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

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

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 $A_2X^2 + A_1X + A_0X = 0$ 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 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 [17], 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

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

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 [17] and to the references therein). The solution of interest is the minimal positive one, which exists as proved in [17].

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 is n the more accurate is the approximation of X to the solution of the physical 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.

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

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

As shown by Chun-Hua Guo [9], 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.

Even though 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 [18], for numerical reasons it is more suitable to apply *ad hoc* iterative algorithms based on matrix iterations. In fact many algorithms of this kind have been devised in the literature. More recently [21], the closed form of the solution has been exploited for investigating new algorithms.

The main available algorithms for computing the minimal positive solution of this class of algebraic Riccati equations are Newton's method [13], Logarithmic Reduction (LR) [10], Cyclic Reduction (CR) [3] and the Structure-preserving Doubling Algorithm (SDA) [12, 14]. All these algorithms share the same order of complexity, that is, $O(n^3)$ arithmetic operations (ops) per step, and 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 [20] but they have linear convergence which turns to sublinear in critical cases. More recently, in [4] an algorithm implementing the Newton iteration with $O(n^2)$ ops per step has been obtained relying on properties of certain structured matrices.

In this paper we provide two other algorithms of complexity $O(n^2)$ which maintain the quadratic convergence. The first one relies on a reduction provided by Ramaswami in [22] which allows one to express the matrix X in terms of the solution of a unilateral quadratic matrix equation of the kind

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

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 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 $X \to D_1 X - XD_2$ have been introduced and systematically studied by Georg Heinig and Karla Rost in the book [16].

In particular, we provide the explicit Cauchy representations of these sequences and determine the equations which 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 [6].

The second method that we introduce is based on the structured doubling algorithm introduced in [14]. As in the above case, it can be proven 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 which implements the SDA iteration in structured form in $O(n^2)$ operations per step.

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 [8], 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.

Numerical experiments show the effectiveness of our techniques.

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. It is interesting to point out that the analogous generalization of the algorithm of [4] based on Newton's iteration would cost $O(r^2n^2)$ ops per step.

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 5, we show that the shift technique can be used by our algorithm with no increasing of the computational cost. Section 6 deals with an alternative implementation of part of the algorithm in order to overcome numerical problems in critical cases. This section has a 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 *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 $u = \begin{bmatrix} u_1^T \\ u_2^T \end{bmatrix}^T$, $v = \begin{bmatrix} v_1^T \\ v_2^T \end{bmatrix}^T$; the equation is said

- *transient*, if $u_1v_1 - u_2v_2 > 0$;
- *positive recurrent*, if $u_1v_1 - u_2v_2 < 0$;
- *null recurrent*, or *critical*, if $u_1v_1 - u_2v_2 = 0$.

Equation (1.2) 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 [22] (see also [10]) that S is the minimal nonnegative solution of (1.1) if and only if the matrix

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

is the minimal nonnegative solution of the following unilateral equation

$$A_1Y^2 + A_0Y + A_{-1} = 0 \tag{2.1}$$

where

$$A_{-1} = \begin{bmatrix} I - tE & 0 \\ tB & 0 \end{bmatrix}, \quad A_0 = \begin{bmatrix} -I & tC \\ 0 & -I - tA \end{bmatrix}, \quad A_1 = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}, \tag{2.2}$$

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 and its convergence properties [2], [1] for computing the minimal nonnegative solution of (2.1). Let $A_i^{(0)} = A_i$, $i = -1, 0, 1$ and $\widehat{A}^{(0)} = A_0$, and define the sequences

$$\begin{aligned} A_0^{(k+1)} &= A_0^{(k)} - A_{-1}^{(k)} K^{(k)} A_1^{(k)} - A_1^{(k)} K^{(k)} A_{-1}^{(k)}, \quad K^{(k)} = \left(A_0^{(k)}\right)^{-1}, \\ A_{-1}^{(k+1)} &= -A_{-1}^{(k)} K^{(k)} A_{-1}^{(k)}, \quad A_1^{(k+1)} = -A_1^{(k)} K^{(k)} A_1^{(k)}, \\ \widehat{A}_0^{(k+1)} &= \widehat{A}_0^{(k)} - A_1^{(k)} K^{(k)} A_{-1}^{(k)}. \end{aligned} \tag{2.3}$$

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], [1], [10]. In particular, the sequence

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

converges to G. The following result holds [2], [10]

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 [17].

A useful formulation which enables us to perform a structure analysis of the matrix sequences generated by CR is the functional formulation provided in [2].

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 [2]

$$\psi^{(k+1)}(z^2) = \frac{1}{2}(\psi^{(k)}(z) + \psi^{(k)}(-z)). \tag{2.4}$$

2.4. Structured doubling algorithm. The structured doubling algorithm [14] is another algorithm for computing the solution of a nonsymmetric algebraic Riccati equation. The algorithm can be described as follows. Choose $\gamma \geq \max\{e_{i,i}, a_{i,i} : i = 1, \ldots, n\}$; let

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

and

$$\begin{aligned} E_0 &= I - 2\gamma V^{-1}, \\ F_0 &= I - 2\gamma W^{-1}, \\ G_0 &= 2\gamma(E + \gamma I)^{-1} C W^{-1}, \\ H_0 &= 2\gamma W^{-1} B (E + \gamma I)^{-1}. \end{aligned} \tag{2.5}$$

For $k \geq 0$, calculate

$$\begin{aligned} E_{k+1} &= E_k(I - G_k H_k)^{-1} E_k, \\ F_{k+1} &= F_k(I - H_k G_k)^{-1} F_k, \\ G_{k+1} &= G_k + E_k(I - G_k H_k)^{-1} G_k F_k, \\ H_{k+1} &= H_k + F_k(I - H_k G_k)^{-1} H_k E_k. \end{aligned} \tag{2.6}$$

We have the following convergence result [14, 8].

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 *diagonals. 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 $R = \operatorname{diag}(r_1, \ldots, r_n)$ and $S = \operatorname{diag}(s_1, \ldots, s_n)$, with $r_i \neq s_j$ for all i, j, such that

$$\nabla_{R,S}(C) = UV, \tag{2.7}$$

where $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{r\times n}$, 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 $\begin{bmatrix} r_1 & \ldots & r_n \end{bmatrix}^T$ and $\begin{bmatrix} s_1 & \ldots & s_n \end{bmatrix}^T$. We will call a matrix T *Trummer-like*[1] if there is a diagonal matrix $D = \operatorname{diag}(d_1, \ldots, d_n)$, with $d_i \neq d_j$ for all $i \neq j$, such that

$$\nabla_{D,D}(T) = UV, \tag{2.8}$$

where $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{r\times n}$, 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 T is uniquely determined by its generators U, V and the two vectors $\begin{bmatrix} d_1 & \ldots & d_n \end{bmatrix}^T$ and $\begin{bmatrix} T_{11} & \ldots & T_{nn} \end{bmatrix}^T$ (the latter one being the main diagonal of T).

[1] The name comes from the so-called *Trummer problem*, see [5] and the references therein for further details.

Using the relations (2.7) and (2.8) we can easily reconstruct a Cauchy-like or a Trummer-like matrix from its generators with $O(rn^2)$ arithmetical 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 1**.

```
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;
endfunction
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);
  endfor
  y = T * x;
endfunction
```

Algorithm 1: Cauchy-like (Trummer-like) matrix-matrix product

The problem of solving a linear system with Cauchy matrix C was treated by Gohberg, Kailath and Olshevsky in [6]. 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

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

then the Schur complement $\hat{C} = C_{22} - c_{21}c_{11}c_{12}$ solves the displacement equation

$$R_1\hat{C} - \hat{C}S_2 = (U_2 - \frac{1}{c_{11}}c_{21}u_1)(V_2 - \frac{1}{c_{11}}c_{12}v_1). \tag{2.9}$$

Using this fact, one can perform Gaussian elimination on the matrix C in $O(n^2r)$ operations: at each step, instead of computing the entries of the Schur complement of C, one computes its generators as given by (2.9). 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 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. 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.

```
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(1, n); %active column of L of the LU factorization
  x = b;
  for k = 1:n
    % 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) * b(k);
    if(abs(dg(k)) < 1.d-10) warn "Near-to-singular matrix";
    % 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) - v(:,k) * U(k,k+1:n) / dg(k);
    %updates the diagonal
    dg(k+1:n) = dg(k+1:n) - l(k+1:n) * U(k,k+1:n);
  endfor
endfunction
```

Algorithm 2: Solution of a linear system with Trummer-like matrix

3. Structure analysis of the Cyclic Reduction and the main algorithm. In the following, we consider here 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. Note that setting $r = 1, \widetilde{e} = e, \widetilde{q} = q$ yields (1.2).

3.1. Block structure. It is easy to see that performing the cyclic reduction with initial matrices of the form (2.2) not all the blocks of the involved matrices fill up.

THEOREM 3.1. *Let $A_{-1}^{(k)}, A_0^{(k)}, \widehat{A}_0^{(k)}, A_1^{(k)}$, $k \geqslant 0$, be the iterates generated by the CR* (2.3) *with initial matrices* (2.2)*. Then,*

1. *The iterates are of the form*

$$A_{-1}^{(k)} = \begin{bmatrix} * & 0 \\ * & 0 \end{bmatrix}, \; A_0^{(k)} = \begin{bmatrix} -I & * \\ * & * \end{bmatrix}, \; \widehat{A}_0^{(k)} = \begin{bmatrix} -I & tC \\ * & -I - tA \end{bmatrix}, \; A_1^{(k)} = \begin{bmatrix} 0 & 0 \\ 0 & * \end{bmatrix},$$

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

2. The $(2,1)$ *block of* $A_0^{(k)}$ *and* $\widehat{A}_0^{(k)}$ *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 (2.3) for updating $A_0^{(k+1)}$ and $\widehat{A}_0^{(k+1)}$, 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. Applying the reduction (2.1) to the NARE (3.1), for the matrix function $\varphi^{(0)}(z) = A_{-1}z^{-1} + A_0 + A_1 z$ we get

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

Using the Sherman-Morrison formula [7] 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

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

Now, since

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

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

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

yields

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

Using the functional formulation (2.4) of CR and the linearity of $\nabla_{\mathcal{D},\mathcal{D}}$, we can easily prove by induction that

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

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

We can 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* (2.2) *for the problem* (3.1)*. Then,*

$$\begin{aligned}
\nabla_{\mathcal{D},\mathcal{D}}\, A_{-1}^{(k)} =& A_{-1}^{(k)} \begin{bmatrix}\widetilde{q}\\ 0\end{bmatrix} s_0^{(k)} + A_0^{(k)} \begin{bmatrix}0\\ \widetilde{e}\end{bmatrix} t_{-1}^{(k)} + u_0 \begin{bmatrix}e^T & -q^T\end{bmatrix} A_{-1}^{(k)},\\
\nabla_{\mathcal{D},\mathcal{D}}\, A_0^{(k)} =& A_{-1}^{(k)} \begin{bmatrix}\widetilde{q}\\ 0\end{bmatrix} s_1^{(k)} + A_0^{(k)} \begin{bmatrix}\widetilde{q}\\ 0\end{bmatrix} s_0^{(k)} + A_0^{(k)} \begin{bmatrix}0\\ \widetilde{e}\end{bmatrix} t_0^{(k)}\\
&+ A_1^{(k)} \begin{bmatrix}0\\ \widetilde{e}\end{bmatrix} t_{-1}^{(k)} + u_0 \begin{bmatrix}e^T & -q^T\end{bmatrix} A_0^{(k)},\\
\nabla_{\mathcal{D},\mathcal{D}}\, A_1^{(k)} =& A_0^{(k)} \begin{bmatrix}\widetilde{q}\\ 0\end{bmatrix} s_1^{(k)} + A_1^{(k)} \begin{bmatrix}0\\ \widetilde{e}\end{bmatrix} t_0^{(k)} + u_0 \begin{bmatrix}e^T & -q^T\end{bmatrix} A_1^{(k)},
\end{aligned} \tag{3.3}$$

with

$$\begin{aligned}
s_0^{(0)} &= \begin{bmatrix}e^T & 0\end{bmatrix}, & s_0^{(k+1)} &= s_0^{(k)} - s_1^{(k)} K^{(k)} A_{-1}^{(k)},\\
s_1^{(0)} &= \begin{bmatrix}0 & q^T\end{bmatrix}, & s_1^{(k+1)} &= -s_1^{(k)} K^{(k)} A_1^{(k)},\\
t_{-1}^{(0)} &= -\begin{bmatrix}e^T & 0\end{bmatrix}, & t_{-1}^{(k+1)} &= -t_{-1}^{(k)} K^{(k)} A_{-1}^{(k)},\\
t_0^{(0)} &= -\begin{bmatrix}0 & q^T\end{bmatrix}, & t_0^{(k+1)} &= t_0^{(k)} - t_{-1}^{(k)} K^{(k)} A_1^{(k)},\\
u_0 &= -\begin{bmatrix}\widetilde{q}\\ \widetilde{e}\end{bmatrix}.
\end{aligned} \tag{3.4}$$

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 (2.3) 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(A_{-1}^{(k)} \begin{bmatrix}\widetilde{q}\\ 0\end{bmatrix} s_0^{(k)} + A_0^{(k)} \begin{bmatrix}0\\ \widetilde{e}\end{bmatrix} t_{-1}^{(k)} + u_0 \begin{bmatrix} e & -q\end{bmatrix} A_{-1}^{(k)}\right) K^{(k)} A_{-1}^{(k)} +\\
& A_{-1}^{(k)} K^{(k)} \left(A_{-1}^{(k)} \begin{bmatrix}\widetilde{q}\\ 0\end{bmatrix} s_1^{(k)} + A_0^{(k)} \begin{bmatrix}\widetilde{q}\\ 0\end{bmatrix} s_0^{(k)} + A_0^{(k)} \begin{bmatrix}0\\ \widetilde{e}\end{bmatrix} t_0^{(k)} + A_1^{(k)} \begin{bmatrix}0\\ \widetilde{e}\end{bmatrix} t_{-1}^{(k)} +\right.\\
& \left. u_0 \begin{bmatrix} e & -q\end{bmatrix} A_0^{(k)}\right) K^{(k)} A_{-1}^{(k)}\\
& - A_{-1}^{(k)} K^{(k)} \left(A_{-1}^{(k)} \begin{bmatrix}\widetilde{q}\\ 0\end{bmatrix} s_0^{(k)} + A_0^{(k)} \begin{bmatrix}0\\ \widetilde{e}\end{bmatrix} t_{-1}^{(k)} + u_0 \begin{bmatrix} e & -q\end{bmatrix} A_{-1}^{(k)}\right)\\
=& - A_{-1}^{(k)} K^{(k)} A_{-1}^{(k)} \begin{bmatrix}\widetilde{q}\\ 0\end{bmatrix} \left(s_0^{(k)} - s_1^{(k)} K^{(k)} A_{-1}^{(k)}\right) +\\
& \left(A_0^{(k)} - A_{-1}^{(k)} K^{(k)} A_1^{(k)}\right) \begin{bmatrix}0\\ \widetilde{e}\end{bmatrix} \left(-t_{-1}^{(k)} K^{(k)} A_{-1}^{(k)}\right) - u_0 \begin{bmatrix} e & -q\end{bmatrix} A_{-1}^{(k)} K^{(k)} A_{-1}^{(k)}\\
=& A_{-1}^{(k+1)} \begin{bmatrix}\widetilde{q}\\ 0\end{bmatrix} s_0^{(k+1)} + A_0^{(k+1)} \begin{bmatrix}0\\ \widetilde{e}\end{bmatrix} t_{-1}^{(k+1)} + u_0 \begin{bmatrix} e & -q\end{bmatrix} A_{-1}^{(k+1)},
\end{aligned}
$$

Here, we made use of the facts that $A_0^{(k)} K^{(k)} = K^{(k)} A_0^{(k)} = I$, which follows from the definition of $K^{(k)}$, and

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

due the position of the zero blocks in $A_{-1}^{(k)}$ and $A_1^{(k)}$, as proved in Theorem 3.1. □
Note that the displacement equations proven in the above theorem can be derived by imposing

$$
\begin{aligned}
\tilde{s}^{(k)}(z)\varphi^{(k)}(z) &= s_0^{(k)} + s_1^{(k)} z,\\
\tilde{t}^{(k)}(z)\varphi^{(k)}(z) &= t_0^{(k)} + t_{-1}^{(k)} z^{-1},\\
\varphi^{(k)}(z)\tilde{u}^{(k)}(z) &= u_0
\end{aligned}
$$

in equation (3.2).

We can say more about the meaning of $s_0^{(k)}$, $s_1^{(k)}$, $t_0^{(k)}$ and $t_{-1}^{(k)}$.

THEOREM 3.3. *Let $s_0^{(k)}$, $s_1^{(k)}$, $t_0^{(k)}$ and $t_{-1}^{(k)}$ be the sequences defined in* (3.4). *Then, for all $k \geqslant 0$,*

1. $s_0^{(k)} = \begin{bmatrix} -e^T & q^T\end{bmatrix} A_0^{(k)} \begin{bmatrix} I & 0\\ 0 & 0\end{bmatrix}$;
2. $s_1^{(k)} = \begin{bmatrix} -e^T & q^T\end{bmatrix} A_1^{(k)}$;
3. *The (2,2) block of $A_0^{(k)}$ equals $-I - t\Delta - t\widetilde{e}t_0^{(k)}$;*
4. *The (2,1) block of $A_{-1}^{(k)}$ equals $-t\widetilde{e}t_{-1}^{(k)}$.*

All the stated results can be easily proved by induction.

3.3. The main algorithm. The structure relations (3.3) allow us to develop a faster version of the CR iteration, with computational cost $O(n^2 r)$ per step. In fact, Algorithms 1 and 2 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 store the eight generator matrices

$$\begin{aligned} &A_{-1}^{(k)} \begin{bmatrix} \widetilde{q} \\ 0 \end{bmatrix}, \quad A_0^{(k)} \begin{bmatrix} \widetilde{q} \\ 0 \end{bmatrix}, \quad A_0^{(k)} \begin{bmatrix} 0 \\ \widetilde{e} \end{bmatrix}, \quad A_1^{(k)} \begin{bmatrix} 0 \\ \widetilde{e} \end{bmatrix}, \quad t_0^{(k)}, \quad t_{-1}^{(k)}, \\ &\begin{bmatrix} e^T & -q^T \end{bmatrix} A_{-1}^{(k)}, \quad \begin{bmatrix} e^T & -q^T \end{bmatrix} A_0^{(k)}, \quad \begin{bmatrix} e^T & -q^T \end{bmatrix} A_1^{(k)}, \end{aligned} \tag{3.5}$$

(the first four have size $n \times r$, while the last four have size $r \times n$) and update them at each step of the algorithm. Note that $s_0^{(k)}$ and $s_1^{(k)}$ can be recovered from the above generators using the results of theorem 3.3. Since the $(1,1)$ blocks of $A_{-1}^{(k)}$ and the $(2,2)$ block of $A_1^{(k)}$ are Trummer-like matrices, we need to compute their diagonal as well. This can be done noticing that a Trummer-like matrix T can be written as $\bar{D} + \text{Trummer}(\mathcal{D}, U, V)$, where $\bar{D}$ is its diagonal and $\text{Trummer}(\mathcal{D}, U, V)$ 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 $T = A_{-1}^{(k)}$ we have

$$A_{-1}^{(k)} \begin{bmatrix} \widetilde{q} \\ 0 \end{bmatrix} - \text{Trummer}(\mathcal{D}, U, V) \begin{bmatrix} \widetilde{q} \\ 0 \end{bmatrix} = \bar{D} \begin{bmatrix} \widetilde{q} \\ 0 \end{bmatrix},$$

and the whole left-hand side can be computed from the generators. Due to the position of the zero blocks in $A_{-1}^{(k)}$, only the first n entries of $\bar{D}$ are nonzero, and each of them can be computed as

$$\bar{D}_{ii} = \frac{\left(\bar{D} \begin{bmatrix} \widetilde{q} \\ 0 \end{bmatrix} \right)_i}{\widetilde{q}_i}.$$

By applying the same technique to $A_1^{(k)} \begin{bmatrix} 0 \\ \widetilde{e} \end{bmatrix}$ we can compute the $(2,2)$ block of $A_1^{(k)}$ from its generators.

At each step of the iteration we recover the diagonals and $s_0^{(k)}$, $s_1^{(k)}$, then update the eight generator matrices: $t_0^{(k)}$ and $t_{-1}^{(k)}$ are updated using the formulas (3.4), while the other six vectors are updated using the formulas (2.3) directly, e.g.

$$A_{-1}^{(k+1)} \begin{bmatrix} \widetilde{q} \\ 0 \end{bmatrix} = -A_1^{(k)} K^{(k)} \left(A_1^{(k)} \begin{bmatrix} \widetilde{q} \\ 0 \end{bmatrix} \right),$$

and the analogous formulas for the other five. The matrix products involving $K^{(k)}$ are done using Algorithm 2 on $A_0^{(k)}$ (remember that $K^{(k)} = \left(A_0^{(k)} \right)^{-1}$), and the ones involving $A_{-1}^{(k)}$ and $A_1^{(k)}$ are done using Algorithm 1. Since we apply these algorithms to $n \times r$ matrices (or their transposed version to $r \times n$ matrices), the whole computational cost is $O(n^2 r)$. The algorithm is briefly sketched in Algorithm 3.

An obvious choice for the stopping criterion would be to compute the iterate $X^{(k)}$ at each step and explicitly calculating the residual of the Riccati equation (1.1). However, this is quite expensive. Theorem 2.1 provides another good choice. In all

```
function X=fastcr(D,Δ,e,q,ẽ,q̃)
 k=0;
 initialize the generator matrices
   (using (3.4) and (2.3))
 do
  k=k+1;
  calculate the diagonals of A_{-1}^{(k)} and A_1^{(k)}
  recover s_0^{(k)} and s_1^{(k)}
  update the generators
 while(stopping criterion)
  build the generators of Â_0^{(k)}
  calculate X=the (2,1) block of -(Â^{(k)})^{-1} A_{-1}^{(0)}
end function
```

Algorithm 3: Structured cyclic reduction

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 matrices we need are two of the eight 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 slightly accelerated by skipping the computations with the zero blocks, thus reducing all the involved computations to $n \times n$ matrix computations. This way, one sees that we only need to update ten $n \times r$ matrices (instead of eight $2n \times r$) 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

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

Matrices $\mathcal{H}$ and $\mathcal{H}_\gamma$ commute, since the latter is a rational function of the former. This fact can be expressed as

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

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 [14], or also by direct verification from equations (2.5) and (2.6), 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

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

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

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

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

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

which provide low displacement rank representations for the SDA iterates. Using these relations, we go on along the lines of Algorithm 3. At each step, we only store in memory the generators

$$\begin{matrix} E_k\widetilde{q}, & F_k\widetilde{e}, & (H_k\widetilde{q} + \widetilde{e}), & (\widetilde{q} + G_k\widetilde{e}), \\ e^T E_k, & q^T F_k, & (e^T + q^T H_k), & (e^T + q^T G_k), \end{matrix} \tag{4.4}$$

and update them accordingly to (2.6), using the Cauchy-like structure to carry out the computations. 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 diagonal can be recovered using the fact that the both $e^T E_k$ and $F_k\widetilde{e}$ are known, in addition to their Trummer-like generators.

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

```
function X=fastsda(D,Δ,e,q,ẽ,q̃)
 k=0;
 initialize the generator matrices
   (using (2.5) and (4.3))
   and the diagonals of E_0, F_0
 do
  k=k+1;
  update the generators using (4.3)
  calculate the diagonals of E_k and F_k
 while(stopping criterion)
 recover H_k from its generators and return X = H_k
end function
```

Algorithm 4: Structured SDA

5. 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 exposed algorithms is linear instead of quadratic; moreover, it has been shown in [11] that a $O(\varepsilon)$ perturbation to the coefficients of the equation leads to a $O(\sqrt{\varepsilon})$ variation in the solution. All these drawbacks can be removed by means of the shift technique originally introduced be He, Meini and Rhee in [15] and applied to algebraic Riccati equations in [8], [3] and [4].

The shift technique applied to this problem consists in replacing the Riccati equation (3.1) with the equation

$$X\widetilde{C}X - X\widetilde{E} - \widetilde{A}X + \widetilde{B} = 0, \tag{5.1}$$

with

$$\widetilde{A} = A - \eta v_2 p_2^T, \quad \widetilde{B} = B + \eta v_2 p_1^T, \quad \widetilde{C} = C - \eta v_1 p_2^T, \quad \widetilde{E} = E + \eta v_1 p_1^T, \tag{5.2}$$

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), $0 < \eta \leqslant d_1$, 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 $p^T v = 1$. It is proved in [8] that the minimal nonnegative solution of (1.1) is the minimal nonnegative solution of (5.1), and that the latter is noncritical. Therefore, cyclic reduction applied to this problem converges quadratically to the common minimal nonnegative solution X of the two equations.

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

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

More generally, in the case (3.1) one can choose p^T as one of the rows of $\begin{bmatrix} e^T & q^T \end{bmatrix}$. This choice preserves the property that M is a diagonal plus rank r M-matrix, maintaining the problem in the form of equation (3.1), and therefore allows one to use the presented algorithm as in the nonshifted version.

6. Another approach for 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 solution X. This phenomenon, well illustrated in the numerical experiments of the next Section 7, is caused by the large cancellation errors encountered in the computation of the diagonal of X, due to the singularity of the displacement operator $\nabla_{\mathcal{D},\mathcal{D}}$.

In this section we propose a method to overcome 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 $\mathrm{rank}\nabla_{\mathcal{D}_1,\mathcal{D}_2}(A) \leq \mathrm{rank}\nabla_{\mathcal{D},\mathcal{D}}(A) + 1$.

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

$$\xi_i = (\gamma_i + \gamma_{i+1})/2, \quad i = 1, 2, \ldots, 2n-1, \quad \xi_{2n} > \gamma_{2n}. \tag{6.1}$$

Then it is easily verified that the quantities

$$\sigma_i = \frac{\prod_{j=1}^{2n}(\xi_j - \gamma_i)}{\prod_{j=1,\ i\neq i}^{2n}(\gamma_j - \gamma_i)} \tag{6.2}$$

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

$$u_i = \sqrt{\sigma_i}, \tag{6.3}$$

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

$$\theta_j = (\sum_i (u_i/(\gamma_i - \xi_j))^2)^{-1/2}, \tag{6.4}$$

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

$$(D + uu^T)S = SD_1, \quad S = (\theta_i u_j/(\xi - \gamma_j)), \quad D_1 = \mathrm{diag}(\xi_1, \ldots, \xi_{2n}), \tag{6.5}$$

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 (3.5) with new

expressions. However, in this way we would lose the block structure of the vectors involved in (3.5) 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 (3.3), 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

$$\begin{aligned} a_{i,i} =& \sum_j (B)_{i,j}(S)_{i,j} \\ =& \theta_i \sum_\ell \left[v_i^{(\ell)} \sum_j \frac{(w^{(\ell)T}S)_j u_j}{(\gamma_i - \xi_j)(\xi_i - \gamma_j)} \right] + \theta_i (Au)_i \sum_j \frac{(u^T S)_j u_j}{(\gamma_i - \xi_j)(\xi_i - \gamma_j)} \end{aligned} \tag{6.6}$$

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

$$A_i^{(k+1)}u = -A_i^{(k)}(A_0^{(k)})^{-1}A_i^{(k)}u \tag{6.7}$$

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.

The following algorithm synthesize the computation of the diagonal entries of A in the general case.

```
# precomputation step:
choose u
compute:
  ξ_i, i = 1:m by means of (6.1)
  u_i, i = 1:m by means of (6.2) and (6.3)
  θ_i. i = 1:m by means of (6.4)
  u^T S, where S is as defined in (6.5), using Algorithm 1

function d=altdiag(D,v,w,Au)
# output:
# d = diag(A).
# input:
# D, v, w such that
# DA − AD = vw^T
# and Au=A*u
# Au can be computed in any way, e.g. using (6.7)
 compute the vectors w^T S by using Algorithm 1
 for i=1:m
  apply (6.6) and obtain d(i) = a_{i,i}
 end for
end function
```

Algorithm 5: Computation of the diagonal entries of an $m \times m$ Trummer-like matrix A

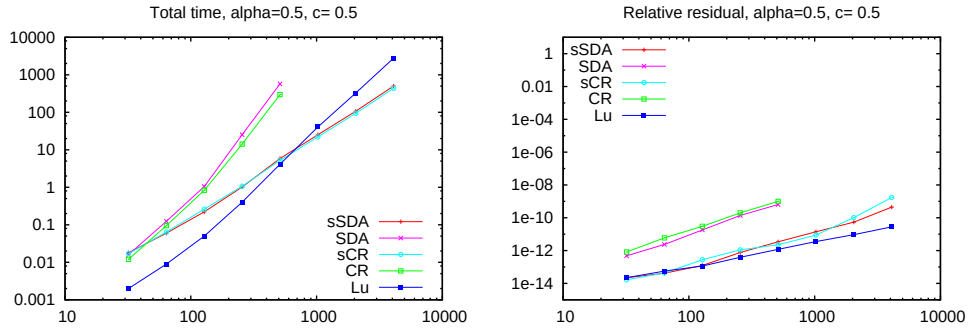

TABLE 7.1
Tests with $\alpha = 0.5, c = 0.5$

7. Numerical experiments. The proposed algorithms have been implemented in Fortran 90 and tested on a 2.8GHz Xeon biprocessor, compiled with Lahey Fortran compiler v. 6.20c. The experiments performed are those mentioned in [19], 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 with the original version of SDA and CR with Ramaswami's reduction, and with the algorithm mentioned in [19], which is a fast Newton-like algorithm specialized for problem (1.2) with cost $2/3n^3 + o(n^3)$ per step and quadratical convergence (in noncritical cases). Our structure-preserving algorithm are labeled `sSDA` and `sCR` in the legend, to distinguish them from the original versions. We report some of the most significant results in Tables 7 and 7 the total time elapsed for the experiments and the residual, calculated as

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

and capped to 1 (since values larger than 1 simply means that no meaningful convergence is reached). Note that the expression appearing at the numerator inside the norm symbols is an alternative way of writing the Riccati equation.

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 overcome Lu's algorithm [19], which is faster for low dimensions but grows 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-to-critical cases. Eventually, for sufficiently high dimension, the convergence is lost. Based on the intermediate results, we believe tha the loss of accuracy is due to the computation of the diagonal entries of the Trummer-like matrices involved, which suffers from cancellation problems. In order to overcome them, we have developed the technique of Section 6. Though changing the involved displacement operator seems the way to overcome the problem, application of Algorithm 5 has provided no significant improvement in the errors. The

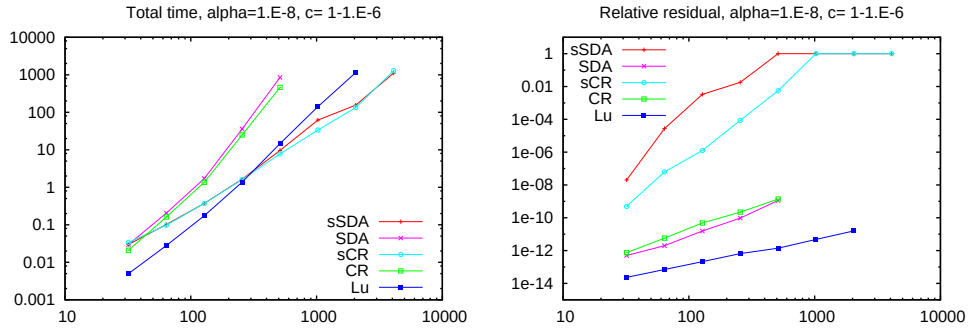

TABLE 7.2
Tests with $\alpha = 10^{-8}, c = 1 - 10^{-6}$

problem of how to reduce the error in near-to-critical cases is still under investigation.

8. 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 structure of the problem, since that is not at all apparent from their definitions.

The presented algorithms provide a new approach for 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 $r \approx 10$–15 or larger. For such equations, the analogous generalization of the structured Lu methods has complexity $O(n^2r^2)$, as can easily be deduced from the derivation in [4], while structured SDA and CR have complexity growing as $O(n^2r)$. Therefore, it is expected that our methods become competitive with structured Lu starting from these values of r.

Turning to numerical stability, some more work is needed to get stable versions of our algorithms for near-to-critical cases. Apparently, the cancellation problems in the calculation of the diagonal of the involved Trummer-like matrices cannot be overcome easily. An alternative method to the direct calculation of the diagonals (which would require $O(n^3)$ ops) and to those presented here (which do not solve the stability issues) is required. Different techniques, alternative to the approach of Section 6, 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 looks simpler to analyze than CR. Moreover, since SDA is faster than Newton-based methods for the general NARE, it would be interesting to see if this holds for our structured equations as well.

REFERENCES

[1] D. A. Bini, L. Gemignani, and B. Meini. Computations with infinite Toeplitz matrices and polynomials. *Linear Algebra Appl.*, 343/344:21–61, 2002. Special issue on structured and infinite systems of linear equations.

[2] D. A. Bini, G. Latouche, and B. Meini. *Numerical methods for structured Markov chains.* Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 2005. , Oxford Science Publications.

[3] Dario A. Bini, Bruno Iannazzo, Guy Latouche, and Beatrice Meini. On the solution of algebraic Riccati equations arising in fluid queues. *Linear Algebra Appl.*, 413(2-3):474–494, 2006.

[4] Dario A. Bini, Bruno Iannazzo, and Federico Poloni. A fast Newton's method for a nonsymmetric algebraic Riccati equation. Technical report, Dipartimento di Matematica, Università di Pisa, Pisa, Italy, 2007. Submitted for publication.

[5] A. Gerasoulis. A fast algorithm for the multiplication of generalized Hilbert matrices with vectors. *Math. Comp.*, 50(181):179–188, 1988.

[6] I. Gohberg, T. Kailath, and V. Olshevsky. Fast Gaussian elimination with partial pivoting for matrices with displacement structure. *Math. Comp.*, 64(212):1557–1576, 1995.

[7] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[8] C.-H. Guo, B. Iannazzo, and B. Meini. On the doubling algorithm for a (shifted) nonsymmetric algebraic Riccati equation. Technical report, Dipartimento di Matematica, Università di Pisa, Italy, May 2005.

[9] Chun-Hua Guo. Nonsymmetric algebraic Riccati equations and Wiener-Hopf factorization for M-matrices. *SIAM J. Matrix Anal. Appl.*, 23(1):225–242 (electronic), 2001.

[10] Chun-Hua Guo. Efficient methods for solving a nonsymmetric algebraic Riccati equation arising in stochastic fluid models. *J. Comput. Appl. Math.*, 192(2):353–373, 2006.

[11] Chun-Hua Guo and Nicholas J. Higham. A Schur–Newton method for the matrix pth root and its inverse. Technical Report 2005.9, Manchester Institute for Mathematical Sciences (MIMS), Manchester, UK, October 2005. To appear in SIAM J. Matrix Anal. Appl.

[12] Chun-Hua Guo, Bruno Iannazzo, and Beatrice Meini. On the doubling algorithm for a (shifted) nonsymmetric algebraic riccati equation. Technical report, Dipartimento di Matematica, Università di Pisa, Pisa, Italy, May 2006. Submitted for publication.

[13] Chun-Hua Guo and Alan J. Laub. On the iterative solution of a class of nonsymmetric algebraic Riccati equations. *SIAM J. Matrix Anal. Appl.*, 22(2):376–391 (electronic), 2000.

[14] Xiao-Xia Guo, Wen-Wei Lin, and Shu-Fang Xu. A structure-preserving doubling algorithm for nonsymmetric algebraic Riccati equation. *Numer. Math.*, 103(3):393–412, 2006.

[15] C. He, B. Meini, and N. H. Rhee. A shifted cyclic reduction algorithm for quasi-birth-death problems. *SIAM J. Matrix Anal. Appl.*, 23(3):673–691 (electronic), 2001/02.

[16] Georg Heinig and Karla Rost. *Algebraic methods for Toeplitz-like matrices and operators*, volume 13 of *Operator Theory: Advances and Applications*. Birkhäuser Verlag, Basel, 1984.

[17] Jonq Juang and Wen-Wei Lin. Nonsymmetric algebraic Riccati equations and Hamiltonian-like matrices. *SIAM J. Matrix Anal. Appl.*, 20(1):228–243 (electronic), 1999.

[18] Jonq Juang and Wen-Wei Lin. Nonsymmetric algebraic Riccati equations and Hamiltonian-like matrices. *SIAM J. Matrix Anal. Appl.*, 20(1):228–243 (electronic), 1999.

[19] Lin-Zhang Lu. Newton iterations for a non-symmetric algebraic Riccati equation. *Numer. Linear Algebra Appl.*, 12(2-3):191–200, 2005.

[20] Lin-Zhang Lu. Solution form and simple iteration of a nonsymmetric algebraic Riccati equation arising in transport theory. *SIAM J. Matrix Anal. Appl.*, 26(3):679–685 (electronic), 2005.

[21] V. Mehrmann and H. Xu. Explicit solutions for a Riccati equation from transport theory. Technical report, Department of Mathematics, Kansas University, 2008.

[22] V. Ramaswami. Matrix analytic methods for stochastic fluid flows. In *Proceedings of the 16th International Teletraffic Congress*, pages 19–30. Elsevier Science, Edinburg, 1999.