fractal and fractional

Article

Structured Doubling Algorithm for a Class of Large-Scale Discrete-Time Algebraic Riccati Equations with High-Ranked Constant Term

Bo Yu , Chengxu Jiang * and Ning Dong *

School of Science, Hunan University of Technology, Zhuzhou 412007, China
* Correspondence: jiangcx@hut.edu.cn (C.J.); dongning@hut.edu.cn (N.D.)

Abstract: Consider the computation of the solution for a class of discrete-time algebraic Riccati equations (DAREs) with the low-ranked coefficient matrix G and the high-ranked constant matrix H. A structured doubling algorithm is proposed for large-scale problems when A is of lowrank. Compared to the existing doubling algorithm of $O(2^k n)$ flops at the k-th iteration, the newly developed version merely needs $O(n)$ flops for preprocessing and $O((k+1)^3 m^3)$ flopsfor iterations and is more proper for large-scale computations when $m \ll n$. The convergence and complexity of the algorithm are subsequently analyzed. Illustrative numerical experiments indicate that the presented algorithm, which consists of a dominant time-consuming preprocessing step and a trivially iterative step, is capable of computing the solution efficiently for large-scale DAREs.

Keywords: discrete-time algebraic Riccati equation; doubling algorithm; low-ranked matrix; high-ranked constant term

check for updates

Citation: Yu, B.; Jiang, C.; Dong, N. Structured Doubling Algorithm for a Class of Large-Scale Discrete-Time Algebraic Riccati Equations with High-Ranked Constant Term. *Fractal Fract.* **2023**, *7*, 193. https://doi.org/10.3390/fractalfract7020193

Academic Editors: Carlo Cattani, Neus Garrido, Francisco I. Chicharro and Paula Triguero-Navarro

Received: 11 January 2023
Revised: 5 February 2023
Accepted: 13 February 2023
Published: 14 February 2023

1. Introduction

Consider a discrete-time control system

$$x_{k+1} = Ax_k + Bu_k, \quad k = 0, 1, 2, \ldots,$$

where $A \in \mathbb{C}^{n\times n}$ and $B \in \mathbb{C}^{n\times l}$ with $l \leq n$. Here, $\mathbb{C}^{n\times m}$ stands for sets of $n \times m$ complex matrices. The linear quadratic regulator (LQR) control minimizes the energy or the cost functional

$$J_c(x_k, u_k) \equiv \sum_{k=0}^{\infty} [x_k^* H x_k + u_k^* R u_k]$$

with the Hermitian constant term $H \in \mathbb{C}^{n\times n}$ being positive semi-definite [1]. Here, the symbol "*" is the conjugate transpose of a vector or a matrix.

The corresponding optimal control is

$$u_k = -Fx_k$$

and the feedback gain matrix

$$F := (R + B^* XB)^{-1}(B^* XA)$$

can then be expressed in terms of the unique positive semi-definite stabilizing solution X of the discrete-time algebraic Riccati equation (DARE) [2]

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

where $G = BR^{-1}B^*$ with $R \in \mathbb{C}^{l\times l}$, $H \in \mathbb{C}^{n\times n}$ is Hermitian and positive semi-definite. In many control problems, the matrix $A \in \mathbb{C}^{n\times n}$ is sparse in the sense that the matrix-vector product Av and the inverse-vector product $A^{-1}v$ require $O(n)$ flops, respectively. The recent applications of the discrete-time control system can be found in [3] such as the wheeled robot and the airborne pursuer. There are also some applications (e.g., the singular Kalman filter) about the fractional Riccati equation, see [4,5] and the references therein.

The existence of the unique positive semi-definite solution X of DARE (1) has been well studied if (A, B) is d-stabilizable and (H, A) is observable, see [6,7] and their references for more details. The structure-preserving doubling algorithm (SDA) is one of the most efficient methods [7] to compute the unique positive semidefinite solution X via the following iteration

$$\begin{cases} A_{k+1} = A_k(I - G_kH_k)^{-1}A_k, \\ H_{k+1} = H_k + A_k^*H_k(I - G_kH_k)^{-1}A_k, \\ G_{k+1} = G_k + A_k(I - G_kH_k)^{-1}G_kA_k^* \end{cases} \tag{2}$$

with $A_0 = A$, $G_0 = -G$, $H_0 = H$. Regardless of the structure of coefficient matrices, the computational complexity of each iteration is about $O(n^3)$, obviously not fitting for large-scale problems. When the constant matrix H is low-ranked, the solution X is commonly numerically low-ranked and can be approximated by H_k in terms of a series of decomposed matrix factors, making the SDA feasible for large-scale DAREs [8]. If only the feedback gain matrix F is required without outputting the solution X, an adaptive version of the SDA in [9] still works for large-scale problems even if H is high-ranked. In that case, the solution X is no longer numerically low-ranked but can be stored in a sequence of matrix-vector products [9]. In both situations, the computational complexity of the SDA at the kth iteration costs about $O(2^kn)$ flops (i.e., the exponential increase in k), resulting in the intolerable iteration time when k is large.

In this paper, we consider DAREs with A of the low-ranked structure (which may not be sparse)

$$A = C_1SC_2^* \tag{3}$$

with $C_1, C_2 \in \mathbb{C}^{n\times m}$ and $S \in \mathbb{C}^{m\times m}$ ($m \ll n$). The motivation behind this is that the complexity of the SDA at the k-th iteration might be further reduced in this case and the DAREs, with the structure (3), have several applications in circuit-controlling areas, for example, the circuits system with C_1 and C_2 being the mesh inductance matrices, composed of the product of several mesh matrices (n is the number of meshes) and S being the resistance matrix [10]. To obtain the optimal feedback gain to control the circuit system, one is required to find the solution of the DARE (1).

The main contribution we made under the low-ranked structure (3) is that the computational complexity of the SDA at the k-th iteration can be further reduced to $O((k+1)^3m^3)$, far less than $O(2^kn)$ when $m \ll n$. As a result, the most time-consuming part of the SDA lies in the preprocessing step with a fixed computational complexity $O(n)$, and the other part for the iterations might be accordingly insignificant. Numerical experiments are implemented to validate the effectiveness of the presented algorithm, constituting a useful complement to the solver for computing the solution of DAREs.

The rest of the paper is organized as follows. In Section 2, we develop the structured SDA for DAREs with a low-ranked structure of A and construct its convergence. A detailed complexity analysis as well as the design of the termination criterion are established in Section 3. Section 4 is devoted to numerical experiments to indicate the efficiency of the proposed algorithm, and the conclusion is drawn in the last section.

Notation. Symbols $\mathbb{R}^{n\times n}$ and $\mathbb{C}^{n\times n}$ in this paper stand for sets of $n \times n$ real and complex matrices, respectively. I_n is the $n \times n$ identity matrix. For a matrix $A \in \mathbb{C}^{n\times n}$, $\sigma(A)$ and $\rho(A)$ denote, respectively, the spectrum and spectral radius of A. A Hermitian matrix $A > 0$ (≥ 0) when all its eigenvalues are positive (non-negative). Additionally, $M > N$ ($M \geq N$) if and only if $M - N > 0$ (≥ 0).

We also need the concept of the numerically low-ranked matrix.

Definition 1 ([8]). *A matrix A is said to be numerically low-ranked with respect to tolerance $\epsilon > 0$ if $\mathrm{rank}(A) \leq c_\epsilon$ for a constant c_ϵ associated with ϵ but independent of the size of A.*

2. Structured Doubling Algorithm

In this section, we describe the structured iteration scheme for DAREs with a high-ranked constant term and low-ranked A in (3). To avoid the inversion of large-scale matrices, the Sherman–Morrison–Woodbury formula (SMWF) [11,12] is first applied to the sparse-plus-low-ranked matrices to represent the corresponding structured matrices. Then, we aim at preserving the sparsity or the low-ranked structure of the iteration sequence rather than forming it explicitly. As a result, the SDA is capable of being implemented only with some small-scale matrices, referred to as kernels, and the complexity of the iteration can be ignored more easily than that of the preprocessing step for large-scale problems .

2.1. Iteration Scheme

Given the initial matrices $A_0 = C_1 S C_2$, $H_0 = H$, $G_0 = BR^{-1}B^*$, $S_0 = S$, $T_0 = 0$, $G_0 = R^{-1}$, and $B_0 = B$, the SDA will be organized according to the following format:

$$\begin{cases} A_k = C_1 S_k C_2^*, \\ H_k = H + C_2 T_k C_2^*, \\ G_k = B_k R_k B_k^* \end{cases} \tag{4}$$

for $k \geq 0$, where $S_k, T_k \in \mathbb{C}^{m\times m}$, $B_k \in \mathbb{C}^{n\times(km+l)}$, and $R_k \in \mathbb{C}^{(km+l)\times(km+l)}$. One merit of the above scheme (4) is that the sizes of kernels S_k and T_k are always invariant (i.e., $m \times m$) during iterations. Although the column of B_k and the size of R_k increase linearly with respect to k, the enhanced scale is generally small due to the fast convergence of the SDA. Then, G_k still hopefully maintains a low-ranked structure and could be derived and stored in an economic way.

Let $\Sigma_k = R_k^{-1} - B_k^* H_k B_k$. By applying the Sherman–Morrison–Woodbury formula (SMWF) [11], we have

$$(I - G_k H_k)^{-1} = I + B_k \Sigma_k^{-1} B_k^* H_k, \quad (I - H_k G_k)^{-1} = I + H_k B_k \Sigma_k^{-1} B_k^*. \tag{5}$$

Insertion (5) into the SDA (2) with currently available A_k, H_k and G_k yield $A_{k+1} = C_1 S_{k+1} C_2^*$, $H_{k+1} = H + C_2 T_{k+1} C_2^*$, $G_{k+1} = B_{k+1} R_{k+1} B_{k+1}^*$ with

$$\begin{cases} S_{k+1} = S_k (C_2^* C_1 + \Phi_k^* \Sigma_k^{-1} \Psi_k) S_k, \\ T_{k+1} = T_k + S_k^* (C_1^* H_k C_1 + \Psi_k^* \Sigma_k^{-1} \Psi_k) S_k, \\ B_{k+1} = [C_1, \ B_k], \\ R_{k+1} = \begin{bmatrix} \widetilde{R}_k & \\ & R_k \end{bmatrix} \end{cases} \tag{6}$$

and

$$\Phi_k = C_2^* B_k, \quad \Psi_k = C_1^* H_k B_k, \quad \widetilde{R}_k = S_k \Phi_k \Sigma_k^{-1} \Phi_k^* S_k^*.$$

The main computational task of (6) is the update of $H_k B_k$, $B_k^* H_k B_k$ in Ψ_k, Φ_k, Σ_k and the solutions of two linear system associated with Σ_k. Regardless of the concrete structure, the complexity of such calculations is $O(2^k n)$ [8,9]. A deeper observation made here will show that such computations can be further down to the complexity of $O((k+1)^3 m^3)$, far less than that of the preprocessing for large-scale problems with $m \ll n$. In fact, by setting $B_0 = B$, it follows from (6) that

$$B_k = [\overbrace{C_1, \ C_1, \ \ldots, \ C_1,}^{km} \ \overbrace{\ B \ }^{l}\ ] \, n$$

and thus

$$\Phi_k = [\overbrace{C_2^* C_1,\ C_2^* C_1,\ \ldots,\ C_2^* C_1}^{km},\ \overbrace{C_2^* B}^{l}]\ m\,. \tag{7}$$

Analogously, we have

$$\begin{aligned} H_k B_k &= H B_k + C_2 T_k C_2^* B_k \\ &= [\overbrace{HC_1 + C_2 T_k C_2^* C_1 \quad \ldots \quad HC_1 + C_2 T_k C_2^* C_1}^{km}\ \ \overbrace{HB + C_2 T_k C_2^* B}^{l}]\ n \end{aligned}$$

and

$$\Psi_k = [\overbrace{C_1^* H C_1 + C_1^* C_2 T_k C_2^* C_1 \quad \ldots \quad C_1^* H C_1 + C_1^* C_2 T_k C_2^* C_1}^{km}\ \ \overbrace{C_1^* H B + C_1^* C_2 T_k C_2^* B}^{l}]\ m\,.$$

Furthermore, as

$$\begin{aligned} B_k^* H_k B_k &= B_k^* H B_k + B_k^* C_2 T_k C_2^* B_k \\ &= \begin{array}{c} \begin{bmatrix} \overbrace{C_1^* H C_1 + C_1^* C_2 T_k C_2^* C_1 \quad \ldots \quad C_1^* H C_1 + C_1^* C_2 T_k C_2^* C_1}^{km} & \overbrace{C_1^* H B + C_1^* C_2 T_k C_2^* B}^{l} \\ \vdots \qquad\qquad \ddots \qquad\qquad \vdots & \vdots \\ C_1^* H C_1 + C_1^* C_2 T_k C_2^* C_1 \quad \ldots \quad C_1^* H C_1 + C_1^* C_2 T_k C_2^* C_1 & C_1^* H B + C_1^* C_2 T_k C_2^* B \\ B^* H C_1 + B^* C_2 T_k C_2^* C_1 \quad \ldots \quad B^* H C_1 + B^* C_2 T_k C_2^* C_1 & B^* H B + B^* C_2 T_k C_2^* B \end{bmatrix} \begin{array}{l} \left.\vphantom{\begin{matrix} a \\ a \\ a \end{matrix}}\right\} km \\ \} l \end{array} \end{array}, \end{aligned} \tag{8}$$

the update of the matrix

$$\Sigma_k = R_k^{-1} - B_k^* H_k B_k = \begin{bmatrix} \widetilde{R}_{k-1}^{-1} & \\ & R_{k-1}^{-1} \end{bmatrix} - B_k^* H_k B_k$$

will be of size $(km + l) \times (km + l)$. Now, suppose that matrices $C_1^* H C_1$, $C_1^* H B$, $B^* H B$, $C_2^* C_1$, and $C_2^* B$ are available in the preprocessing step, then Φ_k in (7) does not require additional computations. Additionally, Ψ_k and Σ_k can be obtained via updating several small scale matrix multiplications of the size $m \times m$, i.e., $(C_2^* C_1)^* T_k (C_2^* C_1)$, $(C_2^* C_1)^* T_k (C_2^* B)$, and $(C_2^* B)^* T_k (C_2^* B)$, and replicating them $km + l$ times (here $\widetilde{R}_{k-1}^{-1}$ and R_{k-1}^{-1} are assumed to be available in the last iteration for computing Σ_k). Consequently, the left computation lies in solving two linear systems $\Sigma_k U = \Phi_k$ and $\Sigma_k V = \Psi_k$ of size $(km + l) \times (km + l)$. We summarize the whole process in Algorithm 1 as below; the concrete complexity analysis in the next section shows that the iteration only costs about $O((k+1)^3 m^3)$ flops.

Remark 1. *The output matrices B_ϵ and R_ϵ are numerically low-ranked with respect to the tolerance ϵ. $\hat{T}$ is the matrix from the convergence of T_k given in the next subsection.*

Remark 2. *The QR decomposition of C_2 is for the derivation of the relative residual and also could be implemented in the preprocessing step. The computational complexity of the preprocessing part is about $O(n)$ flops, taking the dominant CPU time compared with the iteration part.*

Remark 3. *The computations of the iteration part and of the relative residual in the DARE cost about $O((k+1)^3 m^3)$ and $O(m^3)$ flops, respectively, much less than $O(n)$ of the preprocessing part when $m \ll n$. Hence, the main computation of Algorithm 1 concentrates on the preprocessing part.*

2.2. Convergence

To establish the convergence of Algorithm 1, we first review some results for iteration format (2).

Algorithm 1. Structured SDA for DAREs.

```
Input:       $C_1, C_2, S, B, R^{-1} = R^{-*}, H$ and tolerances $\tau_g$ and $\epsilon$, and $m_{\max}$;
Output:      $B_\epsilon \in \mathbb{C}^{n\times m_\epsilon}$, $R_\epsilon \in \mathbb{C}^{m_\epsilon\times m_\epsilon}$, $T_\epsilon \in \mathbb{C}^{m_\epsilon\times m_\epsilon}$, and normalized relative residual $\widetilde{r}_\epsilon$;
Preprocess:  Compute $C_1^*HC_1, C_1^*HB, B^*HB, C_2^*C_1, C_2^*B$ and the economic QR decomposition of $C_2$.
Iteration:   Set $T_0 = 0, S_0 = S, R_0 = -R^{-1}, B_0 = B, H_0 = H, \Sigma_0 = -(R + B^*HB)$,
                $\Phi_0 = C_2^*B, \Psi_0 = C_1^*HB$ and $k = 0$;
             For $k \geq 1$, do until convergence:
                Compute the relative residual $\widetilde{r}_k$ as in (11).
                If $\widetilde{r}_k \leq \epsilon$, set $B_\epsilon = B_k, R_\epsilon = R_k, T_\epsilon = T_k$ and $\widetilde{r}_\epsilon = \widetilde{r}_k$; Exit;
                End If
                Compute
                   $S_{k+1} = S_k(C_2^*C_1 + \Phi_k^*\Sigma_k^{-1}\Psi_k)S_k$;
                   $T_{k+1} = T_k + S_k^*(C_1^*H_kC_1 + \Psi_k^*\Sigma_k^{-1}\Psi_k)S_k$;
                   $R_{k+1} = \begin{bmatrix} S_k\Phi_k\Sigma_k^{-1}\Phi_k^*S_k^* & \\ & R_k \end{bmatrix}$;
                   Obtain $B_{k+1}^*H_{k+1}B_{k+1}$ in (8) with preprocessed matrices.
                   $\Sigma_{k+1}^{-1} = (I - R_{k+1}B_{k+1}^*H_{k+1}B_{k+1})^{-1}R_{k+1}$,
                   $\Phi_{k+1} = [C_2^*C_1,\ \Phi_k]$,
                   $\Psi_{k+1} = [C_1^*HC_1 + C_1^*C_2T_{k+1}C_2^*C_1, ..., C_1^*HC_1 + C_1^*C_2T_{k+1}C_2^*C_1,$
                              $C_1^*HB + C_1^*C_2T_{k+1}C_2^*B]$;
                Set $k \leftarrow k + 1$.
             End Do
```

Theorem 1 ([13]). *Assume that X and Y are the Hermitian and positive semi-definite solutions of the DARE (1) and its dual equation*

$$\mathcal{D}_d(Y) = -Y + AY(I + HY)^{-1}A^* + G = 0, \tag{9}$$

respectively. Let $P := (I + GX)^{-1}A$ and $Q := (I + HY)^{-1}A^*$. Then, the matrix sequences $\{A_k\}$, $\{G_k\}$ and $\{H_k\}$ generated by the SDA (2) satisfy

$$\begin{aligned}
&(1)\ A_k = (I + G_kX)P^{2^k};\\
&(2)\ H \leq H_k \leq H_{k+1} \leq X,\ X - H_k = (P^*)^{2^k}(X + XG_kX)P^{2^k};\\
&(3)\ G \leq G_k \leq G_{k+1} \leq Y,\ Y - G_k = (Q^*)^{2^k}(Y + YH_kY)Q^{2^k}.
\end{aligned} \tag{10}$$

It follows from (10) that

$$\begin{aligned}
&\|A_k\| \leq (1 + \|X\| \cdot \|Y\|)\|P^{2^k}\|,\\
&\|H_k - X\| \leq \|X\|(1 + \|X\| \cdot \|Y\|)\|P^{2^k}\|^2,\\
&\|G_k - Y\| \leq \|Y\|(1 + \|X\| \cdot \|Y\|)\|Q^{2^k}\|^2.
\end{aligned}$$

indicating that sequences $\{A_k\}$, $\{H_k\}$ and $\{G_k\}$ converge quadratically to zero, X, and Y, respectively, if $\rho(P) < 1$ and $\rho(Q) < 1$. By noting the decomposition $A_k = C_1S_kC_2^*$, the sequence $\{S_k\}$ must converge to zero. On the other hand, the decomposition $H_k = H + C_2T_kC_2^*$ implies that the sequence $\{T_k\}$ converges to some matrix $\hat{T} \in \mathbb{C}^{m\times m}$ such that the solution of the DARE $X = H + C_2\hat{T}C_2^*$. At last, the decomposition

$$\begin{aligned} G_k &= B_k R_k B_k^* \\ &= [C_1, ..., C_1, B] \cdot \begin{bmatrix} S_k \Phi_k^* \Sigma_k^{-1} \Phi_k S_k^* & & & \\ & \ddots & & \\ & & S_1 \Phi_1^* \Sigma_1^{-1} \Phi_1 S_1^* & \\ & & & S_0 \Phi_0^* \Sigma_0^{-1} \Phi_0 S_0^* \end{bmatrix} \cdot \begin{bmatrix} C_1^* \\ \vdots \\ C_1^* \\ B^* \end{bmatrix} \\ &= B S_0 \Phi_0^* \Sigma_0^{-1} \Phi_0 S_0^* B^* + \sum_{i=1}^{k} C_1 S_k \Phi_k^* \Sigma_k^{-1} \Phi_k S_k^* C_1^* \end{aligned}$$

indicates that the solution Y of the dual DARE has a numerically low-ranked decomposition $Y \approx B_\epsilon R_\epsilon B_\epsilon^*$ with respect to a sufficient small tolerance $\epsilon > 0$. So, we have the following corollary.

Corollary 1. *Suppose that X and Y are the Hermitian and positive semi-definite solutions of the DARE (1) and its dual form (9), respectively. Then, for Algorithm 1, the sequence $\{S_k\}$ converges to zero matrix quadratically, and $\{T_k\}$ converges to some matrix $\hat{T}$ with $X = H + C_2 \hat{T} C_2^*$. Moreover, for sufficiently large k, the matrix R_k is numerically low-ranked with respect to tolerance ϵ. That is, the solution Y of the dual Equation (9) has the low-ranked approximation $Y \approx B_\epsilon R_\epsilon B_\epsilon^*$, where matrices B_ϵ and R_ϵ associate with ϵ but independently of the size of Y.*

3. Computational Issues

3.1. Residual and Stop Criterion

Recalling the low-ranked structures of G and A, the residual of the DARE is

$$\begin{aligned} & -H_k + A^* H_k (I + G H_k)^{-1} A + H \\ = \; & C_2(-T_k + S^* C_1^* H_k (I + B R^{-1} B^* H_k)^{-1} C_1 S) C_2^* \\ = \; & C_2(-T_k + S^* (\Pi_k - \Xi_k \Theta_k^{-1} \Xi_k^*) S) C_2^* \end{aligned}$$

with

$$\begin{aligned} \Pi_k &= C_1^* H_k C_1 = C_1^* H C_1 + (C_2^* C_1)^* \cdot T_k \cdot C_2^* C_1, \\ \Xi_k &= C_1^* H_k B = C_1^* H B + (C_2^* C_1)^* \cdot T_k \cdot C_2^* B, \\ \Theta_k &= R + B^* H_k B = R + B^* H B + (C_2^* B)^* \cdot T_k \cdot C_2^* B. \end{aligned}$$

Let $C_2 = Q_{C_2} R_{C_2}$ ($Q_{C_2} \in \mathbb{C}^{n \times m}$, $R_{C_2} \in \mathbb{C}^{m \times m}$) be the economic QR decomposition of C_2, derived from the preprocessing step. The matrix norm of the residual is

$$r_k = \| R_{C_2} (-T_k + S^* (\Pi_k - \Xi_k \Theta_k^{-1} \Xi_k^*) S) R_{C_2}^* \|$$

and Algorithm 1 can be terminated by the normalized relative residual

$$\text{NRRes} = r_k / (t_k + s_k + m_k) := \widetilde{r}_k < \epsilon \tag{11}$$

with

$$r_k = \| R_{C_2} T_k R_{C_2}^* \|, \;\; s_k = \| R_{C_2} S_k^* \Pi_k S_k R_{C_2}^* \|, \;\; m_k = \| R_{C_2} S_k^* \Xi_k \Theta_k^{-1} \Xi_k^* S_k R_{C_2}^* \|.$$

Note that the calculation of NRRes only associates with several matrix operations with the small-scale $m \times m$, requiring $O(m^3)$ flops and far less than $O(n)$ when $m \ll n$.

3.2. Complexity Analysis

The main flops of Algorithm 1 come from the preprocessing step of forming matrices $C_1^* H C_1$, $C_1^* H B$, $B^* H B$, $C_2^* C_1$, $C_2^* B$ and QR decomposing $C_2 = Q_{C_2} R_{C_2}$ with the Householder transformation in [14,15]. Table 1 lists the details, where only the matrix $R_{C_2} \in \mathbb{C}^{m\times m}$ stored as $Q_{C_2} \in \mathbb{C}^{n\times m}$ is orthornormal satisfying $Q_{C_2}^* Q_{C_2} = I_m$.

Table 1. Complexity and memory of the preprocessing step in Algorithm 1.

Items	Flops	Memory
HC_1, HB	$2m(m+l)n$	$(m+l)n$
$C_1^*HC_1$, C_1^*HB, B^*HB	$2(m^2+ml+l^2)n$	m^2+ml+l^2
$C_2^*C_1$, C_2^*B	$2m(m+l)n$	m^2+ml
$C_2 = Q_{C_2}R_{C_2}$	$(10m^2+4ml+4l^2)n$	m^2
Total	$(16m^2+10ml+6l^2)n$	$3m^2+2ml+l^2+(m+l)n$

It is seen from Table 1 that the computation and the storage are both of $O(n)$ flops when $m, l \ll n$. We subsequently analyze the complexity of the iteration part. Assume that the LU decomposition is employed for solving the linear system $MZ = N$ with $M, Z, N \in \mathbb{C}^{(km+l)\times(km+l)}$. The flops and memory of the kth iteration are summarized in Table 2 below.

Table 2. Complexity and memory at kth iteration in Algorithm 1.

Items	Flops	Memory
$\Sigma_k^{-1}\Phi_k^*$, $\Sigma_k^{-1}\Psi_k^*$	$16(km+l)^2m$	$2(km+l)m$
$\Phi_k^*\Sigma_k^{-1}\Psi_k^*$, $\Phi_k^*\Sigma_k^{-1}\Phi_k^*$, $\Psi_k^*\Sigma_k^{-1}\Psi_k^*$	$6m^2(km+l)$	$3m^2$
S_{k+1}	$4m^3$	m^2
T_{k+1}	$8m^3$	m^2
R_{k+1}	$4m^3$	m^2
$B_{k+1}^*H_{k+1}B_{k+1}$	$2m(2m^2+2ml+l^2)$	$((k+1)m+l)^2$
Σ_{k+1}	$2((k+1)m)^3$	$((k+1)m)^2$
Φ_{k+1}	—	$((k+1)m+l)m$
Ψ_{k+1}	$4m^2(m+l)$	$((k+1)m+l)m$
Total	$(24+2(k+1)^3)m^3$ $+2ml(4m+l)$ $+2m(km+l)$ $(8(km+l)+3m)$	$(4(k+1)^2+2k+6)m^2$ $+(2k+6)ml$ $+l^2$

Table 2 shows that the complexity of the kth iteration in Algorithm 1 is about $O((k+1)^3m^3)$, far less than $O(n)$ of the preprocessing step when $m \ll n$. Thus, the dominantly calculating cost of Algorithm 1 locates at the preprocessing step; however, it is still far less than the exponentially increasing complexity $O(2^kn)$ [8,9] when k grows large.

4. Numerical Experiments

In this section, we will show the effectiveness of Algorithm 1 to calculate the solution X of the large-scale DARE (1). The code was programmed by Matlab 2014a [16], and all computations were implemented in a ThinkPad notebook with 2.4 GHz Intel i5-6200 CPU and 8G memory. The stop criterion is the NRRes in (11) with a proper tolerance ϵ. To show the location of the dominant computations in Algorithm 1, we record the ratio of iteration time and total time in the percentage

$$\mathrm{R_t} = \frac{\text{TIME-I}}{\text{TIME-P + TIME-I}} \times 100\%, \tag{12}$$

where "TIME-P" represents the pre-processing time elapsed for forming matrices associated with n, and "TIME-I" stands for the costed CPU time for iterations.

Example 1. *The first example is devised to measure the actual error between the true solution X and the approximated solution H_k computed from Algorithm 1. Let $S = 1$, $C_1 = \mathbf{1}/\|\mathbf{1}\| \in \mathbb{R}^{n\times 1}$ and $C_2 \in \mathbb{R}^{n\times 1}$ be a vector such that $C_1^* C_2 = 0$ and $C_i^* C_i = 1$ $(i = 1, 2)$, where $\mathbf{1}$ is a vector with all elements 1. Set $B^* = [0, 0, ..., 0, 1] \in \mathbb{R}^{1\times n}$, $R = 1$ and $H = I_n$. Then, the solution of the DARE is*

$$X = I_n - UU^*$$

with

$$U = wC_2$$

and

$$w^2 = \frac{C_{2_n}^2 - 2 + \sqrt{(C_{2_n}^2 - 2)^2 + 4C_{2_n}^2(2 - C_{1_n}^2)}}{2C_{2_n}^2}$$

being the root of the equation $(1 - w^2)(2 + w^2 * C_{2_n}^2) - C_{1_n}^2 = 0$. Here, C_{1_n} and C_{2_n} represent the n-th element of C_1 and C_2, respectively. The coefficient matrices are $A = C_1 S C_2^*$ and $G = BR^{-1}B^*$. The principle of selecting the above vectors and matrices is for the convenient construction of the true solution of the DARE. Then, we can evaluate the error between the computed approximated solution and the true solution.

We consider the medium scales with $n = 1000$, 3000, and 5000 to test the accuracy of Algorithm 1, which is terminated when the NRRes is less the prescribed $\epsilon = 1.0 \times 10^{-13}$. Numerical experiments show that Algorithm 1 always takes three iterations to obtain the approximate solution for all tested dimensions n. The obtained results on NRRes and Errors are listed in Table 3.

Table 3. Residual and actual errors in Example 1.

n	1000	3000	5000
NRRes_1	9.99×10^{-1}	9.99×10^{-1}	9.99×10^{-1}
NRRes_2	2.49×10^{-7}	2.77×10^{-7}	9.99×10^{-8}
NRRes_3	6.24×10^{-17}	3.62×10^{-17}	4.48×10^{-18}
$\|\|H_1 - X\|\|$	9.99×10^{-1}	9.99×10^{-1}	9.99×10^{-1}
$\|\|H_2 - X\|\|$	2.26×10^{-7}	3.33×10^{-7}	2.75×10^{-7}
$\|\|H_3 - X\|\|$	1.24×10^{-14}	1.25×10^{-14}	1.24×10^{-14}
R_t	37.5%	12.6%	6.8%

It is seen from the table that Algorithm 1 is efficient to calculate the solution of the DARE. In fact, for different dimensions, the actual error between H_k and the solution X is less than the prescribed accuracy after three iterations, and the derived relative residual is down to a lower level about 10^{-17} to 10^{-18}. Especially, the value of R_t gradually decreases with the rising scale of n, indicating that the CPU time for iterations takes only a small part of the whole for large-scale problems.

Example 2. *Randomly generate matrices $C_1, C_2, B \in \mathbb{R}^{n\times m}$ and define*

$$C_1 := \frac{C_1}{\sqrt{2}\|C_1\|^{1/2}},\ C_2 := \frac{C_2}{\sqrt{2}\|C_2\|^{1/2}},\ B := \frac{B}{\|B\|^{1/2}}.$$

Set $R = S = I_m$ and consider the DARE (1) with $A = C_1 C_2^*$, $G = BB^*$, and $H = I - BB^* - \frac{1}{2}C_2 C_2^* + C_2 C_1^* BB^* C_1 C_2^*$. It is not difficult to see the solution of the DARE is $X = I - BB^*$. Similarly, the principle of selecting the above matrices is for the convenience of evaluating the error.

We take $n = 5000, 6000, 7000$ to test the error between the true solution and the computed solution. The obtained results together with the NRRes are plotted in Figures 1–3. Still, R_t represents the ratio of the iteration time and the total time.

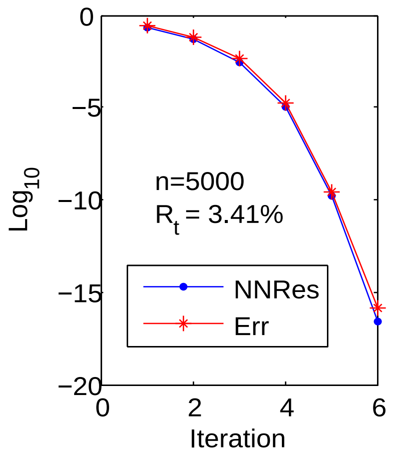

Figure 1. History of NRRes and Error for $n = 5000$ in Example 2.

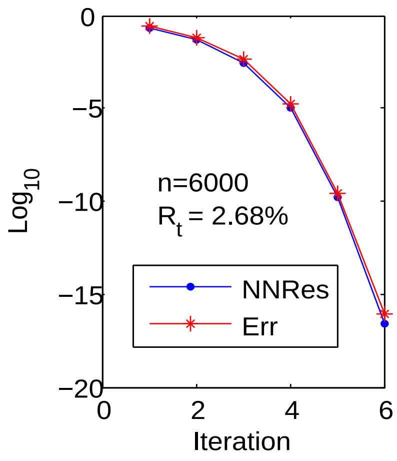

Figure 2. History of NRRes and Error for $n = 6000$ in Example 2.

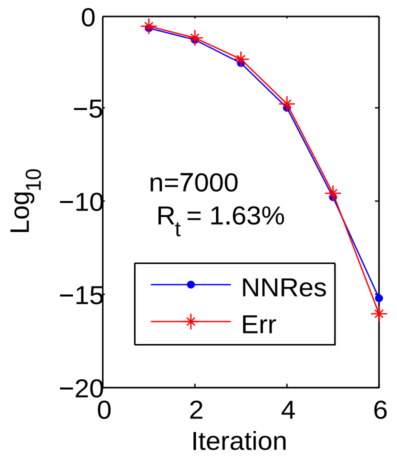

Figure 3. History of NRRes and Error for $n = 7000$ in Example 2.

Figures 1–3 show that as the number of iterations increases, the NRRes and errors decrease exponentially and Algorithm 1 terminates at the 6th iteration. In all experiments, the preprocessing time for three cases varied from 0.1 to 0.2 s, while the iterative time only took from 0.0032 to 0.0035 s, costing a small part of the whole CPU time. More experiments also indicated that the ratio $\mathrm{R_t}$ became smaller as the scale of the problem increased.

Example 3. *This example comes from a proper modification of the circuits from the magneto-quasistatic Maxwell equations ([17,18]). The matrix* $S \in \mathbb{R}^{632\times632}$ *represents the DC resistance matrix of each current filament (see Figure 4) and* C_1 *as well as* $C_2 \in \mathbb{R}^{n\times632}$ *associated with the mesh matrices. Let* $R = 1$, $B^\top = [1,0,\ldots,0] \in \mathbb{R}^{1\times n}$ *and* $H = I_n$. *We randomly generate the matrix* $U \in \mathbb{R}^{n\times632}$ *and define*

$$U = \frac{U}{\|U\|^{1/2}}, \quad C_1 = U, \quad C_2 = U.$$

The tolerance ϵ *is taken as* 10^{-14}, *and the dimensions are* $n = i \times 10^5$ $(i = 1,2,\ldots,6)$.

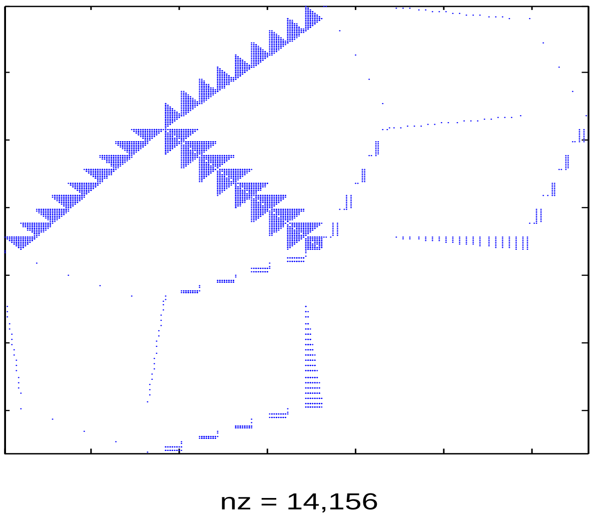

Figure 4. The structure of the DC resistance matrix of each current filament.

For all cases in our experiments, Algorithm 1 was observed attaining the relative residual level below 1.01×10^{-16} at the 4-th iteration. The elapsed CPU time and the ratio $\mathrm{R_t}$ are plotted in Figure 5, where "$\mathrm{T_{pre}}$" and "$\mathrm{T_{it}}$" record the CPU time for the preprocessing and for the iteration, respectively. One can see from the figure that as the scale n rises, the preprocessing time becomes more dominant (about 112 s at $n = 600{,}000$) but the iteration time remains almost unchanged (about 3.5 s for all n). The gradually reduced ratio $\mathrm{R_t}$ also illustrates that the main computations of Algorithm 1 when solving the large-scale problems lie in the preprocessing step of $O(n)$ flops, much less than the exponentially increasing one of $O(2^k n)$ in [8,9].

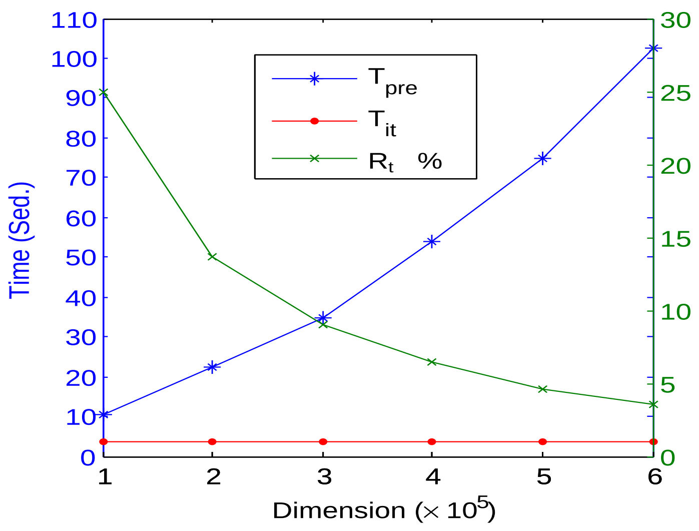

Figure 5. Preprocessing time (T_{pre}), iteration time (T_{it}), and R_t for different dimensions in Example 3.

5. Conclusions

We have proposed an efficient algorithm to solve the large-scale DAREs with low-ranked matrices A and G and a high-rank matrix H. Compared with the SDA of the complexity $O(2^k n)$ in [8,9], the newly developed algorithm only requires preprocessing step of $O(n)$ flops and iteration step of $O((k+1)^3 m^3)$ flops. For large-scale problems with $m \ll n$, the main computations of the whole algorithm lie in the preprocessing step with several matrix multiplications and an economic QR decomposition, while the elapsed CPU time for the iteration part is trivial. Some numerical experiments validate the effectiveness of the proposed algorithm. For future work, we may investigate the possibility of the SDA for solving large-scale DAREs with the structure of sparse-plus-low-rank in A, where the possible difficulty might be understanding the concrete structure of the iterative matrix and knowing how to compute and store it efficiently.

Author Contributions: Conceptualization, B.Y.; methodology, N.D.; software, C.J.; validation, B.Y.; and formal analysis, N.D. All authors have read and agreed to the final version of this manuscript.

Funding: This work was supported partly by the NSF of China (11801163), the NSF of Hunan Province (2021JJ50032, 2023JJ50040), and the Key Foundation of the Educational Department of Hunan Province (20A150).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Athan, M.; Falb, P.L. *Optimal Control: An Introduction to the Theory and Its Applications*; McGraw-Hill: New York, NY, USA, 1965.
2. Lancaster, P.; Rodman, L. *Algebraic Riccati Equations*; Clarendon Press: Oxford, UK, 1999.
3. Rabbath, C.A.; Léchevin N. *Discrete-Time Control System Design with Applications*; Springer Science and Business Media: Berlin/Heidelberg, Germany, 2013.
4. Nosrati K.; Shafiee M. On the convergence and stability of fractional singular Kalman filter and Riccati equation. *J. Frankl. Inst.* **2020**, *357*, 7188–7210. [CrossRef]
5. Trujillo, J.J.; Ungureanu, V.M. Optimal control of discrete-time linear fractional-order systems with multiplicative noise. *Int. J. Control.* **2018**, *91*, 57–69. [CrossRef]
6. Chu, E.K.-W.; Fan, H.-Y.; Lin, W.-W. A structure-preserving doubling algorithm for continuous-time algebraic Riccati equations. *Lin. Alg. Appl.* **2005**, 396, 55–80. [CrossRef]
7. Chu, E.K.-W.; Fan, H.-Y.; Lin, W.-W.; Wang, C.-S. A structure-preserving doubling algorithm for periodic discrete-time algebraic Riccati equations. *Int. J. Control* **2004**, *77*, 767–788. [CrossRef]
8. Chu, E.K.-W.; Weng, P.C.-Y. Large-scale discrete-time algebraic Riccati equations—Doubling algorithm and error analysis. *J. Comp. Appl. Maths.* **2015**, 277, 115–126. [CrossRef]

9. Yu, B.; Fan, H.-Y.; Chu, E.K.-W. Large-scale algebraic Riccati equations with high-rank constant terms. *J. Comput. Appl. Math.* **2019**, *361*, 130–143. [CrossRef]
10. Kamon, M.; Wang, F.; White, J. Generating nearly optimally compact models from Krylov-subspace based reduced order models. *IEEE Trans. Circuits -Syst.-Ii Analog Digit. Signal Process.* **2000**, *47*, 239–248. [CrossRef]
11. Golub, G.H.; Van Loan, C.F. *Matrix Computations*, 3rd ed.; Johns Hopkins University Press: Baltimore, MD, USA, 1996.
12. Yu, B.; Li, D.-H.; Dong, N. Low memory and low complexity iterative schemes for a nonsymmetric algebraic Riccati equation arising from transport theory. *J. Comput. Appl. Math.* **2013**, *250*, 175–189. [CrossRef]
13. Lin, W.-W.; Xu, S.-F. Convergence analysis of structure-preserving doubling algorithms for Riccati-type matrix equations. *SIAM J. Matrix Anal. Appl.* **2006**, *28*, 26–39. [CrossRef]
14. Bhatia, R. *Matrix Analysis, Graduate Texts in Mathematics*; Springer: Berlin/Heidelberg, Germany, 1997.
15. Higham, N.J. *Functions of Matrices: Theory and Computation*; SIAM: Philadelphia, PA, USA, 2008.
16. Higham, D.J.; Higham, N.J. *MATLAB Guide*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2016.
17. Miguel, S.L.; Kamon, M.; Elfadel, I.; White, J. A coordinate transformed Arnoldi algorithm for generating guaranteed stable reduced order models of RLC circuits. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, USA, 10–14 November 1996; pp. 288–294.
18. Odabasioglu, A.; Celik, M.; Pileggi, L.T. PRIMA: Passive Reduced order Interconnect Macro modeling Algorithm. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **1998**, *17*, 645–654. [CrossRef]