Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik

A Hybrid Method for the Numerical Solution of Discrete-Time Algebraic Riccati Equations

Peter Benner **Heike Faßbender**

Report 99–12

Berichte aus der Technomathematik

Report 99–12 **November 1999**

A Hybrid Method for the Numerical Solution of Discrete-Time Algebraic Riccati Equations

Heike Faßbender and Peter Benner

Abstract. A discrete-time algebraic Riccati equation (DARE) is a set of nonlinear equations. One of the oldest, best studied, numerical methods for solving it, is Newton's method. Finding a stabilizing starting guess which is already close to the desired solution is crucial. We propose to compute an approximate solution of the DARE by the (butterfly) SZ algorithm applied to the corresponding symplectic pencil where zero and infinity eigenvalues are removed using an iterative deflation strategy. This algorithm is a fast, reliable and structure-preserving algorithm for computing the stable deflating subspace of the symplectic matrix pencil associated with the DARE. From this, a stabilizing starting guess for Newton's method is easily obtained. The resulting method is very efficient and produces highly accurate results. Numerical examples demonstrate the behavior of the resulting hybrid method.

Keywords. discrete-time algebraic Riccati equation, Newton's method, SZ algorithm, symplectic matrix pencil.

1. Introduction

The standard (discrete-time) linear-quadratic optimization problem consists in finding a control trajectory $\{u(k), k = 0, 1, 2, \dots\}$, minimizing the cost functional

$$\mathcal{J}(x_0, u) = \sum_{k=0}^{\infty} [x(k)^T Q x(k) + u(k)^T R u(k)]$$

in terms of u subject to the dynamical constraint

$$x(k+1) = Ax(k) + Bu(k), \qquad x(0) := x_0,$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times n}$, $Q \in \mathbb{R}^{p\times p}$, and $R \in \mathbb{R}^{m\times m}$. Furthermore, we assume Q and R to be symmetric. Under certain conditions there is a unique control law,

$$u(k) = \mathcal{K}(X_*)x(k), \qquad \mathcal{K}(X_*) := -(R + B^T X_* B)^{-1} B^T X_* A,$$

minimizing $\mathcal{J}$ in terms of u subject to the dynamical constraint. The matrix X_* is the unique symmetric stabilizing solution of the algebraic matrix Riccati equation

$$0 = \mathcal{DR}(X) = Q - X + A^T X A - A^T X B (R + B^T X B)^{-1} B^T X A. \tag{1.1}$$

That is, $X_* = X_*^T$ is the solution of (1.1) and all eigenvalues of $A - B\mathcal{K}(X_*)$ are inside the unit circle: $\sigma(A - B\mathcal{K}(X_*)) \subset \mathcal{D}_1(0)$, where $\mathcal{D}_1(0) = \{\zeta \in \mathbb{C}, |\zeta| < 1\}$. The equation (1.1) is usually referred to as discrete-time algebraic Riccati equation

1991 *Mathematics Subject Classification.* Primary 65H10, 49N05; Secondary 15A24, 65F15.

(DARE). It appears 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, e.g., [**1, 2, 3, 28, 22, 31**].

The DARE (1.1) can be considered as a nonlinear set of equations. Therefore, Newton's method has been one of the first methods proposed to solve DAREs [**19**]. Finding a stabilizing starting guess which is already close to the desired solution is crucial. It is well known that (under certain reasonable assumptions) if X_0 is a stabilizing starting guess, then all iterates are stabilizing and converge globally quadratic to the desired solution X_* (see, e.g., [**19, 22, 25**]). Despite the ultimate rapid convergence, the iteration may initially converge slowly. This can be due to a large initial error $||X_* - X_0||$ or a disastrously large first Newton step resulting in a large error $||X_* - X_1||$. In both cases, it is possible that many iterations are required to find the region of rapid convergence.

Here we propose to compute an approximate solution $\widehat{X}$ of $\mathcal{DR}(X)$ by the (butterfly) SZ algorithm. This solution is then used as a starting guess for Newton's method. The resulting hybrid method for solving (1.1) is a very efficient method and produces highly accurate results.

Assume R to be positive definite and define

$$L - \lambda M = \begin{bmatrix} A & 0 \\ Q & I \end{bmatrix} - \lambda \begin{bmatrix} I & -BR^{-1}B^T \\ 0 & A^T \end{bmatrix}. \tag{1.2}$$

Using furthermore the standard control-theoretic assumptions that

- (A, B) is stabilizable,
- (Q, A) is detectable,
- Q is positive semidefinite,

then $L - \lambda M$ has no eigenvalues on the unit circle and there exists a unique stabilizing solution X_* of the DARE (1.1); see, e.g., [**22**]. It is then easily seen that $L - \lambda M$ 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 $L - \lambda M$ corresponding to the n eigenvalues $\lambda_1, \ldots, \lambda_n$ inside the unit circle using the relation

$$\begin{bmatrix} A & 0 \\ Q & I \end{bmatrix} \begin{bmatrix} I \\ -X \end{bmatrix} = \begin{bmatrix} I & -BR^{-1}B^T \\ 0 & A^T \end{bmatrix} \begin{bmatrix} I \\ -X \end{bmatrix} \Lambda,$$

where $\Lambda \in \mathbb{R}^{n \times n}$, $\sigma(\Lambda) = \{\lambda_1, \ldots, \lambda_n\}$. 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 $L - \lambda M$, then $X_* = -Y_2 Y_1^{-1}$ is the desired solution of the Riccati equation (1.1). See, e.g., [**22, 23, 25**], and the references therein.

It is worthwhile to note that $L - \lambda M$ of the form (1.2) is a symplectic matrix pencil. A symplectic matrix pencil $L - \lambda M, L, M \in \mathbb{R}^{2n \times 2n}$, is defined by the property

$$LJL^T = MJM^T,$$

where

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

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 $L - \lambda M$ with left eigenvector

x, then $\overline{\lambda^{-1}}$ is an eigenvalue of $L-\lambda M$ 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.

The numerical computation of a deflating subspace of a (symplectic) matrix pencil $L-\lambda M$ is usually carried out by an iterative procedure like the QZ algorithm. The QZ algorithm is numerically backward stable but it ignores the symplectic structure. Applying the QZ algorithm to a symplectic matrix pencil results in a general $2n \times 2n$ matrix pencil in generalized Schur form from which the eigenvalues and deflating subspaces can be read off. Sorting the eigenvalues in the generalized Schur form such that the eigenvalues inside the unit circle are contained in the upper left $n \times n$ block, this method results in the popular generalized Schur vector method for solving DAREs [**26**]. Due to roundoff errors unavoidable in finite-precision arithmetic, the computed eigenvalues will in general not 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 $L-\lambda M$ 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, e.g., [**29**]. An estimated $40n^3$ flops are necessary for ordering the generalized Schur form. This results in a total cost of roughly $415n^3$ flops for computing a starting guess for Newton's method using the QZ algorithm, employing standard assumptions about convergence of the QZ iteration (see, e.g., [**17**]).

Here we propose to use the butterfly SZ algorithm for computing the deflating subspace of $L-\lambda M$. The butterfly SZ algorithm [**11, 16**] is a fast, reliable and efficient algorithm especially designed for solving the symplectic eigenproblem. It makes use of the fact that symplectic matrix pencils can be reduced to matrix pencils of the form

$$K-\lambda N = \begin{bmatrix} C & F \\ 0 & C^{-1} \end{bmatrix} - \lambda \begin{bmatrix} 0 & I_n \\ I_n & T \end{bmatrix}, \tag{1.3}$$

where C, F are diagonal and T is symmetric tridiagonal. This form is determined by just $4n-1$ parameters. The symplectic matrix pencil $K-\lambda N$ is called a symplectic butterfly pencil. 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. Using the so obtained solution as a starting guess for Newton's method, the resulting method for solving discrete-time algebraic Riccati equations is a very efficient method and produces highly accurate results.

In Section 2 Newton's method is reviewed. Section 3 briefly describes the (butterfly) SZ algorithm for computing the deflating subspace of $L-\lambda M$. Combined with a strategy to deflate zero and infinity eigenvalues from the symplectic pencil in order to deal with discrete-time algebraic Riccati equations with singular A matrix, the hybrid method described in Section 4 consisting of the SZ algorithm followed

by a few Newton iteration steps results in an efficient and accurate method for solving discrete-time algebraic Riccati equations. Numerical experiments are reported in Section 5.

2. Newton's method

The function $\mathcal{DR}(X)$ is a rational matrix function and $\mathcal{DR}(X) = 0$ defines a system of nonlinear equations. Hence it is straightforward to apply Newton's method to DAREs. Inspired by Kleinman's formulation of Newton's method for continuous-time algebraic Riccati equations [**20**], Hewer [**19**] proposed an analogous scheme for solving DAREs. A discussion of its convergence properties can be found in [**25, 22**].

Given a symmetric matrix X_0, the method can be given in algorithmic form as follows:

ALGORITHM 2.1.
FOR $k = 0, 1, 2, \ldots$
1. $K_k \leftarrow \mathcal{K}(X_k) = (R + B^T X_k B)^{-1} B^T X_k A$.
2. $A_k \leftarrow A - BK_k$.
3. $\mathcal{R}_k \leftarrow \mathcal{DR}(X_k)$.
4. Solve for N_k in the Stein equation

$$A_k^T N_k A_k - N_k = -\mathcal{R}_k. \tag{2.1}$$

5. $X_{k+1} \leftarrow X_k + N_k$.

END FOR
END

We have the following result for Algorithm 2.1 [**19, 25, 22**].

THEOREM 2.2. *If*

i) (A, B) *is stabilizable,*
ii) $R = R^T \geq 0$,
iii) *a unique stabilizing solution* X_* *of (1.1) exists such that* $R + B^T X_* B > 0$,
iv) X_0 *is stabilizing,*

then for the iterates produced by Algorithm 2.1 we have:

a) *All iterates* X_k *are stabilizing, i.e.,* $\sigma\left(A - B\mathcal{K}(X_k)\right) \subset \mathcal{D}_1(0)$ *for all* $k \in \mathbb{N}_0$.
b) $X_* \leq \ldots \leq X_{k+1} \leq X_k \leq \ldots \leq X_1$.
c) $\lim_{k\to\infty} X_k = X_*$.
d) *There exists a constant* $\gamma > 0$ *such that*

$$\|X_{k+1} - X_*\| \leq \gamma \|X_k - X_*\|^2, \qquad k \geq 1,$$

i.e., the X_k *converge globally quadratic to* X_*.

The formulation of Algorithm 2.1 is analogous to the standard formulation of Newton's method as given, e.g., in [**15**, Algorithm 5.1.1] for the solution of nonlinear equations. Because of its robustness in the presence of rounding errors, we prefer to calculate the Newton step explicitly as in Algorithm 2.1 rather than to use the mathematically equivalent formulation of the Newton step [**19, 25, 22**],

$$A_k^T X_{k+1} A_k - X_{k+1} = -Q - K_k^T R K_k \tag{2.2}$$

which determines X_{k+1} directly. The coefficient matrices of the two Stein equations are the same, but the right-hand-sides are different. Define the Stein operator as the linear map

$$\Gamma_k : Z \longrightarrow (A - B\mathcal{K}(X_k))^T Z + Z(A - B\mathcal{K}(X_k)), \tag{2.3}$$

which is also the Fréchet derivative of $\mathcal{DR}(X)$ at X_k. Let us assume that the condition number of Γ_k permits us to solve the Stein equations (2.1) and (2.2) to ℓ correct significant digits. Loosely speaking this implies that when X_{k+1} is calculated directly as in (2.2), then its accuracy is limited to ℓ significant digits. On the other hand, in Algorithm 2.1, the accuracy of the computed Newton step N_k is limited to ℓ significant digits. Therefore, the sum $X_k + N_k$ has up to ℓ *more* correct digits than X_k. The accuracy of Algorithm 2.1 is ultimately limited only by the accuracy to which $\mathcal{K}(X_k)$, A_k, $\mathcal{DR}(X_k)$, and the sum $X_k + N_k$ are calculated.

The computational cost for Algorithm 2.1 mainly depends upon the cost for the numerical solution of the Stein equation (2.1). This can be done using the Bartels–Stewart algorithm [**7, 6**]. Then the cost for the solution of the Stein equation is about $32n^3$ flops. The computations for forming A_k and $\mathcal{R}_k$ can be arranged such that for $n \approx m$, they require $\approx 17n^3$ flops while for $m \ll n$, these matrices can be formed using only $\approx 3n^3$ flops. For an average value of $m = n/2$, the computational cost for one step of Algorithm 2.1 is about $42n^3$ flops.

One major difficulty is to find a stabilizing initial guess X_0. There exist stabilization procedures for discrete-time linear systems (see, e.g., [**4, 21, 29**]). But these may give large initial errors $\|X_* - X_0\|$ (see, e.g., [**8**]). The procedure suggested in [**21**] is even infeasible for numerical computations as it is based on explicitly summing up $A^k BB^T (A^T)^k$ for k up to n, thereby often causing overflow already for small values of n. This problem can be overcome in case A is stable. In that case, one can start from $X_0 = 0$.

Despite the ultimate rapid convergence indicated by Theorem 2.2 d), the iteration may initially converge slowly. This can be due to a large initial error $\|X_* - X_0\|$ or a disastrously large first Newton step resulting in a large error $\|X_* - X_1\|$. In both cases, it is possible that many iterations are required to find the region of rapid convergence. An ill-conditioned Stein equation makes it difficult to compute an accurate Newton step. An inaccurately computed Newton step can cause the usual convergence theory to break down in practice. Sometimes rounding errors or a poor choice of X_0 cause Newton's method to converge to a non-stabilizing solution.

For these reasons, Newton's method is usually not used by itself to solve DAREs. However, when it is used as a defect correction method or for iterative refinement of an approximate solution obtained by a more robust method, it is often able to squeeze out the maximum possible accuracy [**25**] after only one or two iterations. Therefore we propose here to find a stabilizing initial guess using the butterfly SZ algorithm.

An approach to overcome slow initial convergence is suggested in [**8, 9**]. There, a line search strategy is suggested that usually accelerates the convergence during the first iteration steps. The strategy is particularly successful if A is known to be stable and one can start from $X_0 = 0$. Still, for unstable A matrix, this procedure relies on some initial stabilization and even for a stable A matrix, $X_0 = 0$ may result in a bad first step. Therefore this approach will also benefit from a good starting guess. Note, however, that most frequently a starting guess obtained from

the SZ algorithm is so close to the desired solution that line search will usually not improve Newton's method significantly. We will therefore not discuss this topic here any further. Still, in case the solution of the DARE via the deflating subspace approach is ill-conditioned, line search may improve the convergence behavior.

3. The butterfly SZ algorithm

For simplicity let us assume at the moment that A in (1.2) is nonsingular. Premultiplying $L - \lambda M$ by $\left[\begin{smallmatrix} I & 0 \\ 0 & A^{-T} \end{smallmatrix}\right]$ results in a symplectic matrix pencil

$$L' - \lambda M' = \begin{bmatrix} A & 0 \\ A^{-T}Q & A^{-T} \end{bmatrix} - \lambda \begin{bmatrix} I & -BR^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{3.1}$$

where L', M' are both symplectic, that is,

$$L'JL'^T = M'JM'^T = J,$$

as a matrix $X \in \mathbb{R}^{n\times n}$ is symplectic if $XJX^T = J$. In [**11, 16**] it is shown that for the symplectic matrix pencil $L' - \lambda M'$ there exist numerous symplectic matrices Z and nonsingular matrices S which reduce $L' - \lambda M'$ to a symplectic butterfly pencil $K - \lambda N$ (1.3)

$$S(L' - \lambda M')Z = K - \lambda N = \begin{bmatrix} C & F \\ 0 & C^{-1} \end{bmatrix} - \lambda \begin{bmatrix} 0 & -I \\ I & T \end{bmatrix},$$

where C and F are diagonal matrices, and T is a symmetric tridiagonal matrix. (More general, not only the symplectic matrix pencil in (3.1), but any symplectic matrix pencil $L' - \lambda M'$ with symplectic matrices L', M' can be reduced to 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 $K - \lambda N$ [**11, 16**]. The SZ algorithm preserves the symplectic butterfly form in its iterations. It is the analogue of the SR algorithm (see [**10, 16**]) 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 $K - \lambda N$. Choose a spectral transformation function q and compute a symplectic matrix Z_1 such that

$$Z_1^{-1} q(K^{-1}N)e_1 = \alpha e_1$$

for some scalar α. Then transform the pencil to

$$\widetilde{K} - \lambda\widetilde{N} = (K - \lambda N)Z_1.$$

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

$$\widehat{K} - \lambda\widehat{N} = S^{-1}(\widetilde{K} - \lambda\widetilde{N})\widetilde{Z},$$

where $\widehat{K} - \lambda\widehat{N}$ is 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. For a detailed discussion of the butterfly SZ algorithm see [**11, 16**].

Hence, in order to compute an approximate solution of the DARE (1.1) by the butterfly SZ algorithm, first the symplectic matrix pencil $L - \lambda M$ as in (1.2) has to be formed, then the symplectic matrix pencil $L' - \lambda M'$ as in (3.1) is computed. Next symplectic matrices Z_0 and S_0 are computed such that

$$\widehat{L} - \lambda\widehat{M} := S_0^{-1}L'Z_0 - \lambda S_0^{-1}M'Z_0$$

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

$$S_1^{-1}\widehat{L}Z_1 - \lambda S_1^{-1}\widehat{M}Z_1$$

is a symplectic butterfly pencil and the symmetric tridiagonal matrix $\widehat{T}$ in the lower right block of $S_1^{-1}\widehat{M}Z_1$ is reduced to quasi-diagonal 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

$$\breve{L} = S_2^{-1}S_1^{-1}\widehat{L}Z_1Z_2 = \begin{bmatrix} \phi_{11} & \phi_{12} \\ 0 & \phi_{22} \end{bmatrix},$$

$$\breve{M} = S_2^{-1}S_1^{-1}\widehat{M}Z_1Z_2 = \begin{bmatrix} \psi_{11} & \psi_{12} \\ 0 & \psi_{22} \end{bmatrix},$$

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,

$$Z = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}, \tag{3.2}$$

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

$$X_* = -Z_{21}Z_{11}^{-1}. \tag{3.3}$$

This algorithm requires about $193n^3$ arithmetic operations in order to compute the desired deflating subspace of $L - \lambda M$ and is therefore cheaper than the QZ algorithm which requires about $415n^3$ arithmetic operations (both flop counts do not take into account the cost for forming $L-\lambda M$ and the cost for solving the linear system (3.3) at the end, as these steps are the same for both algorithms). The cost of the different steps of the approach described above are given as follows. The computation of $A^{-T}Q$ and A^{-T} using an LR decomposition of A requires about $\frac{14}{3}n^3$ arithmetic operations. A careful flop count reveals that the initial reduction of $L' - \lambda M'$ to butterfly form $\widehat{L} - \lambda\widehat{M}$ requires about $75n^3$ arithmetic operations. For computing Z_0, an additional $28n^3$ arithmetic operations are needed. The butterfly SZ algorithm requires about $O(n^2)$ arithmetic operations for the computation of $\breve{L} - \lambda\breve{M}$ and additional $85n^3$ arithmetic operations for the computation of Z (this estimate is based on the assumption that $\frac{2}{3}$ iterations per eigenvalue are necessary as observed in [**11**]). Hence, the entire algorithm described, requires about $\frac{578}{3}n^3$ arithmetic operations.

Instead of generating the symplectic matrix Z as in (3.2), one can work with $n\times n$ matrices X, Y and T such that finally $X = -Z_{21}Z_{11}^{-1} = X_*$, $Y = Z_{11}^{-1}Z_{12}$, and $T = Z_{11}^{-1}$. Starting from $X = Y = 0, T = I$, this can be implemented without accumulating the intermediate symplectic transformations used in the butterfly SZ algorithm, just using the parameters that determine these transformations. As for every symplectic matrix Z written in the form (3.2), $Z_{21}Z_{11}^{-1}$ is symmetric, this

approach guarantees that all intermediate (and the final) X are symmetric. Such an approach, called symmetric updating, was first proposed by Byers and Mehrmann [**14**] in the context of solving continuous-time algebraic Riccati equations via the Hamiltonian SR algorithm and has also been proposed for solving DAREs with an SR algorithm in [**5**]. A flop count shows that this approach is not feasible here. For the initial reduction of $L' - \lambda M'$ to butterfly form about $5n^4$ arithmetic operations are needed in order to compute X, Y, and Z. The butterfly SZ algorithm requires about $11n^4$ arithmetic operations for the symmetric updating. Hence, the symmetric updating requires $\mathcal{O}(n^4)$ arithmetic operations. Moreover, this increase in computational cost is not rewarded with a significantly better computation of X.

So far, we have assumed that A is nonsingular such that the symplectic pencil $L'-\lambda M'$ (3.1) can be formed. In case A is singular, the symplectic matrix pencil $L-\lambda M$ (1.2) has zero and infinity eigenvalues. Mehrmann proposes in [**25**, Algorithm 15.16] the following algorithm to deflate zero and infinite eigenvalues of $L - \lambda M$: Assume that $\operatorname{rank}(A) = k$. First, use the QR decomposition with column pivoting [**17**] to determine an orthogonal matrix $V \in \mathbb{R}^{n\times n}$, an upper triangular matrix $U \in \mathbb{R}^{n\times n}$ and a permutation matrix $P \in \mathbb{R}^{n\times n}$ such that

$$PA = UQ = [0\ U_2] \begin{bmatrix} V_1 \\ V_2 \end{bmatrix},$$

where $U_2 \in \mathbb{R}^{n\times n-k}$ and $V_2 \in \mathbb{R}^{n-k\times n}$ have full rank. Then form

$$T = \begin{bmatrix} V^T & 0 \\ -QV^T & V^T \end{bmatrix}, \qquad \text{and} \qquad \widetilde{T} = \begin{bmatrix} V(I+GQ)^{-1} & 0 \\ U^TPQ(I+GQ)^{-1} & V \end{bmatrix},$$

where for ease of notation G denotes $BR^{-1}B^T$. Now as $AV^T = P^TU = P^T[0\ U_2]$, we obtain

$$\begin{aligned}
&\widetilde{T}(L-\lambda M)T = \\
&= \begin{bmatrix} V(I+GQ)^{-1}P^TU & 0 \\ U^TPQ(I+GQ)^{-1}P^TU & I \end{bmatrix} - \lambda \begin{bmatrix} I & -V(I+GQ)^{-1}GV^T \\ 0 & U^TP(-Q(I+GQ)^{-1}G+I)V^T \end{bmatrix} \\
&= \left[\begin{array}{cc|cc} 0 & \widetilde{A}_1 & 0 & 0 \\ 0 & \widetilde{A} & 0 & 0 \\ \hline 0 & 0 & I & 0 \\ 0 & \widetilde{Q} & 0 & I \end{array}\right] - \lambda \left[\begin{array}{cc|cc} I & 0 & \widetilde{G}_{11} & \widetilde{G}_{12} \\ 0 & I & \widetilde{G}_{21} & \widetilde{G} \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & \widetilde{A}_1^T & \widetilde{A}^T \end{array}\right]
\end{aligned}$$

where $\widetilde{A}_1, \widetilde{G}_{12} \in \mathbb{R}^{k\times n-k}$, $\widetilde{A}, \widetilde{Q}, \widetilde{G} \in \mathbb{R}^{n-k\times n-k}$. The first k columns of T span the right deflating subspace of $L - \lambda M$ corresponding to k zero eigenvalues and the rows $n+1, n+2, \ldots, n+k$ of $\widetilde{T}$ span the left deflating subspace corresponding to k infinity eigenvalues. We may therefore delete rows and columns $1, 2, \ldots, k$, $n+1, n+2, \ldots, n+k$ and proceed with the reduced pencil

$$\begin{bmatrix} \widetilde{A} & 0 \\ \widetilde{Q} & I \end{bmatrix} - \lambda \begin{bmatrix} I & \widetilde{G} \\ 0 & \widetilde{A} \end{bmatrix}.$$

There is no guarantee that $\widetilde{A}$ is nonsingular. Hence, the procedure described above has to be repeated until the resulting symplectic matrix pencil has no more zero and infinity eigenvalues and $\widetilde{A}$ is nonsingular. Note that neither the rank of A nor the number of zero eigenvalues of A determine the number of zero and infinity eigenvalues of the symplectic pencil. For instance, in Example 10 of [**12**], $n = 6$,

rank$(A) = 5$, A has three zero eigenvalues and $L - \lambda M$ as in (1.2) has two zero and infinite eigenvalues each. All the computation in this algorithm can be carried out in a numerically reliable way. The solution of the linear systems with $I + GQ$ is well-conditioned, since Q and G are symmetric positive semidefinite.

If after the first iteration, $\widetilde{A}$ is nonsingular, then this process requires $(11n^2 + 6rn + 2r^2)n$ flops; the initial QR decomposition in order to check the rank of A costs $\frac{4}{3}n^3$ flops. Note that this initial decomposition is always computed. But in case A has full rank, then the symplectic pencil $L' - \lambda M'$ in (3.1) is formed using the original data in order not to introduce unnecessary rounding errors. In doing so, the QR decomposition of the A matrix should be used when computing $A^{-T}Q$ and A^{-T} instead of computing an LR decomposition of A.

Combined with a strategy to deflate zero and infinity eigenvalues from the symplectic pencil in order to deal with discrete-time algebraic Riccati equations with singular A matrix, the butterfly SZ algorithm is an efficient tool to compute a starting guess for Newton's method. Usually the same number of iterations is required as when refining an approximation computed by the generalized Schur vector method. Even if one or two iterations more are necessary due to the loss of accuracy caused by using non-orthogonal transformations, this is well compensated by the cheaper SZ iteration.

4. A hybrid method for DAREs

Combining the strategy to deflate zero and infinity eigenvalues from the symplectic pencil with the SZ algorithm followed by a few Newton iteration steps results in an efficient and accurate hybrid method.

Altogether, we propose the following algorithm to solve the DARE (1.1).

ALGORITHM 4.1.

Input: The coefficient matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $Q = Q^T \in \mathbb{R}^{n\times n}$, and $R \in \mathbb{R}^{m\times m}$.

Output: An approximation $\tilde{X} = \tilde{X}^T \in \mathbb{R}^{n\times n}$ to the stabilizing solution of the DARE.

1. Form the symplectic pencil $L - \lambda M$ as in (1.2).
2. Use the procedure described in Section 3 to deflate all zero and infinite eigenvalues of $L - \lambda M$. That is, compute a nonsingular transformation matrix T_1 and a symplectic matrix S_1 such that

$$T_1(L - \lambda M)S_1 = \begin{bmatrix} 0 & \tilde{A}_1 & 0 & 0 \\ 0 & \tilde{A} & 0 & 0 \\ 0 & 0 & I_k & 0 \\ 0 & \tilde{Q} & 0 & I_{n-k} \end{bmatrix} - \lambda \begin{bmatrix} I_k & 0 & -\tilde{G}_{11} & -\tilde{G}_{12} \\ 0 & I_{n-k} & -\tilde{G}_{12}^T & -\tilde{G}_{22} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \tilde{A}_1^T & \tilde{A}^T \end{bmatrix}$$

   with $\widetilde{A}$ nonsingular and the first k columns of S_1 span the deflating subspace of $L - \lambda M$ corresponding to all zero eigenvalues.
3. Apply the butterfly SZ algorithm described in Section 3 (and in detail in [**11**]) to the symplectic pencil

$$\tilde{L} - \lambda\tilde{M} := \begin{bmatrix} I_{n-k} & 0 \\ 0 & \tilde{A}^{-T} \end{bmatrix} \begin{bmatrix} \tilde{A} & 0 \\ \tilde{Q} & I_{n-k} \end{bmatrix} - \lambda \begin{bmatrix} I_{n-k} & -\tilde{G}_{22} \\ 0 & \tilde{A}^T \end{bmatrix}$$

such that

$$\tilde{T}_2(\tilde{L}-\lambda\tilde{M})\tilde{S}_2 = \begin{bmatrix} \phi_{11} & \phi_{12} \\ 0 & \phi_{22} \end{bmatrix} - \lambda \begin{bmatrix} \psi_{11} & \psi_{12} \\ 0 & \psi_{22} \end{bmatrix},$$

where the eigenvalues of $\phi_{11} - \lambda\psi_{11}$ are the stable nonzero eigenvalues of $L - \lambda M$.

4. Partition $\tilde{S}_2 = \left[\begin{smallmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{smallmatrix}\right]$ where $S_{jj} \in \mathbb{R}^{n-k\times n-k}$, $j = 1, 2$. Set

$$Z := S_1 \begin{bmatrix} I_k & 0 & 0 & 0 \\ 0 & S_{11} & 0 & S_{12} \\ 0 & 0 & I_k & 0 \\ 0 & S_{21} & 0 & S_{22} \end{bmatrix}.$$

Then the first n columns of Z span the stable deflating subspace of $L - \lambda M$ and an approximate solution $\widehat{X}$ of the DARE can be computed as in (3.3).

5. Use Newton's method (possibly endowed with a line search strategy as proposed in [**9**]) and starting guess $X_0 = \widehat{X}$ in order to iteratively refine the solution of the DARE to the highest achievable accuracy.

Note that all left transformation matrices need not be accumulated. The flop count for Algorithm 4.1 can be summarized assuming that $m = n/2$, $k = 0$, and that two Newton iterations are required to obtain the attainable accuracy: Then, Steps 1. and 2. need $\frac{13}{24}n^3$ flops and $\frac{4}{3}n^3$ flops, respectively, Step 3. requires $\frac{576}{3}n^3$ flops (using the QR decomposition of A computed in Step 2.) while $\frac{8}{3}n^3$ flops are needed to solve the linear system in the 4. Step. Two Newton iteration under the given assumptions take $84n^3$ flops. Altogether this results in approximately $280n^3$ flops. Note that given these flop counts, 5 Newton iterations are needed to reach the cost of the generalized Schur vector method. If compared to an iteratively refined generalized Schur vector method, this comparison becomes even more favorable. For the same number of Newton iterations, the hybrid method is about twice as fast as the iteratively refined generalized Schur vector method.

Also note that as long as the SZ algorithm yields a stabilizing starting guess $X_0 = \widehat{X}$, the hybrid algorithm can be considered as numerically backward stable as it computes the solution to a nearby DARE; see [**25**, § 10]. This is due to the fact that the Newton iteration as formulated in Section 2 can be considered as a defect correction method for DAREs and the ultimate accuracy obtained by Newton's method is only limited by condition number of the DARE; see, e.g., [**27**].

5. Numerical Experiments

We have implemented Algorithm 4.1 as MATLAB[1] functions. The implementation of Newton's method for DAREs used here is described in [**8**]. As stopping criterion for Newton's method we used a criterion based on a normalized residual of the form

$$\|\mathcal{DR}(X_k)\|_F \leq n\varepsilon\|X_k\| \max\{\|A\|_F, \|B\|_F, \|R\|_F, \|Q\|_F\},$$

where ε is the machine precision.

In this section we compare the results obtained by Algorithm 4.1 with those obtained from the Schur vector method as proposed in [**26**], i.e., applying the QZ algorithm to the symplectic pencil $L-\lambda M$ from (1.2) and re-ordering the eigenvalues appropriately, followed by Newton's method in the same implementation as used in our hybrid method. In the tests, results of Newton's method without line search are

[1] MATLAB is a trademark of The MathWorks, Inc.

reported. Only in Example 14, line search reduced the number of required Newton iterations by one in both methods.

The two algorithms are compared for the examples from the DARE benchmark collection [**12, 13**]. We do not give results for Examples 3 and 4 as there, R is singular and hence the approach via the symplectic pencil is not possible.

All computations were done using MATLAB Version 5.3 [**24**] under Linux on a Tangent workstation with a 200 MHz PentiumPro CPU using IEEE double precision arithmetic. The machine precision was $\varepsilon \approx 2.2204 \times 10^{-16}$.

Table 1 reports the residual norms $\|\mathcal{DR}(X)\|_F$ for the solutions computed by the SZ and Schur vector algorithms before and after applying Newton's method and `NIT`, the number of Newton steps required to attain the final accuracy. Moreover, the size n of the problem as well as the condition numbers of A and of the DARE are also provided. The condition K_{DARE} of the DARE is estimated using the method proposed in [**18**]. If $\operatorname{cond}(A) = \|A\|_2 \left\|A^{-1}\right\|_2 = \infty$, then A is singular. In that case, the number in parentheses indicates the number of zero eigenvalues of the symplectic pencil $L - \lambda M$ in (1.2). Recall that these have to be deflated before applying the SZ algorithm. Table 2 gives information about relative errors for those examples where the exact solution is known. The information about the example and the number of iteration steps needed by Newton's method is not duplicated, though. In both tables X_* denotes the exact stabilizing solution, X_{SZ} and X_{QZ} are the initial guesses for Newton's method obtained from the butterfly SZ algorithm and the Schur vector method, respectively, and $X_{\mathtt{NIT}}$ is the final approximation to X_* after `NIT` iteration steps of Newton's method. From both tables it can be seen that for all examples, the final accuracy is similar for both methods. This is to be expected as the accuracy obtained by Newton's method is only limited by the conditioning of the DARE and not by the initial error $\|X_* - X_0\|_F$. In most cases, the number of Newton steps needed is equal or differs by one only. The biggest differences are in Examples 10 and 11. In Example 10, four Newton steps are needed to refine the Schur vector based approximation to final accuracy while no Newton steps are needed by Algorithm 4.1. On the other hand, in Example 11, two Newton steps are needed by the hybrid method while none are required for the Schur vector solution. But even in this case, given the flop counts from the previous sections, Algorithm 4.1 still computes the solution faster than the Schur vector method. That is, the hybrid method is always faster than the Schur vector method with iterative refinement based on Newton's method. In Example 10, Algorithm 4.1 is almost three times faster than the iteratively refined generalized Schur vector method. Note that for all examples with known exact solution, the attained accuracy is of order machine precision times condition of the DARE.

6. Concluding remarks

We have discussed the numerical solution of discrete-time Riccati equations. By initializing Newton's method with a starting guess computed by the butterfly SZ algorithm combined with a method for deflating zero and infinite eigenvalues to the corresponding symplectic matrix pencil, an efficient and accurate hybrid method is derived. This method is usually about twice as fast as the generalized Schur vector method with iterative refinement by Newton's method, while the accuracy obtained is in both cases only limited by the conditioning of the DARE and not by the SZ or QZ method.

Example no.	n	cond (A)	K_{DARE}	Algorithm 4.1			Schur vector method		
				SZ	Newton	NIT	QZ	Newton	NIT
				$\|\|\mathcal{DR}(X_{SZ})\|\|_F$	$\|\|\mathcal{DR}(X_{\mathtt{NIT}})\|\|_F$		$\|\|\mathcal{DR}(X_{SZ})\|\|_F$	$\|\|\mathcal{DR}(X_{\mathtt{NIT}})\|\|_F$	
1	2	1.2×10^2	18.9	1.9×10^{-13}	3.0×10^{-14}	1	4.8×10^{-14}	4.8×10^{-16}	0
2	2	1.1	4.7	1.4×10^{-16}	1.4×10^{-16}	0	4.4×10^{-17}	4.4×10^{-17}	0
5	2	∞ (1)	1.9	0	0	0	2.2×10^{-15}	2.2×10^{-15}	0
6	4	1.0	30.6	9.9×10^{-12}	4.1×10^{-15}	1	9.9×10^{-14}	4.3×10^{-15}	1
7	4	19.9	7.9×10^2	4.3×10^{-12}	2.2×10^{-16}	1	1.1×10^{-15}	1.1×10^{-15}	0
8	4	3.8×10^2	5.1×10^4	5.9×10^{-13}	5.9×10^{-13}	0	8.3×10^{-14}	8.3×10^{-14}	0
9	5	23.5	1.0×10^2	2.2×10^{-9}	8.0×10^{-15}	1	1.9×10^{-13}	5.1×10^{-15}	1
10	6	∞ (2)	3.9	8.3×10^{-15}	8.3×10^{-15}	0	1.02	4.6×10^{-16}	4
11	9	1.6×10^6	74.2	7.5×10^{-4}	1.1×10^{-13}	2	8.3×10^{-12}	8.3×10^{-12}	0
12	2	∞ (2)	2.7	0	0	0	8.1×10^6	0	1
13	3	∞ (1)	2.5	3.3	2.7×10^{-7}	1	3.7×10^{-8}	3.7×10^{-8}	0
14	4	∞ (3)	1.8×10^8	5.8×10^{-2}	0	3	4.1×10^{-2}	1.9×10^{-8}	2
15	100	∞ (100)	2.8×10^2	0	0	0	1.4×10^{-12}	1.4×10^{-12}	0

TABLE 1. Residual norms of SZ and Schur vector solutions before and after iterative refinement.

Example no.	Algorithm 4.1		Schur vector method	
	SZ	Newton	QZ	Newton
	$\frac{\|X_{SZ}-X_*\|_F}{\|X_*\|_F}$	$\frac{\|X_{\text{NIT}}-X_*\|_F}{\|X_*\|_F}$	$\frac{\|X_{QZ}-X_*\|_F}{\|X_*\|_F}$	$\frac{\|X_{\text{NIT}}-X_*\|_F}{\|X_*\|_F}$
1	9.4×10^{-15}	6.1×10^{-15}	4.5×10^{-16}	4.5×10^{-16}
5	0	0	4.2×10^{-16}	4.2×10^{-16}
12	0	0	8.1×10^{-6}	0
13	4.1×10^{-7}	3.2×10^{-14}	4.2×10^{-15}	4.2×10^{-15}
14	4.2×10^{-2}	1.6×10^{-9}	2.9×10^{-2}	1.5×10^{-8}
15	0	0	4.5×10^{-15}	4.5×10^{-15}

TABLE 2. Relative errors of SZ and Schur vector solution before and after iterative refinement.

References

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

[2] B.D.O. Anderson and J.B. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, NJ, 1979.

[3] B.D.O. Anderson and B. Vongpanitlerd. *Network Analysis and Synthesis. A Modern Systems Approach*. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[4] E.S. Armstrong and G. T. Rublein. A stabilization algorithm for linear discrete constant systems. *IEEE Trans. Automat. Control*, AC–21:629–631, 1976.

[5] G. Banse. *Symplektische Eigenwertverfahren zur Lösung zeitdiskreter optimaler Steuerungsprobleme*. Dissertation, Fachbereich 3 – Mathematik und Informatik, Universität Bremen, Bremen, FRG, June 1995. In German.

[6] A. Y. Barraud. A numerical algorithm to solve $A^T X A - X = Q$. *IEEE Trans. Automat. Control*, AC-22:883–885, 1977.

[7] R.H. Bartels and G.W. Stewart. Solution of the matrix equation $AX + XB = C$: Algorithm 432. *Comm. ACM*, 15:820–826, 1972.

[8] P. Benner. *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*. Logos–Verlag, Berlin, Germany, 1997. *Also:* Dissertation, Fakultät für Mathematik, TU Chemnitz–Zwickau, 1997.

[9] P. Benner. Accelerating Newton's method for discrete-time algebraic Riccati equations. In A. Beghi, L. Finesso, and G. Picci, editors, *Mathematical Theory of Networks and Systems*, pages 569–572, Il Poligrafo, Padova, Italy, 1998.

[10] P. Benner and H. Faßbender. The symplectic eigenvalue problem, the butterfly form, the SR algorithm, and the Lanczos method. *Linear Algebra Appl.*, 275/276:19–47, 1998.

[11] P. Benner, H. Faßbender, and D.S. Watkins. SR and SZ algorithms for the symplectic (butterfly) eigenproblem. *Linear Algebra Appl.*, 287:41–76, 1999.

[12] P. Benner, A.J. Laub, and V. Mehrmann. A collection of benchmark examples for the numerical solution of algebraic Riccati equations II: Discrete-time case. Technical Report SPC 95_23, Fakultät für Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1995. Available from `http://www.tu-chemnitz.de/sfb393/spc95pr.html`.

[13] P. Benner, A.J. Laub, and V. Mehrmann. Benchmarks for the numerical solution of algebraic Riccati equations. *IEEE Control Systems Magazine*, 7(5):18–28, 1997.

[14] R. Byers and V. Mehrmann. Symmetric updating of the solution of the algebraic Riccati equation. *Methods of Operations Research*, 54:117–125, 1985.

[15] J. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice Hall, Englewood Cliffs, New Jersey, 1983.

[16] H. Faßbender. *Symplectic Methods for Symplectic Eigenproblems.* Habilitationsschrift, Fachbereich 3 – Mathematik und Informatik, Universität Bremen, 28334 Bremen, (Germany), 1998.
[17] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, third edition, 1996.
[18] T. Gudmundsson, C. Kenney, and A.J. Laub. Scaling of the discrete-time algebraic Riccati equation to enhance stability of the Schur solution method. *IEEE Trans. Automat. Control*, 37:513–518, 1992.
[19] G.A. Hewer. An iterative technique for the computation of steady state gains for the discrete optimal regulator. *IEEE Trans. Automat. Control*, AC-16:382–384, 1971.
[20] D.L. Kleinman. On an iterative technique for Riccati equation computations. *IEEE Trans. Automat. Control*, AC-13:114–115, 1968.
[21] D.L. Kleinman. Stabilizing a discrete, constant, linear system with application to iterative methods for solving the Riccati equation. *IEEE Trans. Automat. Control*, AC-19:252–254, 1974.
[22] P. Lancaster and L. Rodman. *The Algebraic Riccati Equation.* Oxford University Press, Oxford, 1995.
[23] A.J. Laub. Algebraic aspects of generalized eigenvalue problems for solving Riccati equations. In C.I. Byrnes and A. Lindquist, editors, *Computational and Combinatorial Methods in Systems Theory*, pages 213–227. Elsevier (North-Holland), 1986.
[24] The Mathworks, Inc., 24 Prime Park Way, Natick, MA 01760-1500 (USA). *Using* MATLAB *Version 5*, June 1997.
[25] V. Mehrmann. *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution.* Number 163 in Lecture Notes in Control and Information Sciences. Springer-Verlag, Heidelberg, July 1991.
[26] T. Pappas, A.J. Laub, and N.R. Sandell. On the numerical solution of the discrete-time algebraic Riccati equation. *IEEE Trans. Automat. Control*, AC-25:631–641, 1980.
[27] P.H. Petkov, N.D. Christov, and M.M. Konstantinov. *Computational Methods for Linear Control Systems.* Prentice-Hall, Hertfordshire, UK, 1991.
[28] A. Saberi, P. Sannuti, and B.M. Chen. *H_2 Optimal Control.* Prentice-Hall, Hertfordshire, UK, 1995.
[29] V. Sima. *Algorithms for Linear-Quadratic Optimization*, volume 200 of *Pure and Applied Mathematics.* Marcel Dekker, Inc., New York, NY, 1996.
[30] D.S. Watkins and L. Elsner. Theory of decomposition and bulge chasing algorithms for the generalized eigenvalue problem. *SIAM J. Matrix Anal. Appl.*, 15:943–967, 1994.
[31] K. Zhou, J.C. Doyle, and K. Glover. *Robust and Optimal Control.* Prentice-Hall, Upper Saddle River, NJ, 1995.

ZENTRUM FÜR TECHNOMATHEMATIK, FACHBEREICH 3 - MATHEMATIK UND INFORMATIK, UNIVERSITÄT BREMEN, D-28334 BREMEN, GERMANY
E-mail address: heike@math.uni-bremen.de

ZENTRUM FÜR TECHNOMATHEMATIK, FACHBEREICH 3 - MATHEMATIK UND INFORMATIK, UNIVERSITÄT BREMEN, D-28334 BREMEN, GERMANY
E-mail address: benner@math.uni-bremen.de

Berichte aus der Technomathematik ISSN 1435-7968

`http://www.math.uni-bremen.de/zetem/berichte.html`

— Vertrieb durch den Autor —

Reports **Stand: 1. November 1999**

98–01. Peter Benner, Heike Faßbender:
An Implicitly Restarted Symplectic Lanczos Method for the Symplectic Eigenvalue Problem, Juli 1998.

98–02. Heike Faßbender:
Sliding Window Schemes for Discrete Least-Squares Approximation by Trigonometric Polynomials, Juli 1998.

98–03. Peter Benner, Maribel Castillo, Enrique S. Quintana-Ortí:
Parallel Partial Stabilizing Algorithms for Large Linear Control Systems, Juli 1998.

98–04. Peter Benner:
Computational Methods for Linear–Quadratic Optimization, August 1998.

98–05. Peter Benner, Ralph Byers, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Solving Algebraic Riccati Equations on Parallel Computers Using Newton's Method with Exact Line Search, August 1998.

98–06. Lars Grüne, Fabian Wirth:
On the rate of convergence of infinite horizon discounted optimal value functions, November 1998.

98–07. Peter Benner, Volker Mehrmann, Hongguo Xu:
A Note on the Numerical Solution of Complex Hamiltonian and Skew-Hamiltonian Eigenvalue Problems, November 1998.

98–08. Eberhard Bänsch, Burkhard Höhn:
Numerical simulation of a silicon floating zone with a free capillary surface, Dezember 1998.

99–01. Heike Faßbender:
The Parameterized SR Algorithm for Symplectic (Butterfly) Matrices, Februar 1999.

99–02. Heike Faßbender:
Error Analysis of the symplectic Lanczos Method for the symplectic Eigenvalue Problem, März 1999.

99–03. Eberhard Bänsch, Alfred Schmidt:
Simulation of dendritic crystal growth with thermal convection, März 1999.

99–04. Eberhard Bänsch:
Finite element discretization of the Navier-Stokes equations with a free capillary surface, März 1999.

99–05. Peter Benner:
Mathematik in der Berufspraxis, Juli 1999.

99–06. Andrew D.B. Paice, Fabian R. Wirth:
Robustness of nonlinear systems and their domains of attraction, August 1999.

99–07. Peter Benner, Enrique S. Quintana-Ortí, Gregorio Quintana-Ortí:
Balanced Truncation Model Reduction of Large-Scale Dense Systems on Parallel Computers, September 1999.

99–08. Ronald Stöver:
Collocation methods for solving linear differential-algebraic boundary value problems, September 1999.

99–09. Huseyin Akcay:
Modelling with Orthonormal Basis Functions, September 1999.

99–10. Heike Faßbender, D. Steven Mackey, Niloufer Mackey:
Hamilton and Jacobi come full circle: Jacobi algorithms for structured Hamiltonian eigenproblems, Oktober 1999.

99–11. Peter Benner, Vincente Hernández, Antonio Pastor:
On the Kleinman Iteration for Nonstabilizable System, Oktober 1999.

99–12. Peter Benner, Heike Faßbender:
A Hybrid Method for the Numerical Solution of Discrete-Time Algebraic Riccati Equations, November 1999.