# Solving Large-Scale Continuous-Time Algebraic Riccati Equations by Doubling

Tiexiang Li* Eric King-wah Chu† Wen-Wei Lin‡
Peter Chang-Yi Weng§

**Abstract**

We consider the solution of large-scale algebraic Riccati equations with numerically low-ranked solutions. For the discrete-time case, the structure-preserving doubling algorithm has been adapted, with the iterates for $A$ not explicitly computed but in the recursive form $A_k = A_{k-1}^2 - D_k^{(1)} S_k^{-1} [D_k^{(2)}]^\top$, with $D_k^{(1)}$ and $D_k^{(2)}$ being low-ranked and $S_k^{-1}$ being small in dimension. For the continuous-time case, the algebraic Riccati equation will be first treated with the Cayley transform before doubling is applied. With $n$ being the dimension of the algebraic equations, the resulting algorithms are of an efficient $O(n)$ computational complexity per iteration, without the need for any inner iterations, and essentially converge quadratically. Some numerical results will be presented. For instance in Section 5.2, Example 3, of dimension $n = 20209$ with 204 million variables in the solution $X$, was solved using MATLAB on a MacBook Pro within 45 seconds to machine accuracy of $O(10^{-16})$.

**Keywords.** continuous-time algebraic Riccati equation, doubling algorithm, Krylov subspace, large-scale problem

**AMS subject classifications.** 15A24, 65F50, 93C05

## 1 Large-Scale Algebraic Riccati Equations

Let the system matrix $A$ be large and sparse, possibly with band structures. The discrete-time algebraic Riccati equation (DARE):

$$\mathcal{D}(X) \equiv -X + A^\top X(I + GX)^{-1}A + H = 0, \tag{1a}$$

and the continuous-time algebraic Riccati equation (CARE):

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

*Department of Mathematics, Southeast University, Nanjing, 211189, People's Republic of China; txli@seu.edu.cn

†School of Mathematical Sciences, Building 28, Monash University, VIC 3800, Australia; Phone: +61-3-99054480, FAX: +61-3-99054403; eric.chu@monash.edu

‡Department of Mathematics, National Taiwan University, Taipei 106, Taiwan; wwlin@math.ntu.edu.tw

§School of Mathematical Sciences, Building 28, Monash University, VIC 3800, Australia; peter.weng@monash.edu

with the low-ranked

$$G = BR^{-1}B^\top, \quad H = CT^{-1}C^\top, \tag{1c}$$

where $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{n\times l}$ and $m, l \ll n$, arise often in linear-quadratic optimal control problems [26, 34].

The solution of CAREs and DAREs has been an extremely active area of research; see, e.g., [16, 26, 34]. The usual solution methods such as the Schur vector method, symplectic SR methods, the matrix sign function, the matrix disk function or the doubling method have not made (full) use of the sparsity and structure in $A$, $G$ and $H$. Requiring in general $O(n^3)$ flops and workspace of size $O(n^2)$, these methods are obviously inappropriate for the large-scale problems we are interested in here.

For control problems for parabolic PDEs and the balancing based model order reduction of large linear systems, large-scale CAREs and DAREs have to be solved [2, 4, 12, 7, 27, 28]. As stated in [8, 10], "the basic observation on which all methods for solving such kinds of matrix equations are based, is that often the (numerical) rank of the solution is very small compared to its actual dimension and therefore it allows for a good approximation via low rank solution factors". Importantly, without solving the corresponding algebraic Riccati equations, alternative solutions to the optimal control problem require the deflating subspace of the corresponding Hamiltonian matrices or (generalized) symplectic pencils which are prohibitively expensive to compute.

Benner, Fassbender and Saak have done much on large-scale algebraic Riccati equations; see [8, 10, 36, 37] and the references therein. They built their methods on (inexact) Newton's methods with inner iterations for the associated Lyapunov and Stein equations. We shall adapt the structure-preserving doubling algorithm (SDA) [13, 14, 32], making use of the sparsity in $A$ and the low-ranked structures in $G$ and $H$. For other applications of the SDA, see [15].

## 2 Structure-Preserving Doubling Algorithm for DAREs

We shall abbreviate the discussion for DAREs; please consult [30] for details.

The structure-preserving doubling algorithm (SDA) [14], assuming $(I+GH)^{-1}$ exists, has the following form:

$$\left\{\begin{array}{rcl} G & \leftarrow & G + A(I+GH)^{-1}GA^\top, \\ H & \leftarrow & H + A^\top H(I+GH)^{-1}A, \\ A & \leftarrow & A(I+GH)^{-1}A. \end{array}\right. \tag{2}$$

We shall apply the Sherman-Morrison-Woodbury formula (SMWF) to $(I+GH)^{-1}$ and make use of the low-ranked forms of $G$ and $H$ in (1c).

### 2.1 Large-Scale SDA

From the first glance, the iteration for $A$ in the SDA in (2) appears doomed, with $O(n^3)$ operations for the products of full matrices. However, with the low rank form in (1c), we shall prove (for $k = 1, 2, \cdots$)

$$\begin{array}{rcl} A_k & = & A_{k-1}^2 - D_k^{(1)} S_k^{-1} \left[D_k^{(2)}\right]^\top, \\ G_k & = & B_k R_k^{-1} B_k^\top, \\ H_k & = & C_k T_k^{-1} C_k^\top. \end{array}$$

The application of the SMWF on $(I_n + G_k H_k)^{-1}$ yields

$$A_{k+1} = A_k(I_n + G_k H_k)^{-1} A_k$$

$$\begin{aligned} &= A_k \left[ I_n - G_k C_k T_k^{-1} \left( I_l + C_k^\top G_k C_k T_k^{-1} \right)^{-1} C_k^\top \right] A_k \\ &= A_k \left[ I_n - B_k \left( I_m + R_k^{-1} B_k^\top H_k B_k \right)^{-1} R_k^{-1} B_k^\top H_k \right] A_k. \end{aligned}$$

It will be obvious that it is more convenient to work with $S_k^{-1}$, $R_k^{-1}$ and $T_k^{-1}$, and we retain the inverse notation only for historical reasons, although there is no actual inversion involved. Consequently, we have

$$A_{k+1} = A_k^2 - D_{k+1}^{(1)} S_{k+1}^{-1} \left[ D_{k+1}^{(2)} \right]^\top, \tag{3}$$

with the rank $l$ update defined by

$$D_{k+1}^{(1)} = A_k G_k C_k, \quad D_{k+1}^{(2)} = A_k^\top C_k, \quad S_{k+1}^{-1} = T_k^{-1} \left( I_l + C_k^\top G_k C_k T_k^{-1} \right)^{-1}, \tag{4a}$$

or the rank $m$ update

$$D_{k+1}^{(1)} = A_k B_k, \quad D_{k+1}^{(2)} = A_k^\top H_k B_k, \quad S_{k+1}^{-1} = \left( I_m + R_k^{-1} B_k^\top H_k B_k \right)^{-1} R_k^{-1}, \tag{4b}$$

all involving $O(n^3)$ operations for a dense $A$. The operation counts will be reduced to $O(n)$ with the assumption that the maximum number of nonzero components in any row or column of $A$ is much less than $n$ (see Table 2 in Section 4.2). The trick is *not* to form $A_k$ explicitly. Note that we have to store all the $B_i$, $C_i$, $R_i^{-1}$ and $T_i^{-1}$ for $i = 0, 1, \cdots, k-1$ to facilitate the multiplication of low-ranked matrices by $A_k$ or $A_k^\top$.

We may choose between (4a) and (4b) based on the sizes $l$ and $m$. Ignoring the small saving in the inversion of smaller matrices, the compression and truncation in the next section produces the leaner $B_k$ and $C_k$, which makes the choice irrelevant. However, this choice may be important when $G$ or $H$ are not low-ranked.

For $k = 0$, we have

$$A_0 = A, \quad D_0^{(1)}, \ D_0^{(2)} = 0, \quad S_0 = I. \tag{5}$$

The induction proof of the general form of $A_k$ in (3)–(4b) can be completed by considering the initial $k = 1$ case, which is trivial.

For $B_k$, $C_k$ and $R_k$, applying the SMWF to $(I + G_k H_k)^{-1}$ in the SDA, we have

$$\begin{aligned} G_{k+1} &= G_k + A_k G_k A_k^\top - A_k G_k C_k T_k^{-1} \left( I_l + C_k^\top G_k C_k T_k^{-1} \right)^{-1} C_k^\top G_k A_k^\top \\ &= G_k + A_k G_k A_k^\top - A_k B_k \left( I_m + R_k^{-1} B_k^\top H_k B_k \right)^{-1} R_k^{-1} B_k^\top H_k G_k A_k^\top, \end{aligned}$$

and

$$\begin{aligned} H_{k+1} &= H_k + A_k^\top H_k A_k - A_k^\top H_k G_k C_k T_k^{-1} \left( I_l + C_k^\top G_k C_k T_k^{-1} \right)^{-1} C_k^\top A_k \\ &= H_k + A_k^\top H_k A_k - A_k^\top H_k B_k (I_m + R_k^{-1} B_k^\top H_k B_k)^{-1} R_k^{-1} B_k^\top H_k A_k. \end{aligned}$$

These imply that

$$B_{k+1} = [B_k, \ A_k B_k], \quad C_{k+1} = [C_k, \ A_k^\top C_k], \tag{6}$$

$$R_{k+1}^{-1} = R_k^{-1} \oplus \left[R_k^{-1} - R_k^{-1}B_k^\top C_k T_k^{-1}\left(I_l + C_k^\top G_k C_k T_k^{-1}\right)^{-1} C_k^\top B_k R_k^{-1}\right] \quad (7a)$$

$$= R_k^{-1} \oplus \left[R_k^{-1} - \left(I_m + R_k^{-1}B_k^\top H_k B_k\right)^{-1} R_k^{-1}B_k^\top H_k B_k R_k^{-1}\right], \quad (7b)$$

$$T_{k+1}^{-1} = T_k^{-1} \oplus \left[T_k^{-1} - T_k^{-1}C_k^\top G_k C_k T_k^{-1}\left(I_l + C_k^\top G_k C_k T_k^{-1}\right)^{-1}\right], \quad (8a)$$

$$= T_k^{-1} \oplus \left[T_k^{-1} - T_k^{-1}C_k^\top B_k\left(I_m + R_k^{-1}B_k^\top H_k B_k\right)^{-1} R_k^{-1}B_k^\top C_k T_k^{-1}\right] \quad (8b)$$

with

$$B_0 = B, \quad C_0 = C, \quad R_0 = R, \quad T_0 = T. \quad (9)$$

The existence of $R_k^{-1}$, $T_k^{-1}$ and $(I_n + G_k H_k)^{-1}$ guarantees the same for other inverses in (7a)–(8b). Note that $R_k^{-1}$, $S_k^{-1}$ and $T_k^{-1}$ are symmetric for all $k$. Again, the choice in (7a)–(8b) may be relevant when $G$ or $H$ are not low-ranked.

For well-behaved DAREs, we have $H_k = C_k T_k^{-1} C_k^\top \to X$ and $G_k = B_k R_k^{-1} B_k^\top \to Y$ (solution of the adjoint DARE) as $k \to \infty$.

Note that the ranks of $X$ and $Y$ have been observed to be numerically low-ranked. Under suitable assumptions [13, 14], the convergence of the SDA implies the convergence of $A_k = O(|\lambda|^{2^k}) \to 0$, for some $|\lambda| < 1$. Together with (6)–(8b), we see that $B_{k+1}$ and $C_{k+1}$ equal, respectively, the sums of $B_k$ and $C_k$ and the diminishing components $A_k B_k$ and $A_k^\top C_k$. Thus the observation about the low numerical ranks of $X$ and $Y$ has been shown to be true.

## 2.2 Compression and Truncation of $B_k$ and $C_k$

Now we shall consider an important aspect of the SDA for large-scale DAREs (SDA_ls) — the growth of $B_k$ and $C_k$. Obviously, as the SDA converges, increasingly smaller but fatter low-ranked components are added to $B_k$ and $C_k$. Apparent from (6), the growth in the sizes and ranks of these iterates is potentially exponential. Let the computational complexity of the SDA_ls be $O(n) = \alpha n + O(1)$. If the convergence is slow relative to the growth in $B_k$ and $C_k$, the algorithm will fail, with $\alpha$ growing exponentially (see Table 2 in Section 4.2).

To reduce the dimensions of $B_k$, $C_k$, $D_k^{(1)}$ and $D_k^{(2)}$, we shall compress their columns by orthogonaization. Consider the QR decompositions with column pivoting:

$$B_k = Q_{1k}M_{1k} + \widetilde{Q}_{1k}\widetilde{M}_{1k},$$
$$C_k = Q_{2k}M_{2k} + \widetilde{Q}_{2k}\widetilde{M}_{2k}$$

with

$$\|\widetilde{M}_{1k}\| \le \tau_1, \quad \|\widetilde{M}_{2k}\| \le \tau_2$$

where $\tau_i$ $(i = 1, 2)$ are some small tolerances controlling the compression and truncation process, $n_{1k}$ and $n_{2k}$ are respectively the numbers of columns in $B_k$ and $C_k$ bounded from above by some corresponding $m_{\max}$ and $l_{\max}$,

$$r_{1k} = \text{rank}B_k \le n_{1k} \le m_{\max} \ll n,$$
$$r_{2k} = \text{rank}C_k \le n_{2k} \le l_{\max} \ll n,$$

and for $i = 1, 2$, $Q_{ik} \in \mathbb{R}^{n \times r_{ik}}$ are unitary and $M_{ik} \in \mathbb{R}^{r_{ik} \times n_{ik}}$ are full-ranked and upper triangular. We then have

$$B_k R_k^{-1} B_k^\top = Q_{1k} \left( M_{1k} R_k^{-1} M_{1k}^\top \right) Q_{1k}^\top + O(\tau_1), \tag{10}$$

$$C_k T_k^{-1} C_k^\top = Q_{2k} \left( M_{2k} T_k^{-1} M_{2k}^\top \right) Q_{2k}^\top + O(\tau_2), \tag{11}$$

and we should replace $B_k$ and $R_k^{-1}$ (or, $C_k$ and $T_k^{-1}$) respectively by the leaner $Q_{1k}$ and $M_{1k} R_k^{-1} M_{1k}^\top$ (or, $Q_{2k}$ and $M_{2k} T_k^{-1} M_{2k}^\top$). We may ignore compressing and truncating $D_k^{(1)}$ and $D_k^{(2)}$ after compressing and truncating $B_k$ and $C_k$. As a result, we ignore the $O(\tau_i)$ terms, control the growth of $r_{ik}$ while sacrificing a hopefully negligible bit of accuracy.

Interestingly, we need only $R$, $T$ and $I + G_k H_k$ to be invertible (which imply the invertibility of $R_k$ and $T_k$ for all $k$), opening up the possibility of dealing with DAREs with indefinite $R$s and $T$s [35].

Equations (3) (used recursively but *not* explicitly), (4a) (or (4b)), (6), (7a) (or (7b)), (8a) (or (8b)), (10) and (11), together with the corresponding initial conditions (5) and (9), constitute the SDA_ls.

## 2.3 SDA and Krylov Subspaces

There is an interesting relationship between the SDA_ls and Krylov subspaces. Define 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 (3) and (6), we can see that

$$B_0 = B \in \mathcal{K}_0(A, B), \quad B_1 = [B, AB] \in \mathcal{K}_1(A, B)$$

and, for some low-ranked $F$,

$$B_2 = \left[B_1, A_1 B_1\right] = \left[B, AB; (A^2 - ABF^\top)(B, AB)\right] \in \mathcal{K}_2(A, B).$$

(We have abused notations, with $V \in \mathcal{K}_k(A, B)$ meaning $\text{span}\{V\} \in \mathcal{K}_k(A, B)$.) Similarly, it is easy to show that

$$B_k \in \mathcal{K}_k(A, B), \quad C_k \in \mathcal{K}_k(A^\top, C).$$

In other words, the general SDA is closely related to approximating the solutions $X$ and $Y$ using Krylov subspaces, with additional components vanishing quadratically. However, for problems of small size $n$, $B_k$ and $C_k$ become full-ranked after a few iterations.

## 2.4 Error of SDA_ls

The SDA_ls can be interpreted as a Galerkin method, or directly from (2). With

$$\delta_k \equiv \max\{\|\delta G_k\|, \|\delta H_k\|, \|\delta A_k\|\}, \tag{12}$$

we can show

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

with $c_k \to 0$ as $k \to \infty$. A more detailed discussion can be found in [30, Section 2.5]. Essentially, we limit the rank of the approximation to $X$, trading off the accuracy in $X$ with the efficiency of the SDA_ls. Assume that the compression and truncation in (10) and (11) create errors of $O(\tau_i)$ $(i = 1, 2)$ in $G_k$ and $H_k$, respectively. It is easy to see from (12) that errors of the same magnitude will propagate through to $A_{k+1}$, $G_{k+1}$ and $H_{k+1}$. The fact that $A_k \to 0$ implies $c_k \to 0$ and contributes towards diminishing these errors. From our numerical experience, the trade-off between the ranks of $G_k$ and $H_k$ and the accuracy of the approximate solutions to $X$ and $Y$ is the key to the success of our computation. If these ranks grow out of control, unnecessary and insignificant small additions to the iterates overwhelm the computation in terms of flop counts and memory requirement. Limiting the ranks will obviously reduce the accuracy of the approximate solution. We found we have to experiment with the tolerances for the compression/truncation and convergence while trying to achieve a balance between accuracy and the feasibility/efficiency of the SDA.

# 3 CAREs

One possible approach for large-scale CAREs is to transform them to DAREs using Cayley transforms.

## 3.1 SDA after Cayley Transform

From [13], the matrices $A$, $G$ and $H$ in the CARE (1b) are first treated with the Cayley transform:

$$A_0 = I + 2\gamma \left(A_\gamma + G A_\gamma^{-\top} H\right)^{-1}, \tag{13}$$

$$G_0 = 2\gamma A_\gamma^{-1} G \left(A_\gamma^{\top} + H A_\gamma^{-1} G\right)^{-1}, \tag{14}$$

$$H_0 = 2\gamma \left(A_\gamma^{\top} + H A_\gamma^{-1} G\right)^{-1} H A_\gamma^{-1}, \tag{15}$$

with $A_\gamma \equiv A - \gamma I$ and a suitable $\gamma > 0$ chosen to optimize the condition of various matrix inversions. A simple application of the SMWF implies

$$\begin{aligned} & (A_\gamma + G A_\gamma^{-\top} H)^{-1} \\ = \; & A_\gamma^{-1} - A_\gamma^{-1} G A_\gamma^{-\top} C \cdot T^{-1} (I_l + C^\top A_\gamma^{-1} G A_\gamma^{-\top} C T^{-1})^{-1} \cdot C^\top A_\gamma^{-1} \qquad (16a) \\ = \; & A_\gamma^{-1} - A_\gamma^{-1} B \cdot \left(I_m + R^{-1} B^\top A_\gamma^{-\top} H A_\gamma^{-1} B\right)^{-1} R^{-1} \cdot B^\top A_\gamma^{-\top} H A_\gamma^{-1}. \qquad (16b) \end{aligned}$$

It is not hard to see, with the above initial $A_0$, $G_0$ and $H_0$, that the SDA_ls still works, again with exactly the same forms and updating formulae for $A_k$, $B_k$, $C_k$, $D_k^{(1)}$, $D_k^{(2)}$ and the inverses of $R_k$, $S_k$ and $T_k$. One relevant difference for CAREs is that $A_0 \neq A$ but satisfies, from (13), (16a) and (16b),

$$A_0 = \left(I_n + 2\gamma A_\gamma^{-1}\right) - D_0^{(1)} S_0^{-1} \left[D_0^{(2)}\right]^\top \tag{17}$$

with

$$B_0 = A_\gamma^{-1} B, \quad C_0 = A_\gamma^{-\top} C. \tag{18}$$

The corresponding rank $l$ and $m$ perturbed updates have the forms, respectively,

$$D_0^{(2)} = C_0, \quad D_0^{(1)} = A_\gamma^{-1} G C_0, \quad S_0^{-1} = 2\gamma \left(I_l + T^{-1} C_0^\top G C_0\right)^{-1} T^{-1}; \tag{19a}$$

$$D_0^{(1)} = B_0, \quad D_0^{(2)} = A_\gamma^{-\top} H B_0, \quad S_0^{-1} = 2\gamma \left(I_m + R^{-1} B_0^\top H B_0\right)^{-1} R^{-1}. \tag{19b}$$

Note that all computations can be realized in $O(n)$ operations, assuming that the operations $A_\gamma^{-1}B$ and $A_\gamma^{-\top}C$ are achievable in $O(n)$ flops; see [20, §9.1] for a banded $A$.

Similarly, we have

$$R_0^{-1} = 2\gamma \left[R^{-1} - R^{-1} B^\top C_0 \cdot \left(I_l + T^{-1} C_0^\top G C_0\right)^{-1} T^{-1} \cdot C_0^\top B R^{-1}\right] \tag{20a}$$

$$= 2\gamma \left[R^{-1} - R^{-1} B_0^\top H B_0 \left(I_m + R^{-1} B_0^\top H B_0\right)^{-1} R^{-1}\right], \tag{20b}$$

and

$$T_0^{-1} = 2\gamma \left[T^{-1} - T^{-1} \left(I_l + C_0^\top G C_0 T^{-1}\right)^{-1} C_0^\top G C_0 T^{-1}\right] \tag{21a}$$

$$= 2\gamma \left[T^{-1} - T^{-1} C^\top B_0 \cdot R^{-1} \left(I_m + B_0^\top H B_0 R^{-1}\right)^{-1} \cdot B_0^\top C T^{-1}\right]. \tag{21b}$$

For CAREs, we have

$$B_k \in \mathcal{K}_k(A_\gamma^{-1},\, A_\gamma^{-1} B), \quad C_k \in \mathcal{K}_k(A_\gamma^{-\top},\, A_\gamma^{-\top} C). \tag{22}$$

Note that the Krylov subspaces $\mathcal{K}_k(A^{\pm 1}, B)$ and $\mathcal{K}_k(A^{\pm\top}, C)$ have been used in the solution of CAREs and Lyapunov equations in [19], [21]–[25]. From (22) and [30, 31], we can see clearly the appropriate choices of Krylov subspaces for DAREs and CAREs, as well as the corresponding Stein and Lyapunov equations. A summary is contained in Table 1:

| Equation | $X$ | $Y$ |
|---|---|---|
| DARE, Stein equation | $\mathcal{K}_k(A^\top, C)$ | $\mathcal{K}_k(A, B)$ |
| CARE, Lyapunov equation | $\mathcal{K}_k(A_\gamma^{-\top},\, A_\gamma^{-\top} C)$ | $\mathcal{K}_k(A_\gamma^{-1},\, A_\gamma^{-1} B)$ |

Table 1: Krylov subspaces for solution $X$ and adjoint solution $Y$

We summarize the algorithm below, with the particular choice of (3), (4a), (6), (7a), (8b), (10) and (11). We would like to emphasize that care has to be exercised in Algorithm 1 below, with the multiplications by $A_{k+1}$ and $A_{k+1}^\top$ carried out recursively using (3) and (4a) or (4b). Otherwise, computations cannot be carried out in $O(n)$ complexity. Similar care has to be taken in the computation of residuals (used in Algorithm 1 below) or differences of iterates (as an alternative convergence control), as discussed in Section 4.2 later.

**Algorithm 1 (SDA_ls)**

Input: $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $R^{-1} = R^{-\top} \in \mathbb{R}^{m\times m}$, $C \in \mathbb{R}^{n\times l}$, $T^{-1} = T^{-\top} \in \mathbb{R}^{l\times l}$ shift $\gamma > 0$, positive tolerances $\tau_1$, $\tau_2$ and $\epsilon$, and $m_{\max}$, $l_{\max}$;

Output: $B_\epsilon \in \mathbb{R}^{n\times m_\epsilon}$, $R_\epsilon^{-1} = R_\epsilon^{-\top} \in \mathbb{R}^{m_\epsilon\times m_\epsilon}$, $C_\epsilon \in \mathbb{R}^{n\times l_\epsilon}$ and $T_\epsilon^{-1} = T_\epsilon^{-\top} \in \mathbb{R}^{l_\epsilon\times l_\epsilon}$, with $C_\epsilon T_\epsilon^{-1} C_\epsilon^\top$ and $B_\epsilon R_\epsilon^{-1} B_\epsilon^\top$ approximating, respectively, the solutions $X$ and $Y$ to the large-scale CARE (1b) and its adjoint;

Compute $A_\gamma = A - \gamma I$;
Set $k = 0$, $\widetilde{r}_0 = 2\epsilon$; $B_0 = A_\gamma^{-1} B$, $C_0 = A_\gamma^{-\top} C$;
$R_0^{-1} = 2\gamma \left[ R^{-1} - R^{-1} B^\top C_0 \cdot \left( I_l + T^{-1} C_0^\top G C_0 \right)^{-1} T^{-1} \cdot C_0^\top B R^{-1} \right]$,
$T_0^{-1} = 2\gamma \left[ T^{-1} - T^{-1} C^\top B_0 \cdot R^{-1} \left( I_m + B_0^\top H B_0 R^{-1} \right)^{-1} \cdot B_0^\top C T^{-1} \right]$;
$D_0^{(2)} = C_0$, $D_0^{(1)} = A_\gamma^{-1} G C_0$, $S_0^{-1} = 2\gamma \left( I_l + T^{-1} C_0^\top G C_0 \right)^{-1} T^{-1}$,
$A_0 = I_n + 2\gamma A_\gamma^{-1} - D_0^{(1)} S_0^{-1} \left[ D_0^{(2)} \right]^\top$;
Compute $h = \|H_0\| = \|C_0 T_0^{-1} C_0^\top\|$;
Do until convergence:
  If the relative residual $\tilde{r}_k = |d_k/(h_k + \widetilde{m}_k + h)| < \epsilon$,
    Set $B_\epsilon = B_k$, $R_\epsilon^{-1} = R_k^{-1}$, $C_\epsilon = C_k$ and $T_\epsilon^{-1} = T_k^{-1}$;
    Exit
  End If
  Compute $B_{k+1} = [B_k,\ A_k B_k]$, $C_{k+1} = [C_k,\ A_k^\top C_k]$;
    $R_{k+1}^{-1} = R_k^{-1} \oplus \left[ R_k^{-1} - R_k^{-1} B_k^\top C_k T_k^{-1} \left( I_l + C_k^\top G_k C_k T_k^{-1} \right)^{-1} C_k^\top B_k R_k^{-1} \right]$,
    $T_{k+1}^{-1} = T_k^{-1} \oplus \left[ T_k^{-1} - T_k^{-1} C_k^\top B_k \left( I_m + R_k^{-1} B_k^\top H_k B_k \right)^{-1} R_k^{-1} B_k^\top C_k T_k^{-1} \right]$;
    with $A_{k+1} = A_k^2 - D_{k+1}^{(1)} S_{k+1}^{-1} \left[ D_{k+1}^{(2)} \right]^\top$,
    $D_{k+1}^{(1)} = A_k G_k C_k$, $D_{k+1}^{(2)} = A_k^\top C_k$, $S_{k+1}^{-1} = T_k^{-1} \left( I_l + C_k^\top G_k C_k T_k^{-1} \right)^{-1}$;
  Compress $B_{k+1}$ and $C_{k+1}$, using the tolerances $\tau_1$ and $\tau_2$, and modify $R_{k+1}^{-1}$ and $T_{k+1}^{-1}$, as in (10) and (11);
  Compute $k \leftarrow k+1$, $d_k = \|\mathcal{D}(H_k)\|$, $h_k = \|H_k\|$ and $\widetilde{m}_k = \|M_k\|$, as in Section 4.2;
End Do

# 4 Computational Issues

## 4.1 Residuals and Convergence Control

Consider the difference of successive iterates:

$$\delta G_k \equiv B_k R_k^{-1} B_k^\top - B_{k+1} R_{k+1}^{-1} B_{k+1}^\top = \widetilde{B}_{k+1} \widetilde{R}_{k+1}^{-1} \widetilde{B}_{k+1}^\top,$$

we have

$$\widetilde{B}_{k+1} \equiv [B_k,\ B_{k+1}], \quad \widetilde{R}_{k+1}^{-1} \equiv R_k^{-1} \oplus \left( -R_{k+1}^{-1} \right).$$

Similarly, with $\delta H_k \equiv C_k T_k^{-1} C_k^\top - C_{k+1} T_{k+1}^{-1} C_{k+1}^\top$, we have

$$\delta H_k = \widetilde{C}_{k+1} \widetilde{T}_{k+1}^{-1} \widetilde{C}_{k+1}^\top$$

with

$$\widetilde{C}_{k+1} \equiv [C_k,\ C_{k+1}]\,, \quad \widetilde{T}_{k+1}^{-1} \equiv T_k^{-1} \oplus \left(-T_{k+1}^{-1}\right).$$

For the residual $r_k \equiv \|\mathcal{D}(H_k)\|$ of the DARE, the corresponding relative residual equals

$$\widetilde{r}_k \equiv \frac{r_k}{\|H_k\| + \|M_k\| + \|H\|}, \quad M_k \equiv A^\top H_k A - A^\top H_k B(R + B^\top H_k B)^{-1} B^\top H_k A,$$

and with the help of the SMWF, we have

$$\begin{aligned} \mathcal{D}(H_k) &= -H_k + A^\top H_k A - A^\top H_k B(R + B^\top H_k B)^{-1} B^\top H_k A + H \\ &= \widehat{C}_k \widehat{T}_k^{-1} \widehat{C}_k^\top \end{aligned}$$

with

$$\begin{aligned} \widehat{C}_k &\equiv \left[C_k,\ A^\top C_k,\ C\right], \\ \widehat{T}_k^{-1} &\equiv \left(-T_k^{-1}\right) \oplus \breve{T}_k^{-1} \oplus T^{-1}, \\ \breve{T}_k^{-1} &\equiv T_k^{-1} - T_k^{-1} C_k^\top B(R + B^\top H_k B)^{-1} B^\top C_k T_k^{-1}. \end{aligned}$$

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

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

All the calculations in this subsection involve the norms of similar low-rank symmetric matrices. For $H_k$, as in (11), we can orthogonalize $C_k$ and transform $T_k^{-1}$ accordingly, analogous to (10) and (11). With the orthogonal $\widetilde{B}_k$, $\widetilde{C}_k$, $\widehat{C}_k$, $C_k$, $C$ and $A^\top C_k$, and the transformed $\widetilde{R}_k^{-1}$, $\widetilde{T}_k^{-1}$, $\widehat{T}_k^{-1}$, $T_k^{-1}$, $T^{-1}$ and $\breve{T}_k^{-1}$ respectively, we have the efficient formulae

$$\begin{aligned} \|\delta G_k\| &= \|\widetilde{R}_{k+1}^{-1}\|, \quad \|\delta H_k\| = \|\widetilde{T}_{k+1}^{-1}\|, \quad r_k = \|\widehat{T}_k^{-1}\|, \\ \|H_k\| &= \|T_k^{-1}\|, \quad \|H\| = \|T^{-1}\|, \quad \|M_k\| = \|\breve{T}_k^{-1}\| \end{aligned} \tag{23}$$

for the 2- and F-norms.

For CAREs, and persist with the same notations, we have

$$r_k \equiv \|\mathcal{C}(H_k)\|, \quad \widetilde{r}_k \equiv \frac{r_k}{\|A^\top H_k + H_k A\| + \|H_k G H_k\| + \|H\|}.$$

Similarly, we have

$$\begin{aligned} \mathcal{C}(H_k) &= A^\top H_k + H_k A - H_k G H_k + H \\ &= \widehat{C}_k \widehat{T}_k^{-1} \widehat{C}_k^\top \end{aligned}$$

with

$$\begin{aligned} \widehat{C}_k &\equiv \left[A^\top C_k,\ C_k,\ C\right], \quad \breve{T}_k^{-1} \equiv T_k^{-1} C_k^\top G C_k T_k^{-1}, \\ \breve{T}_k^{-1} &\equiv \begin{bmatrix} 0 & T_k^{-1} \\ T_k^{-1} & 0 \end{bmatrix}, \quad \widehat{T}_k^{-1} \equiv \begin{bmatrix} 0 & T_k^{-1} \\ T_k^{-1} & -\breve{T}_k^{-1} \end{bmatrix} \oplus T^{-1}. \end{aligned}$$

After orthogonalizing $\widehat{C}_k$, $C_k$, $C$ and $\widetilde{C}_k$ and transforming respectively $\widehat{T}_k^{-1}$, $T_k^{-1}$, $T^{-1}$, $\breve{T}_k^{-1}$ and $\check{T}_k^{-1}$, we have similar results as in (23):

$$\begin{aligned} r_k &= \|\widehat{T}_k^{-1}\|, \quad \|H_k\| = \|T_k^{-1}\|, \quad \|H\| = \|T^{-1}\|, \\ \|H_k G H_k\| &= \|\breve{T}_k^{-1}\|, \quad \|A^\top H_k + H_k A\| = \|\check{T}_k^{-1}\|. \end{aligned} \tag{24}$$

If we replace the symmetry term $\|A^\top H_k + H_k A\|$ with $2\|A^\top H_k\|$ in the denominator of the relative residual, we then have

$$A^\top H_k = (A^\top C_k) T_k^{-1} C_k^{-1}.$$

We then need to orthogonalize $A^\top C_k$ as well and transform $T_k^{-1}$ from the left and the right, yielding $\|A^\top H_k\| = \|T_k^{-1}\|$.

From our numerical experience, the estimates from right-hand-sides of (23) and (24) are less accurate than those using the direct definitions on the left-hand-sides, probably because of the numerical errors from the orthogonalization involved. Typically, when the quantities are near the machine accuracy of $O(10^{-16})$, the estimates are around ten times the exact values. We feel it is a fair price to pay, for avoiding the $O(n^3)$ complexity. Users may recompute these quantities using the original definitions after convergence of the SDA_ls (if that is possible).

## 4.2 Operation and Memory Counts

We shall assume that $c_\gamma mn$ flops are required in the solution of $A_\gamma Z = R$ or $A_\gamma^\top Z = R$, with $R \in \mathbb{R}^{n\times m}$. A start up cost of $\left[c_\gamma(l+m) + 4l^2 + 1\right] n$ flops for the the SDA_ls is made up of the following:

(1) setting up $A_\gamma = A - \gamma I_n$, requiring $n$ flops;

(2) setting up $B_0 = A_\gamma^{-1} B$ and $C_0 = A_\gamma^{-1} C$, requiring $c_\gamma(l+m)n$ 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.

The operation and memory counts of Algorithm 1 (SDA_ls) for the $k$th iteration are summarized in Table 2 below. In the third column, the number of variables is recorded. Only the dominant $O(n)$ operations or memory requirement are included. Note that most of the work is done in the computation of $B_{k+1}$ and $C_{k+1}$, for which $A_k B_k$ and $A_k^\top C_k$ have to be calculated recursively, as $A_k$ is not available explicitly. In Table 2, we shall use the notation $N_k \equiv \sum_{j=1}^{k}(l_j + m_j)$. The operation count for the QR decomposition of an $n \times r$ matrix is $4nr^2$ flops [18, p. 250].

With $l_k$ and $m_k$ controlled by the compression and truncation in Section 2.2, the operation count will be dominated by the calculation of $B_{k+1}$ and $C_{k+1}$. In our numerical examples in Section 5, 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 $2^{k+1}$ factor in the total flop count. However, the last iteration is virtually free, as there is no need to prepare $B_{k+1}$ and $C_{k+1}$ for the next iteration.

| Computation | Flops | Memory |
|---|---|---|
| $B_{k+1}$, $C_{k+1}$ | $\left[2^{k+1}c_\gamma(l_k+m_k)+(6l_k+1)N_k\right]n$ | $N_{k+1}n$ |
| $R_{k+1}^{-1}$, $T_{k+1}^{-1}$ | $4l_km_kn$ | $O(l_k^2+m_k^2)$ |
| $D_{k+1}^{(1)}$, $D_{k+1}^{(2)}$ | $O(l_k^3+m_k^3)$ | – |
| $S_{k+1}^{-1}$ | $O(l_k^3)$ | $O(l_k^2)$ |
| Compress $B_{k+1}$, $C_{k+1}$ | $4(l_k^2+m_k^2)n$ | – |
| Modify $R_{k+1}^{-1}$, $T_{k+1}^{-1}$ | $O(l_k^3+m_k^3)$ | – |
| $\tilde{r}_{k+1}$ | $\left[4(2l_k+l)^2+8l_k^2+2l_km\right]n$ | – |
| Total | $\left[2^{k+1}c_\gamma(l_k+m_k)+(6l_k+1)N_k+ 4(l_k^2+m_k^2+l_km_k)+4(2l_k+l)^2+8l_k^2+2l_km\right]n$ | $N_{k+1}n$ |

Table 2: Operation and Memory Counts for the $k$th Iteration in Algorithm 1 (SDA_ls)

# 5 Numerical Examples

## 5.1 Test Examples

We have tested the SDA_ls on selected numerical examples from [11]. The suite of challenging problems involves 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. The accuracy of the approximate solutions for the CARE examples is good, with relative residuals of $O(10^{-16})$, as compared to the lesser accuracy achieved in [30, 31] for DAREs and Stein/Lyapunov equations. Scaling the ARE or varying the values of $\gamma$ (the shift in the Cayley transform from CAREs to DAREs), $\tau_i$ $(i = 1, 2)$, $l_{\max}$ or $m_{\max}$ may improve the accuracy of the approximate solution or the speed of convergence. For example, a much worse relative residual of $O(10^{-10})$ was achieved for $\gamma = 10^5$.

We have not attempted to select an optimal $\gamma$ for the Cayley transform of the CAREs, accepting gratefully the good results from $\gamma = 0.5$. From our experience in [13] and from the numerical tests below, we found $\gamma$ is easy and insensitive to choose. Typically, the condition number of $A_\gamma = A - \gamma I$ drops rapidly from infinity at $\gamma = \lambda_1(A)$ (the smallest positive eigenvalue of $A$) and usually $\gamma = \lambda_1(A)+\epsilon$, with a small $\epsilon > 0$, is acceptable. For CAREs from PDE boundary control problems, an inexpensive search for the smaller values of $n$ will lead to acceptable choices.

The cooling of steel profiles examples have components in $A$, $G$ and $H$ of magnitudes, respectively, of $O(10^3)$, $O(10^{-12})$ and $O(1)$. The resulting CAREs can be badly scaled, possibly leading to ill-condition. (Note from (1b) that a large error of $O(10^{-2})$ magnitude in $X$ may produce an negligible $O(10^{-16})$ contribution to the residual, because of the small elements in $G$.) We attempted to confirm this by estimating the corresponding condition numbers, as in [38, 39], although the various norms of $n^2 \times n^2$ matrices make the task difficult. From [6, 29], bounds for a condition number $K$ can be estimated by solving three large-scale Lyapunov equations

$$(A - GX)^\top Z_i + Z_i(A - GX) = -X^i, \quad (i = 0, 1, 2). \tag{25}$$

With $G$ and $H_k$ (approximating $X$ for a large enough $k$) being low-ranked, the techniques in [31] can be modified to solve (25) (with $H_k$ in place of $X$) in $O(n)$ flops for $i = 1, 2$, yielding the

lower bound $\widetilde{K}_L$ for the corresponding condition number $\kappa_{CARE}$. For the full-ranked $Z_0$ (with the right-hand-side $-I$ in (25)) and the sharper lower bound $K_L$ and upper bound $K_U$ of $\kappa_{CARE}$, the solution of (25) by doubling is of $O(n^2)$ complexity and expensive. For Examples 1–4 in this section, we present the bounds for $\kappa_{CARE}$ in Table 3. Note that we do not need the upper bounds $K_U$ to confirm ill-condition. In addition, for large-scale problems, the upper bound $\widetilde{K}_U$ involves the condition number $\kappa(X) \equiv \|X\| \cdot \|X^{-1}\|$ which will be theoretically large and practically impossible to estimate using the low-ranked approximation $H_k$. For details of the solution of (25) and the estimation of $\kappa_{CARE}$, see [31]. Note that the estimation of condition shares the same level of difficulty and stability as the solution of the original problem.

The efficient estimation or bounding of condition numbers for large-scale CAREs and DAREs remain an interesting open problem.

| Example | $n$ | $\widetilde{K}_L$ | $K_L$ | $K_U$ |
|---|---|---|---|---|
| 1 | 1357 | 3.4677e+2 | 9.8222e+2 | 3.0356e+3 |
| 2 | 5177 | 1.5472e+2 | 4.1797e+2 | 1.3560e+3 |
| 3 | 20209 | 4.7620e+2 | 5.7338e+2 | 2.8438e+3 |
| 4 | 79841 | 5.8594e+1 | 1.0032e+2 | 4.1309e+2 |

Table 3: Bounds for Condition Numbers, Cooling of Steel Profile Examples

For our examples. the condition number of $O(10^1)$ to $O(10^3)$ seem to suggest that the scaling problems of the CAREs does not make the condition of their solution too bad. Recall that worse accuracies ($\geq O(10^{-9})$) were obtained for Lyapunov equations from other Krylov subspace methods for similar examples in [9], [10, Figure 6.1, page 12] and [36, Figure 8.7, page 111]. The stability and perturbation analysis of large-scale CAREs and DAREs is still an open problem.

## 5.2 Numerical Results

The numerical results in Examples 1–3 were computed using MATLAB [33] Version R2010a, on a MacBook Pro with a 2.66GHz Intel Core 2 Duo processor and 4GB RAM, with machine accuracy $eps = 2.22 \times 10^{-16}$. For Example 4, the memory requirement exceeded the capacity of the MacBook Pro used for the other examples. A Dell PowerEdge R910 computer, with 4 $\times$ 8-core Intel Xeon 2.26 GHz CPUs and 1024GB RAM was used instead.

### Example 1

The cooling of steel profile example is quoted from Benner and Saak [11], with $n = 1357$, $m = 7$, $l = 6$.

From Table 4, the accuracy of $O(10^{-16})$, better than those in [9, 10], is achieved within seven iterations, in 2.55 seconds.

In our experiments in Examples 1–4, we relax $m_k$ to a maximum value of 150 and restricted $l_k$ by setting various $\tau_2$ or bounds for $l_k$. The reasoning behind the strategy is that $H_k = C_k T_k^{-1} C_k^\top$, which approximates the solution $X$ of the CARE after convergence. Letting $B_k$ to achieve high accuracy and using $\tau_s$ to control the balance between the growth of $C_k$ and the accuracy of $H_k$ yield acceptable results. On the other hand, other results suggest that the accuracy and growth of both $H_k$ and $G_k$ should be controlled in an equal manner, in order to achieve some sort of

optimal efficiency. However, this alternative strategy requires a more extensive and expensive search.

The sub-total CPU time $t_k = \sum_{i=1}^{k} \delta t_i$, with $\delta t_i$ being the CPU time required for the $i$th iteration.

| $k$ | $\|\delta H_k\|$ | $\|\delta H_k\|/\|H_k\|$ | $r_k$ | $\widetilde{r}_k$ | $m_k$ | $l_k$ | $\delta t_k$ | $t_k$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 2.1136e−02 | 4.0953e−01 | 1.4687e−02 | 1.2456e−01 | 14 | 12 | 5.00e−02 | 6.00e−02 |
| 2 | 2.4908e−02 | 3.2566e−01 | 7.1286e−03 | 4.4522e−02 | 28 | 17 | 4.00e−02 | 1.00e−01 |
| 3 | 1.8013e−02 | 1.9080e−01 | 1.7043e−03 | 8.9433e−03 | 56 | 21 | 1.00e−01 | 2.00e−01 |
| 4 | 5.3687e−03 | 5.3854e−02 | 1.0030e−04 | 5.0258e−04 | 112 | 25 | 2.50e−01 | 4.50e−01 |
| 5 | 3.3637e−04 | 3.3632e−03 | 3.5878e−07 | 1.7928e−06 | 150 | 27 | 6.80e−01 | 1.13e+00 |
| 6 | 1.2102e−06 | 1.2101e−05 | 8.3177e−12 | 4.1562e−11 | 150 | 28 | 1.42e+00 | 2.55e+00 |
| 7 | 3.1334e−11 | 3.1329e−10 | 5.3625e−17 | 2.6796e−16 | 150 | 29 | | |

Table 4: Example 1 ($\gamma = 0.5$, $\tau_1 = 10^{-30}$, $\tau_2 = 10^{-15}$, $m_k \leq 150$, $l_k \leq 50$)

**Example 2**

The cooling of steel profile example is quoted from Benner and Saak [11], with $n = 5177$, $m = 7$, $l = 6$. From Tables 5, the accuracy of $O(10^{-16})$ is achieved within five iterations, in 1.65 seconds.

| $k$ | $\|\delta H_k\|$ | $\|\delta H_k\|/\|H_k\|$ | $r_k$ | $\widetilde{r}_k$ | $m_k$ | $l_k$ | $\delta t_k$ | $t_k$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 9.0093e−03 | 1.8108e−01 | 2.0396e−03 | 2.0074e−02 | 14 | 12 | 5.00e−02 | 7.00e−02 |
| 2 | 2.5058e−03 | 4.7999e−02 | 1.0743e−04 | 1.0268e−03 | 28 | 18 | 1.00e−01 | 1.70e−01 |
| 3 | 1.3923e−04 | 2.6601e−03 | 3.0540e−07 | 2.9143e−06 | 56 | 19 | 3.30e−01 | 5.10e−01 |
| 4 | 3.9724e−07 | 7.5897e−06 | 7.2655e−12 | 6.9333e−11 | 112 | 21 | 1.14e+00 | 1.65e+00 |
| 5 | 1.0134e−11 | 1.9361e−10 | 4.9851e−17 | 4.7572e−16 | 150 | 23 | | |

Table 5: Example 2 ($\gamma = 0.5$, $\tau_1 = 10^{-30}$, $\tau_2 = 10^{-15}$, $m_k \leq 150$, $l_k \leq 50$)

**Example 3**

The cooling of steel profile example is quoted from Benner and Saak [11], with $n = 20209$, $m = 7$, $l = 6$. From Tables 6, the accuracy of $O(10^{-16})$ is achieved within seven iterations, in 44.8 seconds.

**Example 4**

The cooling of steel profile example is quoted from Benner and Saak [11], with $n = 79841$, $m = 7$, $l = 6$. From Table 7, the accuracy of $O(10^{-16})$ is achieved within nine iterations and 1,970 seconds (on the Dell PowerEdge R910). As in previous examples, the cost for the final iteration is minimal, as no preparation is required for the next iteration.

| $k$ | $\|\delta H_k\|$ | $\|\delta H_k\|/\|H_k\|$ | $r_k$ | $\widetilde{r}_k$ | $m_k$ | $l_k$ | $\delta t_k$ | $t_k$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 2.0970e−03 | 6.9503e−02 | 9.5344e−04 | 1.5267e−02 | 14 | 12 | 3.10e−01 | 3.40e−01 |
| 2 | 1.3979e−03 | 4.6276e−02 | 2.1852e−04 | 3.4613e−03 | 28 | 18 | 4.80e−01 | 8.20e−01 |
| 3 | 4.2151e−04 | 1.3954e−02 | 2.1310e−05 | 3.3655e−04 | 56 | 22 | 1.57e+00 | 2.39e+00 |
| 4 | 5.8714e−05 | 1.9437e−03 | 9.8193e−07 | 1.5503e−05 | 112 | 27 | 5.02e+00 | 7.41e+00 |
| 5 | 3.4402e−06 | 1.1389e−04 | 5.8239e−09 | 9.1950e−08 | 150 | 30 | 1.29e+01 | 2.03e+01 |
| 6 | 2.2067e−08 | 7.3052e−07 | 3.6993e−13 | 5.8405e−12 | 150 | 32 | 2.45e+01 | 4.48e+01 |
| 7 | 1.4286e−12 | 4.7294e−11 | 4.3294e−17 | 6.8354e−16 | 150 | 34 | | |

Table 6: Example 3 ($\gamma = 0.5$, $\tau_1 = 10^{-30}$, $\tau_2 = 10^{-15}$, $m_k \le 150$, $l_k \le 50$)

| $k$ | $\|\delta H_k\|$ | $\|\delta H_k\|/\|H_k\|$ | $r_k$ | $\widetilde{r}_k$ | $m_k$ | $l_k$ | $\delta t_k$ | $t_k$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 4.9846e−03 | 3.3034e−01 | 2.4666e−03 | 7.5553e−02 | 14 | 12 | 2.14e+00 | 2.19e+00 |
| 2 | 3.6912e−03 | 1.9665e−01 | 6.1205e−04 | 1.6040e−02 | 28 | 18 | 7.55e+00 | 9.74e+00 |
| 3 | 1.3897e−03 | 6.9826e−02 | 2.0793e−04 | 5.2166e−03 | 56 | 22 | 1.90e+01 | 2.87e+01 |
| 4 | 9.3791e−04 | 4.6948e−02 | 4.7165e−05 | 1.1799e−03 | 112 | 26 | 5.37e+01 | 8.24e+01 |
| 5 | 2.8650e−04 | 1.4341e−02 | 4.2899e−06 | 1.0732e−04 | 150 | 30 | 1.34e+02 | 2.16e+02 |
| 6 | 3.9604e−05 | 1.9823e−03 | 1.8623e−07 | 4.6586e−06 | 150 | 34 | 2.71e+02 | 4.87e+02 |
| 7 | 2.3081e−06 | 1.1553e−04 | 1.0998e−09 | 2.7512e−08 | 150 | 38 | 5.02e+02 | 9.88e+02 |
| 8 | 1.4933e−08 | 7.4744e−07 | 7.1130e−14 | 1.7794e−12 | 150 | 42 | 9.79e+02 | 1.97e+03 |
| 9 | 9.8762e−13 | 4.9435e−11 | 1.3150e−17 | 3.2896e−16 | 150 | 45 | | |

Table 7: Example 4 ($\gamma = 0.5$, $\tau_1 = 10^{-30}$, $\tau_2 = 10^{-15}$, $m_k \le 150$, $l_k \le 50$)

# 6 Conclusions

We have proposed a structure-preserving doubling algorithm for the large-scale discrete-time algebraic Riccati equation (1a), the SDA_ls, with $A$ being large and sparse(-like), and $B$ and $C$ being low-ranked. Similar continuous-time algebraic Riccati equations (1b) can be treated after an application of Cayley transform. The trick is to apply the Sherman-Morrison-Woodbury formula when appropriate and not to form $A_k$ (the iterate for $A$) explicitly. For well-behaved DAREs (or CAREs), with eigenvalues of the corresponding symplectic pencil (or Hamiltonian matrix) not on or near the unit circle (or the imaginary axis) and $I + G_k H_k$ being invertible for all $k$, low-ranked approximations to the solutions $X$ and $Y$ can be obtained efficiently. The convergence of the SDA_ls is quadratic, ignoring the compression and truncation of $B_k$ and $C_k$, as shown in [13, 14, 32]. The computational complexity and memory requirement are both $O(n)$, provided that the growth of $B_k$ and $C_k$ is controlled.

Similar to the methods in [8, 10], our technique can be applied when $A$ is large and sparse, or is a product (inverse) of such matrices (a matrix). The feasibility of the SDA_ls depends on whether $Av$ and $A^\top v$ for DAREs (or $A^{-1}v$ and $A^{-\top}v$ for CAREs) can be formed efficiently, for an arbitrary vector $v$.

In comparison to the techniques proposed previously, e.g. in [8, 10], there is no need for any inner iteration using ADI or Smith iteration, for the Lyapunov or Stein equations arisen from the inexact Newton's iteration. The associated estimation of parameters or initial starting values can also be avoided. Consequently, when successful, the SDA_ls solves large-scale DAREs

and CAREs efficiently. For instance in Section 5.2 , Example 3, of dimension $n = 20209$ with 204 million variables in the solution $X$, was solved using MATLAB on a MacBook Pro within 45 seconds to machine accuracy of $O(10^{-16})$.

For related research projects, strategies for the optimal setting of parameters and optimization of the computation and data structures in the SDA_ls, pre-processing of AREs to optimize their balance or condition, extension of the SDA_ls to Stein and Lyapunov equations as well as periodic systems, implementation on GPUs and other parallel computing platforms, associated computation of controllability and observability Gramians and balanced truncation methods for model order reduction and applications to Riccati differential equations (for optimal control problems with finite time horizon) and real-life problems such as boundary control of PDE models, are being investigated.

## Acknowledgement

This research work is partially supported by the National Science Council and the National Centre for Theoretical Sciences in Taiwan. The first author was also supported by the Majorscientific Research Fundation of Southeast University grant No. 3207011102 and the NSFC grant No. 11101080. Parts of this project were completed while the first author visited Monash University and the second author visited the National Taiwan University, and we would like to acknowledge the support from these Universities. The fourth author has been supported by a Monash Graduate Scholarship and a Monash International Postgraduate Research Scholarship. We would also like to thank Professor Peter Benner for pointing us to the "Cooling of Steel Profiles" examples in [11], which we adapted in Section 5.

## References

[1] L. AMODEI AND J.-M. BUCHOT. An invariant subspace method for large-scale algebraic Riccati equation, *Appl. Numer. Maths.*, 60:1-67–1082, 2010.

[2] A. ANTOULAS. *Approximation of Large-Scale Dynamical Systems*, SIAM Publications, Philadelphia, PA, 2005.

[3] U. BAUR. Low rank solution of data-sparse Sylvester equations, *Numer. Lin. Alg. Applic.*, **15** (2008) 837–851.

[4] P. BENNER. Solving large-scale control problems, *IEEE Control Systems Magazine*, **14** (2004) 44–59.

[5] P. BENNER. Editorial of special issue on "Large-Scale Matrix Equations of Special Type", *Numer. Lin. Alg. Appl.*, **15** (2008) 747–754.

[6] P. BENNER, A.J. LAUB AND V. MEHRMANN. Benchmarks for the numerical solution of algebraic Riccati equations, *IEEE Control Systems Magazine*, **7** (1997) 18–28.

[7] P. BENNER, V. MEHRMANN AND D. SORENSEN, EDS.. *Dimension Reduction of Large-Scale Systems*, Vol. **45** of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005.

[8] P. BENNER AND H. FASSBENDER. On the numerical solution of large-scale sparse discrete-time Riccati equations, *Adv. Comp. Maths.*, (to appear).

[9] P. BENNER AND 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 (http://ala2010.pmf.uns.ac.rs/presentations/4g1220pb.pdf).

[10] P. BENNER AND J. SAAK. A Galerkin-Newton-ADI method for solving large-scale algebraic Riccati equations, *DFG Priority Programme 1253 "Optimization with Partial Differential Equations"*, Preprint SPP1253-090, January 2010.

[11] P. BENNER AND J. SAAK. A semi-discretized heat transfer model for optimal cooling of steel profiles, in *Dimension Reduction of Large-Scale Systems*, ed. P. Benner, V. Mehrmann and D.C. Sorensen, Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, **45** (2005) 353–356.

[12] A. BENSOUSSAN, G. DA PRATO, M.C. DELFOUR AND S.K. MITTER. Representation and control of infinite dimensional systems, *Syst. & Control: Foundations & Applic.*, Birkhäuser Boston Inc., Boston, MA, 2nd Ed., 2007.

[13] E.K.-W. CHU, H.-Y. FAN AND W.-W. LIN. A structure-preserving doubling algorithm for continuous-time algebraic Riccati equations, *Linear Alg. Appl.*, **396** (2005) 55–80.

[14] E.K.-W. CHU, H.-Y. FAN, W.-W. LIN AND C.-S. WANG. A structure-preserving doubling algorithm for periodic discrete-time algebraic Riccati equations, *Int. J. Control*, **77** (2004) 767–788.

[15] E.K.-W. CHU, T.M. HUANG, W.-W. LIN AND C.-T. WU. Palindromic eigenvalue problems: a brief survey, *Taiwanese J. Math.*, **14** (2010) 743–779.

[16] B. DATTA. *Numerical Methods for Linear Control Systems*, Elsevier Academic Press, Boston, 2004.

[17] A. GHAVIM AND A. LAUB. Backward error, sensitivity, and refinement of computed solutions of algebraic Riccati equations, *Numer. Lin. Alg. Applic.*, **2** (1995) 29—49.

[18] G.H. GOLUB AND C.F. VAN LOAN. *Matrix Computations*, 2nd Ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[19] M. HEYOUNI AND K. JBILOU. An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equations, *Elect. Trans. Numer. Anal.*, 33:53–62, 2009.

[20] A. ISERLES. *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, Cambridge, 2003.

[21] K. JBILOU. Block Krylov subspace methods for large continuous-time algebraic Riccati equations, *Numer. Algorithms*, **34** (2003) 339–353.

[22] K. JBILOU. An Arnoldi based algorithm for large algebraic Riccati equations. *Appl. Math. Lett.*, **19** (2006) 437–444.

[23] K. JBILOU. Low rank approximate solutions to large Sylvester matrix equations, *Appl. Math. Comp.*, **177** (2006) 365–376.

[24] K. JBILOU. ADI preconditioned Krylov methods for large Lyapunov matrix equations, Preprint, 2008.

[25] K. JBILOU AND A. RIQUET. Projection methods for large Lyapunov matrix equations, *Lin. Alg. Appl.*, **415** (2006) 344–358.

[26] P. LANCASTER AND L. RODMAN, *Algebraic Riccati Equations*, Clarendon Press, Oxford, 1995.

[27] I. LASIECKA AND R. TRIGGIANI. *Control Theory for Partial Differential Equations: Continuous and Approximation Theories; I. Abstract Parabolic Systems*, Cambridge University Press, Cambridge, 2000.

[28] J.-R. LI AND J. WHITE. Low-rank solution of Lyapunov equations, *SIAM Review*, **46** (2004) 693–713.

[29] C. KENNEY AND G. HEWER. The sensitivity of the algebraic and differential Riccati equations, *SIAM J. Cont. Optim.*, **28** (1990) 50–69.

[30] T. LI , E.K.-W. CHU AND W.-W. LIN. Solving large-scale discrete-time algebraic Riccati equations by doubling, *Technical Report*, NCTS, National Chiao Tung University, Hsinchu, Taiwan, 2011.

[31] T. LI , E.K.-W. CHU, W.-W. LIN AND C.-Y. WENG. Solving large-scale Stein, Lyapunov and Sylvester equations by doubling, *Technical Report*, NCTS, National Chiao Tung University, Hsinchu, Taiwan, 2011.

[32] W.-W. LIN AND S.-F. XU, Convergence analysis of structure-preserving doubling algorithms for Riccati-type matrix equations, *SIAM J. Matrix Anal. Appl.*, **28** (2006) 26–39.

[33] MATHWORKS, *MATLAB User's Guide*, 2010.

[34] V.L. MEHRMANN, *The Autonomous Linear Quadratic Control Problem*, Lecture Notes in Control and Information Sciences, Vol. **163**, Springer Verlag, Berlin, 1991.

[35] M.A. RAMI AND X.Y. ZHOU, Linear matrix inequalities, Riccati equations, and indefinite stochastic linear quadratic controls, *IEEE Trans. Autom. Control*, **45** (2000) 1131–1143.

[36] J. SAAK. *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.

[37] J. SAAK, H. MENA AND P. BENNER. *Matrix Equation Sparse Solvers (MESS): a Matlab Toolbox for the Solution of Sparse Large-Scale Matrix Equations*, Chemnitz University of Technology, Germany, 2010.

[38] J.-G. SUN. Perturbation theory for algebraic Riccati equations, *SIAM J. Matrix Anal. Applic.*, **19** (1998) 39–65.

[39] J.-G. SUN. Condition numbers of algebraic Riccati equations in the Frobenius norm, *Lin. Alg. Applic.*, **350** (2002) 237–261.