A Game Theoretic Algorithm to Solve Riccati and Hamilton–Jacobi–Bellman–Isaacs (HJBI) Equations in H_∞ Control

Brian D. O. Anderson[1], Yantao Feng[2], and Weitian Chen[3]

[1] Research School of Information Sciences and Engineering, the Australian National University, Canberra ACT 0200, Australia; National ICT Australia, Tower A, 7 London Circuit, Canberra ACT 2601, Australia
brian.anderson@anu.edu.au
[2] Research School of Information Sciences and Engineering, the Australian National University, Canberra ACT 0200, Australia; National ICT Australia, Tower A, 7 London Circuit, Canberra ACT 2601, Australia
alex.feng@anu.edu.au
[3] Research School of Information Sciences and Engineering, the Australian National University, Canberra ACT 0200, Australia
weitian.chen@anu.edu.au

Summary. In this chapter, we propose a new algorithm to solve Riccati equations and certain Hamilton–Jacobi–Bellman–Isaacs (HJBI) equations arising in H_∞ control. The need for the algorithm is motivated by the existence of H_∞ problems for which standard Riccati solvers break down, but which can be handled by the algorithm. By using our algorithm, we replace the problem of solving H_∞ Riccati equations or HJBI equations by the problem of solving a sequence of H_2 Riccati equations or Hamilton–Jacobi–Bellman (HJB) equations. The algorithms have some advantages such as a simple initialization, local quadratic rate of convergence, and a natural game theoretic interpretation. Some numerical examples are given to demonstrate advantages of our algorithm.

Key words: Riccati, HJBI, iterative, game theoretic, convergence

1 Introduction

This chapter addresses computational issues in H_∞ control, in particular advancing algorithms for solving H_∞ Riccati equations and their generalization and Hamilton–Jacobi–Bellman–Isaacs (HJBI) equations. Though algorithms are not especially well developed for HJBI equation solution, there are certainly standard software packages allowing H_∞ Riccati equation solution, e.g.,

A. Chinchuluun et al. (eds.), *Optimization and Optimal Control*,
Springer Optimization and Its Applications 39, DOI 10.1007/978-0-387-89496-6_15,

RICPACK (see [5]) and MATLAB, and it is natural to ask why another algorithm should be needed, at least for this class of equation. Therefore, we spend some time describing the motivation for this work.

The motivation actually goes back to the problem of solving H_2 Riccati equations. Again, well-established software tools exist, for example, LAPACK and BLAS (see [4]). However, there exist examples of H_2 Riccati equations where these tools break down, as we review later. One technique which can remain viable in such situations is the recursive algorithm of Kleinman (see [24]). The Kleinman algorithm replaces the task of solving the Riccati equation directly by the task of solving a recursive sequence of Lyapunov equations. Now just as there exist examples where standard H_2 Riccati solvers break down, so is this true with standard H_∞ Riccati solvers. It is natural then to ask, Can the problem be fixed by the extension of the algorithm of Kleinman to this situation? The immediate answer is no (see Example 2 in the Appendix of [27] for a demonstration of this). However, as this chapter sets out, there is a fix, motivated by the Kleinman algorithm and in some ways constituting a significant extension of the Kleinman algorithm. The relevant ideas were originally presented in [26, 27].

To the extent that an H_2 Riccati equation is effectively a particular example of a Hamilton–Jacobi–Bellman (HJB) equation, and an H_∞ Riccati equation a particular example of a HJBI equation, it is natural to seek generalizations of the Riccati algorithms. Such generalizations may not just be useful in a few situations where numerical problems arise, but might be generally useful, given the limited development to this point of standard packages for solving HJB and HJBI equations. Indeed, there is an old generalization of the Kleinman algorithm, which replaces solution of the HJB nonlinear partial differential equation by the recursive solution of a sequence of linear partial differential equations (see [29]). There is, however, no corresponding algorithm for HJBI equations, and this chapter's second main contribution is to offer such an algorithm.

Apart from the ability of the algorithms we present to solve problems that may defeat conventional solvers, we note their following specific advantages: (1) a simple initialization; (2) local quadratic rate of convergence; (3) a natural game theoretic interpretation; (4) high numerical stability and reliability.

We shall now provide a high-level description of the algorithms. First, we replace the problem of solving an H_∞ Riccati equation by the problem of solving a sequence of H_2 Riccati equations. By using our algorithm, we transfer an H_∞ problem into a sequence of optimal control problems; by doing so, we indeed transfer a difficult problem into a sequence of less difficult problems. Since any single H_2 Riccati equation can be solved using the Kleinman algorithm, i.e., by solving an iterative sequence of Lyapunov equations, it is also apparent that an H_∞ equation can, if desired, be solved by using a nested double iteration of Lyapunov equations.

Second, and by way of generalization, we replace the problem of solving an HJBI equation by the problem of solving a sequence of Hamilton–Jacobi–

Bellman (HJB) equations. As for the Riccati case, an HJBI equation can be solved using a nested double iteration of linear partial differential equations. Whether one uses the single or double iteration is of course optional.

We shall now present more details on the approach for Riccati equations. Consider the following algebraic Riccati equation (ARE) in the variable P:

$$0 = A^T P + PA + PRP + Q, \tag{1}$$

where A, Q, R are real $n \times n$ matrices with Q and R symmetric. Here, $(\cdot)^T$ denotes the transpose of $(\cdot)$. Associated with this Riccati equation is a $2n \times 2n$ Hamiltonian matrix

$$H := \begin{pmatrix} A & R \\ -Q & -A^T \end{pmatrix}.$$

Generally speaking, existing methods to solve AREs can be divided into two categories:

1. Direct: solutions of ARE (1) can be constructed via computation of an n- dimensional invariant subspace of the Hamiltonian matrix H (for example, using the Schur algorithm in [28]).
2. Iterative: a sequence of matrices which converge to the unique stabilizing solution of special classes of the ARE (1) is constructed (for example, using the Kleinman algorithm in [24]).

Several different direct methods to solve the ARE (1) are given in [1, 5, 11, 15, 25, 28, 31, 33, 35]. However, compared with iterative methods to solve ARE (1), direct methods present computational disadvantages in some situations. For example, in Example 6 in [28], the solution to an H_2 ARE obtained by the Schur algorithm in [28] is inaccurate but the iterative solution obtained by the Kleinman algorithm in [24] is accurate to 13 digits in just two iterations.

Traditionally, in H_2 control, one needs to solve AREs with $Q \geq 0$ and $R \leq 0$. In H_∞ control, one needs to solve AREs with positive semidefinite Q and sign indefinite R. Although the Kleinman algorithm in [24] has been shown to have many advantages such as convergence for any suitable initial condition and a local quadratic rate of convergence [24], these advantages are strictly restricted to AREs arising in H_2 control where R in (1) must be negative semidefinite. It is not difficult to adjust the Kleinman method (which is effectively an implementation of an equation solver using Newton's method) to also handle the separate case where $R \geq 0$, but still sign-indefinite R cannot be handled. So the question naturally arises, "Can one extend the Kleinman algorithm in [24] to solve AREs with a sign indefinite quadratic term, as those that arise in H_∞ control?" The answer is that an iterative algorithm with very simple initialization to solve such a class of AREs will be given in this chapter, but the algorithm cannot be obtained by simply permitting indefinite R to occur in the Kleinman algorithm.

In the Kleinman algorithm, when a suitable initial condition is chosen and some necessary assumptions hold, it is proved that a series of Lyapunov

equations can be recursively constructed at each iteration, and positive semidefinite solutions of these Lyapunov equations converge to the stabilizing solution of the corresponding H_2 ARE. In our proposed algorithm, an ARE with a sign indefinite quadratic term is replaced by a sequence of H_2 AREs (each of which could be solved by the Kleinman algorithm if desired, though this need not happen), and the solution of the original ARE with a sign indefinite quadratic term is obtained by recursively solving these H_2 AREs.

Besides the Kleinman algorithm, there are some other iterative methods to solve AREs [1–3, 11–16, 22, 25, 32, 33, 35], some of which exhibit quadratic convergence. Among iterative methods to solve the ARE (1), Newton-type algorithms are typical and widely used [1, 11–16, 25, 33, 35]. In fact, Newton's method can be used to solve more than just symmetric AREs like (1). It can also be used to solve non-symmetric AREs where Q and R in (1) are not necessarily symmetric [20, 21]. However, besides Newton-type algorithms, there are other iterative algorithms again to solve AREs with a sign indefinite quadratic term, for example, the matrix sign function method (see [15, 33, 35]). However, there are also disadvantages when the matrix sign function method is used to solve AREs, for example, when the eigenvalues of the corresponding Hamiltonian matrix of a given ARE are close to the imaginary axis, this method will perform poorly or even fail.

As noted above, in the work presented in this chapter, we reduce the problem of solving a generic Riccati equation with a sign indefinite quadratic term to one of generating successive iterations of solutions of conventional H_2 AREs with a negative semidefinite quadratic term (each of which is then amenable to the Kleinman algorithm). Consequently, we are reducing a Riccati equation that has no straightforwardly initialized iterative scheme for its solution to a number of successive iterations of Riccati equations, each of which can (if desired) be solved by an existing iterative scheme (e.g., the Kleinman algorithm).

Although linear optimal control theory, as well as linear H_∞ control theory, has been well developed in the past decades, matters become more complicated when a nonlinear control system is considered. For example, in nonlinear optimal control, HJB equations may need to be solved to obtain an optimal control law. However, HJB equations are first-order, nonlinear partial differential equations that have been proven to be impossible to solve in general and are often very difficult to solve even for specific nonlinear systems. Since these equations are difficult to solve analytically, there has been much research directed toward approximating their solutions. For example, the technique of successive approximation in policy space [8–10] can be used to approximate the solutions of HJB equations iteratively. In fact, it can be shown (see [29]) that the technique of policy space iteration can be used to replace a nonlinear HJB partial differential equation by a sequence of linear partial differential equations. Also, in some sense, the iterative procedure to solve HJB equations in [29] is a generalization of the Kleinman algorithm in [24], since both of them obtain solutions by constructing a sequence of monotonic functions

or matrices while the algorithm in [29] can be used in more general cases than just the LQ problem.

In nonlinear H_∞ control, given a disturbance attenuation level $\gamma > 0$, in order to solve the H_∞ suboptimal control problem, one needs to solve Hamilton–Jacobi–Bellman–Isaacs (HJBI) equations. It is clear that HJBI equations are generally more difficult to solve than HJB equations, since the disturbance inputs are additionally reflected in HJBI equations. Recall that the Riccati equation algorithm to be presented will reduce an ARE with an indefinite quadratic term to a sequence of AREs with a negative semidefinite quadratic term, which are more easily solved by an existing algorithm (e.g., the Kleinman algorithm). If we regard HJB equations as the general version of AREs with a negative semidefinite quadratic term and HJBI equations as the general version of AREs with an indefinite quadratic term, then the question arising here is, "Can we approximate the solution of an HJBI equation by obtaining the solutions of a sequence of HJB equations and thereby extend the recursive H_∞ Riccati algorithm to nonlinear control systems?" In this chapter, we will answer this question to some degree, that is, we extend the Riccati algorithm for a specific class of nonlinear control systems and develop an iterative procedure to solve a broad class of HJBI equations associated with the nonlinear H_∞ control problem. It is important to note that others have made a direct attack on HJBI equations using single and double iterations, but their methods do not allow a simple initialization of the algorithms, which is a severe disadvantage (see [36] and [7]). To implement the algorithms in [36] and [7], one has to choose a stabilizing control law achieving the prescribed attenuation level, which is not always straightforward to obtain.

Besides the advantages mentioned above, our algorithm can be expected to have a higher accuracy and numerical stability than existing algorithms to solve HJBI equations since our algorithm in the linear time-invariant case (i.e., solving H_∞ algebraic Riccati equations) has shown higher accuracy and numerical reliability, see Example 2 in Section 3.5 for a demonstration of this.

The notation is as follows: $\mathbb{R}$ denotes the set of the real numbers; $\mathbb{R}^+$ denotes the set of the nonnegative numbers; $(\cdot)^T$ denotes the transpose of a vector or a matrix; $\overline{\sigma}(\cdot)$ denotes the maximum singular value of a matrix; $\mathbb{Z}$ denotes the set of integers with $\mathbb{Z}_{\geq a}$ denoting the set of integers greater or equal to $a \in \mathbb{R}$; $\mathbb{R}^n$ denotes an n-dimensional Euclidean space; $\mathbb{S}^{n\times n}$ denotes the set of n-dimensional real symmetric matrices. Let $X \in \mathbb{R}^{n\times n}$ be a real matrix, then $X \geq 0$ means that X is positive semidefinite; Let $X,\ Y \in \mathbb{R}^{n\times n}$ be two positive semidefinite matrices, then $X \geq Y$ means that the matrix $X - Y$ is positive semidefinite (i.e., $X - Y \geq 0$). Let $P_k \in \mathbb{R}^{n\times n}$ be a matrix sequence for $k \in \mathbb{Z}_{\geq 0}$, if $P_k \geq 0$ and $P_{k+1} \geq P_k$ for all $k \in \mathbb{Z}_{\geq 0}$, then the sequence P_k is called monotonically non-decreasing.

For a given control system, denote the state space by $\mathbb{X} \subseteq \mathbb{R}^n$, the set of control input values by $\mathbb{U} \subseteq \mathbb{R}^m$, the set of disturbance input values by $\mathbb{W} \subseteq \mathbb{R}^q$, and the set of output values by $\mathbb{Y} \subseteq \mathbb{R}^p$. Moreover, define $\mathbb{X}_0$ as a neighborhood of the origin in $\mathbb{R}^n$, $\mathbb{U}_0$ as a neighborhood of the origin in $\mathbb{R}^m$,

$\mathbb{W}_0$ as a neighborhood of the origin in $\mathbb{R}^q$, and $\mathbb{Y}_0$ as a neighborhood of the origin in $\mathbb{R}^p$. Define the function space $\mathcal{X}_0$ as follows:

$$\mathcal{X}_0 = \left\{ x : \mathbb{R}^+ \to \mathbb{X}_0 \;\middle|\; \int_{t_0}^{t_1} \|x(t)\|^2 dt < \infty \quad \forall t_0, t_1 \in \mathbb{R}^+ \right\}.$$

Function spaces $\mathcal{U}_0,\ \mathcal{W}_0$, and $\mathcal{Y}_0$ are defined similarly as $\mathcal{X}_0$.

A matrix is said to be *Hurwitz* if all of its eigenvalues have negative real parts.

2 Solving the LQ Problem by the Kleinman Algorithm

As noted in the previous section, for the linear time-invariant case of our algorithm, we replace the problem of solving an H_∞ Riccati equation by the problem of solving a sequence of H_2 Riccati equations; then each of these H_2 Riccati equations can be solved by the Kleinman algorithm. The Kleinman algorithm, originally used to solve the LQ problem, will be reviewed in this section. By using the Kleinman algorithm, we can replace the problem of solving an H_2 Riccati equation by the problem of solving a sequence of Lyapunov equations; then each Lyapunov equation can be solved by existing numerical algorithms. By doing so, we transfer a nonlinear matrix equation (an H_2 Riccati equation) into a sequence of linear equations (Lyapunov equations).

Consider a continuous-time linear system described by

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}$$

with a cost functional defined as

$$J = \int_0^\infty \left(x^T C^T C x + u^T u \right) dt.$$

In the LQ problem, it is well known that the feedback control law that minimizes the value of the cost J is

$$u = -B^T K x,$$

with K solving

$$0 = A^T K + KA - KBB^T K + C^T C. \tag{2}$$

If $(A,\ B)$ is stabilizable and $(C,\ A)$ is detectable, it can be shown that (2) has a unique stabilizing solution. In such a situation, the Kleinman algorithm can be used to solve (2). The Kleinman algorithm to solve (2) is given as follows:

1. choose an initial stabilizing state feedback gain L_0;
2. $A_0 := A - BL_0$;

3. obtain V_0 by solving the Lyapunov equation

$$0 = A_0^T V_0 + V_0 A_0 + L_0^T L_0 + C^T C;$$

4. $L_k := B^T V_{k-1}, \quad k = 1, 2, \ldots;$
5. $A_k := A - BL_k, \quad k = 1, 2, \ldots;$
6. obtain V_k by solving the following Lyapunov equation:

$$0 = A_k^T V_k + V_k A_k + L_k^T L_k + C^T C, \quad k = 1, 2, \ldots.$$

It can be proved that the sequence V_k is monotonically non-increasing and converges to the stabilizing solution K of (2).

In H_2 control, typically, the Kleinman algorithm for solving H_2-type AREs with a negative semidefinite quadratic term is well suited as a "second iterative stage" refinement to achieve the prescribed accuracy for the stabilizing solution of the ARE. For example, if an approximate stabilizing solution is known (e.g., one observes using the Schur method), and this is stabilizing, then one to two iterations are sufficient to achieve the limiting accuracy (because of the guaranteed final quadratic rate of convergence of a typical Newton algorithm). Now as noted, the Kleinman algorithm reduces a quadratic (Riccati) equation (with a negative semidefinite quadratic term) to several successive iterations of linear (Lyapunov) equations; the complexity of solving algebraic Lyapunov and Riccati equations with sound numerical methods (e.g., Schur form-based reductions) is $O(n^3)$ for both. When Schur form-based reductions are used to solve Lyapunov equations, the computation for such (Schur form-based) reductions needs about $25n^3$ flops (see [18]), where 1 flop equals 1 addition/subtraction or 1 multiplication/division. About $7n^3$ flops are necessary to solve the reduced equation and to compute the solution. The basic method is described in [6]. For the solutions of AREs, the Schur approach of Laub [28] requires $240n^3$ flops of which $25(2n)^3$ flops are required to reduce a $2n \times 2n$ Hamiltonian matrix to real Schur form and the rest accounts for the computation of the eigenvalues and solving a linear equation of order n (i.e., $\frac{5}{3}n^3$ flops). Consequently, both Riccati and Lyapunov equations require $O(n^3)$ computations. Hence the advantage of iterative schemes such as the Kleinman algorithm (which will require several Lyapunov equations to be solved, typically) is not always the speed of computation, but rather it is the numerical reliability of the computations to reach the prescribed accuracy of a solution.

3 Solving H_∞ Riccati Equations

This section includes five subsections: (1) the summarizing theorem; (2) an algorithm to solve H_∞ Riccati equations; (3) an examination of the rate of convergence of the algorithm; (4) a game theoretic interpretation of the algorithm, (5) numerical examples.

3.1 The Summarizing Theorem

In this subsection, we will restrict attention to the unique stabilizing solution Π for the following ARE:

$$0 = \Pi A + A^T\Pi - \Pi(B_2B_2^T - B_1B_1^T)\Pi + C^TC, \tag{3}$$

where A, B_1, B_2, C are real matrices with compatible dimensions. Note that stabilizing solutions to AREs are always unique (see [37]) when they exist, but for AREs with a sign indefinite quadratic term, the unique stabilizing solution Π may not always be positive semidefinite. Since our interest arises from AREs used for H_∞ control, in this case, in order to obtain an H_∞ controller, we need to solve AREs with a sign indefinite quadratic term and the stabilizing solutions of these AREs are also required to be positive semidefinite if such an H_∞ controller exists. So we focus on a unique stabilizing solution to (3) that happens to be also positive semidefinite when this exists. The algorithm we will propose in Section 3.2 has two aspects: (1). Check the existence of the unique stabilizing solution, which is also positive semidefinite, of (3) and (2). Construct the unique stabilizing solution, which is also positive semidefinite, of (3) if such a solution exists.

Motivated by the right-hand side of (3), we define a function F which will be used in our summarizing theorem:

$$\begin{aligned} F:\ \mathbb{R}^{n\times n} &\longrightarrow \mathbb{R}^{n\times n} \\ P \quad &\longmapsto PA + A^TP - P(B_2B_2^T - B_1B_1^T)P + C^TC. \end{aligned} \tag{4}$$

In this section, we set up the summarizing theorem by constructing two positive semidefinite matrix series P_k and Z_k, and we also prove that the series P_k is monotonically non-decreasing and converges to the unique stabilizing solution Π (which is also positive semidefinite) of ARE (3) if such a solution exists.

Theorem 1. (The summarizing theorem) *Let A, B_1, B_2, C be real matrices with compatible dimensions. Suppose that (C, A) has no unobservable modes on the $j\omega$-axis and (A, B_2) is stabilizable, define $F : \mathbb{R}^{n\times n} \longrightarrow \mathbb{R}^{n\times n}$ as in (4). Suppose there exists a stabilizing solution Π, which is also positive semidefinite, of ARE (3).*
Then

(I) *two square matrix series Z_k and P_k can be defined for all $k \in \mathbb{Z}_{\geq 0}$ recursively as follows*:

$$P_0 = 0, \tag{5}$$

$$A_k = A + B_1B_1^TP_k - B_2B_2^TP_k, \tag{6}$$

$Z_k \geq 0$ is the unique stabilizing solution of

$$0 = Z_kA_k + A_k^TZ_k - Z_kB_2B_2^TZ_k + F(P_k), \tag{7}$$

$$P_{k+1} = P_k + Z_k; \tag{8}$$

(II) *the two series* P_k *and* Z_k *in part* (I) *have the following properties*:
(1) $(A + B_1 B_1^T P_k, B_2)$ *is stabilizable* $\forall k \in \mathbb{Z}_{\geq 0}$.
(2) $F(P_{k+1}) = Z_k B_1 B_1^T Z_k \ \forall k \in \mathbb{Z}_{\geq 0}$.
(3) $A + B_1 B_1^T P_k - B_2 B_2^T P_{k+1}$ *is Hurwitz* $\forall k \in \mathbb{Z}_{\geq 0}$.
(4) $\Pi \geq P_{k+1} \geq P_k \geq 0 \ \forall k \in \mathbb{Z}_{\geq 0}$;

(III) *the limit*

$$P_\infty := \lim_{k \to \infty} P_k$$

exists with $P_\infty \geq 0$. *Furthermore,* $P_\infty = \Pi$ *is the unique stabilizing solution of ARE* (3)*, which is also positive semidefinite.*

Proof. See [26, 27]. □

The following corollary gives a condition under which there does *not* exist a stabilizing solution $\Pi \geq 0$ to $F(\Pi) = 0$. This is useful for terminating the recursion in finite iterations. If there does not exist a stabilizing solution $\Pi \geq 0$ to (3), there are two possible situations in Theorem *1*: (1) The stabilizability condition (II1) fails at some iteration; and (2) The sequence P_k in Theorem 1 diverges to infinity.

Corollary 1. *Let* A, B_1, B_2, C *be real matrices with compatible dimensions. Suppose that* (C, A) *has no unobservable modes on the* $j\omega$*-axis and* (A, B_2) *is stabilizable, and let* $\{P_k\}$ *and* $F : \mathbb{R}^{n \times n} \longrightarrow \mathbb{R}^{n \times n}$ *be defined as in Theorem 1. If* $\exists k \in \mathbb{Z}_{\geq 0}$ *such that* $(A + B_1 B_1^T P_k, B_2)$ *is not stabilizable, then there does not exist a stabilizing solution* $\Pi \geq 0$ *to* $F(\Pi) = 0$.

Proof. Restatement of Theorem 1, implication (II1). □

3.2 Algorithm

Let A, B_1, B_2, C be real matrices with compatible dimensions and $\Delta > 0$ be a specified tolerance. Suppose that (C, A) has no unobservable modes on the $j\omega$-axis and (A, B_2) is stabilizable. Then an iterative algorithm for finding the positive semidefinite stabilizing solution of (3), when it exists, is given as follows:

1. Let $P_0 = 0$ and $k = 0$.
2. Set $A_k = A + B_1 B_1^T P_k - B_2 B_2^T P_k$.
3. Construct (for example, using the Kleinman algorithm in [24], though this is not necessary) the unique real symmetric stabilizing solution $Z_k \geq 0$ which satisfies

$$0 = Z_k A_k + A_k^T Z_k - Z_k B_2 B_2^T Z_k + F(P_k), \tag{9}$$

   where $F : \mathbb{R}^{n \times n} \longrightarrow \mathbb{R}^{n \times n}$ is defined in (4).
4. Set $P_{k+1} = P_k + Z_k$.
5. If $\overline{\sigma}(B_1^T Z_k)^2 < \Delta$, then set $\Pi = P_{k+1}$ and exit. Otherwise, go to step 6.

6. If $(A+B_1B_1^TP_{k+1}, B_2)$ is stabilizable, then increment k by 1 and go back to step 2. Otherwise, exit as there does not exist a real symmetric stabilizing solution $\Pi \geq 0$ satisfying $F(\Pi) = 0$.

From Corollary 1 we see that if the stabilizability condition in step 6 fails at some $k \in \mathbb{Z}_{\geq 0}$, then there does not exist a stabilizing solution $\Pi \geq 0$ to $F(\Pi) = 0$ and the algorithm should terminate (as required by step 6). But when this stabilizability condition is satisfied $\forall k \in \mathbb{Z}_{\geq 0}$, construction of the series P_k and Z_k is always possible and either P_k converges to Π (which is captured by step 5) or P_k just diverges to infinity, which again means that there does not exist a stabilizing solution $\Pi \geq 0$ to $F(\Pi) = 0$.

Remark 1. It is worth pointing out that when the Kleinman algorithm is used to solve (9), how to stop the Kleinman iteration can be an important issue. In fact, a simple criterion to stop the Kleinman iteration is to compute the residue of (9). When the residue of the right-hand side terms of (9) is small enough, the Kleinman iteration should be stopped.

Remark 2. For a numerical method to check the stabilizability of a matrix pair in step 6, one can refer to the staircase algorithm in [34]. In the staircase algorithm, some SVDs are performed to formulate a staircase form of the matrix pair, then the stabilizability of the matrix pair is checked based on the staircase form. Since the computational complexity of SVD is $O(n^3)$ (see [18]), the computational complexity of the staircase algorithm is also $O(n^3)$.

3.3 Rate of Convergence

The following theorem states that the local rate of convergence of the algorithm given in Section 3.2 is quadratic.

Theorem 2. *Given the suppositions of Theorem* 1*, and two series* P_k*,* Z_k *as defined in Theorem* 1 *Part* I*, then there exists a* $\theta > 0$ *such that the rate of convergence of the series* P_k *is quadratic in the region* $\|P_k - \Pi\| < \theta$*.*

Proof. We prove the rate of convergence of P_k by proving the rate of convergence of Z_k. Let $\acute{A}_k = A + B_1B_1^TP_k - B_2B_2^TP_{k+1}$ and $\tilde{A} = A + B_1B_1^T\Pi - B_2B_2^T\Pi$. Note that $\acute{A}_k$ is Hurwitz (see Theorem 1 Part II) and $\tilde{A}$ is Hurwitz since Π is the stabilizing solution of (3) (see [39]), and let $\tilde{Y}$ and $\acute{Y}_k$ be uniquely defined by

$$0 = \tilde{Y}\tilde{A} + \tilde{A}^T\tilde{Y} + I, \tag{10}$$

$$0 = \acute{Y}_k\acute{A}_k + \acute{A}_k^T\acute{Y}_k + I, \tag{11}$$

where I is the identity matrix with appropriate dimensions. The matrices $\tilde{Y}$ and $\acute{Y}_k$ are positive definite because of the stability properties of $\acute{A}_k$ and $\tilde{A}$. Since $\tilde{Y}$ and $\acute{Y}_k$ are uniquely defined and $\lim_{k\to\infty} \acute{A}_k = \tilde{A}$, then $\lim_{k\to\infty} \acute{Y}_k = \tilde{Y}$, and thus for any small $\gamma > 0$, $\exists K_1 \in \mathbb{Z}_{\geq 0}$ such that

$$\overline{\sigma}(\acute{Y}_k - \tilde{Y}) \leq \gamma \quad \forall k \geq K_1.$$

This implies that

$$\overline{\sigma}(\acute{Y}_k) \leq \overline{\sigma}(\tilde{Y}) + \gamma \quad \forall k \geq K_1. \tag{12}$$

Now, define a monotonically non-increasing sequence ε_k by

$$\varepsilon_k = \sup_{m \geq k} \overline{\sigma}(Z_m).$$

From (7) and Theorem 1, we have $\forall k \in \mathbb{Z}_{\geq 1}$:

$$0 = Z_k A_k + A_k^T Z_k - Z_k B_2 B_2^T Z_k + Z_{k-1} B_1 B_1^T Z_{k-1}, \tag{13}$$

which can be equivalently rewritten as follows:

$$0 = Z_k \acute{A}_k + \acute{A}_k^T Z_k + Z_k B_2 B_2^T Z_k + Z_{k-1} B_1 B_1^T Z_{k-1}. \tag{14}$$

Now, there exists $\eta > 0$ (e.g., $\eta = 4\max\{\overline{\sigma}(B_1)^2, \overline{\sigma}(B_2)^2\}$), independent of k, such that

$$Z_k B_2 B_2^T Z_k + Z_{k-1} B_1 B_1^T Z_{k-1} \leq \eta \varepsilon_{k-1}^2 I \quad \forall k \in \mathbb{Z}_{\geq 1}. \tag{15}$$

Multiplying $(\eta \varepsilon_{k-1}^2)$ on each side of (11), we obtain

$$0 = (\eta \varepsilon_{k-1}^2) \acute{Y}_k \acute{A}_k + (\eta \varepsilon_{k-1}^2) \acute{A}_k^T \acute{Y}_k + (\eta \varepsilon_{k-1}^2) I. \tag{16}$$

Then subtracting (14) from (16), we obtain

$$\begin{aligned} 0 = {} & (\eta \varepsilon_{k-1}^2 \acute{Y}_k - Z_k) \acute{A}_k + \acute{A}_k (\eta \varepsilon_{k-1}^2 \acute{Y}_k - Z_k) + \eta \varepsilon_{k-1}^2 I \\ & - (Z_k B_2 B_2^T Z_k + Z_{k-1} B_1 B_1^T Z_{k-1}). \end{aligned} \tag{17}$$

Now note that $\acute{A}_k$ is Hurwitz and the inequality (15) holds, then by (17) we have (see Lemma 3.18 in [39])

$$\eta \varepsilon_{k-1}^2 \acute{Y}_k \geq Z_k. \tag{18}$$

Since (18) holds, $\overline{\sigma}(Z_k) \leq \eta \varepsilon_{k-1}^2 \overline{\sigma}(\acute{Y}_k)$ (see [18]). Hence $\forall k \geq K_1 \geq 1$,

$$\begin{aligned} \varepsilon_k = \sup_{m \geq k} \overline{\sigma}(Z_m) &\leq \sup_{m \geq k} \left[\eta \varepsilon_{m-1}^2 \overline{\sigma}(\acute{Y}_m) \right] \\ &\leq \eta(\overline{\sigma}(\tilde{Y}) + \gamma) \sup_{m \geq k} \varepsilon_{m-1}^2 = \eta(\overline{\sigma}(\tilde{Y}) + \gamma) \varepsilon_{k-1}^2. \end{aligned}$$

Now let $M := \eta(\overline{\sigma}(\tilde{Y}) + \gamma)$ and define $\delta_k := \frac{M}{c} \varepsilon_k$ for $0 < c < 1$, then

$$\delta_k = \frac{M}{c} \varepsilon_k \leq \frac{M^2}{c} \varepsilon_{k-1}^2 = c \frac{M^2}{c^2} \varepsilon_{k-1}^2 = c \delta_{k-1}^2.$$

Thus, $\forall k \geq K_1 \geq 1$ such that $\delta_{k-1} < 1$, we obtain a quadratic rate of convergence, which concludes the proof. □

3.4 Game Theoretic Interpretation of the Algorithm

In this subsection, for the purpose of motivation, interpretation, and further research, a game theoretic interpretation of the algorithm will be given. At the same time, we will also note that this interpretation is closely linked to an optimal control concept of approximation in policy space [8–10]. In this section, we will show that a game theory performance index can be approximated by a series of successive optimal control cost functions. At each iteration, the optimal policy is found to minimize the corresponding cost functions. With the increment of each iteration, the optimal policies approach the final optimum and the saddle point of the cost functions approaches the saddle point of the game theory performance index. In fact, it can be shown (see [29]) that the technique of policy space iteration can be used to replace nonlinear Hamilton–Jacobi partial differential equations by a sequence of linear partial differential equations (even when the approximations and the optimal feedback law are nonlinear functions of the state). This is important because it is difficult [29] to solve Hamilton–Jacobi equations directly to obtain optimal feedback control in many cases.

Consider the dynamical system

$$\dot{x} = Ax + B_1 w + B_2 u \tag{19}$$

with the game theory performance index

$$J(x_0, u, w) = \int_0^\infty \left(u^T u + x^T C^T C x - w^T w\right) dt, \tag{20}$$

where x_0 denotes the initial state of the system, x is the state vector, u denotes the control input, w denotes the disturbance input, and A, B_1, B_2, C are given real matrices with compatible dimensions and appropriate stabilizability/detectability conditions. In this game, u minimizes the cost function J while w maximizes it. It is well known [19] that the optimal control law and the worst case disturbance (a saddle point of $J(x_0, u, w)$) are given by

$$u_{\text{optimal}} = -B_2^T \Pi x, \tag{21}$$

$$w_{\text{worst}} = B_1^T \Pi x, \tag{22}$$

where $\Pi \geq 0$ is the unique stabilizing solution to (3). See [19] for more details on such game theory problems.

Let us now propose a heuristic induction which gives a game theoretic interpretation to our proposed algorithm. Suppose that at iteration k we have a trial control law $u_k = -B_2^T P_k x$ with P_k defined as in Theorem 1 Part I. Then we set $w_k = B_1^T P_k x$. Note that this is NOT the worst case w_k corresponding to $u_k = -B_2^T P_k x$ (unless $P_k = \Pi$), but it is a strategy we wish to impose since it will connect the heuristic ideas of this section to the earlier algorithm. The choice is also motivated by what happens at the optimum, as $P_k \longrightarrow \Pi$ when

$k \longrightarrow \infty$. With this choice of w fixed, we now wish to find a new optimal control, i.e., u now has to minimize the following LQ cost function:

$$J_k(x_0, u) = \int_0^\infty \left(u^T u + x^T C^T C x - x^T P_k B_1 B_1^T P_k x\right) dt, \tag{23}$$

subject to

$$\dot{x} = (A + B_1 B_1^T P_k)x + B_2 u, \tag{24}$$

where $w_k = B_1^T P_k x$ has been substituted in (19) and (20) to yield the above problem. We first consider the following equation:

$$0 = W_k A_k + A_k^T W_k - W_k B_2 B_2^T W_k + F(P_k), \tag{25}$$

where $A_k = A + B_1 B_1^T P_k - B_2 B_2^T P_k$. Since this is the same equation as (9), we conclude that W_k satisfies the same equation as Z_k. Now let $\Lambda_{k+1} = P_k + W_k$, then existence of W_k is equivalent to existence of Λ_{k+1}. Necessary and sufficient conditions for the existence of W_k are the following: (A_k, B_2) is stabilizable and $(F(P_k), A_k)$ has no unobservable modes on the jw-axis. These conditions were analyzed in the proof of Theorem 1 and were shown to be fulfilled via the existence of the stabilizing solution $\Pi \geq 0$ to (3). Under appropriate conditions, the LQ problem defined by (23) and (24) has an optimal solution for u given by

$$u_{k+1} = -B_2^T \Lambda_{k+1} x,$$

where Λ_{k+1} is the unique stabilizing solution to

$$\begin{aligned} 0 =& \Lambda_{k+1}(A + B_1 B_1^T P_k) + (A + B_1 B_1^T P_k)^T \Lambda_{k+1} \\ & - \Lambda_{k+1} B_2 B_2^T \Lambda_{k+1} + (C^T C - P_k B_1 B_1^T P_k). \end{aligned} \tag{26}$$

We will now show that Λ_{k+1} actually equals P_{k+1} as defined in Theorem 1 Part I. To do this, we will equivalently show that $\Lambda_{k+1} - P_k$ equals Z_k. Using $\Lambda_{k+1} = P_k + W_k$ in (26), we have

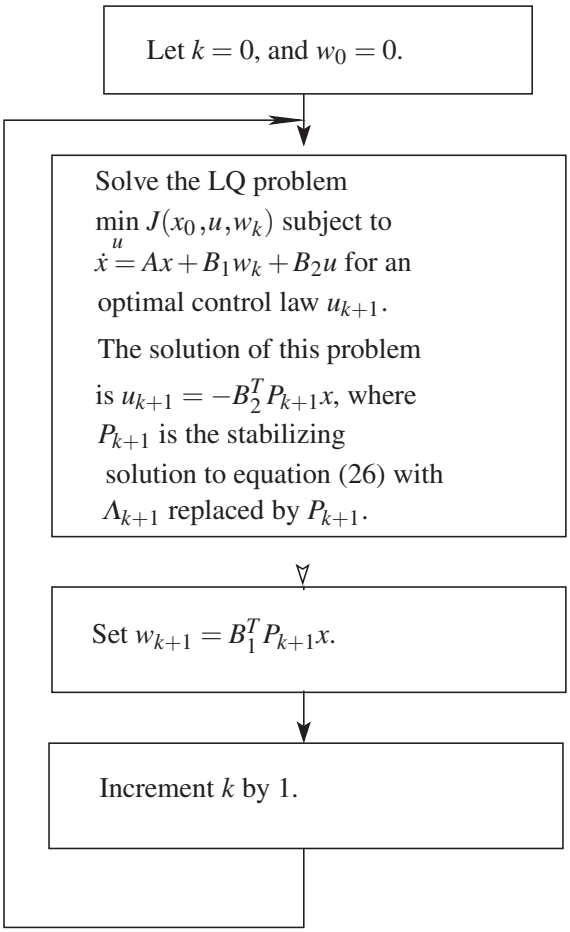
$$\begin{aligned} 0 =& P_k(A + B_1 B_1^T P_k) + W_k(A + B_1 B_1^T P_k) \\ & + (A + B_1 B_1^T P_k)^T P_k + (A + B_1 B_1^T P_k)^T W_k \\ & - P_k B_2 B_2^T P_k - W_k B_2 B_2^T P_k - P_k B_2 B_2^T W_k \\ & - W_k B_2 B_2^T W_k + C^T C - P_k B_1 B_1^T P_k. \end{aligned} \tag{27}$$

The terms above independent of W_k are

$$P_k A + A^T P_k + P_k(B_1 B_1^T - B_2 B_2^T) P_k + C^T C,$$

which are equal to $F(P_k)$. Then (27) reduces to (25). Now note that $\Lambda_{k+1} = P_k + W_k$ is a stabilizing solution to (26), meaning that $A + B_1 B_1^T P_k - B_2 B_2^T (W_k + P_k)$ is Hurwitz. But $A + B_1 B_1^T P_k - B_2 B_2^T (W_k + P_k) = (A_k - B_2 B_2^T W_k)$. Hence W_k also makes $A_k - B_2 B_2^T W_k$ Hurwitz. Since Z_k is the

unique stabilizing solution to (9) and since W_k satisfies the same equation and is also stabilizing, we conclude that $W_k = Z_k$ (see [37]), thereby in turn giving $\Lambda_{k+1} = P_{k+1}$. Since the optimal control law that minimizes cost function (23) subject to constraint (24) is $u_{k+1} = -B_2^T P_{k+1} x$, we now set (according to our game plan) $w_{k+1} = B_1^T P_{k+1} x$ and proceed in this manner as outlined in the following chart:



This heuristic game plan converges to the optimal control (21) and the worst case disturbance (22), thereby giving a game theoretic interpretation to the proposed algorithm.

3.5 Numerical Examples

In this subsection, three examples are given. Example 1 provides a random test to compare our algorithm with the MATLAB command CARE and shows that our algorithm has good efficiency and accuracy when compared with CARE. Example 2 shows that our algorithm still works well when some other approaches (such as the MATLAB command CARE, the Schur method of [28],

and the matrix sign function method of [33]) do not work. Example 3 demonstrates that there in fact does not exist a stabilizing solution $\Pi \geq 0$ to (3) when the stabilizability condition in step 6 of the algorithm is not satisfied, and the sequence P_k diverges quadratically in such a situation.

Example 1

In this example, to show the efficiency and accuracy of our algorithm compared with the MATLAB command CARE, we have a random test including 200 samples (100 examples for the specified tolerance $\Delta = 0.1$ and 100 examples for the specified tolerance $\Delta = 0.01$). Note that the MATLAB command CARE always works well in this example (i.e., the residue for the solutions of AREs obtained by using CARE is always small).

The test procedure is as follows:

1. consider a state-space representation for a dynamic system
$$\dot{x} = Ax + Bu,$$
$$y = Cx + Du,$$
where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times(p+r)}$, $C \in \mathbb{R}^{m\times n}$, and u, x, y are input, state, output, respectively;
2. set the example number $i = 0$;
3. choose n, m, p, r randomly and uniformly among the integers from 1 to 100;
4. generate a random system by using MATLAB command sysrand with n, m, p, r obtained in step 3 and obtain A, B, C, D by MATLAB command unpck;
5. partition the matrix $B = [B_1 \quad B_2]$, where $B_1 \in \mathbb{R}^{n\times p}$ and $B_2 \in \mathbb{R}^{n\times r}$;
6. try MATLAB command CARE to solve the corresponding ARE with $E = I_n, S = 0, B = [B_1 \quad B_2], Q = C^TC, R = \begin{pmatrix} -I_p & 0 \\ 0 & I_r \end{pmatrix}$, where I_n, I_p, and I_r are identity matrices with dimensions n, p, r, respectively. If there does not exist a stabilizing solution to the ARE, go back to step 3;
7. use our algorithm to solve this ARE. For our algorithm, the iteration will be stopped when $\overline{\sigma}(B_1^T Z_k)^2 < \Delta = 0.1$, where Z_k is the matrix sequence defined in Theorem 1 Part I;
8. let the solution of this ARE obtained by our algorithm be X_1 and the solution of this ARE obtained by the MATLAB command CARE be X_2;
9. set $i = i + 1$, and let $T_i = \frac{\|X_1 - X_2\|}{\|X_2\|}$;
10. repeat steps 3–9 until $i = 100$;
11. replace $\Delta = 0.1$ in step 7 by $\Delta = 0.01$, set $i = 0$, and repeat steps 3–10.

For each random example in Tables 1 and 2, "Iterations" indicates the number of necessary iterations to obtain the specified tolerance Δ in step 7; "$O(T_i)$" is the order of magnitude of T_i (defined in step 9) or the order of magnitude

of the normalized comparison error between the stabilizing solutions obtained by our algorithm and the stabilizing solutions obtained by the MATLAB command CARE; "Number of examples" means the number of random examples that required "Iterations" number to converge. From Tables 1 and 2, we can conclude that our proposed algorithm works well in most cases with good efficiency (only three to five iterations in most examples) and accuracy when compared with the MATLAB command CARE.

Table 1. Illustration of iteration count of our algorithm and accuracy comparison with CARE for 100 random examples ($\overline{\sigma}(B_1^T Z_k)^2 < \Delta = 0.1$)

Iterations / $O(T_1)$	2	3	4	5	6	7	8
10^{-4}	1	5	4		1	1	
10^{-5}		7	6	7	1	2	3
10^{-6}		13	10	3	2		
10^{-7}		5	3	3	1	1	
10^{-8}		1	4	7			
10^{-9}		4	3				
10^{-10}			2				
10^{-11}							
10^{-12}							
10^{-13}							
Number of examples	1	35	32	20	5	4	3

A summary of the results in Tables 1 and 2 is as follows:

1. In both Tables 1 and 2, our proposed algorithm converges in ONLY three to five iterations in most examples.
2. When the prescribed tolerance is $\Delta = 0.1$, we have $T_i < 10^{-3}$ for each random example in Table 1; when the prescribed tolerance is $\Delta = 0.01$, we have $T_i < 10^{-5}$ for each random example in Table 2.

Example 2

The following example illustrates that the proposed algorithm works well when other traditional methods fail. This example is a slight modification of Example 6 in [28]. Choose the matrices $A \in \mathbb{R}^{21\times 21}, B_1 \in \mathbb{R}^{21\times 1}, B_2 \in \mathbb{R}^{21\times 1}, C \in \mathbb{R}^{21\times 21}$ in (3) as follows:

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ & \ddots & \ddots & \bigcirc & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ & & & \ddots & 1 \\ 0 & & \cdots & & 0 \end{pmatrix}, \quad B_1 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \delta \\ 0 \end{pmatrix}, \quad B_2 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

Table 2. Illustration of iteration count of our algorithm and accuracy comparison with CARE for 100 random examples ($\overline{\sigma}(B_1^T Z_k)^2 < \Delta = 0.01$)

Iterations / O(T$_1$)	2	3	4	5	6	7	8	9	10	11
10^{-4}										
10^{-5}										
10^{-6}		1	2	1	2	1				
10^{-7}	1	6	7	5	2	1	1		2	1
10^{-8}	1	6	2	4	1	1	2	1		
10^{-9}		2	11	1	1	1				
10^{-10}		1	5	2	1	2	1			
10^{-11}			1	4	1	2		1		1
10^{-12}			3		1					
10^{-13}			4	1	2					
Number of examples	2	16	35	18	11	8	4	2	2	2

$$C = \text{diag}\{1, 0, \cdots, 0\},$$

where $\delta = 10^{-2}$ is the introduced modification. In this example, the Schur method in [28] does not produce an accurate result, similarly to the MATLAB command CARE. The algorithm proposed by Kleinman in [24] cannot be used because the term $(B_2 B_2^T - B_1 B_1^T)$ in the Riccati equation (3) is not positive semidefinite. However, the algorithm proposed in Section 3.2 easily gives the solution with the specified accuracy as will be shown next.

First we attempt the Schur method in [28] with this example. Let F be defined as in (4), and S_1 be the solution to (3) by using this Schur method. We evaluate the accuracy of the solution S_1 by calculating $\rho[F(S_1)]$. The smaller $\rho[F(S_1)]$ is, the closer S_1 is to the correct solution. After calculation, we obtain $\rho[F(S_1)] = 1.9802 \times 10^3$ which is far too large. Thus, we can conclude that the Schur method in [28] fails to give an accurate solution in this example. Similarly, let S_2 be the solution obtained by the MATLAB command CARE. For this solution, we can obtain $\rho[F(S_2)] = 1.9811 \times 10^3$ which again is too large. So we conclude that MATLAB command CARE also fails to give a solution in this example. If we were to try to refine the very coarse solution obtained by the Schur method in [28] using Kleinman's method in [24], this too fails as this algorithm diverges with each iteration (as expected). This can be shown as follows: let X_k with $k \in \mathbb{Z}_{\geq 1}$ denote the iterative series produced by the Kleinman algorithm, then we obtain $\rho[F(X_1)] = 5.7083 \times 10^2$, $\rho[F(X_2)] = 5.9959 \times 10^2$, ... , $\rho[F(X_{20})] = 8.2965 \times 10^7$, ... , $\rho[F(X_{100})] = 6.6206 \times 10^9$. If we use the matrix sign function method in [33] to solve this ARE and let Q be the corresponding solution, we obtain $\rho[F(Q)] = 1.235 \times 10^8$ which is again far too large.

However, when we use our proposed algorithm, we note that a unique stabilizing solution $P_4 > 0$ to (3) can be found with limiting accuracy after only four iterations with $\rho[F(P_4)] = \overline{\sigma}(B_1^T Z_3)^2 = 2.9205 \times 10^{-5}$.

Example 3

The following example shows that if $(A+B_1B_1^TP_{k+1}, B_2)$ is not stabilizable in step 5 of the algorithm, then there does not exist a stabilizing solution $\Pi \geq 0$ to (3). Choose

$$A = \begin{pmatrix} 1 & 100 \\ 1 & 0 \end{pmatrix},\ B_1 = \begin{pmatrix} 10 \\ 0 \end{pmatrix},\ B_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix},\ C = \begin{pmatrix} 1\ 0 \end{pmatrix}.$$

We note that (C, A) has no unobservable modes on the $j\omega$-axis and (A, B_2) is stabilizable. When we run our algorithm, we find that $(A + B_1B_1^TP_1, B_2)$ is not stabilizable after one iteration since

$$P_1 = \begin{pmatrix} 0.3517 & 2.6442 \\ 2.6442 & 22.9964 \end{pmatrix}.$$

This is consistent with the fact that there does not exist a stabilizing solution $\Pi \geq 0$ to (3). In fact, we can find the unique stabilizing solution

$$\Pi = \begin{pmatrix} -0.5744 & -4.9147 \\ -4.9147 & -37.8481 \end{pmatrix},$$

which is clearly not positive semidefinite. Meanwhile, we find that the sequence P_k diverges quadratically on noting that $\rho[F(P_1)] = 7.1156 \times 10^2,\ \rho[F(P_2)] = 5.9180 \times 10^3, \ldots,\ \rho[F(P_{16})] = 4.7489 \times 10^{15}$.

4 Solving an HJB Equation by a Sequence of Linear PDEs

As we indicated in the introduction section, for our algorithm in the nonlinear case, we replace the problem of solving HJBI equations by the problem of solving a sequence of HJB equations. However, HJB equations are first-order, generally nonlinear partial differential equations and difficult to solve in general. In this section, we review the technique of [29]. By using the technique of [29], we can replace the problem of solving an HJB equation by the problem of solving a sequence of linear PDEs; by doing so, we transfer a difficult problem into a sequence of less difficult problems.

Consider a dynamical system represented by

$$\dot{x} = f(x, u, t), \quad x(t_0) = x_0, \tag{28}$$

where the n-vector x is the plant state, f is a continuously differentiable n-vector function, and $u(x,t) \in \mathbb{R}^n \times \mathbb{R}$ is an r-vector function defined on $\mathbb{R}^n \times \mathbb{R}$. The solution of (28) is denoted as $\phi_u(t) = \phi_u(t; x_0, t_0)$.

Let G be a closed subset of $\mathbb{R}^n \times \mathbb{R}$ to which all motions of the system (28) are restricted. Let the target set S be a closed subset of G. In this section, the function u is called an *admissible feedback control law* if

(1) it is continuously differentiable with values $u(x,t)$ belonging to a locally compact set $U_0 \in \mathbb{R}^r$ for all t;
(2) it has the property that when substituted into (28), any motion beginning in $G - S$ reaches S, or approaches S, in a uniform asymptotic manner without leaving G.

The class of functions satisfying the above properties is denoted by U_1. The terminal time $t_1 = t_1(x_0, t_0)$ is defined to be the time when the motion $(\phi_u(t), t)$ becomes a member of S, or, in the asymptotic case, $t_1 = \infty$. Note that in the finite time case, S itself might be simply $\mathbb{R}^n \times t_1$. (Indeed, we shall focus in what follows on this possibility).

In the following, we first consider the situation when t_1 is finite, then consider the situation when t_1 is infinite.

4.1 t_1 Is Finite

The system performance is evaluated by the functional

$$J(x_0, t_0, t_1; u) = \lambda[\phi_u(t_1), t_1] + \int_{t_0}^{t_1} L[\phi_u(\alpha), u(\phi_u(\alpha), \alpha), \alpha] d\alpha, \tag{29}$$

where L and λ are nonnegative scalar, continuously differentiable functions. It is assumed that L is strictly convex in u.

We define

$$V^0(x_0, t_0, t_1) = \inf_{u \in U_1} J(x_0, t_0, t_1; u). \tag{30}$$

Let H be defined as

$$H(x, p, t, u) = \langle f(x, u, t), p \rangle + L(x, u, t), \tag{31}$$

where p is an n-vector, u is an r-vector, and $<,>$ denotes the inner product. Assume that H has a unique absolute minimum for each x, p, and t with respect to the values $u \in U_0$, and let the associated location of minimum be denoted as $c(x, p, t)$. Assuming that c is a continuously differentiable function of x, p, and t, we define the Hamiltonian as

$$H^0(x, p, t) = H(x, p, t, c(x, p, t)) = \min_{u \in U_0} H(x, p, t, u). \tag{32}$$

The HJB equation we consider is

$$0 = V_t + H^0(x, V_x, t) \tag{33}$$

subject to the boundary condition

$$V(x,t_1)=\lambda(x,t_1),$$

where $V(x,t)$ is a scalar function defined on $\mathbb{R}^n\times\mathbb{R}$, $V_t=\frac{\partial V}{\partial t}$, and $V_x=\frac{\partial V}{\partial x}$. It can be shown [23] that if $V(x,t)$ is twice continuously differentiable in all arguments, if it satisfies (33) in G and the boundary condition $V(x,t_1)=\lambda(x,t_1)$ on S, and in addition if the function $u^0(x,t)=c(x,V_x,t)$ is admissible, then $V(x,t)=V^0(x,t)$. It is shown in [29] that for any optimal control problem described by (28), (29), and (30), we have

$$\frac{dJ}{dt}(\phi_u(t;x_0,t_0),t;u)=-L[\phi_u(t;x_0,t_0),u(\phi_u(t;x_0,t_0),t),t], \tag{34}$$

with $V(x,t)=\lambda(x,t)$ or, denoting $\phi_u(t;x_0,t_0)$ by x, we have

$$\dot{J}(x,t;u)=-L(x,u,t), \tag{35}$$

with $V(x,t)=\lambda(x,t)$. Given any optimal feedback control described by (28), (29) and (30), we define $\mathcal{V}$ as the set of all continuously differentiable functions $V:\mathbb{R}^n\times\mathbb{R}\to\mathbb{R}$ such that $V(x,t_1)=\lambda(x,t_1)$ on S. Let $\mathcal{V}^0$ be the subset of $\mathcal{V}$ such that if $u(x,t)=c(x,V_x,t)$ then u is admissible.

The following theorem comes from [29], and it constructs a monotone nonincreasing function sequence which converges to the solution of (33). Meanwhile, it provides a technique of transferring an HJB equation into a sequence of linear PDEs.

Theorem 3. [29] *Define a mapping $T:\mathcal{V}_0\to\mathcal{V}$ for $V\in\mathcal{V}_0$ and $T(V)=J(x,t;u)$ with $u(x,t)=c(x,V_x,t)$. Suppose $T(V^n)\in\mathcal{V}^0$ for $n=1,2,3,\ldots$. If $V^1\in\mathcal{V}^0$ and $V^{n+1}=T(V^n)$, $n=1,2,3,\ldots$, then*

$$V(x,t)\le V^{n+1}(x,t)\le V^n(x,t)\le V^1(x,t), \tag{36}$$

where V is the solution of (33). For every sequence V^n, there exists a function V^* such that $V^n(x,t)\downarrow V^*(x,t)$ pointwise on G. If G is bounded, the convergence is uniform. If T is continuous in $\mathcal{V}_0\subset\mathcal{V}$ and $V^*\in\mathcal{V}^0$, then $V^*=V$. Furthermore, suppose V^1 is given, then the sequence V^n can be recursively obtained by solving the following sequence of linear PDEs:

$$0=\left\langle V_x^{n+1},f(x,c(x,V_x^n,t),t)\right\rangle+V_t^{n+1}+L(x,c(x,V_x^n,t),t),$$

with the boundary condition $V^n(x,t_1)=\lambda(x,t_1)$.

Proof. See [29]. □

4.2 t_1 Is Infinite

In this subsection, we assume that there exists a solution of HJB equation (33) which minimizes the cost function (29) when t_1 is infinite. In such a situation, it is clear that the scalar function L in (29) approaches zero when $t_1\to\infty$.

Evidently, for infinite time problems, key interest centers around stabilizing control laws. This motivates us to make the following assumption, which generalizes controllability/stabilizability and observability/detectability assumptions commonly made for H_2 problems.

Assumption A1 There exists a feedback law for (28) ensuring that the closed-loop system is uniformly asymptotically stable with associated finite performance index, and if a feedback control law ensures that the achieved performance with $t_1 = \infty$ and an arbitrary $x(t_0)$ is finite, then the associated closed-loop system is uniformly asymptotically stable.

The following theorem gives an infinite time version of Theorem 3.

Theorem 4. *Let the mapping $T : \mathcal{V}_0 \to \mathcal{V}$ be as defined in Theorem 3 and suppose V is a solution of (33) with infinite terminal time t_1. Suppose $T(V^n) \in \mathcal{V}^0$ for $n = 1, 2, 3, \ldots$. If $V^1 \in \mathcal{V}^0$ and $V^{n+1} = T(V^n)$, $n = 1, 2, 3, \ldots$, then*

$$V(x,t) \le V^{n+1}(x,t) \le V^n(x,t) \le V^1(x,t), \tag{37}$$

where V is the solution of (33). For every sequence V^n, there exists a function V^* such that $V^n(x,t) \downarrow V^*(x,t)$ pointwise on G. If T is continuous in $\mathcal{V}_0 \subset \mathcal{V}$ and $V^* \in \mathcal{V}^0$, then $V^* = V$. Furthermore, suppose a bounded V^1 is chosen such that the system $\dot{x} = f(x, c(x, V_x^1, t), t)$ is uniformly asymptotically stable, then the sequence V^n can be recursively obtained by solving the following sequence of linear PDEs:

$$0 = \left\langle V_x^{n+1}, f(x, c(x, V_x^n, t), t) \right\rangle + V_t^{n+1} + L(x, c(x, V_x^n, t), t),$$

with the boundary condition

$$\lim_{t_1 \to \infty} V^n(x, t_1) = \lim_{t_1 \to \infty} \lambda(x, t_1),$$

and the closed-loop system $\dot{x} = f(x, c(x, V_x^n, t), t)$ is uniformly asymptotically stable for $n = 1, 2, \ldots$.

Proof. To show the algorithm in [29] holds for infinite time t_1, we can first show that (35) holds for infinite t_1 by using the argument in [29]. Then by using similar argument in Lemma 2, Theorem 2, and Theorem 5 of [29], we can construct the sequence V^n satisfying (37). The convergence of V^n can be proved by using the argument in Lemma 3 and Theorem 7 in [29]. We now use an inductive argument to prove the uniform asymptotical stability of the closed-loop system $\dot{x} = f(x, c(x, V_x^n, t), t)$. It is clear that the closed-loop system $\dot{x} = f(x, c(x, V_x^1, t), t)$ is uniformly asymptotically stable. Now we assume that the closed-loop system $\dot{x} = f(x, c(x, V_x^n, t), t)$ is uniformly asymptotically stable for $n = k$, then we need to show that the closed-loop system $\dot{x} = f(x, c(x, V_x^n, t), t)$ is uniformly asymptotically stable for $n = k+1$. Note that $V^1(x,t)$ is finite, then by (37), $V^k(x,t)$ and $V^{k+1}(x,t)$ are both finite. Then by Assumption A1, we conclude that the closed-loop system $\dot{x} = f(x, c(x, V_x^{k+1}, t), t)$ is also uniformly asymptotically stable, which completes the proof. □

Based on the observations of the Kleinman algorithm and the algorithm in Theorems 3 and 4, we can find there are some similarities between them: both these two algorithms are used to solve equations arising in optimal control; in both these two algorithms, monotone non-increasing sequences are constructed to approximate the solutions of the desired equations (i.e., Riccati equations or HJB equations). In view of these similarities, we can reasonably suppose that the algorithm in [29] is a generalization of the Kleinman algorithm, and that it will have some similar features to those of the Kleinman algorithm, for example, a local quadratic rate of convergence. The second point is now under consideration and should be resolved in the near future.

5 Solving an HJBI Equation by a Sequence of HJB Equations

As we indicated in the introduction section, HJBI equations arise in nonlinear H_∞ control and they are first-order nonlinear partial differential equations and more difficult to solve than HJB equations. In this section, we present the recursive algorithm to solve HJBI equations. By using our algorithm, we replace the problem of solving an HJBI equation by the problem of solving a sequence of HJB equations; then by using the technique in Section 4, we can, if we wish, transfer each of these HJB equations to a sequence of linear PDEs. There are of course existing methods to solve HJBI equations, such as the method in [36] and the Galerkin approximation method in [7]. However, as we indicated in Section 1, there are some clear disadvantages for these algorithms. For example, when the methods in [36] and [7] are used to solve HJBI equations, it is difficult to choose an initial condition. Compared with the methods in [36] and [7], our algorithm has a very simple initial condition, viz, $V_0 = 0$. There are five subsections in this section: (1) preliminaries and definitions; (2) the summarizing theorem; (3) the algorithm; (4) rate of convergence and game theoretic interpretation of our algorithm; (5) a numerical example.

5.1 Preliminaries and Definitions

We work with the nonlinear control system $\Gamma : \mathcal{U}_0 \times \mathcal{W}_0 \to \mathcal{Y}_0$ given by the following equations:

$$x(0) = x_0, \tag{38}$$

$$\dot{x}(t) = f(x) + g_1(x)w(t) + g_2(x)u(t), \tag{39}$$

$$y(t) = h(x), \tag{40}$$

where $x \in \mathcal{X}_0$ is the state; $x_0 \in \mathbb{X}_0$ is the initial state; $u \in \mathcal{U}_0$ is the input; $w \in \mathcal{W}_0$ is the disturbance; $y \in \mathcal{Y}_0$ is the output. $f : \mathbb{X}_0 \to \mathbb{R}^n, g_1 : \mathbb{X}_0 \to \mathbb{R}^{n\times q}$,

$g_2 : \mathbb{X}_0 \to \mathbb{R}^{n\times m}$, and $h : \mathbb{X}_0 \to \mathbb{R}^p$ are smooth functions with $f(0) = 0$ and $h(0) = 0$. It is assumed further that f, g_1, g_2 are such that (39) has a unique solution for any $u \in \mathcal{U}_0$, $w \in \mathcal{W}_0$, and $x_0 \in \mathbb{X}_0$. Throughout Section 5, it is further assumed that the functions f, g_1, g_2, h defined in the system Γ can be represented in the following form:

$$f(x) = Ax + f_r(x), \tag{41}$$
$$g_1(x) = B_1 + g_{1r}(x), \tag{42}$$
$$g_2(x) = B_2 + g_{2r}(x), \tag{43}$$
$$h(x) = Cx + h_r(x), \tag{44}$$

where $x := x(t)$, A, B_1, B_2, C are real constant matrices with suitable dimensions and $f_r(x)$, $g_{1r}(x)$, $g_{2r}(x)$, $h_r(x)$ are higher order remainder terms in power expansions.

The steady-state HJBI equation associated with the system Γ we treat in Section 5 is

$$\begin{aligned} 0 &= 2\left(\frac{\partial \Pi(x)}{\partial x}\right)^T f(x) + \left(\frac{\partial \Pi(x)}{\partial x}\right)^T \big(g_1(x)g_1^T(x) \\ &\quad - g_2(x)g_2^T(x)\big)\left(\frac{\partial \Pi(x)}{\partial x}\right) + h^T(x)h(x), \\ \Pi(0) &= 0 \end{aligned} \tag{45}$$

where f, g_1, g_2, h are real functions in the system Γ, $x \in \mathbb{X}_0$ is the state vector of the system Γ, and $\Pi : \mathbb{X}_0 \to \mathbb{R}^+$ is the unique local nonnegative stabilizing solution we seek. Here, a solution of (45) is called a local *stabilizing solution* if this solution is such that the closed loop of the system Γ is locally exponentially stable under the feedback inputs $u^* = -g_2^T(x)\frac{\partial \Pi(x)}{\partial x}$ and $w^* = g_1^T(x)\frac{\partial \Pi(x)}{\partial x}$. In such a situation, the vector field

$$\tilde{f}(x) = f(x) + g_1(x)g_1^T(x)\frac{\partial \Pi(x)}{\partial x} - g_2(x)g_2^T(x)\frac{\partial \Pi(x)}{\partial x} \tag{46}$$

is locally exponentially stable at the equilibrium point $x^* = 0$.

In the remainder of this subsection, we give some definitions which will be useful in our summarizing theorem.

We now define linear stabilizability of a matrix function pair by the stabilizability of the linear parts of this matrix function pair.

Definition 1. *[30] Let f, g_2 be the real functions defined in the system Γ and suppose* (41) *and* (43) *hold. The pair (f, g_2) is called linearly*[1] *stabilizable if there exists a matrix $\hat{D}$ such that $(A + B_2\hat{D})$ is a Hurwitz matrix.*

[1] There exists nonlinear systems which cannot be stabilized by linear controllers, for example, the nonlinear system $\dot{x} = \sqrt{x} + u$. This is the reason for the term linearly stabilizable, as opposite to "stabilizable."

Next, we define a function Θ, motivated by the right-hand side of the HJBI equation (45) that will be useful throughout Section 5.

Definition 2. *Let f, g_1, g_2, h be the real vector functions defined in the system Γ, and $x \in \mathbb{X}_0$ be the state value of Γ. Let $\mathbb{T}$ be the set which includes all smooth mappings from $\mathbb{X}_0$ to $\mathbb{R}$ and define $\Theta : \mathbb{T} \to \mathbb{T}$ as*

$$(\Theta(V))(x) = 2\left(\frac{\partial V(x)}{\partial x}\right)^T f(x) + \left(\frac{\partial V(x)}{\partial x}\right)^T \big(g_1(x)g_1^T(x) - g_2(x)g_2^T(x)\big)\left(\frac{\partial V(x)}{\partial x}\right) + h^T(x)h(x) \tag{47}$$

for all $V \in \mathbb{T}$, $x \in \mathbb{X}_0$.

We define two functions $\hat{f}_V$ and $\bar{f}_V$ which will be used to simplify the expressions in our main results and the HJB equations in our proposed algorithm (see (51) for example).

Definition 3. *Let f, g_1, g_2, h be the real vector functions defined in the system Γ. Let $\mathbb{T}, \Theta$ be defined as in Definition 2. Suppose there exists a local nonnegative stabilizing solution $\Pi \in \mathbb{T}$ to (45). Let $V \in \mathbb{T}$, let $\hat{f}_V : \mathbb{X}_0 \to \mathbb{R}$ be defined as*

$$\hat{f}_V(x) = f(x) + g_1(x)g_1^T(x)\frac{\partial V(x)}{\partial x} - g_2(x)g_2^T(x)\frac{\partial V(x)}{\partial x}$$

for all $x \in \mathbb{X}_0$, and let $\bar{f}_V : \mathbb{X}_0 \to \mathbb{R}$ be defined as

$$\bar{f}_V(x) = f(x) + g_1(x)g_1^T(x)\frac{\partial V(x)}{\partial x} - g_2(x)g_2^T(x)\frac{\partial \Pi(x)}{\partial x}$$

for all $x \in \mathbb{X}_0$.

5.2 The Summarizing Theorem

In this subsection, we set up the summarizing theorem by constructing two nonnegative function series $Z_k(x)$ and $V_k(x)$, and we also prove that $V_k(x)$ is monotonically increasing and converges to the unique local nonnegative stabilizing solution $\Pi(x)$ of (45) if such a solution exists.

Theorem 5. *Consider the system Γ, and let A, B_1, B_2, C be the real matrices appearing in (41), (42), (43), and (44), respectively. Let $x \in \mathbb{X}_0$ be the state of the system Γ. Define $\Theta : \mathbb{T} \to \mathbb{T}$ as in (47). Suppose (C, A) is detectable, (A, B_2) is stabilizable and there exists a stabilizing solution $\Pi \geq 0$ to (3). Let A_k, Z_k, and P_k be the matrix sequences appearing in Theorem 1. Then*

(I) *there exists a unique local nonnegative stabilizing solution $\Pi(x)$ to (45);*

(II) *two unique real function sequences* $Z_k(x)$ *and* $V_k(x)$ *for all* $k \in \mathbb{Z}_{\geq 0}$ *can be defined recursively as follows:*

$$V_0(x) \;=\; 0 \quad \forall x \in \mathbb{X}_0, \tag{48}$$

$Z_k(x)$ *is the unique local nonnegative stabilizing solution of*

$$\begin{aligned} 0 \;=\; & 2\hat{f}_{V_k}^T(x)\frac{\partial Z_k(x)}{\partial x} - \left(\frac{\partial Z_k(x)}{\partial x}\right)^T g_2(x)g_2^T(x)\frac{\partial Z_k(x)}{\partial x} + \\ & + \left(\Theta(V_k)\right)(x) \quad \forall x \in \mathbb{X}_0 \end{aligned} \tag{49}$$

with $0 = Z_k(0),\ 0 = \frac{\partial Z_k(x)}{\partial x}\Big|_{x=0}$, *and then*

$$V_{k+1} \;=\; V_k + Z_k; \tag{50}$$

(III) *the two series* $V_k(x)$ *and* $Z_k(x)$ *in part* (II) *have the following properties*:

(1) $(f(x)+g_1(x)g_1^T(x)\frac{\partial V_k(x)}{\partial x}, g_2(x))$ *is linearly stabilizable* $\forall k \in \mathbb{Z}_{\geq 0}\ \forall x \in \mathbb{X}_0$,

(2) $(\Theta(V_{k+1}))(x) = \left(\frac{\partial Z_k(x)}{\partial x}\right)^T g_1(x)g_1^T(x)\frac{\partial Z_k(x)}{\partial x}\ \forall k \in \mathbb{Z}_{\geq 0}\ \forall x \in \mathbb{X}_0$,

(3) $f(x) + g_1(x)g_1^T(x)\frac{\partial V_k(x)}{\partial x} - g_2(x)g_2^T(x)\frac{\partial V_{k+1}(x)}{\partial x}$ *is locally exponentially stable at the origin* $\forall k \in \mathbb{Z}_{\geq 0} \quad \forall x \in \mathbb{X}_0$,

(4) $\Pi(x) \geq V_{k+1}(x) \geq V_k(x) \geq 0\ \forall k \in \mathbb{Z}_{\geq 0}\ \forall x \in\ \mathbb{X}_0$,

(5) $Z_k(x) = \frac{1}{2}x^T Z_k x + O_0(x)\ \forall k \in \mathbb{Z}_{\geq 0}\ \forall x \in \mathbb{X}_0$,
$V_k(x) = \frac{1}{2}x^T P_k x + O_1(x)\ \forall k \in \mathbb{Z}_{\geq 0}\ \forall x \in \mathbb{X}_0$,

where Z_k *and* P_k *are the matrix sequences appearing in Theorem 1 and* $O_0(x)$ *and* $O_1(x)$ *are terms of higher order than quadratic.*

(IV). *For all* $x \in \mathbb{X}_0$, *the limit*

$$V_\infty(x) := \lim_{k\to\infty} V_k(x)$$

exists with $V_\infty(x) \geq 0$. *Furthermore,* $V_\infty = \Pi$ *is the unique local nonnegative stabilizing solution to* (45).

Proof. See [17]. □

From Theorem 5 (I) and (III1), we know that if there exists a local nonnegative stabilizing solution of (45), then (III1) holds. However, by Definition 1, we can check the linear stabilizability of a matrix function pair by checking its linear part. Hence we have the following corollary which gives a condition under which there does *not* exist a local stabilizing solution $\Pi(x) \geq 0$ to $\Theta(\Pi) = 0$. This is useful for terminating the recursion after a finite number of iterations.

Corollary 2. *Let* A, B_1, B_2, C *be the real matrices appearing in* (41), (42), (43), *and* (44). *Let* P_k *be the matrix sequence appearing in Theorem* 1. *Suppose*

that (C, A) is detectable and (A, B_2) is stabilizable. Let $x \in \mathbb{X}_0$ be the state of the system Γ. Define $\Theta : \mathbb{T} \to \mathbb{T}$ as in Definition 2. If $\exists k \in \mathbb{Z}_{\geq 0}$ such that $(A + B_1 B_1^T P_k, B_2)$ is not stabilizable, then there does not exist a local nonnegative stabilizing solution to $\Theta(\Pi) = 0$.

Proof. See [17]. □

Remark 3. It is worth pointing out that the sequence of HJB equations (49) is associated with a sequence of nonlinear optimal control problems; hence by using our algorithm, we have transferred a nonlinear H_∞ control problem into a sequence of nonlinear optimal control problems.

5.3 Algorithm

Let f, g_1, g_2, h be the real functions defined in the system Γ and suppose (41), (42), (43), and (44) hold. Let P_k be the matrix sequence appearing in Theorem 1. Let $\hat{f}_V$ be defined in Definition 3. Suppose (A, B_2) is stabilizable and (C, A) is detectable; an iterative algorithm for finding the local nonnegative stabilizing solution of (45) is given as follows:

1. Let $V_0 = 0$ and $k = 0$.
2. Construct (for example, using the algorithm in Theorem 4, though this is not necessary) the unique local nonnegative stabilizing solution $Z_k(x)$ which satisfies
$$0 = 2\hat{f}_{V_k}^T(x)\frac{\partial Z_k(x)}{\partial x} - \left(\frac{\partial Z_k(x)}{\partial x}\right)^T g_2(x) g_2^T(x) \frac{\partial Z_k(x)}{\partial x} + (\Theta(V_k))(x), \quad (51)$$
with $0 = Z_k(0)$, $0 = \frac{\partial Z_k(x)}{\partial x}\Big|_{x=0}$, where Θ is defined in Definition 2 and $\hat{f}$ is defined in Definition 3.
3. Set $V_{k+1}(x) = V_k(x) + Z_k(x)$.
4. Rewrite $Z_k(x) = \frac{1}{2} x^T Z_k x + O_0(x)$ (note that this is always possible from Theorem 5 if $Z_k(x)$ exists), where $O_0(x)$ are terms of higher order than quadratic and $Z_k \geq 0$ is the matrix sequence appearing in Theorem 1.
5. If $\overline{\sigma}(Z_k) < \mu$ where μ is a specified accuracy, then set $\Pi(x) = V_{k+1}(x)$ and exit. Otherwise, go to step 6.
6. If $(A + B_1 B_1^T P_k, B_2)$ is stabilizable, then increment k by 1 and go back to step 2. Otherwise, exit as there does not exist a local nonnegative stabilizing solution Π satisfying $\Theta(\Pi) = 0$.

From Corollary 2 we see that if the stabilizability condition in step 6 fails for some $k \in \mathbb{Z}_{\geq 0}$, then there does not exist a local nonnegative stabilizing solution Π to $\Theta(\Pi) = 0$ and the algorithm should terminate (as required by step 5). But when this stabilizability condition is satisfied $\forall\ k \in \mathbb{Z}_{\geq 0}$, construction of the series $V_k(x)$ and $Z_k(x)$ is always possible and either $V_k(x)$ converges to $\Pi(x)$ (which is captured by step 5) or $V_k(x)$ just diverges to infinity, which again means that there does not exist a stabilizing solution $\Pi(x) \geq 0$ to (45).

5.4 Rate of Convergence and Game Theoretic Interpretation

It can be shown that our algorithm to solve HJBI equations has a local quadratic rate of convergence and a game theoretic interpretation. Due to space restrictions, we omit the two parts here, please see [17] for more details.

5.5 A Numerical Example

In this subsection, a numerical example will be given. The example provides a numerical comparison between the method of characteristics [38] and our algorithm to solve an HJBI equation arising in nonlinear systems, and it shows that our proposed algorithm converges faster than the method of characteristics for this particular example.

Example 1. In [38], the method of characteristics is used to solve HJBI equations recursively. The following example comes from [36], and it illustrates the proposed algorithm outperforming the method of characteristics in [38] when solving HJBI equations. The comparison is possible because in this particular case, we are able to obtain the exact solution of the HJBI equation. The scalar system is given by

$$\dot{x}(t) = u(t) + xw(t) \tag{52}$$

with output $y(t) = x$. For this example, we have $f(x) = 0,\ g_1(x) = x,\ g_2(x) = 1,\ h(x) = 1,\ \ A = 0,\ \ B_1 = 0,\ B_2 = 1,\ C = 1$ and it is clear that (A, B_2) is stabilizable and (C, A) is detectable. Now the steady-state HJBI equation becomes

$$x^2 - \left(\frac{\partial \Pi(x)}{\partial x}\right)^2 (1 - x^2) = 0, \tag{53}$$

with $\Pi(0) = 0$. We have (without any approximation)

$$\frac{\partial \Pi(x)}{\partial x} = \pm \frac{x}{\sqrt{1 - x^2}}\ \forall x \in (-1, 1), \quad \Pi(0) = 0. \tag{54}$$

a. Exact solution

Since $\Pi(0) = 0$ and we seek the solution for which $\Pi(x) \geq 0$ in a neighborhood of the origin, we have

$$\frac{\partial \Pi(x)}{\partial x} = \frac{x}{\sqrt{1 - x^2}} \tag{55}$$

for $-1 < x < 1$. Now the closed-loop saddle point solution for the system (52) is $u^*(x) = -\frac{x}{\sqrt{1-x^2}}, w^*(x) = \frac{x^2}{\sqrt{1-x^2}}$ and the closed loop of the system (52) under the saddle point inputs u^* and w^* is

$$\dot{x} = -x\sqrt{1 - x^2} \tag{56}$$

for $-1 < x < 1$. Then it is clear that $x^* = 0$ is a local stable equilibrium point for the system (56). We approximate the value of $\Pi(x)$ by approximating the value of $\frac{\partial \Pi(x)}{\partial x}$. From (55), we know that the value of $\Pi(x)$ is symmetric about the origin. In view of this, we only approximate the value of $\frac{\partial \Pi(x)}{\partial x}$ for $0 \leq x < 1$ in the following. The exact solution of $\frac{\partial \Pi(x)}{\partial x}$ in (53) can be approximated by both our algorithm and the method of characteristics in [38].

b. Our algorithm

To approximate $\frac{\partial \Pi(x)}{\partial x}$ in (53), we carry out our proposed algorithm from Section 5.3. For convenience, we denote $(\cdot)_{k,x} = \frac{\partial (\cdot)_k}{\partial x}$ in the following for $k = 0, 1, 2, 3$. After a straightforward computation, we obtain the first three approximations $V_{1,x}, V_{2,x}, V_{3,x}$ of $\frac{\partial \Pi(x)}{\partial x}$ in (53) as follows:

$$\begin{aligned}
V_{1,x} &= Z_{0,x} = x, \\
Z_{1,x} &= x^3 - x + x\sqrt{x^4 - x^2 + 1}, \\
V_{2,x} &= x^3 + x\sqrt{x^4 - x^2 + 1}, \\
Z_{2,x} &= f_2 + \sqrt{f_2^2 + x^2 Z_{1,x}^2}, \\
V_{3,x} &= x^5 + x^3\sqrt{x^4 - x^2 + 1} + \sqrt{f_2^2 + x^2 Z_{1,x}^2},
\end{aligned}$$

where $f_2 = x^5 - x^3 + (x^3 - x)\sqrt{x^4 - x^2 + 1}$.

c. Algorithm of characteristics

If we use the method in [38] to approximate the local nonnegative stabilizing solution $\Pi(x)$ to the HJBI equation (5), the first three approximations $\overline{V}_{1,x}, \overline{V}_{2,x}, \overline{V}_{3,x}$ of $\frac{\partial \Pi(x)}{\partial x}$ in (53) are

$$\begin{aligned}
\overline{V}_{1,x} &= x, \\
\overline{V}_{2,x} &= x + \frac{1}{2}x^3, \\
\overline{V}_{3,x} &= x + \frac{1}{2}x^3 + \frac{7}{16}x^5 + \frac{9}{80}x^7 + \frac{437}{53760}x^9.
\end{aligned}$$

We plot these approximations together in Fig. 1 (we ignore the first approximations for both algorithms since they are identical) to compare their convergence to the "exact solution," which is given by (55).

From Fig. 1, we can see that our algorithm has better accuracy than the method of characteristics in [38], noting in particular the following points:

1. For both the second approximation and the third approximation, our algorithm is more accurate than the method in [38].
2. The second approximation (dotted line) of our algorithm is very close to the third approximation (dashed line) of the method in [38].
3. The third approximation of our algorithm (thin solid line) is very close to the exact solution (thick solid line).

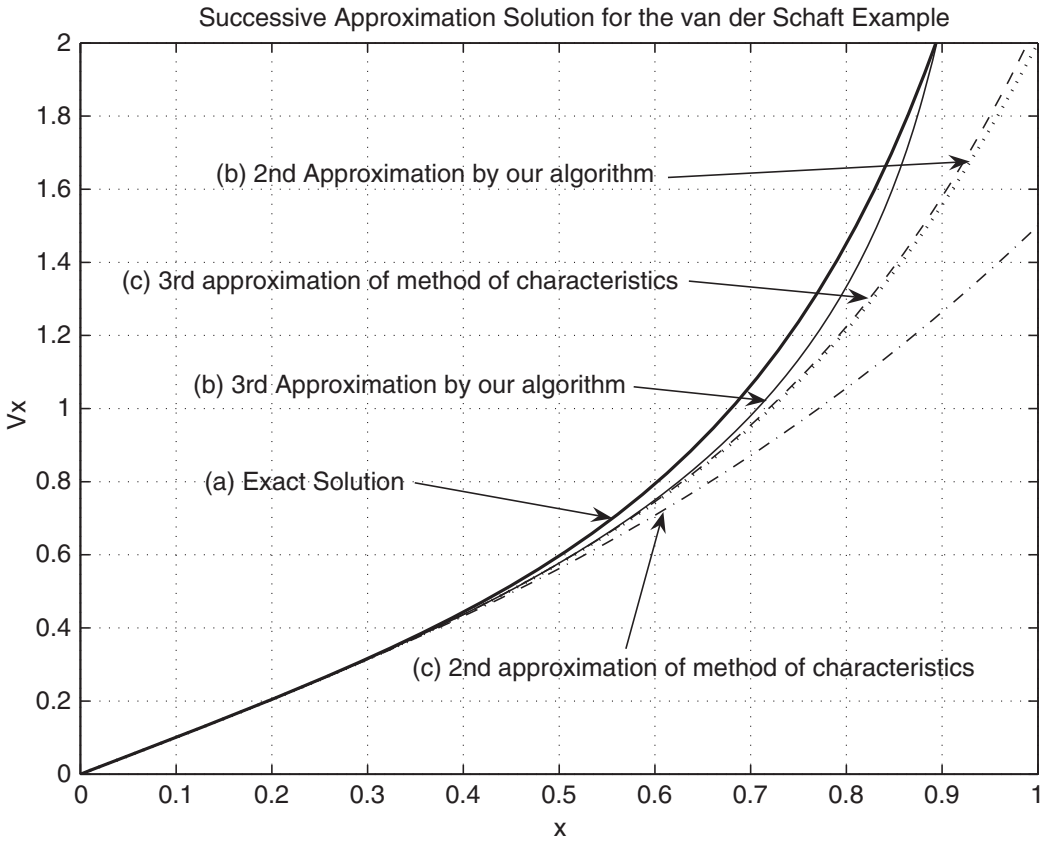


Fig. 1. Demonstration and comparison of algorithm

6 Conclusions

There have been two main thrusts in this chapter. First, motivated by the knowledge that standard Riccati equation solvers can encounter numerical problems on occasions, which can often be fixed for H_2 Riccati equations using a Kleinman algorithm, we developed a new algorithm to solve H_∞ Riccati equations. The algorithm is recursive, with each recursive step requiring solution of an H_2 Riccati equation (itself amenable to solution via the Kleinman algorithm). The algorithm has the following advantages: (1) a simple initialization; (2) local quadratic rate of convergence; (3) a natural game theoretic interpretation; (4) high numerical stability and reliability. Under some suitable assumptions, we can compute nonnegative stabilizing solutions of Riccati equations and HJBI equations recursively.

Second, motivated as much by the sparsity of solution methods for HJBI equations as by numerical problems arising from time to time in known solution procedures, we have illustrated how, for a class of HJBI equations, one can replace the problem of solving a single such equation by the problem of solving a sequence of HJB equations, each of which can be tackled using a sequence of linear partial differential equations.

The ideas presented may be valid in much broader game theoretic contexts than those considered here. One recent extension we have achieved is to the solution of H_∞ periodic Riccati differential equations (see [4]), and it is possible that our algorithm can be extended to solve H_∞ periodic HJBI equations.

Acknowledgment

This work has been supported in part by Australian Research Council Discovery Project Grant DP0664427.

References

1. Abou-Kandil, H., Freiling, G., Ionescu, V., Jank, G.: Matrix Riccati Equations in Control and Systems Theory, Birkhäuser, Basel, Switzerland (2003).
2. Anderson, B.D.O., Hitz, K.L., Diem, N.D.: Recursive algorithms for spectral factorization. IEEE Trans. Circuits Syst. CAS-21 (6), 742–750, November (1974).
3. Anderson, B.D.O.: Second order convergence algorithms for the steady-state Riccati equation. Int. J. Contr. 28(2), 295–306 (1978).
4. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users Guide: Third Edition, ser. Software Environments Tools. Philadelphia: SIAM, 1999.
5. Arnold, III W.F., Laub, A.J.: Generalized eigenproblem algorithms and software for algebraic Riccati equations, *Proceedings of the IEEE*, 72(12), 1746–1754 (1984).
6. Bartels, R.H., Stewart, W., Solution of the matrix $AX + XB = C$. Commun. ACM, 15, 820–826 (1972).
7. Beard, R.W., McLain, T.W.: Successive Galerkin approximation algorithm for nonlinear optimal and robust control. Int. J. Control, 71(5), 717–743 (1998).
8. Bellman, R.E.: Dynamic programming, Princeton, Princeton University Press, (1957).
9. Bellman, R.E., Kalaba, R.E.: Quasilinearization and Nonlinear Boundary-Value Problems, American Elsevier, New York (1965).
10. Bellman, R.E.: Introduction to the Mathematical Theory of Control Process, Academic, New York (1971).
11. Benner, P.: Computational methods for linear-quadratic optimization. Suppl. Rend. Circ. Mat. Palermo, Ser. II, 58, 21–56 (1999).
12. Benner, P., Hernández, V., Pastor, A.: On the Kleinman iteration for nonstabilizable systems. J. Math. Control Signals Syst. 16(1) 76–93 (2003).
13. Cherfi, L., Abou-Kandil, H., Bourles, H.: Iterative Method for General Algebraic Riccati Equation, *ACSE 05 conference*, 19–21 Dec. 2005, CICC, Cairo, Egypt.
14. Damm, T.: Rational Matrix Equations in Stochastic Control, volume 297 of Lecture Notes in Control and Information Sciences, Springer-Verlag, Berlin, (2004).

15. Datta, B.N., Numerical Methods for Linear Control Systems, Elsevier New York, (2004).
16. Dieci, L.: Some numerical considerations and Newton's method revisited for solving algebraic Riccati equations. IEEE Trans. Automa. Control, 36, 608–616 (1991).
17. Feng, Y., Anderson, B.D.O., Rotkowitz, M.: A game theoretic algorithm to compute local stabilizing solutions to HJBI equations in nonlinear H_∞ control. Automatica, 45(4), 881–888 April (2009).
18. Golub, G.H., Van Loan, C.F.: Matrix Computations, The John Hopkins University Press, Baltimore and London (1996).
19. Green, M., Limebeer, D.J.N.: Linear Robust Control, Prentice Hall, Englewood Cliffs, NJ (1995).
20. Guo, C.H.: A note on the minimal nonnegative solution of a nonsymmetric algebraic Riccati equation. Linear Algebra Appl. 357, 299–302 (2002).
21. Guo, C.H.: Efficient methods for solving a nonsymmetric algebraic Riccati equation arising in stochastic fluid models. J. Comput Appl Math, 192, 353–373 (2006).
22. Hitz, K.L., Anderson, B.D.O.: An iterative method of computing the limiting solution of the matrix Riccati differential equation. Proc. IEE, 119(9), 1402–1406, September (1972).
23. Kalman, R.E.: The Theory of Optimal Control and the Calculus of Variations. R. Bellman (ed.), Mathematical Optimization Techniques, University of California Press, Berkeley, 309–330 (1963), Chap. 16.
24. Kleinman, D.L.: On an iterative technique for Riccati equation computation. IEEE Trans. Automat. Control 13, 114–115 (1968).
25. Lancaster, P., Rodman, L.: The Algebraic Riccati Equation, Oxford University Press, Oxford, (1995).
26. Lanzon, A., Feng, Y., Anderson, B.D.O.: An iterative algorithm to solve Algebraic Riccati equations with an indefinite quadratic term. *Proceedings of European Control Conference 2007*, 3033–3039, Greece.
27. Lanzon, A., Feng, Y., Anderson, B.D.O., Rotkowitz, M.: Computing the positive stabilizing solution to algebraic riccati equations with an indefinite quadratic term via a recursive method. IEEE Trans. Automat. Contr. 53(10), 2280–2291, November (2008).
28. Laub, A.J.: A Schur method for solving algebraic Riccati equation. IEEE Trans. Automat. Control 24, 913–921 (1979).
29. Leake, R.J., Liu, R.: Construction of suboptimal control sequences. SIAM J. Control, 5(1) 54–63 (1967).
30. Lukes, D.L.: Optimal regulation of nonlinear dynamical systems. SIAM J. Control 7(1), 75–100 (1969).
31. Martensson, K.: On the matrix Riccati equation: Inf. Sci. 3, 17–49 (1971).
32. Mehrmann, V., Tan, E.: Defect correction methods for the solution of algebraic Riccati equations. IEEE Trans. Automat. Control 33, 695–698 (1988).
33. Mehrmann, V.: The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution, number 163 in Lecture Notes in Control and Information Sciences. Springer-Verlag, Heidelberg (July 1991).
34. Mehrmann, V.: Numerical Methods for Eigenvalue and Control Problems. Frontiers in Numerical Analysis. Springer New York, NY (2002).

35. Sima, V.: Algorithms for Linear-Quadratic Optimization, volume 200 in Pure and Applied Mathematics, Marcel Dekker, New York, (1996).
36. van der Schaft, A.J.: L_2-gain analysis of nonlinear systems and nonlinear systems and nonlinear state feedback H_∞ control. IEEE Trans. Automat. Control 37, 770–784 (1992).
37. Willems, J.C.: Least Squares Stationary Optimal Control and the Algebraic Riccati Equation. IEEE Trans. Automat. Control, AC-16(6), 621–634 (1971).
38. Wise, K.A., Sedwick, J.L.: Successive approximation solution of the HJI equation. *Proceeding IEEE Conference on Decision and Control*, 1387–1991 (1994).
39. Zhou, K., Doyle, J.C., Glover, K.: Robust and Optimal Control, Prentice Hall, Upper Saddle River, NJ (1996).