Computational and Applied Mathematics (2024) 43:256
https://doi.org/10.1007/s40314-024-02774-8

Low-rank generalized alternating direction implicit iteration method for solving matrix equations

Juan Zhang[1] · Wenlu Xun[2]

Received: 4 March 2024 / Revised: 3 May 2024 / Accepted: 9 May 2024 /
Published online: 27 May 2024

Abstract
This paper presents an effective low-rank generalized alternating direction implicit iteration (R-GADI) method for solving large-scale sparse and stable Lyapunov matrix equations and continuous-time algebraic Riccati matrix equations. The method is based on generalized alternating direction implicit iteration (GADI), which exploits the low-rank property of matrices and utilizes the Cholesky factorization approach for solving. The advantage of the new algorithm lies in its direct and efficient low-rank formulation, which is a variant of the Cholesky decomposition in the Lyapunov GADI method, saving storage space and making it computationally effective. When solving the continuous-time algebraic Riccati matrix equation, the Riccati equation is first simplified to a Lyapunov equation using the Newton method, and then the R-GADI method is employed for computation. Additionally, we analyze the convergence of the R-GADI method and prove its consistency with the convergence of the GADI method. Finally, the effectiveness of the new algorithm is demonstrated through corresponding numerical experiments.

Keywords Lyapunov equation · Continuous-time algebraic Riccati equation · Low-rank generalized alternating direction implicit iteration

Mathematics Subject Classification 62F15 · 62J05 · 65F08 · 65F45

1 Introduction

This paper focuses on the numerical solution of large-scale continuous-time algebraic Riccati matrix equations (CARE):

Juan Zhang The work was supported in part by Hunan Youth Science and Technology Innovation Talents Project (2021RC3110), the Key Project of Education Department of Hunan Province (2023YFB3001604).

✉ Juan Zhang
zhangjuan@xtu.edu.cn

[1] Key Laboratory of Intelligent Computing and Information Processing of Ministry of Education, Hunan Key Laboratory for Computation and Simulation in Science and Engineering, School of Mathematics and Computational Science, Xiangtan University, Xiangtan, Hunan, China

[2] School of Mathematics and Computational Science, Xiangtan University, Xiangtan, Hunan, China

$$A^T X + XA + Q - XGX = 0, \tag{1}$$

where $Q = C^T C$ is symmetric and positive definite, $G = BB^T$ is symmetric and positive semi-definite, and $A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times m},\ C \in \mathbb{R}^{p\times n}$ are known matrices, and $X \in \mathbb{R}^{n\times n}$ is unknown matrix. Here, rank(C)=p, rank(B)=m and $p,\ m \ll n$. The numerical treatment of this type of equation plays a significant role in various fields. For instance, linear quadratic regulators (Anderson and Moore 1990), linear model reduction systems based on equilibrium (Roberts 1980), parabolic partial differential equations and transport theory (Jonckheere and Silverman 1983; Saak 2009), Wiener-Hopf factorization of Markov chains (Williams 1982), and factorization of rational matrix functions (Clancey and Gohberg 1981), etc. In this paper, we assume that the coefficient matrix A is sparse. Typically, the stable solution to equation (1) is desired, where the solution X is symmetric and positive semi-definite, and $A - GX$ is stable, meaning its eigenvalues have negative real parts. Such stable solutions exist and under certain assumptions, it is unique (Lancaster and Rodman 1995). When n is large, it is common to seek low-rank approximations of the symmetric positive semi-definite solution in the form of $ZZ^T \approx X$, where the column rank of Z is low, i.e., rank(Z) $\ll n$. Since storing the full matrix X requires a significant amount of memory, considering only the storage of the matrix Z allows us to optimize resource utilization when dealing with large-scale problems.

Firstly, we present an application of Eq. (1) by considering a linear time-invariant control system (Kleinman 1968):

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \quad x(0) = x_0, \\ y(t) = Cx(t), \end{cases}$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^m$ is the control vector, and $y(t) \in \mathbb{R}^p$ is the output vector. Quadratic optimal control aims to minimize

$$J(x_0, u) = \frac{1}{2}\int_0^{+\infty} (y(t)^T y(t) + u(t)^T u(t))dt.$$

Assuming that (A, B) is stabilizable, i.e., there exists a matrix S such that $A - BS$ is stable. And (C, A) is detectable, i.e., (A^T, C^T) is stable, there exists a unique optimal solution $\bar{u}$ that minimizes the functional $J(x_0, u)$ (Wonham 1968), which can be determined through the feedback operator P, i.e., $\bar{u}(t) = Px(t)$, where $P = B^T X$, and $X \in \mathbb{R}^{n\times n}$ is the unique symmetric positive semi-definite stable solution of the matrix Eq. (1).

There are many methods have been proposed for the numerical solution of Eq. (1), including the Schur method (Laub 1979), matrix sign function (Roberts 1980; Bai and Demmel 1998), structured doubling algorithm (Guo et al. 2005), symplectic Lanczos method (Benner and Fassbender 1997), and projection methods based on the global Arnoldi process of the Krylov subspace (Jbilou 2003; Heyouni and Jbilou 2009). However, these methods often require multiple iterations to obtain an accurate approximate solution, leading to significant increases in computational time and memory requirements. To address this issue, some scholars have also investigated approximate low-rank solutions for computing large sparse matrix equations. Typically, we combine the Newton's iteration method with the alternating direction implicit (ADI) algorithm to solve such equations, and take advantage of the quadratic local convergence properties of Newton's method. However, at each Newton iteration step, solving a large Lyapunov matrix equation is required to obtain the next iteration solution. The continuous-time Lyapunov matrix equation (Lu and Wachspress 1991) as follows:

$$F^T X + XF = Q, \tag{2}$$

where $Q = C^T C$, $F \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{p\times n}$. From Eq. (1), we observe that equation (2) is a specific case of Eq. (1) when $G = 0$. If the spectrum of matrix F is in the positive-real half-plane. Under these conditions F and $-F^T$ have no common characteristic roots (Rutherford 1932), and there is a unique symmetric solution X.

Heinkenschloss et al. (2016) proposed the low-rank Kleinman-Newton ADI iteration method to solve Eq. (1). In the computational process, the Lyapunov matrix equation is solved using a low-rank Cholesky factorization. This method is based on solving linear systems with shifted matrices $A+\alpha I$, where α is the ADI parameter. However, determining the optimal ADI parameter and finding approximations for the Lyapunov equation increase the burden on memory requirements and computational time. Recently, Wong and Balakrishnan (2005) introduced an algorithm called Quadratic ADI (qADI) method to solve the algebraic Riccati equation (1). Their method is a direct extension of the Lyapunov ADI method. Additionally, Wong and Balakrishnan provided a low-level variant of this algorithm. However, this variant has a significant drawback: in every step, all low-rank factors need to be reconstructed, which greatly affects the performance of the algorithm. In addition to the qADI method, several approaches for solving large-scale Riccati equations have emerged in recent literature. For instance, Amodei and Buchot (2010) obtain approximate solutions by computing low-dimensional subspaces of the associated Hamiltonian matrix. Benner et al. (2018) propose a novel ADI iteration method called Riccati ADI (RADI), which expands each factor by several multiples of columns or rows while keeping the elements from previous steps unchanged. Their method yields low-rank Lyapunov ADI iteration formulas.

In addition, there are currently many algorithms available for solving the Lyapunov Eq. (2). For instance, fixed-point iteration (Astudillo and Gijzen 2016), Krylov subspace methods (Druskin et al. 2011), and rank-2 updates (Ren et al. 2018) are commonly employed. Zhou et al. (2015) proposed a generalized Hermitian and skew-Hermitian splitting (GHSS) iterative method, demonstrating convergence under certain assumptions. ADI iterative methods (Benner et al. 2009) can accelerate convergence if the optimal shifts of A and A^T can be effectively estimated. Therefore, for stable Lyapunov Eq. (2), when solving large-scale sparse problems, ADI iteration methods are often preferred as they preserve sparsity and are more amenable to parallelization in most cases. Recent theoretical results (Penzl 2000; Simoncini 2008; Li and White 2002) indicate that using the Cholesky factorization-alternating direction implicit (CF-ADI) algorithm to compute low-rank approximations for the Lyapunov equation is effective. Based on the (Jiang et al. 2021), the GADI iteration for solving large-scale sparse linear systems can be described as follows:

$$Ax = b, \quad A \in \mathbb{R}^{n\times n}, \quad x, b \in \mathbb{R}^n. \tag{3}$$

- Firstly, the matrix A is split, assuming that A can be represented as $A = M + N$, and then assign parameters to obtain

$$\begin{aligned} &\alpha x + Mx = \alpha x - Nx + b,\\ &\alpha x + Nx = Nx - (1-\omega)\alpha x + (1-\omega)\alpha x + \alpha x = Nx - (1-\omega)\alpha x + (2-\omega)\alpha x. \end{aligned}$$

- Next, the algorithm is obtained by alternating between these two splittings. Given an initial $x_0 = 0$, the GADI iteration computes a sequence x_k as follows

$$\begin{cases} (\alpha I + M)x_{k+\frac{1}{2}} = (\alpha I - N)x_k + b,\\ (\alpha I + N)x_{k+1} = (N - (1-\omega)\alpha I)x_k + (2-\omega)\alpha x_{k+\frac{1}{2}}, \end{cases} \tag{4}$$

with $\alpha > 0, \quad 0 \le \omega < 2$.

- Specifically, the matrix A is split into $A = H + S$, where $H = \frac{A+A^*}{2}$ is a Hermiten matrix, and $S = \frac{A-A^*}{2}$ is a skew-Hermitian matrix, then GADI-HS format is obtained

$$\begin{cases} (\alpha I + H)x_{k+\frac{1}{2}} = (\alpha I - S)x_k + b, \\ (\alpha I + S)x_{k+1} = (S - (1-\omega)\alpha I)x_k + (2-\omega)\alpha x_{k+\frac{1}{2}}. \end{cases} \tag{5}$$

In this paper, we propose a low-rank generalized alternating direction implicit iteration (R-GADI) algorithm, which is an improvement over the GADI algorithm for solving the Lyapunov equation. We represent the solution as a low-rank approximation $X \approx VW^T$, where rank(V) and rank$(W) \ll n$. During the computation, the R-GADI method provides a low-rank approximation of the solution X, eliminating the need to store X at each iteration and reducing storage requirements. Additionally, we combine the Kleinman-Newton method with R-GADI (referred to as Kleinman-Newton-RGADI) to solve the Riccati Eq. (1).This method is a variant of the Newton-GADI algorithm (Li et al. 2022), which significantly reduces the total number of ADI iterations and thus lowers the overall computational cost. Finally, numerical examples in the paper demonstrate the effectiveness of the proposed algorithm.

The remaining structure of this paper is as follows: in Sect. 2, we introduce the R-GADI iteration format for solving the Lyapunov equation and demonstrates the consistency between R-GADI and GADI iterations in terms of convergence. The selection of parameters, algorithm complexity, and comparison with other methods are discussed, along with relevant numerical examples. In Sect. 3, we first transform the Riccati equation into the Lyapunov equation using the Kleinman-Newton method, and then present the R-GADI iteration format. Convergence, algorithm complexity, and additional numerical examples are also discussed to validate the effectiveness of the proposed algorithm. Finally, in Sect. 4 concludes the paper by summarizing the findings and offering some concluding remarks.

In this article, we use the following notation: $\mathbb{R}^{n\times m}$ denotes the set of all $n\times m$ real matrices. If $A \in \mathbb{R}^{n\times n}$, then A^T and A^{-1} represent the transposition and inverse of A, respectively. The sets of eigenvalues and singular values of A are denoted as $\Lambda(A) = \{\lambda_i(A),\ i = 1, 2, \cdots, n\}$ and $\Sigma(A) = \{\sigma_i(A),\ i = 1, 2, \cdots, n\}$, where $\lambda_i(A)$ and $\sigma_i(A)$ are the i-th eigenvalue and the i-th singular value of A, respectively. $\rho(A) = \max_{1\le i\le n}\{|\lambda_i(A)|\}$ represents the spectral radius of A. $A > 0$ ($A \ge 0$) indicates that A is positive definite (positive semidefinite), $\|A\|_2$ denotes the 2-norm of A, Re(A) and Im(A) represent the real and imaginary parts of the eigenvalues of A, respectively. $A \otimes I$ denotes the Kronecker product of A and I.

Definition 1 Let $A_1 = [a_{ij}] \in \mathbb{C}^{m\times n},\ \ B_1 \in \mathbb{C}^{p\times q}$, then

$$A_1 \otimes B_1 = \begin{pmatrix} a_{11}B_1 & a_{12}B_1 & \cdots & a_{1n}B_1 \\ a_{21}B_1 & a_{22}B_1 & \cdots & a_{2n}B_1 \\ \vdots & \vdots & & \vdots \\ a_{m1}B_1 & a_{m2}B_1 & \cdots & a_{mn}B_1 \end{pmatrix} \in \mathbb{C}^{mp\times nq},$$

it's called Kronecker product of A_1 and B_1.

Definition 2 If the vectorization operator vec satisfies $\mathbb{C}^{m\times n} \to \mathbb{C}^{mn}$:

$$vec(X_1) = (x_1^T, x_2^T, \cdots, x_n^T)^T,\ \ X_1 = [x_1, x_2, \cdots, x_n] \in \mathbb{C}^{m\times n},$$

then this operator is called a straightening operator.

2 Low rank GADI for solving Lyapunov equation

2.1 Derivation of iterative format

Firstly, we consider the ADI iterative method for solving the Lyapunov Eq. (2) with a single parameter.

$$\begin{cases} (F^T + \alpha I)X_{k+\frac{1}{2}} = Q - X_k(F - \alpha I), \\ X_{k+1}(F + \alpha I) = Q - (F^T - \alpha I)X_{k+\frac{1}{2}}. \end{cases} \tag{6}$$

By (6), we can obtain iterative format for X_{k+1},

$$X_{k+1} = (F^T - \alpha I)(F^T + \alpha I)^{-1}X_k(F - \alpha I)(F + \alpha I)^{-1} + 2\alpha(F^T + \alpha I)^{-1}Q(F + \alpha I)^{-1}.$$

Since $F - \alpha I$ is interchangeable with $(F + \alpha I)^{-1}$, we can derive a low-rank ADI (R1-ADI) iterative formula with a single parameter.

$$\begin{cases} V_1 = \sqrt{2\alpha}(F^T + \alpha I)^{-1}C^T, \quad V_1 \in \mathbb{R}^{n\times p}, \\ V_k = [(F^T - \alpha I)(F^T + \alpha I)^{-1}V_{k-1}, V_1], \quad V_k \in \mathbb{R}^{n\times kp}, \\ X_k = V_k V_k^T, \quad X_k \in R^{n\times n}. \end{cases} \tag{7}$$

Next, we consider the ADI iterative method with two parameters for solving the Lyapunov Eq. (2).

$$\begin{cases} (F^T + \alpha I)X_{k+\frac{1}{2}} = Q - X_k(F - \alpha I), \\ X_{k+1}(F + \beta I) = Q - (F^T - \beta I)X_{k+\frac{1}{2}}. \end{cases} \tag{8}$$

Similarly, we can obtain a low-rank ADI (R2-ADI) iterative formula with two parameters.

$$\begin{cases} V_1 = \sqrt{\alpha + \beta}(F^T + \alpha I)^{-1}C^T, \quad V_1 \in \mathbb{R}^{n\times p}, \\ V_k = [(F^T - \beta I)(F^T + \alpha I)^{-1}V_{k-1}, V_1], \quad V_k \in \mathbb{R}^{n\times kp}, \\ W_1 = \sqrt{\alpha + \beta}(F^T + \beta I)^{-1}C^T, \\ W_k = [(F^T + \beta I)^{-1}(F^T - \alpha I)W_{k-1}, W_1], \\ X_k = V_k W_k^T, \quad X_k \in \mathbb{R}^{n\times n}. \end{cases} \tag{9}$$

We apply the GADI iterative framework to solve the Lyapunov Eq. (2). Firstly, the straightening operator is applied and from the Kronecker product, we have

$$(F^T \otimes I + I \otimes F^T)x = q, \quad x = \text{vec}(X), \quad q = \text{vec}(Q). \tag{10}$$

Secondly, by applying the GADI iterative method in Eq. (10), we obtain the following expression.

$$\begin{cases} (\alpha I_{n^2} + I \otimes F^T)x_{k+\frac{1}{2}} = (\alpha I_{n^2} - F^T \otimes I)x_k + q, \\ (\alpha I_{n^2} + F^T \otimes I)x_{k+1} = (F^T \otimes I - (1-\omega)\alpha I_{n^2})x_k + (2-\omega)\alpha x_{k+\frac{1}{2}}. \end{cases} \tag{11}$$

We rewrite Eq. (11) into matrix form as

$$\begin{cases} (\alpha I + F^T)X_{k+\frac{1}{2}} = X_k(\alpha I - F) + Q, \\ X_{k+1}(\alpha I + F) = X_k(F - (1-\omega)\alpha I) + (2-\omega)\alpha X_{k+\frac{1}{2}}. \end{cases} \tag{12}$$

We select the appropriate parameter α to ensure that both matrices $\alpha I + F^T$ and $\alpha I + F$ are invertible. From the first equation of (12), we can obtain

$$X_{k+\frac{1}{2}} = (\alpha I + F^T)^{-1}X_k(\alpha I - F) + (\alpha I + F^T)^{-1}C^T C,$$

we substitute it into the second equation, then

$$\begin{aligned} X_{k+1} &= X_k(F-(1-\omega)\alpha I)(\alpha I+F)^{-1} \\ &\quad + (2-\omega)\alpha X_{k+\frac{1}{2}}(\alpha I+F)^{-1} \\ &= X_k(F-(1-\omega)\alpha I)(\alpha I+F)^{-1} \\ &\quad + (2-\omega)\alpha[(\alpha I+F^T)^{-1}X_k(\alpha I-F) \\ &\quad + (\alpha I+F^T)^{-1}C^TC](\alpha I+F)^{-1} \\ &= X_k(F-(1-\omega)\alpha I)(\alpha I+F)^{-1} \\ &\quad + (2-\omega)\alpha(\alpha I+F^T)^{-1}X_k(\alpha I-F)(\alpha I+F)^{-1} \\ &\quad + (2-\omega)\alpha(\alpha I+F^T)^{-1}C^TC(\alpha I+F)^{-1}. \end{aligned}$$

Taking the initial value $X_0=0$, we have $X_1=(2-\omega)\alpha(\alpha I+F^T)^{-1}C^TC(\alpha I+F)^{-1}$.

Let

$$V_1 = W_1 = \sqrt{(2-\omega)\alpha}(\alpha I+F^T)^{-1}C^T,$$

where $X_1 = V_1W_1^T$. We can also get

$$\begin{aligned} X_2 &= X_1(F-(1-\omega)\alpha I)(\alpha I+F)^{-1} \\ &\quad + (2-\omega)\alpha(\alpha I+F^T)^{-1}X_1(\alpha I-F)(\alpha I+F)^{-1} \\ &\quad + (2-\omega)\alpha(\alpha I+F^T)^{-1}C^TC(\alpha I+F)^{-1} \\ &= V_1W_1^T(F-(1-\omega)\alpha I)(\alpha I+F)^{-1} \\ &\quad + (2-\omega)\alpha(\alpha I+F^T)^{-1}V_1W_1^T(\alpha I-F)(\alpha I+F)^{-1} + V_1W_1^T. \end{aligned}$$

Let

$$\begin{aligned} &V_2 = [V_1, \sqrt{(2-\omega)\alpha}(\alpha I+F^T)^{-1}V_1, V_1], \\ &W_2 = [(\alpha I+F^T)^{-1}(F^T-(1-\omega)\alpha I)W_1, \sqrt{(2-\omega)\alpha}(\alpha I+F^T)^{-1}(\alpha I-F^T)W_1, W_1], \end{aligned}$$

where $X_2 = V_2W_2^T$.

Based on the previous derivation method, we can obtain the R-GADI iterative format for soliving the Lyapunov Eq. (2).

$$\begin{cases} V_1 = \sqrt{(2-\omega)\alpha}(\alpha I+F^T)^{-1}C^T, \\ V_k = [V_{k-1}, \sqrt{(2-\omega)\alpha}(\alpha I+F^T)^{-1}V_{k-1}, V_1], \quad V_k \in R^{n\times(2^k-1)p}, \\ W_1 = \sqrt{(2-\omega)\alpha}(\alpha I+F^T)^{-1}C^T, \\ W_k = [(\alpha I+F^T)^{-1}(F^T-(1-\omega)\alpha I)W_{k-1}, \sqrt{(2-\omega)\alpha}(\alpha I+F^T)^{-1}(\alpha I-F^T)W_{k-1}, W_1], \\ X_k = V_kW_k^T. \end{cases} \tag{13}$$

Next, the R-GADI algorithm is given as Algorithm 1:

2.2 Convergence analysis

Some simple properties of Kronecker product can be easily derived from the Definitions 1 and 2.

Algorithm 1 R-GADI iteration for solving Lyapunov equation 2

```
Require: Matrix $F$, $C$, parameters $\alpha$ and $\omega$, $k_{max}$ and residual limit $\varepsilon = 1 \times 10^{-15}$;
Ensure: Approximation $X \approx V_k W_k^T$ for the solution of the Lyapunov equation $F^T X + XF = Q$.
1: $X_0 = 0$; $Q = C^T C$;
2: for $k = 1, \cdots, k_{max}$ do
3:   if $k = 1$ then
4:     Solve $\frac{1}{\sqrt{(2-\omega)\alpha}}(F^T + \alpha I)V_1 = C^T$ for $V_1$;
5:     $W_1 = V_1$;
6:   else
7:     $V_k = [V_{k-1}, \sqrt{(2-\omega)\alpha}(F^T + \alpha I)^{-1} V_{k-1}, \sqrt{(2-\omega)\alpha}(F^T + \alpha I)^{-1} C^T]$;
8:     $W_k = [(\alpha I + F^T)^{-1}(F^T - (1-\omega)\alpha I)W_{k-1}, \sqrt{(2-\omega)\alpha}(\alpha I + F^T)^{-1}(\alpha I - F^T)W_{k-1}, \sqrt{(2-\omega)\alpha}(F^T + \alpha I)^{-1} C^T]$;
9:   end if
10:  $X_k = V_k W_k^T$;
11:  Compute $\mathrm{Res}(X_k) = \frac{\|F^T X_k + X_k F - Q\|_2}{\|Q\|_2}$;
12:  if $\mathrm{Res}(X_k) < \varepsilon$ then
13:    stop;
14:  end if
15: end for
```

Lemma 1 *(Xu (2011)) Let* $\alpha \in \mathbb{C}$, $A_1 \in \mathbb{C}^{m\times n}$, $B_1 \in \mathbb{C}^{p\times q}$, $X_1 \in \mathbb{C}^{n\times p}$, $C_1 \in \mathbb{C}^{m\times n}$, $D_1 \in \mathbb{C}^{p\times q}$, *then*

(a) $\alpha(A_1 \otimes B_1) = (\alpha A_1) \otimes B_1 = A_1 \otimes (\alpha B_1)$;
(b) $(A_1 \otimes B_1)(C_1 \otimes D_1) = (A_1 C_1) \otimes (B_1 D_1)$;
(c) $(A_1 \otimes B_1)^T = A_1^T \otimes B_1^T$;
(d) $vec(A_1 X_1 B_1) = (B_1^T \otimes A_1)vec(X_1)$;
(e) $\Lambda(I_p \otimes A_1 - B_1^T \otimes I_n) = \{\lambda - \mu : \lambda \in \Lambda(A_1),\ \mu \in \Lambda(B_1)\}$.

Lemma 2 *Let* $\mathscr{R}(X) = F^T X + XF - Q$, *and* $\{X_k\}$ *be an approximate solution sequence of the Lyapunov Eq. (2) generated by GADI iteration (12), and* X *is a symmetric positive definite solution of Eq. (2). Then for any* $k \geq 0$, *we have*

(1) $(\alpha I + F^T)(X_{k+\frac{1}{2}} - X) = (X_k - X)(\alpha I - F)$;
(2) $(\alpha I + F^T)(X_{k+\frac{1}{2}} - X_k) = -\mathscr{R}(X_k)$;
(3) $\mathscr{R}(X_{k+\frac{1}{2}}) = (X_{k+\frac{1}{2}} - X_k)(F - \alpha I)$;
(4) $(X_{k+1} - X)(\alpha I + F) = (X_k - X)F + (1-\omega)\alpha(X_{k+\frac{1}{2}} - X_k) + (X_{k+\frac{1}{2}} - X)$;
(5) $(X_{k+1} - X_{k+\frac{1}{2}})(\alpha I + F) = (X_{k+\frac{1}{2}} - X_k)((1-\omega)\alpha I - F)$.

Proof (1) From the first equation of (12), we have

$$(\alpha I + F^T)(X_{k+\frac{1}{2}} - X) = X_k(\alpha I - F) + Q - (\alpha I + F^T)X.$$

Since $Q - F^T X = XF$, then

$$\begin{aligned}(\alpha I + F^T)(X_{k+\frac{1}{2}} - X) &= XF - X_k F + X_k \alpha - \alpha X \\ &= (X_k - X)(\alpha I - F).\end{aligned}$$

(2)

$$(\alpha I + F^T)(X_{k+\frac{1}{2}} - X_k) = X_k(\alpha I - F) + Q - (\alpha I + F^T)X_k$$

$$\begin{aligned}&= Q - X_k F - F^T X_k\\ &= -\mathscr{R}(X_k).\end{aligned}$$

(3) Due to $F^T X_{k+\frac{1}{2}} - Q = X_k(\alpha I - F) - \alpha X_{k+\frac{1}{2}}$, then

$$\begin{aligned}\mathscr{R}(X_{k+\frac{1}{2}}) &= F^T X_{k+\frac{1}{2}} + X_{k+\frac{1}{2}} F - Q\\ &= X_k(\alpha I - F) - \alpha X_{k+\frac{1}{2}} + X_{k+\frac{1}{2}} F\\ &= (X_{k+\frac{1}{2}} - X_k)(F - \alpha I).\end{aligned}$$

(4) From the second equation of (12), we have

$$\begin{aligned}(X_{k+1} - X)(\alpha I + F) &= X_k(F - (1-\omega)\alpha I) + (2-\omega)\alpha X_{k+\frac{1}{2}} - X(\alpha I + F)\\ &= X_k F - X_k(1-\omega)\alpha I + (1-\omega)\alpha X_{k+\frac{1}{2}} + \alpha X_{k+\frac{1}{2}} - \alpha X - XF\\ &= (X_k - X)F + (1-\omega)\alpha(X_{k+\frac{1}{2}} - X_k) + \alpha(X_{k+\frac{1}{2}} - X).\end{aligned}$$

(5)

$$\begin{aligned}(X_{k+1} - X_{k+\frac{1}{2}})(\alpha I + F) &= X_k(F - (1-\omega)\alpha I) + (2-\omega)\alpha X_{k+\frac{1}{2}}\\ &\quad - X(\alpha I + F) - X_{k+\frac{1}{2}}(\alpha I + F)\\ &= X_k F - X_{k+\frac{1}{2}} F + (1-\omega)\alpha(X_{k+\frac{1}{2}} - X_k)\\ &= (X_{k+\frac{1}{2}} - X_k)((1-\omega)\alpha I - F).\end{aligned}$$

□

From the Lemma 2, we can prove that Theorem 3 holds.

Theorem 3 *Let X be the symmetric positive definite solution of Lyapunov Eq. (2), and let the initial matrix $X_0 = 0$, and parameter $\alpha > 0,\ 0 \le \omega < 2$. Then the matrix sequence $\{X_k\}$ generated by the previous GADI iteration (12) holds the following inequality.*

$$\|X_{k+1} - X\|_2 \le \delta(\alpha)\|X_k - X\|_2 + \eta(\alpha,\omega)\|\mathscr{R}(X_k)\|_2,$$

where

$$\begin{aligned}\delta(\alpha) &= \|F(\alpha I + F)^{-1}\|_2 + \alpha\|(\alpha I + F^T)^{-1}\|_2\|(\alpha I - F)(\alpha I + F)^{-1}\|_2,\\ \eta(\alpha,\omega) &= |1-\omega|\alpha\|(\alpha I + F^T)^{-1}\|_2\|(\alpha I + F)^{-1}\|_2.\end{aligned}$$

Proof From Lemma 2, we can conclude that

$$\begin{aligned}X_{k+1} - X &= [(X_k - X)F + (1-\omega)\alpha(X_{k+\frac{1}{2}} - X_k) + \alpha(X_{k+\frac{1}{2}} - X)](\alpha I + F)^{-1}\\ &= (X_k - X)F(\alpha I + F)^{-1} - (1-\omega)\alpha(\alpha I + F^T)^{-1}\mathscr{R}(X_k)(\alpha I + F)^{-1}\\ &\quad + \alpha(\alpha I + F^T)^{-1}(X_k - X)(\alpha I - F)(\alpha I + F)^{-1}.\end{aligned}$$

Therefore, we can get

$$\begin{aligned}\|X_{k+1} - X\|_2 &\le \|(X_k - X)F(\alpha I + F)^{-1}\|_2 + |1-\omega|\alpha\|(\alpha I + F^T)^{-1}\mathscr{R}(X_k)(\alpha I + F)^{-1}\|_2\\ &\quad + \alpha\|(\alpha I + F^T)^{-1}(X_k - X)(\alpha I - F)(\alpha I + F)^{-1}\|_2\\ &\le \|(X_k - X)\|_2\|F(\alpha I + F)^{-1}\|_2\end{aligned}$$

$$
\begin{aligned}
&+|1-\omega|\alpha\|(\alpha I+F^T)^{-1}\|_2\|\mathscr{R}(X_k)\|_2\|(\alpha I+F)^{-1}\|_2\\
&+\alpha\|(\alpha I+F^T)^{-1}\|_2\|X_k-X\|_2\|(\alpha I-F)(\alpha I+F)^{-1}\|_2\\
=&[\|F(\alpha I+F)^{-1}\|_2+\alpha\|(\alpha I+F^T)^{-1}\|_2\|(\alpha I-F)(\alpha I+F)^{-1}\|_2]\|X_k-X\|_2\\
&+|1-\omega|\alpha\|(\alpha I+F^T)^{-1}\|_2\|(\alpha I+F)^{-1}\|_2\|\mathscr{R}(X_k)\|_2.
\end{aligned}
$$

□

Theorem 4 *Let X be the symmetric positive definite solution of Lyapunov Eq. (2), and let parameters $\alpha > 0$ and $0 \le \omega < 2$, then for any $k = 0, 1, 2, \cdots$, the matrix $V_k W_k^T$ defined by (13) converges to X. This is equivalent to the convergence of the iterative sequence X_k converging to X, where X_k is obtained from (12).*

Proof Here, we only need to prove the convergence of the iterative format (12), since $\alpha > 0$ and A are stable, then $\alpha I_{n^2} + I \otimes F^T$ and $\alpha I_{n^2} + F^T \otimes I$ are non-singular. From the iteration framework (11), we can obtain

$$x_{k+\frac{1}{2}} = (\alpha I_{n^2} + I \otimes F^T)^{-1}(\alpha I_{n^2} - F^T \otimes I)x_k + (\alpha I_{n^2} + I \otimes F^T)^{-1}q,$$

$$
\begin{aligned}
x_{k+1} =& (\alpha I_{n^2} + F^T \otimes I)^{-1}[(F^T \otimes I - (1-\omega)\alpha I_{n^2})\\
&+ (2-\omega)\alpha(\alpha I_{n^2} + I \otimes F^T)^{-1}(\alpha I_{n^2} - F^T \otimes I)]x_k\\
&+ (2-\omega)\alpha(\alpha I_{n^2} + F^T \otimes I)^{-1}(\alpha I_{n^2} + I \otimes F^T)^{-1}q\\
=& (\alpha I_{n^2} + F^T \otimes I)^{-1}(\alpha I_{n^2} + I \otimes F^T)^{-1}[(\alpha^2 I_{n^2}\\
&+ (I \otimes F^T)(F^T \otimes I) - (1-\omega)\alpha(I \otimes F^T + F^T \otimes I)]x_k\\
&+ (2-\omega)\alpha(\alpha I_{n^2} + F^T \otimes I)^{-1}(\alpha I_{n^2} + I \otimes F^T)^{-1}q.
\end{aligned}
$$

Let

$$
\begin{aligned}
M &= I \otimes F^T + F^T \otimes I,\\
G(\alpha,\omega) &= (2-\omega)\alpha(\alpha I_{n^2} + F^T \otimes I)^{-1}(\alpha I_{n^2} + I \otimes F^T)^{-1},\\
T(\alpha,\omega) &= (\alpha I_{n^2} + F^T \otimes I)^{-1}(\alpha I_{n^2} + I \otimes F^T)^{-1}[(\alpha^2 I_{n^2}\\
&\quad +(I \otimes F^T)(F^T \otimes I) - (1-\omega)\alpha M],
\end{aligned}
$$

then

$$x_{k+1} = T(\alpha,\omega)x_k + G(\alpha,\omega)q.$$

Next, we need to prove that for $\alpha > 0,\ 0 \le \omega < 2$, there is $\rho(T(\alpha,\omega)) < 1$. Since

$$2\alpha M = -(\alpha I_{n^2} - I \otimes F^T)(\alpha I_{n^2} - F^T \otimes I) + (\alpha I_{n^2} + I \otimes F^T)(\alpha I_{n^2} + F^T \otimes I),$$

then we can obtain

$$
\begin{aligned}
T(\alpha,\omega) &= (\alpha I_{n^2} + F^T \otimes I)^{-1}(\alpha I_{n^2} + I \otimes F^T)^{-1}[(\alpha^2 I_{n^2} + (I \otimes F^T)(F^T \otimes I) - (1-\omega)\alpha M]\\
&= (\alpha I_{n^2} + F^T \otimes I)^{-1}(\alpha I_{n^2} + I \otimes F^T)^{-1}[(\alpha I_{n^2} - I \otimes F^T)(\alpha I_{n^2} - F^T \otimes I) + \omega\alpha M]\\
&= \frac{1}{2}(\alpha I_{n^2} + F^T \otimes I)^{-1}(\alpha I_{n^2} + I \otimes F^T)^{-1}[2(\alpha I_{n^2} - I \otimes F^T)(\alpha I_{n^2} - F^T \otimes I) + \omega 2\alpha M]\\
&= \frac{1}{2}[(2-\omega)(\alpha I_{n^2} + F^T \otimes I)^{-1}(\alpha I_{n^2} + I \otimes F^T)^{-1}(\alpha I_{n^2} - I \otimes F^T)(\alpha I_{n^2} - F^T \otimes I)\\
&\quad + \omega I_{n^2}]
\end{aligned}
$$

$$= \frac{1}{2}[(2-\omega)T(\alpha) + \omega I_{n^2}],$$

where

$$T(\alpha) = (\alpha I_{n^2} + F^T \otimes I)^{-1}(\alpha I_{n^2} + I \otimes F^T)^{-1}(\alpha I_{n^2} - I \otimes F^T)(\alpha I_{n^2} - F^T \otimes I).$$

Due to $\lambda(T(\alpha,\omega)) = \frac{1}{2}[(2-\omega)\lambda(T(\alpha)) + \omega]$, then we have

$$\rho(T(\alpha,\omega)) \leq \frac{1}{2}[(2-\omega)\rho(T(\alpha)) + \omega].$$

Let

$$\widetilde{T}(\alpha) = (\alpha I_{n^2} + I \otimes F^T)^{-1}(\alpha I_{n^2} - I \otimes F^T)(\alpha I_{n^2} - F^T \otimes I)(\alpha I_{n^2} + F^T \otimes I)^{-1},$$

it can be seen that $T(\alpha)$ is similar to $\widetilde{T}(\alpha)$ through the matrix $\alpha I_{n^2} + F^T \otimes I$. Therefore, we get

$$\begin{aligned}\rho(T(\alpha)) &\leq \| (\alpha I_{n^2} + I \otimes F^T)^{-1}(\alpha I_{n^2} - I \otimes F^T)(\alpha I_{n^2} - F^T \otimes I)(\alpha I_{n^2} + F^T \otimes I)^{-1} \|_2 \\ &\leq \| (\alpha I_{n^2} + I \otimes F^T)^{-1}(\alpha I_{n^2} - I \otimes F^T) \|_2 \| (\alpha I_{n^2} - F^T \otimes I)(\alpha I_{n^2} + F^T \otimes I)^{-1} \|_2 \\ &= \| F_L \|_2 \| F_R \|_2,\end{aligned}$$

where

$$F_L = (\alpha I_{n^2} + I \otimes F^T)^{-1}(\alpha I_{n^2} - I \otimes F^T), \quad F_R = (\alpha I_{n^2} - F^T \otimes I)(\alpha I_{n^2} + F^T \otimes I)^{-1}.$$

For $x \in \mathbb{R}^{n^2 \times 1}$, we get

$$\begin{aligned}\| F_L \|_2^2 &= \max_{\|x\|_2=1} \frac{\| (\alpha I_{n^2} - I \otimes F^T)x \|_2^2}{\| (\alpha I_{n^2} + I \otimes F^T)x \|_2^2} \\ &= \max_{\|x\|_2=1} \frac{\| (I \otimes F^T)x \|_2^2 - \alpha x^T(I \otimes F + I \otimes F^T)x + \alpha^2}{\| (I \otimes F^T)x \|_2^2 + \alpha x^T(I \otimes F + I \otimes F^T)x + \alpha^2} \\ &\leq \max_{\|x\|_2=1} \frac{\| (I \otimes F^T)x \|_2^2 - 2\alpha \min Re(\lambda(I \otimes F)) + \alpha^2}{\| (I \otimes F^T)x \|_2^2 + 2\alpha \min Re(\lambda(I \otimes F)) + \alpha^2} \\ &\leq \frac{\| I \otimes F^T \|_2^2 - 2\alpha \min Re(\lambda(I \otimes F)) + \alpha^2}{\| I \otimes F^T \|_2^2 + 2\alpha \min Re(\lambda(I \otimes F)) + \alpha^2} \\ &= \frac{\| I \otimes F^T \|_2^2 - 2\alpha \min Re(\lambda(F)) + \alpha^2}{\| I \otimes F^T \|_2^2 + 2\alpha \min Re(\lambda(F)) + \alpha^2}.\end{aligned}$$

Similarly, we have a conclusion

$$\| F_R \|_2^2 \leq \frac{\| F^T \otimes I \|_2^2 - 2\alpha \min Re(\lambda(F)) + \alpha^2}{\| F^T \otimes I \|_2^2 + 2\alpha \min Re(\lambda(F)) + \alpha^2},$$

since A is stable and $\alpha > 0$, then

$$\| F_L \|_2 < 1, \quad \| F_R \|_2 < 1.$$

Therefore, $\rho(T(\alpha)) < 1$ and $\rho(T(\alpha,\omega)) \leq \frac{1}{2}[(2-\omega)\rho(T(\alpha)) + \omega] < 1$. □

2.3 Selection of parameters

In this section, we first compare the convergence rates of the GADI method and the ADI method, where $\rho(T(\alpha,\ \omega))$ and $\rho(T(\alpha))$ used here are defined in the previous proof of Theorem 3. Afterwards, we will provide a more practical method to select parameters.

Theorem 5 *Assuming that the eigenvalues of matrix F have positive real parts, define*

$$\rho(T(\alpha,\omega)) = |\xi| = |a+bi|, \quad \rho(T(\alpha)) = |\eta| = |c+di|,$$

(i) *if $|\eta|^2 \le c$, then*

$$\rho(T(\alpha)) < \rho(T(\alpha,\omega)) < 1;$$

(ii) *if $|\eta|^2 > c$ and $0 < \omega < \frac{4(|\eta_k|^2 - c_k)}{(1-c_k)^2 + d_k^2} < 2$, we have*

$$\rho(T(\alpha,\omega)) < \rho(T(\alpha)) < 1.$$

Theorem 6 *Let $\sigma_j(A)$ and $\lambda_j(F)$ represent the singular and eigenvalues of the coefficient matrix F of the Lyapunov Eq. (2), respectively, with $j = 1, 2, \cdots, n$. Let*

$$\mu = \max_j \sigma_j(F), \quad \nu = \min_j Re(\lambda_j(F)),$$

then we can obtain the optimal parameters α^* as follows

$$\alpha^* = arg \min_\alpha \frac{\mu^2 - 2\alpha\nu + \alpha^2}{\mu^2 + 2\alpha\nu + \alpha^2} = \mu.$$

The proof of Theorems 5 and 6 is similar to the proof process of Theorems 2.7 and 2.9 in Li et al. (2022), and we will not delve into the details here.

For the parameter ω, since $0 \le \omega < 2$, special values 0 and 1 can be selected for validation first. The general method is to select the appropriate parameter ω by analyzing residuals in numerical examples.

2.4 Complexity analysis

According to the iterative scheme (11), this method is primarily used for efficiently solving large sparse Lyapunov equations. The approach involves utilizing Cholesky factorization and representing the solution in the form of low-rank factors. Before starting Algorithm 1, we need to determine the values of parameters α and ω. Finding suitable parameter values can be challenging as the selection of parameters significantly impacts the iterative results and convergence speed. In Algorithm 1, a stopping criterion needs to be set to compute the maximum number of iterations. One approach is to stop the iteration when the change in the approximate solution is small, i.e., $\| X_k - X_{k-1} \| < \varepsilon$, where

$$\| X_k - X_{k-1} \| = \| V_k W_k^T - V_{k-1} W_{k-1}^T \| .$$

However, computing the norm of the approximate solution on high-dimensional matrices can be computationally expensive. We adopt an alternative approach by measuring the relative residual of the approximate solution X_k generated by the low-rank GADI iteration (12) at the k-th step. The relative residual $\mathrm{Res}(X_k)$ is defined as follows:

$$\mathrm{Res}(X_k) = \frac{\|F^T X_k + X_k F - Q\|_2}{\|Q\|_2}.$$

Therefore, with a given accuracy requirement ε, we stop the iteration when $\mathrm{Res}(X_k) < \varepsilon$.

Next, we calculate the computational complexity of Algorithm 1. During the iteration process, the number of columns in matrices V_k and W_k increases with the number of iterations, i.e., $V_k,\ W_k \in \mathbb{R}^{n\times(2^k-1)p}$, where $p \ll n$. Firstly, in the first iteration, the computational cost of obtaining V_1 is $2n^2p$, and at this point, $W_1 = V_1$. Then, in the k-th iteration, the computational cost of obtaining V_k is $2n^2(2^{k-1}-1)p$, and the computational cost of obtaining W_k is $4n^3 + 2n^2(2^{k-1}-1)p + 2n^2p$. Therefore, the total computational complexity of this algorithm is $4n^3 + 4n^2 2^{k-1}p$.

2.5 Numerical experiments

In this section, we use two examples of the linear system to show the numerical feasibility and effectiveness of the R-GADI algorithms. The Lyapunov equation is widely used to determine the stability of linear systems. By solving the Lyapunov equation, such a function can be found to determine the stability of the system. In our numerical experiment, set the iteration tolerance $\varepsilon = 1 \times 10^{-15}$. The whole process is performed on a computer with Intel Core 1.00GHz CPU, 8.00GB RAM, and MATLAB R2018a. IT represents the number of iteration steps, and CPU by the computing time. Denote the numerical symmetric solution as X, the finally relative residual as

$$\mathrm{Res}(X) = \frac{\|F^TX + XF - Q\|_2}{\|Q\|_2}.$$

Example 2.5.1 Consider the linear system of the form $\dot{x} = Fx$, if $V(z) = z^TXz$,then

$$\dot{V}(z) = (Fz)^TXz + z^TX(Fz) = z^TQz,$$

where

$$F = \begin{pmatrix} 5 & 0.3 & 0 & \cdots & 0 \\ 0.2 & 5 & 0.3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0.2 & 5 & 0.3 \\ 0 & \cdots & 0 & 0.2 & 5 \end{pmatrix}_{n\times n},$$

$$C = \begin{pmatrix} 1, & 1, & \cdots, & 1, & 1 \end{pmatrix}_{1\times n}, \quad Q = C^TC.$$

We set the parameters $\omega = 0.015$ and α to be $\sqrt{\lambda_{max}(F)\lambda_{min}(F)}$ and $\max\sigma(F)$, respectively. We compare the numerical results for different matrix dimensions and different choices of α, which are listed in Table 1 below. From the data in the table, it can be observed that during the iteration process, when the matrix dimensions are the same and α is chosen as the maximum singular value of matrix F, the total iteration time is shorter, resulting in better performance.

Next, we set the parameters $\alpha = \max\sigma(F),\ \omega = 0.015$. In the case of matrix dimensions being multiplied, we solve the problem using both the GADI method and the R-GADI method. Table 2 presents the numerical results for relative residual, iteration count, and CPU time. In terms of iteration count and time, the R-GADI method is relatively more efficient. For instance, when $n = 1024$, Fig. 1 shows the iteration count and relative residual obtained by applying these two methods iteratively, while Fig. 2 displays the computation time required by each method. It is evident from the figures that the R-GADI method is effective.

Table 1 Numerical results for Example 2.5.1

n	Algorithm	α	Res	IT	CPU
128	R-GADI	$\sqrt{\lambda_{max}(F)\lambda_{min}(F)}$	4.2129e−16	8	0.06s
128	R-GADI	$\max\sigma(F)$	4.5781e−16	8	0.05s
256	R-GADI	$\sqrt{\lambda_{max}(F)\lambda_{min}(F)}$	4.3544e−16	8	0.24s
256	R-GADI	$\max\sigma(F)$	4.3033e−16	8	0.22s
512	R-GADI	$\sqrt{\lambda_{max}(F)\lambda_{min}(F)}$	2.3044e−16	8	1.34s
512	R-GADI	$\max\sigma(F)$	5.7213e−16	7	1.27s
1024	R-GADI	$\sqrt{\lambda_{max}(F)\lambda_{min}(F)}$	4.3802e−16	8	28.64s
1024	R-GADI	$\max\sigma(F)$	9.9827e−16	7	15.51s
2048	R-GADI	$\sqrt{\lambda_{max}(F)\lambda_{min}(F)}$	9.4348e−15	7	219.72s
2048	R-GADI	$\max\sigma(F)$	1.1144e−15	7	191.26s
4096	R-GADI	$\sqrt{\lambda_{max}(F)\lambda_{min}(F)}$	4.4282e−16	8	4408.41s
4096	R-GADI	$\max\sigma(F)$	8.887e−16	7	1223.03s

Table 2 Numerical results for Example 2.5.1

n	Algorithm	Res	IT	CPU
128	GADI	9.5268e−16	8	0.07s
128	R-GADI	4.5781e−16	8	0.05s
256	GADI	2.1253e−16	8	0.25s
256	R-GADI	4.3033e−16	8	0.22s
512	GADI	4.2985e−16	8	1.49s
512	R-GADI	5.7213e−16	7	1.27s
1024	GADI	5.4699e−16	7	30.95s
1024	R-GADI	9.9827e−16	7	15.51s
2048	GADI	9.8707e−15	7	380.3s
2048	R-GADI	1.1144e−15	7	191.26s
4096	GADI	9.6536e−15	7	2568.69s
4096	R-GADI	8.887e−16	7	1223.03s

Example 2.5.2 Consider the linear system of the form $\dot{x} = Fx$, if $V(z) = z^T Xz$, then

$$\dot{V}(z) = (Fz)^T Xz + z^T X(Fz) = z^T Qz,$$

where

$$F = \begin{pmatrix} 9 & 3 & 0 & \cdots & 0 \\ -2 & 9 & 3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -2 & 9 & 3 \\ 0 & \cdots & 0 & -2 & 9 \end{pmatrix}_{n\times n},$$

$$C = \begin{pmatrix} 1, & 1, & \cdots, & 1, & 1 \end{pmatrix}_{1\times n}, \quad Q = C^T C.$$

We utilize the GADI, R1-ADI, R2-ADI, and R-GADI methods to solve the problem. Table 3 presents the relative residual, iteration count, and time obtained using these four

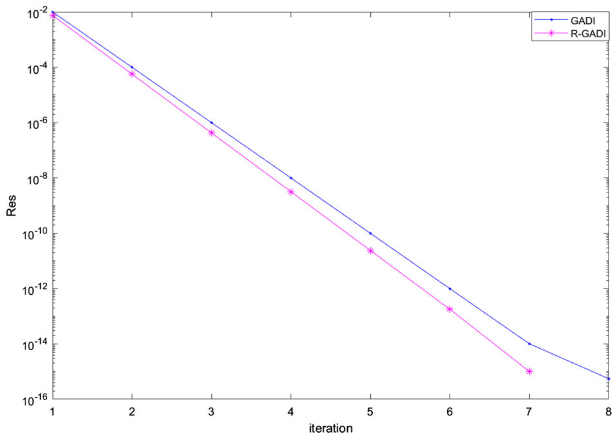

Fig. 1 The residual curve of Example 2.5.1 $n = 1024$

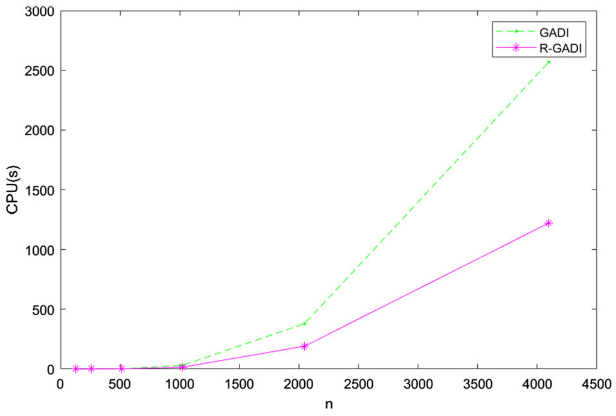

Fig. 2 The time curve of Example 2.5.1

iterative methods as the matrix dimension increases. By comparing these data, we can observe the effectiveness of the R-GADI method in solving the problem. Additionally, Fig. 3 illustrates the correspondence between iteration count and relative residual for these four iterative methods when $n = 1024$. Figure 4 displays the iteration time required by each method, it is evident that the R-GADI method achieves better results in solving the problem.

From the above Fig. 3, we can see that in the initial iteration phase, these four iterative methods exhibit relatively fast convergence. However, as the iterations progress, the convergence speed slows down. Upon reaching a certain number of iterations, it becomes evident that the R-GADI method demonstrates better convergence speed and requires the least amount of time.

Table 3 Numerical results for Example 2.5.2

n	Algorithm	Res	IT	CPU
128	GADI	2.1884e−16	14	0.15s
128	R1-ADI	4.8137e−16	14	0.11s
128	R2-ADI	3.0187e−16	12	0.12s
128	R-GADI	6.2135e−16	10	0.07s
256	GADI	8.0851e−16	13	0.65s
256	R1-ADI	2.5974e−16	14	0.47s
256	R2-ADI	7.9983e−16	11	0.43s
256	R-GADI	3.1158e−16	10	0.29s
512	GADI	4.4317e−16	12	4.22s
512	R1-ADI	2.6518e−16	13	2.29s
512	R2-ADI	4.2145e−16	11	2.27s
512	R-GADI	2.0157e−16	10	1.48s
1024	GADI	1.7196e−16	12	45.34s
1024	R1-ADI	3.5374e−16	12	26.63s
1024	R2-ADI	8.1825e−16	10	26.86s
1024	R-GADI	2.2622e−16	10	15.21s
2048	GADI	6.0359e−16	12	490.81s
2048	R1-ADI	1.2585e−15	12	455.37s
2048	R2-ADI	1.0059e−15	10	450.4s
2048	R-GADI	6.6674e−16	9	234.92s
4096	GADI	2.8412e−16	12	3998.2s
4096	R1-ADI	3.2722e−16	12	3225.4s
4096	R2-ADI	4.3943e−15	9	3025.7s
4096	R-GADI	2.983e−16	9	1495.60s

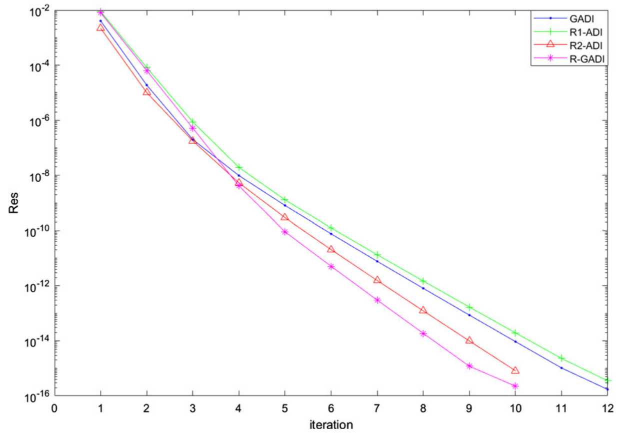

Fig. 3 The residual curve of Example 2.5.2 $n = 1024$

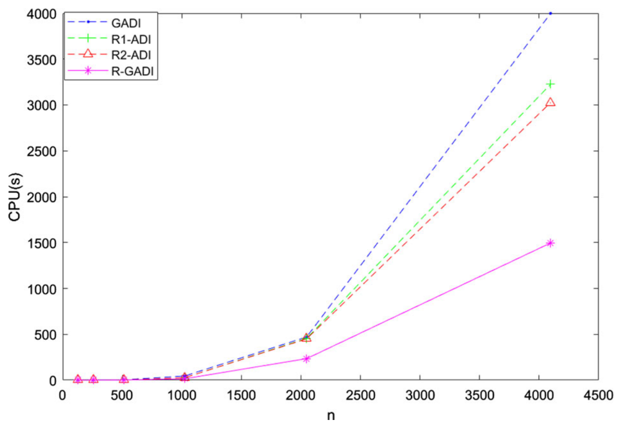

Fig. 4 The time curve of Example 2.5.2

3 Low rank GADI for solving Riccati equation

3.1 Derivation of iterative format

In this section, we investigate the conclusions related to solving the continuous algebraic Riccati Eq. (1) using the GADI method. The algebraic Riccati equation has its origins in numerical problems in control theory and finds wide applications, particularly in the design of quadratic optimal control. For certain linear quadratic optimization control problems, the problem can eventually be transformed into solving the algebraic Riccati equation for a stable solution. Generally, the solution of the algebraic Riccati Eq. (1) is not unique. Therefore, we first provide a sufficient condition for the existence of a unique solution.

Lemma 3 (Xu (2011)) *If the matrix pair (A, B) in the algebraic Riccati Eq. (1) is stabilizable, meaning that for any λ such that $Re\lambda \geq 0$, the matrix $[A - \lambda I \;\; B]$ has full row rank, and if the matrix pair $(C, \; A)$ is detectable, meaning that for any λ and x such that $Re\lambda \geq 0$ and $Ax = \lambda x$, we have $Cx \neq 0$, then Eq. (1) has a unique positive semi-definite solution, and this solution is stable.*

Firstly, we consider the Newton iteration method:

- Define mapping $F : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ is

$$F(X) = A^T X + XA + Q - XGX, \quad X \in \mathbb{R}^{n\times n},$$

then we have

$$\begin{aligned} F(X+E) &= A^T(X+E) + (X+E)A + Q - (X+E)G(X+E) \\ &= A^T X + XA - XGX + Q + A^T E + EA - EGX - XGE - EGE \\ &= F(X) + L(X) + O(\|E\|_2^2) \; (E \to 0), \end{aligned}$$

where

$$\begin{aligned} L(E) &= A^T E + EA - EGX - XGE \\ &= (A - GX)^T E + E(A - GX), \end{aligned}$$

it is a linear operator. From the definition of Fréchet differentiability, it follows that for any $X \in \mathbb{R}^{n\times n}$, F is Fréchet differentiable at X. There is

$$F'(X)(E) = (A - GX)^* E + E(A - GX),\ E \in \mathbb{R}^{n\times n}.$$

Thus, by applying Newton's method $F'(X_k)(E_k) = -F(X_k)$, we can get the iterative format of the Newton iteration method for solving equation (1).

(a) Seeking E_k that satisfies the equation

$$(A - GX_k)^T E_k + E_k(A - GX_k) = -F(X_k). \tag{14}$$

(b) Calculate $X_{k+1} = X_k + E_k$, where $X_0 \in \mathbb{R}^{n\times n}$ is the initial matrix. Rearranging Eq. (14) and using $X_{k+1} = X_k + E_k$, then we can obtain the equation satisfied by X_{k+1}

$$(A - GX_k)^T X_{k+1} + X_{k+1}(A - GX_k) + X_k G X_k + Q = 0. \tag{15}$$

- Next, we apply the Kleinman-Newton method (Kleinman 1968) with an initial feedback matrix $K_0 = 0$. Let $K_k = X_k B$ to obtain another equivalent form of Eq. (15).

$$(A - BK_k^T)^T X_{k+1} + X_{k+1}(A - BK_k^T) = -K_k K_k^T - C^T C, \tag{16}$$

with $A_k = BK_k^T - A,\ M_k = [K_k,\ C^T]$. Therefore, Eq. (16) can be rewritten as the following Lyapunov equation form

$$A_k^T X_{k+1} + X_{k+1} A_k = M_k M_k^T. \tag{17}$$

- In addition, we apply the GADI framework to solve the Lyapunov Eq. (17) and obtain an iterative format

$$\begin{cases} (\alpha_k I + A_k^T) X_{k+1}^{(l+\frac{1}{2})} = X_{k+1}^{(l)}(\alpha_k I - A_k) + Q_k, \\ X_{k+1}^{(l+1)}(\alpha_k I + A_k) = X_{k+1}^{(l)}(A_k - (1-\omega_k)\alpha_k I) + (2-\omega_k)\alpha_k X_{k+1}^{(l+\frac{1}{2})}, \end{cases} \tag{18}$$

where

$$Q_k = M_k M_k^T,\ \ l = 0, 1, \cdots, s,$$
$$X_{k+1}^{(0)} = X_k,\ \ X_{k+1} = X_{k+1}^{(s)},\ \ \alpha_k > 0,\ \ 0 \le \omega_k < 2.$$

Next, we further manipulate the iterative scheme (18) and obtain the following expression under the assumption that $\alpha_k I + A_k^T$ and $\alpha_k I + A_k$ are non-singular.

$$\begin{aligned} X_{k+1}^{(l+1)} &= X_{k+1}^{(l)}(A_k - (1-\omega_k)\alpha_k I)(\alpha_k I + A_k)^{-1} \\ &\quad + (2-\omega_k)\alpha_k(\alpha_k I + A_k^T)^{-1} X_{k+1}^{(l)}(\alpha_k I - A_k)(\alpha_k I + A_k)^{-1} \\ &\quad + (2-\omega_k)\alpha_k(\alpha_k I + A_k^T)^{-1} Q_k (\alpha_k I + A_k)^{-1}. \end{aligned}$$

Due to $X_{k+1}^{(0)} = X_k$, then we have

$$\begin{aligned} X_{k+1}^{(1)} &= X_k(A_k - (1-\omega_k)\alpha_k I)(\alpha_k I + A_k)^{-1} \\ &\quad + (2-\omega_k)\alpha_k(\alpha_k I + A_k^T)^{-1} X_k(\alpha_k I - A_k)(\alpha_k I + A_k)^{-1} \\ &\quad + (2-\omega_k)\alpha_k(\alpha_k I + A_k^T)^{-1} Q_k (\alpha_k I + A_k)^{-1}. \end{aligned}$$

Let the initial value of iteration $X_0 = 0$, we get

$$X_1^{(1)} = (2-\omega_0)\alpha_0(\alpha_0 I + A_0^T)^{-1}Q_0(\alpha_0 I + A_0)^{-1} = V_1^{(1)}(W_1^{(1)})^T,$$

where

$$\begin{aligned} V_1^{(1)} &= \sqrt{(2-\omega_0)\alpha_0}(\alpha_0 I + A_0^T)^{-1}M_0 \\ &= \sqrt{(2-\omega_0)\alpha_0}(\alpha_0 I - A^T)^{-1}C^T, \\ W_1^{(1)} &= V_1^{(1)}. \end{aligned}$$

Therefore, we provide a low rank Kleinman Newton GADI iterative scheme for the corresponding Riccati equation.

$$\begin{cases} A_k = BK_k^T - A,\ M_k = [K_k,\ C^T],\ \ Q_k = M_k M_k^T, \\ V_1^{(1)} = \sqrt{(2-\omega_0)\alpha_0}(\alpha_0 I - A^T)^{-1}C^T, \\ V_{k+1}^{(1)} = \sqrt{(2-\omega_k)\alpha_k}(\alpha_k I + A_k^T)^{-1}M_k, \\ V_{k+1}^{(l)} = [V_{k+1}^{(l-1)}, \sqrt{(2-\omega_k)\alpha_k}(\alpha_k I + A_k^T)^{-1}V_{k+1}^{(l-1)}, V_{k+1}^{(1)}], \\ W_1^{(1)} = V_1^{(1)}, \\ W_{k+1}^{(1)} = \sqrt{(2-\omega_k)\alpha_k}(\alpha_k I + A_k^T)^{-1}M_k, \\ W_{k+1}^{(l)} = [(\alpha_k I + A_k^T)^{-1}(A_k^T - (1-\omega_k)\alpha_k I)W_{k+1}^{(l-1)}, \\ \qquad \sqrt{(2-\omega_k)\alpha_k}(\alpha_k I + A_k^T)^{-1}(\alpha_k I - A_k^T)W_{k+1}^{(l-1)}, W_{k+1}^{(1)}], \\ X_{k+1}^{(l)} = V_{k+1}^{(l)}(W_{k+1}^{(l)})^T, \\ K_{k+1} = V_{k+1}W_{k+1}^T B. \end{cases} \tag{19}$$

Algorithm 2 Low rank GADI iteration for solving Lyapunov equation 17

```
Require: Matrix $A,\ B,\ C$, initial feedback matrix $K_0$ makes $BK_0^T - A$ is stable, $k_{max}$;
Ensure: Approximation $X_{k+1} \approx V_{k+1}W_{k+1}^T$ for the solution of the equation $A_k^T X_{k+1} + X_{k+1}A_k = Q_k$.
1: for $k = 1, \cdots, k_{max}$ do
2:   $A_k = BK_k^T - A$;
3:   $M_k = [K_k,\ C^T]$; $Q_k = M_k M_k^T$;
4:   $V_k^{(1)} = \sqrt{(2-\omega_{k-1})\alpha_{k-1}}(\alpha_{k-1}I + A_{k-1}^T)^{-1}M_{k-1}$;
5:   $W_k^{(1)} = \sqrt{(2-\omega_{k-1})\alpha_{k-1}}(\alpha_{k-1}I + A_{k-1}^T)^{-1}M_{k-1}$;
6:   for $l = 2, 3, \cdots, i_k^{max}$ do
7:     $V_k^{(l)} = [V_k^{(l-1)},\ \sqrt{(2-\omega_{k-1})\alpha_{k-1}}(\alpha_{k-1}I + A_{k-1}^T)^{-1}V_k^{(l-1)},\ V_k^{(1)}]$;
8:     $W_k^{(l)} = [(\alpha_{k-1}I + A_{k-1}^T)^{-1}(A_{k-1}^T - (1-\omega_{k-1})\alpha_{k-1}I)W_k^{(l-1)},\ \sqrt{(2-\omega_{k-1})\alpha_{k-1}}(\alpha_{k-1}I + A_{k-1}^T)^{-1}(\alpha_{k-1}I - A_{k-1}^T)W_k^{(l-1)},\ W_k^{(1)}]$;
9:     $X_k^{(l)} = V_k^{(l)}(W_k^{(l)})^T$;
10:  end for
11:  $V_{k+1} = V_{k+1}^{(i_k^{max})}$, $W_{k+1} = W_{k+1}^{(i_k^{max})}$;
12:  $K_{k+1} = V_{k+1}W_{k+1}^T B$;
13: end for
```

3.2 Convergence analysis

Next, we provide the convergence results of using the Newton method to solve Eq. (15).

Theorem 7 (Xu (2011)) *For the algebraic Riccati Eq. (1), assuming $(A,\ G)$ is stable and $(Q,\ A)$ is detectable, and selecting a symmetric positive semi-definite matrix X_0 such the $A-GX_0$ is stable, then the matrix sequence $\{X_k\}$ generated by the Newton iteration converges quadratically to the unique positive semi-definite solution X of (1). In other words, there exists a constant $\delta > 0$ independent of k, such that for all positive integers*

$$\|X_k - X\|_2 \leq \delta \|X_{k-1} - X\|_2^2.$$

Furthermore, the iteration sequence $\{X_k\}$ exhibits monotonic convergence, i.e.,

$$0 \leq X \leq \cdots \leq X_{k+1} \leq X_k \leq \cdots \leq X_1.$$

Let

$$R(X_{k+1}) = A_k^T X_{k+1} + X_{k+1} A_k - M_k M_k^T$$

be the residual matrix of the Lyapunov Eq. (17), then we have the following proposition.

Proposition 8 *Let X_k be the k-th step iteration generated by the Kleinman-Newton method as described above, then*

$$F(X_{k+1}) = -R(X_{k+1}) - (X_k B - X_{k+1} B)(X_k B - X_{k+1} B)^T.$$

Proof From the residual equation defined above, we can obtain that

$$\begin{aligned}
R(X_{k+1}) &= A_k^T X_{k+1} + X_{k+1} A_k - M_k M_k^T \\
&= (K_k B^T - A^T) X_{k+1} + X_{k+1}(B K_k^T - A) - K_k K_k^T - C^T C \\
&= K_k B^T X_{k+1} - A^T X_{k+1} + X_{k+1} B K_k^T - X_{k+1} A - K_k K_k^T - C^T C \\
&= -F(X_{k+1}) + K_k B^T X_{k+1} + X_{k+1} B K_k^T - X_{k+1} B B^T X_{k+1} - K_k K_k^T - C^T C \\
&= -F(X_{k+1}) + X_k B B^T X_{k+1} + X_{k+1} B B^T X_k \\
&\quad - X_{k+1} B B^T X_{k+1} - X_k B B^T X_k - C^T C \\
&= -F(X_{k+1}) - (X_k - X_{k+1}) B B^T (X_k - X_{k+1}) \\
&= -F(X_{k+1}) - (X_k B - X_{k+1} B)(X_k B - X_{k+1} B)^T.
\end{aligned}$$

Algorithm 3 A low rank Kleinman-Newton GADI method for solving Riccati equation 1

```
Require: Matrix $A, B, C$, initial feedback matrix $K_0$, loop variables $k = 1, \cdots, i_k^{max}$, and the iteration
    tolerance $\varepsilon = 1 \times 10^{-12}$;
Ensure: $X = X_{k+1}^{(i_k^{max})}$ make $V_{k+1}^{(i_k^{max})}(W_{k+1}^{(i_k^{max})})^T \approx X_{k+1}^{(i_k^{max})}$, where $X$ is the solution of equation $A^T X +$
    $XA - XBB^T X + C^T C = 0$.
1: for $k = 1, \cdots, i_k^{max}$ do
2:     $A_k = BK_k^T - A$;
3:     $M_k = [K_k,\ C^T]$; $Q_k = M_k M_k^T$;
4:     Using Algorithm 2 for low rank GADI to find $X_{k+1} \approx V_{k+1} W_{k+1}^T$ is the approximate solution of the
       Lyapunov equation $A_k^T X_{k+1} + X_{k+1} A_k = Q_k$.
5:     $K_{k+1} = V_{k+1} W_{k+1}^T B$;
6:     Compute $\mathrm{Res}(V_{k+1} W_{k+1}^T) = \frac{\|A_k^T V_{k+1} W_{k+1}^T + V_{k+1} W_{k+1}^T A_k - Q_k\|_2}{\|Q_k\|_2}$;
7:     if $\mathrm{Res}(V_{k+1} W_{k+1}^T) < \varepsilon$ then
8:         stop;
9:     end if
10: end for
```

Therefore, from Proposition 8 we can directly prove

$$\begin{aligned}\|F(X_{k+1})\|_2 &\le \|R(X_{k+1})\|_2 + \|(X_kB - X_{k+1}B)(X_kB - X_{k+1}B)^T\|_2 \\ &\le \|R(X_{k+1})\|_2 + \|X_kB - X_{k+1}B\|_2^2.\end{aligned}$$

□

Theorem 9 *Let X_{k+1} is a symmetric positive semi-definite of the Eq. (17), and let $(A_k,\ M_k)$ is stable, and parameter $\alpha > 0,\ 0 \le \omega < 2$, then for all $k = 0, 1, \cdots$, the low-rank form defined in Eq. (19) $V_{k+1}^{(l)}(W_{k+1}^{(l)})^T$ converges to $V_{k+1}W_{k+1}^T$, which is equivalent to the iteration sequence defined in Eq. (18) $\{X_{k+1}^{(l)}\}_{l=0}^{\infty}$ converges to X_{k+1}.*

Proof Here, we only need to prove the convergence of the GADI iteration format in Eq. (18). Applying the flattening operator to Eq. (18) yields

$$\begin{cases}(\alpha_k I_{n^2} + I \otimes A_k^T)\mathrm{vec}(X_{k+1}^{(l+\frac{1}{2})}) = (\alpha_k I_{n^2} - A_k^T \otimes I)\mathrm{vec}(X_{k+1}^{(l)}) + \mathrm{vec}(Q_k), \\ (\alpha_k I_{n^2} + A_k^T \otimes I)\mathrm{vec}(X_{k+1}^{(l+1)}) = (A_{k+1}^T \otimes I - (1-\omega_k)\alpha_k I_{n^2})\mathrm{vec}(X_{k+1}^{(l)}) \\ \qquad\qquad +(2-\omega_k)\alpha_k \mathrm{vec}(X_{k+1}^{(l+\frac{1}{2})}),\end{cases} \tag{20}$$

where $x = \mathrm{vec}(X),\quad q_k = \mathrm{vec}(Q_k)$, then the equivalent iteration format is obtained as follows

$$\begin{cases}(\alpha_k I_{n^2} + I \otimes A_k^T)x_{k+1}^{(l+\frac{1}{2})} = (\alpha_k I_{n^2} - A_k^T \otimes I)x_{k+1}^{(l)} + q_k, \\ (\alpha_k I_{n^2} + A_k^T \otimes I)x_{k+1}^{(l+1)} = (A_k^T \otimes I - (1-\omega_k)\alpha_k I_{n^2})x_{k+1}^{(l)} + (2-\omega_k)\alpha_k x_{k+1}^{(l+\frac{1}{2})}.\end{cases} \tag{21}$$

Next, the proof of this theorem can be referenced to the proof method of Theorem 2.6 in Li et al. (2022). □

3.3 Complexity analysis

Here, we adopt the Kleinman–Newton iteration to transform the Riccati equation into the Lyapunov Eq. (17) and then use the low-rank GADI method for iterative solving. Each iteration in the algorithm requires solving a Lyapunov equation, resulting in significant computational complexity. The Kleinman–Newton iteration method is an improved version of the Newton iteration method, as the right-hand side of Eq. (14) is usually indefinite with a full rank matrix, while the method employed in this paper relies heavily on the low-rank structure of the right-hand side. The rank of the matrix selected in equation is at most $m + p$, and we use the low-rank method to compute the approximate solution, making the Kleinman–Newton iteration method more suitable. We use the product of two low-rank matrices $V_k^{(l)}(W_k^{(l)})^T$ to replace $X_k^{(l)}$. Next, we provide the stopping criterion used in the algorithm.

Firstly, we define the residual matrix of the Riccati equation (17) as follows:

$$F(X_k) = A^T X_k + X_k A - X_k G X_k + Q.$$

During the iteration process, in order to reflect its relative approximation level, the relative residual

$$\mathrm{Res}(X_k) = \frac{\|A^T X_k + X_k A - X_k G X_k + Q\|_2}{\|Q\|_2}$$

is commonly used as a stopping criterion for iteration. However, in Newton iterations, forming the residual matrix $R(V_k W_k^T)$ requires a significant amount of memory. If the number of columns in V_k is much smaller than the number of rows, the relative residual can be effectively used as the stopping criterion.

Additionally, we can observe the variation of the feedback matrix K_{k+1}. We use the criterion

$$R_s = \frac{\|K_{k+1} - K_k\|_2}{\|K_{k+1}\|_2} < \varepsilon,$$

where $K_{k+1} = V_{k+1} W_{k+1}^T B$ and ε is a very small positive number. This criterion is computationally efficient since $K \in \mathbb{R}^{n\times m}$ and $m \ll n$.

The direct iteration method has a memory requirement of $O(n^2)$, and a computational complexity of $O(n^3)$. First, the computational cost of $X_{k+1}^{l+\frac{1}{2}}$ is $4n^2(2n-1)+3n^2$, and the cost of X_{k+1}^{l+1} is $4n^2(2n-1)+4n^2$. Therefore, the total computational cost is $16n^3-n^2$. By comparing Algorithm 2 and Algorithm 3, we can see that they have smaller memory requirements. If A is a sparse matrix, the memory requirement can reach $O(n)$. Next, we calculate their computational complexity. Since $A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{p\times n}$, during the iteration process, the column numbers of matrices V_k and W_k ncrease with the number of iterations. Here, we have $A_k \in \mathbb{R}^{n\times n},\ M_k \in \mathbb{R}^{n\times(m+p)}$. The computational cost of calculating $V_1^{(1)}$ is $2n^2p$, where $W_1^{(1)} = V_1^{(1)}$. The cost of calculating $V_k^{(1)}$ is $2n^2(m+p)$, and the cost of $V_k^{(l)}$ is $n^2(2^l-2)(m+p)$, while the cost of $W_k^{(l)}$ is $4n^3+2n^2(2^l-2)(m+p)$. Therefore, the total computational cost is $4n^3+n^2(3\cdot 2^l(m+p)-2(2m+p))$.

3.4 Numerical experiments

In this section, we use two examples of the quadratic optimal control to show the numerical feasibility and effectiveness of the Kleinman-Newton-RGADI algorithms. The Riccati equation plays a central role in optimal control theory, particularly in linear quadratic regulator (LQR) problems. By solving the Riccati equation, optimal control strategies can be found to minimize the cost of control and state errors. Additionally, the Riccati equation can be utilized for stability analysis of linear systems and designing stable feedback control systems. In our numerical experiment, set the iteration tolerance $\varepsilon = 1\times 10^{-12}$. The whole process is performed on a computer with Intel Core 1.00GHz CPU, 8.00GB RAM, and MATLAB R2018a. IT(inn) and IT(out) represent the number of inter iteration steps and outer iteration steps, respectively, and CPU by the computing time.

Example 3.4.1 Consider the linear time-invariant control system of the form

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t). \end{cases}$$

where

$$A = \begin{pmatrix} -12 & -3 & 0 & \cdots & 0 \\ 2 & -12 & -3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 2 & -12 & -3 \\ 0 & \cdots & 0 & 2 & -12 \end{pmatrix}_{n\times n}, \quad B = \begin{pmatrix} 0.2 \\ 0.2 \\ \vdots \\ 0.2 \\ 0.2 \end{pmatrix}_{n\times 1},$$

$$C = \left(0.1, 0.1, \cdots, 0.1, 0.1\right)_{1\times n}, \quad G = BB^T, \quad Q = C^T C.$$

Table 4 Numerical results for Example 3.4.1

n	Algorithm	Res	out(int)IT	CPU
128	K-N-GADI	4.2542e−15	8(8)	0.36s
128	K-N-RADI	1.7696e−15	4(8)	0.24s
128	K-N-RGADI	2.6821e−15	4(8)	0.21s
256	K-N-GADI	6.9046e−15	8(8)	1.38s
256	K-N-RADI	8.5988e−15	4(8)	0.76s
256	K-N-RGADI	5.0362e−15	4(8)	0.69s
512	K-N-GADI	7.1477e−15	8(9)	11.73s
512	K-N-RADI	1.0959e−14	6(9)	7.13s
512	K-N-RGADI	8.9506e−15	6(9)	6.85s
1024	K-N-GADI	9.2023e−15	8(13)	172.32s
1024	K-N-RADI	4.2303e−14	6(11)	141.4s
1024	K-N-RGADI	5.914e−15	6(12)	134.96s
2048	K-N-GADI	7.4253e−12	10(16)	3412.4s
2048	K-N-RADI	6.038e−13	8(16)	2941.4s
2048	K-N-RGADI	2.1016e−13	8(16)	2805 s

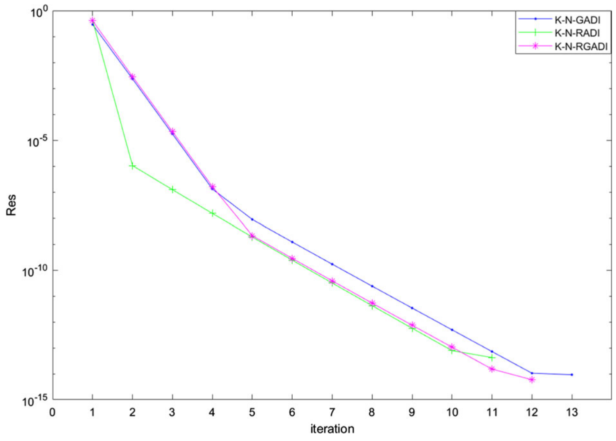

Fig. 5 The residual curve of Example 3.4.1 $n = 1024$

When the matrix dimension increases in multiples, we choose the relative residual as the iteration stopping criterion and use the Kleinman–Newton-GADI (K-N-GADI), Kleinman-Newton-RADI (K-N-RADI), and Kleinman–Newton-RGADI (K-N-RGADI) methods for computation. These iteration methods all start from the initial value $X_0 = 0$ and yield numerical results shown in Table 4. From the table data, we can see that the K-N-RGADI method is more efficient in solving this example problem compared to the K-N-GADI and K-N-RADI methods. Additionally, Fig. 5 clearly shows the variation of iteration steps and relative residual for these three methods when $n = 1024$, while Fig. 5 displays the time consumption of these three iteration methods as the matrix dimension increases. These findings further demonstrate the effectiveness of the K-N-RGADI method.

Furthermore, as the order n of the coefficient matrix in the equation increases multiplicatively, we employ the K-N-RGADI method with the relative change of the feedback matrix (R_s) as the iteration stopping criterion. We compare this method with the K-N-GADI and

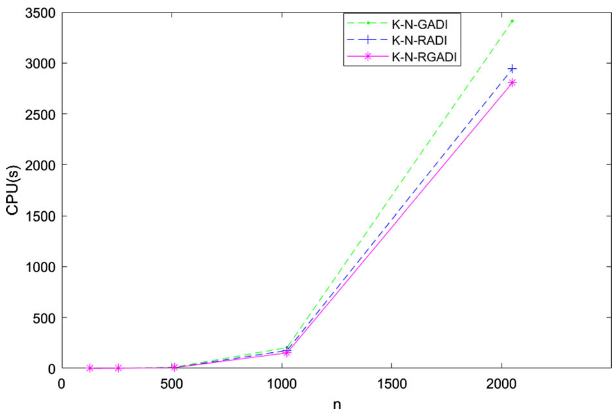

Fig. 6 The time curve of Example 3.4.1

Table 5 Numerical results for Example 3.4.1

K-N-GADI uses Res as the iteration stop standard						
n	128	256	512	1024	2048	4096
CPU	0.36s	1.38s	11.73s	172.42s	3412.4s	–
K-N-RGADI uses Res as the iteration stop standard						
n	128	256	512	1024	2048	4096
CPU	0.21s	0.69s	6.85s	134.96s	2805 s	–
K-N-RGADI uses R_s as the iteration stop standard						
n	128	256	512	1024	2048	4096
CPU	0.12s	0.57s	2.26s	30.95s	213.34s	3478.8s

K-N-RGADI methods that use the relative residual Res as the iteration stopping criterion. From Table 5, we can observe that when using the relative change of the feedback matrix as the iteration stopping criterion, the K-N-RGADI method significantly reduces the running time.

Figure 7 shows the iterative steps and corresponding residual R_s obtained by using the K-N-RGADI method to compute different matrix dimensions n. It can be clearly seen that as the matrix dimension increases, the residual R_s also increases.

Example 3.4.2 Consider the linear time-invariant control system of the form

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \\ y(t) = Cx(t). \end{cases}$$

where

$$A = \begin{pmatrix} -12 & -3 & -2 & 0 & 0 & \cdots & 0 \\ 2 & -12 & -3 & -2 & 0 & \cdots & 0 \\ 1 & 2 & -12 & -3 & -2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 1 & 2 & -12 & -3 & -2 \\ 0 & \cdots & 0 & 1 & 2 & -12 & -3 \\ 0 & \cdots & 0 & 0 & 1 & 2 & -12 \end{pmatrix}_{n\times n}, \quad B = \begin{pmatrix} 0.2 \\ 0.2 \\ \vdots \\ 0.2 \\ 0.2 \end{pmatrix}_{n\times 1},$$

$$C = \left(0.1,\ 0.1,\ \cdots,\ 0.1,\ 0.1\right)_{1\times n}, \quad G = BB^T, \quad Q = C^T C.$$

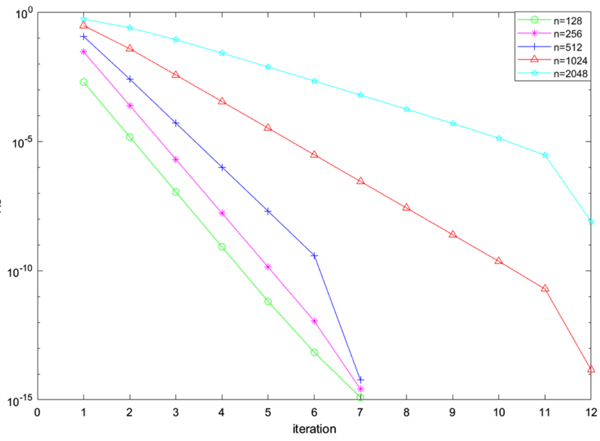

Fig. 7 The residual curve of Example 3.4.1

Table 6 Numerical results for Example 3.4.2

n	Algorithm	Res	out(int)IT	CPU
128	K-N-GADI	2.2313e−15	8(10)	0.46s
128	K-N-RADI	2.2274e−15	4(10)	0.35s
128	K-N-RGADI	3.4297e−15	4(10)	0.32s
256	K-N-GADI	4.8395e−15	8(10)	1.69s
256	K-N-RADI	9.4654e−15	4(10)	1.06s
256	K-N-RGADI	6.6721e−15	4(10)	1.04s
512	K-N-GADI	1.3522e−14	8(9)	13.68s
512	K-N-RADI	1.3458e−14	6(10)	10.56s
512	K-N-RGADI	1.22e−14	6(10)	8.29s
1024	K-N-GADI	7.2204e−14	8(13)	203.93s
1024	K-N-RADI	2.9667e−14	6(12)	176.04s
1024	K-N-RGADI	2.0719e−14	6(12)	152.61s
2048	K-N-GADI	1.2224e−12	10(16)	4019.5s
2048	K-N-RADI	2.5904e−13	8(16)	3145.8s
2048	K-N-RGADI	3.2006e−13	8(16)	2988.2s

We first used the relative residual Res as the iterative stopping criterion and adopted K-N-GADI, K-N-RADI, and K-N-RGADI methods to solve this example problem. The initial iteration value is set to $X_0 = 0$, and the numerical results obtained are shown in Table 6.

Next, we utilize the relative change of the feedback matrix (R_s) as the stopping criterion for the K-N-RGADI iteration method and compare its runtime with that of the K-N-GADI and K-N-RGADI iteration methods using the relative residual Res as the stopping criterion. As shown in Table 7, when using the relative change of the feedback matrix as the iteration stopping criterion, the K-N-RGADI method achieves a shorter runtime.

Table 7 Numerical results for Example 3.4.2

K-N-GADI uses Res as the iteration stop standard						
n	128	256	512	1024	2048	4096
CPU	0.46s	1.69s	13.68s	203.93s	4019.5s	–
K-N-RGADI uses Res as the iteration stop standard						
n	128	256	512	1024	2048	4096
CPU	0.32s	1.04s	8.29s	152.61s	2988.2s	–
K-N-RGADI uses R_s as the iteration stop standard						
n	128	256	512	1024	2048	4096
CPU	0.14s	0.63s	2.58s	35.26s	218.98s	3519.6.5s

4 Conclusions

This paper presents a low-rank GADI algorithm for computing low-rank approximate solutions to large-scale Lyapunov and algebraic Riccati equations. In the computation of low-rank approximate solutions to the algebraic Riccati equation, we combine the Kleinman-Newton method and utilize the low-rank GADI algorithm to solve the Lyapunov equation at each Newton step, resulting in the Kleinman-Newton-RGADI algorithm. Additionally, we observe that the low-rank GADI method exhibits the same convergence properties as the GADI method when solving both the Lyapunov and algebraic Riccati equations with low-rank approximations. Furthermore, numerical examples are provided to compare the effectiveness of the low-rank ADI algorithm and the low-rank GADI algorithm. The results demonstrate that the low-rank GADI method is more efficient. However, like other solvers, the performance of this algorithm heavily relies on the choice of shift parameters, which remains a challenging problem. Currently, there have been some new parameter selection methods developed. For instance, neural networks can be utilized to learn parameters during the process of solving the Lyapunov equation, to further improve the efficiency of the algorithm. Additionally, the Gaussian process regression (GPR) method based on the Bayesian inference, to predict the GADI framework's relatively optimal parameters. Applying these parameter selection methods to our work will be a key focus of our upcoming efforts.

Acknowledgements The authors would like to thank the referees for their constructive comments. Their remarks have helped us to improve the presentation of this paper.

Author Contributions JZ provides the core idea of the paper, and WLX writes the entire paper. All authors agree with the publication of the article.

Funding The work was supported in part by Hunan Youth Science and Technology Innovation Talents Project (2021RC3110), the Key Project of Education Department of Hunan Province (2023YFB3001604).

Data availibility Not applicable.

Declarations

Conflict of interest: The authors declare no Conflict of interest.

Ethical approval Not applicable.

References

Anderson BDO, Moore JB (1990) Optimal Control: Linear Quadratic Methods. Prentice-Hall, Englewood Cliffs, NJ

Roberts JD (1980) Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. Internat J Control 32(4):677–687

Jonckheere EA, Silverman LM (1983) A new set of invariants for linear systems-application to reduced order compensator design. IEEE Trans Autom Control 28(10):953–964

Saak J (2009) Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction. Dissertation, Department of Mathematics, University of Technology Chemnitz, Chemnitz, Germany

Williams D (1982) A “potential-theoretic” note on the quadratic Wiener-Hopf equation for Qmatrices. In Seminar on Probability XVI, Lecture Notes in Mathematics 920:91–94

Clancey K, Gohberg I (1981) Factorization of matrix functions and singular integral operators. In Operator Theory: Advances and Applications, vol. 3. Birkh$\ddot{a}$user Verlag: Basel

Lancaster P, Rodman L (1995) Algebraic Riccati Equations. Clarendon Press, Oxford

Wonham WM (1968) On a matrix Riccati equation of stochastic control. SIAM J Control Opt 6:681–697

Laub AJ (1979) A Schur method for solving algebraic Riccati equations. IEEE Trans Automat Control 24:913–921

Bai ZZ, Demmel J (1998) Using the matrix sign function to compute invariant subspaces. SIAM J Matrix Anal Appl 19:205–225

Guo XX, Xu SF, Lin WW (2005) A structure-preserving doubling algorithm for continuous-time algebraic Riccati equations. Linear Algebra Appl 396:55–80

Benner P, Fassbender H (1997) An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem. Linear Algebra Appl 263:75–111

Jbilou K (2003) Block Krylov subspace methods for large continuous-time algebraic Riccati equations. Numer Algorithms 34:339–353

Heyouni M, Jbilou K (2009) An extended block Arnoldi algorithm for large-scale solutions of the continuous-time algebraic Riccati equation. Electron Trans Numer Anal 33:53–62

Heinkenschloss M, Weichelt HK, Benner P, Saak J (2016) An inexact low-rank Newton-ADI method for large-scale algebraic Riccati equations. Appl Numer Math 108:125–142

Wong N, Balakrishnan V (2005) Quadratic alternating direction implicit iteration for the fast solution of algebraic Riccati equations. In: Proceedings of International Symposium on Intelligent Signal Processing and Communication Systems, 373–376

Amodei L, Buchot JM (2010) An invariant subspace method for large-scale algebraic Riccati equation. Appl Numer Math 60(11):1067–1082

Benner P, Kürschner P, Saak J (2018) RADI: a low-rank ADI-type algorithm for large scale algebraic Riccati equations. Numer Math 138:301–330

Penzl T (2000) A cyclic low-rank smith method for large sparse Lyapunov equations. SIAM J Sci Comput 21:1401–1418

Simoncini V (2008) A new iterative method for solving large-scale Lyapunov matrix equations. SIAM J Sci Comput 29:1268–1288

Li JR, White J (2002) Low rank solution of Lyapunov equations. SIAM J Matrix Anal Appl 24(1):260–280

Jiang K, Su X H, Zhang J (2021) A general alternating-direction implicit framework with Gaussian process regression parameter prediction for large sparse linear systems. arXiv e-prints

Li SF, Jiang K, Zhang J (2022) A general alternating-direction implicit Newton method for solving complex continuous-time algebraic Riccati matrix equation. Math. NA. arXiv:2203.02163v1

Xu SF (2011) Matrix calculation in cybernetics. Higher Education Press, Beijing

Kleinman D (1968) On an iterative technique for Riccati equation computations. IEEE Trans Auto Control 13:114–115

Rutherford DE (1932) On the solution of the matrix equation $AX + XB = C$. Nederl Akad Wetenseh Proc A 35:53–59

Lu A, Wachspress EL (1991) Solution of Lyapunov equations by alternating direction implicit iteration. Computerm Math Applie 21:43–58

Astudillo R, Gijzen MV (2016) Induced Dimension Reduction Method for Solving Linear Matrix Equations. Proc Comput Sci 80:222–232

Druskin V, Knizhnerman L, Simoncini V (2011) Analysis of the Rational Krylov Subspace and ADI Methods for Solving the Lyapunov Equation. Siam J Numer Anal 49(5):1875–1898

Ren H, Wang X, Wang T (2018) Commuting solutions of the Yang-Baxter-like matrix equation for a class of rank-two updated matrices. Comput Math Appl 76:1085–1098

Benner P, Li RC, Truhar N (2009) On the ADI method for Sylvester equations. Elsevier Sci Publ B 233(4):1035–1045

Zhou R, Wang X, Tang XB (2015) A generalization of the Hermitian and skew-Hermitian splitting iteration method for solving Sylvester equations. Appl Math Comput 271:609–617