6.84

Next the doubling algorithm SDA was used to solve the same set of examples. Its iteration solves (23), the desired solution $X_\diamond$ is obtained from the computed solution via (26). The iteration was run until the residuum was less than $n \cdot \|Q\|_F \cdot eps$, where *eps* is Matlab's machine epsilon. This does not imply the same accuracy for the solution $X_\diamond$ of (1). Due to the back substitution (26), the final solution $X_\diamond$ may have a larger residual error. For these examples, only about 7 iterations were needed to determine an $X_\diamond$ which has about the same (or better) accuracy as the solution X_* computed via the *SZ* algorithm. Therefore, for these examples, the doubling algorithm is certainly more efficient than the fixed-point iteration or the *SZ* algorithm.

Finally, the SDA-DARE algorithm was used to solve the same examples. As the iterates X_i converge not only to the solution of the DARE (5), but also to the solution of (1), the iteration is run until

$$\frac{\left|\left|X_i - Q - LX_i^{-1}L^T\right|\right|_F}{\left|\left|X_i\right|\right|_F} < tol. \tag{57}$$

The average number of iterations needed for convergence is similar to that of the SDA algorithm, but each iteration here is more expensive than for the SDA algorithm. Hence, overall, for these examples, the SDA algorithm is the most efficient algorithm. For each n, for about two or three examples out of the 100 examples generated, the SDA-DARE did not quite achieve the same accuracy as the *SZ* algorithm: after about 5 iteration steps, the achieved accuracy just stagnated, usually only slightly larger than the accuracy achieved by the *SZ* algorithm. The matrices Q generated for these tests had a fairly small condition number

$$1 < \kappa_2(Q) < 10^5, \tag{58}$$

and a small norm

$$0.3 < \|Q\|_2 < 1. \tag{59}$$

In order to generate a different set of test matrices, Q and L were constructed as follows (using Matlab notation as before):

```
Q = triu(rand(n));
Q = Q'*Q;
L = rand(n);
```

100 examples of size $n = 5, 6, 7, \ldots, 20$ and $n = 30, 40, 50, \ldots, 60$ were generated and solved as described above. The matrices Q generated for these tests had a small norm

$$1.6 < \|Q\|_2 < 405, \tag{60}$$

but a fairly large condition number, we allowed for

$$1 < \kappa_2(Q) < 10^{13}. \tag{61}$$

As can be seen from Table 2, the fixed-point iteration performed much better for these examples, but the number of iterations necessary for convergence seems to be unpredictable. The doubling iteration SDA performs better than before, less iterations were needed for convergence. But while the iteration is run until the residual is less than $n \cdot \|Q\|_F \cdot eps$, it is clearly seen here that this does not imply the same accuracy for the solution $X_\diamond$ of (1). The larger n is chosen, the worse the residual

$$R_{\text{SDA}} = \frac{\left|\left|X_\diamond - Q - LX_\diamond^{-1}L^T\right|\right|_F}{\left|\left|X_\diamond\right|\right|_F} \tag{62}$$

becomes compared to the residual *tol* obtained by the *SZ* algorithm. Hence, the *SZ* algorithm may require more arithmetic operations, but usually it generates more accurate solutions. For most examples, the SDA-DARE algorithm converges in about 5 iterations to the same accuracy as the *SZ* algorithm, hence it is much more efficient. But as before, for few examples, the SDA-DARE algorithm did not achieve the same accuracy as the *SZ* algorithm as it stagnated at an accuracy $10 \cdot tol$. Rarely, the algorithm stagnated after about 5 iterations at a much larger error.

In case examples with ill-conditioned L are solved, the SDA-DARE and the *SZ* algorithm obviously will be a bad choice, while the fixed-point iteration and the SDA algorithm do not have any (additional) problems with ill-conditioned L.

Example 3. In [3], the following example is considered:

$$L = \begin{bmatrix} 50 & 10 \\ 20 & 60 \end{bmatrix}, \qquad Q = \begin{bmatrix} 3 & 2 \\ 2 & 4 \end{bmatrix}. \tag{63}$$

Table 2: Second set of test examples.

n	Fixed-point iteration			SDA		SDA-DARE
	av	*it* > 84	*it* > 300	*av*	$\sharp(R_{\mathrm{SDA}}) > tol$	*av*
5	56.01	16	2	5.15	23	5.17
6	69.31	27	2	5.39	31	5.42
7	55.13	12	0	5.05	28	5.15
8	60.88	14	0	5.06	47	5.24
9	55.85	14	0	4.97	48	5.19
10	54.83	8	0	4.87	56	5.26
11	51.93	12	0	4.70	56	5.01
12	48.97	5	0	4.60	66	5.13
13	48.40	6	0	4.55	70	5.09
14	51.55	10	0	4.60	68	5.17
15	45.62	3	0	4.41	72	4.98
16	46.64	2	0	4.42	75	5.04
17	46.89	4	0	4.23	84	5.04
18	45.56	4	0	4.15	84	5.00
19	42.77	2	0	4.03	81	4.94
20	45.27	2	0	3.97	88	4.98
30	35.80	0	0	3.49	96	4.79
40	34.07	0	0	3.23	96	4.78
50	32.34	0	0	2.93	98	4.61
60	31.32	0	0	2.82	100	4.44

The solution X_+ is given by

$$X_+ \approx \begin{bmatrix} 51.7993723118 & 16.0998802679 \\ 16.0998802679 & 62.2516164469 \end{bmatrix}. \tag{64}$$

Slow convergence for the fixed point iteration was already observed in [3], after 400 iteration steps one obtains the residual norm

$$\frac{||X_{400} - Q - LX_{400}^{-1}L^T||_F}{||X_{400}||_F} = 3.78 \cdot 10^{-10}, \tag{65}$$

and the error

$$||X_+ - X_{400}||_F = 1.64 \cdot 10^{-8}, \tag{66}$$

since $\rho(X_+^{-1}L^T) = 0.9719$. The doubling iteration SDA yields after 8 iterations

$$\begin{aligned} \frac{||X_\diamond - Q - LX_\diamond^{-1}L^T||_F}{||X_\diamond||_F} &= 6.35 \cdot 10^{-13}, \\ ||X_+ - X_\diamond||_F &= 7.77 \cdot 10^{-11}, \end{aligned} \tag{67}$$

while the SDA-DARE algorithm yields after 9 iterations

$$\begin{aligned} \frac{||X_\diamond - Q - LX_\diamond^{-1}L^T||_F}{||X_\diamond||_F} &= 6.68 \cdot 10^{-13}, \\ ||X_+ - X_\diamond||_F &= 6.92 \cdot 10^{-11}. \end{aligned} \tag{68}$$

The *SZ* algorithm obtains

$$\begin{aligned} \frac{||X_* - Q - LX_*^{-1}L^T||_F}{||X_*||_F} &= 1.79 \cdot 10^{-13}, \\ ||X_+ - X_*||_F &= 6.98 \cdot 10^{-11}. \end{aligned} \tag{69}$$

Hence, the doubling iterations outperform the *SZ* algorithm here.

6. CONCLUSIONS

We have discussed several algorithms for a rational matrix equation that arises in the analysis of stationary Gaussian reciprocal processes. In particular, we have described the application of the *SZ* algorithm for symplectic pencils to solve this equation. Moreover, we have derived a defect correction equation that can be used to improve the accuracy of a computed solution. Several examples comparing the iterative methods with the *SZ* approach show that none of the methods discussed is superior. Usually, both doubling-type algorithms SDA and SDA-DARE compute the approximate solution very fast, but due to the back transformation step, the accuracy of the SDA algorithm can deteriorate significantly. On the other hand, the fixed-point iteration is often very slow. The *SZ* approach needs a predictable computing time which is most often less than that of the fixed-point iteration when a comparable accuracy is requested, but is usually much higher than for the doubling algorithms. The accuracy of the *SZ* approach is not always the best compared to the other methods, but in a number of examples, the doubling algorithms are unable to attain the same accuracy while the fixed-point iteration is significantly slower.

REFERENCES

[1] B. C. Levy, R. Frezza, and A. J. Krener, "Modeling and estimation of discrete-time Gaussian reciprocal processes," *IEEE Transactions on Automatic Control*, vol. 35, no. 9, pp. 1013–1023, 1990.

[2] A. Ferrante and B. C. Levy, "Hermitian solutions of the equation $X = Q + NX^{-1}N^*$," *Linear Algebra and Its Applications*, vol. 247, pp. 359–373, 1996.

[3] C.-H. Guo and P. Lancaster, "Iterative solution of two matrix equations," *Mathematics of Computation*, vol. 68, no. 228, pp. 1589–1603, 1999.

[4] I. G. Ivanov, V. I. Hasanov, and F. Uhlig, "Improved methods and starting values to solve the matrix equations $X \pm A^*X^{-1}A = I$ iteratively," *Mathematics of Computation*, vol. 74, no. 249, pp. 263–278, 2005.

[5] W.-W. Lin and S.-F. Xu, "Convergence analysis of structure-preserving doubling algorithms for Riccati-type matrix equations," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 1, pp. 26–39, 2006.

[6] B. Meini, "Efficient computation of the extreme solutions of $X + A^*X^{-1}A = Q$ and $X - A^*X^{-1}A = Q$," *Mathematics of Computation*, vol. 71, no. 239, pp. 1189–1204, 2002.

[7] M. Reurings, *Symmetric matrix equations*, Ph.D. thesis, Vrije Universiteit, Amsterdam, The Netherlands, 2003.

[8] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, Academic Press, Orlando, Fla, USA, 2nd edition, 1985.

[9] P. Lancaster and L. Rodman, *Algebraic Riccati Equations*, Oxford University Press, Oxford, UK, 1995.

[10] C. D. Ahlbrandt and A. C. Peterson, *Discrete Hamiltonian Systems: Difference Equations, Continued Fractions, and Riccati Equations*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[11] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1979.
[12] B. D. O. Anderson and B. Vongpanitlerd, *Network Analysis and Synthesis. A Modern Systems Approach*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1972.
[13] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice-Hall, Upper Saddle River, NJ, USA, 1995.
[14] G. A. Hewer, "An iterative technique for the computation of the steady state gains for the discrete optimal regulator," *IEEE Transactions on Automatic Control*, vol. 16, no. 4, pp. 382–384, 1971.
[15] V. L. Mehrmann, *The Autonomous Linear Quadratic Control Problem: Theory and Numerical Solution*, vol. 163 of *Lecture Notes in Control and Information Sciences*, Springer, Heidelberg, Germany, 1991.
[16] A. J. Laub, "Algebraic aspects of generalized eigenvalue problems for solving Riccati equations," in *Computational and Combinatorial Methods in Systems Theory*, C. I. Byrnes and A. Lindquist, Eds., pp. 213–227, Elsevier/North-Holland, New York, NY, USA, 1986.
[17] T. Pappas, A. J. Laub, and N. R. Sandell Jr., "On the numerical solution of the discrete-time algebraic Riccati equation," *IEEE Transactions on Automatic Control*, vol. 25, no. 4, pp. 631–641, 1980.
[18] V. Sima, *Algorithms for Linear-Quadratic Optimization*, vol. 200 of *Pure and Applied Mathematics*, Marcel Dekker, New York, NY, USA, 1996.
[19] G. H. Golub and C. F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Md, USA, 3rd edition, 1996.
[20] E. K.-W. Chu, H.-Y. Fan, W.-W. Lin, and C.-S. Wang, "Structure-preserving algorithms for periodic discrete-time algebraic Riccati equations," *International Journal of Control*, vol. 77, no. 8, pp. 767–788, 2004.
[21] P. Benner, *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*, Logos, Berlin, Germany, 1997.
[22] Z. Bai, J. Demmel, and M. Gu, "An inverse free parallel spectral divide and conquer algorithm for nonsymmetric eigenproblems," *Numerische Mathematik*, vol. 76, no. 3, pp. 279–308, 1997.
[23] A. N. Malyshev, "Parallel algorithm for solving some spectral problems of linear algebra," *Linear Algebra and Its Applications*, vol. 188-189, no. 1, pp. 489–520, 1993.
[24] P. Benner and H. Faßbender, "The symplectic eigenvalue problem, the butterfly form, the *SR* algorithm, and the Lanczos method," *Linear Algebra and Its Applications*, vol. 275-276, pp. 19–47, 1998.
[25] P. Benner, H. Faßbender, and D. S. Watkins, "*SR* and *SZ* algorithms for the symplectic (butterfly) eigenproblem," *Linear Algebra and Its Applications*, vol. 287, no. 1–3, pp. 41–76, 1999.
[26] H. Faßbender, *Symplectic Methods for the Symplectic Eigenproblem*, Kluwer Academic/Plenum Publishers, New York, NY, USA, 2000.
[27] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, SIAM, Philadelphia, Pa, USA, 2005.
[28] B. D. O. Anderson, "Second-order convergent algorithms for the steady-state Riccati equation," *International Journal of Control*, vol. 28, no. 2, pp. 295–306, 1978.
[29] S. Barrachina, P. Benner, and E. S. Quintana-Ortí, "Solution of discrete-time Riccati equations via structure-preserving doubling algorithms on a cluster of SMPs," preprint, 2006, Fakultät für Mathematik, TU Chemnitz, Germany, http://www.tu-chemnitz.de/~benner/pub/bbq-sda.pdf.
[30] D. S. Watkins and L. Elsner, "Theory of decomposition and bulge-chasing algorithms for the generalized eigenvalue problem," *SIAM Journal on Matrix Analysis and Applications*, vol. 15, no. 3, pp. 943–967, 1994.
[31] R. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1994.
[32] J. D. Gardiner, A. J. Laub, J. J. Amato, and C. B. Moler, "Solution of the Sylvester matrix equation $A \times B^T + C \times D^T = E$," *ACM Transactions on Mathematical Software*, vol. 18, no. 2, pp. 223–231, 1992.
[33] A. Y. Barraud, "A numerical algorithm to solve $A^T X A - X = Q$," *IEEE Transactions on Automatic Control*, vol. 22, no. 5, pp. 883–885, 1977.
[34] M. Slowik, P. Benner, and V. Sima, "Evaluation of the Linear Matrix Equation Solvers in SLICOT," *SLICOT Working Note*, 2004—1, 2004, http://www.icm.tu-bs.de/NICONET/.

Peter Benner received the Diploma in mathematics from the RWTH Aachen, Germany, in 1993. From 1993 to 1997, he worked on his dissertation at the University of Kansas, Lawrence, USA, and the TU Chemnitz-Zwickau, Germany, where he received his Ph.D. degree in February 1997. In 2001, he finished his Habilitation at the University of Bremen where he was Assistant Professor from 1997 to 2001. He held positions as Visiting Associate Professor at the TU Hamburg-Harburg, Germany, 2001–2002, and as Lecturer at TU Berlin, 2002–2003. Since October 2003, he is Full Professor for Mathematics in Industry and Technology at Chemnitz University of Technology. His research interests are in the areas of scientific computing, numerical mathematics, systems and control theory, and mathematical software. His research focuses on linear and nonlinear matrix equations, model and system reduction, numerical methods for optimal control of systems modeled by evolution equations, stabilization of dynamical systems, and Krylov subspace methods for structured or quadratic eigenproblems.

Heike Faßbender received her Diploma in mathematics from the University of Bielefeld, Germany, in 1989, and the Master of Science degree in computer science from the State University of New York at Buffalo, Buffalo, NY, in 1991. In 1993, she received her Ph.D. degree in mathematics from the University of Bremen, Germany, where she worked as an Assistant Professor from 1993 to 2000. After receiving her Habilitation in mathematics from the University of Bremen, Germany, she was a Professor at the Munich University of Technology, Germany, from 2000 to 2002. Since 2002, she is a Full Professor at the TU Braunschweig (University of Technology), Germany, where she is Director of the Institute Computational Mathematics and Head of the numerical analysis research group. Currently, she is the Dean of the Carl-Friedrich-Gauss-Fakultät of the TU Braunschweig, Germany. Her research interests are in numerical linear algebra with applications to systems and control and signal processing as well as matrix theory. Especially, structure-preserving algorithms for (large-scale) eigenproblems or linear systems are in the focus of her attention.