Journal of Computational and Applied Mathematics 250 (2013) 175–189

Contents lists available at SciVerse ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Low memory and low complexity iterative schemes for a nonsymmetric algebraic Riccati equation arising from transport theory☆

Bo Yu [a,*], Dong-Hui Li [b], Ning Dong [a]

[a] School of Science, Hunan University of Technology, Zhuzhou, 412000, PR China

[b] School of Mathematical Sciences, South China Normal University, Guangzhou, 510631, PR China

ARTICLE INFO

Article history:
Received 1 June 2011
Received in revised form 2 October 2012

MSC:
65F30
65H10
15A24

Keywords:
Nonsymmetric algebraic Riccati equation
Newton's method
Fixed-point iteration
Low memory and low complexity
Factored alternating-direction-implicit iteration

ABSTRACT

We reconsider Newton's method and two fixed-point methods for finding the minimal positive solution of a nonsymmetric algebraic Riccati equation arising from transport theory. We rewrite the subproblem of the Newton and fixed-point iterative schemes into an equivalent form with some special structure. By the use of the particular structure of the subproblem, we then present low memory and low complexity versions of these iterative methods with a factored alternating-direction-implicit iteration. Some properties of eigenvalues for iterative coefficient matrices in solving the subproblem are derived and the convergence of the proposed methods is established. Numerical experiments show that the new iterative schemes are highly efficient to obtain the minimal positive solution. The proposed low memory and low complexity Newton's method is particularly efficient for solving large scale Riccati equation arising from transport theory.

1. Introduction

In transport theory related to the transmission of a particle beam in a solid medium, an integrodifferential equation can be transformed, by the use of Gauss–Legendre quadrature formula, into the following nonsymmetric algebraic Riccati equation (NARE) [1–3]

$$\mathcal{R}(X) = XCX - AX - XD + B = 0, \tag{1.1}$$

where $B, C \in \mathbb{R}^{n\times n}$ are rank one matrices of the forms

$$B = ee^T, \qquad C = qq^T \tag{1.2}$$

with

$$e = (1, 1, \ldots, 1)^T, \qquad q = (q_1, q_2, \ldots, q_n)^T, \qquad q_i = \frac{c_i}{2\omega_i}, \quad i = 1, 2, \ldots, n$$

☆ Supported in part by the NSF of China grant 11071087, Hunan KIPSS 12FEFM03 and the Fund of Hunan Education Department grant 12C0075.

* Corresponding author. Tel.: +86 073122622158.
E-mail addresses: wenyubwenyub@yahoo.com.cn, boyu_hut@yahoo.cn (B. Yu), dhli@scnu.edu.cn (D.-H. Li).

0377-0427/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cam.2013.03.017

and $A, D \in \mathbb{R}^{n\times n}$ are some summations of diagonal matrices and rank one matrices. Specifically, they take the forms

$$A = \Delta - eq^T, \qquad D = \Gamma - qe^T \tag{1.3}$$

with

$$\Delta = \mathrm{Diag}(\delta_1, \delta_2, \ldots, \delta_n), \qquad \Gamma = \mathrm{Diag}(\gamma_1, \gamma_2, \ldots, \gamma_n),$$
$$\delta_i = \frac{1}{c\omega_i(1+\alpha)}, \qquad \gamma_i = \frac{1}{c\omega_i(1-\alpha)}, \quad i = 1, 2, \ldots, n,$$

and $\alpha \in [0, 1), c \in (0, 1]$. The two parameter sets $\{\omega_i\}_{i=1}^n$ and $\{c_i\}_{i=1}^n$ denote the nodes and weights, respectively, of the Gauss–Legendre formula satisfying

$$0 < \omega_n < \cdots < \omega_1 < 1 \tag{1.4}$$

and $\sum_{i=1}^n c_i = 1$ with $c_i > 0$. It is easy to see that the diagonal elements of Δ and Γ defined above satisfy

$$0 < \delta_1 < \delta_2 < \cdots < \delta_n \quad \text{and} \quad 0 < \gamma_1 < \gamma_2 < \cdots < \gamma_n. \tag{1.5}$$

The minimal positive solution of Eq. (1.1) is of great interest in physics. The existence of the minimal positive solution has been well studied in [1,3]. The corresponding numerical iterative methods for finding the minimal positive solution of NARE (1.1) have also been extensively studied [4–9,2,10–15]. Among these methods, Newton's method and the fixed-point methods are two important classes of iterative schemes. In [16], Guo and Laub developed the following Newton's method in the matrix form for solving NARE (1.1)

$$(A - X^{(k)}C)X^{(k+1)} + X^{(k+1)}(D - CX^{(k)}) = B - X^{(k)}CX^{(k)}, \quad k = 0, 1, \ldots, X_0 = 0. \tag{1.6}$$

The complexity at each step, when using the Bartels–Stewart algorithm [17], is about $O(n^3)$ flops. Lu [18] further converted Eq. (1.6) into a linear equation of scale $2n$ and reused Newton's method (1.6) for solving NARE (1.1), but the complexity at each iteration, unfortunately, still remained in $O(n^3)$ flops. Recently, Bini, Iannazzo and Poloni [19] suggested employing the Gohberg–Kailath–Olshevsky (GKO) algorithm [20] for solving the structured linear system given by Lu [18], so that the complexity at each step can be down to $O(n^2)$ flops. The fixed-point methods are another class of effective iterative methods for solving (1.1). Guo and Laub [16] studied the following uniform fixed-point iteration format for solving NARE (1.1):

$$A_1X^{(k+1)} + X^{(k+1)}D_1 = X^{(k)}CX^{(k)} + A_2X^{(k)} + X^{(k)}D_2 + B, \quad k = 0, 1, \ldots, X_0 = 0, \tag{1.7}$$

where matrices A_1 and A_2, D_1 and D_2 were some regular splitting of matrices A and D, respectively, in the sense that $A = A_1 - A_2$ and $D = D_1 - D_2$ with $A_1^{-1} \geq 0, A_2 \geq 0, D_1^{-1} \geq 0$ and $D_2 \geq 0$; see [21] for details. Lu [22] transformed Eq. (1.7) into a couple of vectors and devised a simple iteration (SI) with the complexity about $4n^2$ flops per iteration. Bai, Gao and Lu [23] further designed a class of nonlinear splitting iteration methods, including the nonlinear block Jacobi (NBJ) and the nonlinear block Gauss–Seidel (NBGS) iteration methods, which share the same complexity with SI iteration but can obtain faster convergence.

In this paper, we reconsider Newton's method (1.6) and the fixed-point methods (1.7) to compute the minimal positive solution of (1.1). Especially for the fixed-point methods, we take $A_1 = \Delta, D_1 = \Gamma$ and $A_1 = A, D_1 = D$ in (1.7) since they respectively correspond to the simplest iteration format and the fastest convergent iteration format among a class of fixed-point iterations in [16]. Our purpose is to further reduce the complexity of these two class of methods. We first notice that, by introducing two vectors

$$u^{(k)} = X^{(k)}q + e \quad \text{and} \quad v^{(k)} = (X^{(k)})^T q + e. \tag{1.8}$$

Newton's method and two fixed-point methods can be respectively rewritten as

$$\text{Newton}: (\Delta - u^{(k)}q^T)X^{(k+1)} + X^{(k+1)}(\Gamma - q(v^{(k)})^T) = (e, e - u^{(k)})(e, v^{(k)} - e)^T, \tag{1.9}$$
$$\text{FP1}: \Delta X^{(k+1)} + X^{(k+1)}\Gamma = u^{(k)}(v^{(k)})^T, \tag{1.10}$$
$$\text{FP2}: (\Delta - eq^T)X^{(k+1)} + X^{(k+1)}(\Gamma - qe^T) = (e, u^{(k)} - e)(e, v^{(k)} - e)^T. \tag{1.11}$$

This shows when $u^{(k)}$ and $v^{(k)}$ are obtained at the k-th step, a Sylvester equation of the form

$$FX + XG = E \tag{1.12}$$

is required to be solved. If the scale of (1.12) is small, a direct method such as in [17] is enough. As the scale of (1.12) grows, the alternating-direction-implicit (ADI) like methods are more preferred. Bai, Guo and Xu [24] combined alternate splitting and successive approximating about a nonlinear operator to devise an alternately linearized implicit (ALI) method to solve NARE (1.1). This method was motivated by the Hermitian and skew-Hermitian splitting (HSS) iterations in [25] for solving non-Hermitian positive definite linear systems. In spirit of the HSS iteration, Bai [26] further developed a matrix variant of HSS iteration to solve the Sylvester equation (1.12) and provided its convergence. Here we observe that the subproblem (1.12) of iterations (1.9)–(1.11) has some attractive characteristics as below.

- The coefficient matrices F and G are diagonal or diagonal-plus-rank-one matrices. Both of them belong to a special class of the generalized companion matrices introduced in [27].
- All eigenvalues of F and G, as will be shown in Theorem 4.2, are real positive numbers and can be evaluated in $O(n)$ flops by existing methods such as those in [27] or [28].
- The matrix E on the right-hand side is of rank one or two, much less than the dimension of the Sylvester equation in general.

These nice structured properties urge us to use the well-developed factored alternating-direction-implicit (FADI) iteration [29,30] to solve the Sylvester equation (1.12) and, in this case, the solution of (1.12) can be approximated by

$$Y = S_J \Lambda_J T_J^T,$$

where $\Lambda_J \in \mathbb{R}^{J\times J}$ is a diagonal matrix, S_J and $T_J \in \mathbb{R}^{n\times J}$ are thin and tall matrices since J is far less than n [29]. As a result, vectors in (1.8) can be approximated by

$$u^{(k)} \approx Y^{(k)}q + e \quad \text{and} \quad v^{(k)} \approx (Y^{(k)})^T q + e,$$

and the complexity of iterations (1.9)–(1.11) is desirable to be further reduced as $J \ll n$. We will see later in the numerical experiments that the customary Newton and fixed-point methods (1.9)–(1.11) with incorporating into a low memory version FADI iteration work very well for computing the minimal positive solution of NARE (1.1). They are particularly very efficient for solving middle and large scale equations.

The rest of this paper is organized as follows. We review the FADI iteration in Section 2 and present the low memory and low complexity iterative schemes in Section 3. Section 4 is devoted to some properties of eigenvalues of iterative matrices and the convergence of the proposed algorithms. In Section 5, we do some numerical experiments to test the proposed methods and compare their performances with those of the recently developed NBGS method in [23] and the fast Newton method in [19].

Some words for notations. Throughout this paper, we use "$\circ$" to denote the Hadamard product of vectors or matrices. Let I_r and I be the identity matrices of order r and n, respectively. For matrix $A \in \mathbb{R}^{n\times n}$, we denote by $\sigma(A)$ and $\rho(A)$ its spectrum and spectral radius, respectively. For a diagonal matrix $D \in \mathbb{R}^{n\times n}$ and a vector $d \in \mathbb{R}^n$, diag(D) represents the vector whose elements are the diagonal entries of D and, diag(d) represents the diagonal matrix whose diagonal entries are elements of d. We abbreviate the block diagonal matrix $\begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}$ as Diag[D_1, D_2].

2. The FADI iteration for the Sylvester equation with a low rank structure

The alternating-direction-implicit (ADI) iteration, described below, is an effective method for solving Sylvester equation with coefficient matrices large and sparse [21].

Given two sets of ADI parameters $\{f_j\}$ and $\{g_j\}$ and the initial guess $Y^{(0)} = 0$. For $j = 1, 2, \ldots$ until convergence, proceed the iteration

$$\begin{cases} (F + f_j I)Y^{(j-1/2)} = Y^{(j-1)}(f_j I - G) + E, \\ Y^{(j)}(G + g_j I) = (g_j I - F)Y^{(j-1/2)} + E. \end{cases}$$

It is not difficult to see that $Y^{(j)}$ can be expressed by $Y^{(j-1)}$ explicitly. Specifically, we have

$$Y^{(j)} = (g_j I - F)(F + f_j I)^{-1}Y^{(j-1)}(f_j I - G)(G + g_j I)^{-1} + (f_j + g_j)(F + f_j I)^{-1}E(G + g_j I)^{-1}. \tag{2.1}$$

When the matrix E in Sylvester equation (1.12) has a low rank structure, i.e. $E = UV^T$ with $U, V \in \mathbb{R}^{n\times r}$ and $r \ll n$, the ADI iteration can be implemented in a more economic way, called factored ADI (FADI) iteration as below [29].

Let $S_1 = (F + f_1 I)^{-1}U$, $T_1 = (G^T + g_1 I)^{-1}V$, $\Lambda_1 = (g_1 + f_1)I_r$. For $j = 2, 3, \ldots$ until convergence, compute

$$\begin{cases} S_j = [(F + f_j I)^{-1}U, (g_j I - F)(F + f_j I)^{-1}S_{j-1}], \\ T_j = [(G^T + g_j I)^{-1}V, (f_j I - G^T)(G^T + g_j I)^{-1}T_{j-1}], \\ \Lambda_j = \text{Diag}[(g_j + f_j)I_r, \Lambda_{j-1}], \\ Y^{(j)} = S_j \Lambda_j T_j^T. \end{cases}$$

Since the order of the ADI parameters $\{g_j\}$ and $\{f_j\}$ makes no significance to the ADI iteration [29,30], the FADI iteration with parameters of reverse order, after J step, can be reformulated as

$$\begin{cases} S_J = [S_{J_1}, (g_1 I - F)(F + f_2 I)^{-1}S_{J_1}, \ldots, (g_{J-1} I - F)(F + f_J I)^{-1}S_{J_{J-1}}], \\ T_J = [T_{J_1}, (f_1 I - G^T)(G^T + g_2 I)^{-1}T_{J_1}, \ldots, (f_{J-1} I - G^T)(G^T + g_J I)^{-1}T_{J_{J-1}}], \\ \Lambda_J = \text{Diag}[(g_1 + f_1)I_r, \ldots, (g_J + f_J)I_r], \\ Y^{(J)} = S_J \Lambda_J T_J^T \end{cases} \tag{2.2}$$

with $S_{J_1} = S_1$, $T_{J_1} = T_1$, $S_{J_{j+1}} = (g_j I - F)(F + f_{j+1} I)^{-1}S_{J_j}$ and $T_{J_{j+1}} = (f_j I - G^T)(G^T + g_{j+1} I)^{-1}T_{J_j}$ for $j = 1, \ldots, J-1$.

It should be noted that the convergence of FADI iteration relies heavily on the choice of ADI parameters $\{f_j\}_{j=1}^{J}$ and $\{g_j\}_{j=1}^{J}$, which is usually hard to determine for cases such as in [29,31,32]. For the subproblem (1.12) of iterations (1.9)–(1.11), the nice structure of coefficient matrices buys quite a lot in obtaining the optimal ADI parameters via Wachspress's method [33,34]. Furthermore, the storage of the S_J and T_J in (2.2) is unnecessary since only two matrix–vector products are required per iteration. Therefore, iterations (1.9)–(1.11) can progress in a low complexity and low memory way as described in the next section.

3. Low memory and low complexity iterative schemes

3.1. Low memory and low complexity Newton–FADI iteration method

For the subproblem of Newton iterative scheme (1.9), the two factor matrices S_J and T_J in (2.2) of the FADI iteration can be reformulated as some Hadamard products among vectors.

Proposition 3.1. *Let S_J and T_J be factor matrices generated by the FADI iteration for solving (1.9). For each $i = 1, 2, \ldots, J$, denote by (s_{2i-1}, s_{2i}) and (t_{2i-1}, t_{2i}) the i-th pair of vectors in S_J and T_J, respectively. For $i = 1, 2, \ldots, J$, define vectors*

$$d_{\Delta_i} = diag[(\Delta + f_i I)^{-1}], \qquad d_{\Gamma_i} = diag[(\Gamma + g_i I)^{-1}] \tag{3.1}$$

and scalars

$$h_{1_i} = \frac{q^T(\Delta + f_i I)^{-1}e}{1 - q^T(\Delta + f_i I)^{-1}u^{(k)}}, \qquad h_{2_i} = \frac{q^T(\Delta + f_i I)^{-1}(e - u^{(k)})}{1 - q^T(\Delta + f_i I)^{-1}u^{(k)}}, \tag{3.2}$$

$$\hat{h}_{1_i} = \frac{q^T(\Gamma + g_i I)^{-1}e}{1 - q^T(\Gamma + g_i I)^{-1}v^{(k)}}, \qquad \hat{h}_{2_i} = \frac{q^T(\Gamma + g_i I)^{-1}(v^{(k)} - e)}{1 - q^T(\Gamma + g_i I)^{-1}v^{(k)}}. \tag{3.3}$$

For $i = 2, 3, \ldots, J$, define vectors

$$\hat{d}_{\Gamma_i} = diag[(f_{i-1}I - \Gamma)(\Gamma + g_i I)^{-1}], \qquad \hat{d}_{\Delta_i} = diag[(g_{i-1}I - \Delta)(\Delta + f_i I)^{-1}] \tag{3.4}$$

and scalars

$$h_{s_{2i-1}} = \frac{q^T(\Delta + f_i I)^{-1}s_{2i-1}}{1 - q^T(\Delta + f_i I)^{-1}u^{(k)}}, \qquad h_{s_{2i}} = \frac{q^T(\Delta + f_i I)^{-1}s_{2i}}{1 - q^T(\Delta + f_i I)^{-1}u^{(k)}}, \tag{3.5}$$

$$\hat{h}_{t_{2i-1}} = \frac{q^T(\Gamma + g_i I)^{-1}t_{2i-1}}{1 - q^T(\Gamma + g_i I)^{-1}v^{(k)}}, \qquad \hat{h}_{t_{2i}} = \frac{q^T(\Gamma + g_i I)^{-1}t_{2i}}{1 - q^T(\Gamma + g_i I)^{-1}v^{(k)}}. \tag{3.6}$$

Then we have for $i = 2, \ldots, J$

$$[s_1, s_2] = [d_{\Delta_1} \circ (e + h_{1_1}u^{(k)}), d_{\Delta_1} \circ (e + (h_{2_1} - 1)u^{(k)})], \tag{3.7}$$

$$[t_1, t_2] = [d_{\Gamma_1} \circ (e + \hat{h}_{1_1}v^{(k)}), d_{\Gamma_1} \circ (e + (1 - \hat{h}_{2_1})v^{(k)})], \tag{3.8}$$

$$[s_{2i-1}, s_{2i}] = [\hat{d}_{\Delta_i} \circ (s_{2i-3} + h_{s_{2i-1}}u^{(k)}) + h_{s_{2i-1}}u^{(k)}, \hat{d}_{\Delta_i} \circ (s_{2i-2} + h_{s_{2i}}u^{(k)}) + h_{s_{2i}}u^{(k)}], \tag{3.9}$$

$$[t_{2i-1}, t_{2i}] = [\hat{d}_{\Gamma_i} \circ (t_{2i-1} + \hat{h}_{t_{2i-1}}v^{(k)}) + \hat{h}_{t_{2i-1}}v^{(k)}, \hat{d}_{\Gamma_i} \circ (t_{2i} + \hat{h}_{t_{2i}}v^{(k)}) + \hat{h}_{t_{2i}}v^{(k)}]. \tag{3.10}$$

Proof. By using the Sherman–Morrison–Woodbury formula (see, e.g. [35]) to matrices

$$(\Delta + f_i I - u^{(k)}q^T)^{-1} \quad \text{and} \quad (\Gamma + g_i I - q(v^{(k)})^T)^{-1},$$

equalities (3.7)–(3.10) hold true. □

Once the factor matrices S_J and T_J are constructed by Proposition 3.1, we can use matrix $Y^{(J)} = S_J \Lambda_J T_J^T$ with $\Lambda_J = \mathrm{Diag}[\,(g_1 + f_1)I_2, \ldots, (g_J + f_J)I_2\,]$ as an approximation to the exact k-th step Newton iteration matrix $X^{(k)}$. Moreover, each Newton step can be implemented in a low memory way since two vectors ($u^{(k)}$ and $v^{(k)}$), rather than two matrices (S_J and T_J), are required to be updated and stored. We give all steps of the low memory and low complexity Newton–FADI iteration in Algorithm 1. The storage and the complexity of the algorithm will be given in the last part of this section (see Table 3.1).

Algorithm 1. Low Memory and Low Complexity Newton-FADI Algorithm.

Inputs: Vectors δ, γ, q and initial guess $u^{(1)}$ and $v^{(1)}$. Outputs: u, $v \in \mathbb{R}^n$.

For $k = 1, 2, 3, \ldots,$ until convergence, do
  $u_{old}^{(k)} := u^{(k)}$, $u_{old}^{(k)} := u^{(k)}$.
  Compute extremal eigenvalues $\lambda_1^{(k)}$ and $\lambda_n^{(k)}$ of $\Delta - u_{old}^{(k)} q^T$ and $\tau_1^{(k)}$ and $\tau_n^{(k)}$ of $\Gamma - q(v_{old}^{(k)})^T$.
  Determine the number of ADI iterations J_k and the optimal parameters $\{f_j\}_{j=1}^{J_k}$, $\{g_j\}_{j=1}^{J_k}$.
  $d_\Delta := d_{\Delta_1}$, $d_\Gamma := d_{\Gamma_1}$ with d_{Δ_1}, d_{Γ_1} defined in (3.1).
  Compute $h_1 := h_{1_1}$, $h_2 := h_{2_1}$, $\hat{h}_1 := \hat{h}_{1_1}$, $\hat{h}_2 := \hat{h}_{2_1}$ with (3.2) and (3.3).
  $[s_1, s_2] := d_\Delta \circ [(1 + h_1)u_{old}^{(k)} + e, (h_2 - 1)u_{old}^{(k)} + e]$.
  $[t_1, t_2] = d_\Gamma \circ [(1 + \hat{h}_1)v_{old}^{(k)} + e, (\hat{h}_2 + 1)v_{old}^{(k)} - e]$.
  $u^{(k)} := (g_1 + f_1)(t_1^T q s_1 + t_2^T q s_2)$.
  $v^{(k)} := (g_1 + f_1)(s_1^T q t_1 + s_2^T q t_2)$.
  For $i = 2, 3, \ldots, J_k$, do
    $d_\Delta := \hat{d}_{\Delta_i}$, $d_\Gamma := \hat{d}_{\Gamma_i}$ with $\hat{d}_{\Delta_i}$, $\hat{d}_{\Gamma_i}$ defined in (3.4).
    Compute $h_1 := h_{s_{2i-1}}$, $h_2 := h_{s_{2i}}$, $\hat{h}_1 := \hat{h}_{t_{2i-1}}$, $\hat{h}_2 := \hat{h}_{t_{2i}}$ with $v^{(k)} = v_{old}^{(k)}$ and $u^{(k)} = u_{old}^{(k)}$ in (3.5) and (3.6).
    $[s_1, s_2] := [d_\Delta \circ (s_1 + h_1 u_{old}^{(k)}) + h_1 u_{old}^{(k)}, d_\Delta \circ (s_2 + h_2 u_{old}^{(k)}) + h_2 u_{old}^{(k)}]$.
    $[t_1, t_2] := [d_\Gamma \circ (t_1 + \hat{h}_1 v_{old}^{(k)}) + \hat{h}_1 v_{old}^{(k)}, d_\Gamma \circ (t_2 + \hat{h}_2 v_{old}^{(k)}) + \hat{h}_2 v_{old}^{(k)}]$.
    $u^{(k)} := u^{(k)} + (g_i + f_i)(t_1^T q s_1 + t_2^T q s_2)$.
    $v^{(k)} := v^{(k)} + (g_i + f_i)(s_1^T q t_1 + s_2^T q t_2)$.
  End
  $u^{(k)} := u^{(k)} + e$. $v^{(k)} := v^{(k)} + e$.
  If $\max\{\frac{\|u^{(k)} - u_{old}^{(k)}\|}{\|u^{(k)}\|}, \frac{\|v^{(k)} - v_{old}^{(k)}\|}{\|v^{(k)}\|}\} < tol$, then $u := u^{(k+1)}$, $v := v^{(k+1)}$, stop.
End

3.2. Low memory and low complexity FP-FADI iteration methods

For subproblems of FP iterations (1.10)–(1.11), the concise representations of the factor matrices S_J and T_J can be derived in a similar way to that of Proposition 3.1.

Proposition 3.2. *Let S_J and T_J be the factor matrices generated by the FADI iteration.*

(i) *FP1 iteration. For $i = 1, 2, \ldots, J$, denote by s_i and t_i the i-th vector in S_J and T_J respectively, and d_{Δ_i}, d_{Γ_i}, $\hat{d}_{\Delta_i}$ and $\hat{d}_{\Gamma_i}$ be the same as those in Proposition 3.1. Then we have*

$$s_1 = d_{\Delta_1} \circ u^{(k)}, \qquad t_1 = d_{\Gamma_1} \circ v^{(k)} \tag{3.11}$$

and for $i = 2, 3, \ldots, J$

$$s_i = \hat{d}_{\Delta_i} \circ s_{i-1}, \qquad t_i = \hat{d}_{\Gamma_i} \circ t_{i-1}. \tag{3.12}$$

(ii) *FP2 iteration. For $i = 1, 2, \ldots, J$, denote by (s_{2i-1}, s_{2i}) and (t_{2i-1}, t_{2i}) the i-th pair of vectors in S_J and T_J respectively, and d_{Δ_i}, d_{Γ_i} $\hat{d}_{\Delta_i}$ and $\hat{d}_{\Gamma_i}$ be the same with Proposition 3.1. Define*

$$h_{1_i} = \frac{q^T(\Delta + f_i I)^{-1} e}{1 - q^T(\Delta + f_i I)^{-1} e}, \qquad h_{2_i} = \frac{q^T(\Delta + f_i I)^{-1}(u^{(k)} - e)}{1 - q^T(\Delta + f_i I)^{-1} e}, \tag{3.13}$$

$$\hat{h}_{1_i} = \frac{q^T(\Gamma + g_i I)^{-1} e}{1 - q^T(\Gamma + g_i I)^{-1} e}, \qquad \hat{h}_{2_i} = \frac{q^T(\Gamma + g_i I)^{-1}(v^{(k)} - e)}{1 - q^T(\Gamma + g_i I)^{-1} e} \tag{3.14}$$

for $i = 1, 2, \ldots, J$ and

$$h_{s_{2i-1}} = \frac{q^T(\Delta + f_i I)^{-1} s_{2i-1}}{1 - q^T(\Delta + f_i I)^{-1} e}, \qquad h_{s_{2i}} = \frac{q^T(\Delta + f_i I)^{-1} s_{2i}}{1 - q^T(\Delta + f_i I)^{-1} e}, \tag{3.15}$$

$$\hat{h}_{t_{2i-1}} = \frac{q^T(\Gamma + g_i I)^{-1} t_{2i-1}}{1 - q^T(\Gamma + g_i I)^{-1} e}, \qquad \hat{h}_{t_{2i}} = \frac{q^T(\Gamma + g_i I)^{-1} t_{2i}}{1 - q^T(\Gamma + g_i I)^{-1} e} \tag{3.16}$$

for $i = 2, 3, \ldots, J$. Then we have

$$[s_1, s_2] = [d_{\Delta_1} \circ (1 + h_{1_1})e, d_{\Delta_1} \circ (u^{(k)} + (h_{2_1} - 1)e)], \tag{3.17}$$

$$[t_1, t_2] = [d_{\Gamma_1} \circ (1 + \hat{h}_{1_1})e, d_{\Gamma_1} \circ (v^{(k)} + (\hat{h}_{2_1} - 1)e)], \tag{3.18}$$

and for $i = 2, 3, \ldots, J$

$$[s_{2i-1}, s_{2i}] = [\hat{d}_{\Delta_i} \circ (s_{2i-3} + h_{s_{2i-1}} e) + h_{s_{2i-1}} e, \hat{d}_{\Delta_i} \circ (s_{2i-2} + h_{s_{2i}} e) + h_{s_{2i}} e], \tag{3.19}$$

$$[t_{2i-1}, t_{2i}] = [\hat{d}_{\Gamma_i} \circ (t_{2i-1} + \hat{h}_{t_{2i-1}} e) + \hat{h}_{t_{2i-1}} e, \hat{d}_{\Gamma_i} \circ (t_{2i} + \hat{h}_{t_{2i}} e) + \hat{h}_{t_{2i}} e]. \tag{3.20}$$

Incorporating (3.11)–(3.12) into (2.2) and (3.17)–(3.20) into (2.2) respectively yields the FP1-FADI iteration and FP2-FADI iteration whose steps, with the low memory implementation as above, are listed in Algorithms 2 and 3.

Algorithm 2. Low Memory and Low Complexity FP1-FADI Algorithm.
Inputs: Vectors δ, γ, q and initial guess $u^{(1)}$ and $v^{(1)}$. Outputs: $u, v \in \mathbb{R}^n$.
01. Determine the iteration number J and the optimal ADI parameters $\{f_j\}_{j=1}^J$ and $\{g_j\}_{j=1}^J$.
02. For $k = 1, 2, 3, \ldots,$ until convergence, do
03. $u_{old}^{(k)} := u^{(k)}$, $v_{old}^{(k)} := v^{(k)}$.
04. $d_\Delta := d_{\Delta_1}$, $d_\Gamma := d_{\Gamma_1}$.
05. $s := d_\Delta \circ u_{old}^{(k)}$, $t := d_\Gamma \circ v_{old}^{(k)}$.
06. $u^{(k)} := (f_1 + g_1) t^T q s$, $v^{(k)} := (f_1 + g_1) s^T q t$.
07. For $i = 2, 3, \ldots, J$, do
08. $d_\Delta := \hat{d}_{\Delta_i}$, $d_\Gamma := \hat{d}_{\Gamma_i}$.
09. $s := d_\Delta \circ s$, $t := d_\Gamma \circ t$.
10. $u^{(k)} := u^{(k)} + (f_i + g_i) t^T q s$.
11. $v^{(k)} := v^{(k)} + (f_i + g_i) s^T q t$.
12. End
13. $u^{(k)} := u^{(k)} + e$, $v^{(k)} := v^{(k)} + e$.
14. If $\max\{\frac{\|u^{(k)} - u_{old}^{(k)}\|}{\|u^{(k)}\|}, \frac{\|v^{(k)} - v_{old}^{(k)}\|}{\|v^{(k)}\|}\} < tol$, then $u := u^{(k)}$, $v := v^{(k)}$, stop.
15. End

Algorithm 3. Low Memory and Low Complexity FP2-FADI Algorithm.
Inputs: Vectors δ, γ, q and initial guess $u^{(1)}$ and $v^{(1)}$. Outputs: $u, v \in \mathbb{R}^n$.
01. Compute the extremal eigenvalues of $\Delta - eq^T$ and $\Gamma - qe^T$.
02. Determine the number of FADI iteration J and the optimal parameters $\{f_j\}_{j=1}^J$, $\{g_j\}_{j=1}^J$.
03. For $k = 1, 2, 3, \ldots,$ until convergence, do
04. $u_{old}^{(k)} := u^{(k)}$, $u_{old}^{(k)} := u^{(k)}$.
05. $d_\Delta := d_{\Delta_1}$ and $d_\Gamma := d_{\Gamma_1}$.
06. $h_1 := h_{1_1}$, $h_2 := h_{2_1}$, $\hat{h}_1 := \hat{h}_{1_1}$, $\hat{h}_2 := \hat{h}_{2_1}$ with formulas (3.13) and (3.14).
07. Compute $[s_1, s_2]$ and $[t_1, t_2]$ by (3.17) and (3.18).
08. $u^{(k)} = (g_1 + f_1)(t_1^T q s_1 + t_2^T q s_2)$.
09. $v^{(k)} = (g_1 + f_1)(s_1^T q t_1 + s_2^T q t_2)$.
10. For $i = 2, 3, \ldots, J$, do
11. $d_\Delta := \hat{d}_{\Delta_i}$, $d_\Gamma := \hat{d}_{\Gamma_i}$.
12. $h_1 := h_{s_{2i-1}}$, $h_2 := h_{s_{2i}}$, $\hat{h}_1 := \hat{h}_{t_{2i-1}}$, $\hat{h}_2 := \hat{h}_{t_{2i}}$ with formulas (3.15) and (3.16).
13. Compute $[s_1, s_2]$ and $[t_1, t_2]$ by (3.19) and (3.20).
14. $u^{(k)} := u^{(k)} + (g_i + f_i)(t_1^T q s_1 + t_2^T q s_2)$.
15. $v^{(k)} := v^{(k)} + (g_i + f_i)(s_1^T q t_1 + s_2^T q t_2)$.
16. End
17. $u^{(k)} := u^{(k)} + e$, $v^{(k)} := v^{(k)} + e$.
18. If $\max\{\frac{\|u^{(k)} - u_{old}^{(k)}\|}{\|u^{(k)}\|}, \frac{\|v^{(k)} - v_{old}^{(k)}\|}{\|v^{(k)}\|}\} < tol$, then $u := u^{(k+1)}$, $v := v^{(k+1)}$, stop.
19. End

Once the outputs u and v are obtained by Algorithms 1–3, the minimal positive solution X^* of NARE (1.1) can be approximated readily by $(\bar{X})_{i,j} = \frac{u_i v_j}{\delta_i + \gamma_j}$. On the other hand, one important issue in Algorithms 1–3 is how to compute the extremal eigenvalues, the number of inner iterations and the optimal ADI parameters. We dwell on these details in the next subsection.

3.3. Computation of the extremal eigenvalues, the number of inner iterations and the optimal ADI parameters

Extremal eigenvalues. For Algorithm 1, Theorem 4.2 (see later) clearly shows the intervals where the two pairs of extremal eigenvalues lie in. Thus at each outer iteration, they can be computed by a hybrid Newton–secant iteration as in [28]. In

Table 3.1
Storage and computational flops of Algorithms 1–3 at the k-th outer iteration.

	STORAGE COST	FLOPS	
		EIGENVALUES	FADI
ALG. 1	$11n$	$40K_k n$	$40n + 50(J_k(n) - 1)n$
ALG. 2	$7n$	0	$10n + 10(J(n) - 1)n$
ALG. 3	$11n$	0	$36n + 48(J(n) - 1)n$

addition, the so-called "warm start" strategy is also adopted in our algorithm to proceed the successive iterations. That is, except at the first iteration, the currently obtained minimal (maximal) eigenvalue is chosen as the initial guess in the iterative process for the next minimal (maximal) eigenvalue. It is also noted that the above computation of eigenvalues is required not at all for Algorithm 2 since the coefficient matrices in (1.10) are diagonal and only one time for Algorithm 3 as the extremal eigenvalues are needed before the start of the outer iterations.

The number of inner iterations. Let Y^* be the true solution of (1.12). It follows from (2.1) that the iterative error after J steps takes the form

$$Y^{(J)} - Y^* = \left[\prod_{i=1}^{J}(g_i I - F)(F + f_i I)^{-1}\right](Y^{(0)} - Y^*)\left[\prod_{i=1}^{J}(f_i I - G)(G + g_i I)^{-1}\right].$$

This implies that we should choose parameters $\{f_i\}_{i=1}^J$ and $\{g_i\}_{i=1}^J$ such that the error $\|Y^{(J)} - Y^*\|$ is as small as possible, which results in the following min–max problem [29,33,34]

$$\min_{g_i, f_i} \max_{\substack{x\in\sigma(F)\\ y\in\sigma(G)}} \prod_{i=1}^{J}\left|\frac{(g_i - x)(f_i - y)}{(x + f_i)(y + g_i)}\right|. \tag{3.21}$$

Generally, it is very difficult to solve the min–max optimal problem (3.21). In practice, one may try to find the suboptimal solution by numerical methods such as the heuristics; see, e.g. [29,36]. However, for the subproblem (1.12) in (1.9)–(1.11), matrices F and G are real matrices and their eigenvalues locate in two different real intervals (see Theorem 4.2). Thus the Sylvester equation (1.12) falls into the so called "model problem" which was solved perfectly by Wachspress [33]. Here we follow his way to give the number of inner FDAI iterations.

For the prescribed accuracy ϵ which the error $\|Y^{(J)} - Y^*\|$ intend to attain, denote the minimal and maximal real eigenvalues of F (or G) by a and $\bar{a}$ (or b and $\bar{b}$), respectively. Let real constants $l = \sqrt{1 - l'^2}$ and $l' = d - 1 - \sqrt{d(d-2)}$ be the elliptic modulus and the complementary modulus with $d = \frac{2(a+\bar{b})(\bar{a}+b)}{(a+b)(\bar{a}+\bar{b})}$. Then the number of inner FADI iterations can be computed as

$$J = \left[\!\left[\frac{\epsilon}{\ln g}\right]\!\right], \tag{3.22}$$

where $g = \exp\left(-\frac{\pi^2}{\ln(4/l')}\right)$ and the symbol $[[x]]$ denotes the minimal integer greater than x.

Remark. It should be pointed out that the computation of the number of inner iterations J is associated with the scale n of the NARE (1.1). In fact for different n, the two pairs of extremal eigenvalues $(a, \bar{a})$ and $(b, \bar{b})$ are distinct. So from (3.22) we know that J is various with n. In this sense, the number of inner FADI iterations will be denoted as $J(n)$ to describe flops for each algorithm in the next subsection. But we will see from numerical experiments in Section 5 that the increase of J is not obvious even for remarkable rise of n. This is also the reason why we call Algorithms 1–3 are low complexity algorithms.

The optimal ADI parameters. The optimal ADI parameters also follow from Wachspress's method [33] and Jordan's bi-linear transformation [37]. Define the complete elliptic integral by

$$L = \int_0^1 \frac{d\theta}{\sqrt{(1 - \theta^2)(1 - l^2\theta^2)}}.$$

The optimal ADI parameters after J steps in the interval $[l', 1]$ are given by

$$\bar{\omega}_j = dn(r_j L, l), \quad j = 1, 2, \ldots, J, \tag{3.23}$$

where $r_j = \frac{2j-1}{2J}$ for $j = 1, 2, \ldots, J$ and $dn(\cdot, \cdot)$ is the elliptic function [38]. Then the optimal ADI parameters in intervals $(a, \bar{a})$ and $(b, \bar{b})$ are determined by

$$f_j = \frac{\beta_1\bar{\omega}_j - \beta_2}{-\beta_3\bar{\omega}_j + \beta_4} \quad \text{and} \quad g_j = \frac{\beta_1\bar{\omega}_j + \beta_2}{\beta_3\bar{\omega}_j + \beta_4}, \tag{3.24}$$

where

$$\beta_1 = \bar{a}\bar{d} - a(1+l'), \qquad \beta_2 = a(1+l') - \bar{a}\bar{d}l', \qquad \beta_3 = \bar{d} - 1 - l', \qquad \beta_4 = 1 + l' - \bar{d}l'$$

with the real constant $\bar{d} = \frac{2(a+\bar{b})}{\bar{a}+b}$.

Remark. In practical applications, the value of the elliptic function (3.23) at each r_jL ($j = 1, \ldots, J$) is usually approximated by

$$dn(r_jL, l) \approx \sqrt{l'}q'^{\frac{2r_j-1}{4}} \frac{1 + q'^{1-r_j} + q'^{1+r_j}}{1 + q'^{r_j} + q'^{2-r_j}}$$

with approximately complementary nome $q' = z(1+z^4)$ and $z = \frac{1-\sqrt{l}}{2(1+\sqrt{l})}$ [33]. Such an approximation is also conducted in our numerical experiments in Section 5.

3.4. Storage and flops for Algorithms 1–3

The storage and the computational flops of Algorithms 1–3 at the k-th outer iteration are listed in Table 3.1.

The column "STORAGE COST" in Table 3.1 records the approximative storage of each outer iteration for Algorithms 1–3. The last two columns give the computational flops of Algorithms 1–3 at each outer iteration. Specifically, "EIGENVALUES" represents the complexity of each algorithm for computing the two pairs of extremal eigenvalues of coefficient matrices at the k-th outer iteration. The "FADI" column summarizes the total complexity of each algorithm for solving the subproblem via FADI iteration, where $J_k(n)$ and $J(n)$ stand for the total number of inner iterations for Algorithms 1–3 at the k-th outer iteration.

For Algorithm 1, the computation of two pairs of extremal eigenvalues needs about $40K_kn$ flops, where K_k denotes the average number of iterations to obtain one of the extremal eigenvalues via the hybrid Newton–secant algorithm [28]. Note that such computation of eigenvalues is unnecessary for Algorithm 2 and is required only once for Algorithm 3 before the outer iteration, the related flops are of course zero.

It should be pointed out that the computational flops of Algorithms 1–3 should contain the cost to compute the number of inner iterations and the optimal ADI parameters. However, as we have got their explicit expressions in (3.22) and (3.24), the related computational flops can be neglected. On the other hand, although we see from Table 3.1 that the complexity of each outer iteration is about $O(J_k(n)n)$ for Algorithm 1 and $O(J(n)n)$ for Algorithms 2–3, our numerical results in Section 5 show $J(n) \ll n$ in particular for large scale problems. Hence Algorithms 1–3 possess lower complexity than the methods in [23,19].

4. Eigenvalue properties and convergence

4.1. Eigenvalue properties

In this subsection, we will give some eigenvalue properties of coefficient matrices of iterations (1.9)–(1.11). Without specification, we always assume that the subproblem (1.12) is solved accurately throughout this subsection. The following theorem collects some well-known results about the eigenvalues of a matrix related to NARE (1.1); see [39,8,40,3] and references therein for details.

Theorem 4.1. *Let*

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

with each $n \times n$ block defined by (1.2) and (1.3). Then M is a nonsingular M-matrix or a singular irreducible M-matrix. Let $H = \mathrm{Diag}(I, -I)M$. Then H has $2n$ real eigenvalues $\{-\lambda_n, \ldots, -\lambda_1, \tau_1, \ldots, \tau_n\}$ with the order

$$-\lambda_n < \cdots < -\lambda_2 < -\lambda_1 \le 0 \le \tau_1 < \tau_1 < \cdots < \tau_n.$$

Moreover, the minimal positive solution X^* of Eq. (1.1) is such that $\sigma(D - CX^*) = \{\tau_1, \ldots, \tau_n\}$ and $\sigma(A - X^*C) = \{\lambda_1, \ldots, \lambda_n\}$.

For Newton's method (1.9), we give the order of eigenvalues of coefficient matrices at each iterative step as follows.

Theorem 4.2. *For each $k = 1, 2, \ldots$, the coefficient matrices $\Delta - u^{(k)}q^T$ and $\Gamma - q(v^{(k)})^T$ at the k-th Newton step have distinct eigenvalues $\{\lambda_i^{(k)}\}_{i=1}^n$ and $\{\tau_i^{(k)}\}_{i=1}^n$, respectively. Moreover, they satisfy the following inequalities:*

$$\begin{cases} \lambda_1 < \lambda_1^{(k)} < \delta_1 < \lambda_2^{(k)} < \delta_2 < \cdots < \lambda_n^{(k)} < \delta_n, \\ \tau_1 < \tau_1^{(k)} < \gamma_1 < \tau_2^{(k)} < \gamma_2 < \cdots < \tau_n^{(k)} < \gamma_n, \end{cases}$$

where λ_1 and τ_1 are the minimal nonnegative eigenvalues defined by Theorem 4.1, and δ_i and γ_i, $i = 1, 2, \ldots, n$ are diagonal elements of matrices Δ and Γ respectively.

Proof. Let

$$\phi^{(k)}(x) \triangleq 1 + \sum_{i=1}^{n} \frac{u_i^{(k)} q_i}{x - \delta_i} = 0 \tag{4.1}$$

be the secular equation of $\Delta - u^{(k)} q^T - xI$ [41]. Then $\phi^{(k)}(x)$ is a strictly monotonically decreasing function with respect to x in the interval (δ_i, δ_{i+1}), $i = 1, \ldots, n-1$. Since for each $i = 1, \ldots, n-1$, $\lim_{x \to \delta_i^+} \phi^{(k)}(x) = +\infty$ and $\lim_{x \to \delta_{i+1}^-} \phi^{(k)}(x) = -\infty$, Eq. (4.1) has a unique real root in the open interval (δ_i, δ_{i+1}). On the other hand, as the Newton matrix sequence $\{X^{(k)}\}_{k=1}^{\infty}$ is strictly monotonically increasing and is convergent to the minimal positive solution X^* (see Theorem 2.1 in [16]), so $\{u^{(k)}\}_{k=1}^{\infty}$ monotonically converges to $u^* = X^* q + e$ by noticing (1.8). This together with the fact $\lambda_1 < \delta_1$ (see Lemma 2.1 in [3]) yields

$$\phi^{(k)}(\lambda_1) = 1 + \sum_{i=1}^{n} \frac{u_i^{(k)} q_i}{\lambda_1 - \delta_i} > 1 + \sum_{i=1}^{n} \frac{u_i^* q_i}{\lambda_1 - \delta_i} = 0,$$

where the last equality holds because the equation

$$1 + \sum_{i=1}^{n} \frac{u_i^* q_i}{x - \delta_i} = 0$$

is the secular equation of $\Delta - u^* q^T - xI$. So the last real root of $\phi^{(k)}(x)$ lies in the interval (λ_1, δ_1) due to $\lim_{x \to \delta_1^-} \phi^{(k)}(x) = -\infty$. This shows the relations among the eigenvalues of $\Delta - u^{(k)} q^T$ and the diagonal elements of Δ.

The relations among the eigenvalues of $\Gamma - q(v^{(k)})^T$ and the diagonal elements of Γ can be derived in a similar way. □

For FP1 iteration (1.10), it is clear that the eigenvalues of coefficient matrices are just the diagonal elements of matrices Δ and Γ. While for FP2 iteration (1.11), we have the following corollary analogous to Theorem 4.2.

Corollary 4.3. *The coefficient matrices $\Delta - eq^T$ and $\Gamma - qe^T$ of FP2 iteration (1.11) have distinct eigenvalues $\{\bar{\lambda}_i\}_{i=1}^n$ and $\{\bar{\tau}_i\}_{i=1}^n$, respectively. Moreover, they satisfy the following inequalities:*

$$\begin{cases} 0 < \bar{\lambda}_1 < \delta_1 < \bar{\lambda}_2 < \delta_2 < \cdots < \bar{\lambda}_n < \delta_n, \\ 0 < \bar{\tau}_1 < \gamma_1 < \bar{\tau}_2 < \gamma_2 < \cdots < \bar{\tau}_n < \gamma_n, \end{cases}$$

where δ_i and γ_i, $i = 1, 2, \ldots, n$ are diagonal elements of matrices Δ and Γ respectively.

We can further say another interesting property of the minimal eigenvalues of the coefficient matrices at each Newton iterative step.

Theorem 4.4. *For Newton's method (1.9), let $\{\lambda_1^{(k)}\}_{k=1}^{+\infty}$ and $\{\tau_1^{(k)}\}_{k=1}^{+\infty}$ be the minimal eigenvalues of the coefficient matrix sequences $\{\Delta - u^{(k)} q^T\}_{k=1}^{+\infty}$ and $\{\Gamma - q(v^{(k)})^T\}_{k=1}^{+\infty}$, respectively. Then we have*

$$\begin{cases} \lambda_1 < \cdots < \lambda_1^{(k+1)} < \lambda_1^{(k)} < \cdots < \lambda_1^{(1)} < \delta_1, \\ \tau_1 < \cdots < \tau_1^{(k+1)} < \tau_1^{(k)} < \cdots < \tau_1^{(1)} < \gamma_1, \end{cases}$$

where λ_1 and τ_1 are the same as those in Theorem 4.1.

Proof. Let $\phi^{(k)}(x)$ be defined by (4.1). By the strict monotonicity of the sequence $\{u^{(k)}\}_{k=1}^{+\infty}$, we immediately have

$$\begin{aligned} \phi^{(k)}(\lambda_1^{(k+1)}) &= 1 + \sum_{i=1}^{n} \frac{u_i^{(k)} q_i}{\lambda_1^{(k+1)} - \delta_i} > 1 + \sum_{i=1}^{n} \frac{u_i^{(k+1)} q_i}{\lambda_1^{(k+1)} - \delta_i} \\ &= \phi^{(k+1)}(\lambda_1^{(k+1)}) = 0 = \phi^{(k)}(\lambda_1^{(k)}). \end{aligned}$$

This implies $\lambda_1^{(k+1)} < \lambda_1^{(k)}$ because $\phi^{(k)}(x)$ is a strictly monotonically decreasing function in the open interval (λ_1, δ_1).

Similarly, we can derive the monotone property of the sequence of eigenvalues $\{\tau_1^{(k)}\}_{k=1}^{\infty}$. □

Theorem 4.4 shows the monotonicity and boundedness of the minimal eigenvalues of the coefficient matrices in Newton's method. Intuitively, we think that similar results should be true for the maximum (and all other) eigenvalues of the coefficient matrices. Our numerical experiments also show this possibility. However, at the moment, we could not prove it theoretically. We leave it as an open problem.

Table 5.1
Test results for $(\alpha, c) = (0.99, 0.01)$.

n	Method	$(\alpha, c) = (0.99, 0.01)$				
		ALG. 1	ALG. 2	ALG. 3	FNM	NBGS
32	CPU	0.00	0.00	0.015	0.265	0.015
	IT	11/3	10/4	11/3	3	3
	ERR	0.00e+00	1.77e−15	0.00e+00	4.85e−17	3.12e−17
	RES	6.24e−17	6.24e−17	6.24e−17	2.56e−16	2.28e−16
64	CPU	0.00	0.00	0.015	0.265	0.015
	IT	12/3	11/4	12/3	3	3
	ERR	0.00e+00	1.77e−15	0.00e+00	4.85e−17	3.12e−17
	RES	5.89e−17	5.89e−17	5.89e−17	2.56e−16	2.39e−16
128	CPU	0.00	0.00	0.015	0.078	0.000
	IT	13/3	13/4	13/3	3	3
	ERR	0.00e+00	1.81e−15	0.00e+00	6.24e−17	3.98e−17
	RES	5.89e−17	5.89e−17	5.89e−17	3.00e−16	2.51e−16
256	CPU	0.00	0.00	0.015	0.265	0.015
	IT	15/3	15/4	15/3	3	3
	ERR	0.00e+00	1.83e−15	0.00e+00	5.46e−17	3.64e−17
	RES	5.37e−17	5.37e−17	5.37e−17	3.85e−16	2.51e−16
512	CPU	0.00	0.00	0.015	0.921	0.015
	IT	17/3	16/4	17/3	3	3
	ERR	0.00e+00	1.84e−15	0.00e+00	4.64e−17	6.07e−17
	RES	5.11e−17	2.05e−16	5.11e−17	4.40e−16	2.53e−16
1024	CPU	0.031	0.015	0.015	3.571	0.075
	IT	18/3	18/4	18/3	3	3
	ERR	0.00e+00	1.84e−15	0.00e+00	5.70e−17	4.57e−17
	RES	5.11e−17	2.05e−16	5.11e−17	5.93e−16	2.51e−16
2048	CPU	0.046	0.015	0.046	14.53	0.390
	IT	20/3	20/4	20/3	3	3
	ERR	0.00e+00	1.84e−15	0.00e+00	5.67e−17	4.28e−17
	RES	4.68e−17	4.69e−17	4.68e−17	8.33e−16	8.33e−16
4096	CPU	0.125	0.031	0.171	66.95	2.51
	IT	21/3	21/4	21/3	3	3
	ERR	0.00e+00	1.84e−15	0.00e+00	5.54e−17	2.54e−16
	RES	2.34e−16	2.05e−16	2.34e−16	1.08e−15	1.08e−15

4.2. Convergence of Algorithms 1–3

In this subsection, we construct the convergence of Algorithms 1–3 with the solution of subproblem (1.12) approximated by the FADI iteration. We first define a vector $w^T = (u^T, v^T)$ and two matrices [42]

$$\mathcal{L}(w) = \begin{bmatrix} \mathrm{diag}(Pv) & 0 \\ 0 & \mathrm{diag}(Qu) \end{bmatrix} \quad \text{and} \quad \mathcal{K}(w) = \begin{bmatrix} \mathrm{diag}(Pv) & \mathrm{diag}(u)P \\ \mathrm{diag}(v)Q & \mathrm{diag}(Qu) \end{bmatrix}$$

with $P := (P)_{ij} = \frac{q_j}{\delta_i+\gamma_j}$ and $Q := (Q)_{ij} = \frac{q_j}{\delta_j+\gamma_i}$. It is clear that $\mathcal{L}(w)$ and $\mathcal{K}(w)$ are both linear operators of w and there exist $l_c > 0$ and $k_c > 0$ such that

$$\|\mathcal{L}(w)\| \le l_c\|w\| \quad \text{and} \quad \|\mathcal{K}(w)\| \le k_c\|w\|. \tag{4.2}$$

Then the Riccati equation (1.1) can be reformulated as

$$\mathcal{R}(w) = w - \mathcal{L}(w)w - e_{2n} = 0,$$

with $e_{2n}^T = (e^T, e^T)$. Note that the first and the second F-derivative of $\mathcal{R}$ at w are given by

$$\mathcal{R}'(w) = I_{2n} - \mathcal{K}(w) \quad \text{and} \quad \mathcal{R}''(w)(\cdot, w) = \mathcal{K}(w),$$

and the second F-derivative $\mathcal{R}''(w)(\cdot,\cdot)$ is independent of w. We now turn to the convergence of Algorithms 1–3.

For Algorithm 1, denote by $w_{NA}^{(k-1)}$ the $(k-1)$-th iteration point generated by Newton–FADI iteration and by $w_{NS}^{(k)}$ the exact solution of the k-th Newton's equation, i.e.

$$w_{NS}^{(k)} = w_{NA}^{(k-1)} - (\mathcal{R}'(w_{NA}^{(k-1)}))^{-1}\mathcal{R}(w_{NA}^{(k-1)}), \tag{4.3}$$

provided that $\mathcal{R}'(w_{NA}^{(k-1)})$ is nonsingular. Since the optimal ADI parameters can be derived, it follows from the inner FADI iteration process that the error between $w_{NA}^{(k)}$ and $w_{NS}^{(k)}$ can satisfy

$$\|w_{NA}^{(k)} - w_{NS}^{(k)}\| < \eta^{(k-1)}\|w_{NA}^{(k-1)} - w_{NS}^{(k)}\|, \tag{4.4}$$

Table 5.2
Test results for $(\alpha, c) = (0.5, 0.5)$.

n	Method	$(\alpha, c) = (0.5, 0.5)$				
		ALG. 1	ALG. 2	ALG. 3	FNM	NBGS
32	CPU	0.000	0.015	0.015	0.031	0.000
	IT	21/5	21/22	21/10	5	9
	ERR	4.85e−17	3.86e−15	1.85e−15	1.51e−16	6.43e−16
	RES	2.12e−16	1.85e−15	2.55e−16	2.60e−16	2.02e−16
64	CPU	0.000	0.015	0.015	0.046	0.000
	IT	23/5	23/22	23/10	5	9
	ERR	3.12e−17	3.81e−15	1.87e−15	1.21e−16	6.33e−16
	RES	2.52e−16	1.87e−15	2.92e−16	2.50e−16	2.18e−16
128	CPU	0.015	0.015	0.015	0.156	0.000
	IT	24/5	24/21	24/10	5	9
	ERR	4.39e−17	1.65e−14	1.86e−15	1.74e−16	6.30e−16
	RES	7.95e−16	7.89e−15	6.97e−16	3.12e−16	2.27e−16
256	CPU	0.015	0.015	0.031	0.375	0.015
	IT	26/4	26/21	26/10	4	8
	ERR	3.28e−14	1.65e−14	1.85e−15	3.29e−16	5.06e−14
	RES	1.07e−15	7.93e−15	8.72e−16	4.91e−16	1.00e−15
512	CPU	0.015	0.031	0.031	1.312	0.062
	IT	28/4	28/20	28/9	4	8
	ERR	3.28e−14	7.15e−14	5.90e−14	3.29e−14	5.06e−14
	RES	1.47e−15	3.42e−14	3.80e−15	4.91e−16	1.57e−15
1024	CPU	0.031	0.046	0.078	5.203	0.218
	IT	29/4	29/20	29/9	4	8
	ERR	3.28e−14	7.15e−14	5.90e−14	3.29e−14	5.06e−14
	RES	5.13e−15	3.42e−14	5.00e−15	6.30e−16	1.19e−15
2048	CPU	0.078	0.109	0.187	20.85	1.078
	IT	30/4	31/19	30/9	4	8
	ERR	3.29e−14	3.09e−13	5.90e−14	3.31e−14	5.06e−14
	RES	1.61e−14	1.48e−13	1.31e−14	1.20e−15	1.36e−15
4096	CPU	0.218	0.281	0.468	92.26	6.015
	IT	32/4	32/19	32/9	4	8
	ERR	3.28e−14	3.09e−13	5.90e−14	3.31e−14	5.06e−14
	RES	1.72e−14	1.47e−13	1.40e−14	1.58e−15	1.56e−15

where $\eta^{(k-1)}$ is the solution of the min–max problem in (3.21) and the RHS in (4.4) means the last outer iteration point $w_{NA}^{(k-1)}$ is just the initial iteration point at current inner iteration. Then the convergence of Algorithm 1 can be described as below.

Theorem 4.5. *Let $\mathcal{B}(w^*, \zeta)$ be a sufficient small ball of radius ζ about w^* and $\{w_{NA}^{(k)}\}_{k=1}^{\infty}$ be the sequence generated by Algorithm 1 to solve NARE (1.1) with $\alpha \neq 0$, $c \neq 1$. Given the initial iteration $w_{NA}^{(1)} \in \mathcal{B}(w^*, \zeta)$, if there exist a constant $C_F > 0$ such that $\eta^{(k)} \leq C_F \|w_{NA}^{(k)} - w^*\|$, then the sequence $\{w_{NA}^{(k)}\}_{k=1}^{\infty}$ generated by Algorithm 1 is quadratically convergent to the minimal positive solution w^*.*

Proof. We prove the theorem by showing that if the current iteration vector $w_{NA}^{(k)} \in \mathcal{B}(w^*, \zeta)$, then the next iteration vector satisfies $\|w_{NA}^{(k+1)} - w^*\| < \|w_{NA}^{(k)} - w^*\|$. In fact, it follows from [18] that $\mathcal{R}'(w^*)$ is a nonsingular matrix and we can choose a sufficiently small ζ such that

$$\|I - \mathcal{R}'(w_{NA}^{(k)})(\mathcal{R}'(w^*))^{-1}\| \leq \|(\mathcal{R}'(w^*))^{-1}\| \cdot \|\mathcal{K}(w_{NA}^{(k)} - w^*)\| \leq \zeta k_c \|(\mathcal{R}'(w^*))^{-1}\| < 1/2.$$

Then $\mathcal{R}'(w_{NA}^{(k)})$ is nonsingular by Banach lemma (see, e.g. [43]) and

$$\|(\mathcal{R}'(w_{NA}^{(k)}))^{-1}\| \leq \frac{\|(\mathcal{R}'(w^*))^{-1}\|}{1 - \|I - \mathcal{R}'(w_{NA}^{(k)})(\mathcal{R}'(w^*))^{-1}\|} \leq 2\|(\mathcal{R}'(w^*))^{-1}\|.$$

Therefore, the next Newton step is well defined and its exact solution $w_{NS}^{(k+1)}$ satisfies

$$\begin{aligned} \|w_{NS}^{(k+1)} - w^*\| &\leq \|(\mathcal{R}'(w_{NA}^{(k)}))^{-1}\| \cdot \|\mathcal{R}'(w_{NA}^{(k)})(w_{NA}^{(k)} - w^*) - \mathcal{R}(w_{NA}^{(k)})\| \\ &\leq \frac{1}{2}\|(\mathcal{R}'(w_{NA}^{(k)}))^{-1}\| \cdot \|\mathcal{K}(w_{NA}^{(k)} - w^*)(w_{NA}^{(k)} - w^*)\| \\ &\leq C_N \|w_{NA}^{(k)} - w^*\|^2 \end{aligned} \tag{4.5}$$

Table 5.3
Test results for $(\alpha, c) = (0.01, 0.99)$.

n	Method	$(\alpha, c) = (0.01, 0.99)$				
		ALG. 1	ALG. 2	ALG. 3	FNM	NBGS
32	CPU	0.015	0.140	0.156	0.031	0.000
	IT	27/8	22/269	22/131	8	67
	ERR	3.40e−16	6.46e−15	6.60e−15	1.39e−15	5.25e−15
	RES	1.04e−15	1.60e−14	1.07e−15	5.16e−16	2.31e−15
64	CPU	0.015	0.171	0.187	0.062	0.000
	IT	29/8	23/262	24/128	8	65
	ERR	2.54e−16	1.34e−14	1.26e−14	1.77e−15	1.31e−14
	RES	1.51e−15	3.29e−14	2.01e−14	7.60e−16	5.74e−15
128	CPU	0.015	0.203	0.234	0.203	0.000
	IT	31/8	25/255	26/125	8	64
	ERR	5.09e−16	2.82e−14	2.47e−14	2.62e−15	2.07e−14
	RES	2.38e−15	6.97e−14	3.74e−14	9.50e−16	9.40e−15
256	CPU	0.031	0.250	0.328	0.671	0.062
	IT	32/8	27/249	28/122	8	62
	ERR	2.15e−16	5.32e−14	4.81e−14	3.69e−15	5.19e−14
	RES	8.77e−15	1.31e−13	6.83e−14	1.37e−15	2.24e−14
512	CPU	0.031	0.390	0.484	2.503	0.437
	IT	34/8	29/242	29/119	8	61
	ERR	1.00e−15	1.11e−13	9.37e−14	4.98e−15	8.21e−14
	RES	1.08e−14	2.77e−13	1.56e−15	2.16e−15	3.61e−14
1024	CPU	0.062	0.640	0.921	9.730	1.640
	IT	35/8	30/236	30/115	8	59
	ERR	2.07e−15	2.10e−13	2.27e−13	7.12e−15	2.05e−13
	RES	2.85e−14	5.21e−13	4.12e−13	3.10e−15	8.95e−14
2048	CPU	0.187	1.359	2.390	38.54	7.781
	IT	36/8	32/229	32/112	8	58
	ERR	3.31e−16	4.42e−13	4.41e−13	9.80e−15	3.24e−13
	RES	7.77e−14	1.09e−12	6.81e−13	4.82e−15	1.14e−13
4096	CPU	0.593	3.265	5.500	162.71	42.12
	IT	37/8	33/223	33/109	8	56
	ERR	7.77e−16	8.34e−13	8.58e−13	1.25e−14	8.12e−13
	RES	2.02e−13	2.06e−12	1.42e−12	3.21e−14	3.52e−13

with $C_N = 3k_c\|(\mathcal{R}'(w^*))^{-1}\|$. By noting (4.4) and the assumption in the theorem, we have

$$\begin{aligned}\|w_{NA}^{(k+1)} - w^*\| &\le \|w_{NA}^{(k+1)} - w_{NS}^{(k+1)}\| + \|w_{NS}^{(k+1)} - w^*\| \\ &\le \eta^{(k)}\|w_{NA}^{(k)} - w_{NS}^{(k+1)}\| + C_N\|w_{NA}^{(k)} - w^*\|^2 \\ &\le (C_F + C_F C_N \zeta + C_N)\|w_{NA}^{(k)} - w^*\|^2. \end{aligned} \tag{4.6}$$

Now reduce ζ if needed such that $(C_F + C_F C_N \zeta + C_N)\zeta < 1$. Then we have

$$\|w_{NA}^{(k+1)} - w^*\| \le (C_F + C_F C_N \zeta + C_N)\zeta\|w_{NA}^{(k)} - w^*\| < \|w_{NA}^{(k)} - w^*\|,$$

which implies the sequence $\{w_{NA}^{(k)}\}_{k=1}^{\infty}$ is convergent as $w_{NA}^{(k+1)} \in \mathcal{B}(w^*, \zeta)$. Moreover, we know from (4.6) that the convergence is quadratic. □

As for the convergence of Algorithms 2 and 3, we similarly denote $w_{FA1}^{(k)}$ and $w_{FA2}^{(k)}$ the k-th iteration point generated by Algorithms 2 and 3, respectively. Again, $w_{FS1}^{(k)}$ and $w_{FS2}^{(k)}$ represent the exact solution from the k-th FP1 iteration equation

$$w_{FS1}^{(k)} = \mathcal{L}(w_{FA1}^{(k-1)}) + e_{2n}$$

and FP2 iteration equation

$$(I - \mathcal{K}(e_{2n}))w_{FS2}^{(k)} = (\mathcal{L}(w_{FA2}^{(k-1)}) - \mathcal{K}(e_{2n}))w_{FA2}^{(k-1)} + e_{2n},$$

respectively. Let the inner FADI iterations in Algorithms 2 and 3 satisfy

$$\|w_{FAi}^{(k)} - w_{FSi}^{(k)}\| < \eta^{(k-1)}\|w_{FAi}^{(k-1)} - w_{FSi}^{(k)}\|, \quad i = 1, 2, \tag{4.7}$$

where $\eta^{(k-1)}$ is the solution of the min–max problem in (3.21). We then have the following convergence theorem of Algorithms 2 and 3. As the proof is analogous to that of Theorem 4.5, we omit it.

Table 5.4
Test results for $(\alpha, c) = (10^{-4}, 1 - 10^{-4})$.

n	Method	$(\alpha, c) = (10^{-4}, 1 - 10^{-4})$				
		ALG. 1	ALG. 2	ALG. 3	FNM	NBGS
32	CPU	0.062	1.281	1.437	0.046	0.031
	IT	35/19	22/2389	22/1222	13	556
	ERR	3.33e−15	7.07e−15	6.97e−15	6.51e−15	6.91e−15
	RES	1.29e−15	1.94e−14	1.31e−14	7.76e−16	3.98e−15
64	CPU	0.015	1.359	1.687	0.109	0.031
	IT	37/11	23/2320	24/1187	11	541
	ERR	5.52e−15	1.41e−14	1.41e−14	1.20e−14	1.36e−14
	RES	1.82e−15	3.80e−14	2.58e−14	1.23e−15	8.13e−15
128	CPU	0.031	1.650	2.093	0.265	0.125
	IT	38/12	25/2251	26/1153	11	526
	ERR	1.95e−14	2.84e−14	2.82e−14	1.63e−14	2.76e−14
	RES	2.21e−15	7.62e−14	4.87e−14	1.86e−15	1.63e−16
256	CPU	0.046	2.218	3.000	0.921	0.546
	IT	39/11	27/2182	28/1119	11	511
	ERR	4.74e−14	5.67e−14	5.59e−15	1.89e−14	5.51e−14
	RES	4.81e−15	1.52e−13	9.20e−14	2.92e−15	3.25e−14
512	CPU	0.062	3.421	4.546	3.343	3.421
	IT	41/11	29/2113	29/1084	11	496
	ERR	2.35e−14	1.13e−13	1.13e−13	1.16e−14	1.10e−13
	RES	5.79e−15	3.06e−13	2.11e−13	4.08e−15	6.35e−14
1024	CPU	0.140	5.484	8.328	13.37	13.39
	IT	42/11	30/2044	30/1050	11	481
	ERR	8.07e−14	2.26e−13	2.25e−13	3.30e−14	2.20e−13
	RES	9.42e−15	6.12e−13	4.83e−13	6.04e−15	1.27e−13
2048	CPU	0.281	11.64	21.37	53.24	62.28
	IT	43/11	32/1975	32/1016	11	466
	ERR	3.53e−14	4.54e−13	4.48e−13	2.48e−14	4.39e−13
	RES	2.16e−14	1.22e−12	8.14e−13	8.60e−15	2.55e−13
4096	CPU	0.796	27.21	48.37	215.75	336.78
	IT	44/11	33/1906	33/982	11	451
	ERR	2.35e−14	9.07e−13	8.91e−13	4.16e−14	8.74e−13
	RES	4.16e−14	2.45e−12	1.72e−12	1.11e−14	5.08e−13

Theorem 4.6. *Let $\{w_{FA1}^{(k)}\}_{k=1}^{\infty}$ and $\{w_{FA2}^{(k)}\}_{k=1}^{\infty}$ be two sequences generated by Algorithms 2 and 3 respectively to solve NARE (1.1) with $\alpha \neq 0$, $c \neq 1$.*

(i) *Given the initial iteration $w_{FP1}^{(1)} \in \mathcal{B}(w^*, \zeta)$. If there exists a constant $C_{F1} > 0$ such that $\eta^{(k)} \leq C_{F1}\|w_{FA1}^{(k)} - w^*\|$ and $\bar{\eta} + \bar{\eta} l_c + l_c < 1$ with $\bar{\eta} = \sup_k\{\eta^{(k)}\}$ and l_c defined in (4.2), then the sequence $\{w_{FA1}^{(k)}\}_{k=1}^{\infty}$ generated by Algorithm 2 is linearly convergent to the minimal positive solution w^*.*

(ii) *Given the initial iteration $w_{FP2}^{(1)} \in \mathcal{B}(w^*, \zeta)$. If there exists a constant $C_{F2} > 0$ such that $\eta^{(k)} \leq C_{F2}\|w_{FA2}^{(k)} - w^*\|$ and $\bar{\eta} + \bar{\eta}\bar{C} + \bar{C} < 1$ with $\bar{\eta} = \sup_k\{\eta^{(k)}\}$ and $\bar{C} = \|(I_{2n} - \mathcal{K}(e_{2n}))^{-1}\|(\|\mathcal{L}(w^*)\| + l_c(\|(w^*)\| + \zeta) + \|\mathcal{K}(e_{2n})\|)$, then the sequence $\{w_{FA2}^{(k)}\}_{k=1}^{\infty}$ generated by Algorithm 3 is linearly convergent to the minimal positive solution w^*.*

5. Numerical experiments

The purpose of this section is to show the effectiveness of Algorithms 1–3 in computation. Our algorithms were coded by MATLAB 7.1 and were run on a PC with 1.80 GHz AMD Sempron 3000+ processor and 512M RAM.

The test problem comes from [23]. The scalars c_i and ω_i, $i = 1, 2, \ldots, n$, are given by the Gauss–Legendre quadrature formula on the interval $[0, 1]$. Specifically, we first divide the interval $[0, 1]$ into $n/4$ intervals of equal length and then apply the 4-nodes Gauss–Legendre quadrature formula to each subinterval; see [19,16] for details.

The stopping criterion of iterations is the relative residual error

$$\mathrm{ERR} = \max\left(\frac{\|u^{(k)} - u^{(k-1)}\|_1}{\|u^{(k)}\|_1}, \quad \frac{\|v^{(k)} - v^{(k-1)}\|_1}{\|v^{(k)}\|_1}\right) \leq n \cdot \mathrm{eps} \tag{5.1}$$

or the total number of iterations is greater than 100 for Algorithm 1, or 20 000 for Algorithms 2 and 3, where $\mathrm{eps} = 2^{-53} \approx 1.1 \times 10^{-16}$ is the relative accuracy of the floating point operation of the computer.

Table 5.5
Test results for $(\alpha, c) = (10^{-6}, 1 - 10^{-6})$.

n	Method	$(\alpha, c) = (10^{-6}, 1 - 10^{-6})$				
		ALG. 1	ALG. 2	ALG. 3	FNM	NBGS
32	CPU	0.250	10.23	11.78	0.062	0.156
	IT	43/100	24/19 439	22/10 060	14	4545
	ERR	9.27e−13	7.09e−15	7.04e−15	4.45e−15	7.08e−15
	RES	1.11e−13	1.95e−14	1.37e−14	1.60e−15	4.31e−15
64	CPU	0.250	10.92	13.71	0.328	0.250
	IT	44/100	23/18 745	24/9710	37	4396
	ERR	4.33e−13	1.41e−14	1.42e−14	8.75e−15	1.41e−14
	RES	1.33e−13	3.84e−14	2.63e−14	2.20e−15	8.30e−15
128	CPU	0.312	13.31	17.07	0.578	1.000
	IT	45/100	25/18 056	26/9366	24	4246
	ERR	2.96e−13	2.83e−14	2.83e−14	1.67e−14	2.81e−14
	RES	1.18e−13	7.73e−14	5.06e−14	2.63e−15	1.69e−14
256	CPU	0.421	17.67	23.35	1.343	4.375
	IT	47/100	27/17 360	28/9019	16	4095
	ERR	1.75e−13	5.68e−14	5.68e−14	4.35e−14	5.66e−14
	RES	1.37e−13	1.54e−13	9.45e−14	4.27e−15	3.39e−14
512	CPU	0.625	26.64	35.07	5.171	27.25
	IT	48/100	29/16 667	29/8674	17	3945
	ERR	2.08e−13	1.13e−13	1.13e−13	7.97e−14	1.13e−13
	RES	1.95e−13	3.10e−13	2.15e−13	6.01e−15	6.83e−14
1024	CPU	0.921	43.10	65.10	18.64	107.35
	IT	49/79	30/15 975	30/8327	15	3795
	ERR	2.20e−13	2.27e−13	2.27e−13	5.83e−14	2.26e−13
	RES	2.11e−13	6.79e−13	4.93e−13	9.15e−15	1.35e−13
2048	CPU	0.421	90.68	169.59	67.45	492.70
	IT	49/15	32/15 283	32/7981	14	3645
	ERR	4.28e−13	4.54e−13	4.54e−13	3.80e−13	4.53e−13
	RES	1.94e−13	1.22e−12	8.35e−13	1.20e−14	2.69e−13
4096	CPU	1.234	216.14	378.48	262.54	2610.03
	IT	50/19	33/14 589	33/7635	18	3495
	ERR	8.38e−13	9.08e−13	9.08e−13	4.25e−13	9.06e−13
	RES	2.75e−13	2.47e−12	1.77e−12	5.33e−14	5.39e−13

We compared the performances of Algorithms 1–3 with that of the quite recent developed nonlinear block Gauss–Seidel (NBGS) algorithm [23] and the fast Newton method (FNM) [19] for problems with different sizes and different parameters α and c. All these methods were tested in CPU time, total number iterations as well as accuracy.

The obtained results are listed in Tables 5.1–5.5 where the "n" column gives the sizes of the problem, the "CPU" row denotes the CPU time used in seconds, the "IT" row represents the maximal inner iterations and the total number of outer iterations. The scalars in each "IT" line have the meaning "the maximal inner iteration/ the total number of the outer iterations". The "ERR" row in the tables represents the final value of the relative errors defined by (5.1) and, the "RES" row reports the relative residual errors of NARE (1.1) defined by (see also in [6])

$$\mathrm{RES} = \frac{\|\Delta\bar{X} + \bar{X}\Gamma - (\bar{X}q + e)(\bar{X}^T q + e)^T\|_1}{\|(\bar{X}q + e)(\bar{X}^T q + e)^T\|_1},$$

where $(\bar{X})_{i,j} = \frac{u_i v_j}{\delta_i + \gamma_j}$ is the computed minimal positive solution.

We see from Tables 5.1–5.5 that in most cases, all the methods stopped regularly and the inequality ERR $\leq n \cdot$ eps was finally satisfied. Tables 5.1–5.2 shows that when the problem is far from the critical case, performances of Algorithms 1–3 were not much different. Specially for middle and large scale problems, Algorithms 1–3 performed better than NBGS and FNM in CPU time. The results in Tables 5.3–5.5 show that when the problem is close to the critical case, Algorithm 1 performed better than Algorithms 2 and 3 both in CPU time and in the total outer number of iterations. Such a feature is more evident as NARE (1.1) approaches nearer to the critical case. On the other hand, for the problem with middle and larger sizes, all of Algorithms 1–3 needed less CPU time to attain the prescribed accuracy than the NGBS and the FNM did. However, when the problem is very close to the critical case, we also observe from Table 5.5 that Algorithm 1 failed to attain the prescribed accuracy in 100 steps for the small scale problems. In our numerical experiments, we proceeded up the number of outer iterations to 500 and found that the relative errors were not improved much. This seems to show that Algorithm 1 may be more appropriate for solving the large size problems near to the critical cases.

Acknowledgments

Part of this work was based on first author's communication with Dr. Eugene L. Wachspress. The authors are grateful to Dr. Yong-Hua Gao and Federico G. Poloni for their help in codes and would like to thank the editor and two anonymous referees for their helpful comments to improve the original paper.

References

[1] J. Juang, Existence of algebraic matrix Riccati equations arising in transport theory, Linear Algebra Appl. 230 (1995) 89–100.
[2] J. Juang, I.-D. Chen, Iterative solution for a certain class of algebraic matrix Riccati equations arising in transport theory, Transport Theory Statist. Phys. 22 (1993) 65–80.
[3] J. Juang, W.-W. Lin, Nonsymmetric algebraic Riccati equations and Hamiltonian-like matrices, SIAM J. Matrix Anal. Appl. 20 (1998) 228–243.
[4] L. Bao, Y.-Q. Lin, Y.-M. Wei, A modified simple iterative method for nonsymmetric algebraic Riccati equations arising in transport theory, Appl. Math. Comput. 181 (2006) 1499–1504.
[5] D.A. Bini, B. Meini, F. Poloni, From algebraic Riccati equations to unilateral quadratic matrix equations: old and new algorithms, in: Proceeding of Numerical Methods for Structured Markov Chains, Dagstuhl Seminar Proceedings, IBFI, Schloss Dagstuhl, Germany 2008.
[6] D.A. Bini, B. Meini, F. Poloni, Fast solution of a certain Riccati equation through Cauchy-like matrices, Electron. Trans. Numer. Anal. 33 (2009) 84–104.
[7] Y.-H. Gao, Z.-Z. Bai, On inexact Newton methods based on doubling iteration scheme for nonsymmetric algebraic Riccati equations, Numer. Linear Algebra Appl. 18 (2011) 325–341.
[8] C.-H. Guo, Nonsymmetric algebraic Riccati equations and Wiener–Hopf factorization for M-matrices, SIAM J. Matrix Anal. Appl. 23 (2001) 225–242.
[9] X.-X. Guo, W.-W. Lin, S.-F. Xu, A structure-preserving doubling algorithm for nonsymmetric algebraic Riccati equation, Numer. Math. 103 (2006) 393–412.
[10] J. Juang, Z.-T. Lin, Convergence of an iterative technique for algebraic matrix Riccati equations and applications to transport theory, Transport Theory Statist. Phys. 21 (1992) 87–100.
[11] J. Juang, H.-C. Lin, P. Nelson, Global existence, aymptotics and uniqueness for the reflection kernel of the angularly shifted transport equation, Math. Models Methods Appl. Sci. 5 (1995) 239–251.
[12] P. Nelson, Convergence of a certain monotone iteration in the reflection matrix for a nonmultiplying half-space, Transport Theory Statist. Phys. 13 (1984) 97–106.
[13] B. Yu, N. Dong, A Structure-preserving doubling algorithm for quadratic matrix equations arising form damped mass-spring system, Adv. Model. Optim. 12 (2010) 85–100.
[14] B. Yu, N. Dong, Q. Tang, F.-H. Wen, On iterative methods for the quadratic matrix equation with M-matrix, Appl. Math. Comput. 218 (2011) 3303–3310.
[15] B. Yu, D.-H. Li, N. Dong, Convergence of the cyclic reduction algorithm for a class of weakly overdamped quadratics, J. Comput. Math. 30 (2012) 139–156.
[16] C.-H. Guo, A.J. Laub, On the iterative solution of a class of nonsymmetric algebraic Riccati equations, SIAM J. Matrix Anal. Appl. 22 (2000) 376–391.
[17] R.H. Bartels, G.W. Stewart, Solution of the matrix equation $AX + XB = C$, Comm. ACM. 15 (1972) 820–826.
[18] L.-Z. Lu, Newton iterations for a nonsymmetric algebraic Riccati equation, Numer. Linear Algebra Appl. 12 (2005) 191–200.
[19] D.A. Bini, B. Iannazzo, F. Poloni, A fast Newton's method for a nonsymmetric algebraic Riccati equation, SIAM J. Matrix Anal. Appl. 30 (2008) 276–290.
[20] I. Gohberg, T. Kailath, V. Olshevsky, Fast Gaussian elimination with partial pivoting for matrices with displacement structure, Math. Comp. 64 (1995) 1557–1576.
[21] R. Varga, Matrix Iterative Analysis, second ed., Springer-Verlag, Berlin, Heidelberg, 2000.
[22] L.-Z. Lu, Solution form and simple iteration of a nonsymmetric algebraic Riccati equation arising in transport theory, SIAM J. Matrix Anal. Appl. 26 (2005) 679–685.
[23] Z.-Z. Bai, Y.-H. Gao, L.-Z. Lu, Fast iterative schemes for nonsymmetric algebraic Riccati equations arising from transport theory, SIAM J. Sci. Comput. 30 (2008) 804–818.
[24] Z.-Z. Bai, X.-X. Guo, S.-F. Xu, Alternately linearized implicit iteration methods for the minimal nonnegative solutions of the nonsymmetric algebraic Riccati equations, Numer. Linear Algebra Appl. 13 (2006) 655–674.
[25] Z.-Z. Bai, G.H. Golub, M.K. Ng, Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems, SIAM J. Matrix Anal. Appl. 24 (2003) 603–626.
[26] Z.-Z. Bai, On Hermitian and skew-Hermitian splitting iteration methods for continuous Sylvester equations, J. Comput. Math. 29 (2011) 185–198.
[27] D.A. Bini, L. Gemignani, V.Y. Pan, Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations, Numer. Math. 100 (2005) 373–408.
[28] V. Mehrmann, H.-G. Xu, Explicit solutions for a Riccati equation from transport theory, SIAM J. Matrix Anal. Appl. 30 (2008) 1339–1357.
[29] P. Benner, R.-C. Li, N. Truhar, On the ADI method for Sylvester equations, J. Comput. Appl. Math. 233 (2009) 1035–1045.
[30] J.-R. Li, J. White, Low-rank solution of Lyapunov equations, SIAM J. Matrix Anal. Appl. 24 (2002) 260–280.
[31] P. Benner, H. Mena, J. Saak, On the parameter selection problem in the Newton–ADI iteration for large-scale Riccati equations, Electron. Trans. Numer. Anal. 29 (2008) 136–149.
[32] P. Benner, J. Saak, A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations, Preprint SPP1253-090, DFG Priority Programme Optimization with Partial Differential Equations, 2010. URL: http://www.am.unierlangen.de/home/spp1253/wiki/index.php/Preprints.
[33] E.L. Wachspress, The ADI Model Problem, Windsor, CA, 1995.
[34] E.L. Wachspress, ADI iteration parameters for solving Lyapunov and Sylvester equations, Technical Report, March 2009.
[35] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[36] T. Penzl, A cyclic low-rank smith method for large sparse Lyapunov equations, SIAM J. Sci. Comput. 21 (2000) 1401–1418.
[37] E.L. Wachspress, Optimum alternating-direction-implicit iteration parameters for a model problem, J. Soc. Ind. Appl. Math. 10 (1962) 339–350.
[38] N.I. Akhiezer, Elements of the Theory of Elliptic Functions, American Mathematical Society, Providence, RI, 1990.
[39] C.-H. Guo, B. Iannazzo, B. Meini, On the doubling algorithm for a (shifted) nonsymmetric algebraic Riccati equations, SIAM J. Matrix Anal. Appl. 29 (2007) 1083–1100.
[40] C.-H. Guo, N.J. Higham, Iterative solution of a nonsymmetric algebraic Riccati equation, SIAM J. Matrix Anal. Appl. 29 (2007) 396–412.
[41] R.-C. Li, Solving secular equations stably and efficiently, Technical Report, Department of Mathematics, University of California, Berkeley, CA, LAPACK Working Note 89, 1993.
[42] C.-H. Guo, W.-W. Lin, Convergence rates of some iterative methods for nonsymmetric algebraic Riccati equations arising in transport theory, Linear Algebra Appl. 432 (2010) 283–291.
[43] C.T. Kelley, Iterative Methods for Linear and Nonlinear Equations, Society for Industrial and Applied Mathematics, Philadelphia, 1995.