Journal of Computational and Applied Mathematics 361 (2019) 130–143

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

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

Large-scale algebraic Riccati equations with high-rank constant terms

Bo Yu [a,b], Hung-Yuan Fan [c,*], Eric King-wah Chu [d]

[a] School of Science, Hunan University of Technology, Zhuzhou 412000, China
[b] Department of Mathematics and Statistics, Changsha University of Science and Technology, Changsha 410004, China
[c] Department of Mathematics, National Taiwan Normal University, Taipei 116, Taiwan
[d] School of Mathematics, Monash University, 9 Rainforest Walk, Melbourne, Victoria 3800, Australia

ARTICLE INFO

Article history:
Received 27 May 2016
Received in revised form 25 January 2019

MSC:
15A24
65F30
93C05

Keywords:
Algebraic Riccati equation
Feedback gain
High-rank constant term
Large-scale problem
LQR optimal control

ABSTRACT

We consider the numerical solution of large-scale algebraic Riccati equations with high-rank constant terms. The solutions are not numerically low-rank, so the previously successful methods based on low-rank representations are not directly applicable. We modify the doubling algorithm, making use of the low-rank in the input matrix B. We also solve the challenging problems in the estimation of residuals and relative errors, convergence control and the output of the modified algorithm. Illustrative numerical examples are presented.

1. Introduction

One success story of science and technology in the twentieth and twenty first centuries is the ability of mankind to influence or control systems around them. Notable examples of such systems are household appliances, machines, automobiles, aircrafts, computers, the weather, the oceans and human societies. These diverse systems can be approximated or modelled, at least locally, as linear time-invariant (LTI) systems. Algebraic, differential or difference Riccati equations are then solved to construct the optimal controls. Much advances have been made in the solution of these equations in the past few decades. It seems, for a while, that not much more can or needs to be done for algebraic Riccati equations (AREs). This paper proposes the first algorithm for an important and unsolved special case for large-scale AREs with high-rank constant terms.

1.1. Algebraic Riccati equations

Consider the LTI control system in continuous-time:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t),$$

* Corresponding author.
E-mail addresses: boyu_hut@126.com (B. Yu), hyfan@math.ntnu.edu.tw (H.-Y. Fan), eric.chu@monash.edu (E.K.-w. Chu).

https://doi.org/10.1016/j.cam.2019.04.014
0377-0427/

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{l\times n}$ with $m, l \leq n$. The linear quadratic regulator (LQR) control minimizes the energy or cost functional $J_c(x,u) \equiv \int_0^\infty \left[x(t)^\top Hx(t) + u(t)^\top Ru(t)\right] dt$, with the constant term $H \equiv C^\top T^{-1}C \geq 0$ and $R, T > 0$. Here, a symmetric matrix $P > 0$ (≥ 0) when all its eigenvalues are positive (non-negative). Also, $P > Q$ ($P \geq Q$) if and only if $P - Q > 0$ (≥ 0). The corresponding optimal control $u(t) = F_c x(t)$ and the feedback gain $F_c \equiv -R^{-1}B^\top X$ can then be expressed in terms of the unique positive semidefinite stabilizing solution X of the continuous-time algebraic Riccati equation (CARE) [1–5], with $G \equiv BR^{-1}B^\top \geq 0$:

$$\mathcal{C}(X) \equiv A^\top X + XA - XGX + H = 0. \tag{1}$$

See [6,7] on related research in parameter estimation and Kalman state filtering.

Analogously, we also consider the LTI control system in discrete-time:

$$x(t+1) = Ax(t) + Bu(t), \quad y(t) = Cx(t).$$

The LQR control minimizes $J_d(x,u) \equiv \sum_{t=0}^\infty \left[x(t)^\top Hx(t) + u(t)^\top Ru(t)\right]$. The corresponding optimal control $u(t) = F_d x(t)$ and the feedback gain $F_d \equiv -(R + B^\top XB)^{-1}B^\top XA$ can be expressed in terms of the unique positive semidefinite stabilizing solution X of the discrete-time algebraic Riccati equation (DARE) [1,3,4,8]:

$$\begin{aligned}\mathcal{D}(X) \equiv\ & -X + A^\top X(I + GX)^{-1}A + H \\ =\ & -X + A^\top XA - A^\top XB(R + B^\top X)^{-1}B^\top XA + H = 0,\end{aligned} \tag{2}$$

by the Sherman–Morrison–Woodbury formula (SMWF) [9]:

$$(M + UDV^\top)^{-1} = M^{-1} - M^{-1}U(D^{-1} + V^\top M^{-1}U)^{-1}V^\top M^{-1}.$$

1.2. Previous work

The solution of AREs has been an active area of research, due to its importance in optimal control and filtering. Many practitioners in control theory and applied mathematics researched the problem, contributed dozens of methods [2–4,8]. Classical methods made use of canonical forms, determinants and polynomial manipulation and state-of-the-art methods solve AREs in a numerically stable manner; see [2,8] and their references for more details. One favourite approach formulates the AREs as eigenvalue problems [10] and has been implemented in MATLAB [11] (in the commands `care` and `dare`). The other favourite is the Newton–Kleinman method [12]. For modern numerical methods for AREs for systems of moderate dimensions, please consult [2,8,10,13–16].

For control problems for parabolic PDEs and the balancing based model order reduction of large linear systems, large-scale CAREs, DAREs, Lyapunov and Stein equations have to be solved [1,13,16–24]. Without solving the corresponding AREs, alternative solutions require the invariant or deflating subspaces of the corresponding Hamiltonian matrices or symplectic pencils which are prohibitively expensive to compute.

Benner and his collaborators have done much on large-scale AREs with low-rank structures; see [14,16,18,24,25] and the references therein. The sub-problems of (inexact) Newton's methods are the Lyapunov and Stein equations which are solved with the ADI iterations. (The initialization of Newton's method and the choice of parameters for the ADI iterations remain challenging.) Consult [22,23,26] on the Krylov subspace methods and [27–35] on the related gradient based iterations for Lyapunov or Sylvester equations. The structure-preserving doubling algorithm (SDA) [2,8] has been adapted for large-scale problems [36,37], making use of the structure in A and the fact that $G \equiv BR^{-1}B^\top$, $H \equiv C^\top T^{-1}C$ are low-rank (i.e. $m, l \ll n$).

1.3. High-rank H

In many large-scale control problems, the constant term $H \equiv C^\top T^{-1}C$ in the AREs is low-rank with $C \in \mathbb{R}^{l\times n}$ and $l \ll n$. A nice property in this case is that the unique positive definite stabilizing solution X can be approximated numerically by a low-rank matrix $C_a^\top T_a^{-1}C_a$ with $C_a \in \mathbb{R}^{l_a\times n}$ and $l_a \ll n$. Many practitioners then employed the Newton–Galerkin–ADI method [16,19,20] or SDA [36,37] to form C_a and T_a as their computation and storage were efficient.

Recently, [38–40] (all unpublished, the only known references for this problem) raised the problem associated with a "high-rank" constant term H. It has been suggested that the (large-scale) CAREs "can be handled using algorithm ure rules" [39, Page 62], on integrals like

$$K_1 = B^\top X_1 = B^\top \int_0^\infty e^{t(A-BK_0)^\top}(H + K_0^\top K_0)e^{t(A-BK_0)}\, dt,$$

where $K_0 \equiv B^\top X_0$ is known. A variant of the Newton–Kleinman method is applied when an initial stabilizing X_0 is available.

Unfortunately for a large-scale CARE with a high-rank H, the solution X is no longer numerically low-rank (more details in Sections 2.1 and 2.2). Consequently, the Newton–Galerkin–ADI method, Krylov subspace methods or SDA no longer work. Note that even storing or outputting X are nontrivial, so are the computation of the residual or relative errors for any approximation H_k to X and the corresponding convergence control of any iterative procedure for $\{H_k\}$.

1.4. Main contributions

The efficient solution of large-scale AREs with high-rank constant terms is an important problem which has not been solved. This paper proposes the first algorithm for the problem, which is different to the Newton–Kleinman method in [39], without the difficult initial stabilization step.

Our main contributions are summarized as follows:

(1) There are some preliminary materials in [38,40] and some talks (as in [39]) presented on CAREs with high-rank H. Our paper is the first on the numerical solution of the problem.
(2) We modify the SDA for AREs with high-rank H, revealing the high-rank form of X. Consequently, our algorithm outputs directly the feedback gains $F_d, F_c \in \mathbb{R}^{m\times n}$, which have only m rows and are feasible to be represented and stored.
(3) We show that large-scale AREs for high-rank H can be solved efficiently by the modified doubling algorithm, the SDA_h, which output the feedback gains. We propose algorithms of $O(c_a n)$ computational complexity (with constants involving 2^k, which is in turn O(1) for small values of k). Here $c_a n$ is the flop count required for multiplying a vector v to A or H (or one of their transposes) for DAREs, or the flop count of one solve of the linear system associated with $A - \gamma I$ (or its transpose) for CAREs, and k is the number of iterations required for our (near) quadratically convergent algorithm and is normally small.
(4) We reveal the new difficulties associated with the computation of the residuals and relative errors, for the convergence control, and propose remedies. These difficulties arise from the high-rank constant term H and are not relevant previously for the solution of AREs with low-rank H in [14,16,18,24,25,36,37].

2. Discrete-time algebraic Riccati equations

Consider the DARE (2) with a *high-rank* constant term $H \geq 0$ but $G = BR^{-1}B^\top \in \mathbb{R}^{n\times n}$ with rank $m \leq n$. For large-scale DAREs, we further assume that $m \ll n$ and A, H are sparse or structured, with at most c_a nonzero elements on each row or column. More generally, we may assume that A, H are "sparse-like", with multiplication to row or column vectors realizable in $O(c_a n)$ flops. Sparse-like matrices include low-rank updates of banded or sparse matrices.

For the large-scale CARE (1) with $m \ll n$, we need the matrices being efficiently invertible and multiplicable to vectors; see Section 4. CAREs are mathematically equivalent to the DARE (2), through the Cayley transformation (22).

Applying the SDA [8] to the DARE in (2):

$$\left\{\begin{array}{lcl} G_{k+1} &=& G_k + A_k(I + G_kH_k)^{-1}G_kA_k^\top, \\ H_{k+1} &=& H_k + A_k^\top H_k(I + G_kH_k)^{-1}A_k, \\ A_{k+1} &=& A_k(I + G_kH_k)^{-1}A_k, \end{array}\right. \tag{3}$$

with $A_0 = A$, $G_0 = G = BR^{-1}B^\top$, $H_0 = H$, we obtain an efficient algorithm for the solution of the DARE with a high rank H. Recall from [8] that $H_k \to X$, $G_k \to Y$ (the solution to the dual equation (14) of the DARE (2)) and $A_k \to 0$, all quadratically ignoring errors. Similar to A_k for large-scale AREs in [36,37], H_k increases in complexity with respect to k thus cannot be evaluated or stored explicitly, but with its recursion utilized in products with vectors. For large-scale DAREs in [36], both G and H are low-rank and the SMWF can be applied for $(I + G_kH_k)^{-1}$ using the structures in G or H. For our case with a high-rank H, we rely only on the low-rank in G or B. There are complications involving the computation of residuals and convergence control in the SDA, because the high rank of the approximate solution H_k. As elaborated in Section 3, we output instead the approximate feedback gain which is low-rank.

2.1. SDA_h

The details of the SDA_h are described inductively as follows. Consider

$$G_k = B_kR_kB_k^\top, \quad H_k = M_k + C_kT_kC_k^\top, \tag{4}$$

with $B_0 = B$, $R_0 = R^{-1}$, $M_0 = H$, $C_0 = 0$ and $T_0 = 1$. We show (4) holds for all values of k by induction. Note that we used a different convention in [37], where R_k and T_k are written as inverses to emphasize their invertibility. In this paper, R_k, T_k and S_k are positive definite after the truncation and compression in Section 3.1.

First, the application of the SMWF on $I + G_kH_k$ implies that

$$(I_n + G_kH_k)^{-1} = I_n - B_k\left(I_{m_k} + R_kB_k^\top H_kB_k\right)^{-1}R_kB_k^\top H_k. \tag{5}$$

From (3) and (5), it is obvious that

$$G_{k+1} = G_k + A_k[I_n - B_k\left(I_{m_k} + R_kB_k^\top H_kB_k\right)^{-1}R_kB_k^\top H_k]G_kA_k^\top, \tag{6}$$

with $B_{k+1} = [B_k, A_kB_k]$ and $R_{k+1} = R_k \oplus \left[R_k - \left(I_{m_k} + R_kB_k^\top H_kB_k\right)^{-1}R_kB_k^\top H_kB_kR_k\right]$. Similarly for A_k, we have

$$A_{k+1} = A_k^2 - D_{k+1}^{(1)}S_{k+1}[D_{k+1}^{(2)}]^\top, \tag{7}$$

with $D_{k+1}^{(1)} = A_k B_k$, $D_{k+1}^{(2)} = A_k^\top H_k B_k$ and $S_{k+1} = \left(I_{m_k} + R_k B_k^\top H_k B_k\right)^{-1} R_k$. With $H_k = M_k + C_k T_k C_k^\top$, similar argument implies

$$\begin{aligned} H_{k+1} &= H_k + A_k^\top H_k \left[I_n - B_k \left(I_{m_k} + R_k B_k^\top H_k B_k\right)^{-1} R_k B_k^\top H_k\right] A_k \\ &= H_k + A_k^\top H_k A_k - D_{k+1}^{(2)} S_{k+1} [D_{k+1}^{(2)}]^\top, \end{aligned} \tag{8}$$

$$M_{k+1} = M_k + A_k^\top M_k A_k, \quad C_{k+1} = [C_k, A_k^\top C_k, A_k^\top H_k B_k],$$

$$T_{k+1} = T_k \oplus T_k \oplus \left[-\left(I_{m_k} + R_k B_k^\top H_k B_k\right)^{-1} R_k\right]. \tag{9}$$

These imply that A_{k+1} and H_{k+1} inherit the same forms as A_k and H_k in (7) and (8) respectively. From the compression and truncation in Section 3.1, we have B_k being orthogonal.

It is obvious that the full-rank H_k or M_k increases in complexity with increasing k thus cannot be evaluated or stored explicitly. The low-rank matrices $D_k^{(1)}$, $D_k^{(2)}$, S_k, C_k and T_k can be stored and the products of A_k, H_k and M_k with selected vectors can be evaluated recursively.

2.2. Structure of M_k

The structure of M_k has to be explored further because of its importance in our computation. With $M_0 = H$ and "l.m". representing low-rank matrices, we have

$$\begin{aligned} M_1 &= M_0 + A_0^\top M_0 A_0 = H + A^\top H A, \\ M_2 &= M_1 + A_1^\top M_1 A_1 = (H + A^\top H A) + A^\top (I + GH)^{-\top} A^\top (H + A^\top H A) A (I + GH)^{-1} A \\ &= \sum_{j=0}^{3} (A^\top)^j H A^j + \text{l.m.}, \quad \cdots \\ M_{k+1} &= M_k + A_k^\top M_k A_k = \sum_{j=0}^{2^k-1} (A^\top)^j H A^j + (A^\top)^{2^k} \left[\sum_{j=0}^{2^k-1} (A^\top)^j H A^j + \text{l.m.}\right] A^{2^k} + \text{l.m.} \\ &= \sum_{j=0}^{2^{k+1}-1} (A^\top)^j H A^j + \text{l.m.}. \end{aligned}$$

The low-rank matrices are easy to handle and can be absorbed into the $C_{k+1} T_{k+1} C_{k+1}^\top$ in H_{k+1} (as in (4)). Consequently, the main problem comes from the first term

$$\breve{M}_{k+1} \equiv \sum_{j=0}^{2^{k+1}-1} (A^\top)^j H A^j. \tag{10}$$

For CAREs, A has the form of a shifted inverse and the structure or sparsity in H will not be preserved even for small values of k.

2.3. Stein equations

Consider the special case of (2) with $G = 0$:

$$\mathcal{S}(X) \equiv -X + A^\top X A + H = 0. \tag{11}$$

The corresponding SDA becomes the Smith method, with the approximate solution H_k equal to $\breve{M}_{k+1}$ in (10), increasing in complexity. This gives rise to difficulties in convergence control and the evaluation of residuals and successive differences.

In many applications (such as the computation of Gramians in model-order reduction), we ultimately want to estimate the singular values of the solution X of (11). In that situation, the recursion for H_k can be utilized in such an estimation process, in the associated multiplications to some row and column vectors. Similar comments hold for the Lyapunov equations in Section 4.3. Note that this is in the same spirit as in Section 3.3, where the approximations H_k to X are not low-rank but the corresponding approximated feedback gains F_k in (18) and (27) are much simpler (in lower dimensions) thus cheaper to store or output.

3. Computational issues

3.1. Truncation and compression

Now we recall an important aspect of the SDA for large-scale DAREs — the growth of B_k. As $H_k = M_k + C_k^\top T_k C_k^\top$ is not computed explicitly and we may control the growth of C_k. We choose not to, because the recursions for A_k and H_k can be applied efficiently enough with only compression on B_k, as seen in (7)–(9).

Obviously, as the SDA converges, increasingly smaller but higher-rank components are added to B_k. Apparent from (6), s of these iterates is potentially exponential. If the convergence is slow relative to the growth in B_k, the algorithm will fail.

To reduce the dimensions of B_k, we compress their columns by orthogonalization. As in [36,37], consider the QR decompositions with column pivoting:

$$B_k = Q_k U_k + \widetilde{Q}_k \widetilde{U}_k, \quad \|\widetilde{U}_k\| \le \tau_g.$$

From here on, all norms are the 2-norm. Here τ_g is some small tolerance controlling the compression and truncation process, n_k is the number of columns in B_k bounded from above by $m_{\max}$ and its rank satisfies $r_k^{(b)} \equiv \operatorname{rank} B_k \le n_k \le m_{\max} \ll n$. Also $Q_k \in \mathbb{R}^{n\times r_k^{(b)}}$ is unitary and $U_k \in \mathbb{R}^{r_k^{(b)}\times n_k}$ is full-rank and upper triangular. We have

$$B_k R_k B_k^\top = Q_k \left(U_k R_k U_k^\top\right) Q_k^\top + O(\tau_g), \tag{12}$$

replacing respectively B_k and R_k by the leaner Q_k and $U_k R_k U_k^\top$. We ignore $O(\tau_g)$ terms, control the growth of $r_k^{(b)}$ while sacrificing a hopefully negligible bit of accuracy. We also restrict the widths of B_k, now relabelled $m_k = r_k^{(b)}$ after the compression and truncation in (12), by setting a reasonable upper limit $m_{\max}$. Similar procedures have been successfully applied in [36,37].

3.2. Residuals and differences of successive iterates

For the DARE (2) with $H_k = M_k + C_k T_k C_k^\top$, we have the usual relative residual for $H_k \approx X$:

$$\begin{aligned}\mathcal{D}(H_k) &= -H_k + A^\top H_k \left[I_n - B\left(R + B^\top H_k B\right)^{-1} B^\top H_k\right] A + H,\\ \widehat{r}_k &\equiv \frac{\|\mathcal{D}(H_k)\|}{\|H\| + \|H_k\| + \|A^\top H_k A\| + \|A^\top H_k B(R + B^\top H_k B)^{-1} B^\top H_k A\|},\end{aligned} \tag{13}$$

analogous to (28), with the help of (2) or the SMWF. This simplifies to $\mathcal{D}(H_k) =$

$$-M_k + A^\top M_k A + H + \left[-C_k T_k C_k^\top + A^\top C_k T_k C_k^\top A - A^\top H_k B_k (I + R_k B_k^\top H_k B_k)^{-1} R_k B_k^\top H_k A\right].$$

The norms in (13) are not easy to compute, although can be estimated using the power method [9], which may be attempted after convergence.

Next consider the differences of successive H_k and G_k, respectively:

$$\begin{aligned}\delta H_k &\equiv H_{k+1} - H_k = (M_{k+1} - M_k) + (C_{k+1} T_{k+1} C_{k+1}^\top - C_k T_k C_k^\top)\\ &= A_k^\top M_k A_k + [A_k^\top C_k, A_k^\top H_k B_k]\left[T_k \oplus \left(I_{m_k} - R_k B_k^\top H_k B_k\right)^{-1} R_k\right][A_k^\top C_k, A_k^\top H_k B_k]^\top,\\ \delta G_k &\equiv G_{k+1} - G_k = B_{k+1} R_{k+1} B_{k+1}^\top - B_k R_k B_k^\top = A_k B_k \Delta G_k B_k^\top A_k^\top,\end{aligned}$$

with δG_k being low-rank and $\Delta G_k \equiv R_k - \left(I_{m_k} + R_k B_k^\top H_k B_k\right)^{-1} R_k B_k^\top H_k B_k R_k$. Both differences δH_k and δG_k converge to zero quadratically while $A_k \to 0$ as $k \to \infty$. Obviously, convergence may be controlled efficiently using δG_k, after $A_k B_k$ has been orthogonalized and δG_k transformed as in Section 3.1, leading to $\|\delta G_k\| = \|\Delta G_k\|$. Note that $\delta G_k \in \mathbb{R}^{n\times n}$ while $\Delta G_k \in \mathbb{R}^{m_k \times m_k}$ with $m_k \ll n$.

3.2.1. Convergence control and accuracy of H_k

From the previous section, it is obvious that the computation of residuals for H_k and the similar quantities is nontrivial and costly. In this section, we describe an alternative of convergence control which bypasses these difficulties. First we quote the following theorem for the SDA:

Theorem 3.1 (*[41, Theorem 3.1]*). *Assume that $X, Y > 0$ satisfy the DARE (1) and its dual:*

$$\mathcal{D}_a(Y) \equiv -Y + AY(I + HY)^{-1} A^\top + G = 0, \tag{14}$$

and let $S \equiv (I + GX)^{-1} A$ and $T \equiv (I + HY)^{-1} A^\top$. Then the matrix sequences $\{A_k\}$, $\{G_k\}$ and $\{H_k\}$ generated by the SDA satisfy

(a) $A_k = (I + G_k X) S^{2^k}$;

(b) $H \le H_k \le H_{k+1} \le X$ and $X - H_k = \left(S^\top\right)^{2^k} (X + X G_k X) S^{2^k} \le \left(S^\top\right)^{2^k} (X + XYX) S^{2^k}$;

(c) $G \le G_k \le G_{k+1} \le Y$ and $Y - G_k = \left(T^\top\right)^{2^k} (Y + Y H_k Y) T^{2^k} \le \left(T^\top\right)^{2^k} (Y + YXY) T^{2^k}$.

Theorem 3.2. *implies that the relative accuracy of H_k and G_k are bounded as below:*

$$r_k^{(x)} \equiv \frac{\|H_k - X\|}{\|X\|} \le \|S\|^{2^{k+1}} \|I + XY\|, \quad r_k^{(y)} \equiv \frac{\|G_k - Y\|}{\|Y\|} \le \|T\|^{2^{k+1}} \|I + XY\|. \tag{15}$$

It is well-known [41] that the spectral radii $\rho(S) = \rho(T)$.
With Theorem 3.2, we can control the convergence of the SDA_h in the following fashion:

(i) Within an iterative step, compute the (dual) residual $r_k \equiv \|\mathcal{D}_a(G_k)\|$ in $O(n)$ complexity, using the low-rank structure of G_k. Terminate iteration when r_k is small within some tolerance, relative to the data in the dual DARE (as in (28)). This is not that controversial, comparing to the use of $\|A_k\|$ in [2,8]. The accuracy achieved in H_k is further guaranteed in (ii) and (iii) below.
(ii) The relative error of H_k is bounded as in (15) with $\|S\| = \|(I + GX)^{-1}A\|$ estimated (with power method [9]). This ignores the effects of any truncation and compression of G_k and H_k, thus underestimates the error.
(iii) Further indication of the accuracy of H_k can be obtained by applying the power method [9] to estimate δH_k, although only reflecting the relative error of H_{k-1} (thus overestimating the relative error for H_k). This can be costly and should be attempted only after convergence (in the sense of r_k in (i) above).
(iv) The relative residual $\widehat{r}_k$ in (13) is expensive to evaluate. We have applied the power method to estimate the residual after convergence for Example 3 (in Section 5) with $n = 1357$. We estimated $\widehat{r}_6 = 4.12 \times 10^{-14}$, consistent with the accuracy of other estimates $dH_5 = 8.70 \times 10^{-9}$ and $\phi_6 = 3.72 \times 10^{-27}$. However, this estimation costed more than 4 times the CPU time for the solution of the CARE. The estimation of the relative residuals for higher values of n are even more expensive.
Note that residuals do not reflect directly on the accuracy of the approximate solutions, without the knowledge of the corresponding condition numbers.
(v) For CAREs, the theorem and similar techniques can be applied after the Cayley transform.

3.2.2. Estimation of $\|S\|$ and $\|I + XY\|$

After convergence, for $r_k^{(x)}$ in (15), we have

$$\|I + XY\| \approx \|I + H_kG_k\| = \|I + G_kH_k\|,\ \|S\| \approx \|(I + GH_k)^{-1}A\|.$$

With the help of the SMWF, we have

$$\begin{aligned}&\|I + XY\|^2 \approx \|I + G_kH_k\|^2 = \|I + B_kR_kB_k^\top H_k\|^2\\ &= \rho\left(I + B_kR_kB_k^\top H_k + H_kB_kR_kB_k^\top + B_kR_kB_k^\top H_k^2B_kR_kB_k^\top\right) = 1 + \rho\left(\widehat{B}_k\widehat{R}_k\widehat{B}_k^\top\right),\end{aligned} \tag{16}$$

with

$$\widehat{B}_k \equiv [B_k,\ H_kB_k],\quad \widehat{R}_k \equiv \begin{bmatrix} R_kB_k^\top H_k^2B_kR_k & R_k\\ R_k & 0\end{bmatrix},$$

$$\begin{aligned}\|S\|^2 &\approx \|(I + GH_k)^{-1}A\|^2 = \|A - B\left(I + R^{-1}B^\top H_kB\right)^{-1}R^{-1}B^\top H_kA\|^2\\ &= \rho\Big\{AA^\top - B\left(I + R^{-1}B^\top H_kB\right)^{-1}R^{-1}B^\top H_kAA^\top\\ &\quad - AA^\top H_kBR^{-1}(I + B^\top H_kBR^{-1})^{-1}B^\top\\ &\quad + B\left(I + R^{-1}B^\top H_kB\right)^{-1}R^{-1}B^\top H_kAA^\top H_kBR^{-1}(I + B^\top H_kBR^{-1})^{-1}B^\top\Big\}\\ &= \rho\left(AA^\top + \breve{B}_k\breve{R}_k\breve{B}_k^\top\right),\end{aligned} \tag{17}$$

where $\widetilde{R}_k \equiv (I_{m_k} + R^{-1}B^\top H_kB)^{-1}R^{-1}$, $\breve{B}_k \equiv [B, AA^\top H_kB]$ and

$$\breve{R}_k \equiv \begin{bmatrix} \widetilde{R}_kB^\top H_kAA^\top H_kB\widetilde{R}_k^\top & -\widetilde{R}_k\\ -\widetilde{R}_k^\top & 0\end{bmatrix}.$$

Consequently, after orthogonalizing $\widehat{B}_k$ in (16) and modifying $\widehat{R}_k$, we have $\|I + XY\|^2 \approx \|I + G_kH_k\|^2 = 1 + \rho(\widehat{R}_k)$. We also have $\|S\| \approx \|(I + GH_k)^{-1}A\| = [\rho(AA^\top + \breve{B}_k\breve{R}_k\breve{B}_k^\top)]^{1/2}$, involving the spectral radius of a symmetric low-rank update of the sparse matrix $AA^\top$. The right hand side of (15) can be estimated using the power method [9].

3.3. Feedback gains

When solving large-scale problems, it is impossible to output the high-rank approximate solution $H_k \in \mathbb{R}^{n\times n}$ explicitly. For large-scale AREs and (non-)linear matrix equations in [36,37,42], for instance, the solutions can be represented approximately as numerically low-rank matrices. These can then be stored in an $O(n)$ memory requirement. For our large-scale ARE with a high-rank H, the approximate solution H_k is represented recursively as in (4) and (9) in a nontrivial manner. These difficulties can be bypassed by outputting the estimated feedback gain $F_k \equiv -(R+B^\top H_kB)^{-1}B^\top H_kA \in \mathbb{R}^{m\times n}$, approximating the feedback gain $F_d \equiv -(R + B^\top XB)^{-1}B^\top XA$, for the associated LQR control.
From (4), (7) and (9), we have

$$F_{k+1} = -(R + B^\top H_{k+1}B)^{-1}B^\top H_{k+1}A, \tag{18}$$

with $F_0 = -(R + B^\top HB)^{-1}B^\top HA$. Expanding (18) in terms of F_k does not lead to a more efficient updating formula.

For convergence control, we may utilize

$$\begin{aligned}\delta F_k &\equiv F_{k+1} - F_k \\ &= -(R + B^\top H_{k+1}B)^{-1}B^\top H_{k+1}A + (R + B^\top H_k B)^{-1}B^\top H_k A \\ &= -(R + B^\top H_{k+1}B)^{-1}B^\top \delta H_k \left[I_n - B(R + B^\top H_k B)^{-1}B^\top H_k\right] A.\end{aligned}$$

3.4. Algorithm and operation counts

We summarize the SDA_h below. We only truncate and compress B_{k+1} but with M_{k+1}, T_{k+1} and C_{k+1} in (9) implicitly containing $H_{k+1} = H_k + A_k^\top H_k A_k - D_{k+1}^{(2)} S_{k+1}[D_{k+1}^{(2)}]^\top$. Practically, we only update S_{k+1}, $A_{k+1}B_{k+1}$, $H_{k+1}B_{k+1}$ and $A_{k+1}^\top H_{k+1}V_{k+1}$ with $V_{k+1} = B_{k+1}$ or $A_{k+1}B_{k+1}$.

Recall that the maximum number of nonzero elements in any row or column in A is bounded from above by c_a, for flop counts in the second column in Table 1. In the third column, the number of variables is recorded. Only the dominant operations or memory requirements are included. Most of the work occurs in computing $\widehat{B}_{k+1}$, for which $A_k B_k$ has to be calculated recursively, as A_k and H_k are not available explicitly. We use the notation $N_k \equiv \sum_{j=1}^{k} m_j$ and a QR decomposition of an $n \times r$ matrix requires $4nr^2$ flops [9, p. 250].

For the details, let s_k and s'_k be the numbers of flops for the products of an arbitrary row or column vectors to A_k and H_k, respectively. After some tedious counting, we deduce that

$$\begin{bmatrix} s_k \\ s'_k \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 + m_{k-1} & 2 \end{bmatrix} \begin{bmatrix} s_{k-1} \\ s'_{k-1} \end{bmatrix} + n \begin{bmatrix} 1 + 5m_{k-1} \\ 0.5m_{k-1}^2 + 4m_{k-1} + 2 \end{bmatrix}.$$

Consequently, for $k \geq 1$, we have $s_k, s'_k \leq \psi_k n$, with $s_0, s'_0 \leq 2c_a mn$ and

$$\begin{aligned}\psi_k &\leq 2(2 + m_{\max})^k c_a m + \frac{1}{4}(m_{\max}^2 + 18m_{\max} + 6)\sum_{j=0}^{k}(2 + m_{\max})^j \\ &< (2 + m_{\max})^k \left[2c_a m + \frac{k+1}{4}(m_{\max}^2 + 18m_{\max} + 6)\right]. \end{aligned} \tag{19}$$

Algorithm 1 (**SDA_h for DAREs with High-Rank** H)

Input: $A, H \in \mathbb{R}^{n\times n}$ and orthogonal $B \in \mathbb{R}^{n\times m}$, $R^{-1} = R^{-\top} \in \mathbb{R}^{m\times m}$, positive tolerances τ_g and ϵ, and $m_{\max}$;

Output: $B_\epsilon \in \mathbb{R}^{n\times m_\epsilon}$, $R_\epsilon \in \mathbb{R}^{m_\epsilon \times m_\epsilon}$, feedback gain $F_\epsilon \in \mathbb{R}^{m\times n}$, dual residual $\widetilde{r}_\epsilon$ and estimated relative error $r_\epsilon^{(x)}$;

Set $A_0 = A$, $D_0^{(1)} = D_0^{(2)} = 0 \in \mathbb{R}^{n\times l}$, $S_0 = I_l$, $B_0 = B$, $R_0 = R^{-1}$, $M_0 = H$, $C_0 = 0$ and $T_0 = 1$; $m_0 = m$, $\widetilde{r}_0 = 2\epsilon$ and $k = 0$;

Do until convergence:

If $\widetilde{r}_k \leq \epsilon$, Set $B_\epsilon = B_k$, $R_\epsilon = R_k$ and $\widetilde{r}_\epsilon = \widetilde{r}_k$;

Estimate $r_\epsilon^{(x)}$ as in (15); Exit; End If

Compute $\widehat{B}_{k+1} = [B_k,\ A_k B_k]$,

$\widehat{R}_{k+1} = R_k \oplus \left[R_k - \left(I_{m_k} + R_k B_k^\top H_k B_k\right)^{-1} R_k B_k^\top H_k B_k R_k\right]$,

$D_{k+1}^{(1)} = A_k B_k$, $D_{k+1}^{(2)} = A_k^\top H_k B_k$,

$S_{k+1} = \left(I_{m_k} + R_k B_k^\top H_k B_k\right)^{-1} R_k$;

with the recursions:

$A_{k+1} = A_k^2 - D_{k+1}^{(1)} S_{k+1}\left[D_{k+1}^{(2)}\right]^\top$,

$H_{k+1} = H_k + A_k^\top H_k A_k - D_{k+1}^{(2)} S_{k+1}[D_{k+1}^{(2)}]^\top$.

Compute B_{k+1} by truncating $\widehat{B}_{k+1}$ w.r.t. τ_g;

Set $R_{k+1} \leftarrow R_{k+1}^\tau$, $m_{k+1} \leftarrow \text{rank}B_{k+1}$;

If $m_{k+1} > m_{\max}$, Set $m_{k+1} \leftarrow m_{\max}$;

Compute $r_{k+1} \equiv \|\mathcal{D}_a(B_{k+1}R_{k+1}B_{k+1}^\top)\|$;

Compute $q_k \equiv \|G\| + \|G_k\| + \|AG_k(I + HG_k)^{-1}A^\top\|$;

Compute $\widetilde{r}_{k+1} \equiv r_{k+1}/q_k$; End If

Set $k \leftarrow k + 1$.

End Do

Recall that m_k is controlled by the truncation in Section 3.1. In our numerical examples in Section 5, the flop count at the end of Algorithm 1 dominates. This corresponds to the $(2 + m_{\max})^k$ factor in the upper bound of the total flop count. With respect to k, this exponentially growing factor looks menacing. Nonetheless, it behaves as a constant with respect

Table 1
Operation and memory counts for the kth iteration in Algorithm 1 (SDA_h).

Computation	Flops	Memory
$\widehat{B}_{k+1}$	$m_k s_k$	$N_{k+1}n$
$\widehat{R}_{k+1}$	$m_k s'_k + 3m_k^2 n$	$O(m_k^2)$
$D_{k+1}^{(1)}$, $D_{k+1}^{(2)}$	$m_{k+1}\left[(s_k + s'_k) + 2m_k n\right]$	–
S_{k+1}	$m_k(s'_k + m_{k+1}n)$	–
B_{k+1}	$4m_k^2 n$	–
R_{k+1}	$O(m_k^3)$	–
$\widetilde{r}_{k+1}$	$8m_k^2 n$	–
Total	$\left[(m_k + m_{k+1})s_k + (2m_k + m_{k+1})s'_k + 3m_k(5m_k + m_{k+1})n\right]$	$N_{k+1}n$

to n, especially with the fast (near) quadratic convergence of the SDA without (with) truncation. This is evident for the examples in Section 5, requiring only 5 to 6 iterations to machine accuracy.

Note that we have not included, in Table 1, the workload for the estimation of ϕ_k (in (30)), $\|\delta H_k\|$ (in (29)) or $\|\widehat{r}_k\|$ (in (13)) by power method. As elaborated later, the associated computation is relatively expensive and should only be conducted selectively after convergence; see Examples 3 and 4 for the corresponding CPU times.

3.5. Error analysis

A detailed error analysis for the SDA can be found in [36, Section 3]. Here, we just repeat an abbreviated error analysis for completeness, as from [37, Section 2.4].

Let δG_k, δH_k and δA_k be the errors in G_k, H_k and A_k, respectively, from round-off or the truncation process. Let $\delta_k \equiv \max\{\|\delta G_k\|, \|\delta H_k\|, \|\delta A_k\|\}$ and $\tau \equiv \max\{\tau_g, \tau_h\}$. From the SDA_h in (3) and the truncation process in Section 3.1, we can show that

$$\delta_{k+1} \le (1 + c_k)(\delta_k + \tau) + O((\delta_k + \tau)^2), \tag{20}$$

with $a_k \equiv \|A_k\|$, $\alpha_k \equiv \|(I+G_kH_k)^{-1}\|$, $\beta_k \equiv \max\{\|G_k\|, \|H_k\|\}$ and $c_k \le a_k\alpha_k(2\beta_k+2a_k\alpha_k\beta_k^2+a_k)$. When the SDA_h converges, we have α_k, β_k converging to constants and a_k to zero as $k \to \infty$. This indicates that the error δ_k is not magnified through (3). After N iterations, we have

$$\delta_{N+1} \le \mathcal{M}_0^{(N)}\delta_0 + \left[\sum_{j=1}^{N} \mathcal{M}_j^{(N)}\right]\tau + \text{HOTs} \le \mathcal{M}_0^{(N)}(\delta_0 + N\tau) + \text{HOTs}, \tag{21}$$

with the error magnifying factors $\mathcal{M}_j^{(N)} \equiv \prod_{k=j}^{N}(1+c_k)$. Here HOTs denote the higher order terms in δ_k ($k = 0, \ldots, N$) and τ. With the SDA_h converges reasonably quickly and $c_k \to 0$, $\mathcal{M}_0^{(N)}$ will not be too far away from unity. These indicate that the errors in SDA_h behave well and Algorithm 1 is stable in the sense of (20) or (21).

4. Continuous-time algebraic Riccati equations

For CAREs, we assume that linear systems associated with $A - \gamma I$ and its transpose are efficiently solvable in $\widetilde{c}_a n$ flops. In other words we assume that $\widetilde{c}_a$ is not large. For A being banded, for example, we have $\widetilde{c}_a = O(1)$ when Gaussian elimination with partial pivoting is applied.

4.1. Cayley transform

From [2], the matrices A, G and H in the CARE (1) are first treated with the Cayley transform:

$$A_0 = I + 2\gamma\widetilde{A}_\gamma, \quad G_0 = 2\gamma A_\gamma^{-1}G\widetilde{A}_\gamma, \quad H_0 = 2\gamma\widetilde{A}_\gamma H A_\gamma^{-1}, \tag{22}$$

with $\widetilde{A}_\gamma \equiv \left(A_\gamma + GA_\gamma^{-\top}H\right)^{-1}$, $A_\gamma \equiv A - \gamma I$ and $\gamma > 0$. The SMWF implies

$$\widetilde{A}_\gamma = A_\gamma^{-1} - A_\gamma^{-1}B \cdot \left(I_m + R^{-1}B^\top A_\gamma^{-\top}HA_\gamma^{-1}B\right)^{-1} R^{-1} \cdot B^\top A_\gamma^{-\top}HA_\gamma^{-1}. \tag{23}$$

With the above initial A_0, G_0 and H_0, the SDA_h still works, again with exactly the same forms and updating formulae for A_k, B_k, $D_k^{(1)}$, $D_k^{(2)}$ and the kernels R_k and S_k. One relevant difference for CAREs is that $A_0 \ne A$ but satisfies, from (22) and (23),

$$A_0 = \left(I_n + 2\gamma A_\gamma^{-1}\right) - D_0^{(1)}S_0\left[D_0^{(2)}\right]^\top, \quad B_0 = A_\gamma^{-1}B. \tag{24}$$

The corresponding rank m perturbed update has the form

$$D_0^{(1)} = B_0, \quad D_0^{(2)} = A_\gamma^{-\top} H B_0, \quad S_0 = 2\gamma \left(I_m + R^{-1} B_0^\top H B_0\right)^{-1} R^{-1}. \tag{25}$$

Note that all computations can be realized efficiently, with $A_\gamma^{-1} B$ computed in $O(\widetilde{c}_a mn)$ flops.

Similarly, we have $C_0 = 0$, $T_0 = 1$ and

$$R_0 = 2\gamma \left[R^{-1} - R^{-1} B_0^\top H B_0 \left(I_m + R^{-1} B_0^\top H B_0\right)^{-1} R^{-1}\right]. \tag{26}$$

4.2. Algorithm and operation counts

The initial A_0, G_0 and H_0 in (22) are different for CAREs, as compared to (3) for DAREs, because of the Cayley transform. These create some differences in the SDA_h for CAREs, which are summarized in this section.

Residuals

For the CARE (1), we have the residual for H_k:

$$\begin{aligned} \check{r}_k \equiv \mathcal{C}(H_k) &= \mathcal{C}(M_k) + [\mathcal{C}(C_k T_k C_k^\top) - H] - M_k B R^{-1} B^\top C_k T_k C_k^\top - C_k T_k C_k^\top B R^{-1} B^\top M_k, \\ \mathcal{C}(M_k) &= A^\top M_k + M_k A - M_k B R^{-1} B^\top M_k + H, \\ \mathcal{C}(C_k T_k C_k^\top) - H &= A^\top C_k T_k C_k^\top + C_k T_k C_k^\top A - C_k T_k C_k^\top B R^{-1} B^\top C_k T_k C_k^\top. \end{aligned}$$

As for DAREs in Section 3.2, the norm of the residual is not easy to compute, although may be estimated using the power method [9] with high additional costs after convergence.

Feedback gain

For continuous-time, we have the simpler update for the approximate feedback gain:

$$\begin{aligned} F_{k+1} &= -R^{-1} B^\top H_{k+1} = -R^{-1} B^\top (M_{k+1} + C_{k+1} T_{k+1} C_{k+1}^\top) \\ &= -R^{-1} B^\top \left[(M_k + A_k^\top M_k A_k) + C_{k+1} T_{k+1} C_{k+1}^\top\right] \\ &= F_k - R^{-1} B^\top (A_k^\top M_k A_k + \check{C}_{k+1} \check{T}_{k+1} \check{C}_{k+1}^\top), \end{aligned} \tag{27}$$

with $\check{C}_{k+1} = [A_k^\top C_k, A_k^\top H_k B_k]$, $\check{T}_{k+1} = T_k \oplus \left[-\left(I_{m_k} + R_k B_k^\top H_k B_k\right)^{-1} R_k\right]$ and $F_0 = -R^{-1} B^\top H$. The computation in (18) and (27) can be realized in $O(n)$ complexity.

As for the differences of successive iterates, we have

$$\delta F_k = -R^{-1} B^\top (A_k^\top M_k A_k + \check{C}_{k+1} \check{T}_{k+1} \check{C}_{k+1}^\top).$$

Finally, recall that H_k cannot be explicitly stored or output except in terms of C_k, T_k and the implicit recursion in M_k. These can be used to retrieve the approximate feedback gain F_k.

Operation counts

The operation counts in Table 1 are still valid for CAREs, with $c_a n$ replaced by $\widetilde{c}_a n$, the flop count for one solve of the linear system associated with $A - \gamma I$ or its transpose. There will be an additional start-up cost associated with A_0, B_0 and R_0 in (22), (24), (25) and (26). This amounts to $4\widetilde{c}_a mn$ flops. As for the different costs when A_0 and H_0 are multiplied to row or column vectors, we have to replace c_a in (19) and Table 1 by $2\widetilde{c}_a + c_a + 4m_k + 2$. Qualitatively, the operation counts are of similar orders for both DAREs and CAREs, with $\widetilde{c}_a$ playing a part for CAREs because of the linear systems associated with A_γ or its transpose.

4.3. Lyapunov equations

Consider a special case of (1) with $G = 0$, the Lyapunov equation:

$$\mathcal{C}_0(X) \equiv A^\top X + XA + H = 0.$$

As in Section 2.3 for Stein equations, the singular values of X may be required. Multiplications of the approximate solution (in recursion form) to given vectors can then be realized efficiently.

Table 2
Example 1 ($n = 10{,}000$, $\gamma = 1$, $\tau_g = 10^{-15}$, $m_{\max} = 50$).

k	dH_k	e_k	ϕ_k	r_k	$\widetilde{r}_k$	m_k	δt_k	t_k
1	1.70e−02	2.42e−02	3.13e−02	3.38e−02	3.58e−02	1	0.022	0.022
2	2.48e−02	5.83e−03	1.24e−03	2.19e−03	2.19e−03	2	1.332	1.354
3	4.44e−04	2.28e−05	1.58e−06	8.58e−06	8.58e−06	2	1.331	2.685
4	1.09e−07	1.25e−10	2.49e−12	1.30e−10	1.30e−10	2	3.605	6.290
5	6.52e−15	0.00e−00	6.14e−24	5.75e−17	5.75e−17	2	4.821	11.482

Table 3
Example 1 ($n = 100{,}000$, $\gamma = 1$, $\tau_g = 10^{-15}$, $m_{\max} = 50$).

k	dH_k	e_k	ϕ_k	r_k	$\widetilde{r}_k$	m_k	δt_k	t_k
1	1.70e−02	2.42e−02	3.27e−02	3.38e−02	3.58e−02	1	0.232	0.232
2	2.48e−02	5.83e−03	1.35e−03	2.19e−03	2.19e−03	2	14.702	14.934
3	4.44e−04	2.28e−05	1.88e−06	8.58e−06	8.58e−06	2	14.390	29.324
4	1.09e−07	1.71e−10	3.55e−12	1.30e−10	1.30e−10	2	40.495	69.819
5	6.52e−15	0.00e−00	1.25e−23	5.75e−17	5.75e−17	2	91.299	163.574

5. Numerical examples

Here we illustrate the effectiveness of the SDA_h. Only numerical examples with CAREs, which are obviously more difficult to solve because of the Cayley transform involved, are presented here. Algorithm 1 was coded in MATLAB 2010b [11] on a 64-bit PC with 3.4 GHz Intel Core i3 processor and 8G RAM in the Hunan University of Technology. The machine error is reflected by $\mathtt{eps} = 2.22 \times 10^{-16}$ in MATLAB. Recall that $r_k = \|\mathcal{D}_a(G_k)\|$ is the residual for G_k for the dual equation (14). The stopping criterion in terms of the relative residual $\widetilde{r}_k$ for G_k is

$$\widetilde{r}_k = \frac{r_k}{\|G\| + \|G_k\| + \|AG_k(I + HG_k)^{-1}A^\top\|} \le \epsilon, \tag{28}$$

for a small tolerance $\epsilon > 0$. We also compute the estimated relative error for H_{k-1}:

$$dH_{k-1} \equiv \frac{\|\delta H_{k-1}\|}{\|H_k\|} = \frac{\|H_{k-1} - H_k\|}{\|H_k\|} \tag{29}$$

the upper bound of the relative error $r_k^{(x)}$ in (15):

$$\phi_k = \|(I + GH_k)^{-1}A\|^{2^{k+1}} \cdot \|I + G_kH_k\|, \tag{30}$$

and the normalized residual $\widehat{r}_k$ in (13), estimated by the power method [9] (after convergence). We add the symbol "$*$" when the power method is terminated prematurely after 150 iterations.

Design of numerical examples

Because of the high-rank structure in H (thus H_k and X), the convergence control in (28) is performed making use of the low-rank G_k. For the accuracy of the approximate solution H_k for Examples 3 and 4, we have *estimated* dH_k and ϕ_k for the relative errors, as well as the normalized residual $\widehat{r}_k$, using the power method. Examples 1 and 2 are designed with known explicit solutions. Example 2 shows that SDA_h produces approximate solutions at almost the same accuracy as the original SDA in (3) and various estimates of relative errors and residuals are reasonably consistent. Similarly, Examples 2–4, modifying real-life applications, illustrate the accuracy and efficiency of the SDA_h.

Example 1. We first tested SDA_h on a simple system with $A = -I_n$, $B = (1, 0, \ldots, 0, 1)^\top$, $R = 1$ and $H = 2I_n + BB^\top$ with $n = 10{,}000$, $100{,}000$, $1{,}000{,}000$. The solution of CARE is $X = I_n$ and the feedback gain $F = -B^\top$. We set the truncation tolerance $\tau_g = 10^{-15}$ for B_k and its maximum width $m_{\max} = 50$. We stopped when $\widetilde{r}_k \le \epsilon = 10^{-15}$. The results are listed in Tables 2–4, where $e_k = \|H_k - I_n\|$ evaluates the true relative accuracy at iteration k, $t_k = \sum_{i=1}^{k} \delta t_i$ is the sub-total CPU time and δt_i is the CPU-time required for the ith iteration. The relative errors, known exactly, reflect the accuracies of H_k accurately.

Example 2. We use the real steel profile data from [18], replacing the original matrix A by $A - I_n$. We assign $H = BB^\top - A^\top - A$ for $X = I_n$. We construct explicitly A_{k+1} and H_{k+1} from $D_{k+1}^{(1)}$, S_{k+1}, $D_{k+1}^{(2)}$, A_k and H_k (see also (7) and (8)). As a comparison, we also computed $\widehat{H}_{k+1}$ by the original SDA in (3). Note that only ($n = 1357$, $m = 7$) and ($n = 5177$, $m = 7$) are considered as the SDA fails for larger n. The accumulated CPU times of iterations for various dimensions is plotted in Fig. 1 and the relative errors $\mathrm{E}_h = \|H_k - I\|_2$ and $\mathrm{E}_d = \|\widehat{H}_k - I\|_2$ are listed in Table 5.

The ranks of the approximate solutions were recorded as m_k and SDA_h stopped when the relative residual $\widetilde{r}_k \le 5.0 \times 10^{-15}$. We see from Fig. 1 that SDA_h beats the original SDA in CPU time. Table 5 shows that SDA_h attained almost the same accuracy as the SDA and ϕ_5 (for $n = 1357$) or ϕ_6 (for $n = 5177$) were good estimates of the relative errors.

Table 4
Example 1 ($n = 1{,}000{,}000$, $\gamma = 1$, $\tau_g = 10^{-15}$, $m_{\max} = 50$).

k	dH_k	e_k	ϕ_k	r_k	$\widetilde{r}_k$	m_k	δt_k	t_k
1	1.70e−02	2.42e−02	3.31e−02	3.38e−02	3.58e−02	1	2.232	2.232
2	2.48e−02	5.83e−03	1.39e−03	2.19e−03	2.19e−03	2	219.000	221.232
3	4.44e−04	2.28e−05	1.99e−06	8.58e−06	8.58e−06	2	220.473	441.669
4	1.09e−07	2.08e−10	3.96e−12	1.30e−10	1.30e−10	2	616.191	1057.860
5	6.52e−15	0.00e−00	1.57e−23	5.75e−17	5.75e−17	2	1599.382	2695.124

Table 5
Relative error for SDA and SDA_h in Example 2.

$n = 1357$, $\gamma = 4.35$, $\tau_g = 10^{-30}$, $m_{\max} = 190$					
k	E_h	E_d	m_k	r_k	$\widetilde{r}_k$
1	3.65e−02	3.65e−02	14	2.24e−08	3.15e−03
2	1.33e−03	1.33e−03	28	4.05e−10	5.67e−05
3	1.78e−06	1.78e−06	56	2.49e−13	3.49e−08
4	3.19e−12	3.19e−12	112	2.56e−19	3.59e−14
5	1.88e−15	1.88e−15	190	1.06e−20	1.48e−15
	$dH_4 = 3.19e{-}12$		$\phi_5 = 5.15e{-}22$		
$n = 5177$, $\gamma = 4.35$, $\tau_g = 10^{-30}$, $m_{\max} = 190$					
k	E_h	E_d	m_k	r_k	$\widetilde{r}_k$
1	4.12e−01	4.12e−01	14	7.93e−07	2.31e−02
2	1.70e−01	1.70e−01	28	1.58e−07	4.54e−03
3	2.89e−02	2.89e−02	56	1.06e−08	3.02e−04
4	8.37e−04	8.37e−04	112	9.06e−11	2.58e−06
5	7.01e−07	7.01e−07	190	4.01e−14	1.14e−09
6	4.91e−14	4.91e−14	190	7.36e−20	2.09e−15
	$dH_5 = 4.43e{-}11$		$\phi_6 = 1.19e{-}23$		

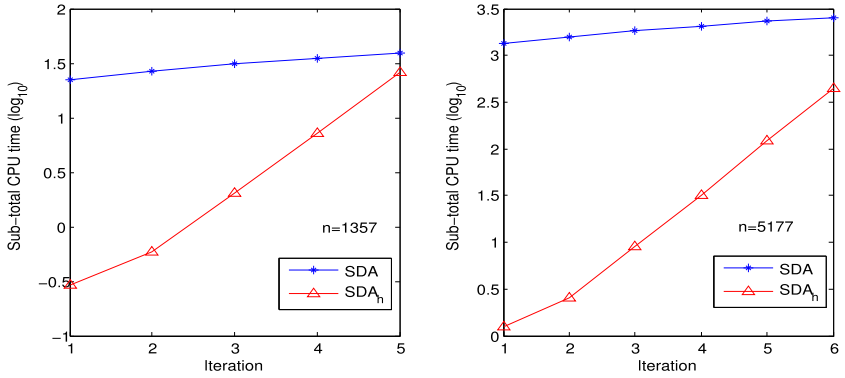

Fig. 1. Sub-total CPU time with log for SDA and SDA_h in Example 2.

Example 3. We apply SDA_h on the steel profile example from [18], with A modified to $A - (\|A\|/2)I_n$ and $H = I_n$, for $n = 1357, 5177, 20209, 79841$ and $m = 7$.

The rank m_k was capped at 190 and SDA_h was terminated when $\widetilde{r}_k \le \epsilon = 5.0\times 10^{-15}$. The accuracy of the approximate solution H_6 achieve high relative residuals of $O(10^{-15})$ efficiently. Adjusting the values of γ (the shift in the Cayley transform from CAREs to DAREs), τ_g and the maximum rank $m_{\max}$ may improve the accuracy of the approximate solution or the speed of convergence further. In our experiments, higher accuracy could also be obtained by setting m_k to a larger positive integer greater than 190, with higher CPU-times and RAM requirements.

The corresponding results are plotted in Fig. 2, where the left and the right vertical axes (in $\log_{10}$ scale) in each subplot are for the residuals (r_k or $\widetilde{r}_k$, with the latter represented by Rr_k in the legend) and CPU Time's (δt_k or t_k), and the horizontal axis records the iteration number and the corresponding truncated rank. The graphs show the residuals decline gently at first and then decrease sharply. Also, δt_k and t_k rose exponentially as predicted. The SDA_h attains good accuracy, with $\widetilde{r}_6 = 4.76\times 10^{-15}, 4.12\times 10^{-14}, 5.80\times 10^{-12}, 2.95\times 10^{-10}$ (represented by Hr_k in Figs. 2 and 3) for $n = 1357, 5177$, 20,209, 79,841 respectively. The iterations have been terminated after 6 iterations (with $m_{\max} = 190$ reached), with $\widetilde{r}_k$ less than $O(10^{-15})$. We have not iterated further because of costs.

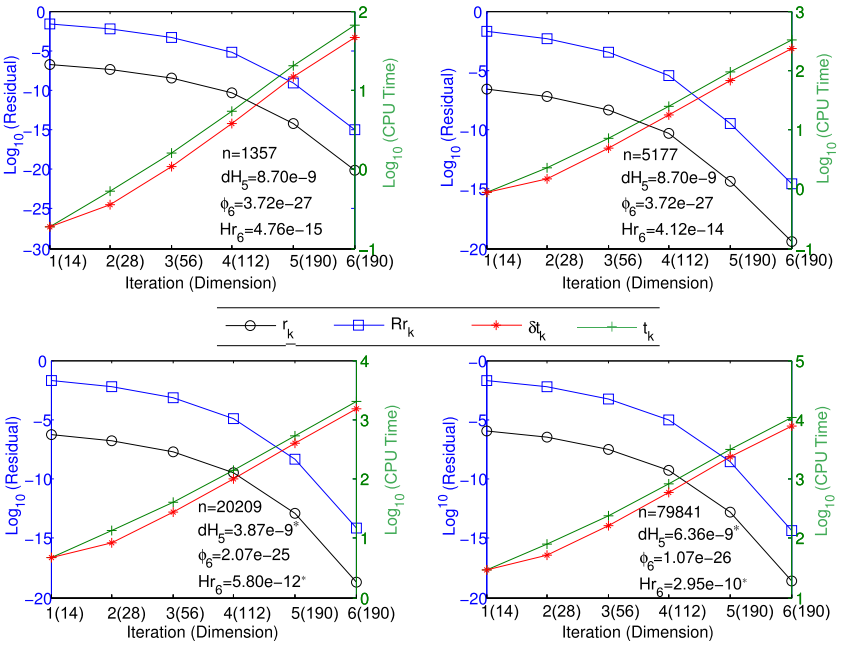

Fig. 2. Residual and CPU time (in $\log_{10}$) for SDA_h in Example 3.

The additional CPU-times for the estimation of dH_5 (or $\widehat{r}_6$) for $n = 1357$, 5177, 20,209, 79,841 are respectively around 4, 35, 178, 842 (or 5, 38, 182, 854) minutes, with the last two calculations failing to converge within 150 iterations in the power method [9] (starred but still producing reasonably accurate estimates). This illustrates the difficulties mentioned before, in the computation associated with H_k because of its high-rank and recursive structure.

The computation has been controlled by $m_{\max} = 190$ and not the truncation tolerance τ_g (set artificially small). As for Example 1, we conclude that dH_k over-estimates the exact relative error e_k while ϕ_k under-estimates. Thus H_6 is probably an approximation to an accuracy of $O(10^{-15})$ to $O(10^{-10})$, bounded by the maximum of $\widetilde{r}_6$ and ϕ_6, as indicated by $\widehat{r}_6$ ($O(10^{-15})$ to $O(10^{-10})$). The linear progression of t_6 with respect to n indicates an $O(n)$ computational complexity.

Example 4. We considered the 2D unsteady Navier–Stokes problem [38–40]. We applied the finite element method for flow problems [43,44], which led to the Riccati equation in X:

$$I + (A + \omega I)^{\top} X + X(A + \omega I) - XBB^{\top} X = 0,$$

with $\omega > 0$, A being the discrete Navier–Stokes operator and B the input matrix derived from the gradient operator. For different numbers of finite elements, we tested the cases $n = 1922$, 3362, 7442, 10082 and γ was 0.35, 0.21, 0.136 and 0.118 respectively. Also, $m = 7$, $\omega = 10.1$, $\tau_g = 10^{-30}$ and $m_{\max} = 800$. In this example, SDA_h was terminated when $\widetilde{r}_k \le \epsilon = 9.0 \times 10^{-15}$ or $m_{\max} \ge 800$. The corresponding results are plotted in Fig. 3, similar to Fig. 2 for Example 3 and showing similar behaviour. The residuals show a gentle decline at first and then decrease sharply. Also, δt_k and t_k rose exponentially as predicted. In our experiments, the SDA_h attains the accuracy of $O(10^{-15})$ (in terms of relative residual) for $n = 1357$, 3362. It has been terminated after 8 iterations (with $m_{\max} = 800$ reached) for $n = 7742$, 10,082, respectively with relative residual $O(10^{-12})$ and $O(10^{-11})$. We have not iterated further because 4899 s for $n = 7742$ and 7542 s for $n = 10{,}082$ have already been consumed. The expensive quantities dH_7 ($O(10^{-8})$ to $O(10^{-6})$), $\widehat{r}_8$ ($O(10^{-14})$ to $O(10^{-9})$) and ϕ_8 (less than 10^{-19}) are presented in Fig. 3, illustrating the efficiency of the SDA_h. Recall that the actual relative error of the approximate solution is likely to be bounded between dH_7 and ϕ_8.

6. Conclusions

Large-scale AREs with low-rank constant terms were solved efficiently by the SDA in [36,37] via the numerically low-rank of the solution X. When the constant terms of AREs are high-rank in some applications [38–40], calculation of the residual and relative error at each step of the SDA becomes difficult because of the high rank of X. With careful estimation of relative residuals and errors, we have proposed the SDA_h for large-scale AREs with high-rank constant terms. Efficient control of convergence and economical output of the feedback gain make the SDA_h feasible for CAREs and DAREs. Numerical experiments have validated the effectiveness of the SDA_h.

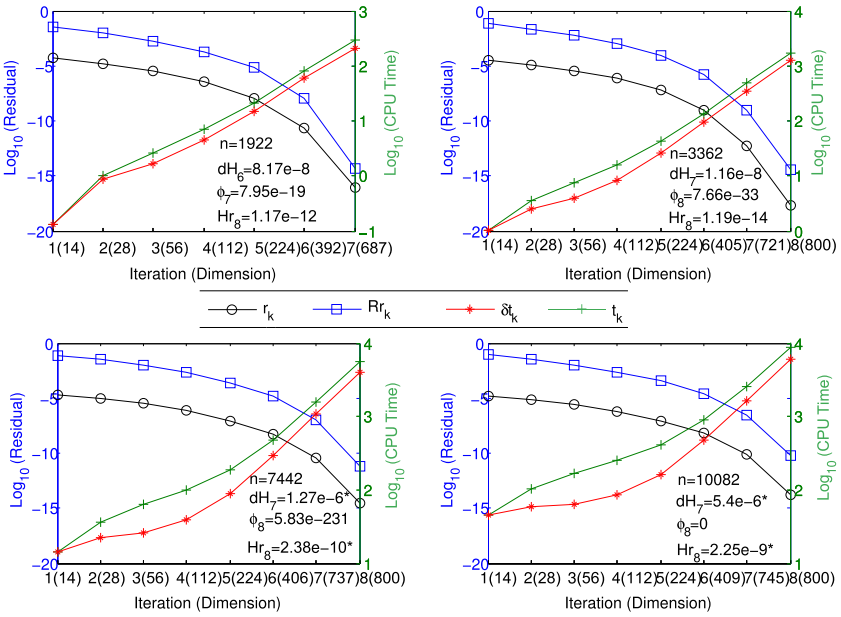

Fig. 3. Residual and CPU time (in $\log_{10}$) for SDA_h in Example 4.

For future work, we may investigate the possibility of solving large-scale differential and difference equations in X with high-rank constant terms. We propose to output the feedback gains $F_d, F_c \in \mathbb{R}^{m\times n}$, which are much smaller than X when $m \ll n$. We do not know how to tackle the case when $m \approx n$ when n is large. It may well be too greedy, wanting to control a large system with many controls and outputs optimally. After all, we need some simplifying structures to build an efficient algorithm.

Acknowledgements

Part of the work occurred when the first author visited Monash University and the third author visited the School of Mathematical Sciences at Fudan University. The first author was supported partly by the NSF of China (grant 11301170, 11801163), Natural Science Foundation of Hunan Province (2017JJ2071) and the Excellent Youth Foundation and General Foundation of Hunan Educational Department (17B071, 17C0466). The second author was supported partly by the Ministry of Science and Technology in the Taiwan, ROC (Grant No: MOST 107-2115-M-003-003).

References

[1] M. Athan, P.L. Falb, Optimal Control: An Introduction to The Theory and Its Applications, McGraw-Hill, New York, 1965.
[2] E.K.-W. Chu, H.-Y. Fan, W.-W. Lin, A structure-preserving doubling algorithm for continuous-time algebraic Riccati equations, Linear Algebra Appl. 396 (2005) 55–80.
[3] P. Lancaster, L. Rodman, Algebraic Riccati Equations, Clarendon Press, Oxford, 1995.
[4] V.L. Mehrmann, The Autonomous Linear Quadratic Control Problem, in: Lecture Notes in Control and Information Sciences, vol. 163, Springer Verlag, Berlin, 1991.
[5] C. Xu, Y. Zheng, H. Su, H. Zeng, Containment for linear multi-agent systems with exogenous disturbances, Neurocomput. 160 (2015) 206–212.
[6] F. Ding, X. Liu, X. Ma, Kalman state filtering based least squares iterative parameter estimation for observer canonical state space systems using decomposition, J. Comput. Appl. Math. 301 (2016) 135–143.
[7] L. Xu, Application of the Newton iteration algorithm to the parameter estimation for dynamical systems, J. Comput. Appl. Math. 288 (2015) 33–43.
[8] E.K.-W. Chu, H.-Y. Fan, W.-W. Lin, C.-S. Wang, A structure-preserving doubling algorithm for periodic discrete-time algebraic Riccati equations, Internat. J. Control 77 (8) (2004) 767–788.
[9] G.H. Golub, C.F. Van Loan, Matrix Computations, third ed., Johns Hopkins University Press, Baltimore, 1996.
[10] A.J. Laub, A schur method for solving algebraic riccati equation, IEEE Trans. Automat. Control AC-24 (1979) 913–921.
[11] Mathworks, MATLAB User's Guide, 2010.
[12] D. Kleinman, On an iterative technique for Riccati equation computations, IEEE Trans. Automat. Control 13 (1968) 114–115.
[13] P. Benner, Solving large-scale control problems, IEEE Control Syst. Mag. 24 (1) (2004) 44–59.
[14] P. Benner, Editorial of special issue on large-scale matrix equations of special type, Numer. Linear Algebra Appl. 15 (9) (2008) 747–754.
[15] P. Benner, H. Fassbender, The symplectic eigenvalue problem, the butterfly form, the SR algorithm, and the Lanczos method, Linear Algebra Appl. 275–276 (1998) 19–47.
[16] P. Benner, H. Fassbender, On the numerical solution of large-scale sparse discrete-time Riccati equations, Adv. Comput. Math. 35 (2011) 119–147.
[17] A. Antoulas, Approximation of Large-Scale Dynamical Systems, SIAM Publications, Philadelphia, 2005.

[18] P. Benner, J. Saak, A semi-discretized heat transfer model for optimal cooling of steel profiles, in: P. Benner, V. Mehrmann, D.C. Sorensen (Eds.), Dimension Reduction of Large-Scale Systems, in: Lecture Notes in Computational Science and Engineering, vol. 45, Springer-Verlag, Berlin, 2005, pp. 353–356.
[19] P. Benner, J. Saak, A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations, in: DFG Priority Programme, in: Optimization with Partial Differential Equations, vol. 1253, 2010, Preprint SPP1253–090, available at http://www.am.uni-erlangen.de/home/spp1253/wiki/images/2/28/Preprint-SPP1253-090pdf.
[20] P. Benner, J. Saak, A Newton-Galerkin-ADI Method for Large-Scale Algebraic Riccati Equations, Applied Linear Algebra 2010, GAMM Workshop Applied and Numerical Linear Algebra, Novi Sad, May 27, 2010; available at http://www.ala2010.pmf.uns.ac.rs/presentations/4g1220pb.pdf.
[21] M. Heyouni, K. Jbilou, An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equations, Electron. Trans. Numer. Anal. 33 (2009) 53–62.
[22] K. Jbilou, Block Krylov subspace methods for large continuous-time algebraic Riccati equations, Numer. Algorithms 34 (2–4) (2003) 339–353.
[23] K. Jbilou, An Arnoldi based algorithm for large algebraic Riccati equations, Appl. Math. Lett. 19 (5) (2006) 437–444.
[24] J. Saak, H. Mena, P. Benner, Matrix Equation Sparse Solvers (MESS): A Matlab Toolbox for the Solution of Sparse Large-Scale Matrix Equations, Chemnitz University of Technology, 2010.
[25] P. Benner, J.-R. Li, T. Penzl, Numerical solution of large Lyapunov equations, Riccati equations and linear-quadratic control problems, Numer. Linear Algebra Appl. 15 (9) (2008) 755–777.
[26] L. Amodei, J.-M. Buchot, An invariant subspace method for large-scale algebraic Riccati equation, Appl. Numer. Math. 60 (11) (2010) 1067–1082.
[27] F. Ding, T.-W. Chen, Gradient based iterative algorithms for solving a class of matrix equations, IEEE Trans. Automat. Control 50 (8) (2005) 1216–1221.
[28] F. Ding, T.-W. Chen, Iterative least squares solutions of coupled Sylvester matrix equations, Systems Control Lett. 54 (2) (2005) 95–107.
[29] F. Ding, T.-W. Chen, On iterative solutions of general coupled matrix equations, SIAM J. Control Optim. 44 (6) (2006) 2269–2284.
[30] F. Ding, P.X. Liu, J. Ding, Iterative solutions of the generalized Sylvester matrix equations by using the hierarchical identification principle, Appl. Math. Comput. 197 (1) (2008) 41–50.
[31] J. Ding, Y. Liu, F. Ding, Iterative solutions to matrix equations of form $A_iXB_i = F_i$, Comput. Math. Appl. 59 (11) (2010) 3500–3507.
[32] F. Ding, H.-M. Zhang, Gradient-based iterative algorithm for a class of the coupled matrix equations related to control systems, IET Control Theory Appl. 8 (15) (2014) 1588–1595.
[33] L. Xie, J. Ding, F. Ding, Gradient based iterative solutions for general linear matrix equations, Comput. Math. Appl. 58 (7) (2009) 1441–1448.
[34] L. Xie, Y.-J. Liu, H.-Z. Yang, Gradient based and least squares based iterative algorithms for matrix equations $AXB + CX^TD = F$, Appl. Math. Comput. 217 (5) (2010) 2191–2199.
[35] H. Zhang, F. Ding, Iterative algorithms for $X + A^\top X^{-1}A = I$ by using the hierarchical identification principle, J. Franklin Inst. 353 (5) (2016) 1132–1146.
[36] E.K.-W. Chu, P.C.-Y. Weng, Large-scale discrete-time algebraic Riccati equations — doubling algorithm and error analysis, J. Comput. Appl. Math. 277 (2015) 115–126.
[37] T. Li, E.K.-W. Chu, W.-W. Lin, P.C.-Y. Weng, Solving large-scale continuous-time algebraic Riccati equations by doubling, J. Comput. Appl. Math. 237 (2013) 373–383.
[38] E. Bänsch, P. Benner, Stabilization of incompressible flow problems by Riccati-based feedback, in: G. Leugering, et al. (Eds.), Constrained Optimization and Optimal Control for Partial Differential Equations, in: International Series of Numerical Mathematics, vol. 160, Birkhäuser, Basel, 2011, pp. 5–20, http://dx.doi.org/10.1007/978-3-0348-0133-1.
[39] P. Benner, ADI-Based methods for algebraic Lyapunov and Riccati equations, CICADA/MIMS Workshop on Numerics for Control and Simulation, Manchester, 2009.
[40] P. Benner, A. Schneider, Balanced truncation for descriptor systems with many terminals, Technical Report, Max Planck Institute Magdeburg, 2013, MPIMD/13-17, available from http://www.mpi-magdeburg.mpg.de/preprints.
[41] W.-W. Lin, S.-F. Xu, Convergence analysis of structure-preserving doubling algorithms for Riccati-type matrix equations, SIAM J. Matrix Anal. Appl. 28 (1) (2006) 26–39.
[42] P.C.-Y. Weng, H.-Y. Fan, E.K.-W. Chu, Low-rank approximation to the solution of a nonsymmetric algebraic Riccati equation from transport theory, Appl. Math. Comput. 219 (2) (2012) 729–740.
[43] J. Donea, A. Huerta, Finite Element Methods for Flow Problems, John Wiley & Sons, 2003.
[44] Mathworks, 2019, https://www.mathworks.com/matlabcentral/fileexchange/60869-2d-unsteady-navier-stokes?s_tid=prof_contriblnk.