Available online at www.sciencedirect.com

ScienceDirect

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Journal of Computational and Applied Mathematics 222 (2008) 645–654

www.elsevier.com/locate/cam

Improved Newton's method with exact line searches to solve quadratic matrix equation[☆]

Jian-hui Long[*], Xi-yan Hu, Lei Zhang

College of Mathematics and Econometrics, Hunan University, Changsha 410082, China

Received 10 April 2007; received in revised form 15 September 2007

Abstract

In this paper, we study the matrix equation $AX^2 + BX + C = 0$, where A, B and C are square matrices. We give two improved algorithms which are better than Newton's method with exact line searches to calculate the solution. Some numerical examples are reported to illustrate our algorithms.
© 2007 Elsevier B.V. All rights reserved.

MSC: 65F30; 65H10; 39B42

Keywords: Quadratic matrix equation; Solvent; Newton's method; Exact line search; Šamanskii technique

1. Introduction

In this paper, our interest is the simplest quadratic matrix equation

$$AX^2 + BX + C = 0, \tag{1.1}$$

where A, B and C are given square matrices. The Eq. (1.1) often occurs in many applications, for example, the quasi-birth–death processes, the quadratic eigenvalue problems (QEP) and pseudo-spectra for quadratic eigenvalue problems, see detail in [5,8,12,14]. Let

$$F(X) = AX^2 + BX + C, \tag{1.2}$$

where A, B and C are square matrices. A matrix S is called a solvent of the Eq. (1.1) if $F(S) = 0$ [4]. In the case of simple solvent, where the derivative is regular. A dominant solvent is a solvent matrix whose eigenvalues strictly dominate the eigenvalues of all other solvents. The paper [10,13] gave two linearly convergent algorithms for computing a dominant solvent of Eq. (1.1) while the algorithms have the drawbacks that it is difficult to check

☆ This research supported by the National Natural Science Foundation of China 10571047 and Doctorate Foundation of the Ministry of Education of China 20060532014.

* Corresponding author.
E-mail address: jianhuipaper@yahoo.com.cn (J.-h. Long).

doi:10.1016/j.cam.2007.12.018

Table 1.1
Newton's method

The ith iteration	Error	The ith iteration	Error
1	2.7394e+010	3	1.7121e+009
5	1.0701e+008	7	6.6871e+006
9	4.1706e+005	11	2.5220e+004
13	1.1082e+003	15	20.6248
16	0.7567	17	0.0017
18	1.1049e−008	19	5.3417e−014

Table 1.2
Newton's method with exact line searches

The ith iteration	t	Error
1	1.9997	2.2820e+003
2	0.8375	427.5598
3	1.1719	36.4079
4	1.0222	1.0573
5	1.0018	0.0015
6	1.0000	5.0189e−009
7	1.0000	3.6296e−014

in advance whether a dominant solvent exists and the convergence can be extremely slow. Davis applied Newton's method to the Eq. (1.1), and gave supporting theory and implementation details in [1,2]. Kratz and Stickel investigated Newton's method for the general degree matrix polynomials in [4]. Higham and Kim improved the global convergence properties of Newton's method with exact line searches and gave a complete characterization of solution in terms of the generalized Schur decomposition in [5]. Various numerical solution techniques are described and compared in [6]. From the above papers, we know that Newton's method is very attractive for solving the Eq. (1.1). It has very nice numerical behavior such as quadratic convergence and good numerical stability, but it has a weak necessary that each iteration is rather expansive. We know also that modified Newton's method is cheaper than Newton's method at each iteration [4], while it is only linearly convergence.

Consider the quadratic matrix equation $AX^2 + BX + C = 0$ with $A, B, C \in R^{n\times n}, n = 120$

$$A = I_n, \qquad B = \begin{pmatrix} 20 & -10 & & & \\ -10 & 30 & -10 & & \\ & -10 & \ddots & \ddots & \\ & & \ddots & 30 & -10 \\ & & & -10 & 20 \end{pmatrix}, \qquad C = \begin{pmatrix} 15 & -5 & & & \\ -5 & 15 & -5 & & \\ & -5 & \ddots & \ddots & \\ & & \ddots & 15 & -5 \\ & & & -5 & 15 \end{pmatrix}.$$

The Tables 1.1 and 1.2 list the numerical results with starting iterative $X_0 = 10^4 I_n$ by Newton's method and Newton's method with exact line searches, respectively, where error $= \|F(X_i)\|_F$, $\|\cdot\|_F$ denotes *Frobenious* norm, t is a real step size scalar in the direction of Newton.

It can be seen from the Tables 1.1 and 1.2 that when error $> 10^{-1}$, $\|F(X_i)\|_F$ by Newton's method with exact line searches comes down faster than by Newton's method, while when error $< 10^{-1}$ Newton's method with exact line searches has the same convergence rate as Newton's method and t sufficiently near or equal to 1.

The paper has two main contributions. First, we incorporate Newton's method with exact line searches and Newton's method in order to reduce the cost of computation. Second, we incorporate Newton's method with exact line searches and Newton's method with Šamanskii technique in order to have faster convergence than Newton's method with exact line searches.

The paper is organized as follows. In Section 2, we introduce some notations and lemmas. In Section 3, two algorithms are presented and analyzed. In order to illustrate our results, several numerical examples are reported in Section 4. At last, some conclusions are given in Section 5.

2. Some notations and lemmas

For matrices $A, B \in C^{n\times n}$ the *Kronecker* product is the matrix $A \otimes B \in C^{n^2\times n^2}$ given by

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{n1}B & \cdots & a_{nn}B \end{pmatrix}, \quad \text{where } A = (a_{ij}),$$

and the vec function of the matrix A is the column

$$\mathrm{vec}(A) = (a_1^{\mathrm{T}}, \ldots, a_n^{\mathrm{T}})^{\mathrm{T}} \in C^{n^2}$$

where $a_1, \ldots, a_n \in C^n$ are the the columns of A.

Definition 2.1 (*[3,4,14]*). Let $G(X) = 0$ be the nonlinear matrix equation, where $G : C^{n\times n} \to C^{n\times n}$. If $G(X+E) - G(X) = G'(X)[E] + G_1(E)$, where $G_1 : C^{n\times n} \to C^{n\times n}$, G' is a bounded linear operator and $\|G_1(E)\|/\|E\| \to 0$ as $\|E\| \to 0$, then the function G is said to be Frechet-differentiable at X with Frechet derivative $G'(X)[E]$ in the direction E.

From Eq. (1.1), we obtain

$$F(X+E) = F(X) + AEX + (AX+B)E + AE^2. \tag{2.1}$$

Definition 2.1 implies that the Frechet derivative of Eq. (1.1) at X in the direction E is given by

$$F'(X)[E] = AEX + (AX+B)E. \tag{2.2}$$

We call $F'(X)$ regular if the mapping $F'(X)$ is injective.

Algorithm 2.1 (*Newton's Method*).

step 0: Given X_0, and ε, set $k = 0$.
step 1: If $\mathrm{error}_k < \varepsilon$, stop.
step 2: Solve for E_k in generalized *Sylvester* equation

$$AE_kX_k + (AX_k + B)E_k = -F(X_k). \tag{2.3}$$

step 3: Update $X_{k+1} = X_k + E_k$, $k = k+1$, and go to step 1, where $\mathrm{error}_k = \|F(X_k)\|_F$.

Lemma 2.1 (*[14]*). *Suppose that $S \in C^{n\times n}$ is a simple solvent of (1.1), i.e. $F(S) = 0$ and $F'(S)$ is regular. Then, if the starting matrix X_0 is sufficiently near S, the sequence $\{X_k\}$ produced by Algorithm 2.1 converges quadratically to S. More precisely, if $\|X_0 - S\| = \varepsilon_0 < \varepsilon = \min\{\frac{c_0}{\|A\|}, \delta\}$, where*

$$c_0 = \inf\left\{\left\|F'(X)[H]\right\| : \|H\| = 1, \|X - S\| \le \delta\right\} > 0$$

for sufficiently small $\delta \in (0, 1]$, then we have:

(i) $\lim_{k\to\infty} X_k = S$,
(ii) $\|X_k - S\| \le \varepsilon_0 q^k$ *with* $q = \frac{\varepsilon_0\|A\|}{c_0} < 1$, *for* $k = 0, 1, 2, \ldots$,
(iii) $\|X_{k+1} - S\| \le \frac{\|A\|}{c_0}\|X_k - S\|^2$ *for* $k = 0, 1, 2, \ldots$,

Remark 2.1. The Algorithm 2.1 has quadratical convergence rate, but it has some weakness. When it solves the generalized *Sylvester* equation defining E_k, it takes at least $56n^3$ *flops*, thus each iteration is rather expansive.

Algorithm 2.2 (*Modified Newton's Method*).

step 0: Given X_0, *and* ε, set $k = 0$.
step 1: If $\mathrm{error}_k < \varepsilon$, stop.

```
step 2: Solve for $E_k$ in generalized Sylvester equation
```

$$AE_kX_0 + (AX_0 + B)E_k = -F(X_k). \tag{2.4}$$

```
step 3: Update $X_{k+1} = X_k + E_k$, $k = k + 1$, and go to step 1,
        where error$_k = \|F(X_k)\|_F$.
```

Lemma 2.2 (*[4]*). *Suppose that X_0 is an $n \times n$ matrix, and*

(i) *$F'(X_0)$ is regular, i.e.* $c_0 = \inf\left\{\left\|F'(X_0)[H]\right\| : \|H\| = 1\right\} > 0$,

(ii) $\|F(X_0)\| = \varepsilon_0 \le \varepsilon = \min\left\{\frac{c_0^2}{4\|A\|}, \frac{c_0}{2}\right\}$.

Then there exists $S \in C^{n\times n}$,such that $F(S) = 0$, $\|S - X_0\| \le \frac{2\varepsilon_0}{c_0}$. More precisely, the sequence $\{X_k\}$ given by Algorithm 2.2 satisfies:

(i) $\lim_{k\to\infty} X_k = S$ *exists with* $F(S) = 0$,

(ii) $\|X_k - S\| \le q\,\|X_{k-1} - S\| \le 2\frac{\varepsilon_0}{c_0}q^k$,

(iii) $\|X_{k+1} - X_k\| \le q\,\|X_k - X_{k-1}\|$ *for* $k = 1, 2, \ldots, q = \frac{4\varepsilon_0\|A\|}{c_0^2} < 1$.

Remark 2.2. The Algorithm 2.2 is only linearly convergence, which means possibly very slow convergence. But at each iterative step, only a linear equation with the same coefficient matrix has to be solved.

Algorithm 2.3 (*Newton's Method with Šamanskii Technique [11]*).

```
step 0: Given $X_0$, $m$ and $\varepsilon$, set $k = 0$.
step 1: If error$_k < \varepsilon$, stop.
step 2: Let $X_{k,0} = X_k$, $i = 1$.
step 3: if $i > m$, go to step 6.
step 4: Solve for $E_{k,i-1}$ in generalized Sylvester equation
```

$$AE_{k,i-1}X_k + (AX_k + B)E_{k,i-1} = -F(X_{k,i-1}). \tag{2.5}$$

```
step 5: Update $X_{k,i} = X_{k,i-1} + E_{k,i-1}$, $i = i + 1$, and go to step 3.
step 6: Update $X_{k+1} = X_{k,m}$, $k = k + 1$, and go to step 1,
        where error$_k = \|F(X_k)\|_F$.
```

Remark 2.3. If $m = 1$, then the Algorithm 2.3 is Newton's method.

In this paper, we only consider the case that $m = 2$. Following we investigate the properties of Algorithm 2.3.

Theorem 2.1. *Suppose that $S \in C^{n\times n}$ is a simple solvent of (1.1), i.e. $F(S) = 0$ and $F'(S)$ is regular. Then, if the starting matrix X_0 is sufficiently near S, the sequence $\{X_k\}$ produced by Algorithm 2.3 converges cubically to S. More precisely, if $\|X_0 - S\| = \varepsilon_0 < \varepsilon = \min\{\frac{c_0}{2\|A\|}, \delta\}$, where,*

$$c_0 = \inf\left\{\left\|F'(X)[H]\right\| : \|H\| = 1, \|X - S\| \le \delta\right\} > 0$$

for sufficient small $\delta \in (0, 1]$, then we have:

(i) $\|X_{k+1} - S\| \le \eta\,\|X_k - S\|^3$ *with* $\eta = \frac{\|A\|^2}{c_0^2}\left(2 + \frac{\|A\|}{c_0}\varepsilon_0\right) \le \frac{3\|A\|^2}{c_0^2}$, $k = 1, 2, \ldots,$

(ii) $\|X_k - S\| \le \varepsilon_0 q^k$, $q = \eta\varepsilon_0^2 < 1$, $k = 1, 2, \ldots,$

(iii) $\lim_{k\to\infty} X_k = S$.

Proof. From the Lemma 2.1, we obtain that $X_{k,1}$ is well defined by

$$X_{k,1} = X_k + E, \qquad AEX_k + (AX_k + B)E = -F(X_k), \tag{2.6}$$
$$X_k \in V = \left\{ X \mid \|X - S\| = \varepsilon_0 < \varepsilon = \min\left\{\delta, \frac{c_0}{2\|A\|}\right\}\right\} \quad \text{and} \quad \left\|X_{k,1} - S\right\| \le \frac{\|A\|}{c_0}\|X_k - S\|^2.$$

Hence, X_{k+1} is well defined by

$$X_{k+1} = X_{k,1} + H, \qquad AHX_k + (AX_k + B)H = -F(X_{k,1}), \tag{2.7}$$

$X_k \in \{X \mid \|X - S\| < \varepsilon_1\} \subset V$, where $\varepsilon_1^2 \le \frac{\varepsilon_0 c_0}{\|A\|}$. By the Lemma 2.1 and the properties of Kronecker product and vec function, we obtain

$$\begin{aligned}
\|X_{k+1} - S\| &= \left\|X_{k,1} + H - S\right\| \\
&= \left\|\mathrm{vec}(X_{k,1} - S) + \mathrm{vec}H\right\| \\
&= \left\|\mathrm{vec}(X_{k,1} - S) - F^*(X_k)^{-1}\mathrm{vec}F(X_{k,1})\right\| \\
&\le \frac{1}{c_0}\left\|\mathrm{vec}F'(X_k)[X_{k,1} - S] - \mathrm{vec}F(X_{k,1})\right\| \\
&= \frac{1}{c_0}\left\|F'(X_k)[X_{k,1} - S] - F(X_{k,1}) + F(S)\right\| \\
&\le \frac{1}{c_0}\|A\|\left\|X_{k,1} - S\right\|\left(2\|X_k - S\| + \left\|X_{k,1} - S\right\|\right) \\
&\le \frac{1}{c_0^2}\|A\|^2\|X_k - S\|^2\left(2\|X_k - S\| + \frac{\|A\|}{c_0}\|X_k - S\|^2\right) \\
&\le \frac{\|A\|^2}{c_0^2}\left(2 + \frac{\|A\|}{c_0}\varepsilon_0\right)\|X_k - S\|^3 \\
&= \eta\|X_k - S\|^3,
\end{aligned}$$

where $F^*(X_k) = X_k^T \otimes A + I \otimes (AX_k + B)$. So the assertion (i) is proved. Now we obtain assertion (ii) by induction. Since $\|X_0 - S\| = \varepsilon_0 \le \varepsilon$ is given, we assume that $\|X_k - S\| \le \varepsilon_0 q^k$ holds. While $q = \varepsilon_0\eta < 1$ is obvious, thus $\|X_k - S\| \le \varepsilon_0$. So we have $\|X_{k+1} - S\| \le \eta\|X_k - S\|^3 \le \eta\varepsilon_0^2\|X_k - S\| = q\|X_k - S\| < \varepsilon_0 q^{k+1}$, hence, the assertion (ii) holds by induction principle. It is obvious that (ii) implies (iii).

Algorithms 2.1 and 2.3 are now compared in terms of computational cost. we know that the heart of the two algorithms is to solve the generalized *Sylvester* equation

$$AEX + (AX + B)E = -F(X). \tag{2.8}$$

To solve (2.8) we can adapt methods described in [5,7] as follows.

(1) Computing Hessenberg-triangular decomposition [9] of A and $AX + B$

$$W^*AZ = T, \qquad W^*(AX + B)Z = H, \quad 15n^3 \textit{ flops}, \tag{2.9}$$

where W and Z are unitary, T is upper triangular and H is upper Hessenberg.

(2) Computing the Schur decomposition [9] of X

$$U^*XU = R, \quad 25n^3 \textit{ flops}, \tag{2.10}$$

where U is unitary and R is upper triangular.

(3) Forming F

$$F = -W^*P(X)U, \quad 4n^3 \textit{ flops}. \tag{2.11}$$

(4) Transforming from Y to E

$$Y = Z^* EU, \quad 4n^3 \textit{ flops}. \tag{2.12}$$

(5) Solving the upper Hessenberg systems

$$(H + r_{kk}T)y_k = f_k - \sum_{i=1}^{k-1} r_{ik}Ty_i, \quad 4n^3 \textit{ flops}. \tag{2.13}$$

Thus, the computational cost that we obtain X_{k+1} from X_k is about $56n^3$ *flops* for Algorithm 2.1. For Algorithm 2.3, in order to obtain X_{k+1} we need to solve two generalized Sylvester equations with the same coefficient matrix. Hence the cost is about $70n^3$ *flops*. Algorithm 2.3 is only $14n^3$ *flops* more than Algorithm 2.1 at each iteration, but the Algorithm 2.3 converges cubically to the solvent S.

Remark 2.4. If the starting matrix X_0 is sufficient near the solvent of Eq. (1.1), then Newton's method with Šamanskii technique has faster convergence than Newton's method. The computational procedure of Newton's method with Šamanskii technique only increases a time of circulation compared to Newton's method and when $m = 1$, it has contained the procedure of Newton's method.

3. Two new algorithms

Algorithm 3.1 (*Newton's Method with Exact Line Searches*).

```
step 0: Given $X_0$, and $\varepsilon$, set $k = 0$.
step 1: If error$_k < \varepsilon$, stop.
step 2: Solve for $E_k$ in generalized Sylvester equation
            $AE_kX_k + (AX_k + B)E_k = -F(X_k).$   (3.1)
step 3: Find $t$ by exact line searches such that
            $\|F(X_k + tE_k)\|_F = \min_{s\in[0,2]} \|F(X_k + sE_k)\|_F$   (3.2)
step 4: Update $X_{k+1} = X_k + tE_k,\ k = k + 1$, and go to step 1,
        where error$_k = \|F(X_k)\|_F$.
```

From [5], we know that Algorithm 3.1 has the global convergence properties and the quadratic convergence rate, but at each iterative step, t, namely, a multiple of the Newton's step, has to be calculated. Observing the Table 1.2, we obtain a question whether t approaches or equal to 1 when the iterative solution X_k is near the solvent S. The answer is yes, under a mild assumption, as we now show. Recalling that Newton's method defines E by

$$AEX + (AX + B)E = -F(X) \tag{3.3}$$

and Newton's method with exact line searches defines t by

$$\|F(X_k + tE_k)\|_F = \min_{s\in[0,2]} \|F(X_k + sE_k)\|_F \tag{3.4}$$

From (3.3), we have

$$F(X + sE) = (1 - s)F(X) + s^2 AE^2. \tag{3.5}$$

Assume that X_j is within a region where quadratic convergence to S and let $X_{j+1} = X_j + E_j$ and $\tilde{X}_{j+1} = X_j + tE_j$ be the Newton update and the update with exact line searches, respectively. Defining $\Delta_j = S - X_j$, So, by Lemma 2.1, we have

$$\|\Delta_{j+1}\| = O(\|\Delta_j\|^2), \tag{3.6}$$

and using (3.5) we have

$$\begin{aligned}\|(1-t)F(X_j)+t^2AE_j^2\| &= \|F(X_j+tE_j)\| \\ &\le \|F(X_j+E_j)\| \\ &= \|F(S-\Delta_{j+1})\| \\ &= O(\|\Delta_j\|^2).\end{aligned} \tag{3.7}$$

Thus $E_j = -\Delta_{j+1}+\Delta_j$, which means that $\|E_j\| = O(\|\Delta_j\|)$ and, Therefore (3.3) says

$$F(X_j) = F(S-\Delta_j) = -D_S(\Delta_j) + O(\|\Delta_j\|^2),$$

where $D_S(\Delta_j) = A\Delta_j S + (AS+B)\Delta_j$. Hence, as long as the Frechet derivative is nonsingular at S, (3.7) implies that $|1-t| = O(\|\Delta_j\|)$. So as long as X_j is sufficient near the solvent S, that is $\|\Delta_j\| \to 0$, then $t \to 1$. In practice, it is difficult to know in advance the solvent S of Eq. (1.1), hence, we use $\|F(X_i)\| \le \varepsilon_0$ to replace $\|\Delta_j\| \to 0$ for convenience. Upon all above analysis, we have the following algorithm.

Algorithm 3.2 (*No. 1*).

```
step 0: Given $X_0$, $\varepsilon_0$ and $\varepsilon$, set $k = 0$.
step 1: If error$_k < \varepsilon$, stop.
step 2: Solve for $E_k$ in generalized Sylvester equation
            $AE_kX_k + (AX_k+B)E_k = -F(X_k).$    (3.8)
step 3: If error$_k < \varepsilon_0$, go to step 6.
step 4: Find $t$ by exact line searches such that
            $\|F(X_k+tE_k)\|_F = \min_{s\in[0,2]} \|F(X_k+sE_k)\|_F$    (3.9)
step 5: Update $X_{k+1} = X_k + tE_k$, $k = k+1$, and go to step 1.
step 6: Update $X_{k+1} = X_k + E_k$, $k = k+1$, and go to step 1,
        where error$_k = \|F(X_k)\|_F$.
```

Clearly Algorithm 3.2 not only keeps the global convergence properties and the quadratic convergence rate, but also need not to compute t when error$_k < \varepsilon_0$.

From Theorem 2.1, we know that Newton's method with Šamskii technique ($m = 2$) converges cubically to the solvent S, so long as the starting matrix X_0 satisfying the conditions $\|X_0 - S\| = \varepsilon_0 < \varepsilon = \min\{\frac{c_0}{2\|A\|}, \delta\}$, *where* $c_0 = \inf\{\left\|F'(X)[H]\right\| : \|H\| = 1, \|X-S\| \le \delta\} > 0$ for sufficient small $\delta \in (0,1]$. Hence, by Algorithms 3.1 and 2.3, we get the following algorithm.

Algorithm 3.3 (*No. 2*).

```
step 0: Given $X_0$, $\varepsilon_0$ and $\varepsilon$, set $k = 0$.
step 1: If error$_k < \varepsilon$, stop.
step 2: Solve for $E_k$ in generalized Sylvester equation
            $AE_kX_k + (AX_k+B)E_k = -F(X_k).$    (3.10)
step 3: If error$_k < \varepsilon_0$, go to step 6.
step 4: Find $t$ by exact line searches such that
            $\|F(X_k+tE_k)\|_F = \min_{s\in[0,2]} \|F(X_k+sE_k)\|_F$    (3.11)
step 5: Update $X_{k+1} = X_k + tE_k$, $k = k+1$, and go to step 1.
step 6: Update $X_{k,1} = X_k + E_k$.
```

Table 4.1
Algorithm 2.1

$n = 20$		$n = 50$	
$s = 1$	$\text{error}_1 = 1.1291\text{e}+004$	$s = 1$	$\text{error}_1 = 1.7853\text{e}+004$
$s = 5$	$\text{error}_5 = 43.3420$	$s = 5$	$\text{error}_5 = 68.8583$
$s = 8$	$\text{error}_8 = 0.3885$	$s = 8$	$\text{error}_8 = 0.6346$
$s = 9$	$\text{error}_9 = 0.0258$	$s = 9$	$\text{error}_9 = 0.0424$
$s = 10$	$\text{error}_{10} = 1.5401\text{e}-004$	$s = 10$	$\text{error}_{10} = 2.5560\text{e}-004$
$s = 11$	$\text{error}_{11} = 5.7274\text{e}-009$	$s = 11$	$\text{error}_{11} = 9.5571\text{e}-009$
$s = 12$	$\text{error}_{12} = 1.4282\text{e}-015$	$s = 12$	$\text{error}_{12} = 2.2820\text{e}-015$

Table 4.2
Algorithm 3.1

$n = 20$			$n = 50$		
$s_1 = 1$	$t = 1.9849$	$\text{error}_1 = 5.3244$	$s_1 = 1$	$t = 1.9872$	$\text{error}_1 = 6.3133$
$s_1 = 2$	$t = 0.5109$	$\text{error}_2 = 0.7510$	$s_1 = 2$	$t = 0.4331$	$\text{error}_2 = 0.7949$
$s_1 = 3$	$t = 1.1099$	$\text{error}_3 = 0.0330$	$s_1 = 3$	$t = 0.9954$	$\text{error}_3 = 0.0690$
$s_1 = 4$	$t = 1.0066$	$\text{error}_4 = 7.4418\text{e}-005$	$s_1 = 4$	$t = 1.0079$	$\text{error}_4 = 6.3845\text{e}-005$
$s_1 = 5$	$t = 1.0000$	$\text{error}_5 = 7.4663\text{e}-010$	$s_1 = 5$	$t = 1.0000$	$\text{error}_5 = 5.1195\text{e}-010$
$s_1 = 6$	$t = 1.0000$	$\text{error}_6 = 2.2122\text{e}-015$	$s_1 = 6$	$t = 1.0000$	$\text{error}_6 = 2.2820\text{e}-015$

step 7: Solve for $E_{k,1}$ in generalized *Sylvester* equation

$$AE_{k,1}X_k + (AX_k + B)E_{k,1} = -F(X_{k,1}). \tag{3.12}$$

step 8: Update $X_{k+1} = X_{k,1} + E_{k,1}$, $k = k + 1$, and go to step 1,
where $\text{error}_k = \|F(X_k)\|_F$.

We know that Newton's method with exact line searches needs $61n^3$*flops* and Algorithm 3.3 needs $70n^3$*flops* as error $< \varepsilon_0$ for each iteration. Hence, Algorithm 3.3 is only $9n^3$ *flops* more than Algorithm 3.1 as error $< \varepsilon_0$ for each iterative step, while $9n^3$ *flops* may be ignored compared with $61n^3$ *flops*. It is obvious that Algorithm 3.3 keeps the global convergence properties of Algorithm 3.1 but has faster convergence rate than Algorithm 3.1 by Remark 2.4.

4. Numerical results

In this section, we use several numerical examples to illustrate our results. The following notations will be used in this section, $s = i, s_1 = i$ and $s_2 = i$ denote the ith iteration of Newton's method, Newton's method with exact line searches, Newton's method with Šamanskii technique, respectively. X_i is the ith iterative solution. $\text{error}_i = \|P(X_i)\|_F$, X denotes the iterative solution when $\|P(X)\|_F \le \varepsilon$ and error $= \|P(X)\|_F$. CPU denotes the computation time.

Example 4.1. Consider the example mentioned in [14], $AX^2 + BX + C = 0$, with

$$A = I_n, \qquad B = I_n, \qquad C = -(H^2 + H),$$

where $H = (h_{ij}), h_{ij} = 1/(i + j - 1)$.

The Tables 4.1–4.4 list the numerical result, $\varepsilon_0 = 0.1$, $\varepsilon = 10^{-11}$. Starting the iteration with $X_0 = 100I_n$.

Observing the Tables 4.1–4.4, we know that Algorithm 2.1 needs the most flops and iteration numbers than other algorithms. Algorithm 3.1 is better than Algorithm 2.1, since it needs less iteration numbers than Algorithm 2.1, but it needs to compute t at each iteration step. Algorithm 3.2 keeps the merits of Algorithm 3.1 and does not need to compute t when error $< \varepsilon_0$. Algorithm 3.3 needs the fewest iteration numbers than others.

Table 4.3
Algorithm 3.2

$n = 20$			$n = 50$		
$s_1 = 1$	$t = 1.9849$	$\text{error}_1 = 5.3244$	$s_1 = 1$	$t = 1.9872$	$\text{error}_1 = 6.3133$
$s_1 = 2$	$t = 0.5109$	$\text{error}_2 = 0.7510$	$s_1 = 2$	$t = 0.4331$	$\text{error}_2 = 0.7949$
$s_1 = 3$	$t = 1.1099$	$\text{error}_3 = 0.0330$	$s_1 = 3$	$t = 0.9954$	$\text{error}_3 = 0.0690$
$s = 1$		$\text{error}_4 = 2.2614\text{e}{-004}$	$s = 1$		$\text{error}_4 = 5.3830\text{e}{-004}$
$s = 2$		$\text{error}_5 = 1.2829\text{e}{-008}$	$s = 2$		$\text{error}_5 = 3.6341\text{e}{-008}$
$s = 3$		$\text{error}_6 = 2.2281\text{e}{-015}$	$s = 3$		$\text{error}_6 = 2.2953\text{e}{-015}$

Table 4.4
Algorithm 3.3

$n = 20$			$n = 50$		
$s_1 = 1$	$t = 1.9849$	$\text{error}_1 = 5.3244$	$s_1 = 1$	$t = 1.9872$	$\text{error}_1 = 6.3133$
$s_1 = 2$	$t = 0.5109$	$\text{error}_2 = 0.7510$	$s_1 = 2$	$t = 0.4331$	$\text{error}_2 = 0.7949$
$s_1 = 3$	$t = 1.1099$	$\text{error}_3 = 0.0330$	$s_1 = 3$	$t = 0.9954$	$\text{error}_3 = 0.0690$
$s_2 = 1$		$\text{error}_4 = 3.3543\text{e}{-006}$	$s_2 = 1$		$\text{error}_4 = 8.6442\text{e}{-006}$
$s_2 = 2$		$\text{error}_5 = 1.7511\text{e}{-015}$	$s_2 = 2$		$\text{error}_5 = 2.7540\text{e}{-015}$

Table 4.5
$\varepsilon_0 = 10^{-1}$

Algorithm	2.1		3.1		3.2			3.3		
X_0	s	Error	s_1	Error	s_1	s	Error	s_1	s_2	Error
iI	8	1.4879e−14	6	5.5123e−14	4	2	3.2015e−13	5	1	3.1657e−14
$10^5 iI$	20	2.2907e−14	6	3.9046e−14	4	2	1.5796e−14	5	1	4.4408e−14
$10^{10} iI$	37	4.8876e−14	7	2.8032e−14	5	2	7.3893e−14	5	2	1.8589e−14

Example 4.2. Consider the example mentioned in [5], $AX^2 + BX + C = 0$, with

$$A = \begin{pmatrix} 17.6 & 1.28 & 2.89 \\ 1.28 & 0.84 & 0.413 \\ 2.89 & 0.413 & 0.725 \end{pmatrix}, \qquad B = \begin{pmatrix} 7.66 & 2.45 & 2.1 \\ 0.23 & 1.04 & 0.223 \\ 0.6 & 0.756 & 0.658 \end{pmatrix}, \qquad C = \begin{pmatrix} 121 & 18.9 & 15.9 \\ 0 & 2.7 & 0.145 \\ 11.9 & 3.64 & 15.5 \end{pmatrix}.$$

Applying Algorithms 2.1 and 3.1–3.3 for $X_0 = 10^j \cdot iI, , i = \sqrt{-1}, j = 0, 5, 10$ respectively. The results are given in Table 4.5.

Example 4.3. Consider the equation $AX^2 + BX + C = 0$ with $A, B, C \in R^{n\times n}$,

$$A = I_n, \qquad B = \begin{pmatrix} 20 & -10 & & & \\ -10 & 30 & -10 & & \\ & -10 & \ddots & \ddots & \\ & & \ddots & 30 & -10 \\ & & & -10 & 20 \end{pmatrix}, \qquad C = \begin{pmatrix} 15 & -5 & & & \\ -5 & 15 & -5 & & \\ & -5 & \ddots & \ddots & \\ & & \ddots & 15 & -5 \\ & & & -5 & 15 \end{pmatrix}.$$

It comes from a damped mass–spring system. We take $n = 50, 100, 150$ and solve the equation by Algorithms 3.1–3.3, respectively. The Tables 4.6–4.8 list the numerical results. Let $\varepsilon_0 = 10^{-1}$ or 10, $\varepsilon = 10^{-12}$. Starting the iteration with $X_0 = 10^5 I_n$.

It can be seen from the Tables 4.6–4.8 that Algorithms 3.2 and 3.3 are better than Algorithms 2.1 and 3.1.

Table 4.6
$n = 50$

Algorithm	s_1	s_2	s	Error	CPU	
2.1	0	0	19	2.9565e−014	0.365935	
3.1	7	0	0	2.2575e−014	0.171445	
3.2	4	0	3	2.0640e−014	0.158533	$\varepsilon_0 = 10$
3.3	5	1	0	4.2407e−014	0.148978	$\varepsilon_0 = 0.1$

Table 4.7
$n = 100$

Algorithm	s_1	s_2	s	Error	CPU	
2.1	0	0	19	4.9027e−014	3.482728	
3.1	7	0	0	2.8602e−014	1.705694	
3.2	4	0	3	2.1056e−014	1.683122	$\varepsilon_0 = 10$
3.3	5	1	0	6.2580e−014	1.509677	$\varepsilon_0 = 0.1$

Table 4.8
$n = 150$

Algorithm	s_1	s_2	s	Error	CPU	
2.1	0	0	19	5.8728e−014	10.466781	
3.1	7	0	0	3.4548e−014	4.754391	
3.2	4	0	3	2.8255e−014	4.683742	$\varepsilon_0 = 10$
3.3	5	1	0	7.5513e−014	4.397326	$\varepsilon_0 = 0.1$

5. Conclusions

Higham and Kim obtained better theoretical and numerical results to solve quadratic matrix equation by Newton's method with exact line searches than by Newton's method in [5]. In this paper we analyzed their algorithms and obtained that when the iterative solution is near the solvent S such that $F(S) = 0$, it is not necessary to use exact line searches. Hence our Algorithm 3.2 need less cost of computation than Algorithm 3.1. Our Algorithm 3.3 first used Šamanskii technique [11], which is often used to structure high-order algorithm to solve nonlinear equations, to solve quadratic matrix equation and obtained better theoretical and numerical results.

Acknowledgements

The authors thank the anonymous referees for their helpful comments which improved the paper.

References

[1] G.J. Davis, Numerical solution of a quadratic matrix equation, SIAM J. Sci. State. Comput. 2 (1981) 164–175.
[2] G.J. Davis, Algorithm 598: An algorithm to compute solvents of the matrix equation $AX^2 + BX + C = 0$, ACM Trans. Math. Software 9 (1983) 246–254.
[3] E. Stickel, On the Fréchet derivative of matrix functions, Linear Algebra Appl. 91 (1987) 83–88.
[4] W. Kratz, E. Stickel, Numerical solution of matrix polynomial equations by Newton's methods, IMA J. Numer. Anal. 7 (1987) 355–369.
[5] N.J. Higham, H.M. Kim, Solving a quadratic matrix equation by Newton's method with exact line searches, SIAM J. Matrix Anal. Appl. 23 (2001) 303–316.
[6] N.J. Higham, H.M. Kim, Numerical analysis of a quadratic matrix equation, IMA J. Numer. Anal. 20 (2000) 499–519.
[7] J.D. Gardiner, A.J. Laub, J.J. Amato, C.B. Moler, Solution of the *sylvester* matrix equation $AXB^T + CXD^T = E$, ACM Trans. Math. Software 18 (1992) 223–231.
[8] P. Lancaster, Lambda-Matrices and Vibrating Systems, Pergamon Press, Oxford, UK, 1966.
[9] G.H. Golub, C.F. Van loan, Matrix Computations, third ed., The Johns Hopkins University Press, 1996.
[10] J.E. Dennis Jr., F. Traub, R.P. Weber, Algorithms for solvents of matrix polynomials, SIAM J. Numer. Anal. 15 (1978) 523–533.
[11] V. Šamanskii, On a modification of the Newton method, Uhrain Mat. Z. 19 (1967) 133–138 (in Russian).
[12] F. Tisseur, K. Meerbergen, The quadratic eigenvalue problem, SIAM Rev. 43 (2001) 235–286.
[13] I. Gohberg, P. lancaster, L. Rodman, Matrix Polynomials, Academic Press, New York, 1982.
[14] H.M. Kim, Numerical methods for solving a quadratic matrix equation, Ph.D. Thesis, Manchester University, 2000.