Electronic Transactions on Numerical Analysis.
Volume 33, pp. 84-104, 2009.

ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

FAST SOLUTION OF A CERTAIN RICCATI EQUATION THROUGH CAUCHY-LIKE MATRICES*

DARIO A. BINI†, BEATRICE MEINI†, AND FEDERICO POLONI‡

Dedicated to Gérard Meurant on the occasion of his 60th birthday

Abstract. We consider a special instance of the algebraic Riccati equation $XCX - XE - AX + B = 0$ encountered in transport theory, where the $n \times n$ matrix coefficients A, B, C, E are rank structured matrices. The equation is reduced to unilateral form $A_1X^2 + A_0X + A_{-1} = 0$ and solved by means of Cyclic Reduction (CR). It is shown that the matrices generated by CR are Cauchy-like with respect to a suitable singular operator and their displacement structure is explicitly determined. The application of the GKO algorithm provides a method for solving this Riccati equation in $O(n^2)$ arithmetic operations (ops) with quadratic convergence. The structured doubling algorithm is analyzed in the same framework and accelerated to $O(n^2)$ ops as well. In critical cases where convergence turns to linear, we present an adaptation of the shift technique which allows us to get rid of the singularity. Numerical experiments and comparisons which confirm the effectiveness of the new approach are reported.

Key words. nonsymmetric algebraic Riccati equation, cyclic reduction, Cauchy matrix, matrix equation, fast algorithm, M-matrix.

AMS subject classifications. 15A24, 65F05, 65H10

1. Introduction.

The numerical treatment of a problem in transport theory related with the transmission of a neutron beam in a solid medium [19] is reduced to the solution of the following nonsymmetric algebraic Riccati equation (NARE):

$$XCX - XE - AX + B = 0, \tag{1.1}$$

where $A, B, C, E \in \mathbb{R}^{n\times n}$ are given by

$$A = \Delta - eq^T, \quad B = ee^T, \quad C = qq^T, \quad E = D - qe^T, \tag{1.2}$$

and

$$\begin{array}{lll} e = (1, 1, \ldots, 1)^T, & & \\ q = (q_1, q_2, \ldots, q_n)^T & \text{with} & q_i = \frac{c_i}{2\omega_i}, \\ \Delta = \mathrm{diag}(\delta_1, \delta_2, \ldots, \delta_n) & \text{with} & \delta_i = \frac{1}{c\omega_i(1+\alpha)}, \\ D = \mathrm{diag}(d_1, d_2, \ldots, d_n) & \text{with} & d_i = \frac{1}{c\omega_i(1-\alpha)}. \end{array} \tag{1.3}$$

The matrices and vectors above depend on the parameters $0 < c \leqslant 1$, $0 \leqslant \alpha < 1$ and on the sequences $0 < \omega_n < \ldots < \omega_2 < \omega_1 < 1$ and $c_i > 0, i = 1, 2, \ldots, n$, such that $\sum_i c_i = 1$. For the physical meaning of these parameters, we refer the reader to [19] and to the references therein. The solution of interest is the minimal positive one, which exists as proved in [19].

It is important to point out that equation (1.1) with coefficients (1.2), (1.3) originates from the numerical discretization of an integral differential equation where the size n of the unknown X corresponds to the number of nodes used for the numerical integration. Therefore, the larger n is, the more accurate is the approximation of X to the solution of the physical

*Received January 30, 2008. Accepted October 23, 2008. Published online September 17, 2009. Recommended by K. Jbilou.

†Dipartimento di Matematica, Università di Pisa, Largo B. Pontecorvo 5, 56127 Pisa, Italy ({bini, meini}@dm.unipi.it).

‡Scuola Normale Superiore, Piazza dei Cavalieri 6, 56126 Pisa, Italy (poloni@sns.it).

model; thus, large values of n are meaningful in practice. It is therefore important to design fast algorithms for the solution of (1.1) for large values of n.

As shown by Chun-Hua Guo [10], this equation falls in the class of nonsymmetric algebraic Riccati equations associated with a nonsingular M-matrix or a singular irreducible M-matrix; in fact, arranging the coefficients as

$$\mathcal{M} = \begin{bmatrix} E & -C \\ -B & A \end{bmatrix} \tag{1.4}$$

yields an M-matrix. We recall that M is an M-matrix if $M = \theta I - N$, where N has nonnegative entries and θ is greater than or equal to the spectral radius $\rho(N)$ of N [1].

The solution of (1.1) with the assumptions (1.2) and (1.3) can be expressed in closed form in terms of the eigenvalues and eigenvectors of a suitable matrix; see [19] and the more recent paper [22]. However, here we propose *ad hoc* iterative algorithms based on matrix iterations, which, unlike [19] and [22], avoid computing eigenvalues and eigenvectors, fully exploit the structure of the problem, and are easily extendable to more general cases.

In fact, many iterative algorithms based on matrix iterations have been devised in the literature. The main available algorithms for computing the minimal positive solution of this class of algebraic Riccati equations are Newton's method [15], Logarithmic Reduction (LR) [12], Cyclic Reduction (CR) [3] and the Structure-Preserving Doubling Algorithm (SDA) [14, 16]. All these algorithms share the same order of complexity, that is, $O(n^3)$ arithmetic operations (ops) per step, and all provide quadratic convergence in the generic case and linear convergence in critical cases.

$O(n^2)$ complexity algorithms have been designed by L.-Z. Lu [21], but they have linear convergence which turns to sublinear in critical cases. More recently, an algorithm implementing the Newton iteration with $O(n^2)$ ops per step has been obtained relying on properties of certain structured matrices [4].

In this paper, we provide two other algorithms of complexity $O(n^2)$ which maintain the quadratic convergence. The first one relies on a transformation provided by Ramaswami in [23] that allows one to express the matrix X in terms of the solution of a unilateral quadratic matrix equation of the form

$$A_1Y^2 + A_0Y + A_{-1} = 0,$$

for suitable $2n \times 2n$ matrices A_{-1}, A_0, A_1. This equation is solved by means of the cyclic reduction algorithm, which has quadratic convergence in the generic case and complexity $O(n^3)$. We prove that the matrix sequences $\{A_j^{(i)}\}_i$, $j = -1, 0, 1$, generated by CR are such that $\mathcal{D}A_j^{(i)} - A_j^{(i)}\mathcal{D}$ has rank at most 5 for any i and $j = -1, 0, 1$, where $\mathcal{D}$ is a suitable diagonal matrix. Matrices of this kind are known as Cauchy-like. Operators of the kind $X \mapsto D_1X - XD_2$ have been introduced and systematically studied by Georg Heinig and Karla Rost in the book [18]. In particular, we provide the explicit Cauchy representations of these sequences and determine the equations that relate the generators of these matrices at two subsequent steps of the algorithm. This result enables us to provide an algorithm which implements CR with complexity $O(n^2)$ based on a modification of the Gohberg-Kailath-Olshevsky (GKO) algorithm [8].

The second method that we introduce is based on the structured doubling algorithm introduced in [16]. As in the above case, it can be proved that the iterates generated by applying SDA to the problem (1.2) are Cauchy-like, and their generators can be computed explicitly in terms of the involved matrices. This allows one to develop an algorithm that implements the SDA iteration in structured form in $O(n^2)$ operations per step.

Since both Ramaswami's Cyclic Reduction method and the SDA are faster than Newton-based methods for the general NARE (1.1), it is interesting to study the behaviour of such methods also in a structured context, such as the problem (1.2), to see if similar performance comparisons can be achieved relative to the Newton-based method proposed in [4].

In critical cases encountered when $\alpha = 0$, $c = 1$, the convergence of CR and SDA turns to linear. We show that the shift technique of [14], which transforms the critical case into a new non-critical Riccati equation, can still be applied with complexity $O(n^2)$ and with quadratic convergence.

Our algorithms are still valid in the more general case where (1.2) holds with

$$A = \Delta - \widetilde{e}q^T, \quad B = \widetilde{e}e^T, \quad C = \widetilde{q}q^T, \quad E = D - \widetilde{q}e^T,$$

and $e, q, \tilde{e}, \tilde{q} \in \mathbb{R}^{n\times r}$. In this case the complexity is $O(rn^2)$ ops per step while the analogous generalization of the algorithm of [4] based on Newton's iteration would cost $O(r^3n^2)$ ops per step. Observe also that the secular equation approach of [19] and [22] is hardly extendable to this general case, since the eigenvalue problem for a diagonal plus rank r matrix is computationally more difficult than solving a secular equation and the separation intervals for the eigenvalues are not available anymore.

The paper is organized as follows. In Section 2, we introduce some of the tools that are needed to prove our results. In Sections 3 and 4, we show how to develop the structured versions of CR and SDA, respectively. Then, in Section 6, we show that the shift technique can be used by our algorithm with no increasing of the computational cost. Section 7 deals with an alternative implementation of part of the algorithm in order to overcome numerical problems in critical cases. This section is of more general interest since it shows how to replace a singular displacement operator with a nonsingular one with a slight increase of the complexity. Numerical experiments and conclusions follow in the last two sections.

2. Preliminary tools.

2.1. Singular and critical equations. Equation (1.1) is said to be *nonsingular* if $\mathcal{M}$, as in (1.4), is a nonsingular M-matrix. If $\mathcal{M}$ is a singular irreducible M-matrix, let its left and right Perron vectors u^T and v be partitioned accordingly to the definition of $\mathcal{M}$ as $u^T = \begin{bmatrix} u_1^T & u_2^T \end{bmatrix}$, $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$; the equation is said to be

- *transient*, if $u_1^T v_1 - u_2^T v_2 > 0$;
- *positive recurrent*, if $u_1^T v_1 - u_2^T v_2 < 0$;
- *null recurrent*, or *critical*, if $u_1^T v_1 - u_2^T v_2 = 0$.

It has been proved in [10, 11] that equation (1.1) is nonsingular if $c < 1$, transient if $c = 1$ and $\alpha > 0$, and null recurrent if $c = 1$ and $\alpha = 0$.

2.2. Transforming a Riccati equation into unilateral form. It has been proved by Ramaswami in [23] (see also [12]) that S is the minimal nonnegative solution of (1.1) if and only if the matrix

$$G = \begin{bmatrix} I - tE + tCS & 0 \\ S & 0 \end{bmatrix} \tag{2.1}$$

is the minimal nonnegative solution of the following unilateral equation

$$A_1 Y^2 + A_0 Y + A_{-1} = 0, \tag{2.2}$$

where

$$A_{-1} = \begin{bmatrix} I - tE & 0 \\ tB & 0 \end{bmatrix}, \quad A_0 = \begin{bmatrix} -I & tC \\ 0 & -I - tA \end{bmatrix}, \quad A_1 = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}, \tag{2.3}$$

ETNA
Kent State University
http://etna.math.kent.edu

and t is such that $1/t \geq \max\{e_{i,i}, a_{i,i} : \quad i = 1, \ldots, n\}$.

2.3. Cyclic reduction. We briefly recall the cyclic reduction algorithm for computing the minimal nonnegative solution of (2.2) and its convergence properties [2, 5]. Let $A_i^{(0)} = A_i$, $i = -1, 0, 1$, and $\widehat{A}^{(0)} = A_0$, and for $k \geq 0$ define the sequences

$$\begin{aligned} &A_0^{(k+1)} = A_0^{(k)} - A_{-1}^{(k)} K^{(k)} A_1^{(k)} - A_1^{(k)} K^{(k)} A_{-1}^{(k)}, \quad K^{(k)} = \left(A_0^{(k)}\right)^{-1}, \\ &A_{-1}^{(k+1)} = -A_{-1}^{(k)} K^{(k)} A_{-1}^{(k)}, \quad A_1^{(k+1)} = -A_1^{(k)} K^{(k)} A_1^{(k)}, \\ &\widehat{A}^{(k+1)} = \widehat{A}^{(k)} - A_1^{(k)} K^{(k)} A_{-1}^{(k)}. \end{aligned} \tag{2.4}$$

Since $\mathcal{M}$ is a nonsingular M-matrix, or an irreducible singular M-matrix, the conditions of applicability ($\det A_0^{(k)} \neq 0$) and convergence of CR are satisfied [2, 5, 12]. In particular, the sequence

$$G^{(k)} = -\left(\widehat{A}^{(k)}\right)^{-1} A_{-1} \tag{2.5}$$

converges to G. The following result holds [5, 12].

THEOREM 2.1. *If* (1.1) *is*

- *nonsingular, then* $\lim_k A_{-1}^{(k)} = \lim_k A_1^{(k)} = 0$ *with quadratic convergence, and* $\lim_k G^{(k)} = G$ *with quadratic convergence.*
- *transient, then* $\lim_k A_1^{(k)} = 0$, $\lim_k A_{-1}^{(k)} = A_{-1}^*$, $\lim_k G^{(k)} = G$ *with quadratic convergence;*
- *positive recurrent, then* $\lim_k A_1^{(k)} = A_1^*$, $\lim_k A_{-1}^{(k)} = 0$, $\lim_k G^{(k)} = G$ *with quadratic convergence;*
- *null recurrent, then* $\lim_k A_1^{(k)} = A_1^*$, $\lim_k A_{-1}^{(k)} = A_{-1}^*$, $\lim_k G^{(k)} = G$ *with linear convergence.*

The last case is known as the *critical case*. For the problem defined by (1.2) and (1.3), we fall in this case only for $c = 1, \alpha = 0$, as proved in [19].

A useful formulation which enables us to perform a structure analysis of the matrix sequences generated by CR is the functional formulation provided in [5]. Let $\varphi^{(k)}(z) = zA_1^{(k)} + A_0^{(k)} + z^{-1}A_{-1}^{(k)}$ and let $\psi^{(k)}(z) = \varphi^{(k)}(z)^{-1}$ where z is a complex variable and $\psi^{(k)}(z)$ is defined for the values of z such that $\det \varphi^{(k)}(z) \neq 0$. The following equation can be easily verified

$$\psi^{(k+1)}(z^2) = \frac{1}{2}(\psi^{(k)}(z) + \psi^{(k)}(-z)). \tag{2.6}$$

2.4. Structured doubling algorithm. The structured doubling algorithm [16] is another algorithm for computing the solution of a nonsymmetric algebraic Riccati equation. The algorithm can be described as follows. Choose $\gamma \geq \max\{e_{i,i}, a_{i,i} : \quad i = 1, \ldots, n\}$; let

$$W = A + \gamma I - B(E + \gamma I)^{-1}C, \quad V = E + \gamma I - C(A + \gamma I)^{-1}B,$$

and

$$\begin{aligned} E_0 &= I - 2\gamma V^{-1}, \\ F_0 &= I - 2\gamma W^{-1}, \\ G_0 &= 2\gamma(E + \gamma I)^{-1}CW^{-1}, \\ H_0 &= 2\gamma W^{-1}B(E + \gamma I)^{-1}. \end{aligned} \tag{2.7}$$

For $k \geq 0$, calculate

$$\begin{aligned}
E_{k+1} &= E_k(I - G_k H_k)^{-1} E_k, \\
F_{k+1} &= F_k(I - H_k G_k)^{-1} F_k, \\
G_{k+1} &= G_k + E_k(I - G_k H_k)^{-1} G_k F_k, \\
H_{k+1} &= H_k + F_k(I - H_k G_k)^{-1} H_k E_k.
\end{aligned} \tag{2.8}$$

We have the following convergence result [14, 16].

THEOREM 2.2. *If* (1.1) *is*

- *nonsingular, then* $\lim_k E_k = \lim_k F_k = 0$ *with quadratic convergence, and* $\lim_k H_k = S$ *with quadratic convergence.*
- *transient, then* $\lim_k F_k = 0$, $\lim_k E_k = E_*$, $\lim_k H_k = S$ *with quadratic convergence;*
- *positive recurrent, then* $\lim_k F_k = F_*$, $\lim_k E_k = 0$, $\lim_k H_k = S$ *with quadratic convergence;*
- *null recurrent, then* $\lim_k F_k = F_*$, $\lim_k E_k = E_*$, $\lim_k H_k = S$ *with linear convergence.*

2.5. Cauchy-like matrices and the GKO algorithm. A *displacement operator* is an operator $\mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ of the form $\nabla_{R,S} : M \mapsto RM - MS$, with $R, S \in \mathbb{R}^{n\times n}$. It is easy to prove the following algebraic properties of displacement operators.

LEMMA 2.3 (properties of displacement operators). *Let* $R, S, T, M, N, D, \Delta \in \mathbb{R}^{n\times n}$, *with* D *and* Δ *diagonal. Then,*

1. $\nabla_{D,D}(\Delta) = 0$*;*
2. $\nabla_{R,S}(M + N) = \nabla_{R,S}(M) + \nabla_{R,S}(N)$*;*
3. $\nabla_{R,S}(MN) = \nabla_{R,T}(M)N + M\,\nabla_{T,S}(N)$*;*
4. $\nabla_{R,S}(M^{-1}) = -M^{-1}\,\nabla_{S,R}(M)M^{-1}$.

A matrix C is called *Cauchy-like* if there are diagonal matrices $R = \operatorname{diag}(r_1, \dots, r_n)$ and $S = \operatorname{diag}(s_1, \dots, s_n)$, with $r_i \neq s_j$ for all i, j, such that

$$\nabla_{R,S}(C) = uv^T, \tag{2.9}$$

where $u, v \in \mathbb{R}^{n\times r}$ and r is small with respect to n, i.e., if $\nabla_{R,S}(C)$ has low rank. Note that C is uniquely determined by its *generators* u, v and the two vectors $\begin{bmatrix} r_1 & \dots & r_n \end{bmatrix}^T$ and $\begin{bmatrix} s_1 & \dots & s_n \end{bmatrix}^T$ by means of the equation

$$c_{i,j} = \frac{1}{r_i - s_j} \sum_{\ell=1}^{r} u_{i,\ell} v_{j,\ell}.$$

We will call a matrix T *Trummer-like*[1] if there is a diagonal matrix $D = \operatorname{diag}(d_1, \dots, d_n)$, with $d_i \neq d_j$ for all $i \neq j$, such that

$$\nabla_{D,D}(T) = uv^T, \tag{2.10}$$

where $u, v \in \mathbb{R}^{n\times r}$, and r is small with respect to n, i.e., if $\nabla_{D,D}(T)$ is low-rank. Note that $\nabla_{D,D}$ is a singular operator, its kernel being the set of all diagonal matrices, and therefore the displacement equation determines only the off-diagonal part of T. It follows that

[1] The name comes from the so-called *Trummer problem*; see [7] and the references therein for further details.

T is uniquely determined by its generators u, v and the two vectors $\begin{bmatrix} d_1 & \dots & d_n \end{bmatrix}^T$ and $\begin{bmatrix} t_{11} & \dots & t_{nn} \end{bmatrix}^T$ (the latter one being the main diagonal of T).

Using the relations (2.9) and (2.10) we can easily reconstruct a Cauchy-like or a Trummer-like matrix from its generators with $O(rn^2)$ arithmetic operations; reconstructing the matrix and then applying the usual matrix-matrix product yields an algorithm for multiplying an $n \times s$ Cauchy-like (Trummer-like) matrix and a generic $n \times s$ matrix in $O(n^2(r+s))$ ops. We refer to these algorithms as Algorithm 2.4.

ALGORITHM 2.4. *Cauchy-like (Trummer-like) matrix-matrix product*

```
function y = camm(r, s, u, v, x)
% returns y = C * x, where C satisfies
% diag(r) * C - C * diag(s) = u * v'
% x may be a vector or a matrix
  n = size(u, 1);
  C = (u * v') ./ (r * ones(1, n) - ones(n, 1) * (s).');
  y = C * x;
end function
function y = trmm(d, dg, u, v, x)
% returns y = T * x, where T satisfies
% diag(d) * T - T * diag(d) = u * v'
% and diag(T) = dg
  n = size(u, 1);
  T = (u * v') ./ (d * ones(1, n) - ones(n, 1) * (d).' + eye(n));
  for i = 1:n
    T(i,i) = dg(i);
  end for
  y = T * x;
end function
```

The problem of solving a linear system with Cauchy matrix C was treated by Gohberg, Kailath and Olshevsky in [8]. Their algorithm, known as the GKO algorithm, is based on the fact that the Schur complement of certain Cauchy-like matrices is Cauchy-like. In our case, if

$$\begin{bmatrix} r_1 & 0 \\ 0 & R_2 \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & C_{22} \end{bmatrix} - \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & C_{22} \end{bmatrix} \begin{bmatrix} s_1 & 0 \\ 0 & S_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ U_2 \end{bmatrix} \begin{bmatrix} v_1 & V_2 \end{bmatrix},$$

where r_1, s_1 and $c_{1,1}$ are scalars, c_{21} and v_i are column vectors, and c_{12}, u_1 are row vectors, then the Schur complement $\hat{C} = C_{22} - c_{21}c_{11}^{-1}c_{12}$ solves the displacement equation

$$R_2\hat{C} - \hat{C}S_2 = (U_2 - \frac{1}{c_{11}}c_{21}u_1)(V_2 - \frac{1}{c_{11}}c_{12}v_1). \tag{2.11}$$

That is, $\nabla_{R_2,S_2}(\widehat{C})$ has rank at most r.

Using this fact, one can perform Gaussian elimination on the matrix C in $O(n^2 r)$ operations: at each step, instead of computing the entries of the Schur complement of C, one computes its generators as given by (2.11). Combining this algorithm with the customary back-substitution, we can solve a linear system with Cauchy-like matrix and $n \times s$ constant coefficient in $O(n^2(r+s))$ operations.

The same technique can be used for systems with a Trummer-like matrix T, as shown in [4], with the additional complication that knowing the generators of $\hat{T}$ is not sufficient

ETNA
Kent State University
http://etna.math.kent.edu

to reconstruct the Schur complement. In fact, we need to compute its diagonal elements separately: at each step, we update the diagonal of T as we were performing a customary Gaussian elimination, and we compute the generators of the off-diagonal elements as in the GKO algorithm. The resulting algorithm is presented as Algorithm 2.5. The algorithm can be combined with partial pivoting to improve its numerical stability, though this will not be necessary in the following since it will only be applied to nonsingular M-matrices.

ALGORITHM 2.5. *Solution of a linear system with Trummer-like matrix*

```
function x = trsv(d, dg, u, v, b)
% returns x = T^{-1} b, where T satisfies
% diag(d) * T - T * diag(d) = u * v'
% and diag(T) = dg
% precondition: Gaussian elimination without pivoting
%  is stable for T
  n = size(u, 1);
  U = zeros(n);   %matrix U of the LU factorization
  l = zeros(n,1); %active column of L of the LU factorization
  x = b;
  for k = 1:n
    if(abs(dg(k)) < 1.d-10) warn "Pivot is too small!";
    % generates a column of L and solves L^{-1}*b on-the-fly
    l(k+1:n) = ((u(k+1:n,:) * (v(k,:))')/dg(k)) ./ (d(k+1:n)-d(k));
    x(k+1:n) = x(k+1:n) - l(k+1:n) * x(k);
    % generates a row of U
    U(k,k) = dg(k);
    U(k,k+1:n) = (u(k,:) * (v(k+1:n,:))' ) ./ (d(k)-d(k+1:n)).';
    % updates the generators to generators of the Schur complement
    u(k+1:n,:) = u(k+1:n,:) - l(k+1:n) * u(k,:);
    v(k+1:n,:) = v(k+1:n,:) - (U(k,k+1:n)') * v(k,:)  / dg(k);
    % updates the diagonal
    dg(k+1:n) = dg(k+1:n) - l(k+1:n) * U(k,k+1:n);
  end for
  % solves the resulting upper triangular system
  x=U\x
end function
```

3. Structure analysis of Cyclic Reduction and the main algorithm. In the following, we consider the case of Riccati equations of the form (1.1) with

$$A = \Delta - \widetilde{e}q^T, \quad B = \widetilde{e}e^T, \quad C = \widetilde{q}q^T, \quad E = D - \widetilde{q}e^T, \tag{3.1}$$

such that $e, q, \widetilde{e}, \widetilde{q} \in \mathbb{R}^{n\times r}$ are positive, and $D, \Delta \in \mathbb{R}^{n\times n}$ are diagonal with positive diagonal entries. Moreover, we ask that $\mathcal{M}$ (as in (1.4)) is an M-matrix, so that the theorems in [10] ensure that a minimal solution exists. Note that setting $r = 1, \widetilde{e} = e, \widetilde{q} = q$ yields (1.2). Here we make no assumptions on the rank of the $n \times r$ matrices involved in (3.1). In fact, the theoretical analysis holds true in general. From the computational point of view it is more convenient to have full rank matrices.

3.1. Block structure. By performing the cyclic reduction with initial matrices of the form (2.3), some structures are preserved in the matrix sequences $\widehat{A}^{(k)}$, $A_i^{(k)}$, $i = -1, 0, 1$:

THEOREM 3.1. *Let $A_{-1}^{(k)}, A_0^{(k)}, \widehat{A}^{(k)}, A_1^{(k)}$, $k \geqslant 0$, be the matrix sequences generated by the CR (2.4) with initial matrices (2.3). Then,*

ETNA
Kent State University
http://etna.math.kent.edu

1. *The matrix sequences are of the form*

$$A_{-1}^{(k)} = \begin{bmatrix} * & 0 \\ * & 0 \end{bmatrix},\ A_0^{(k)} = \begin{bmatrix} -I & * \\ * & * \end{bmatrix},\ \widehat{A}^{(k)} = \begin{bmatrix} -I & tC \\ * & -I - tA \end{bmatrix},\ A_1^{(k)} = \begin{bmatrix} 0 & 0 \\ 0 & * \end{bmatrix},$$

where $*$ denotes a generic $n \times n$ block.

2. *The $(2,1)$ block of $A_0^{(k)}$ and $\widehat{A}^{(k)}$ are the same matrix.*

Proof. All results can be easily proved by induction, noticing how the zero blocks are distributed among the matrices. In particular, the second part follows by observing that, in the formulas (2.4) for updating $A_0^{(k+1)}$ and $\widehat{A}^{(k+1)}$, the term $-A_{-1}^{(k)}K^{(k)}A_1^{(k)}$ only modifies the $(2,1)$ block, and the term $-A_1^{(k)}K^{(k)}A_{-1}^{(k)}$ only modifies the second block column. ◻

3.2. Rank structure. Consider the unilateral matrix equation (2.2) with block coefficients (2.3). For the matrix function $\varphi^{(0)}(z) = A_{-1}z^{-1} + A_0 + A_1 z$, we get

$$\varphi^{(0)}(z) = \begin{bmatrix} (I - tD)z^{-1} - I & 0 \\ 0 & zI - (I + t\Delta) \end{bmatrix} + t \begin{bmatrix} \widetilde{q} \\ \widetilde{e} \end{bmatrix} \begin{bmatrix} z^{-1}e^T & q^T \end{bmatrix}.$$

Using the Sherman-Morrison formula [9] to invert $\varphi^{(0)}(z)$, we have

$$\psi^{(0)}(z) = (\varphi^{(0)}(z))^{-1} = Z(z) - Z(z) \begin{bmatrix} \widetilde{q} \\ \widetilde{e} \end{bmatrix} r(z) \begin{bmatrix} z^{-1}e^T & q^T \end{bmatrix} Z(z)$$

with

$$Z(z) = \begin{bmatrix} (I - tD)z^{-1} - I & 0 \\ 0 & zI - (I + t\Delta) \end{bmatrix}^{-1},$$
$$r(z) = t\left(I_r + t \begin{bmatrix} z^{-1}e^T & q^T \end{bmatrix} Z(z) \begin{bmatrix} \widetilde{q} \\ \widetilde{e} \end{bmatrix} \right)^{-1}.$$

Now, since

$$\mathcal{D}Z(z) = Z(z)\mathcal{D} = \begin{bmatrix} zI & 0 \\ 0 & -I \end{bmatrix} + zZ(z) \text{ with } \mathcal{D} = \begin{bmatrix} I - tD & 0 \\ 0 & I + t\Delta \end{bmatrix},$$

we find that

$$\begin{aligned} \nabla_{\mathcal{D},\mathcal{D}}(\psi^{(0)}(z)) = &- \mathcal{D}Z(z) \begin{bmatrix} \widetilde{q} \\ \widetilde{e} \end{bmatrix} r(z) \begin{bmatrix} z^{-1}e^T & q^T \end{bmatrix} Z(z) \\ &+ Z(z) \begin{bmatrix} \widetilde{q} \\ \widetilde{e} \end{bmatrix} r(z) \begin{bmatrix} z^{-1}e^T & q^T \end{bmatrix} Z(z)\mathcal{D} \\ = &\begin{bmatrix} -z\widetilde{q} \\ \widetilde{e} \end{bmatrix} r(z) \begin{bmatrix} z^{-1}e^T & q^T \end{bmatrix} Z(z) + Z(z) \begin{bmatrix} \widetilde{q} \\ \widetilde{e} \end{bmatrix} r(z) \begin{bmatrix} e^T & -q^T \end{bmatrix}. \end{aligned}$$

Setting

$$\begin{aligned} \tilde{s}^{(0)T}(z) &= -zr(z) \begin{bmatrix} z^{-1}e^T & q^T \end{bmatrix} Z(z), \\ \tilde{t}^{(0)T}(z) &= r(z) \begin{bmatrix} z^{-1}e^T & q^T \end{bmatrix} Z(z), \\ \tilde{u}^{(0)}(z) &= Z(z) \begin{bmatrix} \widetilde{q} \\ \widetilde{e} \end{bmatrix} r(z), \end{aligned}$$

yields

$$\nabla_{\mathcal{D},\mathcal{D}}(\psi^{(0)}(z)) = \begin{bmatrix}\widetilde{q}\\0\end{bmatrix}\tilde{s}^{(0)T}(z) + \begin{bmatrix}0\\\widetilde{e}\end{bmatrix}\tilde{t}^{(0)T}(z) + \tilde{u}^{(0)}(z)\begin{bmatrix}e^T & -q^T\end{bmatrix}.$$

Using the functional formulation (2.6) of CR and the linearity of $\nabla_{\mathcal{D},\mathcal{D}}$, we can easily prove by induction that

$$\nabla_{\mathcal{D},\mathcal{D}}(\psi^{(k)}(z)) = \begin{bmatrix}\widetilde{q}\\0\end{bmatrix}\tilde{s}^{(k)T}(z) + \begin{bmatrix}0\\\widetilde{e}\end{bmatrix}\tilde{t}^{(k)T}(z) + \tilde{u}^{(k)}(z)\begin{bmatrix}e^T & -q^T\end{bmatrix} \tag{3.2}$$

for each $k \geqslant 0$, with

$$\begin{aligned}\tilde{s}^{(k+1)}(z^2) &= \frac{1}{2}(\tilde{s}^{(k)}(z) + \tilde{s}^{(k)}(-z)),\\ \tilde{t}^{(k+1)}(z^2) &= \frac{1}{2}(\tilde{t}^{(k)}(z) + \tilde{t}^{(k)}(-z)),\\ \tilde{u}^{(k+1)}(z^2) &= \frac{1}{2}(\tilde{u}^{(k)}(z) + \tilde{u}^{(k)}(-z)).\end{aligned}$$

Therefore, $\psi^{(k)}(z)$ has displacement rank $3r$ for all $k \geqslant 0$. Also, $\varphi^{(k)}(z)$ has displacement rank $3r$, since

$$\nabla_{\mathcal{D},\mathcal{D}}(\varphi^{(k)}(z)) = \nabla_{\mathcal{D},\mathcal{D}}\left(\psi^{(k)}(z)^{-1}\right) = -\varphi^{(k)}(z)\left(\nabla_{\mathcal{D},\mathcal{D}}(\psi^{(k)}(z))\right)\varphi^{(k)}(z)$$

by part 4 of Lemma 2.3.

Let

$$\begin{aligned}
v_{-1}^{(k)} &= A_{-1}^{(k)}\begin{bmatrix}\widetilde{q}\\0\end{bmatrix}, & r_{-1}^{(k)T} &= \begin{bmatrix}e^T & -q^T\end{bmatrix}A_{-1}^{(k)},\\
v_{0}^{(k)} &= A_{0}^{(k)}\begin{bmatrix}\widetilde{q}\\0\end{bmatrix}, & r_{0}^{(k)T} &= \begin{bmatrix}e^T & -q^T\end{bmatrix}A_{0}^{(k)},\\
w_{0}^{(k)} &= A_{0}^{(k)}\begin{bmatrix}0\\\widetilde{e}\end{bmatrix}, & r_{1}^{(k)T} &= \begin{bmatrix}e^T & -q^T\end{bmatrix}A_{1}^{(k)},\\
w_{1}^{(k)} &= A_{1}^{(k)}\begin{bmatrix}0\\\widetilde{e}\end{bmatrix}, & &\\
u_0 &= -\begin{bmatrix}\widetilde{q}\\\widetilde{e}\end{bmatrix}. & &
\end{aligned} \tag{3.3}$$

Note that these vectors also can be defined by recurrence as follows:

$$\begin{aligned}
v_{-1}^{(0)} &= A_{-1}^{(0)}\begin{bmatrix}\widetilde{q}\\0\end{bmatrix}, & v_{-1}^{(k+1)} &= -A_{-1}^{(k)}K^{(k)}v_{-1}^{(k)},\\
v_{0}^{(0)} &= A_{0}^{(0)}\begin{bmatrix}\widetilde{q}\\0\end{bmatrix}, & v_{0}^{(k+1)} &= v_0^{(k)} - A_{1}^{(k)}K^{(k)}v_{-1}^{(k)},\\
w_{0}^{(0)} &= A_{0}^{(0)}\begin{bmatrix}0\\\widetilde{e}\end{bmatrix}, & w_{0}^{(k+1)} &= w_0^{(k)} - A_{-1}^{(k)}K^{(k)}w_{1}^{(k)},\\
w_{1}^{(0)} &= A_{1}^{(0)}\begin{bmatrix}0\\\widetilde{e}\end{bmatrix}, & w_{1}^{(k+1)} &= -A_{1}^{(k)}K^{(k)}w_{1}^{(k)},\\
r_{-1}^{(0)T} &= \begin{bmatrix}e^T & -q^T\end{bmatrix}A_{-1}^{(0)}, & r_{-1}^{(k+1)T} &= -r_{-1}^{(k)T}K^{(k)}A_{-1}^{(k)},\\
r_{0}^{(0)T} &= \begin{bmatrix}e^T & -q^T\end{bmatrix}A_{0}^{(0)}, & r_{0}^{(k+1)T} &= r_0^{(k)} - r_{-1}^{(k)T}K^{(k)}A_{1}^{(k)} - r_{1}^{(k)T}K^{(k)}A_{-1}^{(k)},\\
r_{1}^{(0)T} &= \begin{bmatrix}e^T & -q^T\end{bmatrix}A_{1}^{(0)}, & r_{1}^{(k+1)T} &= -r_{1}^{(k)T}K^{(k)}A_{1}^{(k)}.
\end{aligned} \tag{3.4}$$

ETNA
Kent State University
http://etna.math.kent.edu

Also, define

$$\begin{aligned} t_{-1}^{(0)T} &= -\begin{bmatrix} e^T & 0 \end{bmatrix}, \quad t_{-1}^{(k+1)T} = -t_{-1}^{(k)T} K^{(k)} A_{-1}^{(k)}, \\ t_0^{(0)T} &= -\begin{bmatrix} 0 & q^T \end{bmatrix}, \quad t_0^{(k+1)T} = t_0^{(k)T} - t_{-1}^{(k)T} K^{(k)} A_1^{(k)}. \end{aligned} \tag{3.5}$$

Finally, set

$$s_0^{(k)T} = r_0^{(k)T} \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix},$$

and notice that in view of the block structure of Theorem 3.1,

$$s_0^{(k+1)T} = s_0^{(k)T} - r_1^{(k)T} K^{(k)} A_{-1}^{(k)}.$$

We can now prove a more precise result concerning the structure of $\varphi^{(k)}(z)$.

THEOREM 3.2. *Let $\varphi^{(k)}(z) = zA_1^{(k)} + A_0^{(k)} + z^{-1}A_{-1}^{(k)}$ be the sequence generated by the application of cyclic reduction to* (2.3) *for the problem* (3.1). *Then,*

$$\begin{aligned} \nabla_{\mathcal{D},\mathcal{D}}(A_{-1}^{(k)}) =& v_{-1}^{(k)} s_0^{(k)T} + w_0^{(k)} t_{-1}^{(k)T} + u_0 r_{-1}^{(k)T}, \\ \nabla_{\mathcal{D},\mathcal{D}}(A_0^{(k)}) =& v_{-1}^{(k)} r_1^{(k)T} + v_0^{(k)} s_0^{(k)T} + w_0^{(k)} t_0^{(k)T} + w_1^{(k)} t_{-1}^{(k)T} + u_0 r_0^{(k)T}, \\ \nabla_{\mathcal{D},\mathcal{D}}(A_1^{(k)}) =& (v_0^{(k)} + u_0) r_1^{(k)T} + w_1^{(k)} t_0^{(k)T}. \end{aligned} \tag{3.6}$$

Proof. The result holds by mathematical induction. The base step is a simple verification; concerning the inductive step, for the sake of brevity, we will only present the analysis relative to $A_{-1}^{(k)}$, since the cases of $A_0^{(k)}$ and $A_1^{(k)}$ are very similar. In view of Lemma 2.3 , from (2.4) we have

$$\begin{aligned} &\nabla_{\mathcal{D},\mathcal{D}}(A_{-1}^{(k+1)}) = \nabla_{\mathcal{D},\mathcal{D}}\left(-A_{-1}^{(k)} K^{(k)} A_{-1}^{(k)}\right) \\ =& -\nabla_{\mathcal{D},\mathcal{D}}\left(A_{-1}^{(k)}\right) K^{(k)} A_{-1}^{(k)} - A_{-1}^{(k)} \nabla_{\mathcal{D},\mathcal{D}}\left(K^{(k)}\right) A_{-1}^{(k)} - A_{-1}^{(k)} K^{(k)} \nabla_{\mathcal{D},\mathcal{D}}\left(A_{-1}^{(k)}\right) \\ =& -\nabla_{\mathcal{D},\mathcal{D}}\left(A_{-1}^{(k)}\right) K^{(k)} A_{-1}^{(k)} + A_{-1}^{(k)} K^{(k)} \nabla_{\mathcal{D},\mathcal{D}}\left(A_0^{(k)}\right) K^{(k)} A_{-1}^{(k)} - A_{-1}^{(k)} K^{(k)} \nabla_{\mathcal{D},\mathcal{D}}\left(A_{-1}^{(k)}\right) \\ =& -\left(v_{-1}^{(k)} s_0^{(k)T} + w_0^{(k)} t_{-1}^{(k)T} + u_0 r_{-1}^{(k)T}\right) K^{(k)} A_{-1}^{(k)} + \\ & A_{-1}^{(k)} K^{(k)} \left(v_{-1}^{(k)} r_1^{(k)T} + v_0^{(k)} s_0^{(k)T} + w_0^{(k)} t_0^{(k)T} + w_1^{(k)} t_{-1}^{(k)T} + u_0 r_0^{(k)T}\right) K^{(k)} A_{-1}^{(k)} \\ & - A_{-1}^{(k)} K^{(k)} \left(v_{-1}^{(k)} s_0^{(k)T} + w_0^{(k)} t_{-1}^{(k)T} + u_0 r_{-1}^{(k)T}\right) \\ =& - A_{-1}^{(k)} K^{(k)} v_{-1}^{(k)} \left(s_0^{(k)T} - r_1^{(k)T} K^{(k)} A_{-1}^{(k)}\right) - \left(w_0^{(k)} - A_{-1}^{(k)} K^{(k)} w_1^{(k)}\right) t_{-1}^{(k)T} K^{(k)} A_{-1}^{(k)} \\ & - u_0^{(k)} r_{-1}^{(k)} K^{(k)} A_1^{(k)} \\ =& v_{-1}^{(k+1)} s_0^{(k+1)T} + w_0^{(k+1)} t_{-1}^{(k+1)T} + u_0^{(k+1)} r_{-1}^{(k+1)T}. \end{aligned}$$

Here, we made use of the following facts:

- $A_{-1} K v_0 = v_{-1}$ and $r_0 K A_{-1} = r_{-1}$, which follows from the definitions of $K^{(k)}$ and (3.3);
- $A_{-1} K w_0 = A_{-1} \begin{bmatrix} 0 \\ \tilde{e} \end{bmatrix} = 0$, due to the position of the zero blocks in $A_{-1}^{(k)}$ and $A_1^{(k)}$, as proved in Theorem 3.1. □

ETNA
Kent State University
http://etna.math.kent.edu

We can say more about the meaning of $t_0^{(k)T}$ and $t_{-1}^{(k)T}$. Let us consider first the $(2,1)$ block of $A_{-1}^{(k)}$; let it be called $F^{(k)}$. Selecting the second block row of the formula for the update of $A_{-1}^{(k)}$, we have

$$\begin{bmatrix} F^{(k+1)} & 0 \end{bmatrix} = -\begin{bmatrix} F^{(k)} & 0 \end{bmatrix} K^{(k)} A_{-1}^{(k)}$$

and $F^{(0)} = t\widetilde{e}e^T$. Thus, by comparing this formula with that of $t_{-1}^{(k)T}$ in (3.5), we see that $\begin{bmatrix} F^{(k)} & 0 \end{bmatrix} = -t\widetilde{e}t_{-1}^{(k)T}$.

Let us now turn to consider the $(2,2)$ block of $A_0^{(k)}$; let it be called $G^{(k)}$. From the block structure of the iterates, it follows that in the update formula (2.4) for $A_0^{(k)}$ only the term $A_{-1}^{(k)}K^{(k)}A_1^{(k)}$ updates the block $G^{(k)}$. Writing down explicitly the update relation, we get

$$\begin{bmatrix} 0 & G^{(k+1)} \end{bmatrix} = \begin{bmatrix} 0 & G^{(k)} \end{bmatrix} - \begin{bmatrix} F^{(k)} & 0 \end{bmatrix} K^{(k)} A_1^{(k)}$$

and $G^{(0)} = -I - t(\Delta - \widetilde{e}q^T)$. By using the above result and combining this update formula with that of $t_0^{(k)T}$ in (3.5), we can see that

$$\begin{bmatrix} 0 & G^{(k)} \end{bmatrix} = \begin{bmatrix} 0 & -I - t\Delta \end{bmatrix} - t\widetilde{e}t_0^{(k)T}. \tag{3.7}$$

3.3. The main algorithm. The structure relations (3.6) allow us to develop a faster version of the CR iteration with computational cost $O(n^2 r)$ per step. In fact, Algorithms 2.4 and 2.5 allow us to perform fast computations with $A_{-1}^{(k)}$, $A_0^{(k)}$ and $A_1^{(k)}$ using only the generators of these matrices. At each step k of the cyclic reduction, we only need to store and update the nine $n \times r$ generators

$$v_{-1}^{(k)}, \quad v_0^{(k)}, \quad w_0^{(k)}, \quad w_1^{(k)}, \quad r_{-1}^{(k)}, \quad r_0^{(k)}, \quad r_1^{(k)}, \quad t_0^{(k)}, \quad t_1^{(k)}. \tag{3.8}$$

Note that $s^{(k)}$ need not be explicitly stored because it easily can be recovered from $r_0^{(k)}$.

Our plan is to perform the update using the Trummer-like operations introduced in Section 2.5. However, in order to do that, we need a method to calculate the diagonal entries of the involved matrices, since they cannot be recovered from the generators, nor computed explicitly during the algorithm (without resorting to an $O(n^3)$ algorithm). Notice first that a Trummer-like matrix T can be written as

$$\operatorname{diag}(T) + \operatorname{Trummer}(\mathcal{D}, U, V),$$

where $\operatorname{Trummer}(\mathcal{D}, U, V)$ is the only Trummer-like matrix with respect to $\nabla_{\mathcal{D},\mathcal{D}}$ with generators U, V and zeroes on the diagonal. Therefore, for any vector x, we have

$$Tx - \operatorname{Trummer}(\mathcal{D}, U, V)x = \operatorname{diag}(T)x,$$

so that knowing the generators of T and the vector product Tx, we can recover T_{ii} as

$$t_{ii} = \frac{(Tx - \operatorname{Trummer}(\mathcal{D}, U, V))_{ii}}{x_i}, \tag{3.9}$$

provided that x_i is nonzero. We can use this technique to calculate the diagonal entries of the $A_i^{(k)}$ which are not known *a priori* from the above results, that is, the first n diagonal entries of $A_{-1}^{(k)}$ and the last n of $A_1^{(k)}$ (note that, in view of Theorem 3.1 and (3.7), the diagonal of

ETNA
Kent State University
http://etna.math.kent.edu

$A_0^{(k)}$ can be easily determined). As for the vector x, it is possible to choose a column of respectively $\begin{bmatrix}\widetilde{q}\\0\end{bmatrix}$ or $\begin{bmatrix}0\\\widetilde{e}\end{bmatrix}$. After recovering the diagonal entries of the $A_i^{(k)}$, we can update the generators using the Trummer matrix-vector arithmetic to perform the updates described in the right-hand part of equations (3.4) and (3.5).

Notice that we do not need to store nor update $\widehat{A}^{(k)}$ explicitly in order to recover the solution X as the $(2,1)$ block of $-\left(\widehat{A}^{(k)}\right)^{-1} A_{-1}^{(0)}$ at the end of the algorithm, because of (2.1) and (2.5). In fact, Theorem 3.1 shows that the four blocks of $\widehat{A}^{(k)}$ are known; moreover, the Schur complement $Z = \widehat{A}_{22}^{(k)} + A_{21}^{(k)} \widehat{A}_{12}^{(k)}$ is diagonal plus rank r, as it can be written as

$$Z = -I - t\Delta + t(\widetilde{e} + A_{21}^{(k)} \widetilde{q}) q^T, \tag{3.10}$$

in view of (3.1) and Theorem 3.1. It is therefore possible to find X by applying block Schur complementation to the system $-\left(\widehat{A}^{(k)}\right)^{-1} (A_{-1}^{(0)} \begin{bmatrix} I & 0\end{bmatrix})$, which yields

$$ZX = tB + A_{12}^{(k)}(I - tE). \tag{3.11}$$

The resulting system involving Z can be solved in $O(rn^2)$ ops using, e.g., the Sherman–Morrison formula or the GKO algorithm (since a diagonal plus rank r matrix is Trummer-like with rank $2r$ with respect to any diagonal matrix D).

The algorithm, whose total cost is $O(rn^2)$, is briefly sketched in Algorithm 3.3.

ALGORITHM 3.3. *Structured cyclic reduction*

```
function X=fastcr(D,Δ,e,q,ẽ,q̃)
 k=0;
 initialize the generator matrices (using (2.4), (3.4) and (3.5))
 do
  k=k+1;
  calculate the diagonals of A_i^(k) with (3.9)
  update the generators using (3.4) and (3.5)
 while(stopping criterion)
  calculate the entries of the (2,1) block K of A_0^(k)
  build the Schur complement Z of Â^(k) according to (3.10)
  solve (3.11) with the Sherman—Morrison formula or GKO
end function
```

An obvious choice for the stopping criterion would be to compute the iterate $X^{(k)}$ at each step and explicitly calculate the residual of the Riccati equation (1.1). However, this is quite expensive. Theorem 2.1 provides another good choice. In all three cases, the sequences $A_{-1}^{(k)}$ and $A_1^{(k)}$ converge; therefore, we can simply check that the norms of the two values

$$A_{-1}^{(k+1)} \begin{bmatrix}\widetilde{q}\\0\end{bmatrix} - A_{-1}^{(k)} \begin{bmatrix}\widetilde{q}\\0\end{bmatrix}, \quad A_1^{(k+1)} \begin{bmatrix}0\\\widetilde{e}\end{bmatrix} - A_1^{(k)} \begin{bmatrix}0\\\widetilde{e}\end{bmatrix}$$

are small enough. This can be done with a small overhead, since the values we need are two of the nine generators and thus are already computed at each step. In the noncritical case, another viable choice is checking that

$$\min\left(\left\| A_{-1}^{(k)} \begin{bmatrix}\widetilde{q}\\0\end{bmatrix}\right\|_1, \left\| A_1^{(k)} \begin{bmatrix}0\\\widetilde{e}\end{bmatrix}\right\|_1\right) < \varepsilon,$$

since at least one of $A_{-1}^{(k)}$ and $A_1^{(k)}$ converges to zero by Theorem 2.1.

Note also that the algorithm can be accelerated by skipping the computations with the zero blocks, thus reducing all the computations to $n \times n$ matrix computations. This way, one sees that one only needs to update ten $n \times r$ matrices (instead of nine $2n \times r$) at each step.

4. Structure analysis of SDA. The structure analysis of SDA, following the same strategy, leads to less cumbersome computations. Let

$$\mathcal{H} = \begin{bmatrix} E & -C \\ B & -A \end{bmatrix}, \quad \mathcal{H}_\gamma = (H + \gamma I)^{-1}(H - \gamma I),$$

and suppose that H_γ is nonsingular. For the problem (3.1), we have

$$\mathcal{H} = \begin{bmatrix} D & 0 \\ 0 & -\Delta \end{bmatrix} - \begin{bmatrix} \widetilde{q} \\ -\widetilde{e} \end{bmatrix} \begin{bmatrix} e^T & q^T \end{bmatrix}.$$

The matrices $\mathcal{H}$ and $\mathcal{H}_\gamma$ commute, since the latter is a rational function of the former. This fact implies that

$$\begin{bmatrix} D & 0 \\ 0 & -\Delta \end{bmatrix} \mathcal{H}_\gamma^{2^k} - \mathcal{H}_\gamma^{2^k} \begin{bmatrix} D & 0 \\ 0 & -\Delta \end{bmatrix} = \begin{bmatrix} \widetilde{q} \\ -\widetilde{e} \end{bmatrix} \begin{bmatrix} e^T & q^T \end{bmatrix} \mathcal{H}_\gamma^{2^k} - \mathcal{H}_\gamma^{2^k} \begin{bmatrix} \widetilde{q} \\ -\widetilde{e} \end{bmatrix} \begin{bmatrix} e^T & q^T \end{bmatrix}, \tag{4.1}$$

which shows that $\mathcal{H}_\gamma^{2^k}$ has low displacement rank with respect to a suitable (singular) operator.

It follows from the results on matrix pencils presented in [16], or also by direct verification from equations (2.7) and (2.8), that

$$\mathcal{H}_\gamma^{2^k} = \begin{bmatrix} I & -G_k \\ 0 & F_k \end{bmatrix}^{-1} \begin{bmatrix} E_k & 0 \\ -H_k & I \end{bmatrix}.$$

Using this relation, it is easy to check that

$$\begin{aligned} &\begin{bmatrix} I & -G_k \end{bmatrix} \mathcal{H}_\gamma^{2^k} = \begin{bmatrix} E_k & 0 \end{bmatrix}, \qquad && \begin{bmatrix} 0 & F_k \end{bmatrix} \mathcal{H}_\gamma^{2^k} = \begin{bmatrix} -H_k & I \end{bmatrix}, \\ &\mathcal{H}_\gamma^{2^k} \begin{bmatrix} I \\ H_k \end{bmatrix} = \begin{bmatrix} E_k \\ 0 \end{bmatrix}, && \mathcal{H}_\gamma^{2^k} \begin{bmatrix} 0 \\ F_k \end{bmatrix} = \begin{bmatrix} G_k \\ I \end{bmatrix}. \end{aligned} \tag{4.2}$$

Now, multiply (4.1) by $\begin{bmatrix} 0 & F_k \end{bmatrix}$ to the left and by $\begin{bmatrix} 0 \\ F_k \end{bmatrix}$ to the right, to get

$$-F_k\Delta + \Delta F_k = -F_k\widetilde{e}(q^T + e^T G_k) + (H_k\widetilde{q} + \widetilde{e})q^T F_k.$$

Similarly, multiplying (4.1) by either $\begin{bmatrix} 0 & F_k \end{bmatrix}$ or $\begin{bmatrix} I & -G_k \end{bmatrix}$ to the left and either $\begin{bmatrix} 0 \\ F_k \end{bmatrix}$ or $\begin{bmatrix} I \\ H_k \end{bmatrix}$ to the right, in all four combinations, yields equations

$$\begin{aligned} DE_k - E_k D &= (\widetilde{q} + G_k\widetilde{e})e^T E_k - E_k\widetilde{q}(e^T + q^T H_k), \\ \Delta F_k - F_k\Delta &= (H_k\widetilde{q} + \widetilde{e})q^T F_k - F_k\widetilde{e}(q^T + e^T G_k), \\ DG_k + G_k\Delta &= (\widetilde{q} + G_k\widetilde{e})(e^T G_k + q^T) - E_k\widetilde{q}q^T F_k, \\ \Delta H_k + H_k D &= (H_k\widetilde{q} + \widetilde{e})(e^T + q^T H_k) - F_k\widetilde{e}e^T E_k, \end{aligned} \tag{4.3}$$

which provide low displacement rank representations for the matrix sequences generated by SDA. Using these relations, we proceed along the lines of Algorithm 3.3. At each step, we only store in memory the eight generators

$$\begin{aligned} v_1^{(k)} &= E_k\widetilde{q}, & v_2^{(k)} &= F_k\widetilde{e}, & v_3^{(k)} &= (\widetilde{q} + G_k\widetilde{e}), & v_4^{(k)} &= H_k\widetilde{q} + \widetilde{e}, \\ w_1^{(k)T} &= e^T E_k, & w_2^{(k)T} &= q^T F_k, & w_3^{(k)T} &= e^T G_k + q^T, & w_4^{(k)T} &= e^T + q^T H_k, \end{aligned} \tag{4.4}$$

and update them accordingly to the following relations (easily derived from (2.8))

$$\begin{aligned} v_1^{(k+1)} &= E_k(I - G_kH_k)^{-1}v_1^{(k)}, \\ v_2^{(k+1)} &= F_k(I - H_kG_k)^{-1}v_2^{(k)}, \\ v_3^{(k+1)} &= v_3^{(k)} + E_k(I - G_kH_k)^{-1}G_kv_2^{(k)}, \\ v_4^{(k+1)} &= v_4^{(k)} + F_k(I - H_kG_k)^{-1}H_kv_1^{(k)} \\ w_1^{(k+1)T} &= w_1^{(k)T}(I - G_kH_k)^{-1}E_k \\ w_2^{(k+1)T} &= w_2^{(k)T}(I - H_kG_k)^{-1}F_k \\ w_3^{(k+1)T} &= w_3^{(k)T} + w_1^{(k)}(I - G_kH_k)^{-1}G_kF_k \\ w_4^{(k+1)T} &= w_4^{(k)T} + w_2^{(k)}(I - H_kG_k)^{-1}H_kE_k \end{aligned} \tag{4.5}$$

using the Cauchy- and Trummer-like structure (4.3) to carry out the computations. Note that $I - G_kH_k$ and $I - H_kG_k$ are Trummer-like, and their generators can be computed using Lemma 2.3; e.g., for $I - G_kH_k$, we have

$$\nabla_{D,D}(I - G_kH_k) = \nabla_{D,D}(I) - \nabla_{D,D}(G_kH_k) = 0 - \nabla_{D,-\Delta}(G_k)H_k - G_k\nabla_{-\Delta,D}(H_k).$$

In addition, we have to keep track of the diagonals of E_k and F_k in order to perform the computations. In the same fashion as CR, these diagonals can be recovered using formula (3.9) using as x one of the columns of $v_1^{(k)} = E_k\widetilde{q}$, $v_2^{(k)} = F_k\widetilde{e}$.

ALGORITHM 4.1. *Structured SDA*

```
function X=fastsda(D,Δ,e,q,ẽ,q̃)
 k=0;
 initialize the generator matrices using (2.7)
 do
  k=k+1;
  calculate the diagonals of E_k,F_k with (3.9)
  update the generators using (4.5)
 while(stopping criterion)
 return X=H_k
end function
```

As a stopping criterion, in the noncritical case, we can use the fact that the sequence $\min(\left\|e^TE_k\right\|_1, \|F_k\widetilde{e}\|_1)$ converges quadratically to zero.

5. On differences between CR and SDA. In view of the similar structure and behaviour of the two algorithms we have presented, one may be lead to think that there is a deep connection between them. Recently, some new results on the relationships between Cyclic Reduction and the SDA have been presented in [6]. It turns out that the SDA can be

ETNA
Kent State University
http://etna.math.kent.edu

expressed as a form of CR applied to a unilateral equation (2.2) whose coefficients are substantially different from those in (2.3). The convergence behaviour of CR for (2.2) depends essentially on the ratio μ_1/μ_2, where μ_1 is the largest (in modulus) eigenvalue of the matrix pencil associated with (2.2) inside the unit circle, and μ_2 is the smallest one outside it; see, e.g., [5]. These eigenvalues are given in terms of those of

$$\mathcal{K} = \begin{bmatrix} E & -C \\ B & -A \end{bmatrix} \tag{5.1}$$

by $\mu_1 = f(\lambda_1)$, $\mu_2 = f(\lambda_2)$, where λ_1, λ_2 are the two eigenvalues closer to zero (on the two different sides of the complex axis) of $\mathcal{K}$, $f(x) = I - tx$ for Ramaswami's CR and $f(x) = \frac{x-\gamma}{x+\gamma}$ in the case of the SDA. Therefore, it turns out that the expressions for the convergence ratio of the two algorithms in terms of the λ_i are significatively different, especially when the shift technique is applied to remove the zero eigenvalues of $\mathcal{K}$ in the singular case. Depending on the location of the eigenvalues of $\mathcal{K}$ in the complex plane, one method may attain a faster convergence than the other. Moreover, since (2.3) is definitely simpler than its analogue for the SDA (which, in the end, is equivalent to (2.7)), there might be a small loss of precision implied in the computations to get the initial values of the two methods. In fact, in the numerical experiments that we are presenting, it turns out that in their structured versions SDA is slightly faster than CR, but also less accurate.

6. The shift technique. In the critical case $c = 1, \alpha = 0$ of the NARE (1.2), several drawbacks are encountered. As reported in Theorems 2.1 and 2.2, the convergence of the presented algorithms is linear instead of quadratic; moreover, it has been shown in [13] that an $O(\varepsilon)$ perturbation to the coefficients of the equation leads to an $O(\sqrt{\varepsilon})$ variation in the solution. All these drawbacks can be removed by means of the shift technique originally introduced be He, Meini, and Rhee in [17] and applied to algebraic Riccati equations in [3, 4, 14].

The shift technique applied to this problem consists in replacing the Riccati equation (1.1), with the coefficient matrices given by (3.1), with the equation

$$X\widetilde{C}X - X\widetilde{E} - \widetilde{A}X + \widetilde{B} = 0, \tag{6.1}$$

with

$$\widetilde{A} = A - \eta v_2 p_2^T, \quad \widetilde{B} = B + \eta v_2 p_1^T, \quad \widetilde{C} = C - \eta v_1 p_2^T, \quad \widetilde{E} = E + \eta v_1 p_1^T, \tag{6.2}$$

where

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

is the right Perron vector of the M-matrix M defined in Equation (1.4), and $p^T = \begin{bmatrix} p_1^T & p_2^T \end{bmatrix} \in \mathbb{R}^{1\times 2n}$ is any positive row vector such that $p^T v = 1$, and $\eta > 0$ is such that $\mathcal{M} + \eta v p^T$ is still an M-matrix. It is proved in [14] that the minimal nonnegative solution of (1.1) is the minimal nonnegative solution of (6.1), and that the latter is noncritical. Therefore, cyclic reduction and SDA applied to this problem converge quadratically to the common minimal nonnegative solution S of the two equations.

As noted in [4], a natural choice for p^T in the problem (1.2) is $p^T = \begin{bmatrix} e^T & q^T \end{bmatrix}$, $0 < \eta \leqslant d_1$, which preserves the property that the corresponding $\widetilde{M}$ as defined in (1.3) is a diagonal plus rank one M-matrix, and therefore allows one to use the algorithm presented here for the case (3.1) with $r = 1$ and

$$\begin{bmatrix} \widetilde{q} \\ \widetilde{e} \end{bmatrix} = \begin{bmatrix} q - \eta v_1 \\ e + \eta v_2 \end{bmatrix}.$$

ETNA
Kent State University
http://etna.math.kent.edu

More generally, in the case (3.1) one can apply the same technique choosing p^T as one of the rows of $\begin{bmatrix} e^T & q^T \end{bmatrix}$ (suitably scaled), and a value of η chosen such that the column with the same index of $\begin{bmatrix} \widetilde{q} \\ \widetilde{e} \end{bmatrix}$ is (entry-wise) greater than ηv. This choice preserves the property that the corresponding $\widetilde{M}$ is a diagonal plus rank r M-matrix, maintaining the problem in the form of equation (3.1) without having to increase the value of r.

7. Another approach to the computation of the diagonal. The algorithms presented in the previous sections provide an effective tool for solving equation (1.1) under the assumptions (1.2) and (1.3). However, in the critical and nearly critical cases where (α, c) is close to $(0, 1)$, numerical instability problems are encountered in the computation of the diagonal entries of the involved matrices via the formula (3.9). This phenomenon is well illustrated in the numerical experiments of Section 8. The cancellation error in formula (3.9) in a single iteration is not particularly high, but its accumulation through the successive iterations of the algorithm has a dramatic impact on the accuracy of the computed solution.

In this section, we propose an attempt to cure this drawback, which in fact is a general technique for transforming a singular displacement operator $\nabla_{\mathcal{D},\mathcal{D}}$ into a new nonsingular operator $\nabla_{\mathcal{D}_1,\mathcal{D}_2}$ for which $\operatorname{rank} \nabla_{\mathcal{D}_1,\mathcal{D}_2}(A) \leq \operatorname{rank} \nabla_{\mathcal{D},\mathcal{D}}(A) + 1$.

For a given vector u, rewrite $R = \nabla_{\mathcal{D},\mathcal{D}}(A) = \mathcal{D}A - A\mathcal{D}$ as

$$R = \mathcal{D}A - A(\mathcal{D} + uu^T - uu^T),$$

so that

$$\mathcal{D}A - A(\mathcal{D} + uu^T) = R - Auu^T =: R_1,$$

where $\operatorname{rank} R_1 \leq \operatorname{rank}(R) + 1$. Moreover, if u is chosen as one of the displacement generators of R, then $\operatorname{rank} R_1 \leq \operatorname{rank} R$. Assume for simplicity that $D = \operatorname{diag}(\gamma_1, \ldots, \gamma_{2n})$, where $0 < \gamma_1 < \cdots < \gamma_n < \gamma_{n+1} < \cdots < \gamma_{2n}$. Set

$$\xi_i = (\gamma_i + \gamma_{i+1})/2, \;\; i = 1, 2, \ldots, 2n - 1, \;\; \xi_{2n} > \gamma_{2n}. \tag{7.1}$$

Then it is easily verified that the quantities

$$\sigma_i = \frac{\prod_{j=1}^{2n} (\xi_j - \gamma_i)}{\prod_{j=1,\; j \neq i}^{2n} (\gamma_j - \gamma_i)} \tag{7.2}$$

are positive and that the matrix $\mathcal{D} + uu^T$, with

$$u_i = \sqrt{\sigma_i}, \tag{7.3}$$

has eigenvalues $\xi_1, \ldots, \xi_{2n}$. Moreover, it is a simple matter to prove that the vector $v^{(j)} = (v_i^{(j)})$, where $v_i^{(j)} = u_i \theta_j / (\xi_j - \gamma_i)$ and

$$\theta_j = \left(\sum_i (u_i / (\gamma_i - \xi_j))^2 \right)^{-1/2}, \tag{7.4}$$

is a normalized eigenvector of $\mathcal{D} + uu^T$ corresponding to the eigenvalue ξ_j. In other words, it holds that

$$(D + uu^T)S = SD_1, \quad S = (\theta_j u_i / (\xi_j - \gamma_i)), \;\; D_1 = \operatorname{diag}(\xi_1, \ldots, \xi_{2n}), \tag{7.5}$$

ETNA
Kent State University
http://etna.math.kent.edu

where $SS^T = I$. Thus, one has

$$DAS - ASD_1 = RS - Auu^T S,$$

and the operator ∇_{D,D_1} is nonsingular.

This approach would allow one to apply the standard Cauchy-like matrix machinery to perform the computation of CR (or SDA) by replacing (3.8) by new expressions. However, in this way we would lose the block structure of the vectors involved in (3.8) with an increase of complexity.

It is also possible to use the new nonsingular operator only for computing the diagonal elements of $A_{-1}^{(k)}$ and $A_1^{(k)}$. For the sake of notational simplicity, let us use A for one of the above matrices and let $\nabla_{\mathcal{D},\mathcal{D}}(A) = \sum_{\ell=1}^{3} v^{(\ell)} w^{(\ell)T} = vw^T$, where the vectors $v^{(\ell)}$ and $w^{(\ell)}$ represent the vectors in the right-hand side of (3.6), and $v = \begin{bmatrix} v_1, v_2, v_3 \end{bmatrix}$, $w = \begin{bmatrix} w_1, w_2, w_3 \end{bmatrix}$. Denote $B = AS$ and observe that

$$a_{i,i} = \sum_j (B)_{i,j}(S)_{i,j} = -u_i \sum_j \frac{\theta_j \left(vw^T S + (Au)u^T S \right)}{(\xi_j - \gamma_i)^2}. \tag{7.6}$$

In the case where $A = A_i^{(k+1)}$, $i = 1, -1$, it holds that

$$A_i^{(k+1)} u = -A_i^{(k)} (A_0^{(k)})^{-1} A_i^{(k)} u, \tag{7.7}$$

so that the product Au can be computed using the Trummer-like representation of the matrices $A_i^{(k)}$. In the SDA case, similarly, $E_{k+1}u$ and $F_{k+1}u$ can be computed from the Trummer-like representations of the matrices at step k.

Algorithm 7.1 synthesizes the computation of the diagonal entries of a Trummer-like matrix A given its generators and the product Au (for a suitable vector u).

ALGORITHM 7.1. *Computation of the diagonal of an $m \times m$ Trummer-like matrix A*

```
# precomputation step:
choose u
compute:
  ξ_i, i = 1:m by means of (7.1)
  u_i, i = 1:m by means of (7.2) and (7.3)
  θ_i. i = 1:m by means of (7.4)
  u^T S, where S is as defined in (7.5), using Algorithm 2.4

function d=altdiag(D,v,w,Au)
# output:
# d = diag(A).
# input:
# D (diagonal matrix), v, w such that
# DA − AD = vw^T
# and Au=A*u
# Au can be computed in any way, e.g. using (7.7)
 compute w^T S by using Algorithm 2.4
 for i=1:m
  apply (7.6) and obtain d(i) = a_{i,i}
 end for
end function
```

ETNA
Kent State University
http://etna.math.kent.edu

n	BIP	Lu	SDA	CR	sSDA	sCR
32	2.E-04	2.E-04	1.4E-03	9.9E-04	1.4E-03	1.6E-03
64	4.E-04	1.8E-03	9.4E-03	7.2E-03	4.8E-03	5.0E-03
128	2.2E-03	1.4E-02	7.5E-02	5.9E-02	1.7E-02	1.8E-02
256	1.2E-02	1.0E-01	1.7E+00	1.0E+00	7.2E-02	7.2E-02
512	7.3E-02	1.2E+00	3.5E+01	1.8E+01	3.9E-01	3.3E-01
1024	3.4E-01	9.9E+00		1.6E+02	1.6E+00	1.3E+00
2048	1.3E+00	8.1E+01		1.2E+03	6.5E+00	5.3E+00
4096	6.25E+00	7.0E+02			2.8E+01	2.4E+00

TABLE 8.1
CPU time for a single iteration of the methods for several dimensions n of the equation.

n	BIP	Lu	SDA	CR	sSDA	sCR
32	9.5E-15 (4)	1.7E-14 (4)	4.8E-13 (11)	8.2E-13 (12)	2.2E-14 (11)	9.5E-16 (12)
64	1.9E-14 (4)	4.4E-14 (4)	2.4E-12 (12)	6.0E-12 (13)	4.2E-14 (12)	1.6E-15 (13)
128	4.2E-14 (4)	1.1E-13 (4)	1.8E-11 (13)	3.0E-11 (14)	1.2E-13 (13)	6.4E-15 (14)
256	1.2E-13 (4)	3.9E-13 (4)	1.4E-10 (14)	2.0E-10 (15)	7.6E-13 (14)	1.8E-14 (15)
512	3.0E-13 (4)	1.2E-12 (4)	6.4E-10 (15)	1.0E-09 (16)	3.4E-12 (15)	2.7E-14 (16)
1024	9.0E-13 (4)	3.5E-12 (4)		9.2E-09 (17)	1.4E-11 (16)	7.3E-14 (17)
2048	2.4E-12 (4)	9.3E-12 (4)			5.4E-11 (17)	6.0E-13 (18)
4096	7.0E-12 (4)	2.8E-11 (4)			4.5E-10 (18)	7.1E-12 (19)

TABLE 8.2
Best accuracy (rel. residual) reached (and in parentheses number of iterations required) for $\alpha = 0.5, c = 0.5$.

8. Numerical experiments. The proposed algorithms have been implemented in Fortran 90 and tested on a 2.8 GHz Xeon biprocessor, compiled with Lahey Fortran compiler v. 6.20c. The experiments performed are those mentioned in [20]: that is, equation (1.1) for (1.2) with $\alpha = 0.5, c = 0.5$ (nonsingular case) and with $\alpha = 10^{-8}, c = 1 - 10^{-6}$ (close to null recurrent case). We have let the dimension n of the matrices vary between 32 and 4096 to get a better grasp on the growth of the computational cost.

The algorithms have been compared to the original version of SDA and CR with Ramaswami's reduction, to the algorithm mentioned in [20] (Lu), which is an $O(n^3)$ Newton-like algorithm specialized for problem (1.2), and to the algorithm in [4] (BIP), which is the structure-preserving $O(n^2)$ version of Lu. Our structure-preserving algorithms are labeled sSDA and sCR in the legend, to distinguish them from the original versions. In table 8.1, we show the CPU time needed for a single iteration of each method for different sizes n of the matrices appearing in the NARE. The CPU time has been calculated by averaging over five successive iterations. In tables 8.2 and 8.3 we report the minimum relative residual, calculated as

$$Res = \frac{\left\|\Delta\widetilde{X} + \widetilde{X}D - (\widetilde{X}q + e)(q^T\widetilde{X} + e^T)\right\|_1}{\max(\left\|\widetilde{X}\widetilde{q} + \widetilde{e}\right\|_1, \left\|e^T + q^T\widetilde{X}\right\|_1)},$$

that can be achieved with the different methods, and the number of iterations needed to actually achieve this residual. Note that the expression appearing in the numerator inside the norm symbols is an alternative way of expressing the residual of the Riccati equation (1.2). For practical reasons, some of the most time-consuming experiments with the non-structured $O(n^3)$ versions of SDA and CR were not performed.

The results are encouraging in terms of computational time. The structured versions of the algorithms perform better than their non-structured counterparts starting from a very low threshold for the size n; further on, the structured algorithms also outperform Lu's algorithm

ETNA
Kent State University
http://etna.math.kent.edu

n	BIP	Lu	SDA	CR	sSDA	sCR
32	6.0E-14 (15)	1.6E-14 (13)	5.0E-13 (19)	7.5E-13 (20)	2.0E-08 (18)	4.6E-11 (19)
64	2.8E-13 (13)	4.9E-14 (13)	2.0E-12 (20)	5.7E-12 (21)	2.6E-05 (17)	4.1E-09 (20)
128	6.8E-13 (13)	1.4E-13 (13)	1.6E-11 (21)	4.8E-11 (22)	8.5E-04 (17)	5.9E-08 (20)
256	2.0E-12 (13)	4.2E-13 (13)	9.5E-11 (22)	2.2E-10 (22)	2.8E-03 (17)	2.6E-06 (20)
512	6.5E-12 (13)	1.4E-12 (13)	1.1E-09 (22)	1.4E-09 (23)	5.5E-02 (17)	4.2E-05 (20)
1024	1.7E-11 (13)	4.8E-12 (13)			2.8E-01 (17)	4.2E-04 (19)
2048	4.9E-11 (13)	1.5E-11 (13)			7.7E-01 (18)	5.3E-04 (21)
4096	1.4E-10 (13)	4.1E-11 (13)			2.1E+00 (19)	8.3E-03 (20)

TABLE 8.3
Best accuracy (rel. residual) reached (and in parentheses number of iterations required) for $\alpha = 10^{-8}, c = 1 - 10^{-6}$.

[20], which is faster for low dimensions but scales as $O(n^3)$ with the size of the problem instead of $O(n^2)$.

In terms of accuracy, the algorithms perform well for cases far from singularity, but show very large residuals for critical and near-critical cases. Eventually, for sufficiently high dimension, the convergence is lost. Based on the intermediate results, we guess that the loss of accuracy is due to the computation of the diagonal entries of the Trummer-like matrices by formula (3.9), which suffers from cancellation problems.

In order to get around this problem, we have developed the technique of Section 7. Though changing the involved displacement operator seems the way to overcome the problem, application of Algorithm 7.1 has provided no significant improvement in the errors. The problem of how to reduce the error in near-critical cases is still under investigation.

9. Conclusions. This work provides a structural analysis of SDA and CR for diagonal plus low-rank equations. It is noteworthy that both algorithms preserve the Cauchy-like structure of the iterates, since this is not at all apparent from their definitions.

The presented algorithms provide a new approach to the solution of the structured algebraic Riccati equations (1.2) and (3.1). While their speed is definitely inferior to that of the structured Lu method presented in [4], the most recently developed numerical algorithm for this NARE, they compare favorably to the previous ones. An interesting application would be applying them to equations of the kind (3.1) ($\mathcal{H}$ diagonal plus rank-r) with r in the range 10–15 or larger. For such equations, the analogous generalization of the structured Lu methods has complexity $O(r^3n^2)$, as can be deduced from the derivation in [4] (the Sherman–Morrison complement of the Jacobian matrix is $nr \times nr$ with block Cauchy structure, but full matrix arithmetic on the $r \times r$ blocks is needed in the computation). Instead, structured SDA and CR have complexity $O(rn^2)$. Therefore, it is expected that our methods become competitive with structured Lu starting from a very small value of r.

Turning to numerical stability, some more work is needed to get stable versions of our algorithms for near-critical cases. Apparently, the cancellation problems in the calculation of the diagonal of the Trummer-like matrices cannot be overcome easily. An alternative to the direct calculation of the diagonals (which would require $O(n^3)$ ops) and to the methods presented here (which do not solve the stability issues) is required. Different techniques, alternative to the approach of Section 7, might be changing the displacement operator from the beginning and doing all the computations using the new operator; or introducing a new operator for each step of the algorithms. In this framework, SDA would appear simpler to analyze than CR.

An interesting question, which was raised by one of the anonymous referees of this paper, is whether we can apply the shift technique even when the equation is near-critical. From the spectral analysis of the problem (see, e.g., [14]), it follows that the properties of the NARE

are related to the eigenvalues of $\mathcal{K}$ (as defined in (5.1)). The critical case corresponds to the case in which $\mathcal{K}$ has a double eigenvalue at zero; in this case, it also holds that the Perron vector of $\mathcal{M}$ is the only eigenvector of $\mathcal{K}$ relative to 0. However, in the near-critical case, this nice relation between the eigenvectors of $\mathcal{M}$ and $\mathcal{K}$ is lost. One would need to work with $\mathcal{K}$ to calculate the eigenvector which is needed for the shift technique to work, but this is prone to numerical issues, as the near-critical case corresponds to the one in which $\mathcal{K}$ has two eigenvalues very close to zero. It could help to use a method to calculate the eigenspace relative to these two eigenvalues without attempting to separate them, such as an inverse subspace iteration with dimension 2.

Acknowledgment. We wish to thank two anonymous referees who have provided many relevant and detailed comments that enabled us to substantially improve the presentation.

REFERENCES

[1] A. BERMAN AND R. J. PLEMMONS, *Nonnegative Matrices in the Mathematical Sciences*, SIAM, Philadelphia, 1994. Revised reprint of the 1979 original.

[2] D. A. BINI, L. GEMIGNANI, AND B. MEINI, *Computations with infinite Toeplitz matrices and polynomials*, Linear Algebra Appl., 343/344 (2002), pp. 21–61.

[3] D. A. BINI, B. IANNAZZO, G. LATOUCHE, AND B. MEINI, *On the solution of algebraic Riccati equations arising in fluid queues*, Linear Algebra Appl., 413 (2006), pp. 474–494.

[4] D. A. BINI, B. IANNAZZO, AND F. POLONI, *A fast Newton's method for a nonsymmetric algebraic Riccati equation*, SIAM. J. Matrix Anal. Appl., 30 (2008), pp. 276–290.

[5] D. A. BINI, G. LATOUCHE, AND B. MEINI, *Numerical Methods for Structured Markov Chains*, Oxford University Press, New York, 2005.

[6] D. A. BINI, B. MEINI, AND F. POLONI, *From algebraic Riccati equations to unilateral quadratic matrix equations: old and new algorithms*, in Numerical Methods for Structured Markov Chains, D. Bini, B. Meini, V. Ramaswami, M.-A. Remiche, and P. Taylor, eds., Dagstuhl Seminar Proceedings, 07461, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Dagstuhl, Germany, 2008.

[7] A. GERASOULIS, *A fast algorithm for the multiplication of generalized Hilbert matrices with vectors*, Math. Comp., 50 (1988), pp. 179–188.

[8] I. GOHBERG, T. KAILATH, AND V. OLSHEVSKY, *Fast Gaussian elimination with partial pivoting for matrices with displacement structure*, Math. Comp., 64 (1995), pp. 1557–1576.

[9] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, third ed., Johns Hopkins University Press, Baltimore, 1996.

[10] C.-H. GUO, *Nonsymmetric algebraic Riccati equations and Wiener-Hopf factorization for M-matrices*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 225–242.

[11] ———, *A note on the minimal nonnegative solution of a nonsymmetric algebraic Riccati equation*, Linear Algebra Appl., 357 (2002), pp. 299–302.

[12] ———, *Efficient methods for solving a nonsymmetric algebraic Riccati equation arising in stochastic fluid models*, J. Comput. Appl. Math., 192 (2006), pp. 353–373.

[13] C.-H. GUO AND N. J. HIGHAM, *A Schur–Newton method for the matrix pth root and its inverse.*, 28 (2008), pp. 788–804.

[14] C.-H. GUO, B. IANNAZZO, AND B. MEINI, *On the doubling algorithm for a (shifted) nonsymmetric algebraic Riccati equation*, SIAM. J. Matrix Anal. Appl., 29 (2007), pp. 1083–1100.

[15] C.-H. GUO AND A. J. LAUB, *On the iterative solution of a class of nonsymmetric algebraic Riccati equations*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 376–391.

[16] X.-X. GUO, W.-W. LIN, AND S.-F. XU, *A structure-preserving doubling algorithm for nonsymmetric algebraic Riccati equation*, Numer. Math., 103 (2006), pp. 393–412.

[17] C. HE, B. MEINI, AND N. H. RHEE, *A shifted cyclic reduction algorithm for quasi-birth-death problems*, SIAM J. Matrix Anal. Appl., 23 (2001/02), pp. 673–691.

[18] G. HEINIG AND K. ROST, *Algebraic Methods for Toeplitz-Like Matrices and Operators*, Operator Theory: Advances and Applications, 13, Birkhäuser, Basel, 1984.

[19] J. JUANG AND W.-W. LIN, *Nonsymmetric algebraic Riccati equations and Hamiltonian-like matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 228–243.

[20] L.-Z. LU, *Newton iterations for a non-symmetric algebraic Riccati equation*, Numer. Linear Algebra Appl., 12 (2005), pp. 191–200.

[21] ———, *Solution form and simple iteration of a nonsymmetric algebraic Riccati equation arising in transport theory*, SIAM J. Matrix Anal. Appl., (2005), pp. 679–685.

[22] V. MEHRMANN AND H. XU, *Explicit solutions for a Riccati equation from transport theory*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1399–1357.

[23] V. RAMASWAMI, *Matrix analytic methods for stochastic fluid flows*, in Proceedings of the 16th International Teletraffic Congress, D. Smith and P. Key, eds., Elsevier, New York, 1999, pp. 1019–1030.