Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2009, Article ID 783920, 17 pages
doi:10.1155/2009/783920

Research Article

Two-Step Relaxation Newton Method for Nonsymmetric Algebraic Riccati Equations Arising from Transport Theory

Shulin Wu[1] and Chengming Huang[2]

[1] *School of Science, Sichuan University of Science and Engineering, Zigong, Sichuan 643000, China*
[2] *School of Mathematics and Statistics, Huazhong University of Science and Technology, Wuhan 430074, China*

Correspondence should be addressed to Shulin Wu, wushulin_ylp@163.com

Received 26 February 2009; Accepted 20 August 2009

Recommended by Alois Steindl

We propose a new idea to construct an effective algorithm to compute the minimal positive solution of the nonsymmetric algebraic Riccati equations arising from transport theory. For a class of these equations, an important feature is that the minimal positive solution can be obtained by computing the minimal positive solution of a couple of fixed-point equations with vector form. Based on the fixed-point vector equations, we introduce a new algorithm, namely, *two-step relaxation Newton*, derived by combining two different relaxation Newton methods to compute the minimal positive solution. The monotone convergence of the solution sequence generated by this new algorithm is established. Numerical results are given to show the advantages of the new algorithm for the nonsymmetric algebraic Riccati equations in vector form.

1. Introduction

In this paper, we are interested in iteratively solving the algebraic Riccati equation arising from transport theory (see, e.g., [1–5] and references therein):

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

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

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

Here $e = (1, 1, \ldots, 1)^T$ and $q = (q_1, q_2, \ldots, q_n)^T$ with $q_i = c_i/2\omega_i$ and

$$\begin{aligned} \Delta &= \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_n), \qquad \delta_i = \frac{1}{c\omega_i(1+\alpha)}, \\ D &= \operatorname{diag}(d_1, d_2, \ldots, d_n), \qquad d_i = \frac{1}{c\omega_i(1-\alpha)}. \end{aligned} \tag{1.3}$$

The parameters c and α satisfy $0 < c \leq 1$, $0 \leq \alpha < 1$, and $\{c_i\}_{i=1}^n$ and $\{\omega\}_{i=1}^n$ are sets of the composite Gauss-Legendre weights and nodes, respectively, on the interval $[0, 1]$ satisfying

$$0 < \omega_n < \omega_{n-1} < \cdots < \omega_1, \qquad c_i > 0 \quad (i = 1, 2, \ldots, n), \qquad \sum_{i=1}^{n} c_i = 1; \tag{1.4}$$

see, for example, [4] for details. Clearly, it holds that

$$\begin{aligned} &0 < \delta_1 < \delta_2 < \cdots < \delta_n, \qquad 0 < d_1 < d_2 < \cdots < d_n, \\ &d_i = \delta_i \quad \text{for } \alpha = 0, \qquad d_i > \delta_i \quad \text{for } \alpha \neq 0, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{1.5}$$

It has been shown that problem (1.1) has positive solution in the sense of componentwise; see [2, 4] for details. Since only the minimal positive solution is physically meaningful, the research in the field of iteration methods centers around the computation of the minimal positive solution of (1.1); see, for example, [3, 5–12]. For the discussion on more general nonsymmetric algebraic Riccati equations arising in real world, we refer the interested reader to [13–17].

In the seminal paper by Lu [9], it was shown that the solution X of (1.1) can be written as

$$X = T \circ \left(uv^T\right) = \left(uv^T\right) \circ T, \tag{1.6}$$

where $\circ$ denotes the Hadamard product, T is the matrix with elements $T_{i,j} = 1/(\delta_i + d_j)$, and u, v are two vectors satisfying the following vector equations:

$$f(u, v) = \begin{pmatrix} u - [u \circ (Pv) + e] \\ v - [v \circ (Qu) + e] \end{pmatrix} = 0, \tag{1.7}$$

where $P = [P_{i,j}] = q_j/(\delta_i + d_j)$ and $Q = [Q_{i,j}] = q_j/(\delta_j + d_i)$.

Let $w = (u^T, v^T)^T \in \mathbb{R}^{2n}$ with $w_i = u_i$ and $w_{n+i} = v_i$, $i = 1, 2, \ldots, n$, and let $g(w) = (g_1(w), g_2(w), \ldots, g_{2n}(w))^T$ with

$$g_i(w) = \begin{cases} \displaystyle\sum_{j=1}^{n} P_{i,j} w_{n+j} & \text{for } 1 \leq i \leq n, \\ \displaystyle\sum_{j=1}^{n} Q_{i-n,j} w_j & \text{for } n+1 \leq i \leq 2n. \end{cases} \tag{1.8}$$

Then the vector equations (1.7) can be uniformly rewritten as

$$w = w \circ g(w) + e. \tag{1.9}$$

Based on the vector equations (1.7), Lu [9] investigated the *simple iteration* (SI) method to compute the minimal positive solution as follows:

$$\begin{aligned} u_{k+1} &= u_k \circ (Pv_k) + e, \\ v_{k+1} &= v_k \circ (Qu_k) + e, \\ v_0 &= 0, \qquad u_0 = 0. \end{aligned} \tag{1.10}$$

It was shown that the solution sequence $\{u_k, v_k\}$ generated by (1.10) converges monotonically to the minimal positive solution of (1.7). We note that at each step the SI method costs about $4n^2$ flops (for the definition of *flops*, see, e.g., [18]), while the *Gauss-Jacobi* (GJ) scheme defined by Juang [2]

$$X_{k+1} = T \circ (X_k q + e)\left(X_k^T q + e\right)^T \tag{1.11}$$

costs about $6n^2$ flops. Hence, the SI method is more efficient than the GJ method.

In (1.10), we note that before computing v_{k+1}, we have obtained u_{k+1}, which should be a better approximation to u than u_k. Based on this consideration, Bao et al. [7] constructed the modified simple iteration method as

$$\begin{aligned} u_{k+1} &= u_k \circ (Pv_k) + e, \\ v_{k+1} &= v_k \circ (Qu_{k+1}) + e, \\ v_0 &= 0, \qquad u_0 = 0. \end{aligned} \tag{1.12}$$

It was shown theoretically and numerically that the modified simple iteration procedure (1.12) is more efficient than the original one (1.10).

Recently, the so-called nonlinear splitting iteration methods defined as

$$\begin{aligned} u_{k+1} &= u_{k+1} \circ [Pv_k] + e, \\ v_{k+1} &= v_{k+1} \circ [Qu_k] + e, \end{aligned} \tag{1.13a}$$

$$\begin{aligned} u_{k+1} &= u_{k+1} \circ [Pv_k] + e, \\ v_{k+1} &= v_{k+1} \circ [Qu_{k+1}] + e \end{aligned} \tag{1.13b}$$

were investigated by Bai et al. [6] and independently by Lin [19]. In [6], methods (1.13a) and (1.13b) were called *nonlinear block Jacobi* (NBJ) iteration method and *nonlinear block Gauss-Seidel* (NBGS) iteration method, respectively, and it was shown that the solution sequence $\{u_k, v_k\}$ generated by the NBJ method and the NBGS method converges monotonically to the

minimal positive solution of (1.7). Moreover, numerical experiments given in [6, 19] show that the convergence speed of these two methods is higher than that of the SI method (1.10).

In this paper, based on the fixed-point equations (1.7), we present an accelerated version of the NBJ scheme, namely, two-step relaxation Newton method, to compute the minimal positive solution. The construction of the new method is based on the relaxation Newton method introduced by Wu et al. [20, 21]. At each iteration, the new algorithm is composed of two-steps: we use the NBJ method as a simple relaxation Newton method to obtain a coarse approximation of the solution of (1.7), and then with this coarse approximation at hand we use a relative complicated relaxation Newton method to get a finer approximation. It is shown that the solution sequence generated by this method converges monotonically to the minimal positive solution of (1.7). We also prove that the new method is more efficient than the NBJ method and its two-step version.

The remainder of this paper is organized as follows. In Section 2, we introduce the relaxation Newton method and the its two-step version. Section 3 is concerned with the monotone convergence of the new method. In Section 4, we test some numerical experiments to compare the performance of the new method with the SI method and the NBJ method in the sense of iteration number and CPU time. In the end of the work of this paper, we have constructed another two-step relaxation Newton algorithm which performs much better than the SI, NBJ methods; unfortunately, at the moment we cannot theoretically prove the convergence of this method to the minimal positive solution of the vector equations (1.7). Therefore, we will just report in Section 4 the numerical results of this method.

2. The Two-Step Relaxation Newton Method

In this section, we focus on introducing the basic idea of the relaxation Newton method investigated by [20, 21] and its two-step version for the following general nonlinear equations:

$$\mathcal{F}(x) = 0. \tag{2.1}$$

2.1. The Relaxation Newton Algorithm

The key idea of the relaxation Newton algorithm is to choose some splitting function $F{:}\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ which is minimally assumed to satisfy a consistency condition

$$F(x, x) = \mathcal{F}(x) \tag{2.2}$$

for any $x \in \mathbb{R}^n$. Then with an initial guess x_0 of the unknown solution x^*, we start with the previous approximation x_k to compute the next approximation x_{k+1} by solving the following problem:

$$F(x_k, x_{k+1}) = 0 \tag{2.3}$$

with some conventional method, such as the classical Newton's method, quasi-Newton methods, and Conjugate-Gradient method. Obviously, the generated sequence $\{x_k\}_0^\infty$ will upon convergence approach to some value x^* which satisfies $F(x^*, x^*) = 0$, that is, $\mathcal{F}(x) = 0$.

for $k = 0, 1, 2, \ldots$.
  with a given initial approximation $\tilde{x}_0$ of x_{k+1};
  for $m = 0, 1, \ldots, M$
    $F_2(x_k, \tilde{x}_m)\Delta\tilde{x}_m = -F(x_k, \tilde{x}_m)$,
    $\tilde{x}_{m+1} = \tilde{x}_m + \Delta\tilde{x}_m$,
  end
  $x_{k+1} = \Delta\tilde{x}_M$
end

Algorithm 1: The relaxation Newton method.

Wu et. al. [20, 21] used the classical Newton's method to solve (2.3), which explains the name: *relaxation Newton*; the deduced algorithm written compactly is shown in Algorithm 1.

In Algorithm 1 and what follows

$$F_2(x, y) = \left.\frac{\partial F(x, z)}{\partial z}\right|_{z=y}. \tag{2.4}$$

If we set $\tilde{x}_0 = x_k$ and $M = 1$ in the relaxation algorithm shown in Algorithm 1, by the consistency condition (2.2), the method can be written as

$$F_2(x_k, x_k)\Delta x_k = -\mathcal{F}(x_k), \qquad x_{k+1} = x_k + \Delta x_k, \quad k = 0, 1, \ldots. \tag{2.5}$$

With a special choice of F, the Jacobi matrix $F_2(x, x)$ will be a diagonal or block diagonal matrix and invertible in $\mathbb{R}^n$, and thus iteration method (2.5) can be processed stably and simultaneously with less storage compared with the classical Newton's method. We will see in the next section how to select the splitting function F such that the Jacobi matrix $F_2(x, x)$ is a diagonal or block diagonal matrix.

2.2. The Two-Step Relaxation Newton Method

Now, suppose that we have two splitting functions $F(x, y)$, $G(x, y, z)$ and an initial approximation x_0 of the solution of (2.1) at hand. We start with the previous approximation x_k to compute a mid-approximation $x_{k+1/2}$ by solving the equations

$$F(x_k, x_{k+1/2}) = 0, \tag{2.6a}$$

and then with $x_k, x_{k+1/2}$ at hand we compute x_{k+1} by solving the equations

$$G(x_k, x_{k+1/2}, x_{k+1}) = 0. \tag{2.6b}$$

The splitting functions F and G are assumed to satisfy the consistency conditions

$$F(x, x) = \mathcal{F}(x), \qquad G(x, x, x) = \mathcal{F}(x), \quad \forall x \in \mathbb{R}^n. \tag{2.7}$$

Similar to the relaxation Newton algorithm described above, we use Newton's method with a single iteration to solve equations (2.6a), (2.6b), and the deduced iteration scheme is

$$F_2(x_k, x_k)(x_{k+1/2} - x_k) = -\mathcal{F}(x_k), \tag{2.8a}$$

$$G_3(x_k, x_{k+1/2}, x_{k+1/2})(x_{k+1} - x_{k+1/2}) = -G(x_k, x_{k+1/2}, x_{k+1/2}), \tag{2.8b}$$

where F_2 is defined by (2.4), and the Jacobi matrix $G_3(x, y, z)$ of the function G is defined by

$$G_3(x, y, z) = \left.\frac{\partial G(x, y, s)}{\partial s}\right|_{s=z}. \tag{2.8c}$$

Throughout this paper, the iteration scheme (2.8a), (2.8b), (2.8c) is called *two-step relaxation Newton* (denoted by "TSRN") algorithm.

Specially, for the vector equations (1.7) we consider in this paper the following splitting functions:

$$F(w_k, w_{k+1/2}) = \begin{pmatrix} u_{k+1/2} - u_{k+1/2} \circ (Pv_k) - e \\ v_{k+1/2} - v_{k+1/2} \circ (Qu_k) - e \end{pmatrix}, \tag{2.9a}$$

$$G(w_k, w_{k+1/2}, w_{k+1}) = \begin{pmatrix} u_{k+1} - u_{k+1} \circ (Pv_{k+1/2}) - u_{k+1/2} \circ [\Phi(v_{k+1} - v_{k+1/2})] - e \\ v_{k+1} - v_{k+1} \circ (Qu_{k+1/2}) - v_{k+1/2} \circ [\Psi(u_{k+1} - u_{k+1/2})] - e \end{pmatrix}, \tag{2.9b}$$

here and hereafter $w_{k+1/2} = (u_{k+1/2}^T, v_{k+1/2}^T)^T$ and $w_k = (u_k^T, v_k^T)^T$ for all $k = 0, 1, 2 \ldots$; Φ and Ψ are diagonal matrices and their diagonal elements are determined as

$$\Phi_{i,i} = \begin{cases} P_{i,i}, & \text{if } i \text{ is odd}, \\ 0, & \text{if } i \text{ is even}, \end{cases} \qquad \Psi_{i,i} = \begin{cases} Q_{i,i}, & \text{if } i \text{ is even}, \\ 0, & \text{if } i \text{ is odd}. \end{cases} \tag{2.10}$$

Generally speaking, the splitting function G is a three variables function, but here we consider a special case—only the second and third variables are involved; see (2.9b). In the end of Section 4, we will see another TSRN algorithm where the function G is defined with 3 arguments (see (4.5b)).

Define

$$\begin{aligned} &D_k = \operatorname{diag}(e - Pv_k), \quad \Lambda_k = \operatorname{diag}(e - Qu_k), \quad \widetilde{D}_k = \operatorname{diag}(e - Pv_{k+1/2}), \\ &\widetilde{\Lambda}_k = \operatorname{diag}(e - Qu_{k+1/2}), \quad A_k = \operatorname{diag}(u_{k+1/2} \circ [\Phi e]), \quad B_k = \operatorname{diag}(v_{k+1/2} \circ [\Psi e]). \end{aligned} \tag{2.11}$$

Then it is clear that

$$F_2(w_k, w_k) = \begin{bmatrix} D_k & \\ & \Lambda_k \end{bmatrix}, \qquad G_3(w_k, w_{k+1/2}, w_{k+1/2}) = \begin{bmatrix} \widetilde{D}_k & -A_k \\ -B_k & \widetilde{\Lambda}_k \end{bmatrix}. \tag{2.12}$$

Therefore, it follows by substituting the function f defined in (1.7) and the splitting functions F, G defined in (2.9a), (2.9b) into (2.8a), (2.8b), (2.8c) that

$$F_2(w_k, w_k)\begin{pmatrix} u_{k+1/2} \\ v_{k+1/2} \end{pmatrix} = \begin{pmatrix} e \\ e \end{pmatrix}, \tag{2.13a}$$

$$G_3(w_k, w_{k+1/2}, w_{k+1/2})\begin{pmatrix} u_{k+1} - u_{k+1/2} \\ v_{k+1} - v_{k+1/2} \end{pmatrix} = -f(u_{k+1/2}, v_{k+1/2}). \tag{2.13b}$$

We note that with the special splitting functions F and G defined in (2.9a), (2.9b), equations (2.13a), (2.13b) implies

$$F(w_k, w_{k+1/2}) = 0, \qquad G(w_k, w_{k+1/2}, w_{k+1}) = 0, \quad k = 0, 1, 2, \ldots, \tag{2.14}$$

while for general splitting functions this is not always true. For diagonal matrices $\widetilde{D}_k$, $\widetilde{\Lambda}_k$, A_k and B_k, routine calculation yields

$$[G_3(w_k, w_{k+1/2}, w_{k+1/2})]^{-1} = \begin{bmatrix} \left[\widetilde{D}_k\widetilde{\Lambda}_k - A_kB_k\right]^{-1} & \\ & \left[\widetilde{D}_k\widetilde{\Lambda}_k - A_kB_k\right]^{-1} \end{bmatrix} \begin{bmatrix} \widetilde{\Lambda}_k & A_k \\ B_k & \widetilde{D}_k \end{bmatrix}. \tag{2.15}$$

Furthermore, by (2.10) we know that A_kB_k must be zero matrix for all $k = 0, 1, 2\ldots$; hence

$$[G_3(w_k, w_{k+1/2}, w_{k+1/2})]^{-1} = \begin{bmatrix} \widetilde{D}_k^{-1} & \left[\widetilde{D}_k\widetilde{\Lambda}_k\right]^{-1}A_k \\ \left[\widetilde{D}_k\widetilde{\Lambda}_k\right]^{-1}B_k & \widetilde{\Lambda}_k^{-1} \end{bmatrix}. \tag{2.16}$$

From (2.9a), (2.9b), (2.10), and (2.16) we obtain the algorithm for implementing the TSRN method as follows.

Algorithm 1 (Two-step relaxation Newton iteration). Starting with the initial value (u_0, v_0), the solution $\{u_k, v_k\}$ is defined, for $k = 0, 1, 2, \ldots$, by

(1) computing explicitly in the following elementwise fashion:

$$u_{k+1/2,i} = \frac{1}{1 - [Pv_k]_i}, \qquad v_{k+1/2,i} = \frac{1}{1 - [Qu_k]_i}, \quad i = 1, 2, \ldots, n; \tag{2.17}$$

(2) performing for $i = 1$ to n:
if i is odd

$$\begin{aligned} u_{k+1,i} &= \frac{1}{1 - [Pv_{k+1/2}]_i}\left(1 - \Phi_{ii}u_{k+1/2,i}v_{k+1/2,i} + \frac{\Phi_{ii}u_{k+1/2,i}}{1 - [Qu_{k+1/2}]_i}\right), \\ v_{k+1,i} &= \frac{1}{1 - [Qu_{k+1/2}]_i}, \end{aligned} \tag{2.18}$$

else

$$u_{k+1,i} = \frac{1}{1 - [Pv_{k+1/2}]_i},$$
$$v_{k+1,i} = \frac{1}{1 - [Qu_{k+1/2}]_i}\left(1 - \Psi_{ii}u_{k+1/2,i}v_{k+1/2,i} + \frac{\Psi_{ii}v_{k+1/2,i}}{1 - [Pv_{k+1/2}]_i}\right); \tag{2.19}$$

(3) if $\|f(u_{k+1}, v_{k+1})\|_\infty \leq \text{Tol}$, stop iteration; else set $u_k = u_{k+1}, v_k = v_{k+1}$ and go to the first step in Algorithm 1.

Clearly, compared with the Newton-type methods investigated by Lu [10] and Lin et al. [8], no LU-factorization is needed at each iteration of the TSRN method.

3. Convergence Analysis

For our proof of the monotone convergence of the TSRN method, we need the following results proved by Lu [9].

Lemma 3.1 (see [9]). *With the function g defined in* (1.8)*, one has*

(1) $u^* > e,\ v^* > e$,

(2) $\max_{1\leq i\leq 2n}\{g_i(e)\} < 1$.

The first conclusion can be obtained straightforwardly from (1.7).

Theorem 3.2. *Let $(u_0, v_0) = (0, 0)$ be the initial point of the TSRN method. The solution sequence $\{u_k, v_k\}_{k=1}^{\infty}$ generated by the TSRN method is strictly and monotonically increasing and converges to the minimal positive solution (u^*, v^*) of the vector equations* (1.7)*; particularly, it holds that*

(1) $0 \leq u_k < u_{k+1/2} < u_{k+1} < u^*$ *and* $0 \leq v_k < v_{k+1/2} < v_{k+1} < v^*$, $k = 0, 1, 2, \ldots$;

(2) $\lim_{k\to+\infty} u_k = u^*$ *and* $\lim_{k\to+\infty} v_k = v^*$.

Proof. We first prove the first conclusion of Theorem 3.2 by induction rule and this is completed in two-steps: first, we prove the correctness of this conclusion for $k = 0$, and then under the induction assumption with index $k = j$, we prove the case for $k = j + 1$.

(1) It is easy to get $(u_{1/2}, v_{1/2}) = (e, e)$. Hence, from the first conclusion of Lemma 3.1, it holds $u_{1/2} < u^*$ and $v_{1/2} < v^*$. By the second conclusion of Lemma 3.1, we get $e - Pv_{1/2} > 0$ and $e - Qu_{1/2} > 0$. Therefore, from (2.16) we know $[G_3(w_0, w_{1/2}, w_{1/2})]^{-1} > 0$. Since

$$u_{1/2} \circ (e - Pv_0) - e = 0, \qquad v_{1/2} \circ (e - Qu_0) - e = 0, \qquad u_{1/2} > u_0, \qquad v_{1/2} > v_0, \tag{3.1}$$

we have $u_{1/2} \circ (e - Pv_{1/2}) - e < 0$ and $v_{1/2} \circ (e - Qu_{1/2}) - e < 0$, that is, $f(u_{1/2}, v_{1/2}) < 0$. Hence, it follows from (2.13b) that $w_1 = w_{1/2} - [G_3(w_0, w_{1/2}, w_{1/2})]^{-1} f(u_{1/2}, v_{1/2}) > w_{1/2}$. We note that for any $\tilde{w}, w, w_-, w_+ \in \mathbb{R}^{2n}$ it holds

$$G(\tilde{w}, w, w_-) = G(\tilde{w}, w, w_+) + G_3(\tilde{w}, w, w_+)(w_- - w_+). \tag{3.2}$$

Hence, $G(w_0, w_{1/2}, w^*) + G_3(w_0, w_{1/2}, w^*)(w_1 - w^*) = G(w_0, w_{1/2}, w_1) = 0$, that is,

$$w_1 - w^* = -[G_3(w_0, w_{1/2}, w^*)]^{-1} G(w_0, w_{1/2}, w^*), \tag{3.3}$$

where $w^* = \begin{pmatrix} u^* \\ v^* \end{pmatrix}$. It is easy to verify $G_3(w_0, w_{1/2}, w^*) = G_3(w_0, w_{1/2}, w_{1/2})$ and this implies $[G_3(w_0, w_{1/2}, w^*)]^{-1} > 0$. Consider

$$u^* \circ (e - Pv^*) = e, \qquad v^* \circ (e - Qu^*) = e, \tag{3.4}$$

$$\begin{aligned} &u^* \circ (e - Pv_{1/2}) - u_{1/2} \circ \Phi(v^* - v_{1/2}) - u^* \circ (e - Pv^*) \\ &\quad = u^* \circ [P(v^* - v_{1/2})] - u_{1/2} \circ \Phi(v^* - v_{1/2}), \\ &v^* \circ (e - Qu_{1/2}) - v_{1/2} \circ \Psi(u^* - u_{1/2}) - v^* \circ (e - Qu^*) \\ &\quad = v^* \circ [Q(u^* - u_{1/2})] - v_{1/2} \circ \Psi(u^* - u_{1/2}), \end{aligned} \tag{3.5}$$

and $u_{1/2} < u^*, v_{1/2} < v^*$, and we know that the right hand of (3.5) is positive, and this implies $G(w_0, w_{1/2}, w^*) > 0$. Therefore, from (3.3) we get $w_1 < w^*$. We have proved that the first conclusion of Theorem 3.2 is correct for $k = 0$.

(2) Assume that the first conclusion of Theorem 3.2 is correct for $k = j$, $j \geq 0$ and we will prove that it is also correct for $k = j + 1$. To this end, we note that under the induction assumption it holds

$$\begin{aligned} e > e - Qu_j > e - Qu_{j+1/2} > e - Qu_{j+1} > e - Qu^* > 0, \\ e > e - Pv_j > e - Pv_{j+1/2} > e - Pv_{j+1} > e - Pv^* > 0, \end{aligned} \tag{3.6}$$

where the inequalities $e - Qu^* > 0$ and $e - Pv^* > 0$ follow directly from equality (3.4).

Consider

$$\begin{aligned} &u_{j+1} \circ (e - Pv_{j+1/2}) - u_{j+1/2} \circ [\Phi(v_{j+1} - v_{j+1/2})] - u_{j+1} \circ (e - Pv_{j+1}) \\ &\quad = u_{j+1} \circ [P(v_{j+1} - v_{j+1/2})] - u_{j+1/2} \circ [\Phi(v_{j+1} - v_{j+1/2})] > 0, \\ &v_{j+1} \circ (e - Qu_{j+1/2}) - v_{j+1/2} \circ [\Psi(u_{j+1} - u_{j+1/2})] - v_{j+1} \circ (e - Qu_{j+1}) \\ &\quad = v_{j+1} \circ [Q(u_{j+1} - u_{j+1/2})] - v_{j+1/2} \circ [\Psi(u_{j+1} - u_{j+1/2})] > 0, \end{aligned} \tag{3.7}$$

and since $w_{j+1/2} < w_{j+1}$, we get

$$\begin{aligned} u_{j+1+1/2} \circ (e - Pv_{j+1}) &= e = u_{j+1} \circ (e - Pv_{j+1/2}) - u_{j+1/2} \circ [\Phi(v_{j+1} - v_{j+1/2})] \\ &> u_{j+1} \circ (e - Pv_{j+1}), \\ v_{j+1+1/2} \circ (e - Qu_{j+1}) &= e = v_{j+1} \circ (e - Qu_{j+1/2}) - v_{j+1/2} \circ [\Psi(u_{j+1} - u_{j+1/2})] \\ &> u_{j+1} \circ (e - Qu_{j+1}), \end{aligned} \tag{3.8}$$

and this coupled with (3.6) gives

$$w_{j+1} < w_{j+1+1/2}. \tag{3.9}$$

From (3.6) and $w_{j+1} < w^*$, we have

$$\begin{aligned} u_{j+1+1/2} \circ \left(e - Pv_{j+1}\right) = e = u^* \circ (e - Pv^*) < u^* \circ \left(e - Pv_{j+1}\right) \\ v_{j+1+1/2} \circ \left(e - Qu_{j+1}\right) = e = v^* \circ (e - Qu^*) < v^* \circ \left(e - Qu_{j+1}\right) \end{aligned} \Longrightarrow \begin{aligned} u_{j+1+1/2} < u^*, \\ v_{j+1+1/2} < v^*, \end{aligned} \tag{3.10}$$

that is, $w_{j+1+1/2} < w^*$; this coupled with (3.6) implies $e - Pv_{j+1+1/2} > 0$ and $e - Qu_{j+1+1/2} > 0$. Hence, we arrive at

$$\left[G_3\left(w_{j+1}, w_{j+1+1/2}, w_{j+1+1/2}\right)\right]^{-1} > 0. \tag{3.11}$$

From (3.9) we have

$$\begin{aligned} u_{j+1+1/2} \circ \left(e - Pv_{j+1+1/2}\right) - e < u_{j+1+1/2} \circ \left(e - Pv_{j+1}\right) - e = 0 \\ v_{j+1+1/2} \circ \left(e - Qu_{j+1+1/2}\right) - e < v_{j+1+1/2} \circ \left(e - Qu_{j+1}\right) - e = 0 \end{aligned} \Longrightarrow f\left(u_{j+1+1/2}, v_{j+1+1/2}\right) < 0; \tag{3.12}$$

this coupled with (3.11) gives

$$w_{j+2} = w_{j+1+1/2} - \left[G_3\left(w_{j+1}, w_{j+1+1/2}, w_{j+1+1/2}\right)\right]^{-1} f\left(u_{j+1+1/2}, v_{j+1+1/2}\right) > w_{j+1+1/2}. \tag{3.13}$$

Next, we prove $w_{j+2} < w^*$. To this end, from (3.2) and (2.14) we have

$$G\left(w_{j+1}, w_{j+1+1/2}, w^*\right) + G_3\left(w_{j+1}, w_{j+1+1/2}, w^*\right)\left(w_{j+2} - w^*\right) = G\left(w_{j+1}, w_{j+1+1/2}, w_{j+2}\right) = 0, \tag{3.14}$$

and this implies

$$w_{j+2} - w^* = -\left[G_3\left(w_{j+1}, w_{j+1+1/2}, w^*\right)\right]^{-1} G\left(w_{j+1}, w_{j+1+1/2}, w^*\right). \tag{3.15}$$

From (3.11), we get

$$\left[G_3\left(w_{j+1}, w_{j+1+1/2}, w^*\right)\right]^{-1} = \left[G_3\left(w_{j+1}, w_{j+1+1/2}, w_{j+1+1/2}\right)\right]^{-1} > 0, \tag{3.16}$$

where the first equality can be easily verified from (2.9b). Consider (3.4) and the following relations:

$$\begin{aligned} u^* \circ \left(e - Pv_{j+1+1/2}\right) - u_{j+1+1/2} \circ \Phi\left(v^* - v_{j+1+1/2}\right) - u^* \circ (e - Pv^*) \\ = u^* \circ \left[P\left(v^* - v_{j+1+1/2}\right)\right] - u_{j+1+1/2} \circ \Phi\left(v^* - v_{j+1+1/2}\right) > 0, \\ v^* \circ \left(e - Qu_{j+1+1/2}\right) - v_{j+1+1/2} \circ \Psi\left(u^* - u_{j+1+1/2}\right) - v^* \circ (e - Qu^*) \\ = v^* \circ \left[Q\left(u^* - u_{j+1+1/2}\right)\right] - v_{j+1+1/2} \circ \Psi\left(u^* - u_{j+1+1/2}\right) > 0, \end{aligned} \tag{3.17}$$

and since $u_{j+1+1/2} < u^*, v_{j+1+1/2} < v^*$, we have $G(w_{j+1}, w_{j+1+1/2}, w^*) > 0$; this coupled with (3.15) and (3.16) gives

$$w_{j+2} < w^*. \tag{3.18}$$

By (3.9), (3.13) and (3.18) we have proved the validity of the first conclusion of Theorem 3.2 for $k = j + 1$. Therefore, by induction rule we have completed the proof of the first conclusion of Theorem 3.2.

From the first conclusion of Theorem 3.2, it is obvious that there exist positive vectors $\widehat{u}^*$ and $\widehat{v}^*$ such that

$$\lim_{k \to +\infty} u_k = \lim_{k \to +\infty} u_{k+1/2} = \widehat{u}^*, \qquad \lim_{k \to +\infty} v_k = \lim_{k \to +\infty} v_{k+1/2} = \widehat{v}^*. \tag{3.19}$$

Therefore, it follows from (2.9a), (2.9b), and (2.14) that

$$\begin{aligned} \widehat{u}^* &= \widehat{u}^* \circ (P\widehat{v}^*) + e, \\ \widehat{v}^* &= \widehat{v}^* \circ (Q\widehat{u}^*) + e. \end{aligned} \tag{3.20}$$

This implies that $(\widehat{u}^*, \widehat{v}^*)$ is a positive solution of the vector equations (1.7). According to the minimal property of (u^*, v^*), it must hold $\widehat{u}^* = u^*$ and $\widehat{v}^* = v^*$. □

To finish this section, we compare the efficiency of the TSRN method with the NBJ method. To this end, we rewrite the original NBJ iteration scheme (1.13a) into the following form:

$$\begin{aligned} \mathcal{U}_{k+1/2} &= \mathcal{U}_{k+1/2} \circ [P\mathcal{V}_k] + e, \\ \mathcal{V}_{k+1/2} &= \mathcal{V}_{k+1/2} \circ [Q\mathcal{U}_k] + e, \\ \mathcal{U}_{k+1} &= \mathcal{U}_{k+1} \circ [P\mathcal{V}_{k+1/2}] + e, \\ \mathcal{V}_{k+1} &= \mathcal{V}_{k+1} \circ [Q\mathcal{U}_{k+1/2}] + e. \end{aligned} \tag{3.21}$$

Clearly, the TSRN method will reduce into the two-step version NBJ method (3.21) if $\Psi = \Phi = 0$ in (2.9b).

The following theorem indicates that the TSRN method is more efficient than the NBJ method.

Theorem 3.3. *Let both the NBJ method and the TSRN method start with the initial value* $(0,0)$*, and* $\{u_k, v_k\}$, $\{\mathcal{U}_k, \mathcal{V}_k\}$ *be the sequences generated by the TSRN method and the NBJ method, respectively. Then it holds that*

$$\begin{aligned} \mathcal{U}_{k+1/2} &< u_{k+1/2}, \\ \mathcal{V}_{k+1/2} &< v_{k+1/2}, \\ \mathcal{U}_{k+1} &< u_{k+1}, \\ \mathcal{V}_{k+1} &< v_{k+1} \end{aligned} \tag{3.22}$$

for $k \geq 2$.

Proof. It is easy to get $(u_{1/2}, v_{1/2}) = (\mathcal{U}_{1/2}, \mathcal{V}_{1/2}) = (e, e)$. Hence, we have

$$\begin{aligned} (u_1 - \mathcal{U}_1) \circ [e - Pe] &= u_{1/2} \circ [\Phi(v_1 - v_{1/2})], \\ (v_1 - \mathcal{V}_1) \circ [e - Qe] &= v_{1/2} \circ [\Psi(u_1 - u_{1/2})]. \end{aligned} \tag{3.23}$$

This coupled with (2.10), the second conclusion of Lemma 3.1, and the first conclusion of Theorem 3.2 gives

$$\begin{aligned} u_{1,i} &> \mathcal{U}_{1,i}, \quad \text{if } i \text{ is odd}, \\ u_{1,i} &= \mathcal{U}_{1,i}, \quad \text{if } i \text{ is even}, \\ v_{1,i} &> \mathcal{V}_{1,i}, \quad \text{if } i \text{ is even}, \\ v_{1,i} &= \mathcal{V}_{1,i}, \quad \text{if } i \text{ is odd}. \end{aligned} \tag{3.24}$$

Since $P,\ Q > 0$, it follows from (3.24) that

$$\begin{aligned} u_{2+1/2} \circ [e - Pv_1] = e = \mathcal{U}_{2+1/2} \circ [e - P\mathcal{V}_1] > \mathcal{U}_{2+1/2} \circ [e - Pv_1], \\ v_{2+1/2} \circ [e - Qu_1] = e = \mathcal{V}_{2+1/2} \circ [e - Q\mathcal{U}_1] > \mathcal{V}_{2+1/2} \circ [e - Qu_1], \end{aligned} \tag{3.25}$$

where in the last inequality we used the relation (3.6) and the fact established by [6]—the sequence $\{\mathcal{U}_k, \mathcal{V}_k\}$ is strictly and monotonically increasing and converges to the minimal positive solution (u^*, v^*). From (3.25), we have $u_{2+1/2} > \mathcal{U}_{2+1/2}$ and $v_{2+1/2} > \mathcal{V}_{2+1/2}$.

Consider

$$\begin{aligned} u_3 \circ [e - Pv_{2+1/2}] - u_{2+1/2} \circ [\Phi(v_3 - v_{2+1/2})] = e = \mathcal{U}_3 \circ [e - P\mathcal{V}_{2+1/2}] > \mathcal{U}_3 \circ [e - Pv_{2+1/2}], \\ v_3 \circ [e - Qu_{2+1/2}] - v_{2+1/2} \circ [\Psi(u_3 - u_{2+1/2})] = e = \mathcal{V}_3 \circ [e - Q\mathcal{U}_{2+1/2}] > \mathcal{V}_3 \circ [e - Qu_{2+1/2}], \end{aligned} \tag{3.26}$$

the we arrive at $u_3 > \mathcal{U}_3$ and $v_3 > \mathcal{V}_3$. Therefore, the proof of (3.22) can be completed by using relations (3.25) and (3.26) repeatedly. □

Remark 3.4. Since the NBJ method is more feasible than the NBGS method in parallel environment, in this paper we only focus on comparing the efficiency between the TSRN and the NBJ methods.

4. Numerical Results

In this section, we compute the minimal positive solution of the vector equations (1.7) by the TSRN method, in order to compare numerically the feasibility and effectiveness of this algorithm with the SI method (1.10) and the NBJ method (1.13a) in the sense of number of iteration (denoted as "IT") and elapsed CPU time in seconds (denoted as "CPU"). Clearly, these methods are very suitable to be performed in parallel environment. We remark that the Newton type methods coupled with LU-factorization [6, 10], NBGS method (1.13b), and the modified simple iteration (1.12) are sequential methods and not suitable to be implemented in parallel environment.

In all experiments, the constants c_i and ω_i, $i = 1, 2, \ldots, n$, are given by the composite Gauss-Legendre quadrature formula on the interval $[0, 1]$. More precisely, the interval $[0, 1]$ is first divided into subintervals of equal length, and the composite Gauss-Legendre quadrature formula with 4 nodes is then applied to each subinterval (see [14]). In our implementations, all iterations start with initial value $(0, 0)$ and are terminated once the current residual error defined as

$$\mathrm{ERR}_k = \max\left\{\|u_k - u_k \circ (Pv_k) - e\|_\infty,\ \|v_k - v_k \circ (Qu_k) - e\|_\infty\right\} \tag{4.1}$$

satisfies $\mathrm{ERR}_k \leq 10^{-13}$. All codes are performed in MATLAB (version 7.0) on an Intel (R) Pentium (R) Dual E2110 @ 1.4 GHz PC with memory 1 GB.

To make a fair comparison, we rewrite equivalently the SI method (1.10) into two-step fashion as follows:

$$\begin{aligned} u_{k+1/2} &= u_k \circ [Pv_k] + e, \\ v_{k+1/2} &= v_k \circ [Qu_k] + e, \\ u_{k+1} &= u_{k+1/2} \circ [Pv_{k+1/2}] + e, \\ v_{k+1} &= v_{k+1/2} \circ [Qu_{k+1/2}] + e, \end{aligned} \tag{4.2}$$

and the vectors $u_{k+1/2}, v_{k+1/2}$ and u_{k+1}, v_{k+1} are explicitly computed in elementwise as

$$\begin{aligned} u_{k+1/2,i} &= u_{k,i}[Pv_k]_i + 1, \\ v_{k+1/2,i} &= v_{k,i}[Qu_k]_i + 1, \\ u_{k+1,i} &= u_{k+1/2,i}[Pv_{k+1/2}]_i + 1, \\ v_{k+1,i} &= v_{k+1/2,i}[Qu_{k+1/2}]_i + 1, \quad i = 1, 2, \ldots. \end{aligned} \tag{4.3}$$

Table 1: Numerical results for $n = 32$ and different (α, c).

		(α, c)				
Method		$(0.1, 0.9)$	$(0.001, 0.995)$	$(10^{-5}, 1 - 2 \times 10^{-5})$	$(10^{-7}, 1 - 10^{-7})$	$(10^{-12}, 1 - 10^{-12})$
IT	SI	37	181	2377	24405	71486
	NBJ	20	84	1040	10609	31092
	TSRN	20	81	1029	10499	30662
CPU	SI	0.018	0.035	0.29	2.7	7.443
	NBJ	0.001	0.026	0.210	1.551	3.525
	TSRN	0.001	0.015	0.110	1.270	3.063

Table 2: Numerical results for $(\alpha, c) = (10^{-7}, 1)$ and different n.

		n							
Method		8	16	32	64	128	256	512	1024
IT	SI	70317	71076	71494	71697	71796	71889	71931	72014
	NBJ	32090	31401	31123	30932	30724	30743	30742	30771
	TSRN	30945	30713	30664	30643	30637	30639	30678	30715
CPU	SI	5.503	5.971	7.450	12.540	26.587	108.811	466.51	1799.747
	NBJ	2.687	2.865	3.441	6.437	11.501	47.355	203.578	751.886
	TSRN	2.392	2.525	3.237	5.608	10.568	46.458	201.795	735.711

The NBJ method is also performed in two-step fashion as defined in (3.21) and the vectors are computed in elementwise as:

$$\begin{aligned} u_{k+1/2,i} &= \frac{1}{1 - [Pv_k]_i}, \\ v_{k+1/2,i} &= \frac{1}{1 - [Qu_k]_i}, \\ u_{k+1,i} &= \frac{1}{1 - [Pv_{k+1/2}]_i}, \\ v_{k+1,i} &= \frac{1}{1 - [Qu_{k+1/2}]_i}, \quad i = 1, 2, \ldots. \end{aligned} \tag{4.4}$$

Example 4.1. In Table 1, for the fixed problem size $n = 32$ but different pairs of (α, c), and in Table 2 for the fixed $(\alpha, c) = (10^{-7}, 1)$ but different problem size n, we list ITs and CPUs for the SI, NBJ and TSRN methods, respectively.

We see clearly in Table 1 that with fixed problem size, the TSRN method converges faster than the SI and NBJ methods, and as α converges to 0 and c converges to 1, the advantages of the TSRN method are more significant. Moreover, for each pair (α, c), the TSRN method needs less CPU time than the SI and NBJ methods, even through the former needs a little more flops at each iteration. When the parameters α and c are fixed, as the problem size n becomes large, the performance of the TSRN method and the NBJ method becomes close, as shown in Table 2.

Table 3: Numerical results for $n = 256$ and different (a,c).

		(α, c)				
Method		$(0.01, 0.99)$	$(10^{-6}, 1-3\times10^{-6})$	$(10^{-7}, 1-10^{-7})$	$(10^{-8}, 1-10^{-9})$	$(10^{-9}, 1-10^{-9})$
IT	NBJ	61	2417	10514	29465	29536
	TSRN	60	2411	10489	29339	29310
	TSRN*	46	1976	8612	24106	24114
CPU	NBJ	0.109	3.783	16.058	45.401	47.972
	TSRN	0.098	3.581	15.121	43.948	44.770
	TSRN*	0.065	2.981	13.242	36.725	36.845

Table 4: Numerical results for $(\alpha, c) = (5\times10^{-9}, 1-3\times10^{-9})$ and different n.

		n							
Method		8	16	32	64	128	256	384	960
IT	NBJ	28552	27834	27476	27329	27276	27275	27234	27211
	TSRN	27409	27205	27146	27130	27133	27143	27158	27196
	TSRN*	23071	22659	22466	22363	22333	22325	22322	22342
CPU	NBJ	2.278	2.827	3.107	5.081	12.481	42.595	95.803	480.821
	TSRN	2.115	2.447	2.997	4.758	10.921	40.984	92.919	472.829
	TSRN*	1.778	1.971	2.503	3.958	8.750	33.970	76.889	441.720

Example 4.2. In the end of the work of this paper, we have constructed another two-step relaxation Newton algorithm (denoted as TSRN* for the moment) with the splitting functions F and G defined as

$$F(w_{k+1/2}, w_k) = \begin{pmatrix} u_{k+1/2} - u_{k+1/2} \circ [Pv_k] - e \\ v_{k+1/2} - v_{k+1/2} \circ [Qu_k] - e \end{pmatrix}, \tag{4.5a}$$

$$G(w_k, w_{k+1/2}, w_{k+1}) = \begin{pmatrix} u_{k+1} - u_{k+1} \circ [Pv_k] - (u_{k+1/2} \circ [P(v_{k+1/2} - v_k)] + Q(u_{k+1/2} - u_k)) - e \\ v_{k+1} - v_{k+1} \circ [Qu_k] - (v_{k+1/2} \circ [Q(u_{k+1/2} - u_k)] + P(v_{k+1/2} - v_k)) - e \end{pmatrix}. \tag{4.5b}$$

With these two splitting functions, we list in Table 3, for fixed problem size $n = 256$ but different (α, c), and in Table 4 for fixed $(\alpha, c) = (5\times10^{-9}, 1-3\times10^{-9})$ but different problem size, ITs and CPUs for the NBJ, TSRN and TSRN* methods (from the numerical results given in the previous example we believe that the NBJ and TSRN, methods converge faster than the SI method, and thus at the moment we omit the comparison of the TSRN* method with the SI method).

For each problem size n and (α, c) tested in Tables 3 and 4, it holds that

$$\max\{\|U_{\text{TSRN}} - U_{\text{TSRN}^*}\|_\infty,\ \|V_{\text{TSRN}} - V_{\text{TSRN}^*}\|_\infty\} \le 10^{-14}, \tag{4.6}$$

where $U_{\text{TSRN}}, V_{\text{TSRN}}$ and $U_{\text{TSRN}^*}, V_{\text{TSRN}^*}$ are the converged vectors generated by the TSRN and TSRN* methods, respectively. Therefore, the TSRN* method really converges to the minimal positive solution of the vector equations (1.7). Moreover, from the numerical results listed

in Tables 3 and 4, we see clearly that the TSRN* method performs much better than the TSRN and NBJ methods. Unfortunately, at the moment we cannot prove theoretically the convergence of the TSRN* method to the minimal positive solution of the vector equations (1.7).

Acknowledgments

This work was supported by the NSF of China (no. 10971077) and by the program for NCET, the State Education Ministry of China.

References

[1] R. Bellman and G. M. Wing, *An Introduction to Invariant Imbedding*, John Wiley & Sons, New York, NY, USA, 1975.
[2] J. Juang, "Existence of algebraic matrix Riccati equations arising in transport theory," *Linear Algebra and Its Applications*, vol. 230, pp. 89–100, 1995.
[3] J. Juang and I. Chen, "Iterative solution for a certain class of algebraic matrix Riccati equations arising in transport theory," *Transport Theory and Statistical Physics*, vol. 22, no. 1, pp. 65–80, 1993.
[4] J. Juang and W.-W. Lin, "Nonsymmetric algebraic Riccati equations and Hamiltonian-like matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 1, pp. 228–243, 1999.
[5] J. Juang and Z. T. Lin, "Convergence of an iterative technique for algebraic matrix Riccati equations and applications to transport theory," *Transport Theory and Statistical Physics*, vol. 21, no. 1-2, pp. 87–100, 1992.
[6] Z.-Z. Bai, Y.-H. Gao, and L.-Z. Lu, "Fast iterative schemes for nonsymmetric algebraic Riccati equations arising from transport theory," *SIAM Journal on Scientific Computing*, vol. 30, no. 2, pp. 804–818, 2008.
[7] L. Bao, Y. Lin, and Y. Wei, "A modified simple iterative method for nonsymmetric algebraic Riccati equations arising in transport theory," *Applied Mathematics and Computation*, vol. 181, no. 2, pp. 1499–1504, 2006.
[8] Y. Q. Lin, L. Bao, and Y. Wei, "A modified Newton method for solving non-symmetric algebraic Riccati equations arising in transport theory," *IMA Journal of Numerical Analysis*, vol. 28, no. 2, pp. 215–224, 2008.
[9] L.-Z. Lu, "Solution form and simple iteration of a nonsymmetric algebraic Riccati equation arising in transport theory," *SIAM Journal on Matrix Analysis and Applications*, vol. 26, no. 3, pp. 679–685, 2005.
[10] L.-Z. Lu, "Newton iterations for a non-symmetric algebraic Riccati equation," *Numerical Linear Algebra with Applications*, vol. 12, no. 2-3, pp. 191–200, 2005.
[11] P. Nelson, "Convergence of a certain monotone iteration in the reflection matrix for a non-multiplying half-space," *Transport Theory and Statistical Physics*, vol. 13, no. 1-2, pp. 97–106, 1984.
[12] A. Shimizu and K. Aoki, *Application of Invariant Embedding to Reactor Physics*, Academic Press, New York, NY, USA, 1972.
[13] Z.-Z. Bai, X.-X. Guo, and S.-F. Xu, "Alternately linearized implicit iteration methods for the minimal nonnegative solutions of the nonsymmetric algebraic Riccati equations," *Numerical Linear Algebra with Applications*, vol. 13, no. 8, pp. 655–674, 2006.
[14] C.-H. Guo and A. J. Laub, "On the iterative solution of a class of nonsymmetric algebraic Riccati equations," *SIAM Journal on Matrix Analysis and Applications*, vol. 22, no. 2, pp. 376–391, 2000.
[15] C.-H. Guo, "Nonsymmetric algebraic Riccati equations and Wiener-Hopf factorization for M-matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 225–242, 2001.
[16] X.-X. Guo, W.-W. Lin, and S.-F. Xu, "A structure-preserving doubling algorithm for nonsymmetric algebraic Riccati equation," *Numerische Mathematik*, vol. 103, no. 3, pp. 393–412, 2006.
[17] C.-H. Guo, B. Iannazzo, and B. Meini, "On the doubling algorithm for a (shifted) nonsymmetric algebraic Riccati equation," *SIAM Journal on Matrix Analysis and Applications*, vol. 29, no. 4, pp. 1083–1100, 2007.
[18] G. H. Golub and C. F. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Md, USA, 3rd edition, 1996.

[19] Y. Q. Lin, "A class of iterative methods for solving nonsymmetric algebraic Riccati equations arising in transport theory," *Computers & Mathematics with Applications*, vol. 56, no. 12, pp. 3046–3051, 2008.
[20] S. Wu, C. M. Huang, and Y. Liu, "Newton waveform relaxation method for solving algebraic nonlinear equations," *Applied Mathematics and Computation*, vol. 201, no. 1-2, pp. 553–560, 2008.
[21] S. L. Wu, B. C. Shi, and C. M. Huang, "Relaxation Newton iteration for a class of algebraic nonlinear systems," *International Journal of Nonlinear Science*. In press.