Numer Algor (2013) 63:727–752
DOI 10.1007/s11075-012-9650-2

ORIGINAL PAPER

Large-scale Stein and Lyapunov equations, Smith method, and applications

Tiexiang Li · Peter Chang-Yi Weng · Eric King-wah Chu · Wen-Wei Lin

Received: 23 November 2011 / Accepted: 17 September 2012 / Published online: 13 October 2012

Abstract We consider the solution of large-scale Lyapunov and Stein equations. For Stein equations, the well-known Smith method will be adapted, with $A_k = A^{2^k}$ not explicitly computed but in the recursive form $A_k = A_{k-1}^2$, and the fast growing but diminishing components in the approximate solutions truncated. Lyapunov equations will be first treated with the Cayley transform before the Smith method is applied. For algebraic equations with numerically low-ranked solutions of dimension n, the resulting algorithms are of an efficient $O(n)$ computational complexity and memory requirement per iteration and converge essentially quadratically. An application in the estimation of a lower bound of the condition number for continuous-time algebraic Riccati equations is presented, as well as some numerical results.

Keywords Krylov subspace · Lyapunov equation · Smith method · Stein equation

T. Li
Department of Mathematics, Southeast University, Nanjing 211189, People's Republic of China
e-mail: txli@seu.edu.cn

P. C.-Y. Weng · E. K.-w. Chu (✉)
School of Mathematical Sciences, Monash University, Building 28, Clayton, VIC 3800, Australia
e-mail: eric.chu@monash.edu

P. C.-Y. Weng
e-mail: peter.weng@monash.edu

W.-W. Lin
Department of Applied Mathematics, National Chiao Tung University, Hsinchu 300, Taiwan
e-mail: wwlin@am.nctu.edu.tw

Mathematics Subject Classifications (2010) 15A24 · 65F50 · 93C05

1 Introduction

Consider the large-scale Stein equation

$$\mathcal{S}(X) \equiv -X + A^\top XA + H = 0, \quad H = CT^{-1}C^\top, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is d-stable (with eigenvalues inside the unit circle), $C \in \mathbb{R}^{n\times l}$ and the symmetric $T^{-1} \in \mathbb{R}^{l\times l}$. We assume that $l \ll n$ and A is large and sparse(-like) or banded, requiring $c_a^s mn$ flops to form the products AZ or $A^\top Z$, where $Z \in \mathbb{R}^{n\times m}$ and c_a^s is independent of n.

Similarly, consider the large-scale Lyapunov equation

$$\mathcal{L}(X) \equiv A^\top X + XA + H = 0, \quad H = CT^{-1}C^\top, \tag{2}$$

where $A \in \mathbb{R}^{n\times n}$ is c-stable (with eigenvalues on the left half-plane). Again, we assume that $l \ll n$ and A is large and sparse(-like) or banded, requiring $c_a^l mn$ flops to evaluate $A^{-1}Z$ or $A^{-\top}Z$, where $Z \in \mathbb{R}^{n\times m}$ and c_a^l is a constant independent of n.

We are looking for efficient algorithms for the symmetric solutions X to (1) and (2), with $O(n)$ computational complexity and memory requirement per iteration. Without utilizing the properties of A and H, the state-of-the art algorithms for these equations [12] are based on techniques for dense matrices, requiring $O(n^3)$ flops and $O(n^2)$ memory storage. They will not be feasible for large systems.

A variety of applications in systems and control theory, such as stability analysis, balanced truncation model order reduction, solution of algebraic Riccati equations (by Newton's method) in optimal control, H_∞ control, system identification, game theory, filtering and image restoration, involve the Stein and Lyapunov equations. Large systems [1–3] arise from the (boundary) control problems modelled by partial differential equations, as in the cooling of steel profiles, circuit design, VLSI computer-aided design and large mechanical space structures.

We are motivated by the pioneering work by Benner, Fassbender and Saak in model order reduction and the solution of large-scale algebraic Riccati equations (AREs), as well as the work by others in large-scale AREs and algebraic matrix equations [4, 6, 7, 26, 27, 30]. For large-scale AREs, the application of Newton's method leads to large-scale Stein/Lyapunov equations, on which Galerkin projection or Krylov subspace based methods were applied.

In this paper, we shall apply the well-known Smith method (SM) [29], with modifications for efficiency, in the same spirit as the low-rank squared Smith method in [13, 25]. For equations with numerically low-ranked solutions, which we shall define later, we attain $O(n)$ computational complexity and memory requirement for the resulting algorithms. Some interesting new insights into the link of Krylov subspaces and the SM reveal the fast convergence and efficiency of our SM. An application in the estimation of condition of algebraic Riccati equations will also be discussed.

Our methods look very much like the low-rank Smith methods in [13, 25] but are substantially different. We have not applied their ADI approach in rewriting Lyapunov equations into Stein equations, with the associated difficulty in estimating the shift parameters. The corresponding accelerated convergence is evidently not as fast as the original SM, as shown in the numerical results from our algorithms. Also, we shall construct our Krylov subspaces implicitly through our iterations, contrasting the explicit Arnoldi process in [25].

In the paper, $\|\cdot\|$ denotes the 2-norm and $\rho(\cdot)$ is the spectral radius.

1.1 Main contributions

The main contributions of this paper are as follows. We shall formalize the discussion on the numerical rank of the solution X to Stein and Lyapunov equations, showing constructively when X is numerically low-ranked. Then we shall show that we can adapt the well-known Smith method efficiently for large-scale Stein and Lyapunov equations. Finally, we shall apply our techniques to the estimation of lower bounds of condition numbers for continuous-time algebraic Riccati equations.

2 Large-scale Stein equations

The original SM [29] considered the (continuous-time) Sylvester equation $XA + BX = C$, transforming by Cayley transform to the discrete-time Sylvester equation $X - UXV = W$. We shall adapt the method for (1). See also the convergence analysis and the posteriori error bound in [29, Sections 3 and 4].

2.1 Smith method

For the Stein equation

$$X = f(X) \equiv A^\top XA + H, \tag{3}$$

consider the functional iteration

$$X_{k+1} = f(X_k) = f^2(X_{k-1}) = \cdots = f^{k+1}(X_0).$$

Alternatively, substitute $X = f(X)$ into the right-hand-side of (3), we obtain

$$X = f^2(X) \equiv f(A^\top XA + H) = (A^2)^\top XA^2 + A^\top HA + H. \tag{4}$$

Next substituting the expression for X in (4) into the right-hand-side of itself, we obtain

$$X = f^{2^2}(X) \equiv (A^{2^2})^\top X A^{2^2} + \sum_{i=0}^{2^2-1} (A^i)^\top H A^i.$$

Repeating the process and substitute the most up-to-date expression $X = f^{2^k}(X)$ back into the right-hand-side of itself, we have

$$X = (A^{2^{k+1}})^\top X A^{2^{k+1}} + \sum_{i=0}^{2^{k+1}-1} (A^i)^\top H A^i. \tag{5}$$

With A being d-stable, we have $A^{2^k} \to 0$ quadratically as $k \to \infty$, and

$$X = \lim_{k\to\infty} H_k, \quad H_k \equiv \sum_{i=0}^{2^k-1} (A^i)^\top H A^i. \tag{6}$$

From (5) and (6), one resulting algorithm will be (for $k \geq 0$)

$$A_{k+1} = A_k^2, \quad H_{k+1} = H_k + A_k^\top H_k A_k, \tag{7}$$

with $A_0 = A$ and $H_0 = H$. Similar to the more general case in [29, Section 2], we have $A_k = A_{k-1}^2 = A^{2^k}$ $(k \geq 1)$, and $H_k \to X$ as $k \to \infty$.

For the large-scale Stein equation (1), we can organize the iterations as follows:

$$H_{k+1} = H_k + A_k^\top H_k A_k = C_{k+1} T_{k+1} C_{k+1}^\top, \tag{8}$$

with

$$C_{k+1} = \left[C_k,\ A_k^\top C_k\right], \quad T_{k+1} = T_k \oplus T_k \tag{9}$$

and the initial values

$$A_0 = A, \quad C_0 = C, \quad T_0 = T^{-1}. \tag{10}$$

A simple error analysis is quoted from [29, Section 2]. As A is d-stable, there exist constants M and λ such that $\rho(A)^2 = \lambda < 1$ and

$$\|A^k\| \cdot \|(A^\top)^k\| \leq M\lambda^k. \tag{11}$$

Consequently, we have

$$\|H_k - X\| \leq M\|H\| \cdot \frac{\lambda^{2^k}}{1-\lambda}, \quad (k \geq 0). \tag{12}$$

From the above convergence result, we next define the numerical rank of the solution X:

Definition 2.1 For a given tolerance $\tau > 0$, the numerical rank of X with respect to τ, $\mathrm{rank}_\tau X$, is defined as the rank of H_k where k is the smallest integer such that

$$\|H_k - X\| \le \tau.$$

From (6), we have

$$\mathrm{rank}_\tau X = \mathrm{rank} H_k \le \mathrm{rank} C_k \le 2^k l, \tag{13}$$

where the equalities hold generically. Note that the right-hand-side in (13) is the number of columns in C_k. Orthogonalization of C_k eliminates linear dependence in its columns, also suggesting to the possibility of compressing C_k further by truncating small but fast growing components, as in Section 2.2 later.

We then define the concept of numerically low rank:

Definition 2.2 The solution $X \in \mathbb{R}^{n\times n}$ to the large-scale Stein equation (1) is said to be numerically low-ranked with respect to the numerical rank tolerance $\tau > 0$ if $\mathrm{rank}_\tau X \le c_\tau$ for a constant c_τ independent of n. In other words, for a given $\tau > 0$ and relative to n, we have $\mathrm{rank}_\tau X = O(1)$.

We then have he following result for the SM applied on the large-scale Stein equation (1):

Theorem 2.1 *Assume the Smith method converges after k iterations, according to some criterion. For a given tolerance $\tau > 0$, the solution X is numerically low-ranked if, relative to n,*

$$2^k = O(1). \tag{14}$$

Generically, the sufficient condition (14) *is also necessary.*

Proof From Definitions 2.1 and 2.2 as well as (13), it is easily seen that the statement in the Theorem holds. □

Remark 2.1 The result in Theorem 2.1 seems so obvious, almost a trick of syntax. However, previous work mostly neglect to define "numerical rank" or the meaning of being "numerically low-ranked". Many have their methods built on the basis of a plot of singular values of X for some given examples. This obscures the associated discussion and theoretical development for large-scale problems. Also, under our new framework of numerical rank, it is easy but important to recognize that if the SM fails to achieve a high enough accuracy within reasonable number of iterations, the solution X is obviously not numerically low-ranked. In such circumstances, we have to lower our expectation on accuracy, accepting whatever approximate solution we can compute within our computing means, in terms of CPU time or memory resources. Equivalently, we see from (7) that the SM for Stein equations is a competition between the quadratic convergence of $A_k = A^{2^k}$ towards zero and the exponential growth in the width of C_k. The latter may be controlled, as suggested in Section 2.2 later. Another vital consequence is that the growth in C_k is a limiting factor on the accuracy of X. If the convergence of A_k is slow, we may have to restrict the width of C_k, thus limiting the size of the corresponding Krylov subspace for the approximate solution H_k (see Sections 2.4 and 2.5), lowering its accuracy.

2.2 Compression and truncation of C_k

The truncation and compression process described in this section is unnecessary in most circumstances. It is described for the situation when the convergence of the SM is slow in comparison with the exponential growth in the dimensions of the iterates H_k. In this situation, the numerical rank of X will be high and we obviously cannot achieve high accuracy in the approximation H_k of X by *any* method. We then have to compromise its accuracy for the sake of less memory and CPU-time consumption. We should then either choose a larger tolerance for the truncation and compression process (τ below), to control the growth in the iterates and adjust it until the accuracy of the approximate solution is acceptable, or simply abandon the truncation and compression process and accept whatever approximate solution obtained within reasonable time.

Obviously from (9) and the fact that $A_k = A^{2^k}$ with the d-stable A, as the SM converges, increasingly smaller low-ranked components are added to C_k. Apparently, the growth in the sizes and ranks of these iterates is potentially exponential.

To reduce the dimensions of C_k, necessary not just when dependencies arise or to control the associated round-off errors, we shall compress their columns by orthogonalization. Consider the QR decomposition with column pivoting:

$$C_k = Q_k M_k + \widetilde{Q}_k \widetilde{M}_k, \quad \|\widetilde{M}_k\| \le \tau,$$

where $[Q_k, \widetilde{Q}_k]$ is unitary and $[M_k^\top, \widetilde{M}_k^\top]^\top$ is upper triangular. Here τ is some small tolerance controlling the compression and truncation process, n_k is the number of columns in C_k bounded from above by some corresponding $l_{\max}$, and

$$\operatorname{rank} C_k \le n_k \le l_{\max} \ll n.$$

We have $Q_k \in \mathbb{R}^{n\times r_k}$ with orthogonal columns and $M_k \in \mathbb{R}^{r_k\times n_k}$ being full-ranked and upper triangular. Consequently, we have

$$C_k T_k C_k^\top = Q_k \left(M_k T_k M_k^\top \right) Q_k^\top + O(\tau), \tag{15}$$

and we should replace C_k and T_k by the lower-ranked Q_k and $M_k T_k M_k^\top$, respectively. As a result, we ignore the $O(\tau)$ term, control the growth of C_k while sacrificing a hopefully negligible bit of accuracy. Equivalently, we may also control the width of C_k, now relabelled $l_k = r_k$ after the compression and truncation in (15), by setting a reasonable upper limit $l_{\max}$.

2.3 Residuals and convergence control

Consider the difference of successive iterates, with

$$dH_k \equiv H_{k+1} - H_k = A_k^\top C_k T_k C_k^\top A_k = \widetilde{C}_{k+1} \widetilde{T}_{k+1} \widetilde{C}_{k+1}^\top, \tag{16}$$

and

$$\widetilde{C}_{k+1} \equiv A_k^\top C_k, \quad \widetilde{T}_{k+1} \equiv T_k.$$

For the residual $r_k \equiv \|\mathcal{S}(H_k)\|$ of the Stein equation, the corresponding relative residual, commonly used for convergence control, equals

$$\widetilde{r}_k \equiv \frac{r_k}{\|H_k\| + \|\breve{M}_k\| + \|H\|}, \quad \breve{M}_k \equiv A^\top H_k A.$$

We then have

$$\mathcal{S}(H_k) = -H_k + A^\top H_k A + H = \widehat{C}_k \widehat{T}_k \widehat{C}_k^\top,$$

with

$$\widehat{C}_k \equiv \left[C_k,\ A^\top C_k,\ C\right], \quad \widehat{T}_k \equiv (-T_k) \oplus T_k \oplus T^{-1}.$$

For relative error estimates and residuals, we also need the norms of

$$H_k = C_k T_k C_k^\top, \quad H = C T^{-1} C^\top, \quad \breve{M}_k = A^\top C_k \breve{T}_k C_k^\top A,$$

with a different notation $\breve{T}_k = T_k$ in $\breve{M}_k$. Note that T_k in H_k and $\breve{T}_k$ in $\breve{M}_k$ will be different after the subsequent orthogonalization of C_k and $A^\top C_k$, respectively, analogous to (15). All the calculations in this subsection involve the norms of similar low-rank symmetric matrices. For example for H_k, we can proceed as in (15), where we orthogonalize C_k and transform T_k. With the orthogonal $\widetilde{C}_k$, $\widehat{C}_k$, C_k, C and $A^\top C_k$, and the transformed $\widetilde{T}_k$, $\widehat{T}_k$, T_k, T^{-1} and $\breve{T}_k$ respectively, we have the efficient formulae, for the 2- and F-norms,

$$\begin{aligned} &\|dH_k\| = \|\widetilde{T}_{k+1}\|, \quad r_k = \|\widehat{T}_k\|, \\ &\|H_k\| = \|T_k\|, \quad \|H\| = \|T^{-1}\|, \quad \|\breve{M}_k\| = \|\breve{T}_k\|. \end{aligned} \tag{17}$$

Equations (5) and (7), the corresponding initial conditions (10) and the compression and truncation in (15), with the recursion $A_k = A_{k-1}^2 = A^{2^k}$ and the quantities in (17), constitute our SM for Stein equations. We summarize the algorithm for large-scale Stein equations in Algorithm 1 below.

We would like to emphasize that care has to be exercised in Algorithm 1, where the multiplications by A_k and $A_k^\top$ are carried out recursively. Otherwise, the computation cannot be realized in $O(n)$ complexity. Similar care has to be taken in the computation of residuals (used in Algorithm 1) or differences of iterates (as an alternative convergence control), as discussed in this subsection.

Note that the recursive multiplication by A_k (or its inverse for Lyapunov equations later) is just one way to produce the operation counts in Section 6. Alternatively, we may be able to achieve similar efficiency by using cheap memory wisely. For example if A is banded, we may be able to compute and store A_k. Assuming that the band-width in A_k doubles in every iteration, a similar operation count can be obtained. This will be cheaper for really large problems, when the communication costs of recursive multiplications by A_k and $A_k^\top$ dominate.

Algorithm 1 SM for large-scale Stein equations

```
Input:  $A \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{n\times l}$, $T^{-1} = T^{-\top} \in \mathbb{R}^{l\times l}$, positive tolerances $\tau$ and $\epsilon$,
        and $l_{\max}$;
Output: $C_\epsilon \in \mathbb{R}^{n\times l_\epsilon}$ and $T_\epsilon = T_\epsilon^\top \in \mathbb{R}^{l_\epsilon\times l_\epsilon}$, with $C_\epsilon T_\epsilon C_\epsilon^\top$ approximating the
        solution $X$;
        Set $k = 0, \widetilde{r}_0 = 2\epsilon$; $A_0 = A$, $C_0 = C$ and $T_0 = T^{-1}$;
        Compute $h = \|H\| = \|CT^{-1}C^\top\|$;
        Do until convergence:
           If the relative residual $\tilde{r}_k = |d_k/(h_k + m_k + h)| < \epsilon$,
              Set $C_\epsilon = C_k$ and $T_\epsilon = T_k$;
              Exit
           End If
           Compute $C_{k+1} = [C_k,\ A_k^\top C_k]$, $T_{k+1} = T_k \oplus T_k$, with $A_{k+1} = A_k^2$;
           Compress $C_{k+1}$, using the tolerance $\tau$, and modify $T_{k+1}$ as in (15);
           Compute $k \leftarrow k+1$, $d_k = \|\mathcal{S}(H_k)\|$, $h_k = \|H_k\|$ and $m_k = \|A_k^\top H_k A_k\|$,
              as in Section 2.3;
        End Do
```

2.4 SM and Krylov subspaces

There is an interesting relationship between the SM and Krylov subspaces. Define, unorthodoxly, the Krylov subspaces

$$\mathcal{K}_k(A, B) \equiv \begin{cases} \text{span}\{B\} & (k = 0), \\ \text{span}\{B, AB, A^2B, \cdots, A^{2^k-1}B\} & (k > 0). \end{cases}$$

From (9), we have

$$C_k \subseteq \mathcal{K}_k(A^\top, C). \tag{18}$$

(We have abused notations, with $V \subseteq \mathcal{K}_k(A, B)$ meaning span$\{V\} \subseteq \mathcal{K}_k(A, B)$.) In other words, the SM is closely related to approximating the solutions X using Krylov subspaces, with additional components vanishing quadratically. However, for problems of moderate size n, C_k becomes full-ranked after a few iterations.

The link between the SM and the Krylov subspaces defined above is important in explaining the fast convergence of the SM. We used to believe the convergence of the SM came solely from the following inequalities from (8):

$$\|dH_k\| \leq \|A_k^\top\| \cdot \|H_k\| \cdot \|A_k\|, \quad dH_k \equiv H_{k+1} - H_k,$$

and the fact that $\|A_k\| \leq \|A\|^{2^k} \to 0$ quadratically as $k \to \infty$. However, we understand now the power of approximation of the Krylov subspaces contributes in (8) as well, creating cancellations and diminishing components in dH_k itself in (16). This is consistent with numerical results from examples associated with A which is barely stable, where the corresponding $A_k \to 0$ slowly but the overall convergence for H_k is much faster.

2.5 Error of SM

We may limit the rank of the approximation to X, trading off the accuracy in the approximate solution for higher efficiency from the SM. Assume that the compression and truncation in (15) create errors of $O(\tau)$ in H_k. It is easy to see from (7)–(9) that errors of the same magnitude will propagate through to H_{k+1}.

Let δH_k and δA_k are the errors in H_k and A_k, created from the compression and truncation of C_k (as described in Section 2.2) or round-off errors respectively. From (7), we have

$$\delta H_{k+1} = \delta H_k + \delta A_k^\top H_k A_k + A_k^\top \delta H_k A_k + A_k^\top H_k \delta A_k + O(\tau^2),$$
$$\delta A_{k+1} = \delta A_k A_k + A_k \delta A_k + \delta A_k^2.$$

Let $\delta_k \equiv \max\{\|\delta H_k\|, \|\delta A_k\|\}$, $a_k \equiv \|A_k\|$ and $\alpha_k \equiv \|H_k\|$, we have

$$\|\delta H_{k+1}\| \le \delta_k\left(1 + 2\alpha_k a_k + a_k^2\right) + O(\tau^2),$$
$$\|\delta A_{k+1}\| \le 2\|A_k\|\|\delta A_k\| + \|\delta A_k\|^2,$$

or

$$\delta_{k+1} \le \max\{1 + a_k(2\alpha_k + a_k),\ 2a_k\} \cdot \delta_k + O(\tau^2),$$

assuming that the round-off errors in δA_k is much smaller in comparison with τ. Recall that $A_k \to 0$ and H_k converges to X. Eventually, a_k converges to zero, α_k converges to a constant and the errors δA_k and δH_k then pass into δA_{k+1} and δH_{k+1} respectively, creating errors of the same order. From our numerical experience, the trade-off between the rank of H_k and the accuracy of the approximate solution to X contributes towards the success of our computation. If the rank grows out of control, unnecessary and insignificant small additions to the iterates overwhelm the computation in terms of flop counts and memory requirement. Limiting the rank will obviously reduce the accuracy of the approximate solution. We found we do not need to experiment much with the tolerances for the compression/truncation and convergence while trying to achieve a balance between accuracy and the feasibility or efficiency of the SM.

We shall also write down the simple error analysis for the truncated SM, in the context of Galerkin methods as in [15–17] and Section 2.4. (See also the Galerkin method in [28], where Krylov subspaces are generated by A as well as A^{-1}, similar to that in (21) later, without the shift γ.) However, there is a hidden difference flowing from the fact that the SM generates automatically the Krylov subspaces in (18), to which C_k, or the approximate solutions H_k, are associated. In most Galerkin methods like the one in [17], the Krylov subspaces have to be generated explicitly by some Arnoldi processes. Both approaches yield small residuals for the matrix equations in different forms. However, the Arnoldi process may not benefit from the diminishing A_k as in the SM.

First, let us consider the following Galerkin method for Stein equations. Let the column spaces of the orthogonal $Z \in \mathbb{R}^{n\times N}$ be some Krylov subspaces associated with the solution of our equation and assume that the approximate solution has the low-rank form $\widetilde{X} \equiv Z\Lambda Z^\top$, with $\Lambda \in \mathbb{R}^{N\times N}$. Assume that $\widetilde{X}$ approximates the solution X to the Stein equation in the sense that the residual $\mathcal{S}(\widetilde{X})$ is small, of order $O(\epsilon)$. As mentioned before, the Krylov subspace spanned by the columns of Z may be the result of an Arnoldi process [14–17] or the SM.

Substituting $\widetilde{X}$ in place of X in (1), we have

$$-Z\Lambda Z^\top + A^\top Z\Lambda Z^\top A + H = O(\epsilon),$$

which leads to the projected or reduced Stein equation in Λ of size N:

$$\widetilde{\mathcal{S}}(\Lambda) \equiv -\Lambda + \widetilde{A}^\top Z\Lambda Z^\top \widetilde{A} + \widetilde{H} = O(\epsilon) \tag{19}$$

with $\widetilde{A} \equiv Z^\top AZ$ and $H \equiv Z^\top HZ$. We next extend Z to the orthogonal matrix $[Z, \widetilde{Z}] \in \mathbb{R}^{n\times n}$. We then have

$$[Z, \widetilde{Z}]^\top \mathcal{S}(\widetilde{X})[Z, \widetilde{Z}] = \begin{bmatrix} \widetilde{\mathcal{S}}(\Lambda) & O(\epsilon) \\ O(\epsilon) & O(\epsilon) \end{bmatrix} = O(\epsilon).$$

In the SM, a particular Λ is produced such that $\widetilde{\mathcal{S}}(\Lambda) = O(\epsilon)$. If the associated Krylov subspaces are known or have been generated from an Arnoldi process, a solution $\widehat{\Lambda}$ to the small projected Stein equation

$$\widetilde{\mathcal{S}}(\Lambda) = 0 \tag{20}$$

can be computed, generating an approximate Galerkin solution $\widehat{X} \equiv Z\widehat{\Lambda}Z$ to the Stein equation in (1). Consequently, $\widetilde{X}$ and $\widehat{X}$ are approximately equal, as they solve two equations differed by at most $O(\epsilon)$. Note that the solvability of (20) (or the stability of $\widetilde{A}$) is inherited from that of (1). Also, the Galerkin method summarized in (20) will be inferior to the SM if the Krylov subspaces are selected inappropriately. Note that many others selected the Krylov subspaces generated arbitrarily by A, its inverse and transpose, by the Arnoldi process. These approaches resulting in subspaces very different from those in (21) by the SM, which also benefits from the diminishing magnitudes of A_k.

3 Large-scale Lyapunov equations

For the Lyapunov equation (2), with A being c-stable, a Cayley transform to the corresponding Stein equation will be required, before the SM in Section 2 can be applied. Consequently, for the Lyapunov equations, we have

$$C_k \subseteq \mathcal{K}_k\big(A_\gamma^{-\top},\ A_\gamma^{-\top}C\big), \quad A_\gamma \equiv A - \gamma I. \tag{21}$$

Note that the Krylov subspaces $\mathcal{K}_k(A^{\pm 1}, C)$ and $\mathcal{K}_k(A^{\pm\top}, C)$ have been used in the solution of continuous-time algebraic Riccati equations (CARE) and Lyapunov equations in [11, 14–23]. From (21), we can see clearly the appropriate choice for Lyapunov equations, as compared to (18) for Stein equations.

We summarize the algorithm for large-scale Lyapunov equations in Algorithm 2 below, modified from Algorithm 1 with different initial A_0, C_0 and T_0. Note that the repeated multiplication by A_γ^{-1} and it transpose can be achieved efficiently through the LU factors of A_γ, computed at the beginning of the algorithm below. In our numerical experiments, sparse matrix inversions (or the solution of corresponding linear systems) were implemented using LU factors from the MATLAB command `lu`. The interesting possibility of inexact inversion will be left for future work.

Algorithm 2 SM for large-scale Lyapunov equations

```
Input:   $A \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{n\times l}$, $T^{-1} = T^{-\top} \in \mathbb{R}^{l\times l}$, shift $\gamma > 0$, positive
         tolerances $\tau$ and $\epsilon$, and $l_{\max}$;
Output:  $C_\epsilon \in \mathbb{R}^{n\times l_\epsilon}$ and $T_\epsilon = T_\epsilon^\top \in \mathbb{R}^{l_\epsilon\times l_\epsilon}$, with $C_\epsilon T_\epsilon C_\epsilon^\top$ approximating
         the solution $X$ to the large-scale Lyapunov equation (2);
         Compute $A_\gamma = A - \gamma I$, and its LU factors;
         Set $k = 0, \widetilde{r}_0 = 2\epsilon$; $C_0 = A_\gamma^{-\top} C$, $T_0 = 2\gamma T^{-1}$, $A_0 = I_n + 2\gamma A_\gamma^{-1}$;
         Compute $h = \|H\| = \|C_0 T_0 C_0^\top\|$;
         Do until convergence:
           If the relative residual $\tilde{r}_k = |d_k/(h_k + m_k + h)| < \epsilon$,
             Set $C_\epsilon = C_k$ and $T_\epsilon = T_k$;
             Exit
           End If
           Compute $C_{k+1} = [C_k,\ A_k^\top C_k]$, $T_{k+1} = T_k \oplus T_k$, with $A_{k+1} = A_k^2$;
           Compress $C_{k+1}$, using the tolerance $\tau$, and modify $T_{k+1}$ as in (15);
           Compute $k \leftarrow k+1$, $d_k = \|\mathcal{S}(H_k)\|$, $h_k = \|H_k\|$ and $m_k = \|A_k^\top H_k A_k\|$,
             as in Section 2.3 (for an equivalent Stein operator $\mathcal{S}$);
         End Do
```

4 Other low-rank Smith methods

In [13], a modified low-rank Smith method for large-scale Lyapunov equations was proposed. The method looks similar to our SM but is actually very different. The Lyapunov equation (2) was first rewritten into a Stein equation, using the ADI approach. Essentially, we can consider that to be a complicated version of our Cayley transform, with its associated difficulty in estimating the shift parameters. The rank of the low-ranked approximation to X seems to be predetermined in [13], contrasting our truncation and compression process in Section 2.2, which is dynamically determined by the diminishing magnitudes of updates dH_k in (16). The supposedly faster convergence is not evident from the numerical simulation in [13]. From our experience, the SM converges to machine accuracy in less than twenty iterations, because of the stability of A or A_γ.

In [25], Sadkane proposed a low-rank Krylov squared Smith method. Apart from the ADI approach in rewriting (2), a block Arnoldi process was employed to construct the Krylov subspaces, contrasting again the implicit dynamic approach in the SM. Note that we restrict the size of the Krylov subspaces, but new components are added in each iteration before the compression and truncation process. Restarts were also employed in [25] when the sizes of the approximating Krylov subspaces were considered to be too large. We believe restarts are unnecessary and wasteful, as indicated in our examples in Section 7. In the SM, the iterates converge faster than λ^{2^k} and restarts give up this advantage, and reset some large k to 0. We have applied the SM to the examples in [25], with much faster convergence to higher accuracies, as summarized in the examples in Section 7.1 below.

5 Estimation of condition numbers for CAREs

Apart from the solution to algebraic Riccati equations using Newton's method, there are many other obvious applications which require the solution of Stein and Riccati equations. In this section, we shall consider the problem of estimating the condition number of a large-scale continuous-time algebraic Riccati equation (CARE).

In [20], the sensitivity of algebraic and differential Riccati equations was considered. In particular, a condition number κ_{CARE} (denoted originally as K) for CAREs

$$A^\top X + XA - XGX + H = 0$$

were investigated. For the estimation of κ_{CARE}, sharp bounds are obtained:

$$K_L \leq \kappa_{\text{CARE}} \leq K_U$$

with

$$K_U = \frac{\|H\| \cdot \|Z_0\| + 2\|A\|\sqrt{\|Z_0\| \cdot \|Z_2\|} + \|G\| \cdot \|Z_2\|}{\|X\|},$$

$$K_L = \frac{\|H\| \cdot \|Z_0\| + 2\|A\| \cdot \|Z_1\| + \|G\| \cdot \|Z_2\|}{3\|X\|},$$

(where K_L absorbs the factor 3 in the denominator, different from [20]) and Z_i obtained from the Lyapunov equations

$$(A - GX)^\top Z_i + Z_i(A - GX) = -X^i, \quad (i = 0, 1, 2). \tag{22}$$

We shall attempt to modify the SM for the solution of (22).

For large-scale systems, we have A being large, sparse(-like) or banded, and the low-ranked $G = BR^{-1}B^\top$. We then need to solve the Lyapunov equations (22) efficiently, preferably in $O(n)$ computational complexity. Substituting an accurate low-ranked approximate solution $\widetilde{H} = \widetilde{C}\widetilde{T}^{-1}\widetilde{C}^\top$ into X in (22), for $i = 1, 2$, we have the large-scale Lyapunov equations

$$(A - G\widetilde{H})^\top Z_i + Z_i(A - G\widetilde{H}) = -H = -\widetilde{H}^i \tag{23}$$

with the low-ranked right-hand-sides involving

$$H = CT^{-1}C^\top = \left(\widetilde{C}\widetilde{T}^{-1}\widetilde{C}^\top\right)^i, \quad (i = 1, 2).$$

This implies that

$$C = \widetilde{C}, \quad T^{-1} = \widetilde{T}^{-1}\ (i = 1), \ \text{ or } \ T^{-1} = \widetilde{T}^{-1}\widetilde{C}^\top\widetilde{C}\widetilde{T}^{-1}\ (i = 2).$$

The matrix operator in (23), instead of A in (2), will be

$$A_c \equiv A - G\widetilde{H} = A - G\widetilde{C}\widetilde{T}^{-1}\widetilde{C}^\top.$$

Note that A_c is in a sparse-plus-low-rank (splr) form. For (23), the replacement of A by A_c in Algorithm 2 only modifies it slightly and does not affect its $O(n)$ computational complexity. Note that the Cayley transform (to an equivalent Stein equation for doubling to be applicable) requires, with the help of the Sherman–Morrison–Woodbury formula (SMWF),

$$\begin{aligned}(A_c - \gamma I)^{-1} &= \left(A_\gamma - G\widetilde{C}\widetilde{T}^{-1}\widetilde{C}^\top\right)^{-1} \\ &= A_\gamma^{-1} + A_\gamma^{-1}G\widetilde{C}\cdot\widetilde{T}^{-1}\left(I_l - \widetilde{C}^\top A_\gamma^{-1}G\widetilde{C}\widetilde{T}^{-1}\right)^{-1}\cdot\widetilde{C}^\top A_\gamma^{-1}, \end{aligned} \tag{24}$$

again in a similar form to A_c. With $v \in \mathbb{R}^n$, note that $A_\gamma^{-\top}v$ and $A_\gamma^{-1}v$, thus $(A_c - \gamma I)^{-1}v$ and $(A_c - \gamma I)^{-\top}v$, can be computed efficiently in $O(n)$ flops. For initial values, we have

$$\begin{aligned} C_0 &= (A_c - \gamma I)^{-\top}C = A_\gamma^{-\top}C + D_0^{(2)}S_0^{-1}\left[D_0^{(1)}\right]^\top C, \quad T_0^{-1} = 2\gamma T^{-1}, \\ A_0 &= I + 2\gamma(A_c - \gamma I)^{-1} = \left(I + 2\gamma A_\gamma^{-1}\right) + 2\gamma D_0^{(1)}S_0^{-1}\left[D_0^{(2)}\right]^\top, \end{aligned}$$

with

$$D_0^{(1)} \equiv A_\gamma^{-1}G\widetilde{C}, \quad D_0^{(2)} \equiv A_\gamma^{-\top}\widetilde{C}, \quad S_0^{-1} \equiv \widetilde{T}^{-1}\left(I_l - \widetilde{C}^\top A_\gamma^{-1}G\widetilde{C}\widetilde{T}^{-1}\right)^{-1}$$

and A_γ in Algorithm 2 replaced with $A_c - \gamma I$ in (24).

For $i = 0$ in (22), notice that the algorithm in (7) has the form

$$A_{k+1} = A_k^2, \quad H_{k+1} = H_k + A_k^\top H_k A_k.$$

With $H_0 = I$, we have

$$H_1 = I + A_0^\top A_0 = I + \left[I + 2\gamma(A_c - \gamma I)^{-\top}\right]\left[I + 2\gamma(A_c - \gamma I)^{-1}\right].$$

It is not hard to deduce, in general, that the corresponding $H_k = I + \sum_{i=0}^{k-1} A_i^\top A_i$ and Z_0 are not low-ranked, although $A_k \to 0$ as $k \to \infty$. After convergence, the estimation of $\|Z_0\| \approx \|H_k\|$ will still be computationally expensive, of $O(n^2)$ computational complexity. Consequently, the estimation of Z_0 may then be vastly more expensive than the solution of the corresponding large-scale CAREs. This makes the estimation of Z_0 for large-scale CAREs nonsensical in practice, until better methods can be found.

Alternatively, Z_0 in (22) can be bounded in terms of $\|Z_1\|$ and $\|Z_2\|$, as in [20]:

$$\frac{\|Z_0\| \cdot \|Z_2\|}{\kappa(X)} \le \|Z_1\|^2 \le \|Z_0\| \cdot \|Z_2\|,$$

giving rise to

$$\frac{\|Z_1\|^2}{\|Z_2\|} \le \|Z_0\| \le \kappa(X) \cdot \frac{\|Z_1\|^2}{\|Z_2\|},$$

and the less sharp bounds

$$\widetilde{K}_L \le \kappa_{\text{CARE}} \le \widetilde{K}_U$$

with

$$\widetilde{K}_U = \frac{\kappa(X)\|H\| \cdot \|Z_1\|^2 + 2\sqrt{\kappa(X)}\|A\| \cdot \|Z_1\| \cdot \|Z_2\| + \|G\| \cdot \|Z_2\|^2}{\|X\| \cdot \|Z_2\|},$$

$$\widetilde{K}_L = \frac{\|H\| \cdot \|Z_1\|^2 + 2\|A\| \cdot \|Z_1\| \cdot \|Z_2\| + \|G\| \cdot \|Z_2\|^2}{3\|X\| \cdot \|Z_2\|}.$$

A large value for one of the norms of Z_0, Z_1 or Z_2 will indicate ill-condition of the CARE.

Unfortunately, we still cannot produce a sharp upper bound in $\widetilde{K}_U$ for large-scale problems. As X is numerically low-ranked, $\|X^{-1}\|$ in $\kappa(X)$ is large, thus producing a large and pessimistic, thus useless, upper bound $\widetilde{K}_U$. Worse still, X is approximated by the low-ranked $\widetilde{H}$, which contains no information of $\|X^{-1}\|$. Anyhow, in most applications for large-scale problems, we are usually interested in detecting ill-condition, thus the lower bound $\widetilde{K}_L$.

It will be interesting to generalize the approach in [20] and this section for large-scale discrete-time algebraic Riccati equations.

6 Operation and memory counts

The operation and memory counts of Algorithms 1 and 2 for the kth iteration are summarized in Table 1 below. In the second column, we assume that $l \ll n$ and A is large and sparse(-like) or banded, requiring $c_a^s mn$ flops to form the products AZ or $A^\top Z$, where $Z \in \mathbb{R}^{n\times m}$ and c_a^s is a constant independent of n. Similarly, we assume that $c_a^l mn$ flops are required to evaluate $A^{-1}Z$ or $A^{-\top}Z$, where c_a^l is a constant independent of n. (Note that there may be a sharper upper bound, as the LU factors of A_γ need to be computed only once at the start of Algorithm 2.) To simplify notation, we use c_a for c_a^s and

Table 1 Operation/memory counts, kth iteration, Algorithms 1 and 2

Computation	Flops	Memory
C_{k+1}	$2^k c_a l_k n$	$N_{k+1} n$
T_{k+1}	–	$O(l_k^2)$
Compress C_{k+1}	$4 l_k^2 n$	–
Modify T_{k+1}	$O(l_k^3)$	–
$\tilde{r}_{k+1}$	$4\left[(2l_k + l)^2 + 2l_k^2\right] n$	–
Total	$\left[2^k c_a l_k + 4l_k^2 + 4(2l_k + l)^2 + 8l_k^2\right] n$	$N_{k+1} n$

c_a^l when considering the Stein and Lyapunov equations, respectively. In the third column, the memory requirement in terms of the number of variables is recorded. Only $O(n)$ operations or memory requirement are included. Note that most of the work is done in the computation of C_{k+1}, for which $A_k^\top C_k$ has to be calculated recursively, as A_k is not available explicitly. In Table 1, we shall use the notation $N_k \equiv \sum_{j=1}^k l_j$. The operation count for the QR decomposition with column pivoting of an $n \times r$ matrix is $4nr^2$ flops [12, p. 250].

With l_k controlled by the compression and truncation in Section 2.2, the operation count will be dominated by the calculation of C_{k+1}. In our numerical examples in Section 7, the flop count near the end of Algorithm 1 dominates, with the work involved in one iteration approximately doubled that of the previous one. This corresponds to the 2^k factor in the total flop count. However, the last iteration is virtually free because there is no need to prepare C_{k+1} for the next iteration. In other words, for the true total CPU time up to a certain iteration, the total CPU time up to the previous iteration should be considered.

In addition, for large-scale Stein equations, there will be a start up cost of $4l^2 n$ flops in the orthogonalization of C and the modification of T^{-1}, in the calculation of $h = \|H\| = \|T^{-1}\|$. For large-scale Lyapunov equations, a start up cost of $\left(c_a^l l + 4l^2 + 1\right) n$ flops for the SM is made up of the following:

(1) setting up A_γ, requiring n flops;
(2) setting up C_0, requiring $c_a^l ln$ flops; and
(3) the orthogonalization of C_0 and the modification of T_0^{-1}, in the calculation of $h = \|H_0\| = \|T_0^{-1}\|$, requiring $4l^2 n$ flops.

We have ignored the one off LU factorization of A_γ at the beginning of Algorithm 2.

From Theorem 2.1 and the operation count in Table 1, we have the following result for the SM:

Corollary 6.1 *Assume the Smith method converges after k iterations to an approximate solution H_k with* $\operatorname{rank} H_k \le 2^k l = O(1)$, *according to some accuracy criterion. Then the Smith method has an $O(n)$ computational complexity and memory requirement.*

7 Numerical examples

Most of the numerical results were computed using MATLAB [24] Version R2010a, on a MacBook Pro with a 2.66 GHz Intel Core 2 Duo processor and 4 GB RAM, with machine accuracy $eps = 2.22 \times 10^{-16}$. However, the Lyapunov equation in Examples S1c and B4 overwhelmed the memory of the MacBook Pro and were solved on a Dell PowerEdge R910 computer.

Various parameters were put into the SM, with $\epsilon = 0.5 \times 10^{-15}$ controlling the convergence in terms of the relative residual $\widetilde{r}_k$, and τ the compression and truncation of C_k in (15). We also set the upper bound $l_{\max}$ for l_k, the ranks of C_k. The sub-total CPU time is $t_k = \sum_{i=1}^{k} \delta t_i$, with δt_i being the CPU time required for the ith iteration.

Typically, we start with a small $l_{\max}$ and a very small τ, with the SM achieving certain accuracy in terms of residuals. The appropriate τ, usually ranging from 10^{-16} to 10^{-10}, can then be chosen to achieve the accuracy we desire (around 10^{-16} to 10^{-15} in the numerical examples in this section). Too small a τ or too large an $l_{\max}$ will be ineffective.

7.1 Examples from [26]

In this section, we apply our SM to the examples in [25]. In the Stein equations in Examples S1a–S1c, we have the parameter $\alpha = 0.45, 0.49, 0.499$, giving the increasingly difficult spectral radii $\sigma(A) = 0.9, 0.98, 0.998$, respectively. We have tried only the most difficult examples with $n = 50,000$. Eight to thirteen iterations were required for the SM for approximate solutions to the large-scale Lyapunov equations to (near-)machine accuracy. Even with $\sigma(A) = 0.998$, Example S1c was easy for the SM.

The Lyapunov equation in the small Example S2 is not a challenge for the SM, even with $\sigma(A) = 0.9999$. Again, a near-machine accuracy was achieved quickly in twelve iterations. In [25], dozens to hundreds of restarts (with many more iterations) were required for worse accuracies. The example shows that the power and accuracy of the SM come from the convergence of $A_k = A^{2^k}$ as well as the approximate Krylov subspaces. The error in the SM turns out to be better than expected, with $\lambda^{2^k} \approx [\sigma(A)]^{2^{13}} = 0.4408$ (after $k = 12$ iterations) being moderately large in the right-hand-side of (12). This combines with a large approximate Krylov subspace of dimension $l_{\max} = 500$ and yields the near machine accuracy of $O(10^{-15})$. We have tried smaller subspaces but less accurate results were obtained.

In all the examples in this subsection, we choose a small $\tau = 10^{-30}$, essentially surrendering the control of the compression and truncation process in the SM to $l_{\max}$, the upper limit of the size of the approximating Krylov subspaces. The actual CPU time was not as informative as the number of iterations required. Note also that we work with A in Example S1, as compare to A^2 in [25, p. 18]. Recall that for the total CPU time up to the kth iteration, t_{k-1} should be considered because the last iteration is virtually free.

Table 2 Example S1a ($n = 50,000, l = 2, \alpha = 0.45, \sigma(A) = 0.9; \tau = 10^{-30}, l_k \leq 50$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	4.0500e−01	3.2610e−01	2.0503e−01	7.5132e−02	3	1.6e−02	1.6e−02
2	2.4655e−01	1.7524e−01	7.0624e−02	2.3371e−02	5	7.8e−02	9.4e−02
3	1.1966e−01	8.0626e−02	1.3747e−02	4.3498e−03	9	1.4e−01	2.3e−01
4	3.0961e−02	2.0679e−02	1.0365e−03	3.2528e−04	17	7.5e−01	9.8e−01
5	2.8277e−03	1.8878e−03	1.3571e−05	4.2572e−06	33	2.5e+00	3.5e+00
6	4.1215e−05	2.7515e−05	5.8842e−09	1.8458e−09	50	7.3e+00	1.1e+01
7	1.9024e−08	1.2701e−08	3.2093e−15	1.0067e−15	50	1.4e+01	2.5e+01
8	9.8856e−15	6.5997e−15	3.0942e−15	9.7063e−16	50		

Remark 7.1 The numerical examples in Tables 2, 3, 4 and 5 reveal an interesting phenomenon. The convergence result in (12) relies solely on the fact that $0 < \lambda < 1$. With $\lambda \approx 1$, the predicted numbers of iteration required for convergence are higher than what are actually required. This faster convergence came from the power of approximation of the Krylov subspaces in (21). This creates cancellations in the increment $dH_k \equiv A_k^\top H_k A_k$ in (7), thus the subsequent faster convergence.

7.2 Examples from [10]

In this subsection, we have tested the SM on selected numerical examples from [9]. The suite of benchmark problems involves Lyapunov equations from the continuous-time systems originated from the boundary control problem modelling the cooling of rail sections. The PDE model was semi-discretized using 2D finite elements to a continuous-time linear system with n variables, where $n = 1357, 5177, 20209, 79841$. For Stein equations, we transformed these continuous-time systems (with $\Delta t = 0.01$) to discrete-time models, as in [8] (i.e., the path indicated by the dashed arrows in Fig. 1).

Lyapunov equations are first transformed to equivalent Stein equations a là Cayley (i.e., the path represented by the solid arrows in Fig 1). Of course, the final Stein equations from the two different paths are not the same, with parameters δt and γ playing their parts. Somehow, for the examples from [9],

Table 3 Example S1b ($n = 50,000, l = 2, \alpha = 0.49, \sigma(A) = 0.98; \tau = 10^{-30}, l_k \leq 150$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	4.8020e−01	3.7088e−01	2.8824e−01	9.9728e−02	3	1.6e−02	1.6e−02
2	3.5745e−01	2.3211e−01	1.3958e−01	4.1578e−02	5	4.7e−02	9.4e−02
3	2.7288e−01	1.5820e−01	5.3697e−02	1.4457e−02	9	1.4e−01	2.3e−01
4	1.7029e−01	9.4270e−02	1.5814e−02	4.0657e−03	17	7.0e−01	9.4e−01
5	8.1265e−02	4.4407e−02	3.1592e−03	8.0069e−04	33	2.5e+00	3.4e+00
6	2.4811e−02	1.3530e−02	3.1885e−04	8.0614e−05	65	9.0e+00	1.2e+01
7	3.4068e−03	1.8576e−03	8.6637e−06	2.1900e−06	115	3.4e+01	4.6e+01
8	1.1086e−04	6.0446e−05	1.7557e−08	4.4380e−09	150	1.3e+02	1.7e+02
9	2.4824e−07	1.3535e−07	2.0079e−13	5.0757e−14	150	2.5e+02	4.2e+02
10	3.0069e−12	1.6395e−12	4.4366e−15	1.1215e−15	150		

Table 4 Example S1c ($n = 50,000, l = 2, \alpha = 0.499, \sigma(A) = 0.998; \tau = 10^{-30}, l_k \leq 300$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	4.9800e−01	3.8086e−01	3.1001e−01	1.0582e−01	3	3.0e−02	3.0e−02
2	3.8720e−01	2.4588e−01	1.6146e−01	4.6852e−02	5	4.0e−02	7.0e−02
3	3.2677e−01	1.8161e−01	7.1849e−02	1.8487e−02	9	4.2e−01	4.9e−01
4	2.4943e−01	1.2963e−01	2.8312e−02	6.8075e−03	17	8.6e−01	1.4e+00
5	1.7730e−01	8.9477e−02	1.0126e−02	2.3565e−03	33	1.7e+00	3.1e+00
6	1.1752e−01	5.8659e−02	3.2761e−03	7.5229e−04	65	6.9e+00	1.0e+01
7	7.0360e−02	3.5003e−02	9.1463e−04	2.0911e−04	117	2.70e+01	3.7e+01
8	3.5132e−02	1.7464e−02	1.9567e−04	4.4687e−05	192	9.9e+01	1.4e+02
9	1.2454e−02	6.1903e−03	2.4948e−05	5.6966e−06	300	3.1e+02	4.5e+02
10	2.3237e−03	1.1550e−03	1.1383e−06	2.5993e−07	300	6.4e+02	1.1e+03
11	1.3407e−04	6.6639e−05	6.6778e−09	1.5248e−09	300	1.2e+03	2.3e+03
12	8.9421e−07	4.4445e−07	6.4879e−13	1.4814e−13	300	2.4e+03	4.7e+03
13	9.3417e−11	4.6431e−11	1.2753e−14	2.9120e−15	300		

the first path (via the Lyapunov equation from the original continuous-time system, represented by the solid arrows in Fig. 1) yields slightly easier Stein equations to solve, as indicated by the numerical examples in this section. We have chosen $\gamma = 0.1$, 0.5 or 100 (for the largest Lyapunov equation with $n = 79841$) after a few tests, yielding good accuracy and efficiency, without any exhaustive attempt for optimality. From our experience in the numerical tests here and [10], the choice of γ is neither sensitive nor critical.

7.2.1 *Example B1 ($n = 1357, l = 6$)*

Although the smallest in the suite of examples on the cooling of steel profiles [9], Example B1 is not the easiest in any sense. Somehow for Stein equations, the SM converges faster (in terms of number of iterations) for the larger examples. The phenomenon can be quantified through the decreasing spectral radii of A_0 for increasing values of n.

From Tables 6 and 7, the machine accuracy of $O(10^{-16})$ is achieved for the relative residual within 15 iterations and 12 s for the Stein equation, and

Table 5 Example S2 ($n = 1,000, \sigma(A) = 0.9999, l = 2, \gamma = 300; \tau = 10^{-30}, l_k \leq 500$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	1.1484e−03	3.6687e−01	8.2658e−04	1.1398e−01	2	2.0e−02	2.0e−02
2	1.4491e−03	3.2635e−01	5.1737e−04	5.3438e−02	4	1.0e−02	3.0e−02
3	1.6022e−03	2.7440e−01	2.6767e−04	2.1664e−02	8	1.0e−02	4.0e−02
4	1.4626e−03	2.1099e−01	1.2879e−04	8.8778e−03	16	2.0e−02	6.0e−02
5	2.0279e−03	2.7481e−01	1.7922e−04	1.1573e−02	32	2.2e−01	2.8e−01
6	1.0496e−02	8.5724e−01	5.0245e−04	1.8600e−02	64	4.3e−01	7.2e−01
7	4.4239e−02	7.9035e−01	7.5340e−04	6.5714e−03	128	1.7e+00	2.4e+00
8	5.2868e−02	4.8908e−01	7.8938e−05	3.6180e−04	256	7.6e+00	1.0e+01
9	4.7654e−03	4.2240e−02	9.5450e−06	4.1946e−05	500	3.1e+01	4.1e+01
10	4.3457e−04	3.8375e−03	3.8276e−08	1.6758e−07	500	6.4e+01	1.1e+02
11	2.8088e−06	2.4803e−05	1.2854e−13	5.6276e−13	500	1.4e+02	2.4e+02
12	1.3282e−11	1.1729e−10	7.8168e−16	3.4224e−15	500		

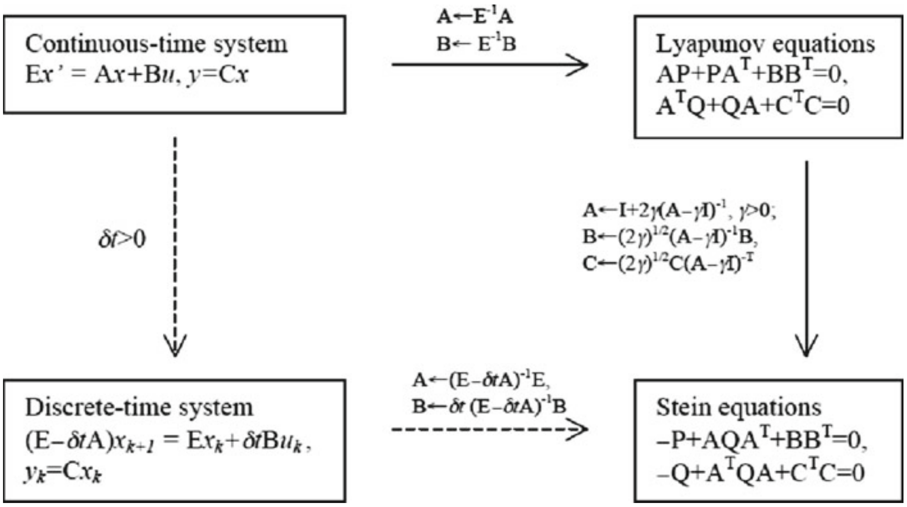
Continuous-time system
$Ex' = Ax+Bu, y=Cx$
$A \leftarrow E^{-1}A$
$B \leftarrow E^{-1}B$
Lyapunov equations
$AP+PA^T+BB^T=0,$
$A^TQ+QA+C^TC=0$
$\delta t>0$
$A \leftarrow I+2\gamma(A-\gamma I)^{-1}, \gamma>0;$
$B \leftarrow (2\gamma)^{1/2}(A-\gamma I)^{-1}B,$
$C \leftarrow (2\gamma)^{1/2}C(A-\gamma I)^{-T}$
$A \leftarrow (E-\delta tA)^{-1}E,$
$B \leftarrow \delta t\,(E-\delta tA)^{-1}B$
Discrete-time system
$(E-\delta tA)x_{k+1} = Ex_k+\delta tBu_k,$
$y_k=Cx_k$
Stein equations
$-P+AQA^T+BB^T=0,$
$-Q+A^TQA+C^TC=0$

Fig. 1 Paths from continuous-time system to Stein equation

5 iterations and less than 0.13 s for the Lyapunov equation. Somehow, the Lyapunov equations are faster to solve than the analogous Stein equation, in terms of number of iterations. Note that the last iteration is virtually free because there is no need to prepare for the next iteration.

Recall that the Stein and Lyapunov equations in Tables 6 and 7 originate from the same system in [9], with the latter coming from the original continuous-time system (then transformed a là Cayley) and the former from the approximating discrete-time system (see Fig. 1). The difference in the total CPU time required is misleading. There is very little difference when the total CPU time is compared after the same number of iterations are completed,

Table 6 Example B1 (Stein equation, with $n = 1357, l = 6; \tau = 10^{-12}, l_k \leq 20$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	1.1953e+01	4.9903e−01	1.1907e+01	1.9907e−01	12	0	0
2	2.3768e+01	4.9805e−01	1.1815e+01	1.1015e−01	17	1.0e−02	2.0e−02
3	4.6985e+01	4.9612e−01	1.1633e+01	5.7862e−02	20	0	2.0e−02
4	9.1815e+01	4.9226e−01	1.1279e+01	2.9348e−02	20	2.0e−02	4.0e−02
5	1.7537e+02	4.8462e−01	1.0682e+01	1.4546e−02	20	2.0e−02	6.0e−02
6	3.3700e+02	4.9266e−01	1.0373e+01	7.5194e−03	20	4.0e−02	1.0e−01
7	6.4504e+02	4.8533e−01	9.7815e+00	3.6650e−03	20	7.0e−02	1.7e−01
8	1.1820e+03	4.7072e−01	8.7002e+00	1.7291e−03	20	1.1e−01	2.8e−01
9	1.9870e+03	4.4177e−01	6.8879e+00	7.6504e−04	20	2.7e−01	5.5e−01
10	2.8211e+03	3.8554e−01	4.3282e+00	2.9564e−04	20	4.6e−01	1.0e+00
11	2.8947e+03	2.8363e−01	1.7230e+00	8.4400e−05	20	8.0e−01	1.8e+00
12	1.6213e+03	1.3723e−01	2.7881e−01	1.1799e−05	20	1.4e+00	3.2e+00
13	3.0741e+02	2.5373e−02	7.5611e−03	3.1203e−07	20	2.8e+00	6.0e+00
14	8.6100e+00	7.1019e−04	5.7341e−06	2.3647e−10	20	5.6e+00	1.2e+01
15	6.5471e−03	5.4003e−07	1.3829e−11	5.7032e−16	20		

Table 7 Example B1 (Lyapunov equation, with $n = 1357, l = 6; \gamma = 0.1, \tau = 10^{-15}, l_k \leq 20$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	1.4975e+01	1.2626e−01	2.2429e+00	9.3598e−03	6	3.0e−02	5.0e−02
2	2.5823e+00	2.1325e−02	5.1696e−02	2.1317e−04	12	2.0e−02	8.0e−02
3	6.0953e−02	5.0311e−04	3.3931e−04	1.3987e−06	18	2.0e−02	1.0e−01
4	5.3255e−04	4.3956e−06	1.1995e−07	4.9448e−10	19	3.0e−02	1.3e−01
5	1.9190e−07	1.5840e−09	5.1290e−14	2.1143e−16	20		

indicating that the Lyapunov equation is slightly more difficult to solve than the Stein equation, most likely because of the inversion of A involved. As each iteration roughly requires twice the amount of work or memory as the previous one, the number of iterations required for convergence dictates the final amount of CPU time consumed. Note that the speed of convergence of the SM is influenced by the minimum distance between $\sigma(A)$ and the unit circle. Somehow, indicated by the faster convergence for the Lyapunov equation in Table 7, this distance for the Stein equation (from the discrete-time system approximating the original continuous-time system; dashed arrows in Fig. 1) is greater than that for the analogous Lyapunov equation (from the original continuous-time system, and then transformed to an equivalent Stein equation; solid arrows in Fig. 1).

7.2.2 Example B2 ($n = 5177, l = 6$)

From Tables 8 and 9, the machine accuracy of $O(10^{-16})$ is achieved within 14 iterations and 25 s for the Stein equation, and five iterations and less than 0.23 s for the Lyapunov equation.

Table 8 Example B2 (Stein equation, with $n = 5177, l = 6; \tau = 10^{-10}, l_k \leq 20$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	1.1815e+01	4.9612e−01	1.1634e+01	1.9630e−01	12	4.0e−02	4.0e−02
2	2.3089e+01	4.9227e−01	1.1281e+01	1.0735e−01	17	6.0e−02	1.0e−01
3	4.4104e+01	4.8464e−01	1.0682e+01	5.5456e−02	19	7.0e−02	1.7e−01
4	8.4369e+01	4.9180e−01	1.0373e+01	2.9288e−02	20	9.0e−02	2.6e−01
5	1.6149e+02	4.8535e−01	9.7827e+00	1.4466e−02	20	1.0e−01	3.6e−01
6	2.9595e+02	4.7075e−01	8.7022e+00	6.8680e−03	20	2.2e−01	5.8e−01
7	4.9760e+02	4.4183e−01	6.8910e+00	3.0487e−03	20	2.4e−01	8.2e−01
8	7.0670e+02	3.8564e−01	4.3320e+00	1.1803e−03	20	4.8e−01	1.3e+00
9	7.2560e+02	2.8381e−01	1.7260e+00	3.3738e−04	20	8.6e−01	2.2e+00
10	4.0680e+02	1.3742e−01	2.7974e−01	4.7239e−05	20	1.6e+00	3.8e+00
11	7.7268e+01	2.5452e−02	7.6087e−03	1.2529e−06	20	3.2e+00	7.0e+00
12	2.1706e+00	7.1451e−04	5.8014e−06	9.5467e−10	20	6.1e+00	1.3e+01
13	1.6593e−03	5.4621e−07	1.4475e−11	2.3820e−15	20	1.2e+01	2.5e+01
14	4.8032e−09	1.5811e−12	2.8131e−12	4.6293e−16	20		

Table 9 Example B2 (Lyapunov equation, with $n = 5177, l = 6; \gamma = 0.5, \tau = 10^{-8}, l_k \leq 20$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	5.2151e+00	1.8108e−01	1.1806e+00	2.0074e−02	6	2.0000e−02	3.0000e−02
2	1.4505e+00	4.8000e−02	6.2186e−02	1.0268e−03	12	2.0000e−02	5.0000e−02
3	8.0593e−02	2.6602e−03	1.7679e−04	2.9145e−06	17	7.0000e−02	1.2000e−01
4	2.2995e−04	7.5900e−06	4.2083e−09	6.9377e−11	17	1.1000e−01	2.3000e−01
5	5.8696e−09	1.9374e−10	2.5362e−14	4.1810e−16	17		

7.2.3 Example B3 ($n = 20209, l = 6$)

From Tables 10 and 11, the machine accuracy of $O(10^{-16})$ is achieved within 12 iterations and 41 s for the Stein equation, and seven iterations and less than 3.3 s for the Lyapunov equation.

7.2.4 Example B4 ($n = 79841, l = 6$)

From Table 12 for the Stein equation, the machine accuracy of $O(10^{-16})$ is achieved within ten iterations and 47 s for the Stein equation.

For the Lyapunov equation, the 4 GB RAM of our MacBook Pro could not cope with the memory requirement. A Dell PowerEdge R910 computer, with 4×8-core Intel Xeon 2.26 GHz CPUs and 1024 GB RAM, was used instead for the equation. From Table 13, the machine accuracy of $O(10^{-16})$ is achieved within five iterations and 12.3 s for the Lyapunov equation. As in the previous examples, the cost for the final iteration is negligible, as no preparation is required for the next iteration. The results in Table 13 confirm the afore-mentioned comments that the Lyapunov equation is easier than the analogous Stein equation, and the larger Stein equations are easier in the sense that less iterations are required, possibly because of the fixed value of l relative to the varying n.

Table 10 Example B3 (Stein equation, with $n = 20209, l = 6; \tau = 10^{-12}, l_k \leq 20$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	1.1290e+01	4.8477e−01	1.0683e+01	1.8674e−01	12	2.0e−02	2.0e−02
2	2.1211e+01	4.8291e−01	1.0375e+01	1.0549e−01	17	1.2e−01	1.5e−01
3	4.0605e+01	4.8539e−01	9.7860e+00	5.4947e−02	20	2.5e−01	4.0e−01
4	7.4431e+01	4.7083e−01	8.7081e+00	2.6722e−02	20	3.1e−01	7.3e−01
5	1.2520e+02	4.4199e−01	6.9003e+00	1.2012e−02	20	3.9e−01	1.1e+00
6	1.7798e+02	3.8594e−01	4.3435e+00	4.6822e−03	20	8.3e−01	1.9e+00
7	1.8305e+02	2.8432e−01	1.7349e+00	1.3445e−03	20	1.3e+00	3.2e+00
8	1.0295e+02	1.3801e−01	2.8256e−01	1.8922e−04	20	2.6e+00	5.9e+00
9	1.9662e+01	2.5695e−02	7.7597e−03	5.0667e−06	20	5.1e+00	1.1e+01
10	5.5780e−01	7.2843e−04	6.0303e−06	3.9348e−09	20	1.0e+01	2.1e+01
11	4.3460e−04	5.6755e−07	1.6740e−11	1.0923e−14	20	2.0e+01	4.1e+01
12	1.3854e−09	1.8092e−12	2.1117e−13	1.3779e−16	20		

Table 11 Example B3 (Lyapunov equation, with $n = 20209, l = 6; \gamma = 0.5, \tau = 10^{-12}, l_k \leq 20$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	5.9351e−02	8.6069e−02	2.0645e−02	1.4435e−02	6	8.0e−02	9.0e−02
2	2.9967e−02	4.3408e−02	4.3733e−03	3.0386e−03	12	1.8e−01	2.7e−01
3	8.0773e−03	1.1700e−02	3.3262e−04	2.3054e−04	17	3.8e−01	6.5e−01
4	8.5182e−04	1.2339e−03	1.2312e−05	8.5326e−06	18	5.5e−01	1.2e+00
5	4.2250e−05	6.1199e−05	6.7101e−08	4.6501e−08	19	7.3e−01	1.9e+00
6	2.5289e−07	3.6631e−07	4.1645e−12	2.8860e−12	20	1.4e+00	3.3e+00
7	1.6079e−11	2.3291e−11	4.8006e−16	3.3268e−16	20		

7.3 Example from [29]

Lastly, we apply our SM to the Lyapunov equations from a 3D example in [28, Example 5.2]. The matrices $B \in \mathbb{R}^{n\times l}$ were generated randomly, with $n = 10648$ and $l = 1, 2, 4, 7$. We controlled the truncation and compression process solely with $l_{\max}$ and choose $\tau = 10^{-30}$, thus putting it out of play. We experimented on the choice of γ using the easiest $l = 1$ case and settle on an acceptable $\gamma = 8$ after a few trials. The examples turn out to be amongst the easiest we have tried, producing much more accurate results (in terms of relative residuals) in less CPU times and using smaller Krylov subspaces, when compared to [28, Table 5.1]. Note a similar computer to that in [28] has been used for the numerical experiments. Still, it will unwise to place too much importance in the comparison of CPU times from difference computers.

7.3.1 Example M1

From Table 14, we achieve the near-machine accuracy of $O(10^{-15})$ within five iterations and 42 s. Comparing with the similar results in [28, Table 5.1], we achieve an accuracy of $O(10^{-9})$ in 19 s with rank $H_k = 16$, against an accuracy of 10^{-8} in 39 s with rank $H_k = 68$. It is difficult to compare the CPU times from different computers but the differences in the accuracy and rank H_k indicate that the SM is competitive against the Krylov subspaces method in [28]. Similar comments can be made from Examples M2, M4 and M7 in the rest of this subsection.

Table 12 Example B4 (Stein equation, with $n = 79841, l = 6; \tau = 10^{-10}, l_k \leq 20$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	1.0382e+01	4.8212e−01	9.7986e+00	1.8242e−01	12	1.0e−01	1.0e−01
2	1.9047e+01	4.7114e−01	8.7305e+00	9.6376e−02	17	3.5e−01	4.5e−01
3	3.2098e+01	4.4259e−01	6.9354e+00	4.5336e−02	17	5.7e−01	1.0e+00
4	4.5787e+01	3.8709e−01	4.3871e+00	1.8132e−02	17	8.7e−01	1.9e+00
5	4.7401e+01	2.8625e−01	1.7691e+00	5.2974e−03	17	1.7e+00	3.6e+00
6	2.6978e+01	1.4024e−01	2.9350e−01	7.6025e−04	17	3.0e+00	6.6e+00
7	5.2605e+00	2.6633e−02	8.3604e−03	2.1108e−05	17	5.9e+00	1.3e+01
8	1.5495e−01	7.8390e−04	6.9933e−06	1.7643e−08	17	1.2e+01	2.5e+01
9	1.2996e−04	6.5745e−07	2.1694e−11	5.4732e−14	17	2.3e+01	4.7e+01
10	4.6240e−10	2.3393e−12	1.2494e−13	3.1522e−16	17		

Table 13 Example B4 (Lyapunov equation, with $n = 79841, l = 6; \gamma = 7.5, \tau = 10^{-16}, l_k \le 30$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	3.0078e−01	1.6538e−01	6.1199e−02	1.6535e−02	12	8.70e−01	1.22e+00
2	7.3776e−02	3.9019e−02	2.6045e−03	6.8754e−04	17	9.50e−01	2.17e+00
3	3.2790e−03	1.7314e−03	4.8226e−06	1.2718e−06	20	2.46e+00	4.63e+00
4	6.0867e−06	3.2139e−06	1.2950e−10	3.4152e−11	23	7.65e+00	1.23e+01
5	1.8938e−10	9.9994e−11	5.7269e−16	1.5103e−16	27		

Table 14 Example M1 ($l = 1; \gamma = 8, \tau = 10^{-30}, l_k \le 30$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	1.5176e+02	2.3356e−01	5.7489e+01	3.9665e−02	2	2.4e+00	3.7e+00
2	6.5779e+01	1.0097e−01	9.5881e+00	6.5956e−03	4	4.9e+00	8.6e+00
3	1.2199e+01	1.8725e−02	1.4969e−01	1.0297e−04	8	1.0e+01	1.9e+01
4	1.8102e−01	2.7786e−04	3.9757e−06	2.7348e−09	16	2.4e+01	4.2e+01
5	5.7914e−06	8.8897e−09	3.2807e−36	2.2568e−15	30		

Table 15 Example M2 ($l = 2; \gamma = 8, \tau = 10^{-30}, l_k \le 55$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	1.5275e+02	2.3344e−01	5.7522e+01	3.9405e−02	4	2.6e+00	3.9e+00
2	6.5923e+01	1.0049e−01	1.0604e+01	7.2431e−03	8	5.5e+00	9.4e+00
3	1.3551e+01	2.0656e−02	1.5176e−01	1.0365e−04	16	1.3e+01	2.2e+01
4	1.8326e−01	2.7934e−04	3.9889e−06	2.7245e−09	32	2.9e+01	5.1e+01
5	5.8074e−06	8.8521e−09	5.2529e−12	3.5879e−15	55		

Table 16 Example M4 ($l = 4; \gamma = 8, \tau = 10^{-30}, l_k \le 110$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	1.6237e+02	2.3602e−01	5.8122e+01	3.7884e−02	8	2.8e+00	4.1e+00
2	6.6759e+01	9.6784e−02	1.0791e+01	7.0125e−03	16	5.7e+00	9.8e+00
3	1.3867e+01	2.0103e−02	1.9250e−01	1.2508e−04	32	1.3e+01	2.3e+01
4	2.3801e−01	3.4504e−04	4.1615e−06	2.7040e−09	64	3.6e+01	5.9e+01
5	5.9636e−06	8.6453e−09	8.9324e−12	5.8041e−15	110		

Table 17 Example M7 ($l = 7$; $\gamma = 8$, $\tau = 10^{-30}$, $l_k \leq 190$)

k	$\|dH_k\|$	$\|dH_k\|/\|H_k\|$	r_k	$\widetilde{r}_k$	l_k	δt_k	t_k
1	1.6879e+02	2.4126e−01	6.5418e+01	4.1782e−02	14	2.9e+00	4.2e+00
2	7.6560e+01	1.0917e−01	1.2438e+01	7.9193e−03	28	6.5e+00	1.1e+01
3	1.5695e+01	2.2379e−02	2.0510e−01	1.3058e−04	56	1.6e+01	2.7e+01
4	2.5823e−01	3.6821e−04	4.2151e−06	2.6836e−09	112	4.7e+01	7.4e+01
5	6.0372e−06	8.6082e−09	1.0092e−11	6.4252e−15	190		

7.3.2 Example M2

From Table 15, we achieve the near-machine accuracy of $O(10^{-15})$ within five iterations and 51 s. Comparing with the similar results in [28, Table 5.1], we achieve an accuracy of $O(10^{-9})$ in 22 s with $\operatorname{rank} H_k = 32$, against an accuracy of 10^{-8} in 51 s with $\operatorname{rank} H_k = 132$.

7.3.3 Example M4

From Table 16, we achieve the near-machine accuracy of $O(10^{-15})$ within five iterations and 59 s. Comparing with the similar results in [28, Table 5.1], we achieve an accuracy of $O(10^{-9})$ in 23 s with $\operatorname{rank} H_k = 64$, against an accuracy of 10^{-8} in 91 s with $\operatorname{rank} H_k = 264$.

7.3.4 Example M7

From Table 17, we achieve the near-machine accuracy of $O(10^{-15})$ within five iterations and 74 s. Comparing with the similar results in [28, Table 5.1], we achieve an accuracy of $O(10^{-9})$ in 27 s with $\operatorname{rank} H_k = 112$, against an accuracy of 10^{-8} in 205 s with $\operatorname{rank} H_k = 448$. From Tables 14–17, there seems to be only a minor increase in difficulty from the increasing values of l. From the numerical results, the (rational) Krylov subspaces in our SM seem to be more effective than those in [28].

8 Conclusions

We have adapted or modified the Smith method for the large-scale Stein equation (1), with A being large and sparse(-like), and C being low-ranked. Similar Lyapunov equations (2) can be treated after Cayley transforms. One possibility is not to form $A_k = A_0^{2^k}$ explicitly, and compress and truncate C_k to control its growth in rank. For well-behaved Stein (or Lyapunov) equations, with the eigenvalues of the stable A not on the unit circle (or the imaginary axis), low-ranked approximations to the solutions X can be obtained efficiently. (For examples with eigenvalues almost touching the unit circle or imaginary axis, see Section 7.1.) The convergence of the SM is quadratic (ignoring the compression and truncation of C_k), as shown in [29]. The computational complexity and memory requirement are both $O(n)$ per iteration, provided that the growth of C_k is not fast or controlled.

In comparison to the techniques proposed previously, e.g. the ADI-type methods in [5, 23], there is no need to estimate parameters to accelerate convergence. Our parameters τ and $l_{\max}$ are comparatively straight-forward to set. The Krylov subspaces generated by the SM seem to be more effective than those in [28], producing more accurate numerical approximations using smaller Krylov subspaces. Note that the SM also benefits from the diminishing A_k in (16), in addition to the power of approximation of the Krylov subspaces.

Acknowledgements The first author was supported by the Major Scientific Research Foundation of Southeast University (No.3207011102), the NSFC (No.11101080) and the SRFDP (No.20110092120023). The second author has been supported by a Monash Graduate Scholarship and a Monash International Postgraduate Research Scholarship. Parts of this project were completed while the first author visited Monash University and when the third author visited the CMMSC and the NCTS at the National Chiao Tung University, and we would like to acknowledge the support from these Universities. The fourth author would like to acknowledge the support from the National Science Council and the National Centre for Theoretical Sciences in Taiwan. He also like to thank the CMMSC and the ST Yau Centre at the National Chiao Tung University for their support.

We would like to thank Professor Peter Benner for pointing us to the "Cooling of Steel Profiles" examples in [9], which we adapted in Section 7.2. Professor Valeria Simoncini has also kindly sent us information on the numerical examples in [28], one of which we adapted in Section 7.3.

We would also like to thank the referees for their valuable comments.

References

1. Antoulas, A.: Approximation of Large-Scale Dynamical Systems. SIAM Publications, Philadelphia, PA (2005)
2. Benner, P.: Solving large-scale control problems. IEEE Control Syst. Mag. **14**, 44–59 (2004)
3. Benner, P.: Editorial of special issue on "Large-Scale Matrix Equations of Special Type". Numer. Linear Algebra Appl. **15**, 747–754 (2008)
4. Benner, P., Ezzatti, P., Kressner, D., Quintana-Orti, E.S., Remón, A.: A mixed-precision algorithm for the solution of Lyapunov equations on hybrid CPU–GPU platforms. Parallel Comput. **37**, 439–450 (2011). doi:10.1016/j.parco.2010.12.002
5. Benner, P., Li, J.-R., Penzl, T.: Numerical solution of large Lyapunov equations, Riccati equations and linear-quadratic control problems. Numer. Linear Algebra Appl. **15**, 755–777 (2008)
6. Benner, P., Li, R.-C., Truhar, N.: On the ADI method for Sylvester equations. J. Comput. Appl. Math. **233**, 1035–1045 (2009)
7. Benner, P., Mehrmann, V., Sorensen, D. (eds.): Dimension Reduction of Large-Scale Systems. Lecture Notes in Computational Science and Engineering, vol. 45. Springer, Berlin/Heidelberg (2005)
8. Benner, P., Fassbender, H.: On the numerical solution of large-scale sparse discrete-time Riccati equations. Adv. Comput. Math. **35**, 119–147 (2011)
9. Benner, P., Saak, J.: A semi-discretized heat transfer model for optimal cooling of steel profiles. In: Benner, P., Mehrmann, V., Sorensen, D.C. (eds.) Dimension Reduction of Large-Scale Systems. Lecture Notes in Computational Science and Engineering, vol. 45, pp. 353–356. Springer, Berlin/Heidelberg (2005)
10. Chu, E.K.-W., Fan, H.-Y., Lin, W.-W.: A structure-preserving doubling algorithm for continuous-time algebraic Riccati equations. Linear Algebra Appl. **396**, 55–80 (2005)

11. Damm, T.: Direct methods and ADI-pre-conditioned Krylov subspace methods for generalized Lyapunov equations. Numer. Linear Algebra Appl. **15**, 853–871 (2008)
12. Golub, G.H., Van Loan, C.F.: Matrix Computations, 2nd edn. Johns Hopkins University Press, Baltimore, MD (1989)
13. Gugercin, S., Sorensen, D.C., Antoulas, A.C.: A modified low-rank Smith method for large-scale Lyapunov equations. Numer. Algor. **32** 27–55 (2003)
14. Jaimoukha, I.M., Kasenally, E.M.: Krylov subspace methods for solving large Lyapunov equations. SIAM J. Numer. Anal. **31**, 227–251 (1994)
15. Jbilou, K.: Block Krylov subspace methods for large continuous-time algebraic Riccati equations. Numer. Algor. **34**, 339–353 (2003)
16. Jbilou, K.: An Arnoldi based algorithm for large algebraic Riccati equations. Appl. Math. Lett. **19**, 437–444 (2006)
17. Jbilou, K.: Low rank approximate solutions to large Sylvester matrix equations. Appl. Math. Comput. **177**, 365–376 (2006)
18. Jbilou, K.: ADI preconditioned Krylov methods for large Lyapunov matrix equations. Linear Algebra Appl. **432**, 2473–2485 (2010)
19. Jbilou, K., Riquet, A.: Projection methods for large Lyapunov matrix equations. Linear Algebra Appl. **415**, 344–358 (2006)
20. Kenney, C., Hewer, G.: The sensitivity of the algebraic and differential Riccati equations. SIAM J. Control Optim. **28**, 50–69 (1990)
21. Kressner, D.: Memory-efficient Krylov subspace techniques for solving large-scale Lyapunov equations. In: IEEE Int. Symp. Computer-Aided Control Systems Design, San Antonio, pp. 613–618 (2008)
22. Lasiecka, I., Triggiani, R.: Control Theory for Partial Differential Equations: Continuous and Approximation Theories; I. Abstract Parabolic Systems. Cambridge University Press, Cambridge (2000)
23. Li, J.-R., White, J.: Low-rank solution of Lyapunov equations. SIAM Rev. **46**, 693–713 (2004)
24. mathworks: MATLAB User's Guide (2010)
25. Sadkane, M.: A low-rank Krylov squared Smith method for large-scale discrete-time Lyapunov equations. Linear Algebra Appl. (2011). doi:10.1016/j.laa.2011.07.021
26. Saak, J.: Efficient numerical solution of large scale algebraic matrix equations in PDE control and model order reduction. Dr. rer. nat. Dissertation, Chemnitz University of Technology, Germany (2009)
27. Saak, J., Mena, H., Benner, P.: Matrix Equation Sparse Solvers (MESS): A Matlab Toolbox for the Solution of Sparse Large-Scale Matrix Equations. Chemnitz University of Technology, Germany (2010)
28. Simoncini, V.: A new iterative method for solving large-scale Lyapunov matrix equations. SIAM J. Sci. Comput. **29**, 1268–1288 (2007)
29. Smith, R.A.: Matrix equation $XA + BX = C$. SIAM J. Appl. Math. **16**, 198–201 (1968)
30. Truhar, N., Li, R.-C.: On ADI method for Sylvester equations. Technical Report 2008-02, Department of Mathematics, University of Texas at Arlington (2008)