A FAST NEWTON'S METHOD FOR A NONSYMMETRIC ALGEBRAIC RICCATI EQUATION[*]

DARIO A. BINI, BRUNO IANNAZZO[†] AND FEDERICO POLONI[‡]

Abstract. A special instance of the algebraic Riccati equation $XCX - XE - AX + B = 0$ where the $n \times n$ matrix coefficients A, B, C, E are rank structured matrices is considered. Relying on the structural properties of Cauchy-like matrices, an algorithm is designed for performing the customary Newton iteration in $O(n^2)$ arithmetic operations (ops). The same technique is used to reduce the cost of the algorithm proposed by L.-Z. Lu in [*Numer. Linear Algebra Appl.*, 12 (2005), pp. 191–200] from $O(n^3)$ to $O(n^2)$ ops still preserving quadratic convergence in the generic case. As a byproduct we show that the latter algorithm is closely related to the customary Newton method by simple formal relations.

In critical cases where the Jacobian at the required solution is singular and quadratic convergence turns to linear, we provide an adaptation of the shift technique in order to get rid of the singularity. The original equation is transformed into an equivalent Riccati equation where the singularity is removed while the matrix coefficients maintain the same structure as in the original equation. This leads to a quadratically convergent algorithm with complexity $O(n^2)$ which provides approximations with full precision.

Numerical experiments and comparisons which confirm the effectiveness of the new approach are reported.

Key words. nonsymmetric algebraic Riccati equation, Newton's iteration, Cauchy matrix, matrix equation, fast algorithm, M-matrix

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

1. Introduction.

Consider the following nonsymmetric algebraic Riccati equation (NARE) arising in transport theory:

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

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

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

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 $c_i > 0, i = 1, 2, \dots, n$, such that $\sum_i c_i = 1$. For more details and for the physical meaning of these parameters, we refer the reader to [14] and to the references therein. The solution of interest is the minimal positive one, which exists as proved by J. Juang and W.-W. Lin in [14].

It was shown by C.-H. Guo [6] that this equation falls in the class of nonsymmetric algebraic Riccati equations associated with a nonsingular M-matrix or a singular

[*]This work was in part supported by MIUR Grant 2006017542

[†]Dipartimento di Matematica, Università di Pisa, Largo B. Pontecorvo 5, 56127 Pisa, Italy ({`bini, iannazzo`} `@mail.dm.unipi.it`).

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

irreducible M-matrix, in fact arranging the coefficients as

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

yields an M-matrix.

For this class of algebraic Riccati equations, several suitable algorithms exist for computing the minimal positive solution: the Newton method [11], the Logarithmic and Cyclic Reduction [2, 7] and the Structure-preserving Doubling Algorithm [10, 12]. All these algorithms share the same order of complexity, that is, $O(n^3)$ arithmetic operations (ops) per step, and provide quadratic convergence.

Observing that the equation (1.1) is defined by a linear number of parameters, it is quite natural to aim to design algorithms which exploit the structure of the matrices and thus have a cost of order lower than $O(n^3)$ ops.

A step in this direction has been done by L.-Z. Lu [16] who has designed a vector iteration whose limit allows one to easily recover the solution. The iteration has a computational cost of $O(n^2)$ ops per step and converges linearly for $\alpha \neq 0$ or $c \neq 1$. The linear convergence is a drawback since the algorithm in many cases needs a large number of iterations to *converge* and it is outperformed by algorithms with quadratic convergence and $O(n^3)$ ops. In fact, the same author in [15] proposes a mixed algorithm to speed up the computation. The algorithm starts with the linear iteration of complexity $O(n^2)$ and switches to a quadratically convergent one, of complexity $O(n^3)$, when some conditions are satisfied.

In this paper we consider the customary Newton method applied to (1.1). By exploiting the rank structure of the matrix coefficients, we design an algorithm for performing the Newton step in $O(n^2)$ ops. The new approach relies on a suitable modification of the fast LU factorization algorithm for Cauchy-like matrices proposed by I. Gohberg, T. Kailath and V. Olshevsky in [4].

The same idea is applied to implement the quadratically convergent iteration of L.-Z. Lu [15] by an algorithm with cost $O(n^2)$ ops. We also provide formal relations between the sequences generated by Lu's and Newton's iteration which enable one to deduce the convergence of Lu's algorithm directly from the well-known properties of Newton's method.

In the critical but still important case where the Jacobian at the solution is singular, the convergence of Newton's (and therefore Lu's) iteration turns to linear; also the mixed iteration proposed in [15] loses its quadratic convergence while the iteration of [16] converges sublinearly.

In this case, which is encountered when $\alpha = 0$, $c = 1$, we can get rid of the singularity of the Jacobian and consequently of all the above mentioned drawbacks. The idea is to apply the shift technique originally introduced by C. He, B. Meini and N.H. Rhee [13] and used in the framework of Riccati equations by C.-H. Guo, B. Iannazzo, and B. Meini in [9] and by C.-H. Guo [7]. With this technique, we replace the original Riccati equation with a new one having the same minimal solution as the original equation (1.1) but where the singularity of the Jacobian is removed. We prove that the matrix coefficients of the new equation share the same rank structure properties of the coefficients of (1.1). This enables us to design a fast Newton iteration which preserves the quadratic convergence and keeps the same $O(n^2)$ complexity even in the critical case.

As a byproduct of this analysis, we find that the approximation to the minimal solution of (1.1) that we compute from the "shifted" equation is much more accurate than the one obtained by applying the algorithm to the original equation. More precisely, it has been shown by C.-H. Guo and N. Higham [8] that in order to achieve high accuracy it is necessary to use the singularity of M in the design of algorithms, otherwise, we can only expect to achieve an accuracy of $O(\sqrt{\varepsilon})$, where ε is the machine precision. With the use of the shift technique [9], the information on the singularity of M is plugged into the algorithm and we may achieve full accuracy in the approximation as confirmed by the numerical experiments.

The paper is organized as follows. After some preliminaries presented in Section 2, we show in Section 3 how to reduce one step of Newton's iteration for (1.1) to the solution of a linear system with a structured matrix and in Section 4 we deal with the problem of solving such a system in $O(n^2)$ ops. In Section 5 we show that the iteration proposed by L.-Z. Lu [15] shares the same displacement structure and thus its complexity can be reduced to $O(n^2)$ as well, and we exploit the connection between it and the Newton iteration. In Section 6 we deal with the critical case where the Jacobian is singular, by using the shift technique. In Section 7 we address the main numerical stability issues, and in Section 8 we present some numerical examples. From the experiments performed so far, our method turns out to be much faster and accurate than the existing methods. Finally, in Section 9, we discuss some possible generalizations of this algorithm together with some research lines.

2. Preliminaries.

A basic tool which we use is the concept of Cauchy-like matrix [4]. A matrix $C = (c_{ij})_{i,j=1,\dots,n}$ is called *Cauchy-like* if its elements are of the form $c_{ij} = \frac{u_i v_j}{r_i - s_j}$ for some constants u_i, v_i, r_i, s_i, $i = 1, \dots, n$ such that $r_i \neq s_j$ for each i, j. If we define $R = \operatorname{diag}(r_1, r_2, \dots, r_n)$ and $S = \operatorname{diag}(s_1, s_2, \dots, s_n)$, we have $RC - CS = uv^T$, where $u = [u_1, u_2, \dots, u_n]^T$ and $v = [v_1, v_2, \dots, v_n]^T$. The operator $C \mapsto RC - CS$ is called *displacement operator*, and u, v^T are called the *generators* of C. Generalizing, if there exist two diagonal matrices R, S such that $RC - CS$ has rank r, we say that C has *displacement rank* r. When r is small with respect to the size of C, C is called a *generalized Cauchy-like matrix* with respect to the pair (R, S).

Note that, using (1.2), the equation (1.1) can be rewritten as

$$XD + \Delta X = (Xq + e)(e^T + q^T X);$$

therefore any solution X is Cauchy-like with respect to $(\Delta, -D)$ and its generators are $u = Xq + e$ and $v^T = e^T + q^T X$.

We will also need some basic facts on M-matrices. A matrix $A = (a_{i,j}) \in \mathbb{R}^{n \times n}$ is called a *Z-matrix* if $a_{ij} \leqslant 0$ for all $i \neq j$. A Z-matrix A is called an *M-matrix* if there exists a nonnegative matrix B with spectral radius $\rho(B) = r$ such that $A = cI_n - B$ and $r \leqslant c$, where I_n is the identity matrix of order n.

The following results are well-known and can be found in [1].

Lemma 2.1. *For a Z-matrix A it holds that:*

1. *A is an M-matrix if and only if there exists a nonzero vector $v \geqslant 0$ such that $Av \geqslant 0$ or a nonzero vector $w \geqslant 0$ such that $w^T A \geqslant 0$;*
2. *if A is nonsingular then A is an M-matrix if and only if $A^{-1} \geqslant 0$.*

Lemma 2.2. *Let A be a nonsingular M-matrix, then the Schur complement of any principal submatrix of A is a nonsingular M-matrix.*

Here and hereafter, inequalities on matrices and vectors are used in the componentwise sense.

Another useful tool is the Sherman–Morrison–Woodbury (SMW) matrix identity [5, p. 50].

Lemma 2.3 (Sherman–Morrison–Woodbury formula). *Let $D \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{k\times k}$ be nonsingular, and $U \in \mathbb{R}^{n\times k}$, $V \in \mathbb{R}^{k\times n}$. Then $D - UCV$ is nonsingular if and only if $C^{-1} - VD^{-1}U$ is nonsingular, and it holds that*

$$(D - UCV)^{-1} = D^{-1} + D^{-1}U(C^{-1} - VD^{-1}U)^{-1}VD^{-1}.$$

The following lemma relates the SMW formula and M-matrices.

Lemma 2.4. *Let D, C, U, V be real matrices satisfying the hypotheses of Lemma 2.3, with D and C diagonal and $D, C, U, V \geqslant 0$. Then, $D - UCV$ is a (nonsingular) M-matrix if and only if $C^{-1} - VD^{-1}U$ is a (nonsingular) M-matrix.*

Proof. Let $C^{-1} - VD^{-1}U$ be a nonsingular M-matrix, the SMW formula yields

$$(D - UCV)^{-1} = D^{-1} + D^{-1}U(C^{-1} - VD^{-1}U)^{-1}VD^{-1},$$

and since all terms on the right-hand side are nonnegative, one has $(D - UCV)^{-1} > 0$, and using Lemma 2.1 one achieves the proof; the converse is analogous. By a continuity argument, the result can be extended to singular M-matrices. □

3. Newton's method.

Newton's iteration applied to (1.1), for a suitable initial value $X^{(0)}$, generates the matrix sequence $\{X^{(k)}\}$ defined by the solution of the Sylvester equation [11]

$$(X^{(k+1)} - X^{(k)})(E - CX^{(k)}) + (A - X^{(k)}C)(X^{(k+1)} - X^{(k)}) = \mathcal{R}(X^{(k)}), \qquad (3.1)$$

where $\mathcal{R}(X) = XCX - XE - AX + B$ is the Riccati operator. Using the Kronecker product notation, this can be written as

$$\operatorname{vec} X^{(k+1)} - \operatorname{vec} X^{(k)} = \left((E - CX^{(k)})^T \otimes I_n + I_n \otimes (A - X^{(k)}C)\right)^{-1} \operatorname{vec} \mathcal{R}(X^{(k)}) \quad (3.2)$$

where the vec operator stacks the columns of a matrix one above the other to form a single vector. Thus Newton's iteration is well-defined when the matrix $M_{X^{(k)}} = (E - CX^{(k)})^T \otimes I_n + I_n \otimes (A - X^{(k)}C)$ is nonsingular for each k. With abuse of notation, we call the matrix M_X *the Jacobian matrix* at X, in fact it is the Jacobian matrix of the vector function $-\operatorname{vec} \circ R \circ \operatorname{vec}^{-1}$ at $\operatorname{vec}(X)$.

The following result, proved in [8] by C.-W. Guo and N. Higham, provides sufficient conditions for the convergence of the Newton method.

Theorem 3.1. *Let*

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

be a nonsingular M-matrix or a irreducible singular M-matrix and $X^{(0)} = 0$. Then Newton's iteration (3.1) *is well-defined and the sequence $\{X^{(k)}\}$ converges monotonically to the minimal positive solution of the NARE* (1.1). *Moreover, the Jacobian matrix $M_{X^{(k)}} = (E - CX^{(k)})^T \otimes I + I \otimes (A - X^{(k)}C) \in \mathbb{R}^{n^2\times n^2}$ is a nonsingular M-matrix for all $k \geqslant 0$.*

Note that the problem stated in equation (1.2) satisfies the hypotheses of the previous theorem: in fact we have

$$M = \begin{bmatrix} D & 0 \\ 0 & \Delta \end{bmatrix} - \begin{bmatrix} q \\ e \end{bmatrix} \begin{bmatrix} e^T & q^T \end{bmatrix},$$

and by Lemma 2.4 M is an M-matrix if and only if

$$0 \leqslant 1 - \begin{bmatrix} e^T & q^T \end{bmatrix} \begin{bmatrix} D^{-1} & 0 \\ 0 & \Delta^{-1} \end{bmatrix} \begin{bmatrix} q \\ e \end{bmatrix},$$

which reduces to

$$1 \geqslant e^T D^{-1} q + q^T \Delta^{-1} e = \sum_{i=1}^{n} \frac{c(1-\alpha)}{2} c_i + \sum_{i=1}^{n} \frac{c(1+\alpha)}{2} c_i = c, \tag{3.3}$$

in view of (1.3). This fact was observed also in [6].

In the following, we will consider a slightly more general case; i.e., when the matrix M is a generic diagonal plus rank-one matrix. Hence, equation (1.2) becomes

$$A = \Delta - \widetilde{e} q^T, \quad B = \widetilde{e} e^T, \quad C = \widetilde{q} q^T, \quad E = D - \widetilde{q} e^T, \tag{3.4}$$

where e, q, $\widetilde{e}$, $\widetilde{q}$ are any nonnegative vectors such that M, as defined in (3.1), is a nonsingular M-matrix or a singular irreducible M-matrix. Such generalization will prove useful when dealing with the critical case.

Observe that Newton's iteration for the coefficients of (1.1) defined in (3.4) can be rewritten as

$$\begin{aligned} X^{(k+1)} D + \Delta X^{(k+1)} &= (X^{(k)} \widetilde{q} - X^{(k+1)} \widetilde{q})(q^T X^{(k)} - q^T X^{(k+1)}) \\ &\quad - (X^{(k+1)} \widetilde{q} + \widetilde{e})(e^T + q^T X^{(k+1)}), \end{aligned} \tag{3.5}$$

i.e., $X^{(k+1)}$ is a generalized Cauchy-like matrix with displacement rank 2. This property holds for all the iterates $X^{(k)}$, $k \geqslant 1$ of Newton's method obtained with any starting matrix $X^{(0)}$.

The Jacobian at $X^{(k)}$, in Kronecker product notation, takes the form

$$M_{X^{(k)}} = D^T \otimes I_n + I_n \otimes \Delta - (e + X^{(k)T} q) \widetilde{q}^T \otimes I_n - I_n \otimes (\widetilde{e} + X^{(k)} \widetilde{q}) q^T.$$

By setting $\mathcal{D} = D^T \otimes I_n + I_n \otimes \Delta$, $u^{(k)} = \widetilde{e} + X^{(k)} \widetilde{q}$, $v^{(k)} = e + X^{(k)T} q$, and

$$U^{(k)} = \begin{bmatrix} v^{(k)} \otimes I_n & I_n \otimes u^{(k)} \end{bmatrix}, \quad V = \begin{bmatrix} \widetilde{q}^T \otimes I_n \\ I_n \otimes q^T \end{bmatrix}, \tag{3.6}$$

we can rewrite $M_{X^{(k)}}$ as

$$M_{X^{(k)}} = \mathcal{D} - U^{(k)} V. \tag{3.7}$$

Since $U^{(k)} \in \mathbb{R}^{n^2 \times 2n}$ and $V \in \mathbb{R}^{2n \times n^2}$, the inversion of $M_{X^{(k)}}$ can be reduced to the inversion of a $2n \times 2n$ matrix using the SMW formula:

$$M_{X^{(k)}}^{-1} = \mathcal{D}^{-1} + \mathcal{D}^{-1} U^{(k)} (I_{2n} - V \mathcal{D}^{-1} U^{(k)})^{-1} V \mathcal{D}^{-1}. \tag{3.8}$$

This provides a new algorithm for implementing the Newton step, denoted by Algorithm 1, which relies on the function `fast_solve` for the fast solution of the system

$$\begin{aligned} R^{(k)} x &= b \\ R^{(k)} &= I_{2n} - V D^{-1} U^{(k)} \end{aligned} \tag{3.9}$$

```
function $X^{(k+1)}$=NewtonStep($X^{(k)}$)
  $u^{(k)} = \widetilde{e} + X^{(k)} * \widetilde{q}$;
  $v^{(k)} = e + X^{(k)T} * q$;
  $\mathcal{R}(X^{(k)}) = u^{(k)} * v^{(k)T} - XD - \Delta X$;
  $R_1 = [\widetilde{q}^T \otimes I_n ; I_n \otimes q^T] * (\mathcal{D}^{-1} * \mathrm{vec}(\mathcal{R}(X^{(k)})))$;
  $R_2 =$ fast_solve($(I_{2n} - V\mathcal{D}^{-1}U^{(k)})R_2 = R_1$);
  $X^{(k+1)} = \mathcal{D}^{-1}(\mathrm{vec}(\mathcal{R}(X^{(k)})) + [v^{(k)} \otimes I_n \;\; I_n \otimes u^{(k)}] * R_2)$;
  return $X^{(k+1)}$
end function
```

Algorithm 1: Fast Newton's Step

in $O(n^2)$ ops. The function `fast_solve` is described in the next section.

Note that since $\mathcal{D}$ is a diagonal matrix of size $n^2 \times n^2$, the matrix–vector product with matrix $\mathcal{D}^{-1}$ costs $O(n^2)$ ops, and the identities

$$(v^T \otimes I_n)\,\mathrm{vec}(M) = Mv,$$
$$(I_n \otimes v^T)\,\mathrm{vec}(M) = M^T v.$$

allow one to compute the remaining products in $O(n^2)$ as well. Therefore the overall cost of Algorithm 1 is $O(n^2)$.

4. Fast Gaussian elimination for Cauchy-like matrices.

We now address the problem of solving the linear system (3.9) given the vector b and the vectors $q, u^{(k)}, v^{(k)}$ such that

$$R^{(k)} = I_{2n} - \begin{bmatrix} \widetilde{q}^T \otimes I_n \\ I_n \otimes q^T \end{bmatrix} \mathcal{D}^{-1} \begin{bmatrix} v^{(k)} \otimes I_n & I_n \otimes u^{(k)} \end{bmatrix}. \tag{4.1}$$

First note that under the hypotheses of Theorem 3.1 $R^{(k)}$ is a nonsingular M-matrix by Lemma 2.4 applied to the nonsingular M-matrix $M_{X^{(k)}}$ of (3.7). Carrying out the products in (4.1) yields

$$R^{(k)} = I_{2n} - \begin{bmatrix} G^{(k)} & H^{(k)} \\ K^{(k)} & L^{(k)} \end{bmatrix} \tag{4.2}$$

with

$$\begin{array}{ll} G^{(k)} = \mathrm{diag}(g_i^{(k)}), & g_i^{(k)} = \sum_{l=1}^n \frac{v_l^{(k)}\widetilde{q}_l}{d_l+\delta_i}, \\ H^{(k)} = (h_{ij}^{(k)}), & h_{ij}^{(k)} = \frac{u_i^{(k)}\widetilde{q}_j}{d_j+\delta_i}, \\ K^{(k)} = (\kappa_{ij}^{(k)}), & \kappa_{ij}^{(k)} = \frac{v_i^{(k)}q_j}{d_i+\delta_j}, \\ L^{(k)} = \mathrm{diag}(l_i^{(k)}), & l_i^{(k)} = \sum_{l=1}^n \frac{u_l^{(k)}q_l}{d_i+\delta_l}. \end{array} \tag{4.3}$$

Thus $G^{(k)}$ and $L^{(k)}$ are diagonal, and $H^{(k)}$ and $K^{(k)}$ are Cauchy-like. Their displacement equations are

$$\Delta H^{(k)} + H^{(k)}D = u^{(k)}\widetilde{q}^T, \quad DK^{(k)} + K^{(k)}\Delta = v^{(k)}q^T.$$

Partition x and b according to the block structure of $R^{(k)}$ as $x = [x_1^T, x_2^T]^T$, $b = [b_1^T, b_2^T]^T$. Performing the block LU factorization of $R^{(k)}$ enables one to rewrite the system $R^{(k)}x = b$ as

$$\begin{bmatrix} I - G^{(k)} & H^{(k)} \\ 0 & S^{(k)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ \widehat{b}_2 \end{bmatrix} \tag{4.4}$$

where $S^{(k)} = I - L^{(k)} - K^{(k)}(I - G^{(k)})^{-1}H^{(k)}$ and $\widehat{b}_2 = b_2 - K^{(k)}(I - G^{(k)})^{-1}b_1$. The matrices $I - G^{(k)}$ and $S^{(k)}$ are nonsingular as they are a principal submatrix and the Schur complement of a nonsingular M-matrix, respectively. Moreover, $S^{(k)}$ enjoys the following displacement structure

$$DS^{(k)} - S^{(k)}D = K^{(k)}(I - G^{(k)})^{-1}u^{(k)}\widetilde{q}^T - v^{(k)}q^T(I - G^{(k)})^{-1}H^{(k)}.$$

This can be easily proved since D, Δ, $I - G^{(k)}$, $I - L^{(k)}$ all commute because they are diagonal, in fact,

$$\begin{aligned} DS^{(k)} &= D(I - L^{(k)}) - DK^{(k)}(I - G^{(k)})^{-1}H^{(k)} \\ &= (I - L^{(k)})D + (K^{(k)}\Delta - vq^T)(I - G^{(k)})^{-1}H^{(k)} \\ &= (I - L^{(k)})D + K^{(k)}(I - G^{(k)})^{-1}\Delta H^{(k)} - v^{(k)}q^T(I - G^{(k)})^{-1}H^{(k)} \\ &= (I - L^{(k)})D - K^{(k)}(I - G^{(k)})^{-1}(H^{(k)}D - u^{(k)}\widetilde{q}^T) - v^{(k)}q^T(I - G^{(k)})^{-1}H^{(k)} \\ &= S^{(k)}D + K^{(k)}(I - G^{(k)})^{-1}u^{(k)}\widetilde{q}^T - v^{(k)}q^T(I - G^{(k)})^{-1}H^{(k)}. \end{aligned}$$

Thus $S^{(k)}$ is a generalized Cauchy-like matrix with displacement rank 2 with respect to the singular operator $DS^{(k)} - S^{(k)}D$. We can use this property to develop an *ad-hoc* variation of the Gohberg–Kailath–Olshevsky (GKO) algorithm for the fast LU factorization of matrices with displacement structure [4]. The GKO algorithm, for a generalized Cauchy-like matrix S with generators M_1 and N_1, essentially goes on as follows (ignoring pivoting for the sake of simplicity):

1. From the generators M_1, N_1 of

$$S = \begin{bmatrix} d_1 & u_1 \\ l_1 & S_2 \end{bmatrix},$$

such that $DS - SD - M_1^T N_1$, recover the first row and the first column of S and store them as the first column of L

$$\begin{bmatrix} 1 \\ \frac{l_1}{d_1} \end{bmatrix}$$

and the first row of U

$$\begin{bmatrix} d_1 & u_1 \end{bmatrix}$$

in the LU factorization of S;

2. Compute the generators M_2, N_2 of the Schur complement $S_2 - \frac{l_1 u_1}{d_1}$ as

$$M_2 = M_{12} - \frac{l_1}{d_1}m_{11}, \quad N_2 = N_{12} - n_{11}\frac{u_1}{d_1};$$

where

$$M_1 = \begin{bmatrix} m_{11} \\ M_{12} \end{bmatrix}, \quad N_1 = \begin{bmatrix} n_{11} & N_{12} \end{bmatrix}.$$

3. Apply the algorithm recursively to compute the LU factorization L_2U_2 of the Schur complement $S_2 - \frac{l_1 u_1}{d_1}$; then reconstruct the factors

$$L = \begin{bmatrix} 1 & \frac{l_1}{d_1} \\ 0 & L_2 \end{bmatrix}, \quad U = \begin{bmatrix} d_1 & u_1 \\ 0 & U_2 \end{bmatrix}.$$

The problem in our context is that d_1 cannot be retrieved from the generators, due to the singularity of the operator $S \mapsto DS - SD$. In fact, it is easy to see that the null space of $DS - SD$ is the set of all diagonal matrices. Thus, we need a different method to compute and update the diagonal elements of S through the LU factorization. Our approach consists in storing the main diagonal of S in a vector s, and updating it at each step of the Gaussian elimination as if we were performing a customary (non structured) Gaussian elimination. This can be achieved at the general step k by using the relation $S_{ii} \leftarrow S_{ii} - L_{ik}U_{ki}$. Since we only have to update n elements at each step, the overhead of updating the diagonal is $O(n^2)$, and thus the complete algorithm requires $O(n^2)$ ops. A simple implementation, which includes partial pivoting, is given in Algorithm 2 and requires $10n^2$ ops.

```
function [PL,U]=fastPLU(d,s,M,N)
% computes S=PLU, for S such that diag(d)*S-S*diag(d) = M*N
% and $S_{ii} = s_i$
% u keeps track of the rows already taken as pivots
u=[1,1,...,1]';L=U=zeros(n,n);
for k=1:n
  $L_{ik}=(\sum_j M_{ij}N_{jk})/(d_i-d_k)$ for all $i \neq k$ such that $u_i = 1$;
  $L_{kk}=s_k$ if $u_i = 1$;
  choose $p$ such that $|L_{pk}| = \max_i |L_{ik}|$;
  $u_p=0$;
  $U_{kk}=L_{pk}$;
  $L_{ik}=L_{ik}/U_{kk}$ for all $i$ such that $u_i = 1$
  $U_{kj}=(\sum_i M_{pi}N_{ij})/(d_p-d_j)$ for all $j = k+1,\dots,n$, $j \neq p$;
  $U_{kp}=s_p$;
  $M_{ij}=M_{ij} - L_{ik}M_{pj}$ for all $j$, $i$ such that $u_i = 1$;
  $N_{ij}=N_{ij} - N_{ik}U_{kj}/U_{kk}$ for all $i = 1,\dots,n$, $j = k+1,\dots,n$;
  $s_i=s_i - L_{ik}U_{ki}$ for all $i$ such that $u_i = 1$;
  return L,U;
end function
```

Algorithm 2: Fast LU factorization

Using this algorithm, complemented with back-substitution, provides an implementation of the function `fast_solve`$(R^{(k)}x = b)$ used in Algorithm 1 of section 3 of complexity $O(n^2)$.

5. Lu's iteration. L.-Z. Lu [15] proposed a different approach for solving the Riccati equation (1.1) when the coefficients are in the form (3.4). The idea is applying Newton's iteration to an equation involving the displacement generators $u = X\widetilde{q} + \widetilde{e}$ and $v = X^T q + e$ of the solution X. His algorithm can be expressed as the following iteration for the sequences $\{\widehat{u}^{(k)}\}$, $\{\widehat{v}^{(k)}\}$, $k \geqslant -1$:

$$\begin{bmatrix} \widehat{u}^{(k+1)} \\ \widehat{v}^{(k+1)} \end{bmatrix} = (\widehat{R}^{(k)})^{-1} \begin{bmatrix} \widetilde{e} - \widehat{H}^{(k)}\widehat{v}^{(k)} \\ e - \widehat{K}^{(k)}\widehat{u}^{(k)} \end{bmatrix}, \tag{5.1}$$

starting from $\widehat{u}^{(-1)} = \widehat{v}^{(-1)} = 0$ (as we will see later on, indexing from $k=-1$ will simplify the subsequent analysis). Here $\widehat{R}^{(k)}$, $\widehat{H}^{(k)}$ and $\widehat{K}^{(k)}$ are defined as

$$\widehat{R}^{(k)} = I_{2n} - \begin{bmatrix} \widehat{G}^{(k)} & \widehat{H}^{(k)} \\ \widehat{K}^{(k)} & \widehat{L}^{(k)} \end{bmatrix}, \tag{5.2}$$

$$\begin{aligned} \widehat{G}^{(k)} &= \operatorname{diag}(\widehat{g}_i^{(k)}), & \widehat{g}_i^{(k)} &= \textstyle\sum_{l=1}^n \frac{\widehat{v}_l^{(k)}\widetilde{q}_l}{d_l+\delta_i}, \\ \widehat{H}^{(k)} &= (\widehat{h}_{ij}^{(k)}), & \widehat{h}_{ij}^{(k)} &= \frac{\widehat{u}_i^{(k)}\widetilde{q}_j}{d_j+\delta_i}, \\ \widehat{K}^{(k)} &= (\widehat{\kappa}_{ij}^{(k)}), & \widehat{\kappa}_{ij}^{(k)} &= \frac{\widehat{v}_i^{(k)}q_j}{d_i+\delta_j}, \\ \widehat{L}^{(k)} &= \operatorname{diag}(\widehat{l}_i^{(k)}), & \widehat{l}_i^{(k)} &= \textstyle\sum_{l=1}^n \frac{\widehat{u}_l^{(k)}q_l}{d_i+\delta_l}, \end{aligned} \tag{5.3}$$

which are, formally, the same relations as in (4.2) and (4.3).

As a first result, since both algorithms are based on the solution of a system with the same structure, we obtain that Algorithm 2 can also be used in the implementation of Lu's iteration to reduce its computational cost to $O(n^2)$. But there is a deeper connection between the two algorithms.

THEOREM 5.1. *Let $\{\widehat{u}^{(k)}\}$, $\{\widehat{v}^{(k)}\}$, $k \geqslant -1$ be the sequences generated by Lu's algorithm for the NARE (3.4), and let $\{X^{(k)}\}$, $k \geqslant 0$ be the sequence generated by Newton's iteration with starting point $X^{(0)} = 0$ for the same problem. Then, for all $k \geqslant 0$, one has*

$$\widehat{u}^{(k)} = X^{(k)}\widetilde{q} + \widetilde{e}, \quad \widehat{v}^{(k)} = X^{(k)T}q + e.$$

Proof. We will prove the result by induction over k. It is easy to check from the definitions that $\widehat{R}^{(-1)} = I_{2n}$, and thus $\widehat{u}^{(0)} = \widetilde{e}$, $\widehat{v}^{(0)} = e$; therefore the base step $k=0$ holds. As a side note, this means that we can save an iteration by starting the computation from $u^{(0)} = \widetilde{e}$, $v^{(0)} = e$.

Assuming by induction that $\widehat{u}^{(k)} = X^{(k)}\widetilde{q} + \widetilde{e} = u^{(k)}$, $\widehat{v}^{(k)} = X^{(k)T}q + e = v^{(k)}$, we find that equations (4.3) and (5.2) define the same matrices; therefore, from now on, we will drop the superscript (k) and the hat symbol to ease the notation.

We have

$$V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X) = V\mathcal{D}^{-1}\operatorname{vec}(uv^T) - V\operatorname{vec}X = \begin{bmatrix} \widetilde{e} - (I-G)u \\ e - (I-L)v \end{bmatrix},$$

in view of the relations

$$\begin{aligned} \mathcal{D}^{-1}\operatorname{vec}(XD + \Delta X) &= \operatorname{vec}X, \\ V\mathcal{D}^{-1}\operatorname{vec}(uv^T) &= \begin{bmatrix} Gu \\ Lv \end{bmatrix}, \end{aligned}$$

which can be easily verified from the definitions of $\mathcal{D}$, G and L, where V is the matrix defined in (3.6).

Applying the operator V to both sides of (3.2) yields

$$\begin{aligned} V\operatorname{vec}(X^{(k+1)}-X) &= V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X) + V\mathcal{D}^{-1}U(I_{2n}-V\mathcal{D}^{-1}U)^{-1}V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X) \\ &= (I_{2n}+V\mathcal{D}^{-1}U(I_{2n}-V\mathcal{D}^{-1}U)^{-1})V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X) \\ &= (I_{2n}-V\mathcal{D}^{-1}U)^{-1}V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X), \end{aligned} \tag{5.4}$$

where the last equation holds since $I+M(I-M)^{-1}=(I-M)^{-1}$.

We recall that $R=(I_{2n}-V\mathcal{D}^{-1}U)$ and

$$\begin{bmatrix} \widehat{u}^{(k+1)} \\ \widehat{v}^{(k+1)} \end{bmatrix} = R^{-1}\begin{bmatrix} \widetilde{e}-Hu \\ e-Kv \end{bmatrix}$$

(the latter one being Lu's iteration). Now we can explicitly compute

$$R\begin{bmatrix} \widehat{u}^{k+1}-u \\ \widehat{v}^{k+1}-v \end{bmatrix} = \begin{bmatrix} \widetilde{e}-Hu \\ e-Kv \end{bmatrix} - \left(I_{2n}-\begin{bmatrix} G & H \\ K & L \end{bmatrix}\right)\begin{bmatrix} u \\ v \end{bmatrix} =$$
$$\begin{bmatrix} \widetilde{e}-Hv \\ e-Ku \end{bmatrix} - \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} Gu+Hv \\ Ku+Lv \end{bmatrix} = \begin{bmatrix} \widetilde{e}-(I-G)u \\ e-(I-L)v \end{bmatrix} = V\mathcal{D}^{-1}\operatorname{vec}\mathcal{R}(X),$$

and substitute it into (5.4) to get

$$V\operatorname{vec}(X^{(k+1)}-X) = \begin{bmatrix} \widehat{u}^{k+1}-u \\ \widehat{v}^{k+1}-v \end{bmatrix}.$$

Finally, using the definition of V in (3.6), we find that

$$\begin{bmatrix} \widehat{u}^{k+1}-u \\ \widehat{v}^{k+1}-v \end{bmatrix} = V\operatorname{vec}(X^{(k+1)}-X) = \begin{bmatrix} X^{(k+1)}\widetilde{q}-X\widetilde{q} \\ X^{(k+1)T}q-X^Tq \end{bmatrix} = \begin{bmatrix} X^{(k+1)}\widetilde{q}+\widetilde{e}-u \\ X^{(k+1)T}q+e-v \end{bmatrix}$$

and thus $\widehat{u}^{k+1}=X^{(k+1)}\widetilde{q}+\widetilde{e}$, $\widehat{v}^{k+1}=X^{(k+1)T}q+e$. $\square$

The theorem brings deeper insight into Newton's and Lu's iterations. For example, Theorem 6 of [15], which states that Lu's iteration is well-defined and converges monotonically to the minimal solution of the NARE, can now be seen as a special case of Theorem 3.1. Moreover, Lu's iteration can be viewed as a structured Newton's iteration exploiting the displacement structure found in (3.5). Therefore, the two algorithms take the same number of iterations to converge, as the computation they perform is the same. Observe also that the Lu version of this algorithm is slightly faster, since it updates only the $2n$ generators of the matrix $\{X^{(k)}\}$ instead of all the n^2 entries. For this reason, we will only present numerical results regarding Lu's iteration.

6. Shift technique.

In the case where $(c,\alpha)=(1,0)$, the Jacobian M_X appearing in the Newton iteration is singular when X is the solution of the NARE. We refer to this as the *critical case.* Several drawbacks are encountered in the critical case, see the analysis of C.-H. Guo and N. Higham in [8] for more details. The singularity of the Jacobian does not guarantee the quadratic convergence of Newton's iteration, in fact, Newton's and therefore Lu's method converge linearly. Moreover, a perturbation $O(\varepsilon)$ in the coefficients of the equation leads to an $O(\sqrt{\varepsilon})$ variation in the solution.

These drawbacks can be easily removed by means of the shift technique originally introduced be He, Meini and Rhee in [13] and applied to Riccati equations in [9] and [2].

A characterization of the critical case can be given in terms of the eigenvalues of the matrix

$$H = \begin{bmatrix} E & -C \\ B & -A \end{bmatrix}, \tag{6.1}$$

obtained by premultiplying the M-matrix M defined in (1.4) by the matrix $J = \mathrm{diag}(I_n, -I_n)$. In fact, the matrix H has a double zero eigenvalue corresponding to a 2×2 Jordan block (see [9] and the references therein).

The shift technique, as described in [9], consists in a rank-one correction to the matrix H of (6.1) which gives $\widetilde{H} = H + \eta v p^T$, where $\eta > 0$, v is a right eigenvector of H corresponding to the zero eigenvalue and p is an arbitrary vector such that $p^T v = 1$.

The nice feature of this transformation is that the Riccati equation associated with the matrix $\widetilde{H}$ has the same minimal solution as the original one, although the new Jacobian matrix at the solution is not singular. This removes the above mentioned drawbacks. Now, the point is to show that it is still possible to provide a fast implementation of Newton's iteration for the new equation obtained by means of the shift technique. This is the goal of this section.

Under the assumptions (1.2), (1.3) a right eigenvector of H corresponding to zero is $v = \begin{bmatrix} v_1^T & v_2^T \end{bmatrix}^T$, where $v_1 = D^{-1}q$, $v_2 = \Delta^{-1}e$. This can be seen by direct inspection using the fact that $e^T D^{-1} q + q^T \Delta^{-1} e = c = 1$ (see equation (3.3)).

The rank-one correction we construct is

$$\widetilde{H} = H + \eta \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} p^T,$$

where $0 < \eta \leqslant d_1$ and $p^T = \begin{bmatrix} e^T & q^T \end{bmatrix}$. It holds that $p^T v = 1$, in fact $p^T v = e^T D^{-1} q + q^T \Delta^{-1} e = 1$. It is proved in [9] that $\widetilde{H}$ has a simple zero eigenvalue.

The matrix $\widetilde{H}$ defines the new Riccati equation

$$X\widetilde{C}X - X\widetilde{E} - \widetilde{A}X + \widetilde{B} = 0, \tag{6.2}$$

with

$$\widetilde{A} = A - \eta v_2 q^T, \quad \widetilde{B} = B + \eta v_2 e^T, \quad \widetilde{C} = C - \eta v_1 q^T, \quad \widetilde{E} = E + \eta v_1 e^T. \tag{6.3}$$

It is proved in [9] that the minimal nonnegative solution of (1.1) is the minimal nonnegative solution of (6.2).

With the choice of $p^T = [e^T \; q^T]$, $\widetilde{H}$ remains a diagonal plus rank-one matrix, so is $\widetilde{M} = J\widetilde{H}$; hence, we only need to prove that $\widetilde{M}$ is an M-matrix to ensure that the algorithms proposed in Sections 3 and 5 can be applied to the equation (6.2). In fact, we have

$$\widetilde{M} = \begin{bmatrix} D & 0 \\ 0 & \Delta \end{bmatrix} - \begin{bmatrix} q - \eta v_1 \\ e + \eta v_2 \end{bmatrix} \begin{bmatrix} e^T & q^T \end{bmatrix},$$

and since we chose $0 < \eta \leqslant d_1 < d_2 < \ldots < d_n$, and $q \geq 0$, then $q - \eta v_1 = (I_n - \eta D^{-1})q \geqslant 0$, thus $\widetilde{M}$ is a Z-matrix. By the Perron-Frobenius theorem applied to $\rho I - M$, there exists a vector $u > 0$ such that $u^T M = 0$, and in the critical case we have $u^T J v = 0$ (observe that $u^T J$ is a left eigenvector of $H = JM$ corresponding to the zero eigenvalue and recall that right and left eigenvectors corresponding to the same eigenvalue in a Jordan block of dimension $n \geqslant 2$ are orthogonal); therefore,

$$u^T \widetilde{M} = u^T M + \eta u^T J v p^T = 0,$$

thus by part 1 of Lemma 2.1 $\widetilde{M}$ is an M-matrix.

In this way, Newton's iteration applied to equation (6.2) provides a quadratically convergent algorithm of complexity $O(n^2)$ for solving the Riccati equation (1.1) in the critical case. Moreover, since the singularity has been removed, it is expected that X, as minimal solution of (6.2), is better conditioned than with respect to the coefficients of (1.1), and that a higher precision can be reached in the computed solution. This fact is confirmed by the numerical experiments as shown in Section 8

7. Numerical stability. Our first concern about numerical stability is proving that the matrix $R^{(k)} = I_{2n} - V\mathcal{D}^{-1}U^{(k)}$ resulting after the application of the SMW formula to the Jacobian $\mathcal{D} - U^{(k)}V$ is well-conditioned whenever the Jacobian is. In the following analysis, we will assume that the norm $\left\|(\mathcal{D} - U^{(k)}V)^{-1}\right\|_1$ is bounded, and we will drop the superscripts (k) to simplify the notation.

Observe that $0 \leqslant \mathcal{D}^{-1} \leqslant (\mathcal{D} - UV)^{-1}$, therefore $\mathcal{D}$ is well-conditioned. Moreover, one has

$$\mathcal{B}^{-1} = \begin{bmatrix} (\mathcal{D} - UV)^{-1} & 0 \\ V(\mathcal{D} - UV)^{-1} & I \end{bmatrix} \begin{bmatrix} I & U \\ 0 & I \end{bmatrix}, \quad \text{with} \quad \mathcal{B} = \begin{bmatrix} \mathcal{D} & -U \\ -V & I \end{bmatrix};$$

therefore $\mathcal{B}$ is an M-matrix and is well-conditioned. Now, $R = I - V\mathcal{D}^{-1}U$ is the Schur complement of $\mathcal{D}$ in $\mathcal{B}$, and thus R^{-1} is a submatrix of $\mathcal{B}^{-1}$ [5]. This implies $\left\|R^{-1}\right\|_1 \leqslant \left\|\mathcal{B}^{-1}\right\|_1$, hence R is well-conditioned, too.

Another stability problem could arise from the generators growth during the fast Gaussian elimination step. Generators growth has been reported in some cases with the GKO algorithm [17], especially when the starting generators are ill-conditioned. This is not our case, since the starting generators are bounded, and no significant generator growth has been observed during our experiments.

8. Numerical experiments. We consider the numerical examples suggested by L.-Z. Lu [15]. The sequences ω_i and c_i, which appear in the discretization as the nodes and weights of a Gaussian quadrature method, are obtained by dividing the interval $[0, 1]$ into $n/4$ subintervals of equal length, and by applying to each one the 4-nodes Gauss–Legendre quadrature.

The computation has been performed with three different choices of the parameters (c, α), namely, $(0.5, 0.5)$, $(1 - 10^{-6}, 10^{-8})$, and $(1, 0)$. The latter is the critical case, and thus the quadratic convergence of Newton's method is not guaranteed. In this case, the algorithms are more prone to numerical problems, since the matrices to be inverted are near-to-singular.

The algorithms have been implemented in Fortran 90 and the tests have been carried out using the Lahey Fortran compiler on a Xeon biprocessor with 2.8 GHz. We have compared the Lu's algorithm presented in [15] with its fast version based on Algorithm 2. In the critical case, we have made also a comparison with the shifted algorithm of Section 6. To compute the step (5.1) of the Lu algorithm we have solved a linear system using the LAPACK `la_gesv` function.

In Table 8.1 we compare the timing of Lu's algorithm which has a computational cost of $O(n^3)$ ops, with its fast version which costs $O(n^2)$. The numerical results highlight the different order of complexity. Observe that in the critical case the shift technique reduces the timings even further.

In Table 8.2 we compare the relative error of the two methods and the number of steps required. Here the error is computed as $\|\widetilde{X} - X\|_1 / \|X\|_1$, where $\widetilde{X}$ and X are the solution computed in double and in quadruple precision, respectively.

	$\alpha = 0.5$, $c = 0.5$		$\alpha = 10^{-8}$, $c = 1 - 10^{-6}$		$\alpha = 0$, $c = 1$		
n	Lu	LuF	Lu	LuF	Lu	LuF	LuS
32	0.002	0.001	0.006	0.002	0.009	0.005	0.003
64	0.015	0.006	0.028	0.008	0.048	0.015	0.004
128	0.101	0.015	0.20	0.036	0.36	0.061	0.013
256	0.709	0.080	1.6	0.16	2.7	0.29	0.064
512	8.591	0.56	18	1.3	31	2.1	0.39

TABLE 8.1

Comparison of CPU time in seconds of Lu's algorithm (Lu), its fast version presented here (LuF), and the shifted algorithm in the critical case (LuS)

	$\alpha = 0.5$, $c = 0.5$	
n	Lu	LuF
32	$4.8 \cdot 10^{-16}$ (4)	$2.3 \cdot 10^{-16}$ (5)
256	$1.6 \cdot 10^{-15}$ (4)	$4.0 \cdot 10^{-16}$ (5)

	$\alpha = 0$, $c = 1$		
n	Lu	LuF	LuS
32	$5.2 \cdot 10^{-8}$ (25)	$4.2 \cdot 10^{-8}$ (26)	$4.4 \cdot 10^{-16}$ (6)
256	$4.6 \cdot 10^{-8}$ (25)	$8.0 \cdot 10^{-8}$ (25)	$1.2 \cdot 10^{-15}$ (6)

TABLE 8.2

Comparison of the relative error (and in parentheses the number of steps) of Lu's algorithm (Lu), its fast version presented here (LuF), and the shifted algorithm in the critical case (LuS)

The stopping criterion is based on the computation of

$$\mathrm{Res} = \frac{\|u_k - u_{k-1}\|_1 + \|v_k - v_{k-1}\|_1}{2}.$$

Observe that the cost of computing Res is negligible.

As one can see, in the critical case the accuracy of the solution obtained with the non-shifted algorithms is of the order of $O(\sqrt{\varepsilon})$, where ε is the machine precision, in strict accordance with [8]. The speed-up obtained is greater than 2 even for small values of n. In the critical case with size $n = 512$ our algorithm is about 80 times faster than Lu's original algorithm. The problems deriving from the large number of steps and the poor accuracy are completely removed by the shift technique.

9. Generalizations and future work. Our algorithm can be easily extended to the case where M is diagonal plus rank k.

A challenging issue is to prove that Lu's iteration for this problem can be effectively computed with less than $O(n^2)$ ops. Actually, the literature provides algorithms for computing the Cauchy matrix–vector product [3] and for approximating the inverse of a Cauchy matrix [18] with $O(n \log^2 n)$ ops. If these algorithms can be adapted to deal with singular displacement operators, and if the approximation and the numerical errors introduced do not destroy the quadratic convergence of Newton's method, then the computational cost of our algorithm could be reduced further.

Another interesting issue is the acceleration of existing algorithms like the (shifted) Structure-preserving Doubling Algorithm [10, 12] or the (shifted) Logarithmic and Cyclic reduction [2, 7], relying on the specific structure of the problem.

10. Acknowledgments. The authors would like to thank Luca Gemignani and Beatrice Meini for their helpful comments and suggestions.

REFERENCES

[1] Abraham Berman and Robert J. Plemmons. *Nonnegative matrices in the mathematical sciences*, volume 9 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994. Revised reprint of the 1979 original.

[2] 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.

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

[4] Israel Gohberg, Thomas Kailath, and Vadim Olshevsky. Fast Gaussian elimination with partial pivoting for matrices with displacement structure. *Math. Comp.*, 64(212):1557–1576, 1995.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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, Italy, May 2005.

[10] 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 pubblication.

[11] 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.

[12] 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.

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

[14] 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.

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

[16] 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.

[17] Franklin T. Luk, editor. *Error analysis of a fast partial-pivoting method for structured matrices*, volume 2563 of *Advanced Signal Processing Algorithms, Proceedings of SPIE*, 1995.

[18] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. A fast algorithm for the inversion of general Toeplitz matrices. *Comput. Math. Appl.*, 50(5-6):741–752, 2005.