A POD projection method for large-scale algebraic Riccati equations

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.
A POD PROJECTION METHOD FOR LARGE-SCALE
ALGEBRAIC RICCATI EQUATIONS

BORIS KRAMER*
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

JOHN R. SINGLER
Department of Mathematics and Statistics
Missouri University of Science and Technology
Rolla, MO 65409-0020, USA

(Communicated by Song Wang)

Abstract. The solution of large-scale matrix algebraic Riccati equations is
important for instance in control design and model reduction and remains an
active area of research. We consider a class of matrix algebraic Riccati equa-
tions (AREs) arising from a linear system along with a weighted inner product.
This problem class often arises from a spatial discretization of a partial diffe-
cntial equation system. We propose a projection method to obtain low rank
solutions of AREs based on simulations of linear systems coupled with proper
orthogonal decomposition. The method can take advantage of existing (black
box) simulation code to generate the projection matrices. We also develop new
weighted norm residual computations and error bounds. We present numerical
results demonstrating that the proposed approach can produce highly accurate
approximate solutions. We also briefly discuss making the proposed approach
completely data-based so that one can use existing simulation codes without
accessing system matrices.

1. Introduction. Riccati equations play an important role in a variety of problems,
including optimal control and filtering. For instance, the solution to the algebraic
Riccati equation (ARE) determines the optimal feedback control solving the linear
quadratic regulator problem. Such feedback control laws are used to stabilize a
dynamical system and to steer the dynamics to desired equilibrium states. More-
over, the problem of optimal state estimation from given measurements of a linear
dynamical system also involves a solution to an ARE. For details about control and
estimation, see [47, Chapter 4] and [23, Chapter 12] and the many references there-
in. Solutions of AREs are also important for certain model reduction algorithms,
such as LQG balanced truncation, [3, Section 7.5] and [68, 12]. Furthermore, op-
timizing sensor and actuator locations in optimal control problems can require the

2010 Mathematics Subject Classification. Primary: 49; Secondary: 65.
Key words and phrases. Algebraic Riccati equations, reduced-order modeling, large-scale, proper
orthogonal decomposition, control theory.

The second author was supported in part by NSF under grant DMS-1217122.
*Corresponding author: B. Kramer.
solution of many AREs; see, e.g., [27, 28, 43, 26] and the references therein. Hence, solving nonlinear matrix AREs of the form

$$A^TPE + E^TPA - E^TPBB^TPE + C^TC = 0 \quad (1)$$

is a key step in many applications. Here, $A, E \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and $C \in \mathbb{R}^{p \times n}$ are given matrices, and $P \in \mathbb{R}^{n \times n}$ is the unknown matrix. Under reasonably weak assumptions on $(E, A, B, C)$, the exact solution $P$ of the ARE (1) exists and is symmetric positive semidefinite; e.g., [47, Theorem 3.7].

We are concerned with approximating the solution of the ARE (1) for large-scale systems, i.e., when $n > 10,000$. One instance where such AREs arise in the applications mentioned above is when the linear dynamical system is derived from a spatial discretization of a partial differential equation (PDE) using, e.g., finite element methods. The dimension $n$ can easily be on the order of $10^6$ or larger for such applications. Even if the matrices $E$ and $A$ are sparse and $m, p \ll n$, the exact solution $P$ of the ARE is a dense $n \times n$ matrix; therefore obtaining or even storing the exact solution is infeasible for many problems. Fortunately, the solution $P$ is often of low numerical rank when $p, m \ll n$ [15, 52] and many recent solution approaches exploit this by constructing factored low rank approximate solutions of the form $P \approx ZZ^T$. Over the past 50 years many methods and techniques have been developed to efficiently solve small or moderate sized nonlinear matrix equations of Riccati type; see, e.g., [18] for an overview. A large amount of recent research has been devoted to the development and analysis of algorithms for large-scale AREs; see the recent survey [15]. Many of the approaches are inspired by computational linear algebra methods and include Krylov subspace methods in a projection framework [58, 38, 39, 40, 36, 61], ADI methods [13, 14, 15], subspace iterations [50, 16], and doubling methods [49, 72, 73].

There has also been interest in developing data-based algorithms that approximate the solution of the ARE using only simulation data. Such algorithms do not require direct access to the matrices $(E, A, B, C)$. This can be important if one has an existing (possibly complex) simulation code for which it is difficult or impossible to access the relevant matrices. In such a data-based setting, researchers have not typically focused on approximating the solution $P$ of the ARE; instead, researchers have primarily focused on approximating quantities that depend on the Riccati solution. For example, one can attempt to approximate the feedback gain matrix $K = B^TPE \in \mathbb{R}^{m \times n}$ needed for optimal feedback control problems, or one can attempt to approximate the optimal control only (which is not in feedback form). Data-based algorithms include approaches based on the Chandrasekhar equations [42, 51, 22, 20, 19, 1], iterative optimization algorithms [55, 59], and a variety of model reduction methods (see, e.g., [6, 5, 9, 8, 48]). All of these approaches have been used successfully on a variety of problems, but they can have drawbacks. For example, the Chandrasekhar equations are nonlinear and can be costly to simulate. Also, the iterative optimization algorithms only provide the optimal control which cannot be used for feedback purposes. Furthermore, these methods have typically not aimed to provide highly accurate approximations of the Riccati solution $P$.

We propose a new projection based method in Section 5 to solve AREs based on simulations of linear systems coupled with proper orthogonal decomposition (POD) and Galerkin projection.\footnote{A preliminary version of this algorithm can be found in [45].} The proposed approach is a first step toward...
an efficient, accurate, completely data-based method that can take advantage of existing simulation codes. This allows for additional flexibility of the algorithm, such as when the matrices come from discretizations of PDEs, efficient domain decomposition and multigrid techniques can be used to solve the linear systems. Moreover, for systems with multiple outputs, the required simulations can be run in parallel. We note that a data-based POD approach has been previously coupled with a Kleinman-Newton iteration in [63] to approximately solve AREs. However, the recent work [61] indicates that projection methods outperform approaches based on the Kleinman-Newton iteration, which we leverage by incorporating the data-based method in a projection framework. We demonstrate that the new POD projection method can be used to accurately compute the feedback gain matrices and also provide an accurate very low rank approximate solution of the ARE, which can be used for model reduction applications, etc.

We consider a class of AREs that arise from a linear system with an inner product weighted by a symmetric positive definite matrix. This type of problem often arises from a spatial discretization of a PDE system. For this problem class, we derive new weighted norm residual computations and error bounds.

In the following Section 2, we describe the problem class and present necessary background material. For readers not familiar with projection methods, we give a brief overview in Section 3. We derive the new weighted norm residual computations and error bounds in Section 4, and then describe the POD-projection approach in Section 5. We present numerical results in Section 6, indicating that the proposed method can be used to accurately compute the feedback gain matrices and also provide an accurate very low rank approximate solution of the ARE, which can be used for model reduction applications, etc. In the conclusion, we briefly discuss transitioning the proposed approach into a completely data-based algorithm.

2. Assumptions, Background, and Notation. Throughout this paper, we consider linear time invariant systems given by

\[ E\dot{x}(t) = Ax(t) + Bu(t), \]
\[ y(t) = Cx(t), \]

where \( E, A \in \mathbb{R}^{n \times n}, \ B \in \mathbb{R}^{n \times m} \) and \( C \in \mathbb{R}^{p \times n} \) and \( m, p \ll n \). Here, \( \dot{x} = dx/dt \) denotes the usual time derivative of \( x \). We assume throughout that the matrix \( E \) is symmetric positive definite (and therefore invertible). This assumption on \( E \) is satisfied for many systems of the above form derived from a spatial discretization of a PDE system. There are systems of interest where \( E \) does not satisfy this assumption (see, e.g., [17]); we do not consider this here.

For certain systems the system (2)-(3) is not stable in the sense that \( E^{-1}A \) has an eigenvalue with positive real part. A system is called asymptotically stable if all eigenvalues of \( E^{-1}A \) have negative real part. Stabilizing unstable systems is one important problem in feedback control applications. Whenever the system is unstable, the observability Gramian given by representation (6) below does not exist and certain ad hoc solution strategies for ARE can have poor performance. To make \( E^{-1}A \) stable, one first finds a stabilizing feedback gain \( K_1 \), for instance via the algebraic Bernoulli equations [2] or by integrating the Chandrasekhar equations until a stabilizing feedback gain is obtained [10]. The new matrix \( E^{-1}(A - BK_1) \) is then stable.
For convenience we assume that $E^{-1}A$ is stable. In light of the preceding discussion, we emphasize that this is not a necessary assumption. In fact, the assumption of stability of $E^{-1}A$ solely implies that the stabilizing feedback has already been applied.

Many algorithms to solve AREs exploit the intrinsic connection to the linear Lyapunov equation

$$A^T X E + E^T X A + C^T C = 0. \quad (4)$$

One should note that (4) is a linear matrix equation obtained from the ARE (1) by ignoring the nonlinear term. The connection between the ARE and Lyapunov equation can have important implications when devising algorithms to solve AREs.

In this paper, we generate a subspace spanned by the singular vectors of an approximate Lyapunov solution to compute approximate solutions for AREs via projection.

Assuming that $E^{-1}A$ is stable, the solution to Lyapunov equation admits a closed form. To see this, rewrite the Lyapunov equation (4) above as

$$\tilde{A}^T \tilde{X} + \tilde{X} \tilde{A} + \tilde{C}^T \tilde{C} = 0, \quad (5)$$

where $\tilde{X} = E^T X E$, $\tilde{A} = E^{-1}A$, $\tilde{C} = C$.

It is well known that the solution of the transformed Lyapunov equation (5) has the following integral representation:

**Theorem 2.1.** ([3, Proposition 4.27]) If $\tilde{A} = E^{-1}A$ is a stable matrix, then the Lyapunov equation (5) has a unique solution $\tilde{X}$, called the **observability Gramian**, which has the representation

$$\tilde{X} = \int_0^\infty e^{t\tilde{A}^T} \tilde{C}^T \tilde{C} e^{t\tilde{A}} dt. \quad (6)$$

The solution of the original Lyapunov equation (4) can be expressed in a similar form.

**Corollary 1.** If $\tilde{A} = E^{-1}A$ is a stable matrix, then the Lyapunov equation (4) has a unique solution $X$ given by

$$X = \int_0^\infty e^{tE^{-1}} A^T E^{-T} C^T C E^{-1} e^{tA} dt.$$

**Proof.** The above theorem gives

$$X = \int_0^\infty E^{-T} e^{(E^{-1}A)^T} C^T C e^{(E^{-1}A) E^{-1}} dt$$

$$= \int_0^\infty \left( E^{-T} e^{tA^T} E^{-T} E^T \right) E^{-T} C^T C E^{-1} \left( E e^{E^{-1}(tA)} E^{-1} \right) dt.$$

Using $M^{-1} e^N M = e^{M^{-1}NM}$ for $M = E^T$, $M^{-1} = E^{-T}$, $N = tA^T$ and also $M = E^{-1}$, $M^{-1} = E$, $N = tA$ gives the result.

In this work, we work with the $E$-weighted vector and matrix norms. We are interested in systems arising from spatial discretization of a PDE system. In a PDE system, the natural state space is often a Hilbert space of functions, such as $L^2$ or $H^1$. When the system (2)-(3) arises from a standard finite element spatial discretization of a partial differential equation system, the matrix $E$ is symmetric positive definite and the $E$-weighted norm of a vector equals the Hilbert space norm.
of the corresponding finite element function. Let \((x,y)_E = y^TEx\) denote the \(E\)-weighted inner product on \(\mathbb{R}^n\) and let \(\|x\|_E = (x,x)^{1/2}_E = (x^TEx)^{1/2}\) denote the corresponding \(E\)-weighted norm.

We denote the standard matrix 2-norm by \(\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2 = \sigma_1(A)\) and the Frobenius norm by \(\|A\|_F^2 = \text{trace}(ATA) = \sum_{i=1}^n \sigma_i^2\). Here, \(\sigma_1 \geq \sigma_2 \geq \cdots \geq 0\) denote the singular values of \(A\).

For notational convenience, we adopt \texttt{Matlab} notation herein. Given a vector \(u \in \mathbb{R}^n\) and \(r \leq n\), the vector \(u(1:r)\) denotes the vector of the first \(r\) components of \(u\). Similarly, for a matrix \(A \in \mathbb{R}^{n \times n}\), we denote by \(A(1:r, 1:r)\) the leading \(r \times r\) submatrix of \(A\).

3. **Direct Projection Methods.** Projection methods have been demonstrated to be efficient methods to compute solutions of Riccati and Lyapunov equations [38, 39, 40, 41, 60, 36, 61]. In this section, we give an overview of the projection approach and discuss the choice of a projection matrix. For a symmetric positive definite weight matrix \(W \in \mathbb{R}^{n \times n}\), assume we have a matrix \(V_r\) with full column rank such that

\[
 V_r = [v_1, \ldots, v_r] \in \mathbb{R}^{n \times r}, \quad V_r^TWV_r = I_r, \tag{7}
\]

where \(v_i \in \mathbb{R}^n\) for each \(i\) and \(r \ll n\) is the reduced dimension. The matrix \(V_r\) is called a projection matrix and is used to obtain a reduced-order model \(E_r, A_r, B_r, C_r\) of the system \(E, A, B, C\) via

\[
 E_r = V_r^TEV_r, \quad A_r = V_r^TAV_r, \quad B_r = V_r^TB, \quad C_r = CV_r. \tag{8}
\]

The reduced-order matrices give the following low-order projected ARE

\[
 A_r^T\Pi_rE_r + E_r^T\Pi_rA_r - E_r^T\Pi_rB_rB_r^T\Pi_rE_r + C_r^TC_r = 0 \quad \in \mathbb{R}^{r \times r}, \tag{9}
\]

which can alternatively be obtained by imposing a Galerkin condition on the residual matrix. In this work, we always take \(W\) to be the symmetric positive definite matrix \(E_r\) and so \(E_r = I_r\).

Assuming the low-order ARE is well posed, the solution \(\Pi_r \in \mathbb{R}^{r \times r}\) can be computed using the well developed methods for moderate sized AREs; e.g., the direct solver \texttt{care} in \texttt{Matlab}. Having solved the low-order ARE (9), one defines a low rank approximate solution to the large-scale ARE (1) as

\[
 P_r := V_r\Pi_rV_r^T \approx P. \tag{10}
\]

Projection methods automatically yield low rank factored solutions. As noted earlier, the solution to the low-rank ARE \(\Pi_r\) is symmetric positive semidefinite. Thus, the eigenvalue decomposition \(\Pi_r = U_rS_rU_r^T\) is used to define the low rank factorization

\[
 P_r = Z_rZ_r^T, \quad Z_r = V_rU_rS_r^{1/2}, \tag{11}
\]

In practice, only the low rank factor \(Z_r\) is stored and used where the solution \(P_r\) would be needed.

All Galerkin projection methods utilize the steps (7)-(11) above. The distinctive feature of a method is the structure and generation of the projection matrix \(V_r\). Krylov subspace methods have been a popular choice for generating \(V_r\). For instance, standard Krylov subspaces have been considered for Lyapunov equations (4) in [58, 38]. More efficient extended Krylov subspaces, incorporating the inverse \(A^{-1}\) as well have been suggested in [30] and henceforth were used in [60] for the case of Lyapunov equations, and to solve algebraic Riccati equations in [36, 61].
As we see later, if the columns of $V_r$ are in the span of the solution to the Lyapunov equation (4) then the projection based method for solving an ARE can be very accurate. To investigate this for a small-scale problem, we compute the direct solution to the Lyapunov equation in Matlab via $X = \text{lyap}(A^T, C^T C, [I], E^T)$ and then take the singular value decomposition of the observability Gramian $X = V \Sigma W^T$, where the columns of $V = [v_1 \ v_2 \ldots \ v_n]$ span the range space of $X$. Truncation of $V$ after $r$ columns yields the projection matrix as $V_r = [v_1 \ v_2 \ldots \ v_r]$, which we use for comparison reasons on a small test problem.

4. Residual Computations and Approximation Errors: Weighted Norms.
To test the accuracy of the approximate solution obtained in the projection method for AREs, many researchers have computed the standard or Frobenius norm of the matrix residual. Below, we prove that the weighted norms of the residual can be reduced to the norm computation of a small square matrix (of size $2r \times 2r$). Therefore, computing the norm of the residual matrix

$$\|P - X_P\|_F = \|\mathcal{R}(P_r)\|_F$$

is computationally tractable, and it scales to large problems. We use this approach for the computations in this paper.

We begin by briefly reviewing the residual QR algorithm from [13]. Let $P_r \in \mathbb{R}^{n \times n}$ be a symmetric positive semidefinite approximate solution of the ARE (1). The residual is defined as

$$\mathcal{R}(P_r) = A^T P_r E + E^T P_r A - E^T P_r B B^T P_r E + C^T C.$$

The residual can be rewritten as

$$\mathcal{R}(P_r) = F G F^T,$$

where $F \in \mathbb{R}^{n \times (2r+p)}$, $G \in \mathbb{R}^{(2r+p) \times (2r+p)}$, and $C \in \mathbb{R}^{p \times n}$. Let $F = QR$ be the thin QR decomposition of $F$, see [34, page 230]. Since $Q^T Q = I$, we have

$$\|\mathcal{R}(P_r)\|_2 = \|R G R^T\|_2, \quad \|\mathcal{R}(P_r)\|_F = \|R G R^T\|_F.$$

Therefore, computing the norm of the residual matrix $\mathcal{R}(P_r)$ (of size $n \gg 1$) can be reduced to the norm computation of a small square matrix (of size $2r + p \ll n$) using a thin QR decomposition. Below, we prove that weighted norms of the residual can be computed with similar formulas at similar computational cost.

4.1. Residual Computations: Weighted Norms. For the weighted residual norm computation, we extend the residual norm QR algorithm from [13] to the weighted norm case. The resulting algorithm is applicable for many low rank AREs, arising from discretizations of PDEs and other infinite-dimensional systems, see [32] and [71, Chapter 1] and the references therein.
where \( \hat{A} = E^{-1}A \) and \( \hat{B} = E^{-1}B \), the ARE on \( X \) with unknown \( \hat{P} \) is given by
\[
\hat{A}^* \hat{P} + \hat{P} \hat{A} - \hat{P} \hat{B} \hat{B}^* \hat{P} + C^* C = 0. \tag{13}
\]
Here, the matrices are viewed as mappings\(^2\) \( \hat{P} : X \to X, \hat{A} : X \to X, \hat{B} : R^m \to X, \) and \( C : X \to R^n \), and the Hilbert adjoints \( \hat{A}^* : X \to X, \hat{B}^* : X \to R^m, \) and \( C^* : R^p \to X \) satisfy
\[
(\hat{A}x, z)_E = (x, \hat{A}^* z)_E, \quad (\hat{B}u, z)_E = (u, \hat{B}^* z)_R^m, \quad (Cx, y)_R^p = (x, C^* y)_E
\]
for all \( x, z \in X, u \in R^m, \) and \( y \in R^p \). As in\([32]\) and\([71, \text{Chapter 1}]\), straightforward computations show that the adjoints are given by
\[
\hat{A}^* = E^{-1} A^T, \quad \hat{B}^* = B^T, \quad C^* = E^{-1} C^T.
\]
Therefore, using \( \hat{A} = E^{-1} A \) and \( \hat{B} = E^{-1} B \), the ARE (13) on \( X \) becomes
\[
E^{-1} A^T \hat{P} + \hat{P} E^{-1} A - \hat{P} E^{-1} B B^T \hat{P} + E^{-1} C^T C = 0.
\]
The ARE on \( X \) is not in a standard form for a matrix algebraic Riccati equation. Multiply the ARE on \( X \) on the left by \( E \) to obtain a standard ARE for the new unknown \( P = \hat{P} E^{-1} \):
\[
A^T P E + E P A - E P B B^T P E + C^T C = 0. \tag{14}
\]
Once an approximation \( P_r \) is found for \( P \), an approximation \( \hat{P}_r \) for \( \hat{P} \) can be found using the relationship between \( \hat{P} \) and \( P \) above:
\[
\hat{P}_r = P_r E. \tag{15}
\]
Define the residual for the ARE (13) on \( X \) by
\[
\hat{R}(\hat{P}_r) = E^{-1} A^T \hat{P}_r + \hat{P}_r E^{-1} A - \hat{P}_r E^{-1} B B^T \hat{P}_r + E^{-1} C^T C.
\]
Assuming \( P_r \) and \( \hat{P}_r \) are related as in (15), then residual \( \hat{R}(\hat{P}_r) \) on \( X \) is related to the “standard” residual \( R(P_r) \) in (12) by
\[
\hat{R}(\hat{P}_r) = E^{-1} R(P_r). \tag{16}
\]
We assume \( P_r = Z_r Z_r^T \) so that the standard residual can be factored as
\[
R(P_r) = F G F^T,
\]
where \( G \) is symmetric.

We want to use the standard residual to compute the operator norm and/or the Hilbert-Schmidt (or Frobenius) norm of \( \hat{R}(\hat{P}_r) \) considered as a mapping from \( X \) to \( X \). Let \( \{ \sigma_k \} \) be the singular values of \( \hat{R}(\hat{P}_r) \). Then these two norms are given by
\[
\| \hat{R}(\hat{P}_r) \|_{L(X)} = \max_{k \geq 1} \sigma_k, \quad \| \hat{R}(\hat{P}_r) \|_{HS(X)} = \left( \sum_{k \geq 1} \sigma_k^2 \right)^{1/2}.
\]

**Proposition 1.** Assume \( P_r = Z_r Z_r^T, \hat{P}_r = P_r E, \) and \( R(P_r) = F G F^T, \) where \( G = G^T. \) Let \( E = L_E L_E^T \) be the Cholesky decomposition of \( E, \) and let \( L_E^{-1} F = U \Delta \) be the QR decomposition of \( L_E^{-1} F, \) where \( U \) is orthogonal and \( \Delta \) is upper triangular. Then
\[
\| \hat{R}(\hat{P}_r) \|_{L(X)} = \| \Delta G \Delta^T \|_2, \quad \| \hat{R}(\hat{P}_r) \|_{HS(X)} = \| \Delta G \Delta^T \|_F.
\]

\(^2\)Here and elsewhere in this work, the inner product and norm on \( R^s \) is the standard dot product and Euclidean norm unless specifically indicated otherwise.
Proof. First, the Hilbert adjoint of $\mathcal{R}(\hat{P}_r) : X \to X$ satisfies
\[
(\mathcal{R}(\hat{P}_r)x,z)_E = (x, [\mathcal{R}(\hat{P}_r)]^* z)_E
\]
for all $x,z \in X$. A direct computation gives $[\mathcal{R}(\hat{P}_r)]^* = E^{-1} [\mathcal{R}(\hat{P}_r)]^T E$. Using the relationship (16) between the residuals and $E^{-T} = E^{-1}$ gives that $\mathcal{R}(\hat{P}_r) : X \to X$ is self-adjoint:
\[
[\mathcal{R}(\hat{P}_r)]^* = E^{-1} [\mathcal{R}(P_r)]^T = E^{-1} \mathcal{R}(P_r) = \mathcal{R}(\hat{P}_r).
\]
Here, we used that the standard residual is symmetric (since $P_r$ is symmetric).

Since $[\mathcal{R}(\hat{P}_r)]^* = \mathcal{R}(\hat{P}_r)$, the nonzero singular values of $\mathcal{R}(\hat{P}_r) : X \to X$ are given by the absolute value of the nonzero eigenvalues of $\mathcal{R}(\hat{P}_r) : X \to X$. Let $\lambda$ be a nonzero eigenvalue with eigenvector $x$, i.e.,
\[
\mathcal{R}(\hat{P}_r)x = \lambda x.
\]
Using the residual relationship (16) gives
\[
[L_E^{-1} \mathcal{R}(P_r)L_E^{-T}]z = \lambda z, \quad z = L_E^T x.
\]
Therefore the nonzero eigenvalues of $\mathcal{R}(\hat{P}_r)$ are equal to the nonzero eigenvalues of $L_E^{-1} \mathcal{R}(P_r)L_E^{-T}$. The standard residual $\mathcal{R}(P_r)$ is symmetric, and therefore the nonzero singular values of $\mathcal{R}(P_r)$ equal the absolute value of the nonzero eigenvalues.

This gives
\[
\|\mathcal{R}(\hat{P}_r)\|_{\mathcal{L}(X)} = \|L_E^{-1} \mathcal{R}(P_r)L_E^{-T}\|_2, \quad \|\mathcal{R}(\hat{P}_r)\|_{HS(X)} = \|L_E^{-1} \mathcal{R}(P_r)L_E^{-T}\|_F.
\]

Since $\mathcal{R}(P_r) = FGF^T$, $L_E^{-1} F = U\Delta$, and $U$ is orthogonal, the result follows. \qed

4.2. Approximation Errors: Weighted Norms. Following Kenney, Laub, and Wette [44], we can use the residual to bound weighted norm errors between $\hat{P}$ and $\hat{P}_r$ assuming the error is small enough. Define the Lyapunov operator
\[
\hat{L}_r(M) = (\hat{A} - \hat{B}\hat{B}^* \hat{P}_r)^* M + M(\hat{A} - \hat{B}\hat{B}^* \hat{P}_r).
\]
If $\hat{A} - \hat{B}\hat{B}^* \hat{P}_r$ is stable, then the Lyapunov equation $\hat{L}_r(M) = Q$ has a unique solution which we denote $\hat{L}_r^{-1} Q$. For $\mathfrak{X} := \mathfrak{L}(X)$ or $\mathfrak{X} := HS(X)$, define the operator norm of $\hat{L}_r^{-1} : \mathfrak{X} \to \mathfrak{X}$ by
\[
\|\hat{L}_r^{-1}\|_{\mathfrak{L}(X)} = \sup_{Q \in \mathfrak{X}} \frac{\|\hat{L}_r^{-1} Q\|_{\mathfrak{X}}}{\|Q\|_{\mathfrak{X}}},
\]
where $\mathfrak{L}(X)$ is the space of linear operators mapping $\mathfrak{X}$ to $\mathfrak{X}$. The above definition gives
\[
\|\hat{L}_r^{-1} Q\|_{\mathfrak{X}} \leq \|\hat{L}_r^{-1}\|_{\mathfrak{L}(X)} \|Q\|_{\mathfrak{X}}.
\]

**Proposition 2.** Let $\mathfrak{X} := \mathfrak{L}(X)$ or $\mathfrak{X} := HS(X)$ as above. If $\hat{P}_r = \hat{P}_r^*$, $\hat{A} - \hat{B}\hat{B}^* \hat{P}_r$ is stable, and
\[
\|\hat{P} - \hat{P}_r\|_{\mathfrak{X}} \leq \frac{1}{3 (\hat{P}_r)^{-1}\|\mathfrak{B}\|_{\mathfrak{L}(R^m, X)}}, \quad 4 \|\hat{L}_r^{-1}\|_{\mathfrak{L}(X)}^2 \|\mathcal{R}(\hat{P}_r)\|_{\mathfrak{X}} \|\hat{B}\|_{\mathfrak{L}(R^m, X)}^2 < 1,
\]
then
\[
\|\hat{P} - \hat{P}_r\|_{\mathfrak{X}} \leq \frac{2 \|\hat{L}_r^{-1}\|_{\mathfrak{L}(X)} \|\mathcal{R}(\hat{P}_r)\|_{\mathfrak{X}}}{1 + \sqrt{1 - 4 \|\hat{L}_r^{-1}\|_{\mathfrak{L}(X)}^2 \|\mathcal{R}(\hat{P}_r)\|_{\mathfrak{X}} \|\hat{B}\|_{\mathfrak{L}(R^m, X)}^2}} \\
\leq \frac{2 \|\hat{L}_r^{-1}\|_{\mathfrak{L}(X)} \|\mathcal{R}(\hat{P}_r)\|_{\mathfrak{X}}}{\|\hat{B}\|_{\mathfrak{L}(R^m, X)}}.
\]
Proof. The proof is similar to the proof of Theorems 2 and 2’ in [44]. For ease of notation, let $\hat{R}_r$ denote the residual $R(\hat{P}_r)$. First, it can be checked that

$$\hat{P}_r - \hat{P} = \hat{L}_r^{-1} \left[ \hat{R}_r - (\hat{P}_r - \hat{P}) \hat{B}^* (\hat{P}_r - \hat{P}) \right].$$

Let $N = (\hat{P}_r - \hat{P}) \hat{B}$. Since $\hat{P} = \hat{P}^*$ and $\hat{P}_r = \hat{P}_r^*$, this gives

$$\|\hat{P}_r - \hat{P}\|_X \leq \|\hat{L}_r^{-1}\|_{\mathcal{L}(X)} \left[ \|\hat{R}_r\|_X + \|NN^*\|_X \right].$$

Next, use

$$\|NN^*\|_X = \|N\|_{\mathcal{L}(\mathbb{R}^m, X)}^2 \leq \|\hat{B}\|_{\mathcal{L}(\mathbb{R}^m, X)}^2 \|\hat{P}_r - \hat{P}\|_X^2$$

to obtain

$$x \leq a_0 + a_1 x^2,$$

where

$$x = \|\hat{P}_r - \hat{P}\|_X, \quad a_0 = \|\hat{L}_r^{-1}\|_{\mathcal{L}(X)} \|\hat{R}_r\|_X, \quad a_1 = \|\hat{L}_r^{-1}\|_{\mathcal{L}(X)} \|\hat{B}\|_{\mathcal{L}(\mathbb{R}^m, X)}^2.$$

The rest of the proof follows [44] exactly. \qed

Notes:

• The quantity $c(\hat{P}_r) = 2 \|\hat{L}_r^{-1}\|_{\mathcal{L}(X)}$ can be viewed as a condition number since

$$\|\hat{P} - \hat{P}_r\|_X \leq c(\hat{P}_r) \|\hat{R}_r\|_X.$$

• Computing the quantity $\|\hat{L}_r^{-1}\|_{\mathcal{L}(X)}$ is unfortunately not straightforward, as is discussed in [44].

• Sun sharpened the original residual error bound of Kenney, Laub, and Wette, and also relaxed the assumptions [66]. It may be possible to extend Sun’s approach to the situation here; we leave this to be considered elsewhere.

• Since $\hat{B}^*: X \to \mathbb{R}^m$ is given by $B^* = \hat{B}^T$, it can be shown that $\hat{B}: \mathbb{R}^m \to X$ has the same nonzero singular values as the matrix $L_E^{-1} B$, where $E = L_E L_E^T$. Therefore, $\|\hat{B}\|_{\mathcal{L}(\mathbb{R}^m, X)} = \|L_E^{-1} B\|_2$.

5. POD Projection Method. Proper Orthogonal Decomposition (POD) is a widely used model reduction method in the engineering and mathematics communities. Depending on the field, it is also known as Karhunen-Loeve expansion or principal component analysis. POD has been used to create low-order models of complex systems in order to speed up simulation, optimization, control, and many other applications; see, e.g., [37, 24, 46, 35, 25, 29, 70, 69] and the references therein.

We propose using POD in the projection framework to approximate solutions to AREs in an accurate and computationally efficient manner. First, a POD method is employed to approximate the dominant eigenvectors of the observability Gramian $X$ solving the Lyapunov equation (4). Those vectors are used to construct a projection matrix $V_r$ which is used in a projection framework to solve the ARE.

In [74], a snapshot based approach was used to approximate solutions of the Lyapunov equation in $n$ dimensions (see also [58, 56]). In particular, the authors suggested using snapshots of simulations of linear systems to compute the observability Gramian (6). In [64], this idea was extended to infinite-dimensional Lyapunov equations and a rigorous convergence theory was presented. Specifically, error bounds and convergence of the low rank, finite-dimensional solution to the infinite-dimensional Gramian were obtained. We follow this approach and first compute an approximation of the observability Gramian $X$ and then we project the ARE using the approximate dominant left singular vectors of $X$. The dimension of
the necessary singular value decomposition is limited by \( \min(n, ps) \), with \( s \) being the number of snapshots collected during the simulation of a linear system.

In control and systems theory, the dual equations of the underlying optimal control problem are

\[
E^T \dot{z}_i(t) = A^T z_i(t), \tag{18}
\]

\[
E^T z_i(0) = c_i^T, \tag{19}
\]

for all \( i = 1, \ldots, p \), where \( C^T = [c_1^T, c_2^T, \ldots, c_p^T] \). Recall that \( E \) is symmetric positive definite; therefore, \( E = E^T \) in the above problem. By the theory of ordinary differential equations, the unique solution to (18)-(19) is given by \( z_i(t) = e^{tE^{-1}A^T} E^{-1} c_i^T \), for \( i = 1, \ldots, p \). Due to the representation of the Lyapunov solution in Corollary 1, the authors in [74] thus suggested to use simulations of the dual equations to approximate the solution of the observability Gramian \( X \). When \( E^{-1}A \) is stable, the Gramian can be rewritten as follows:

\[
X = \int_0^\infty e^{tE^{-1}A^T} E^{-1} C^T E^{-1} e^{tAE^{-1}} dt
\]

\[
= \int_0^\infty e^{tE^{-1}A^T} E^{-1} [c_1^T, \ldots, c_p^T] \begin{bmatrix} c_1 \\ \vdots \\ c_p \end{bmatrix} E^{-1} e^{tAE^{-1}} dt
\]

\[
= \int_0^\infty e^{tE^{-1}A^T} E^{-1} [c_1^T, \ldots, c_p^T] \left( e^{tE^{-1}A^T} E^{-1} [c_1^T, \ldots, c_p^T] \right)^T dt
\]

\[
= \int_0^\infty [z_1(t), \ldots, z_p(t)] [z_1(t), \ldots, z_p(t)]^T dt
\]

\[
= \int_0^\infty \left[ \begin{array}{c} t \\ \vdots \\ t \end{array} \right] [Z(t)]^T dt.
\]

We approximate the observability Gramian by the finite time integral

\[
X \approx X_T := \int_0^T [Z(t)]^T [Z(t)]^T dt,
\]

where \( T \) specifies a final time, chosen so that a good approximation of the infinite integral is obtained. The finite time integral can be approximated by quadrature, such that

\[
X \approx X_{T, \delta} := \sum_{i=1}^s \delta_i [Z(t_i)] [Z(t_i)]^T, \tag{20}
\]

using positive weights \( \delta_i \) and a time grid \( 0 = t_1 < t_2 < \ldots < t_s = T \). Here, let \( s_i = t_{i+1} - t_i \) be the step size for the numerical integration scheme at the \( i^{th} \) time step. In matrix form, the approximation reads as

\[
X \approx X_{T, \delta} := Z \Delta Z^T,
\]

where \( \Delta \) is the diagonal matrix of weights and \( Z \) contains snapshots of simulations, as outlined below. The method of snapshots [65] is used for the POD computations, as briefly reviewed below. We refer the reader to, e.g., [70, Chapters 2–3] and the references above for more detail. We note that the POD computations can be performed using other approaches; see, e.g., [31, 11].

Some technical details for the implementation of the POD snapshot based approach to approximate the Gramian are listed below.
For accurate simulation of the dual system (18)-(19), a proper set of time steps has to be chosen a priori, or adaptively during the time stepping. In this work, we simulated the large-scale test problem with Matlab’s adaptive time stepping solver \texttt{ode23s}, with default absolute and relative error tolerances. In most cases, the snapshots are selected at the time steps chosen by the adaptive solver. In the small scale setting, we used a second order trapezoidal rule integration scheme and used all available snapshots.

In case of highly stiff problems, the time steps \( s_i \) are small, which results in a larger set of snapshots than is needed for computation of \( V_r \). In this case, a subset of snapshots from the previous step is selected, and the singular value decomposition computed from this smaller set of vectors. In practice, we found this approach to not lose significant accuracy, compared to keeping all time snapshots.

The final time \( T \) is the only parameter that needs to be fixed for the POD based approach. One possible approach is to choose \( T \) so that the norms of each solution \( z_i(t) \) are below a certain tolerance. (Since the system is assumed stable, the solutions must tend to zero for large enough time.) For certain simulation codes, it is possible to choose this tolerance and the simulation will determine at time \( T \). Also, we note that different final times can be taken for each simulation of the dual system when \( p > 1 \), but for simplicity we use one final time \( T \).

In this work, the weights for the approximation of the integral are chosen by the trapezoidal rule, which yielded high accuracy in the projection framework as demonstrated in Section 6.

Next, we focus on the construction of the projection matrix \( V_r \) so that it is orthogonal with respect to the \( E \)-weighted inner product, i.e., \( V_r^T E V_r = I \), where \( E \) is a symmetric positive definite matrix. For a given initial condition \( E^{-1}c^T_1 \), simulate system (18)-(19) and assemble the time snapshots in a matrix

\[
Z_i = [z_i(t_1), z_i(t_2), \ldots, z_i(T)] \in \mathbb{R}^{n \times s}.
\]

Simulations starting with every column of \( E^{-1}c^T_1 \) are concatenated in the matrix

\[
Z = [Z_1, Z_2, \ldots, Z_p] \in \mathbb{R}^{n \times ps}.
\]

Further, the approximate observability Gramian in the new variables can be factored as

\[
X_{T,\delta} = YY^T, \quad Y = Z\Delta^{1/2} \in \mathbb{R}^{n \times ps}.
\]

The projection method only requires the \( W \)-orthogonal eigenvectors of \( X_{T,\delta} \) to construct \( V_r \), so there is no need to form the approximate Gramian explicitly, and we can work with the factor \( Y \) instead. For large systems in which the state space dimension exceeds the number of snapshots, the well known method of snapshots to compute the eigenvectors proceeds as follows. First, compute the eigenvalue decomposition

\[
Y^T E Y = \Phi \Lambda \Phi^T \in \mathbb{R}^{ps \times ps},
\]

and rescale (if necessary) the eigenvectors so that they are orthonormal with respect to the standard inner product, i.e., \( \Phi^T \Phi = I \). Here, \( \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots) \), and the eigenvalues are ordered so that \( \lambda_1 \geq \lambda_2 \geq \cdots \geq 0 \). The projection matrix \( V_r \) is given by the matrix consisting of the first \( r \) eigenvectors of \( X_{T,\delta} \), which is given by

\[
V_r = Y \Phi_r \Lambda_r^{-1/2},
\]
where $\Phi_r \in \mathbb{R}^{n \times r}$ denotes the matrix consisting of the first $r$ columns of $\Phi$, and $\Lambda_r \in \mathbb{R}^{r \times r}$ denotes the matrix $\Lambda_r = \text{diag}(\lambda_1, \ldots, \lambda_r)$. The procedure is summarized in pseudocode in Algorithm 1. Note that the loop computation can be executed in parallel.

**Algorithm 1**: POD method to compute projection matrix

Require: $E, A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$, final time $T$, maximal order $r_{\text{max}}$.

Ensure: Projection matrix $V_{r_{\text{max}}}$.

1: for $i = 1, 2, \ldots, p$ do
2: Simulate $E \dot{z}_i = A^T z_i$, $E z_i(0) = c^T_i$ and place time snapshots of solutions in the matrices $Z_i = [z_i(t_{i1}), z_i(t_{i2}), \ldots, z_i(t_{is_i} = T)]$.
3: Compute quadrature weights (see (20)), and let $\Delta_i = \text{diag}(\delta_{i1}, \delta_{i2}, \ldots, \delta_{is_i})$.
4: end for
5: $Y = [Z_1, \ldots, Z_p] \cdot \text{diag}(\Delta_1, \ldots, \Delta_p)^{1/2}$.
6: Compute the singular value decomposition $Y^T E Y = \Phi \Lambda \Phi^T$.
7: $V_{r_{\text{max}}} = Y \Phi_{1:r_{\text{max}}} \Lambda_{1:r_{\text{max}}}^{-1/2}$.

In exact arithmetic, the projection matrix $V_r$ is orthonormal with respect to the $E$-weighted inner product whenever $\lambda_r > 0$. This can be seen by direct computation: since $\Phi_r^T \Phi_r = I$, we have

$$V_r^T E V_r = \Lambda_r^{-1/2} \Phi_r^T (Y^T E Y) \Phi_r \Lambda_r^{-1/2} = \Lambda_r^{-1/2} \Phi_r (\Phi_r \Lambda_r \Phi_r^T) \Phi_r \Lambda_r^{-1/2} = I.$$

However, as is well known, in finite precision arithmetic accuracy can be lost due to forming the required matrix products. In the large-scale problem considered in Section 6 below, we found that $V_r^T E V_r$ began to deviate significantly from the identity matrix as $r$ increased.

To deal with the loss of $E$-orthogonality, we use a $E$-weighted stabilized Gram-Schmidt procedure to form a new $E$-orthonormal matrix $V_r$ with the same span as the $V_r$ matrix constructed above. To be complete, we present a simple implementation in Algorithm 2 below. We include a reorthogonalization step as this is well known to give greater accuracy in finite precision arithmetic; see, e.g., [33, 57] and the references therein.

**Algorithm 2**: $E$-Weighted Stabilized Gram-Schmidt with Reorthogonalization

Require: $V_r = [v_1, v_2, \ldots, v_r] \in \mathbb{R}^{n \times r}$, $E \in \mathbb{R}^{n \times n}$.

Ensure: $V_r \in \mathbb{R}^{n \times r}$ such that span($V_r$) remains unchanged and $V_r^T E V_r = I \in \mathbb{R}^{r \times r}$.

1: for $\ell = 1, 2$ (reorthogonalization) do
2: for $i = 1, \ldots, r$ do
3: $v_i = v_i / (v_i^T E v_i)^{1/2}$.
4: for $j = i + 1, \ldots, r$ do
5: $v_j = v_j - (v_j^T E v_i) v_i$.
6: end for
7: end for
8: end for
5.1. Convergence of Feedback Gains. In many applications, solving the ARE is only a first step towards computing control or filtering feedback gains. For the ARE (1), the feedback gain is given by $K = B^TPE$. Using the factored form of the approximate solution to ARE $P_r = Z_rZ_r^T$, we can efficiently compute the approximate feedback gain as

$$K_r = (B^T Z_r)(Z_r^T E).$$

To test for convergence of the feedback gain in the large-scale setting, we monitor the convergence of the feedback gains by computing the relative weighted norm of the increment $K_r - K_{r-1}$. To do this, we return to the weighted norm framework of Section 4 and consider the feedback gain $K_r = \hat{B}^* \hat{P}_r$ corresponding to the ARE (13) on $X$. Since $\hat{B}^* = B^T$ and $\hat{P}_r = P_rE$, the feedback gains $K_r$ and $K_r$ are identical.

Consider $K_r$ as a mapping from $X$ to $\mathbb{R}^m$. Using a similar technique to the proof of Proposition 1, it can be shown that

$$\|K_r\|_{\mathcal{L}(X,\mathbb{R}^m)} = \|B^TP_rLE\|_2, \quad E = L_EL_E^T,$$

As above, to compute this norm, we use the factorization $B^TP_rLE = (B_rZ_r)(Z_r^T L_E)$. Therefore, we monitor the convergence of the feedback gains by computing the relative weighted norm of the increment:

$$\kappa_r = \frac{\|K_r - K_{r-1}\|_{\mathcal{L}(X,\mathbb{R}^m)}}{\|K_{r-1}\|_{\mathcal{L}(X,\mathbb{R}^m)}} = \frac{\|B^TP_rLE - B^TP_{r-1}L_E\|_2}{\|B^TP_{r-1}L_E\|_2},$$

where again we use the factorizations of $P_r$ and $P_{r-1}$ in this computation. In the large-scale setting, we stop the projection based Riccati solver when $\kappa_r < \kappa_{tol}$.

6. Numerical Results. In this section, we present numerical results for two different test problems arising from spatial discretizations of PDEs. The first problem considers a spatial discretization of heat transfer in the two-dimensional unit square. This serves as an introductory problem at low state space dimensions, where a direct solution to the ARE is available. The second test problem considers a convection-diffusion equation. Both problems have symmetric positive definite mass matrices $E$.

To give a point of comparison, we also present numerical results using the extended Krylov subspace method (EKSM) [36]. We do note that the POD approach will generally require more computational effort and storage compared to EKSM. However, our primary goal is not computational speed and efficiency, but to move toward a tractable data-based algorithm.

Table 1 contains information and parameters for the test models. The quantity $\lambda_{max}(A,E)$ denotes the (approximate) largest generalized eigenvalue of $A$. Also, $\omega(A,E) := \lambda_{max}(1/2(A + A^T), E)$ denotes the (approximate) generalized numerical abscissa of $A$. The numerical abscissa indicates whether solutions of the system (2)-(3) with no control (i.e., $u(t) = 0$) can experience transient growth before decaying to zero; this transient growth is possible only if the numerical abscissa is positive [67, Sections 14 and 17]. Also see [7] for more about the numerical abscissa and related matrix Lyapunov equations. These quantities were computed in Matlab using the routine eig for the generalized eigenvalue problems $Ax = \lambda Ex$ and $(1/2)(A + A^T)x = \lambda Ex$, respectively. Both test cases are stable systems, however the values of the numerical abscissa indicate that the uncontrolled problems are quite different from each other.
Table 1. Parameters of the two test models.

<table>
<thead>
<tr>
<th>Problem</th>
<th>n</th>
<th>m</th>
<th>p</th>
<th>$\lambda_{\text{max}}(A, E)$</th>
<th>$\omega(A, E)$</th>
<th>$A = A^T$?</th>
<th>$E = E^T$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>diffusion</td>
<td>380</td>
<td>1</td>
<td>1</td>
<td>$-6.2 \times 10^{-1}$</td>
<td>$-6.2 \times 10^{-1}$</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>conv.-diff.</td>
<td>167,690</td>
<td>1</td>
<td>1</td>
<td>$-1.6 \times 10^{-1}$</td>
<td>$7.8 \times 10^{-1}$</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Recall, the first step for solving ARE with projection methods consists of generating a projection matrix $V_r \in \mathbb{R}^{n \times r}$. Next, the reduced-order matrices ($E_r, A_r, B_r, C_r$) are computed via projection and the ARE (9) is solved in $r$ dimensions with the direct solver care in Matlab. This routine solves the Hamiltonian eigenvalue problem associated with ARE and is further described in [4]. The low rank approximation of $P$ is then given by $P_r = Z_rZ_r^T$. A general projection-based algorithm to solve ARE is given in Algorithm 3, where the projection matrix $V_r$ can be any low-dimensional basis in which the ARE solution should be computed. For our proposed POD-based method, the projection space $V_r$, i.e., the input to Algorithm 3 is obtained by using Algorithm 1 and then Algorithm 2 to reorthogonalize.

A general projection algorithm following the steps in Section 3 is used to compare the methods, see Algorithm 3. In all cases, following [60, 36] and the remark in [15, p.9], $\sigma_{\text{tol}} = 10^{-12}$ was used for truncation of the singular values of the low rank solution $\Pi_r$. Therefore, a rank reduction is already applied at the reduced-order level, which is in accordance with the low rank structure of the Riccati solution, and prevents numerical ill-conditioning.

**Remark 1.** We also briefly tested setting $\sigma_{\text{tol}} = 0$ in two versions of the convection diffusion problem (Example 2) with smaller values of $n$ (generated using coarser finite element meshes). Of course, there is no further rank reduction in this case. In these tests, the matrix residual for the POD approach (not shown here) levels off at around two orders of magnitude lower when we use the above tolerance. However, this increases the rank of $P_r$.

**Algorithm 3**: Projection based Riccati solver with residual computation

**Require**: $E, A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{p \times n}$, $V_r \in \mathbb{R}^{n \times r}$, tolerances $\sigma_{\text{tol}} > 0$ and $\kappa_{\text{tol}} > 0$.

**Ensure**: The low rank factor $Z_l$ of $P_l = Z_lZ_l^T$, and matrix residual norm vector $[\gamma_1, \ldots, \gamma_r]$ and feedback convergence vector $[\kappa_1, \ldots, \kappa_r]$.

1: **for** $l = 1, 2, \ldots$ and $\kappa_l > \kappa_{\text{tol}}$ **do**
2:   Let $V_l = V_r(:, 1 : l)$. Compute $E_l = V_l^T EV_l$, $A_l = V_l^T AV_l$, $B_l = V_l^T B$, $C_l = CV_l$.
3:   Solve $A_l^T \Pi_l E_l + E_l^T \Pi_l A_l - E_l^T \Pi_l B_l B_l^T \Pi_l E_l + C_l^T C_l = 0$.
4:   Compute (svd or evd): $\Pi_l = U \Sigma U^T$, where $\Sigma = \text{diag} [\sigma_1, \ldots, \sigma_l]$ and $\sigma_1 \geq \ldots \geq \sigma_l$.
5:   Determine $k$ such that $\sigma_{k+1} < \sigma_{\text{tol}} < \sigma_k$; set $\Sigma_k = \text{diag} [\sigma_1, \ldots, \sigma_k]$, $U_k = U(:, 1 : k)$ and compute $S_l = U_k \Sigma_k^{1/2}$.
6:   $Z_l = V_l S_l$, i.e., the low rank factor of $P_l = Z_lZ_l^T$.
7:   Compute $\gamma_l = \| \mathcal{R}(P_l) \|_{\text{HS}(X)}$ as in Section 4.
8:   Compute feedback gains $K_l = (B^T Z_l) (Z_l^T E)$.
9:   Compute $\kappa_l = \| K_l - K_{l-1} \|_{\mathcal{L}(X, R^m)} / \| K_{l-1} \|_{\mathcal{L}(X, R^m)}$ as in Section 5.1.
10: **end for**
For these example problems, the $E$-weighted norm of a vector equals the $L^2$ norm of the corresponding finite element function. The $L^2$ norm of a function is a natural measure of magnitude these problems; therefore, as discussed earlier, we use the $E$-weighted inner product and norm for these examples. Then, the projection matrix $V_r$ (approximately) satisfies $V_r^T EV_r = I$, and so we simply use an identity matrix in place of the projected matrix $E_r = V_r^T EV_r$.

The first problem was computed on a 2010 MacBook Pro with a 2.66 GHz Intel Core i7 Processor and 4GB RAM. Matlab was used as a software in the version of R2012b. The convection diffusion problem was solved on a computer cluster with two 6-core Intel Xeon X5680 CPU’s at 3.33GHz. The cluster has a Random-Access Memory of 48GB and runs on Scientific Linux 6.4 with Matlab in the Version of 2013b. Machine precision on both machines is in accordance with IEEE double precision standard, $\varepsilon_p = 2 \times 10^{-16}$.

6.1. A diffusion problem in 2D. We consider the diffusion problem in 2D:

$$w_t = \mu (w_{xx} + w_{yy}) + b(x, y)u(t)$$

on $\Omega = [0, 1] \times [0, 1]$ with Dirichlet boundary conditions on the bottom, right and top walls:

$$w(t, x, 0) = 0, \quad w(t, 1, y) = 0, \quad w(t, x, 1) = 0,$$

and Neumann boundary condition on the left wall:

$$w_x(t, 0, y) = 0.$$

We choose $b(x, y) = 5$ if $x \geq 1/2$ and $b(x, y) = 0$ otherwise. The outputs are taken to be

$$\eta(t) = \int_\Omega 5w(t, x, y)dxdy.$$

We discretize the PDE in space with a standard piecewise bilinear finite element method with a uniform $21 \times 21$ node mesh in $\Omega$. After removing the nodes at zero-Dirichlet boundary, this results in a finite-dimensional system with representation $(E, A, B, C)$ is obtained, where $n = 380$ and $m = p = 1$. In this example, the system matrix $A$ is symmetric and has a condition number of $1.4 \times 10^2$, so the problem is numerically well conditioned. The dual system was simulated for 30s with a second order trapezoidal rule integration scheme. The norm of the final snapshot was $4.75 \times 10^{-4}$. This problem is chosen to be small so that for comparison purposes we can compute the solution $P$ via a direct solver (care), as well as the actual weighted Hilbert-Schmidt (Frobenius) norm error in the solution, $\|P - P_r\|_{HS(X)}$, for increasing $r$. We examine the convergence of the feedback gains using the relative weighted norm error to the true gain: $\kappa_r = \|K - K_r\|_{\mathcal{L}(X,R^m)} / \|K\|_{\mathcal{L}(X,R^m)}$ as defined in equation (23), where we use $K$ instead of $K_{r-1}$ to compare to the true gain. We terminate the algorithm once the error in the approximate solution $\kappa_r < \kappa_{tol} = 10^{-11}$. For comparison purposes, the weighted residual norm of the actual solution $P$, computed with a direct solver is $\|R(P)\|_{HS(X)} = 1.5 \times 10^{-11}$.

Figure 1, left, shows the convergence of the feedback gains to the true gain as the rank of the approximation increases. The subspace spanned by the leading singular vectors of the related Lyapunov solution (Gramian) is most rich for approximating the Riccati solution and subsequently computing the feedback gain. With only rank $r = 15$, we were thus able to obtain an error in the Riccati solution of approximately $10^{-8}$, and a highly accurate feedback gain. To obtain similar accuracy with the POD
based approach requires \( r = 23 \), and with projection onto the extended Krylov subspace \( r = 30 \) is required.

\begin{figure}[h]
\centering
\begin{tabular}{cc}
\includegraphics[width=0.45\textwidth]{leftplot.png} & \includegraphics[width=0.45\textwidth]{rightplot.png}
\end{tabular}
\caption{Diffusion model with \( n = 380 \). (left): Convergence of the approximate feedback gain vectors to the true gains for increasing \( r \). (right): Weighted Hilbert-Schmidt (Frobenius) norm of the residual \( \|R(P_r)\|_{HS(X)} \) versus the rank of the approximate solution \( P_r \).
}
\end{figure}

In the small scale setting considered here, we shed some light on the accuracy of the residual as a convergence indicator for the convergence to the true solution. In fact, based on equation (17), the “condition number” \( c(P_r) \) occurs on the upper bound of the true error. In Figure 2, left, the error in the norm of the residual is compared to the error in the solution. Figure 2 then shows the ratio of error in the solution, to norm of the residual, \( \|P - P_r\|_{HS(X)}/\|R(P_r)\|_{HS(X)} \). Consequently, the difference between the residual and the error in the actual solution can be several orders of magnitude apart. Nonetheless, the convergence behavior is correlated, and therefore we can use the residual as an indicator for convergence of the method. We complement this by monitoring the refinement of the feedback gains between subsequent iterations for large-scale problems.

The first step of the POD-based solver is to approximate the Lyapunov solution via quadrature as in (20). In that regard, both the number of snapshots included in the integral approximation, as well as the final time \( T \) to stop simulations influence the accuracy of the approximation. We first vary the number of snapshots when using a second order constant time step ODE solver based on the trapezoid rule. In Figure 3, left, we show both the weighted norm of the residual and the error with respect to the true solution for three different simulations, all until a final time \( T = 6 \): In the coarsest case with \( dt = 0.2 \) there are only 31 snapshots; the intermediate experiment with \( dt = 0.05 \) has 121 snapshots; and the finest time grid with \( dt = 0.01 \) has 601 snapshots. Using more snapshots from the system does improve the results; however, even a coarse collection of snapshots yields fairly accurate approximations of the true solution. Figure 3, right, illustrates the influence of the final time \( T \) on the solution quality of the resulting Riccati solution. For these computations, we kept the number of snapshots fixed at 100 to allow for a similar amount of data, and terminated at different final times. Consequently, the time steps are increasing where \( T \) increases. The plot shows, that while terminating at \( T = 1 \) does not
provide enough information about the dynamics for the subspace to be sufficiently rich, picking the final times as $T = 3$ and $T = 6$ yields almost indistinguishable results.

6.2. A convection diffusion problem in 2D. We consider a convection diffusion equation, which is a common partial differential equation model arising in fluid
dynamics and a variety of other application areas. The problem is given by
\[ w_t = \mu (w_{xx} + w_{yy}) - c_1(x,y)w_x - c_2(x,y)w_y + b(x,y)u(t), \]
where \( c_1(x,y) = x \sin(2\pi x) \sin(\pi y) \) and \( c_2(x,y) = y \sin(\pi x) \sin(2\pi y) \). The boundary conditions, the function \( b(x,y) \), and the output are chosen as in the previously discussed heat equation example. We spatially discretize the PDE with piecewise bilinear finite elements on a uniform \( 411 \times 411 \) node mesh in \( \Omega \). After removing the nodes at zero-Dirichlet boundary, the resulting finite-dimensional system with representation \((E,A,B,C)\) has state dimension \( n = 167,960 \) and the number of inputs and outputs are \( m = p = 1 \). The system matrix \( A \) is non-symmetric with a condition number of \( 1.4 \times 10^6 \). The mass matrix \( E \) is well conditioned, \( \text{cond}(E) \approx 14.2 \). Both condition numbers are estimates computed via \texttt{condest} in \texttt{Matlab} for large, sparse matrices. The generalized eigenvalues, as well as the numerical abscissa are listed in Table 1 above. The numerical abscissa is positive, a key difference to the previous heat equation example, yielding a non-normal system. The dual system was simulated using \texttt{ode23s} in \texttt{Matlab} with default error tolerances and options \texttt{mass} and \texttt{jacobian} set as \( E \) and \( A^T \), respectively. For this model, simulations were performed from \( t = 0 \) to \( T = 40s \). The norm of the final snapshot is \( 8.8 \times 10^{-3} \). The adaptive ODE solver returned \( s = 133 \) snapshots of the solution at nonuniform locations.

Since the state space dimension is large for this example, the Gramian approach as well as the computation of the Riccati solution with a direct method are not feasible. Thus, the relative weighted norm change in the feedback gains \( \kappa_r = \frac{\|K_r - K_{r-1}\|_{L(X,R^m)}}{\|K_r\|_{L(X,R^m)}} \) and the weighted Hilbert-Schmidt residual norm \( \|\mathcal{R}(P_r)\|_{\text{HS}(X)} \) are used as convergence indicators. The stopping criterion for the projection based methods is again set to \( \kappa_{\text{tol}} = 10^{-11} \). Note that the full matrix \( P_r \) is never explicitly used or stored, and we only work with the low rank factor \( Z_r \) from equation (11). In particular, the residual computation is performed as in Subsection 4 and the feedback gain matrices are evaluated as \( K_r = B^T P_r E = (B^T Z_r)(Z_r^T E) \).

Figure 4, left, shows the convergence of the relative change in the feedback gains versus the order of the projection \( r \). As can be seen, both methods eventually converge to that level, where the POD based methods shows faster, and smoother convergence. Accordingly, Figure 4, right, shows the corresponding weighted norm of the matrix residual. We can see once more, that the convergence behavior of the residual, and the updates in the feedback gains have similar characteristics.

We performed multiple numerical studies for the convection-diffusion test problem at different finite element mesh sizes, and found similar convergence behavior of both the residual, as well as the difference in the gains.

7. Conclusion. We presented a new POD based projection approach to compute solutions of algebraic Riccati equations. The method relies on proper orthogonal decomposition to compute an approximation of the solution to the related Lyapunov equation via the algorithm in [74, 64]. The resulting dominant left singular vectors are used in the projection framework to solve algebraic Riccati equations. We also developed new weighted norm residual computations and error bounds. Numerical results demonstrate that this POD basis is sufficiently rich for the projection approach to produce accurate solutions at low solution rank. The proposed approach may be naturally implemented by many researchers in a variety of fields who are already familiar with POD computations.
To put our numerical results in perspective, we compared this POD projection method to the extended Krylov subspace method [36]. It was demonstrated that the POD projection approach can be efficiently computed and can give high accuracy at a low solution rank. The POD approach may generally require more computational effort and storage compared to other similar approaches; however, we emphasize that our primary goal is not offline computation time, but to move toward an efficient, highly accurate, completely data-based algorithm. We do note that it is highly likely that the computational efficiency and storage requirements of the proposed POD projection algorithm can be greatly improved using an incremental/adaptive POD algorithm; see, e.g., [54, 53] and the references therein. We intend to explore this in the near future.

As mentioned in the introduction, for certain applications it is of interest to make the proposed approach completely data-based so that access to system matrices is not required. This can sometimes be done directly for AREs arising from parabolic partial differential equations (such as the second and third example problems considered in Section 6). Briefly, assume the simulation code for the parabolic PDE is based on a bilinear form as in finite element methods and other Galerkin methods. Then one only needs to be able to use the existing simulation code to evaluate the bilinear form acting on the relevant POD modes to project the \( A \) operator (see [62, Section 3.1]). Also, it may be possible to modify the techniques from [62] to make the proposed algorithm completely data-based when the bilinear form is not available or the ARE does not arise from a parabolic PDE. We intend to explore these issues in more detail elsewhere.

Furthermore, a rigorous convergence theory for POD-based approximations to infinite-dimensional Lyapunov solutions is available; see [64]. It may be possible to extend the existing theory to obtain rigorous error estimates for the POD projection method considered here for infinite-dimensional algebraic Riccati equations. Moreover, we performed multiple numerical studies for the convection-diffusion test problem at different finite element mesh sizes, and observed similar convergence behavior of both the residual, as well as the difference in the gains. An interesting
future research question is hence to look into mesh-independence results, similar to
[21].

Acknowledgments. B. Kramer would like to thank Professors John A. Burns and
Daniel B. Szyld for helpful discussions on this topic.

REFERENCES


Received February 2016; 1st revision August 2016; final revision September 2016.

E-mail address: bokramer@mit.edu
E-mail address: singlerj@mst.edu