Masters Thesis:

Reconsidering Solution Methods to the Discrete Algebraic Riccati Equation

Christopher A. Kuo-LeBlanc, Advised by James B. Rawlings

Department of Chemical Engineering, University of California, Santa Barbara

January 31, 2023

1 Introduction

This work provides a comprehensive review of techniques for solving the discrete algebraic Riccati equation (DARE) and guidance for selecting an appropriate method for various user-specified DAREs. The work in this thesis is broadly motivated by process control. From a chemical engineering practitioner's perspective, the overriding motivation for process control is safety, but also includes motives for optimizing profit and reducing process variability. Employing automatic controllers at a chemical plant allows for safe operation without the compromise of human error. Additionally, tools from the state estimation sub-field of process control allow operation with less interference from error in our description and observation of the plant. Controllers also offer a way to closely track user specifications for profit optimizations. Utilizing algorithms from the regulation sub-field of process control offers automatic actions to minimize energy use or closely follow product specifications. Economic model predictive control is another tool that directly optimizes control action around an economic objective, such as profit. Finally, automatic controllers offer a way to reduce variability in process variables such as temperature, pressure, concentration, etc. In the pharmaceutical industry, it is often desirable to limit process variability to protect the integrity of temperature-sensitive reagents or products. To further motivate the exact nature of this thesis, we introduce the linear quadratic regulator and estimation problems as methods of process control.

We have broadly motivated process control, but the objectives in this work are more specific. To appreciate the discussion in this thesis we introduce the linear quadratic regulator and estimation problems, abbreviated as the LQR and LQE respectively. Motivation for solving the LQR and LQE problems are two of the most fundamental problems in control theory with the LQE dating back to Gauss. In optimal control, the objective is to operate a dynamic system at a minimum cost. For the LQR and LQE, a process or plant is modeled by a set of linear algebraic equations with a cost described by a quadratic function known as a linear quadratic problem. For the LQR, the cost function represents the cost to operate at a set of state variables (*e.g.* temperature, pressure, etc.) or perform a control action (*e.g.* apply heat, open/close a valve, etc.). In the LQE, the cost function represents the error between the system model and observations and the true process behavior.

When solved the LQR offers a sequence of control actions that drive the process to the desired operating conditions. The LQE solution provides a minimum error estimate of the true system behavior. It turns out the LQR and LQE are both solved by evaluating the stabilizing solution to a nonlinear matrix equation called the discrete algebraic Riccati equation (DARE). The DARE is described by:

$$X = A'XA + Q - (B'XA + S')'(B'XB + R)^{-1}(B'XA + S') \tag{1}$$

In this work, we review solution methods for the DARE. Since its development in the early 1980s, the generalized Schur vector method (GSV) has been the gold standard for computing the solution to the DARE. We offer a comparison of three historically popular methods: the iterative Riccati equation (IRE), the GSV, and Newton's method. We discuss the algorithms in further detail in Section 2. We find that the IRE computes the solution to the DARE quickly, but at the trade-off of accuracy for large systems. Additionally, the IRE is particularly susceptible to numerical instabilities, namely in the stabilizability of the system. The GSV computes the DARE solution to high accuracy but scales poorly with system size. Lastly, Newton's method is ineffective as a stand-alone method as it requires a good initial guess close to the solution. We offer two operating methods that utilize Newton's method as a refinement technique that overcome the drawbacks of the IRE for large systems and are robust to numerical instability. Numerical performance is discussed in Section 4.

Furthermore, we lay the foundation for future work towards implementing Newton's method for the special case of the DARE with singular R. While not particularly relevant to the LQR, the DARE with singular R has applications in the LQE for handling systems with linear constraints and exact measurements. For the stricter case where $R = 0$, we propose that Newton's method is largely unchanged. We follow up in Section 7 with questions to instigate discussion towards the more general case of Newton's method for DAREs with singular R. We move to a technical view of the DARE formulation and its solution methods.

1.1 Linear Dynamical Systems

We start our discussion by establishing how processes are described as linear dynamical systems. A linear dynamical system is the time-dependent description of a process' variables (*e.g.* temperature, pressure, concentration) as a linear function. In a process control setting, linear dynamical systems often take the form:

$$\begin{aligned} x(k+1) =& Ax(k) + Bu(k), \\ y(k) =& Cx(k), \end{aligned}$$

where x are the process variables, u are the control actions, y are measurements of the system, A, B, C are matrices, and k is the k-th discrete increment of time (*i.e.* for some discrete measure of time Δt, $x(k)$ is the process variables at time $k\Delta t$). From here on we denote the time index with a subscript, *e.g.*, $x(k) = x_k$.

1.2 The LQR Problem

The LQR problem for a linear dynamical system is described by:

$$x_{k+1} = Ax_k + Bu_k,$$

is formulated by developing a quadratic performance index J given by:

$$J = x'_{H-1}Qx_{H-1} + \sum_{k=0}^{H-1} \left(x'_k Q x_k + u'_k R u_k + 2x'_k S u_k \right),$$

where H is the time horizon for which control action should be determined, Q, R, S are user-defined matrices determining the cost of the state operation, control action, and state-control interactions respectively. The optimal control sequence is found by minimizing the performance index J and is given by $u(k) = K(X_k)x_k$, where

$$K(X_k) = -(B'X_kB + R)^{-1}(B'X_kA). \tag{2}$$

As found by Kalman (1960), X_k is found through backward iteration of the dynamic Riccati equation:

$$X_{k-1} = A'X_kA + Q - (B'X_kA + S')'(B'X_kB + R)^{-1}(B'X_kA + S').$$

For the infinite-horizon, $H = \infty$, the performance index becomes:

$$J = \sum_{k=0}^{\infty} \left(x_k'Qx_k + u_k'Ru_k + 2x_k'Su_k\right),$$

and the stabilizing solution X to $u_k = K(X)x_k$, is the X that satisfies the DARE in Eqn. 1 such that $|\max(\lambda(A + BK(X)))| < 1$.

1.3 The LQE Problem

The LQE (also known as the LQ Gaussian) problem concerns linear systems driven by additive white Gaussian noise,

$$\begin{aligned} x_{k+1} =& Ax_k + w_k, \\ y_k =& Cx_k + v_k, \end{aligned}$$

where w_k, v_k are the process and measurement noise respectively. The infinite horizon problem has a performance metric given by:

$$J = \mathbb{E}\left[\sum_{k=0}^{\infty} \left((x_{k+1} - AX_k)'Q(x_{k+1} - AX_k) + (y_k - Cx_k)'R(y_k - Cx_k)\right)\right],$$

where Q, R are the process and measurement noise covariances (Rawlings et al. (2017), Ch.

1.4). The estimate update is given by:

$$\hat{x}_{k+1} = A\hat{x}_k + L_{k+1}(y_{k+1} - CA\hat{x}_k),$$

where $\hat{x}$ is the state estimate and $L = XC'(CXC' + R)^{-1}$ is the Kalman gain (Kalman, 1960). The estimate update is computed from finding the stabilizing solution, X, from the DARE:

$$X = AXA' + Q - (CXA')'(CPC' + R)^{-1}(CXA').$$

By computing the state estimate update, the LQE uses past and current measurement information to produce the minimum error estimate for the state.

1.4 The DARE

The solution to the DARE offers an optimal control sequence for plant regulation in the context of the LQR and a best estimate of the states given previous measurements in the context of the LQE. For discussion of the algorithms in Section 2, we use the simplified form:

$$X = A'XA + Q - (B'XA)'(B'XB + R)^{-1}(B'XA). \tag{3}$$

The goal is to find the solution X that satisfies Eqn. (3). The DARE, however, is a nonlinear algebraic matrix equation and may have more than one suitable solution X. We further qualify that the solution must be stabilizing, *i.e.*, that X satisfies $A + BK(X)$ for the LQR ($A - L(X)C$ for the LQE) has eigenvalues with magnitude less than one. When quantifying the performance of an algorithm, we consider accuracy and the time to solution. Accuracy is relevant for determining how close the algorithm's result is to the stabilizing solution. Second, we prioritize time to solution for the obvious reason that it is undesirable to wait long periods of time for a solution. Next, we describe solution methods for the DARE.

2 DARE Solvers

There are a variety of DARE solvers that have evolved over the years. We focus on the GSV, IRE, and Newton's method. In this section, we briefly discuss the formulation for the GSV and IRE with a more in-depth review of Newton's method. Beyond this, we introduce two switch methods that primarily use the IRE followed by Newton's method as a refinement technique. These we will call the $A + BK$ stability switch and the IRE proximity switch.

2.1 The Generalized Real-Schur Vector Method

The GSV was developed in the early 1980s (Pappas et al., 1980; Van Dooren, 1981) and is thoroughly reviewed in Datta (2004) (see Ch. 13). In a brief overview, the GSV forms the symplectic matrix pencil:

$$P = \begin{bmatrix} A & 0 \\ -Q & I \end{bmatrix}, \quad N = \begin{bmatrix} I & BR^{-1}B' \\ 0 & A' \end{bmatrix}. \tag{4}$$

The solution of the DARE is found through solving the generalized eigenvalue problem given by $P - \lambda N = 0$. To compute the solution, a QZ algorithm is imposed to find orthogonal matrices Q', Z such that

$$Q'(P - \lambda N)Z = P_1 = \begin{bmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{bmatrix}$$

and

$$Q'(P - \lambda N)Z = N_1 = \begin{bmatrix} N_{11} & N_{12} \\ 0 & N_{22} \end{bmatrix}.$$

For Q', Z that satisfy the equations above and have generalized eigenvalues of $P_{11} - \lambda N_{11}$ with modulii less than 1, let Z be partitioned as

$$Z = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix}.$$

Then, the columns are the stabilizing solution of Eqn. (3) is given by $X = Z_{21}Z_{11}^{-1}$ (Datta, 2004). It is worth noting that the GSV is computing the solution to a $2n$ generalized eigenvalue problem which implies it has a computation complexity of order $\mathcal{O}(10n^3)$.

2.2 The Iterative Riccati Equation

The IRE dates back to 1960 from Kalman (1960). The algorithm iterates the discrete Riccati equation as follows:

$$X_{k+1} = A'X_kA + Q - (B'X_kA)'(B'X_kB + R)^{-1}(B'X_kA),$$

for the LQR. The IRE is iterated in the infinite horizon problem until $\|X_{k+1} - X_k\|$ is sufficiently small. The IRE converges exponentially fast (Caines and Mayne, 1970; Anderson and Moore, 1979). Since each iterate contains an inverse, the computational complexity of each iteration should be of order $\mathcal{O}(n^3)$.

2.3 Newton's Method

Newton's method was created as an iterative root-finding method. The derivation is based on manipulations of a first-order Taylor series expansion. Newton's method for the DARE is given an in-depth explanation in the following sections.

2.3.1 A Brief History of Newton's Method for the DARE

Newton's method is a well-explored solution to the DARE. Developed in Hewer (1971) as a discrete analog to Kleinman (1974); Hewer's algorithm was motivated by Newton's method's quadratic convergence in search of a quick iterative technique for solving the DARE. Hewer's DARE exclusive

Newton's method was proven to have quadratic convergence in Lancaster and Rodman (1995) (pp. 308-310) for nonsingular R (see Mehrmann (1991)). While not popular as a stand-alone technique, Newton's method is viable as an iterative refinement tool (Datta (2004), pp. 574-579). Benner and Faßbender continued the investigation of Newton's method fairly recently in a thorough review of numerical improvements in Benner and Faßbender (2011). Before exploring the mechanism of Hewer's algorithm, we must first understand the Fréchet derivative.

2.3.2 The Fréchet Derivative

The application of Newton's method is relatively straightforward for a scalar function, $f(x)$. Previously, we needed to evaluate the derivative $f'(x)$. For the DARE, however, we need a more general definition of the derivative. The Fréchet derivative is a generalization of the gradient to arbitrary vector spaces (Long, 2009).

Definition 1 (The Fréchet Derivative.)**.** *Let $f : V \mapsto U$ be a function and let $h \neq 0$ and x be vectors in V. The Fréchet derivative Df of f in the direction h at point x_0, $D[f(x)]_{x_0}h$ is defined implicitly by*

$$f(x_0 + \epsilon h) = f(x_0) + \epsilon D[f(x)]_{x_0}h + h\mathcal{O}(\epsilon).$$

In the limit $\epsilon \to 0$, the $D[f(x)]_{x_0}$ exists if there is a linear operator A, such that,

$$Ah = \lim_{\epsilon \to 0} \frac{f(x + \epsilon h) - f(x)}{\epsilon}, \quad \forall h.$$

Then, the Fréchet derivative of f is given by $D[f(x)]_{x_0} = A$.

While the Fréchet derivative is defined here for arbitrary vector spaces, it directly extends to matrix functions of matrices. Def. (1) is equivalent to the vectorized matrix differential defined in Magnus and Neudecker (2019) Def. 5.3. We move to state some useful properties of the Fréchet derivative.

2.3.3 Useful Properties of the Fréchet Derivative

The Fréchet derivative, by definition, is a linear operator. The defining qualities of a linear operator for the derivative of the matrix function $F : V \mapsto W$ for $X \in V$ are:

- Additivity:

$$D[F(X)]_{X_0}(A + B) = D[F(X)]_{X_0}A + D[F(X)]_{X_0}B, \tag{5}$$

- Homogeneity:

$$D[F(X)]_{X_0}(\alpha A) = \alpha D[F(X)]_{X_0}A,$$

for $A, B \in V$ and some scalar α. Additionally, the chain rule holds for the Fréchet derivative (see Magnus and Neudecker (2019) Thm. 5.12):

$$D[(G \circ F)]_{X_0}A = D[G(X)]_{F(X_0)} \circ D[F(X)]_{X_0}A, \tag{6}$$

for matrix functions $F : V \mapsto W$ and $G : W \mapsto S$ where $X, A \in V$. It follows from the chain rule that the product rule also holds:

$$D[F(X)G(X)]_{X_0}A = D[F(X)]_{X_0}AG(X_0) + F(X_0)D[G(X)]_{X_0}A, \tag{7}$$

for matrix functions $F, G : V \mapsto W$ where $X, A \in V$. Utilizing additivity, chain rule, and product rule, we move to derive a few elementary cases that aid in the derivation of Newton's method for the DARE.

2.3.4 Elementary Fréchet Derivatives

Recall the form of the DARE in Eqn. (3):

$$X = A'XA + Q - (B'XA)'(B'XB + R)^{-1}(B'XA).$$

From the additivity property and the product rule, we can break the dare into four cases: constant matrix functions, linear matrix functions, affine matrix functions, and inverse matrix functions. Starting with constant matrix functions, the derivation is straightforward:

Example: 1 (Constant Matrix Functions.)**.** *Consider the constant matrix function* $F(X) = C$ *where* $C \in V$*,*

$$D[C]_{X_0}H = \lim_{\epsilon \to 0} \frac{1}{\epsilon}(C - C) = 0 \tag{8}$$

Next, we observe the general class of linear matrix functions. The derivation follows:

Example: 2 (Linear Matrix Functions.)**.** *Consider the linear matrix function* $F(X) = AXB$*,*

$$\begin{aligned} D[AXB]_{X_0}H =& \lim_{\epsilon \to 0} \frac{1}{\epsilon}(A[X_0 + \epsilon H]B - AX_0B), \\ =& AHB. \end{aligned} \tag{9}$$

Affine matrix functions directly follow from the application of the additivity property in Eqn. (5) to a combination of linear matrix functions and constant matrix functions.

Example: 3 (Affine Matrix Functions.)**.** *Consider the affine matrix function* $F(X) = AXB + C$*. From our additivity property, we have*

$$D[AXB + C]_{X_0}H = D[AXB]_{X_0}H + D[C]_{X_0}H.$$

But we have the solution to each of these derivatives from Eqn. (8) and Eqn. (9),

$$D[AXB + C]_{X_0}H = AHB.$$

Finally, we consider inverse matrix functions. The derivation requires a trick using the product rule as shown in the following example .

Example: 4 (Inverse Matrix Functions)**.** *We want to compute the Fréchet derivative of the inverse matrix functions $F^{-1}(X) = [F(X)]^{-1}$. Rather than directly computing the derivative, first, consider the domain $X \in V$ where $F(X)$ has an inverse. It follows that $I = F^{-1}(X)F(X)$ for $X \in V$, where I is the identity matrix of appropriate size. If we choose $X_0 \in V$, the derivative of the identity matrix is then given by.*

$$D[I]_{X_0}H = D[F^{-1}(X)F(X)]_{X_0}H.$$

Since the identity matrix is a constant matrix function, Eqn. (8) and product rule yields

$$\begin{aligned} 0 =& D[F^{-1}(X)F(X)]_{X_0}H, \\ =& D[F^{-1}(X)]_{X_0}HF(X_0) + F^{-1}(X_0)D[F(X)]_{X_0}H. \end{aligned}$$

Given that we chose X_0 so that $F^{-1}(X_0)$ exists, we can rearrange for the desired derivative:

$$D[F^{-1}(X)]_{X_0}H = -F^{-1}(X_0)D[F(X)]_{X_0}HF^{-1}(X_0).$$

Equipped with our four elementary cases, we can move to derive Newton's method for the DARE.

2.3.5 Newton's Method for the DARE: Derivation

Due to the relevance in Section 5, we derive the mechanics of Hewer's algorithm. To solve the DARE, we reform our algebraic equation (3) into the following matrix function:

$$F(X) = A'XA - X + Q - (B'XA)'(B'XB + R)^{-1}(B'XA) \tag{10}$$

With the form in Eqn. (10), we can deploy Newton's method to find $F(X) = 0$. This problem formulation is nearly equivalent to finding a suitable X for the DARE in Eqn. (3). To construct an appropriate Newton's method, we need a form for functions on arbitrary vector spaces. The iterative scheme can be derived from a Taylor series expansion similar to the scalar function variant. Observe the following iterative scheme (see Yamamoto (2000) Eqn. 2.3):

$$X_{k+1} = X_k - [D[F(X)]_{X_k}]^{-1} F(X_k). \tag{11}$$

Prior to using the algorithm, we need to compute the derivative of the function $F(X)$ in Eqn. (10). Utilizing the additivity property in Eqn. (5) and the product rule in Eqn. (7), we can separate Eqn. (10) into six elementary functions:

- $F_1(X) = A'XA$,
- $F_2(X) = X$,
- $F_3(X) = Q$,
- $F_4(X) = (B'XA)'$,
- $F_5(X) = (B'XA)$,
- $F_6(X) = B'XB + R$.

Recall the gain from the LQR problem in Eqn. (2). With some simplification, we can apply examples 1-4 to produce the following derivative,

$$D[F(X)]_{X_0} H = [A + BK(X_0)]' H [A + BK(X_0)] - H. \tag{12}$$

Returning to the iterative scheme in Eqn. (11), we can rearrange our equation to $X_{k+1} - X_k = -[D[F(X)]_{X_k}]^{-1} F(X_k)$. To complete an iteration of Newton's method, we need to invert the derivative we computed onto the function in Eqn. (10) evaluated at the previous iterate to compute the error between iterates. Equivalently, we need to solve the following discrete Lyapunov equation to compute the error $N_k = X_{k+1} - X_k$:

$$[A + BK(X_k)]' N_k [A + BK(X_k)] - N_k = -F(X_k).$$

Adding the error N_k to the current iterate X_k produces the next iterate X_{k+1}. This concludes the algorithm described in Hewer (1971). The derivation provided above extends to the DARE in Eqn. (1).

$$F(X) = A'XA - X + Q - (B'XA + S')'(B'XB + R)^{-1}(B'XA + S'). \tag{13}$$

The method we derived here is equivalent to Benner and Faßbender (2011) and Datta (2004) (pp. 574-575). Newton's method computes the solution to a discrete Lyapunov equation whose computational complexity is of order $\mathcal{O}(n^3)$.

2.4 The $A + BK$ Stability Switch and the IRE Proximity Switch

Here we propose utilizing Newton's method as a refinement technique for the IRE. Newton's method struggles as a stand-alone method since it requires an initial guess that satisfies $|\max(\lambda(A + BK(X_0)))|$. Newton's method is unable to converge to the stabilizing solution in Section 4.1. We develop two switch criteria for the transition from the IRE to Newton's method. The first we call the $A+BK$ stability switch (ABKSS). For each iteration of the IRE, a check is added to determine the maximum modulus eigenvalue of the matrix $A+BK(X_k)$. When the maximum modulus eigenvalue is less than 1, the current iterate X_k is substituted into Newton's method as the initial iterate.

We additionally propose the IRE proximity switch as an extension of the ABKSS to utilize the exponential convergence for as long as possible. The switch criteria are instead subject to the proximity of the iterate to the solution. When the difference $\|X_k - X_{k-1}\|$ is less than some tolerance ϵ, the current iterate X_k is substituted into the initial guess for Newton's method. We move to detail the methods for our numerical study in Section 4.

3 Methods

In this section, we detail the software and algorithms used for the numerical examples in Section 4. Furthermore, we elaborate on the methods for generating the examples. All numerical results were produced in GNU Octave, version 6.4.0. with the Control package 3.4.0. All algorithms except the GSV were developed here and use an initial guess $X_0 = Q$.

3.1 Algorithms

In our numerical studies, we compare five algorithms.

- The GSV
- Newton's Method
- The IREPS
- The IRE
- The ABKSS

The IRE, Newton's method, ABKSS, and IREPS are will be considered iterative algorithms in this context. The iterative schemes are terminated when the final iterate is sufficiently close to the solution. Termination criteria are further discussed in Section 3.3.1.

3.1.1 The GSV

The first algorithm is the GSV and acts as a control for contrast. For this study we use the Octave command, `dare`, the default DARE solver for Octave. The GSV generally uses the following pseudocode:

Algorithm 1 GSV for the DARE

1: Form the symplectic pencil

$$P = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix}, \quad N = \begin{pmatrix} I & S \\ 0 & A' \end{pmatrix}$$

2: QZ-factorize the pencil such that P_1 is quasi-upper triangular and N_1 is upper triangular:

$$Q_1 P Z_1 = P_1 = \begin{pmatrix} P_{11} & P_{12} \\ 0 & P_{22} \end{pmatrix}, \quad Q_1 N Z_1 = N_1 = \begin{pmatrix} N_{11} & N_{12} \\ 0 & N_{22} \end{pmatrix}.$$

3: Find orthogonal matrices Q_2, Z_2 s.t. $|\lambda(N_1)| < 1$ and form

$$Z = Z_1 Z_2 = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix}.$$

4: Compute $X = Z_{21} Z_{11}^{-1}$.

Note that the code `dare` utilizes Newton's method as a refinement technique and has a built-in switch to the extended symplectic pencil for DAREs with singular R.

3.1.2 The IRE

Our second algorithm is the IRE for which we use in-house code developed in Octave. The pseudocode is given by:

Algorithm 2 The IRE for the DARE

1: Initialize some initial guess X_0

2: $X_k = X_0$

3: **while** $\frac{\|F(X_k)\|_F}{\|X_k\|_F} > \epsilon$ **do**

4: Iterate the DARE s.t.:

$$X_{k+1} = A'X_kA + Q - (B'X_kA)'(B'X_kB + R)^{-1}(B'X_kA)$$

5: **end while**

3.1.3 Newton's Method

The third algorithm is Newton's method which will be abbreviated as NM in future tables. We use an in-house code developed in Octave. The pseudocode is given by:

Algorithm 3 Newton's Method for the DARE

Input: A, B, Q, R, S and X_0 such that $\max|\sigma(A + BK(X_0))| < 1$ and $B'XB + R > 0$

Output: X_{k+1}, N_k, where $N_k = X_s - X_{k+1}$ with X_s as the stabilizing solution of $F(X) = 0$.

1: Initialize some initial guess X_0

2: $X_k = X_0$

3: **while** $\frac{\|F(X_k)\|_F}{\|X_k\|_F} > \epsilon$ **do**

4: Solve $[A + BK(X_k)]' N_k [A + BK(X_k)] - N_k = \mathcal{R}(X_k)$

5: $X_{k+1} = N_k + X_k$

6: Set $X_k = X_{k+1}$

7: **end while**

3.1.4 The ABKSS

Fourth, we have the in-house ABKSS code developed in Octave. The pseudocode is a combination of the IRE followed by Newton's method as a refinement technique once the IRE terminates with a

stable iterate. That is, we switch to refinement once $A + BK(X_k)$ is stable using the iterate X_k as the initial guess for Newton's method. The pseudocode is given by:

Algorithm 4 ABKSS

1: Initialize some initial guess X_0

2: $X_k = X_0$

3: **while** $\max|\sigma(A + BK(X_0))| > 1$ **do**

4: Iterate the DARE s.t.:

$$X_{k+1} = A'X_kA + Q - (B'X_kA)'(B'X_kB + R)^{-1}(B'X_kA)$$

5: **end while**

6: **while** $\frac{\|\mathcal{R}(X_k)\|_F}{\|X_k\|_F} < \epsilon$ **do**

7: Solve $[\hat{A}(X_k)]'N_k[\hat{A}(X_k)] - N_k + F(X_k) = 0$

8: $X_{k+1} = N_k + X_k$

9: Set $X_k = X_{k+1}$

10: **end while**

3.1.5 The IREPS

Lastly, the IREPS algorithm is an in-house algorithm developed in Octave. The pseudocode is a combination of the IRE followed by Newton's method as a refinement technique. The switch criterion has two necessary conditions: $1)$ the iterate must be stable and $2)$ the iterate must be sufficiently close to the solution. Close-ness to the solution is defined in this case as IRE reaching maximum numerical accuracy or the user defined accuracy. The pseudocode is given by:

Algorithm 5 IREPS

Initialize some initial guess X_0

$X_k = X_0$

while $\max|\sigma(A + BK(X_0))| > 1$ & $\left|\frac{\|F(X_{k+1})\|_F}{\|X_{k+1}\|_F} - \frac{\|F(X_k)\|_F}{\|X_k\|_F}\right| > \delta$ **do**

    Iterate the DARE s.t.:

$$X_{k+1} = A'X_kA + Q - (B'X_kA)'(B'X_kB + R)^{-1}(B'X_kA)$$

end while

while $\frac{\|F(X_k)\|_F}{\|X_k\|_F} < \epsilon$ **do**

    Solve $[\hat{A}(X_k)]'N_k[\hat{A}(X_k)] - N_k + F(X_k) = 0$

    $X_{k+1} = N_k + X_k$

    Set $X_k = X_{k+1}$

end while

3.2 Problem Generation

We consider two types of problems in our numerical studies:

- Randomly generated DAREs,
- Barely stabilizable DAREs.

In this section, we discuss our method of generating the examples in Section 4.

3.2.1 Ranomly Generated DAREs

For the randomly generated dares we are mainly concerned with how an algorithm's performance scales with system size. Let n, m denote the number of system states and inputs/measurements respectively. Then, $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are randomly generated using the octave command `rand`, *e.g.*, `A=rand(n)`. To generate matrices $Q \in \mathbb{R}^{n\times n}, R \in \mathbb{R}^{n\times m}, S \in \mathbb{R}^{m\times m}$, a matrix $P \in \mathbb{R}^{(n+m)\times(n+m)}$ is randomly generated such that,

$$PP' = \begin{bmatrix} Q & S \\ S' & R \end{bmatrix}.$$

We use PP' to force positive definiteness of the resulting matrices Q, R. Note that since `rand` draws from a continuous support, there is a zero probability of generating singular Q or R.

3.2.2 Barely Stabilizable DAREs

To study the performance of our algorithms on barely stabilizable systems, consider the following example:

$$A = \begin{pmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.9 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-10^{-d} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1-10^{-d} & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-10^{-d} \end{pmatrix}, B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 10^{-d} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 10^{-d} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$Q = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \end{pmatrix}, R = \begin{pmatrix} 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.2 \end{pmatrix},$$

where we can tune the parameter d. As d becomes large, the system becomes more difficult to control. At $d = 16$ we reach machine precision and the terms scaling with d in A become 1 and the terms scaling with d in B become 0. As a result, the influence of the inputs on the system dynamics is inversely correlated with d.

3.3 Result Interpretation

We consider three measures of performance for each algorithm:

- Scaled residual,
- Time to solution,
- Number of iterations.

3.3.1 Scaled Residual

The scaled residual is a measure of how close our solution satisfies the DARE. The residual of the solution X from the GSV is given by:

$$\|F(X)\|_F = \|A'XA - X + Q - (B'XA + S')'(B'XB + R)^{-1}(B'XA + S')\|_F,$$

where the $\|(\,\cdot\,)\|_F$ is the Frobenius norm and is computed as the sum of the square of the elements of the argument. The residual of the k-th iterate for the IRE, Newton's method, ABKSS, and IREPS is given by:

$$\|F(X_k)\|_F = \|A'X_kA - X_k + Q - (B'X_kA + S')'(B'X_kB + R)^{-1}(B'X_kA + S')\|_F.$$

For a large enough system size, the residual can be arbitrarily large even if we are close to the solution. So we instead measure the scaled residual, which is given by:

$$\frac{\|F(X)\|_F}{\|X\|_F} \quad \text{or} \quad \frac{\|F(X_k)\|_F}{\|X_k\|_F}.$$

For the studies conducted in this work, we say a solution is sufficiently accurate if the scaled residual is smaller than the square root of machine precision, 1.490×10^{-8}.

3.3.2 Time to Solution

The time to solution is computed as the difference between the wall clock time when the algorithm starts and computes a sufficient solution. The time for each algorithm to complete a solution is an insightful measure of how quickly a DARE can be solved in real-time.

3.3.3 Number of iterations

The number of iterations is a measure of the number of steps an algorithm needs to compute to reach a sufficient solution. This measurement is reserved for the IRE, Newton's method, ABKSS, and IREPS. While the GSV does employ an iterative method to compute eigenvalues, we do not consider it an iterative scheme for computing the solution to the DARE. The number of iterations provides insight into how much each loop corrects the initial guess.

4 Performance

4.1 Randomly Generated Example

We consider three randomly generated examples of variable system size:

1. $n = 100, m = 50;$ 2. $n = 500, m = 250;$ 3. $n = 1000, m = 500;$

where n, m are the number of states and inputs/measurements respectively. Our results are split into three tables for the scaled residual, time to solution, and the number of iterations. From Table 1. we observe that nearly all algorithms compute a solution to the desired accuracy. The exceptions are Newton's method and the IRE. Newton's method by design needs an initial guess that satisfies $|\max(\lambda(A + BK(X_0)))| < 1$. It is unlikely to guess a stabilizing iterate and as a result, none of the Newton's method iterations converge to the stabilizing solution. We omit Newton's method as

a stand-alone method in future studies. The IRE only fails to meet a sufficient scaled residual on the system with $n = 1000$.

Table 1: The scaled residual measures how close Eqn. (10) is to zero and is given by the Frobenius norm of the function evaluated at the solution divided by the Frobenius norm of the solution.

Method	1	2	3
IRE	7.6×10^{-9}	1.1×10^{-8}	7.2×10^{-8}
NM	1.7×10^{2}	9.8×10^{2}	2.0×10^{3}
ABKSS	6.1×10^{-9}	4.3×10^{-12}	6.9×10^{-11}
IREPS	3.8×10^{-13}	4.3×10^{-12}	1.2×10^{-11}
GSV	3.3×10^{-13}	8.3×10^{-12}	4.5×10^{-11}

From Table 2. we find that for small systems, $n < \mathcal{O}(10^3)$ computation times are relatively small. For the best accuracy, one should choose the GSV or IREPS for small problems. As the system size increases, the GSV computation time significantly increases. The clear choice for larger systems is IREPS computing the most accurate solution an order of magnitude faster than the GSV.

Table 2: The time to solution is given in seconds and describes the wall clock time that the algorithm takes to reach the solution or the chosen tolerance.

Method	1	2	3
IRE	8.9×10^{-3}	2.5×10^{-1}	3.1×10^{0}
NM	1.4×10^{-1}	7.7×10^{0}	6.8×10^{1}
ABKSS	8.9×10^{-2}	2.4×10^{0}	2.9×10^{1}
IREPS	2.6×10^{-2}	7.3×10^{-1}	5.4×10^{0}
GSV	5.4×10^{-2}	6.9×10^{0}	6.9×10^{1}

Table 3. adds an interesting detail to the story. From the third system with $n = 1000$, the IRE reaches the maximum number of iterations $k = 30$. Extending the maximum number of iterations to

$k = 1000$, the IRE only reaches a scaled residual of 7.86×10^{-8}. Not only is there no improvement, the residual for $k = 1000$ is also worse than for $k = 30$. Thus, the maximum accuracy of the IRE for the $n = 1000$ problem has been reached and any further changes are likely due to truncation errors. While the reason is unknown, it is likely a numerical instability preventing the IRE from reaching the desired accuracy. We explore further in a study of system size scaling.

Table 3: The number of iterations that each solver needs to reach the chosen accuracy.

Method	1	2	3
IRE	1.0×10^1	9.0×10^0	3.0×10^1
NM	1.6×10^1	1.6×10^1	1.6×10^1
ABKSS	8.0×10^0	8.0×10^0	8.0×10^0
IREPS	1.0×10^1	9.0×10^0	9.0×10^0
GSV	N/A	N/A	N/A

4.2 System Size Scaling

For the system size scaling study, we consider an extension of the previous problem:

1. $n = 10, m = 5$;
2. $n = 50, m = 25$;
3. $n = 100, m = 50$;
4. $n = 500, m = 250$;
5. $n = 1000, m = 500$;
6. $n = 2000, m = 1000$.

Additionally, for each system size, ten DAREs are randomly generated. We present the averaged scaled residual, time to solution, and the number of iterations in Tables 4-6. respectively. Starting with Table 4., the IRE is the only algorithm to fail to meet the desired accuracy for a system size of $n = 2000$. Moreover, the IRE reaches the desired accuracy for all ten randomly generated DAREs at $n = 1000$.

Table 4: The scaled residual measures how close Eqn. (10) is to zero and is given by the Frobenius norm of the function evaluated at the solution divided by the Frobenius norm of the solution.

Method	1	2	3	4	5	6
IRE	1.2×10^{-8}	7.0×10^{-9}	5.3×10^{-9}	2.3×10^{-9}	4.5×10^{-9}	2.9×10^{-8}
ABKSS	4.9×10^{-10}	1.0×10^{-12}	5.5×10^{-13}	1.7×10^{-11}	2.8×10^{-11}	7.6×10^{-11}
IREPS	1.8×10^{-12}	7.3×10^{-13}	5.4×10^{-13}	8.6×10^{-12}	2.8×10^{-11}	7.6×10^{-11}
GSV	9.3×10^{-15}	6.9×10^{-14}	4.4×10^{-13}	6.9×10^{-12}	2.2×10^{-11}	6.3×10^{-11}

Moving to the time to solution in Table 5., we observe a similar trend to the previous study. The computation time for the GSV drastically increases with system size. While the ABKSS is faster than the GSV for large systems, the IREPS computes the solution over an order of magnitude faster than the GSV for systems with 500 or more states. This improves to two orders of magnitude faster at systems of 5000 states.

Table 5: The time to solution is given in seconds and describes the wall clock time that the algorithm takes to reach the solution or the chosen tolerance.

Method	1	2	3	4	5	6
IRE	2.4×10^{-3}	5.1×10^{-3}	1.1×10^{-2}	2.7×10^{-1}	1.4×10^{0}	2.0×10^{1}
ABKSS	9.3×10^{-3}	2.3×10^{-2}	9.6×10^{-2}	2.5×10^{0}	2.0×10^{1}	1.7×10^{2}
IREPS	9.7×10^{-3}	7.8×10^{-3}	2.1×10^{-2}	7.4×10^{-1}	5.9×10^{0}	4.6×10^{1}
GSV	1.9×10^{-3}	7.5×10^{-3}	5.2×10^{-2}	7.0×10^{0}	7.2×10^{1}	6.1×10^{2}

Table 6. has a couple interesting entries. Most importantly we observe the IRE reaching maximum iteration counts for systems with more than 1000 states. Furthermore, both ABKSS and IREPS compute the solution with high accuracy with only a few iterations for large systems size. It is recommended for large systems sizes ($n > 500$) to use an alternative to the GSV. For very large systems it is advised to opt for an IRE-based method with a Newton refinement switch. While it

seems unlikely that the IRE will struggle to meet the desired accuracy in systems with 1000 states, it is a safer option to always opt for a Newton refinement to guarantee higher accuracy and less numerical instability.

Table 6: The number of iterations that each solver needs to reach the chosen accuracy.

Method	1	2	3	4	5	6
IRE	2.3×10^1	2.0×10^1	1.4×10^1	1.0×10^1	9.0×10^0	3.0×10^1
ABKSS	5.0×10^0	8.0×10^0	9.0×10^0	7.0×10^0	7.0×10^0	7.0×10^0
IREPS	1.8×10^1	1.7×10^1	1.3×10^1	9.0×10^0	9.0×10^0	8.0×10^0
GSV	N/A	N/A	N/A	N/A	N/A	N/A

4.3 Barely Stabilizable Systems

The barely stabilizable system described in Section 3.2.2 allows us to observe algorithm performance as the system becomes close to uncontrollable. As d approaches machine precision ($d = 16$), our computers are unable to identify that the eigenvalues of A are less than one and that the influence of the inputs in B is greater than zero. Table 7. demonstrates that the accuracy of the IRE is highly susceptible to the stabilizability of the system. Even with $d = 3$ the IRE fails to reach the desired accuracy. The GSV can compute accurate solutions up to $d = 5$. Beyond that, the GSV is unable to compute a solution and produces an error. Meanwhile, both the ABKSS and IREPS are able to compute an accurate solution up to the limit of machine precision $d = 16$.

Table 7: The scaled residual measures how close the function (1) is to zero and is given by the Frobenius norm of the function evaluated at the solution divided by the Frobenius norm of the solution.

d	IRE	ABKSS	IREPS	GSV
1.0	1.32×10^{-8}	3.58×10^{-9}	4.36×10^{-17}	1.55×10^{-16}
2.0	1.46×10^{-8}	7.19×10^{-13}	1.11×10^{-16}	2.13×10^{-16}
3.0	1.36×10^{-7}	1.40×10^{-9}	3.51×10^{-9}	6.61×10^{-15}
5.0	5.71×10^{-4}	3.99×10^{-11}	9.05×10^{-9}	2.21×10^{-15}
6.0	5.98×10^{-4}	3.97×10^{-12}	1.70×10^{-12}	N/A
16.0	6.00×10^{-4}	9.02×10^{-16}	9.02×10^{-16}	N/A
17.0	6.00×10^{-4}	N/A	N/A	N/A

Since the system size for our example is small ($n = 8, m = 5$), the computation times for all of our algorithms remain small (less than a second). Table 8. only illustrates that the IRE computation time will increase as it will indefinitely iterate since it cannot reach the desired accuracy.

Table 8: The time to solution is given in seconds and describes the wall clock time that the algorithm takes to reach the solution or the chosen tolerance.

d	IRE	ABKSS	IREPS	GSV
1.0	9.94×10^{-3}	9.04×10^{-3}	1.35×10^{-2}	1.62×10^{-3}
2.0	6.21×10^{-2}	4.13×10^{-3}	4.92×10^{-2}	5.50×10^{-4}
3.0	4.12×10^{-1}	4.09×10^{-3}	9.55×10^{-2}	5.51×10^{-4}
5.0	4.13×10^{-1}	4.15×10^{-3}	1.02×10^{-1}	5.98×10^{-4}
6.0	4.12×10^{-1}	4.10×10^{-3}	9.74×10^{-2}	N/A
16.0	4.12×10^{-1}	2.75×10^{-1}	9.59×10^{-2}	N/A
17.0	4.14×10^{-1}	N/A	N/A	N/A

Table 9. further illustrates the ineffectiveness of the IRE iterations. For well-posed problems,

we've seen the IRE reach the desired accuracy in less than 10 iterations. Even with $d = 1$, we find the IRE iteration count exceeds 100. Furthermore, we find that the ABKSS and IREPS struggle to maintain a consistently low iteration count for larger values of d.

Table 9: The number of iterations that each solver needs to reach the chosen accuracy.

d	IRE	ABKSS	IREPS	GSV
1.0	1.09×10^2	5.00×10^0	8.70×10^1	N/A
2.0	7.14×10^2	7.00×10^0	5.03×10^2	N/A
3.0	5.00×10^3	7.00×10^0	1.00×10^3	N/A
5.0	5.00×10^3	7.00×10^0	1.00×10^3	N/A
6.0	5.00×10^3	7.00×10^0	1.01×10^3	N/A
16.0	5.00×10^3	1.00×10^3	1.00×10^3	N/A
17.0	5.00×10^3	N/A	N/A	N/A

5 Special Case: DARE with Semi-Definite R

In this section, we discuss the DARE with $R \geq 0$. The GSV was modified in the 1980s to handle the special case of singular R, however, there is little to no literature on implementing Newton's method for this case. Here we briefly detail the modifications to the GSV and go into depth on framing the problem for Newton's method.

5.1 The Generalized Real-Schur Vector Method

The symplectic pencil in Eqn. (4) contains R^{-1} and thus cannot be evaluated for $R \geq 0$. To solve the generalized eigenvalue problem without evaluating the inverse of R, Arnold and Laub (1984) demonstrate the use of an extended symplectic pencil given by:

$$P = \begin{bmatrix} A & 0 & -B \\ -Q & -I & 0 \\ 0 & 0 & R \end{bmatrix}, \quad \text{and} \quad N = \begin{bmatrix} I & 0 & 0 \\ 0 & A' & 0 \\ 0 & B' & 0 \end{bmatrix}.$$

The solution is computed analogously through QZ factorization. Noting that the extended symplectic pencil is a $(2n+m)\times(2n+m)$ matrix, the GSV for singular R scales with order $\in(\in\setminus + \Updownarrow)^{\ni}$.

5.2 Newton's Method

The special case of the DARE with semi-definite R is an interesting problem explored in Hewer (1973) using techniques from Wonham (1968). However, the proof is inductive in nature pertaining to the iterate of the discrete iterative Riccati equation. In this section, we prove the existence of Newton's method iterates for $R = 0$. We proceed by proving the existence of the derivative of Eqn. (10) for $R = 0$. First, we observe the change in structure for the DARE with $R = 0$,

$$F(X) = A'XA - X + Q(B'XA)'(B'XB)^{-1}(B'XA).$$

Concern with $(B'XB)^{-1}$ term immediately arises as $X \geq 0$ implying that $B'XB \geq 0$. Thus, the inverse does not necessarily exist. The least-squares estimation problem for singular measurement covariance is reviewed in Albert (1972) (see pp. 170). Albert shows the resulting DARE is given by substituting the inverse with the Moore-Penrose pseudoinverse,

$$F(X) = A'XA - X + Q - (B'XA)'(B'XB)^{+}(B'XA). \tag{14}$$

This formulation leads us to the tricky problem of deriving Newton's iteration for the DARE with singular R. The trouble with this derivation is the derivative of the pseudoinverse. Developments have been made towards the derivative of the pseudoinverse. Notably, Golub and Pereyra (1973) provides a general derivative of the pseudoinverse. However, the derivative offered by Golub and Pereyra (1973) Thm. 4.3. must be a "Fréchet differentiable matrix function with local constant

rank." The constant local rank requirement cannot necessarily be guaranteed by the Newton iterations of the DARE with singular R since the iterates are not chosen. We take a different approach to the derivative starting with the limit definition of the pseudoinverse (see Albert (1972), pp. 19).

Theorem 1 (Albert (1972), Thm. 3.4.)**.** *For any $n \times m$ matrix A,*

$$\begin{aligned} A^+ &= \lim_{\delta \to 0} (A'A + \delta^2 I)^{-1} A' \\ &= \lim_{\delta \to 0} A'(AA' + \delta^2 I)^{-1} \end{aligned} \tag{15}$$

always exists. For any n-vector z,

$$\hat{x} = A^+ z,$$

is the vector of minimum norm among those which minimize

$$\|z - Ax\|^2.$$

Before we can take the derivative of the DARE for singular R, we need to compute the derivative of the pseudoinverse. We propose the following derivative for the pseudoinverse given the limit definition from Albert (1972).

Theorem 2 (Derivative of the Generalized Moore-Penrose Pseudoinverse)**.** *For any real matrix $A \in \mathbb{R}^{n \times m}$ the generalized Moore-Penrose pseudoinverse is defined as*

$$A^+ = \lim_{\delta \to 0} (A'A + \delta^2 I)^{-1} A'.$$

If for all directions H, the limit

$$D[A^+]_{A_0} H = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left[(A_0 + \epsilon H)^+ - A_0^+ \right],$$

exists, then A^+ is differentiable and is given by

$$D[A^+]_{A_0}H = -A_0^+ H A_0^+ - \lim_{\delta\to 0^+}(A_0'A_0 + \delta^2 I)^{-1}H'(I - A_0A_0^+). \tag{16}$$

Proof 1 (Proof of Thm. 2.)**.** *Consider any real matrix $A \in \mathbb{R}^{n\times m}$ and its pseudoinverse defined by*

$$A^+ = \lim_{\delta\to 0}(A'A + \delta^2 I)^{-1}A'.$$

Taking the derivative gives,

$$D[A^+]_{A_0}H = D[\lim_{\delta\to 0}(A'A + \delta^2 I)^{-1}]_{A_0}HA' + (A'A + \delta^2 I)^{-1}D[A']_{A_0}H.$$

Note that by the Moore-Osgood theorem, we can swap the limits in the first derivative,

$$D[A^+]_{A_0}H = \lim_{\delta\to 0} D[(A'A + \delta^2 I)^{-1}]_{A_0}HA' + \lim_{\delta\to 0}(A'A + \delta^2 I)^{-1}D[A']_{A_0}H.$$

For $\delta \neq 0$ the inverse $(A'A + \delta^2 I)^{-1}$ exists for any matrix A. Recall the derivative of an invertible matrix (function) from example 4, $D[A^{-1}]_{A_0}H = -A_0^{-1}HA_0^{-1}$. Applying our example yields,

$$\begin{aligned} D[A^+]_{A_0}H = &- \lim_{\delta\to 0}(A_0'A_0 + \delta^2 I)^{-1}D[A'A + \delta^2 I]_{A_0}H(A_0'A_0 + \delta^2 I)^{-1}A' \\ &+ \lim_{\delta\to 0}(A'A + \delta^2 I)^{-1}D[A']_{A_0}H \end{aligned} \tag{17}$$

Without much effort, we can compute the der

$$-\lim_{\delta\to 0}(B'X_0A)'\left[((B'X_0B)^2 + \delta^2 I)^{-1}B'HB(I - (B'X_0B)(B'X_0B)^+)\right](B'X_0A).$$

ivative $D[A']_{A_0}H = H'$. We can compute the derivative as follows:

$$
\begin{aligned}
D[A'A + \delta^2 I]_{A_0} H =& \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left[\left((A_0 + \epsilon H)'(A_0 + \epsilon H) + \delta^2 I \right) - \left(A_0' A_0 + \delta^2 I \right) \right], \\
=& A_0' H + H' A_0.
\end{aligned}
$$

Substituting back into Eqn. (17), we have

$$
\begin{aligned}
D[A^+]_{A_0} H = & - \lim_{\delta \to 0} (A_0' A_0 + \delta^2 I)^{-1} \left[A_0' H + H' A_0 \right] (A_0' A_0 + \delta^2 I)^{-1} \\
& + \lim_{\delta \to 0} (A_0' A_0 + \delta^2 I)^{-1} H'.
\end{aligned}
$$

Finally, with some algebra, our expression reduces to the result we were looking for

$$
D[A^+]_{A_0} H = -A_0^+ H A_0^+ - \lim_{\delta \to 0^+} (A_0' A_0 + \delta^2 I)^{-1} H' (I - A_0 A_0^+).
$$

Now that we have a derivative for the pseudoinverse, we move to take the derivative of part of the DARE with $R = 0$ in Eqn. (14). Taking note that $B' X_0 B$ is symmetric,

$$
\begin{aligned}
D[(B'XB)^+]_{X_0} H = & - (B' X_0 B)^+ B' H B (B' X_0 B)^+ \\
& - \lim_{\delta \to 0} ((B' X_0 B)^2 + \delta^2 I)^{-1} H' (I - (B' X_0 B)(B' X_0 B)^+).
\end{aligned} \tag{18}
$$

It is unfortunate that the above derivative is not differentiable for all X_0 and is not a helpful fact alone. This can be observed if we consider the limit term in the derivative and let $B' X_0 B = A = A' \geq 0$.

$$
\lim_{\delta \to 0} (A'A + \delta^2 I)^{-1} H' (I - A A^+).
$$

Let the direction H' be an arbitrary symmetric matrix M of appropriate size. We can say that

the evaluation of the limit is finite in two cases:

- Case I.

$$\lim_{\delta \to 0} (A'A + \delta^2 I) M < \infty, \tag{19}$$

- Case II.

$$M(I - AA^{+}) = 0. \tag{20}$$

Case I. The singular value decomposition of A is $A = VSV'$, where S are the singular values and V are the singular vectors. Taking advantage of the fact that V is a unitary matrix, we can show the following

$$\begin{aligned}
\lim_{\delta \to 0} (A'A + \delta^2 I)^{-1} M &= \lim_{\delta \to 0} \left(V \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix} V' + \delta^2 V I V' \right)^{-1} M, \\
&= \lim_{\delta \to 0} V \begin{bmatrix} \Sigma_r + \delta^2 & 0 \\ 0 & \delta^2 \end{bmatrix}^{-1} V' M, \\
&= \lim_{\delta \to 0} \left[V_1 (\Sigma_r + \delta^2 I_r)^{-1} V_1' + \frac{1}{\delta^2} V_2 V_2' \right] M,
\end{aligned}$$

where Σ_r are the nonzero singular values of A, V_1 are the singular vectors spanning the range of A', $\mathcal{R}(A')$, and V_2 are the singular vectors spanning the null space of A, $\mathcal{N}(A)$. Since Σ_r is a full rank diagonal matrix, $(\Sigma_r + \delta^2 I_r)^{-1}$ exists for any $\delta \in \mathbb{R}$. Then, for **Case I.** to be satisfied we only need

$$\lim_{\delta \to 0} \frac{1}{\delta} V_2 V_2' M = 0. \tag{21}$$

But this is only true when the columns of M are in the null space of $V_2 V_2'$. Denote the columns of $M \in \mathbb{R}^{n \times n}$ as m_i for $i = 1, \ldots, n$. It is equivalent to say the limit, Eqn. (21), is satisfied if $m_i \in \mathcal{N}(V_2')$, for all $i = 1, \ldots, n$.

Proof 2 (Equivalence of $m_i \in \mathcal{N}(V_2 V_2') \implies m_i \in \mathcal{N}(V_2')$.)**.** *Let $x \in \mathbb{R}^n$ and V_2 be an orthonormal basis for the null space of some matrix A. Let $x \in \mathcal{N}(V_2 V_2')$ or equivalently, $V_2 V_2' x = 0$. Since V_2 is an orthonormal basis for $\mathcal{N}(A)$, it is known that $V_2 V_2' \neq 0$. Left multiplying by x' yields $x' V_2 V_2' x = 0$. But this is just the inner product $\|V_2' x\|^2 = 0$, which directly implies that $V_2' x = 0$ and we are done.*

□

Recall that the columns of V_2 are an orthonormal basis for $\mathcal{N}(A)$, which implies that $\mathcal{N}(V_2')$ is equivalent with $\mathcal{R}(A')$. Then for the limit Eqn. (19) to exist, we need $m_i \in \mathcal{R}(A')$ for all $i = 1, \ldots, n$.

Case II. If the limit Eqn. (19) does not exist, then Eqn. (20) must be satisfied for the derivative Eqn. (16) to exist. The object $I - AA^+$ is the orthogonal projector onto $\mathcal{N}(A')$, *i.e.*, for $x = \hat{x} + \tilde{x}$ with $\hat{x} \in \mathcal{R}(A)$ and $\tilde{x} \in \mathcal{N}(A)$, $(I - AA^+)x = \tilde{x}$. Thus, for $(I - AA^+)x = 0$ we require $x \in \mathcal{R}(A)$. Naturally, this is a challenging requirement for arbitrary matrices A, however, we have the condition that A, M are symmetric real matrices. Thus, $\mathcal{R}(A) \equiv \mathcal{R}(A')$ and $M(I - AA^+) = 0$ if $\{m_i\}_{i=1}^n \in \mathcal{R}(A')$.

Proof 3 (Proof of $\{m_i\}_{i=1}^n \in \mathcal{R}(A') \implies M(I - AA^+) = 0$.)**.** *Consider real symmetric matrices $A, M \in \mathbb{R}^{n \times n}$. Let the columns of M be in $\mathcal{R}(A)$. By definition of the orthogonal projection $I - AA^+$, we have that $(I - AA^+)M = 0$. But we chose M to be symmetric, which implies $M(I - AA^+)' = 0$. The Moore-Penrose pseudoinverse is defined so that $(AA^+)' = AA^+$ for real matrices A. Thus, $(I - AA^+)' = (I - AA^+)$ which implies $M(I - AA^+) = 0$ and we are done.*

□.

Then, we can only satisfy both **Case I.** and **Case II.** if the columns of M are in $\mathcal{R}(A) \equiv \mathcal{R}(A')$ or neither case for real symmetric matrices A, M. Recall Eqn. (18),

$$D[(B'XB)^+]_{X_0}H = -(B'X_0B)^+B'HB(B'X_0B)^+ - \lim_{\delta\to 0}((B'X_0B)^2+\delta^2 I)^{-1}B'HB(I-(B'X_0B)(B'X_0B)^+).$$

For real symmetric X_0, H, we require the columns of $B'HB$ to be in $\mathcal{R}(B'X_0B)$. Since we do not choose H, we cannot guarantee the limit will exist. Despite this, we can show the derivative of the DARE with $R = 0$ still exists. Recall the derivative of the DARE contains the term

$$D[(B'XA)'(B'XB)^+(B'XA)]_{X_0}H.$$

The limit in Eqn. (18) only appears in the form,

$$-\lim_{\delta\to 0}(B'X_0A)'\left[((B'X_0B)^2+\delta^2 I)^{-1}B'HB(I-(B'X_0B)(B'X_0B)^+)\right](B'X_0A).$$

In our discussion of **Case II.**, we stated that $I - AA^+$ is the orthogonal projector onto $\mathcal{N}(A') \equiv \mathcal{N}(A)$ for real symmetric matrix A. We propose the following conjecture.

Conjecture 1. *For real symmetric matrix $X_0 \geq 0$, the product*

$$\left[I-(B'X_0B)(B'X_0B)^+\right](B'X_0A) = 0.$$

Thus the limit,

$$-\lim_{\delta\to 0}(B'X_0A)'\left[((B'X_0B)^2+\delta^2 I)^{-1}B'HB(I-(B'X_0B)(B'X_0B)^+)\right](B'X_0A).$$

vanishes regardless of direction H. It is then implied that the derivative of the DARE described in Eqn. (14),

$$F(X) = A'XA - X + Q - (B'XA)'(B'XB)^{+}(B'XA)$$

exists and has the form

$$D[F(X)]_{X_0}H = [A + BK(X_0)]' H [A + BK(X_0)] - H. \tag{22}$$

where $K(X_0) = -(B'X_0B)^{+}(B'X_0A)$.

Proof 4 (Proof of Conjecture 1)**.** *First, we prove the following equality*

$$\left[I - (B'X_0B)(B'X_0B)^{+}\right] B'X_0 = 0,$$

for real symmetric matrix $X_0 \geq 0$ *in* $\mathbb{R}^{n\times n}$ *and arbitrary matrix* $B \in \mathbb{R}^{n\times m}$*. For shorthand notation define the orthogonal projection onto the null space of* $B'X_0B$ *to be*

$$T = (I - B'X_0B(B'X_0B^{+})).$$

Then by the definition of the Moore-Penrose pseudoinverse, we know that

$$TB'X_0B = 0.$$

It is helpful to show that $X_0 = S'S$*. Since* $X_0 = X_0' \geq 0$*, we can define a matrix* S *such that*

$$\begin{aligned} X_0 =& V\Lambda V', \\ =& V\Lambda^{1/2}\Lambda^{1/2}V', \\ =& S'S, \end{aligned}$$

where V, Λ *are the eigenvectors and eigenvalues of* X_0 *respectively. Then, we can rewrite*

$$TB'X_0B = TB'S'SB = 0.$$

Furthermore, we can right multiply by T' to get

$$TB'S'SBT' = 0.$$

But this is only true if $TB'S' = 0$ and thus $TB'S'S = TB'X_0 = 0$. This directly implies that the limit

$$-\lim_{\delta \to 0}(B'X_0A)' \left[((B'X_0B)^2 + \delta^2 I)^{-1} B'HB(I - (B'X_0B)(B'X_0B)^+)\right](B'X_0A),$$

vanishes regardless of the direction H. The derivative of the DARE then becomes

$$\begin{aligned} D[F(X)]_{X_0}H =& A'HA - H - (B'HA)'(B'X_0B)^+(B'X_0A) \\ &-(B'X_0A)'(B'X_0B)^+(B'HA) \\ &+(B'X_0A)'(B'X_0B)^+B'HB(B'X_0B)^+(B'X_0A). \end{aligned}$$

With some algebra, the above expression reduces to

$$D[F(X)]_{X_0}H = [A + BK(X_0)]' H [A + BK(X_0)] - H,$$

where $K(X_0) = -(B'X_0B)^+(B'X_0A)$ and we are done.

□

6 Conclusions

The DARE is a nonlinear matrix equation whose stabilizing solution is highly sought after in the context of the LQR and LQE problems. With the importance of safety, profit, and reduced process variability; accurate control is highly desirable for chemical plant operation. This work revisits solution methods to the DARE with a focus on quick and accurate computation for real-time plant estimation and regulation. We consider the drawbacks of the IRE, Newton's method, and the GSV for large-scale and barely stabilizable systems. Our findings suggest implementing Newton's method as a refinement technique for the IRE.

In this review of DARE solution methods, we find a variety of interesting results. The most commonly employed DARE solver, the GSV known as the command `dare` in Octave and MATLAB, computes solutions to high accuracy. When tested for large systems, we found that the GSV computation time is drastically increased for systems with more than 1000 states. The slow computation times can be attributed to the computational complexity of the GSV algorithm scaling approximately as $\mathcal{O}(10n^3)$, n is the system size. The IRE offers a much faster computation time, however, it trades accuracy as the system size exceeds 1000 states. Additionally, the IRE can suffer from numerical instability which negatively affects the solution accuracy. Newton's method offers value as a refinement technique to limit susceptibility to numerical instability but is ineffective as a stand-alone method. Even for well-posed problems with small system sizes, Newton's method cannot reach the stabilizing solution without a proper initial guess. Utilizing a stability switch or a proximity switch allows the combination of the IRE and Newton's method to overcome their respective drawbacks. The ABKSS and IREPS algorithms prescribed in this work offer strong performance for large or barely stabilizable systems. As a result, it is advised that the GSV be used for small well-posed problems for its desirable accuracy. As the system becomes larger, an IRE-based method with either a stability or proximity switch should be used to reduce computation time without negatively impacting accuracy.

Furthermore, we investigate the use of Newton's method for DAREs with $R = 0$. In the LQR problem, $R = 0$ is not as interesting as that would be equivalent to setting the influence of the

control action on the controller to zero. In the LQE problem, however, it is more interesting for the purpose of imposing constraints or exact observations in the model. We find through lengthy derivations that Newton's method is largely unchanged and only replaces the inverse operation with the pseudoinverse operation in the discrete Lyapunov step of each iteration. Furthermore, the proof for Newton's method in this special case implies that the discrete Lyapunov equation can be solved regardless of its solution. This work towards providing a Newton's method analog to the GSV with the extended symplectic pencil, however, is not complete. The proofs provided in this work provide intrigue to future work for the general case of the DARE with $R \geq 0$.

This thesis offers a guideline for selecting a DARE solution method based on system size and stabilizability. Since its development in the early 1980s, the GSV has been the gold standard for solving the LQR and LQE problems. With the advancement of computational resources modifications of the GSV have been implemented, including Newton method refinements and capabilities for $R \geq 0$. However, this study revisits the iterative Riccati equation from Kalman's work in the 1960s. We provide a new perspective on the solution to large-scale DAREs with consideration of numerical instability. Two new Newton refinement techniques of the IRE are offered, the ABKSS and IREPS. Both are capable of solving DAREs with more than 1000 states quickly with high accuracy. Furthermore, the groundwork for establishing Newton's method for singular R is developed.

7 Future Work

The future of this work pertains to the IREPS switch and Newton's method for the DARE with singular R. The IREPS switch is a transition of employment of the IRE to Newton's method based on the proximity of the iterate to the solution. In its current form, the IREPS switch criteria is an arbitrary tolerance imposed on the scaled residual of the current iterate. Once the current iterate of the IRE becomes less than said tolerance, the iterate is used as the initial guess to Newton's method. Based on the speed of the IRE from our numerical results it should be employed for as long as possible. Thus, it would likely improve computation times if the switch is triggered when

the numerical error overrides the iterative improvements. That is to say, the switch is activated once there is negligible improvement between multiple iterates or when monotonicity is broken. Then the IRE is used until the desired accuracy is reached or when further improvement cannot be made.

The future of Newton's method for the DARE with singular R is to show the Newton method iteration for the DARE:

$$X = A'XA + Q - (B'XA + S')'(B'XB + R)^{+}(B'XA + S'),$$

when $R \geq 0$. As we saw in Eqn. (18), the limit does not exist in all directions alone. Instead, the limit is canceled by the following $B'XA$ term. Here we would require the columns of $(B'X_0A+S')$ to be in the range space of $(B'X_0B + R)$, which we were not able to guarantee in the time provided for this work. In the future of this work, the proof of Newton's method for $R = 0$ should be extended to $R \geq 0$.

References

A. Albert. *Regression and the Moore-Penrose pseudoinverse*, volume 94 of *Mathematics in Science and Engineering*. Academic Press, New York and London, 1972.

B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Prentice-Hall, Englewood Cliffs, N. J., 1979.

W. F. Arnold and A. J. Laub. Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proc. IEEE*, 72(12):1746–1754, Dec 1984.

P. Benner and H. Faßbender. On the numerical solution of large-scale sparse discrete-time Riccati equations. *Adv. Comput. Math.*, 35(2-4):119, 2011.

P. E. Caines and D. Q. Mayne. On the discrete time matrix Riccati equation of optimal control. *Int. J. Control*, 12(5):785–794, 1970.

B. Datta. *Numerical methods for linear control systems*, volume 1. Academic Press, 2004.

G. H. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM J. Numer. Anal.*, 10(2):413–432, 1973.

G. Hewer. Analysis of a discrete matrix Riccati equation of linear control and Kalman filtering. *J. Math. Anal. Appl.*, 42(1):226–236, 1973.

G. A. Hewer. Iterative technique for computation of steady state gains for discrete optimal regulator. *IEEE Trans. Auto. Cont.*, AC16(4):382–384, 1971.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, J. Basic Engineering*, pages 35–45, Mar 1960.

D. L. Kleinman. Stabilizing a discrete, constant, linear system with application to iterative methods for solving the Riccati equation. *IEEE Trans. Auto. Cont.*, 19:252–254, Jun 1974.

P. Lancaster and L. Rodman. *Algebraic Riccati equations*. Clarendon press, 1995.

K. Long. Gateaux differentials and Fréchet derivatives. *Course notes, Texas Tech University*, 2009.

J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley & Sons Ltd., Hoboken, NJ, third edition, 2019.

V. L. Mehrmann. *The Autonomous Linear Quadratic Control Problem*. Springer-Verlag, Berlin, 1991.

T. Pappas, A. J. Laub, and N. R. Sandell, Jr. On the numerical solution of the discrete-time algebraic Riccati equation. *IEEE Trans. Auto. Cont.*, AC-25(4):631–641, Aug 1980.

J. B. Rawlings, D. Q. Mayne, and M. M. Diehl. *Model Predictive Control: Theory, Design, and Computation*. Nob Hill Publishing, Madison, WI, 2nd edition, 2017. 770 pages, ISBN 978-0-9759377-3-0.

P. Van Dooren. A generalized eigenvalue approach for solving Riccati equations. *SIAM J. Sci. Stat. Comp.*, 2(2):121–135, 1981.

W. M. Wonham. On a Matrix Riccati Equation of Stochastic Control. *SIAM J. Cont.*, 6(4):681–697, 1968. doi: 10.1137/0306044.

T. Yamamoto. Historical developments in convergence analysis for Newton's and Newton-like methods. *J. Comput. Appl. Math.*, 124(1-2):1–23, 2000.