LEARNING TO CONTROL LINEAR TIME-INVARIANT SYSTEMS WITH DISCRETE TIME REINFORCEMENT LEARNING

A Thesis

by

ERIC JAMES NELSON

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirement for the degree of

MASTER OF SCIENCE

Chair of Committee,	Thomas Ioerger
Committee Members,	Dylan A. Shell
	John Valasek
Head of Department,	Dilma Da Silva

December 2016

Major Subject: Computer Science

ABSTRACT

Reinforcement learning (RL) is a powerful method for learning policies in environments with delayed feedback in a model-free way. Another powerful method for obtaining control policies is the Linear Quadratic Regulator (LQR) problem, which utilizes knowledge of a linear model to derive the optimal policy. However, the continuously evolving dynamics of linear systems pose a challenge to using RL techniques as the underlying theory is discrete in nature. Therefore, reinforcement learners must discretize the dynamics by sampling at specific time intervals. The time interval used by the reinforcement learner directly affects the quality of the control policy that is learned. This work attempts to characterize the quality of learned policies as a function of the sample time of the reinforcement learner and in comparison with the optimal control as it is derived from the LQR framework. It is shown that in the limit as the sample time is decreased to zero, the policies generated converge to the optimal policies. In addition, any non-zero sample time introduces error into the learned policies. This error increases exponentially as a function of the sample time used to train the reinforcement learner.

To my mother, father, and brother.

ACKNOWLEDGMENTS

I would like to thank Texas A&M University, the Office of Graduate and Professional Studies, the Department of Computer Science and Engineering, and the faculty and staff for supporting me throughout my time here and giving me an opportunity to expand my knowledge.

I would also like to give a special thanks to Dr. Thomas Ioerger, whose passion, knowledge, and dedication helped me grow as both a graduate student and a person. Without the dedication and support of Dr. Ioerger, this work would not have been possible.

NOMENCLATURE

RL	Reinforcement Learning
CT	Control Theory
LQR	Linear Quadratic Regulator
LTI	Linear Time Invariant
AOA	Angle of Attack
MARE	Matrix Algebraic Riccati Equation
CARE	Continuous-time Algebraic Riccati Equation
HJB	Hamilton-Jacobi-Bellman
RLS	Recursive Least-Squares
RK	Runge-Kutta

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Linear systems are ubiquitous in science and engineering. They are described by models derived from their physical properties and the equations describing their behavior. Often times, it is a goal for scientists and engineers to design controls in order to force the systems to behave in accordance to an objective function. There are two main methods for deriving such controls: control theory and reinforcement learning. The control theory method uses the model to analytically derive an optimal control. In contrast, reinforcement learning learns an optimal control through the use of a feedback signal which means a model is not necessary.

Reinforcement learning is beneficial in that it can learn a control in a model-free way. Typically, a reinforcement learning task is modeled as a *Markov Decision Process* with discrete states and a set of discrete actions from which a learner can choose. When state-transition probabilities or reward functions are unknown a MDP becomes a reinforcement learning problem [1]. A typical reinforcement learner learns by starting in an initial state, taking an action, ending up in a new state, getting a reward, updating internal parameters, and finally repeating. The goal of the reinforcement learner is to learn a policy that maximizes the long-term expected discounted reward.

Linear systems can represent a large number of physical phenomena (e.g. inverted pendula, vehicle dynamics, mixing in chemical vats). These systems are described through the physical equations governing them meaning the equations describing them are inher-

ently continuous in state, time, and control input. Linear systems are of particular interest in control theory (CT) due to the fact that there are well-studied methods for determining optimal or near optimal controls. However, unlike using reinforcement learning to control a system, these mechanisms require knowledge of the model.

Although these two subjects (RL and CT) are closely related there is a major disconnect between the two as one is inherently continuous in time and the other inherently discrete. This disconnect introduces some interesting questions that will be answered in this work.

- Can a reinforcement learner learn to control a linear system?
- How close to optimal can the policy learned be? What should the sort of rewards should the learner be given?
- How often should a reinforcement learner sample the state and decide on a new action?
- How does the sample time from the previous question affect the control policy?
- Is there a bound on the error for a given sample time?

1.1 Motivation

It is often the case that the true model of a system is not known and therefore it is useful to study the effect of using a reinforcement learner to learn to control such systems. When a digital controller is involved, there are limitations on how often the state of a system can be sampled. For example, the sample time is clearly limited by the cycle time

of the processor. An assumption of a **step-input model** is made in this work. That is, a control is selected by a reinforcement learner and it must be held for the entire sample time. Clearly, such a control is not as good as a continuously varying control input could be and therefore some cost will be incurred for any discrete sample time. This limitation is particularly critical to study for systems which rapidly change state. It is desirable to make this sample time as large as possible for a given system because cheaper hardware may be used or processing cycles can be saved. The goal of this work is to provide insight into the relationship between the selection of a sample time and quality of control.

1.2 Scope & Contribution

This work is only concerned with the control of linear systems. Non-linear systems present their own challenges and are often more difficult to analyze. However, there is no limitation placed on the order of the linear system in this work and some of the examples used are of higher order. The results only hold over the controllable subspace of the system. In other words, the goal cannot be to drive the system to an uncontrollable state.

In this work, it is shown that: a reinforcement learner can learn to control linear systems, the value function can be represented exactly through the use of a weighted set of basis functions, there exists a bound on the error produced from using a learned policy for a given sample time. The results have a real-world impact in that money or processing cycles can be saved though the intelligent selection of a sample time.

1.3 Previous Work

This work spurred from previous work done by Kenton Kirkpatrick, a previous doctoral candidate at Texas A&M University [2]. He attempts to learn an optimal sample time adaptively by modifying the reinforcement learning algorithm and reward functions [3] [2]. He shows that it is possible to learn the best sample time from a set of sample times through this modification. This work differs from his in that it attempts to derive a theoretical description of the effects of the sample time that is not limited to a subset of possible sample times.

The reinforcement learning algorithms used in Kenton's work are based on work by Bellman and his dynamic programming in which a problem is broken down into many sub-problems that must also be optimized [4]. Learning in the dynamic programming sense requires building an estimate of the *value function* which assigns a value for being in a state and following a policy. The value assigned is the expected long-term reward for being in a state s and following a policy π.

$$V^{\pi}(s) = E_{\pi}[R(s)] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^{k} r_{k+1}] = R(s) + \gamma \sum_{s'} V^{\pi}(s')$$

This representation of value requires knowing at least some model of the system in terms of the state transition function. Therefore, in order to factor out the state transition function, Q-learning was introduced by Watkins. In Q-learning, we learn a Q-function, instead of a value function, which assigns a value for being in a state s and taking an action a.

$$Q^{*}(s,a) = (1-\alpha)Q^{*}(s,a) + \alpha[r + \gamma \max_{a'} Q^{*}(s',a')]$$

This factors out the need to know the state transition function and thus we can learn in a model free way [5].

Lagoudakis and Parr show that reinforcement learning can be done in continuous state spaces by representing Q-functions as a linear combination of a set of basis functions $[\phi_i ... \phi_N]$ [6].

$$V(s) = \sum_{i}^{N} w_i \phi_i(s)$$

However, in this representation, the selection of the basis functions becomes the main issue and no general guidelines are given. This is likely due to the goal of generality in their work. They additionally show that policy iteration can be done with this representation of a value function using a least-squares weight updating technique.

Steven J. Bradtke's work focuses on the convergence of reinforcement learning algorithms for LQR problems. These problems have the property that the optimizing control is represented as a linear feedback control.

$$u^*(t) = -\mathbf{K}x(t)$$

He shows that a recursive least squares update rule can be used to learn a Q-function [7]. However, he shows that it is possible for the algorithm to converge to a non-optimal Q-function in the LQR setting. This is due to the representation of the Q-function as a linear combination over the basis set of the quadratic combination of state and control variables.

$$Q(s,a) = \begin{bmatrix} x \\ u \end{bmatrix} H_u \begin{bmatrix} x & u \end{bmatrix}$$

The convergence proof of the Q-learning algorithm proposed by Watkins relies on the table

representation of the Q-function so that the values in the cells of the tables can be incrementally updated. The question of what Q-function do we converge to can be answered by defining a fixed point to which the algorithm will converge.

$$\begin{bmatrix} x & u \end{bmatrix}^T H_u \begin{bmatrix} x \\ u \end{bmatrix} = x'Qx + u'Ru + \gamma \begin{bmatrix} Ax+Bu & a \end{bmatrix} H_u \begin{bmatrix} Ax+Bu \\ a \end{bmatrix}$$

This equation specifies (n + m)(n + m + 1)/2 equations with an equal number of unknowns. There are multiple solutions possible to this system of equations and therefore it is possible to converge to a non-optimal Q-function. However, Bradtke shows that the policy iteration algorithm he proposes that updates the weights on the set of quadratic basis and control variables by a Recursive Least Squares (RLS) update rule will converge to the optimal Q-function.

Brogan's book Modern Control Theory describes solving the LQR problem in both continuous and discrete time [8]. In it, he also discusses the application of Pontryagin's minimum principle and the minimization of the Hamiltonian as a necessary condition for optimality. These two ideas are used to derive the bounds on the error in this work.

Johnson et al. attempt to solve an N-player nonzero-sum game that is subject to continuous-time nonlinear dynamics [9]. Their solution is based on solving the Hamilton-Jacobi-Bellman equation that defines an optimality condition. Similar reasoning is used in this work in solving for optimal controls.

1.4 Background

This section serves to introduce the topics used in the theoretical section to derive results. It serves as a foundation for the other sections.

1.4.1 Linear Systems

State-Space Representation

Modern control theory introduces the idea of the state of a system. In this representation, a set of state variables are chosen and the system is described by how these variables evolve over time. Any linear time-invariant (LTI) system with n state variables, r input variables, and m output variable can be written in terms of change in state by a set of simultaneous differential equations,

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

where x is the state, y is the output and u is the control input. $\mathbf{A}$ is an $n \times n$ matrix that describes the system dynamics. $\mathbf{B}$ is an $n \times r$ matrix that describes how the inputs affect the change in state. $\mathbf{C}$ is an $m \times m$ matrix and $\mathbf{D}$ is an $m \times r$ matrix. We can solve the differential equation for x by employing the Laplace transform.

$$\mathbf{X}(s) = (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s) \tag{1.1}$$

then taking the inverse Laplace transform,

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)\, d\tau \tag{1.2}$$

Therefore, we can see that the internal behavior (state space trajectory) is not, in general, linear and instead the system evolves exponentially as a function of time. These trajectories have a transient phase of rapid response before slowing to a new equilibrium point which we call the long-term or asymptotic response. If we assume that **A** is negative definite then as a control is applied, the exponential terms will approach zero as time passes. Eventually, the system will converge to a new equilibrium point as the exponential terms approach zero.

Example

Consider the following linear system,

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} -1 & 2 & 0 \\ -1 & -4 & 1 \\ 0 & 0 & -1 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \mathbf{u}(t)$$

The solution can be derived by using the Laplace transform. So we focus on the $(sI - A)^{-1}$ term first

$$(s\mathbf{I} - \mathbf{A}) = \begin{bmatrix} s+1 & -2 & 0 \\ 1 & s+4 & -1 \\ 0 & 0 & s+1 \end{bmatrix}$$

$$(s\mathbf{I} - \mathbf{A})^{-1} = \begin{bmatrix} \frac{2}{s+2} - \frac{1}{s+3} & \frac{2}{s+2} - \frac{2}{s+3} & -\frac{2}{s+2} + \frac{1}{s+3} + \frac{1}{s+1} \\ \frac{1}{s+3} - \frac{1}{s+2} & \frac{2}{s+3} - \frac{1}{s+2} & \frac{1}{s+2} - \frac{1}{s+3} \\ 0 & 0 & \frac{1}{s+1} \end{bmatrix}$$

To solve for $x(t)$ with

$$\mathbf{x}(0) = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \qquad \text{and} \qquad \mathbf{u}(t) = \begin{bmatrix} U_1(t) \\ U_2(t) \end{bmatrix}$$

where $\mathbf{u}(t)$ is a step function. We now have the parts to evaluate Equation 1.1. Multiplying the matrices out we can start to formulate the full solution in Laplace space,

$$(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{x}(0) = \begin{bmatrix} \frac{1}{s+1} \\ 0 \\ \frac{1}{s+1} \end{bmatrix}$$

and

$$(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s) = \begin{bmatrix} \frac{1}{s+1} & \frac{2}{s+2} - \frac{2}{s+3} \\ 0 & \frac{2}{s+3} - \frac{1}{s+2} \\ \frac{1}{s+1} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{s} \\ \frac{1}{s} \end{bmatrix} = \begin{bmatrix} -\frac{1}{s+1} - \frac{1}{s+2} + \frac{2}{3(s+3)} + \frac{4}{3s} \\ \frac{1}{2(s+2)} - \frac{2}{3(s+3)} + \frac{1}{6s} \\ \frac{1}{s} - \frac{1}{s+1} \end{bmatrix}$$

Therefore,

$$\mathbf{X}(s) = \begin{bmatrix} \frac{1}{s+1} \\ 0 \\ \frac{1}{s+1} \end{bmatrix} + \begin{bmatrix} -\frac{1}{s+1} - \frac{1}{s+2} + \frac{2}{3(s+3)} + \frac{4}{3s} \\ \frac{1}{2(s+2)} - \frac{2}{3(s+3)} + \frac{1}{6s} \\ \frac{1}{s} - \frac{1}{s+1} \end{bmatrix}$$

so that

$$x(t) = \begin{bmatrix} e^{-t} - e^{-t} - e^{-2t} + \frac{2}{3}e^{-3t} + \frac{4}{3} \\ \frac{1}{2}e^{-2t} - \frac{2}{3}e^{-3t} + \frac{1}{6} \\ e^{-t} + 1 - e^{-t} \end{bmatrix} = \begin{bmatrix} -e^{-2t} + \frac{2}{3}e^{-3t} + \frac{4}{3} \\ \frac{1}{2}e^{-2t} - \frac{2}{3}e^{-3t} + \frac{1}{6} \\ 1 \end{bmatrix} \tag{1.3}$$

Solving for the Optimal Continuous Time Trajectory

Next, we need to find a policy **u** that optimizes some objective. The calculus of variations will prove to be useful in finding such a policy. Mathematically, we can describe the objective function as an integral of a cost function,

$$J = \int_a^b f(x, y, y')\, dx$$

where we want to find the function y that minimizes J over the range $[a, b]$ and satisfies initial and final boundary conditions. In this paper, we consider that the goal is to reach a target state x_d, and $f(\cdot)$ is the distance squared as a cost function, $f = \mid x_t - x_d \mid^2$ or, for generality, $f = \mathbf{x^T Q x}$ in matrix form, where **Q** is a matrix of weights. Using Calculus of Variations, a necessary condition for an extremum function is that it satisfies the Euler-Lagrange equation.

$$\frac{\partial f}{\partial y} - \frac{d}{dx}\left(\frac{\partial f}{\partial y'}\right) = 0$$

The optimal control for a given objective function can be found in a similar manner. This will correspond to how we find an optimal policy in the case of continuous state spaces. For example, take the objective function [8]

$$J = \int_0^{t_f} \mathbf{x}(t)^T \mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)\mathbf{R}\mathbf{u}(t)\, dt$$

First we define the pre-Hamiltonian

$$\mathcal{H}(x,u,p,t) = L(x,u,t) + p^T(t) f(x,u,t)$$

$$\dot{p} = \frac{\partial \mathcal{H}}{\partial \mathbf{x}}$$

$$\mathcal{H} = \mathbf{x}^T \mathbf{Q}\mathbf{x} + \mathbf{u}^T \mathbf{R}\mathbf{u} + \mathbf{p}^T (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u})$$

Pontryagin's minimum principle specifies that the optimal policy is one that minimizes the Hamiltonian at every time t [8]. The Hamiltonian for this example is minimized by setting

$$\frac{\partial \mathcal{H}}{\partial \mathbf{u}} = 0 = 2\mathbf{R}\mathbf{u} + \mathbf{p}^T \mathbf{B}$$

$$\mathbf{u}^*(t) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{B}^T \mathbf{p}(t)$$

where $\mathbf{p}(t)$ is the co-state vector that can be solved for through the use of boundary conditions. This gives us the optimal continuous time policy $\mathbf{u}^*(t)$ that minimizes J. Furthermore, the optimal continuous-time trajectory can be derived as:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}^*(t)$$

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) - \mathbf{B}\mathbf{K}\mathbf{x}(t)$$

$$\mathbf{x}(t) = e^{(\mathbf{A}-\mathbf{B}\mathbf{K})t} x(0)$$

where $\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}$

1.4.2 Solving for the Optimal Continuous Time Value Function

One can solve for the optimal value function by solving the Hamilton-Jacobi-Bellman (HJB) equation [9]. If we define the value function as

$$V = \int_0^\infty \mathbf{x}(t)^T \mathbf{Q}\mathbf{x}(t) + \mathbf{u}(t)^T \mathbf{R}\mathbf{u}(t)\, dt$$

where $\mathbf{Q}$ and $\mathbf{R}$ are positive definite matrices. Then we can define a differential equivalent [10]

$$0 = r(x,u) + \nabla V^T \dot{x}$$

The RHS of this equation is the Hamiltonian of the system.

$$\mathcal{H}(x,u,\nabla V) = r(x,u) + \nabla V^T \dot{x}$$

and we arrive at

$$0 = \min_u \mathcal{H}(x,u,\nabla V)$$

which is the HJB equation.

If we assume that the value function is a quadratic function in the Euclidean distance to the origin then we can derive an analytical solution for the value function based on solving the Matrix Algebraic Riccati Equation (MARE) equation $\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} + \mathbf{Q} + \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} = 0$ [10].

$$V(x) = \mathbf{x}^T\mathbf{P}\mathbf{x}$$

$$\mathbf{P} = \int_0^\infty e^{t\mathbf{A}^T}\mathbf{Q}e^{t\mathbf{A}}\,dt$$

where $\mathbf{P}$ is the solution to the MARE equation. Note that the integral in the definition of $\mathbf{P}$ doesn't need to be solved directly. Instead, $\mathbf{P}$ is the solution of the $n(n+1)/2$ simultaneous equations defined by the MARE equation. Also, note that these results only hold if the origin is the goal. Therefore, if the goal is to control the system to a target equilibrium point, x_d, that is not at the origin, then a change of variables where the new origin is defined

as $w = x - x_d$ can be carried out using the chain rule

$$\frac{d\mathbf{w}}{dt} = \dot{\mathbf{w}} = \frac{d\mathbf{w}}{d\mathbf{x}}\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} + \mathbf{B}(\mathbf{U} + \Delta\mathbf{u})$$

$$\dot{\mathbf{w}} = \mathbf{A}\mathbf{w} + \mathbf{A}\mathbf{x}_d + \mathbf{B}\mathbf{U} + \mathbf{B}\Delta\mathbf{u} = 0$$

$$\dot{\mathbf{w}} = \mathbf{A}\mathbf{w} + \mathbf{B}\Delta\mathbf{u}$$

where $\mathbf{U} = -\mathbf{B}^{-1}\mathbf{A}\mathbf{x}_d$.

1.4.3 LQR Problem

The LQR is a special class of control problems that are well studied. The assumptions for this problem in this work are as follows:

1. Continuous LTI dynamics $\dot{x} = Ax + Bu$
2. Infinite horizon (minimize objective as $t \to \infty$)
3. Deterministic (i.e. No probabilistic transitions or noise)
4. The goal must be to regulate to the origin ($x_d = 0$)

   If this is not the case, then we can change variables (see Section 1.4.2).
5. The system is controllable. A system is controllable if the controllability matrix $\mathbf{R}$ has full row rank.

$$\mathbf{R} = [B \;\; AB \;\; A^2B \;\; \cdots \;\; A^{n-1}B]$$

Given those assumptions, we define an objective function that we would like to minimize,

$$J = \int_0^\infty x(t)^T \mathbf{Q} x(t) + u(t)^T \mathbf{R} u(t)\, dt$$

where $\mathbf{Q}$ and $\mathbf{R}$ are positive definite weighting matrices. As derived using the HJB equa-

tion, the objective function is minimized by selecting the control

$$u(t) = -\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}x(t)$$

meaning that the minimal cost to get to the goal from $x(0)$ is

$$V = x(0)\mathbf{P}x(0)$$

where $\mathbf{P}$ is the solution to the Continous-time Algebraic Riccati Equation (CARE).

1.4.4 Reinforcement Learning

Reinforcement learning (RL) is a method of machine learning in which an agent learns a control policy by attempting to maximize the returned rewards for taking an action in a given state [11]. It does this by exploring the state and action spaces and observing the rewards in order to approximate the *value function* which is the expectation of the sum of discounted rewards for starting in state s and following the policy π. An important assumption made by the RL framework is that state changes are discrete; when an action is taken, a transition to a new state occurs, and the reward is observed, instead of happening continuously in time. If the value function is known, the optimal policy π^* can be derived as:

$$\pi^*(s) = \arg\max_{s'} V(s')$$

Markov Decision Processes

Much of the theory of reinforcement learning is based on what are called *Markov Decision Processes* (MDPs) [1]. A MDP is defined on a set of states S and actions A available to an agent. The size of the state and action spaces are traditionally assumed to

be finite. The reward function $R(s,a)$ defines the feedback given to the agent for being in state s, taking action a, and ending in state s'. Actions can either be deterministic or non-deterministic. If the current state depends only on the previous state and action taken then the problem is said to satisfy the *Markov property*. Given this definition of an MDP, we now define the value function for a given policy (mapping of states to actions) π as

$$V^{\pi}(s) = E_{\pi}[R(s)] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_{k+1}]$$

Where γ is a *discount factor* that tunes how much the agent should care about future rewards versus the current reward. In other words, the value of being in state s is equal to the expected value of sum of the current reward and all future rewards for being in state s and following π until termination.

If the reward and transition functions are known, an MDP can be solved by employing dynamic programming techniques [4]. Value iteration is one such technique in which one can iterate on the value function by taking the max over the actions $V(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$. The iterations continue until the LHS of the equation is equal to the RHS. The equation above is called the *Bellman equation*, which is based on the Bellman Principle of Optimality.

Q-learning

Some typical methods of reinforcement learning, such as the value iteration algorithm, require the learner to know some model of the underlying system (i.e. the reward function or state-transition function). However, it is often the case that one does not know these functions. Q-learning can be used to learn policies without having to know a-priori

the reward or state-transition functions. Instead of a value function, we now define a Q-function that represents the value of being in state s and taking action a

$$Q^{\pi}(s,a) = E_{\pi}[R(s,a)] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^{k} r_{k+1}]$$

The Q-function is typically represented as a table called a Q-table where the rows and columns are the finite states and actions. Q-learning can be extended to infinite state spaces by representing Q-functions as the sum of weighted basis functions

$$\hat{Q}(s,a) = \sum_{i=1}^{N} w_i \phi_i(s,a)$$

This representation requires choosing a set of basis functions, $\phi_i(s,a)$, that can accurately represent the true value function for a given problem.

Algorithm 1 Q-learning algorithm.

```
1:  function Qlearn(Q)
2:      Q = Q_0 = initial guess
3:      for i ← 1 to ∞ do
4:          s = new random state
5:          while s is not terminal do
6:              Choose a (randomly, ε-greedy, etc.)
7:              Observe r, s'
8:              Q(s,a) = (1 − α)Q(s,a) + α[r + γ(max_a Q(s',a))]
9:          end while
10:     end for
11: end function
```

Policy Iteration

Given an initial policy π_0 one can find an optimal policy by performing *policy iteration*. In policy iteration, the value function for the current policy is solved for or estimated. Next, the policy can be improved by checking if at every state we can change an action

that improves the overall value. This leads to a new policy that can be used in the value function estimation step. These iterations can continue to be done with the guarantee that the policy is improving at each step. See Algorithm 2.

Algorithm 2 Policy Iteration algorithm.

```
1: function PolicyIteration(N, M, π_0)
2:     π = π_0
3:     for i ← 1 to N do
4:         V^π = PolicyEval(π, M)
5:         π = PolicyUpdate(V^π)
6:     end for
7:     return π*
8: end function
```

Adaptive Policy Iteration

The policy iteration algorithm used in our experiments was proposed by Steven Bradkte in his work on using reinforcement learners on LQR problems [7]. In this algorithm, one updates the estimate of the true Q-function by adjusting weights through a Recursive Least-Squares (RLS) update rule. Given the value function for a policy U is represented as

$$V_U(x_t) = x_t' K_U x_t$$

and the single-step reward function is represented as

$$r_t = x'Qx + u'Ru$$

Updating θ_k can be accomplished through the following two equations

$$\theta_k(i) = \theta_k(i-1) + \frac{P_k(i-1)\phi_t(r_t - \phi_t'\theta_k(i-1))}{1 + \phi_t' P_k(i-1)\phi_t}$$

Algorithm 3 Adaptive Policy Iteration (API) algorithm.

```
1:  function API(N)
2:      θ = θ_0 = initial guess
3:      for k ← 1 to ∞ do
4:          P_k = P_0
5:          for i ← 1 to N do
6:              u_i = U_k x_i + e_i
7:              x_i = rk(x_i, u_i)
8:              Update θ_k using RLS
9:          end for
10:         Update U_k based on θ_k
11:         θ_{k+t} = θ_k
12:     end for
13: end function
```

$$P_k(i) = P_k(i-1) - \frac{P_k(i-1)\phi_t\phi_t' P_k(i-1)}{1+\phi_t' P_k(i-1)\phi_t}$$

In addition, updating U_k can be accomplished through the following equation

$$U_{k+1} = -H^{-1}_{U_k(22)} H_{U_k(21)}$$

where

$$H_k = \begin{bmatrix} Q+\gamma A' K_u A & \gamma A' K_u B \\ \gamma B' K_u A & R+\gamma B' K_u B \end{bmatrix}$$

An exploratory signal e_t that is persistently exciting is necessary in order to ensure convergence of the value function. In this work, like Bradkte suggests, a random signal from a normal distribution is used as it performed well in experiments. Although, such an exploratory signal doesn't satisfy the persistent excitation condition.

2 ANALYSIS & THEORY*

2.1 LTI System Behavior

This section will discuss how the behavior of a linear system can be decomposed into a transient and long term portions which represent how the system behaves in the short term after an action is taken and where the system will converge to.

2.1.1 Decomposition

In this section, we prove that a given linear system can be decomposed into a linear function of the transient and long term behaviors of the system.

Theorem 2.1 *If* $\mathbf{A}$ *has both* real *and* distinct *eigenvalues, then each entry in the solution to the state equation (Eq. 1.2) can be written as the sum of exponentials and constants.*

Recall the solution to the state space equations in the Laplace domain given by Equation 1.1. Note that $x(0)$ and $\mathbf{B}$ are both constant and $\mathbf{U}(s)$ is a step input. Therefore, let us focus on the $(s\mathbf{I}-\mathbf{A})^{-1}$ term first. Using $|s\mathbf{I}-\mathbf{A}|=0$, we can obtain the eigenvalues of $\mathbf{A}$, which we assume by the theorem are both real and distinct. We can write

$$|s\mathbf{I}-\mathbf{A}| = (s-\lambda_1)(s-\lambda_2)...(s-\lambda_n) \tag{2.1}$$

*Portions of this section are reprinted with permission from "Learning to Control First Order Linear Systems with Discrete Time Reinforcement Learning" by Nelson, E. Ioerger, T.R, 2016. Proceedings of the 2016 International Conference on Embedded Systems, Cyber-physical Systems, & Applications, pp. 155-161, Copyright 2016 by CSREA Press.

which is a polynomial of degree n. Let M_{ij} be the matrix obtained by eliminating the i^{th} row and j^{th} column from $\mathbf{A}$. $|M_{ij}|$ is called the minor of a_{ij} and has a degree $\leq (n-1)$. Let

$$A_{ij} = (-1)^{i+j}|M_{ij}| \tag{2.2}$$

be called the cofactor of a_{ij}. The adjoint of $\mathbf{A}$ is the matrix in which

$$adj(A)_{ij} = A_{ji} \tag{2.3}$$

In other words, the adjoint of $\mathbf{A}$ is the transpose of the matrix of cofactors. Since

$$(s\mathbf{I}-\mathbf{A})^{-1} = \frac{adj(s\mathbf{I}-\mathbf{A})}{|s\mathbf{I}-\mathbf{A}|}$$

We can use Eqs 2.1, 2.2, and 2.3 to write

$$\begin{aligned}
&\frac{adj(s\mathbf{I}-\mathbf{A})_{ij}}{(s-\lambda_1)(s-\lambda_2)...(s-\lambda_n)} \\
&= \frac{(s\mathbf{I}-\mathbf{A})_{ji}}{(s-\lambda_1)(s-\lambda_2)...(s-\lambda_n)} \\
&= \frac{(-1)^{j+i}|M_{ji}|}{(s-\lambda_1)(s-\lambda_2)...(s-\lambda_n)}
\end{aligned}$$

Because the degree of $|M_{ji}|$ is at most $(n-1)$ the degree of the numerator is strictly less than the degree of the denominator.

Therefore, we can use partial fraction decomposition since there are no repeated roots in the denominator and the degree of the numerator is strictly less than the degree of the denominator to get the form

$$\frac{k_1}{(s-\lambda_1)} + \frac{k_2}{(s-\lambda_2)} + ... + \frac{k_n}{(s-\lambda_n)}$$

Finally, taking the inverse Laplace transform we get:

$$k_1 e^{\lambda_1 t} + k_2 e^{\lambda_2 t} + ... + k_n e^{\lambda_n t}$$

for each entry in $(s\mathbf{I}-\mathbf{A})^{-1}$ as shown below.

$$(s\mathbf{I}-\mathbf{A})^{-1} = \begin{bmatrix} k_{1_{11}}e^{\lambda_{11}t} + \ldots + k_{n_{1_n}}e^{\lambda_{1_n}t} \\ k_{1_{21}}e^{\lambda_{21}t} + \ldots + k_{n_{2_n}}e^{\lambda_{1_n}t} \\ \vdots \\ k_{1_{n1}}e^{\lambda_{n1}t} + \ldots + k_{n_{nn}}e^{\lambda_{1_n}t} \end{bmatrix}$$

Since $\mathbf{x}(0)$ is a vector of constants for the first half of equation 1.1 we get

$$= \begin{bmatrix} c_1k_{1_{11}}e^{\lambda_{11}t} + \ldots + c_1k_{n_{1_n}}e^{\lambda_{1_n}t} \\ c_2k_{1_{11}}e^{\lambda_{11}t} + \ldots + c_2k_{n_{1_n}}e^{\lambda_{1_n}t} \\ \vdots \\ c_nk_{1_{11}}e^{\lambda_{11}t} + \ldots + c_nk_{n_{1_n}}e^{\lambda_{1_n}t} \end{bmatrix}$$

For the second half of equation 1.1, we have two parts

$$(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B} = \begin{bmatrix} b_1k_{1_{11}}e^{\lambda_{11}t} + \ldots + b_1k_{n_{1_n}}e^{\lambda_{1_n}t} \\ b_2k_{1_{11}}e^{\lambda_{11}t} + \ldots + b_2k_{n_{1_n}}e^{\lambda_{1_n}t} \\ \vdots \\ b_nk_{1_{11}}e^{\lambda_{11}t} + \ldots + b_nk_{n_{1_n}}e^{\lambda_{1_n}t} \end{bmatrix}$$

For the unit step in each dimension of $\mathbf{U}(s)$

$$\mathbf{U}(s) = \frac{1}{s}\mathbf{K^T} = \begin{bmatrix} \frac{k_1}{s} & \frac{k_2}{s} & \ldots & \frac{k_n}{s} \end{bmatrix}^T$$

we take the inverse Laplace transform to get

$$\mathbf{U}(t) = \mathbf{U} = \mathbf{K^T}H(t)$$

where $H(t)$ is the Heaviside step function. So we can see that $(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{U}(s)$ is equal to

the matrix for $(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{BU}$. Therefore,

$$\mathbf{x}(t) = \mathcal{L}^{-1}((s\mathbf{I}-\mathbf{A})^{-1}\mathbf{x}(0) + (s\mathbf{I}-\mathbf{A})^{-1}\mathbf{BU}(s))$$

$$= \mathcal{L}^{-1}(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{x}(0) + \mathcal{L}^{-1}(s\mathbf{I}-\mathbf{A})^{-1}\mathbf{BU}(s)) =$$

$$\begin{bmatrix} C_1 k_{1_{11}} e^{\lambda_{11}t} + \ldots + C_1 k_{n_{1_n}} e^{\lambda_{1n}t} \\ C_2 k_{1_{21}} e^{\lambda_{21}t} + \ldots + C_2 k_{n_{2_n}} e^{\lambda_{2n}t} \\ \vdots \\ C_n k_{1_{n1}} e^{\lambda_{n1}t} + \ldots + C_n k_{n_{nn}} e^{\lambda_{nn}t} \end{bmatrix}$$

where $C_i = k_i(b_i + c_i)$ and i is equal to the row index. Therefore, as long as the eigenvalues of the matrix $\mathbf{A}$ are both *real* and *distinct* the solution to the state equation can be written as a sum of exponential functions and constants.

If we group the exponential terms together, they form what we call the transient response matrix $\mathcal{T}$ of the system. The constants form the long term response matrix $\mathcal{N}$ of the system. The λ terms in the transient response matrix are the time constants of the system. We can now define

$$x(t) = \mathcal{T}(t) + \mathcal{N}$$

For the example problem in section 1.4.1, separating exponentials from constants in equation 1.3, we have

$$\mathcal{T} = \begin{bmatrix} -e^{-2t} + \frac{2}{3}e^{-3t} \\ \frac{1}{2}e^{-2t} - \frac{2}{3}e^{-3t} \\ 0 \end{bmatrix}, \quad \mathcal{N} = \begin{bmatrix} \frac{4}{3} \\ \frac{1}{6} \\ 1 \end{bmatrix}$$

The smallest time constant is given by $\tau = \frac{1}{3}$. Thus, the system reaches about 37% of the way to the new equilibrium point $\begin{bmatrix} \frac{4}{3} & \frac{1}{6} & 1 \end{bmatrix}^T$ in about 1/3 sec.

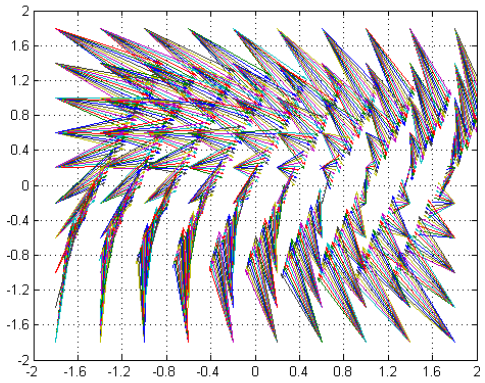


Figure 2.1: A depiction of the linear mapping ($\Delta x = \mathbf{M}(\Delta t)\mathbf{x}(0) + \mathbf{L}$) for an input action for the example problem. Each vector shows the result of taking the same action from different coordinates in state space and holding it for $\Delta t = 0.2$. [12]

Assuming the step-function input model, for a given Δt the response is linear

$$\Delta x = M(\Delta t)x(0) + L$$

as shown in Figure 2.1.

2.2 Reinforcement Learning for the LQR Problem

In reinforcement learning, it is important that the states exhibit the Markov property as defined in Section 1.4.4 as the underlying theory relies on this assumption. We claim that the Markov property holds in the context of the LQR problem.

Theorem 2.2 *Continuous LTI systems that are sampled at a given rate (Δt) exhibit the Markov property.*

In order to prove this, we need to show that the next state is dependent only on the current state and action. A discretized version of the system can be defined with the introduction of a Δt

$$X_{k+1} = A_d x_k + B_d u_k$$

where

$$A_d = e^{A\Delta t} \quad B_d = \Big[\int_0^{\Delta t} e^{A\tau}\, d\tau\Big] B$$

Using these definitions it can be clearly seen that the next state is only dependent on the current state (x_k) and the current action (u_k). Therefore, the LQR systems in this work exhibit the Markov property.

Assuming the objective function that we want to minimize is defined as

$$V = \int_0^{\infty} x(t)^T \mathbf{Q} x(t) + u(t)^T \mathbf{R} u(t)\, dt$$

Then, we know from the LQR theory, which is based on Lyapunov theory, that the value function can be represented as [13] [7] [12]

$$V^*(x) = x^T \mathbf{P} x$$

The proof is as follows.

The HJB equation is defined as

$$0 = \min_u [x' \mathbf{Q} x + u' \mathbf{R} u + \nabla V^T (\mathbf{A} x + \mathbf{B} u)]$$

trying $V = x' \mathbf{P} x$

$$0 = \min_u [x' \mathbf{Q} x + u' \mathbf{R} u + \mathbf{P} x \mathbf{A} x + \mathbf{P} x \mathbf{B} u]$$

now we take the partial over u in order to minimize

$$0 = 2\mathbf{R}u + \mathbf{B}^T\mathbf{P}x$$

$$u(t) = -\frac{1}{2}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}x(t)$$

Therefore, a solution to the HJB equation and therefore an optimal control is $u = -\mathbf{K}x(t)$ which means that the value function is represented as:

$$V = x'\mathbf{P}x$$

In addition, this work uses a weighted set of basis functions to represent the Q-function that is being learned. It has been shown that the Q-function for the LQR problem can be directly represented as a linear function over the quadratic set of state and control variables of the system [7]. That is,

$$Q(s,a) = \begin{bmatrix} x \\ u \end{bmatrix} H_u \begin{bmatrix} x & u \end{bmatrix}$$

How do we update the weights on the set of quadratic variables? By defining the reward for each episode as

$$r_t = \underbrace{x(t)^T\mathbf{Q}x(t)}_{\textbf{Penalize distance to goal}} + \underbrace{u(t)^T\mathbf{R}u(t)}_{\textbf{Penalize magnitude of control}} = (x - x_d)^2 + u^2 \tag{2.4}$$

Assuming, for simplicity, that $Q = R = I$. Note that although this might seem like the learner is being given copious amounts of information, it is just a number as given in many other reinforcement learning tasks. In addition, the reward is at a single point in time which is distinct from the value function which represents following a policy from a given state and accumulating the rewards along the whole path.

In his work, Bradtke proposed using a least-squares update rule to approximate Q-

functions. Using the approximate Q-functions, policy iteration can be done. The benefit of following such an algorithm is that the state and action spaces can be continuous and no discretization is necessary [7].

2.2.1 Change of Variables

The value function as defined in Section 2.2 is based on the origin being the desired goal of the system. That is, Q must be quadratic around the origin. If this is not the case, then it is necessary to perform a change of variables in order to shift the origin to the desired position in state-space. Figure 2.2 shows the shifted optimal trajectory caused by changing variables.

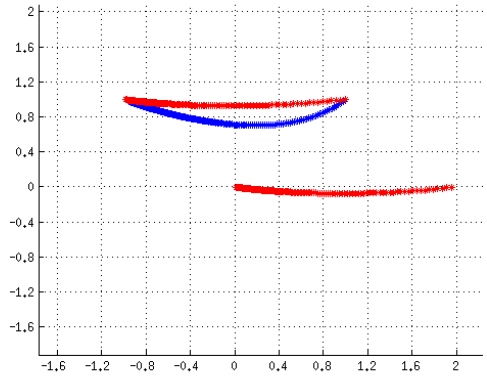


Figure 2.2: The optimal trajectory (red) and the discrete trajectory (blue), the optimal trajectory is also plotted in x-space. The bottom right trajectory shows the shifted variable evolving in w-space.

Therefore, if the goal is to control the system to a target equilibrium point, x_d, that is not at the origin, then a change of variables where the new origin is defined as $w = x - x_d$ can be carried out using the chain rule, as was shown in Section 1.4.1,

$$\frac{d\mathbf{w}}{dt} = \dot{\mathbf{w}} = \frac{d\mathbf{w}}{d\mathbf{x}}\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} + \mathbf{B}(\mathbf{U} + \Delta\mathbf{u})$$

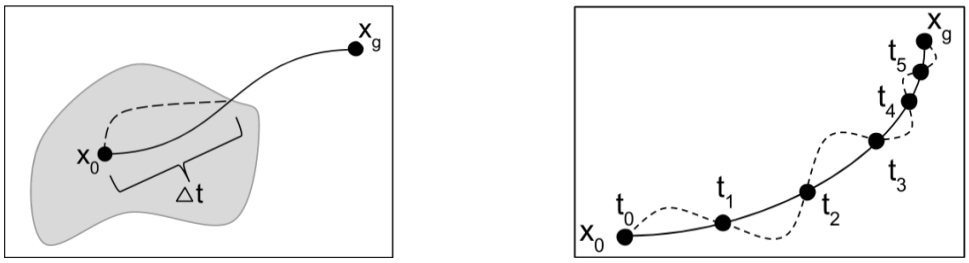
$$\dot{\mathbf{w}} = \mathbf{A}\mathbf{w} + \mathbf{A}\mathbf{x}_d + \mathbf{B}\mathbf{U} + \mathbf{B}\Delta\mathbf{u} = 0$$

$$\dot{\mathbf{w}} = \mathbf{A}\mathbf{w} + \mathbf{B}\Delta\mathbf{u}$$

where $\mathbf{U} = -\mathbf{B}^{-1}\mathbf{A}\mathbf{x}_d$. In words, this means that in order to get a new equilibrium point we must input a constant control that shifts the system to that new equilibrium. It should also be noted that the system must be controllable in order to ensure that such a control exists.

2.3 Effect of Sample Time

If we assume continuous control inputs that are bounded by some constant μ (i.e. $|U| \leq \mu$) then a *reachable* region from the current state is defined given a Δt. This is illustrated in Figure 2.3 as the light gray region. A state x_2 is said to be *reachable* from x_1 if there exists a control input $u(t)$ such that $|u(t)| < \mu$ that can transfer the system from x_1 to x_2 in finite amount of time. It should be noted that the shape of the region is dependent on the dynamics of the system and is not necessarily circular. The optimal trajectory, $x^*(t)$, is represented as the solid black line in the figure and a constant input trajectory for a given action is shown as a dotted line. The dotted line represents choosing an action and holding it constant for Δt. It is curved because of the non-linear, transient response of the system. There is always a constant action that can move the system to the same point on the continuous time trajectory after Δt.



(a) (b)

Figure 2.3: Differences between the optimal continuous-time trajectory and the trajectory produced by a sequence of discrete inputs. Solid line shows continuous-time trajectory. The shaded region shows the reachable space within Δt. The dashed line shows trajectory by a sequence of constant inputs. [12]

Theorem 2.3 *The constant control chosen by the optimally, converged learner will be one which minimizes the Hamiltonian, and this control will be a constant step input which causes the state trajectory to intersect with the optimal continuous-time trajectory after* Δt.

This can be proven using the ideas from Pontryagin's Minimum Principle and Bellman's optimality equation. Pontryagin's Minimum Principle states that the optimal policy is the one which minimizes the control Hamiltonian at every time t along the optimal trajectory.

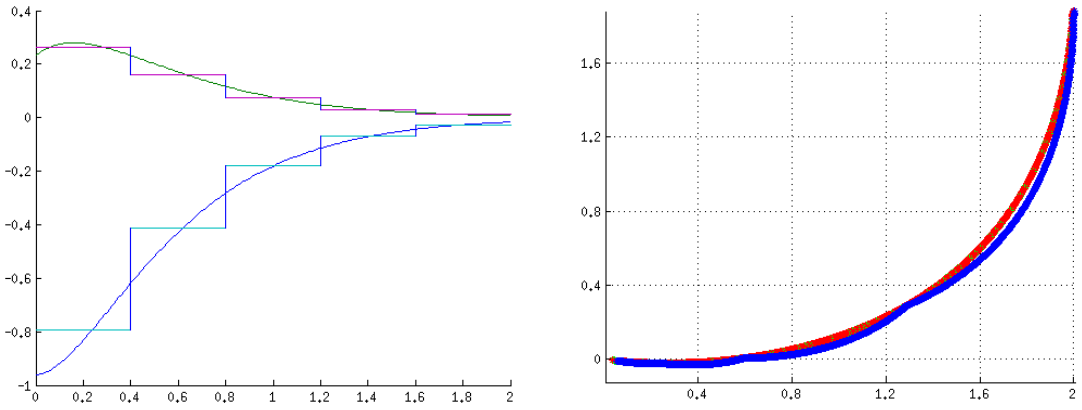
$$\mathcal{H}(x^*(t), u^*(t), \lambda^*(t), t) \leq \mathcal{H}(x^*(t), u(t), \lambda^*(t), t), \quad \forall u \in \mathcal{U}$$

and Bellman's optimality equation describes the optimal value function,

$$V^*(s) = \min_a R(s,a) + \gamma V^*(\mathcal{T}(s,a))$$

The optimal value function can be derived by following the optimal trajectory as described by Pontryagin's Minimum Principle as following the optimal control policy in continuous time leads to the optimal value at every time t. The inner portion of Bellman's optimality

equation $R(s,a)+\gamma V^*(\mathcal{T}(s,a))$ is minimized by choosing the action a such that it intersects with the optimal trajectory after Δt. Therefore, the optimal action for the reinforcement learner to choose would be the one which causes the state to intersect with the optimal trajectory after Δt. This proof assumes that the under the learned value function is equal to the optimal value function $V^{\Delta t} = V^*$ while it is not strictly guaranteed that this is true, we show that as $\Delta t \to 0$ the learned value function approximates the optimal $V^{\Delta t} \approx V^*$. Therefore, this assumption holds for reasonable selections of Δt.



(a) (b)

Figure 2.4: (a) Shows the discretization ($\Delta t = 0.4$) between continuous and optimal control inputs. The plot shows two different control inputs to the system over time. (b) Shows the path difference between following the discrete and continuous policies.

Next, we show that in the limit as $\Delta t \to 0$, we get better control policies. For a smaller Δt, the reachable region is smaller which implies the maximum distance away from the optimal trajectory is also smaller. The error of the policy for a given Δt is defined by the difference of the integral of the distance to the goal for the trajectories produced by both policies. An example approximation of an optimal policy and trajectory is shown in Figure

2.4.

An important goal is to show that the value function learned by reinforcement learning will converge in the limit (as $\Delta t \to 0$) to the optimal value function for continuous time control,

$$Q^* = \int_0^\infty (\mathbf{x(t)} - \mathbf{x_d})^T (\mathbf{x(t)} - \mathbf{x_d}) + u^T u \, dt$$

Although the discrete-time value function is conventionally defined as the expected long-term discounted sum of rewards (i.e. $V = \Sigma \gamma^i r_t$), in order to get it to converge to V^*, we modify by setting $\gamma = 1$ and multiply by Δt,

$$V_{\Delta t} = \Delta t \; \Sigma_{t=0}^{\infty} \; r_t = \Delta t \; \Sigma_{t=0}^{\infty} \; (\mathbf{x(t)} - \mathbf{x_d})^T (\mathbf{x(t)} - \mathbf{x_d})$$

We are able to set $\gamma = 1$ because we are only considering controllable systems which means that as the system evolves the rewards will approach 0 and therefore a discount factor is not necessary for convergence.

Since for any time step Δt, there is always a constant action that can move the system to the same point on the continuous time trajectory, as proven above, then the rewards $r_t = (\mathbf{x(t)} - \mathbf{x_d})^T (\mathbf{x(t)} - \mathbf{x_d}) + u^T u$ will match at these discrete points. Thus, the discrete-time value function is simply the numerical approximation of the continuous-time value function, Q^*.

$$Q^* = \int_0^\infty (\mathbf{x(t)} - \mathbf{x_d})^T (\mathbf{x(t)} - \mathbf{x_d}) + u^T u \, dt \approx \Sigma_{t=0}^{\infty} \, \Delta t \left[(\mathbf{x(t)} - \mathbf{x_d})^T (\mathbf{x(t)} - \mathbf{x_d}) + u^T u \right]$$

$$\lim_{\Delta t \to 0} Q_{\Delta t} \to Q^*$$

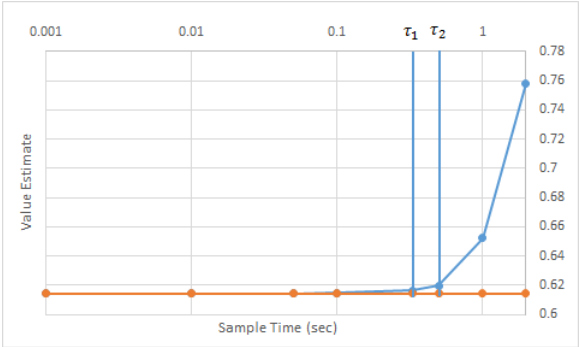


Figure 2.5: The value estimates from the learned policies as a function of Δt.

Furthermore, we can we put a bound on the error in the value function. The error is a function of the time step size (as with any numerical quadrature), as well as the maximum deviation between the optimal continuous-time trajectory and the constant-input trajectory between the discrete time points, where the coordinates coincide. This is bounded because the amount the constant-input trajectory can diverge from the continuous time path within Δt is limited by the dynamics of the system, which will force them to follow similar paths. These ideas are illustrated in Figure 2.5, the points along the curve show different Δt's that were used to train a reinforcement learner until convergence. The time constants of each of the example system are marked as vertical lines. The example system is discussed further throughout Section 3. It can be seen that the value approaches the optimal value asymptotically as $\Delta t \to 0$. The value estimates are cost for reaching the goal (i.e. the lower the better).

The response to a given input, and therefore the error, is dependent on the underlying system dynamics, and it is also dependent on how quickly a system responds to a new input. The smallest system time constant, τ, defines how rapidly the system responds to a new

control input. Therefore, we propose that a requirement for the selection of a Δt is that it be less than τ in order to ensure that these rapid responses can be controlled. At $\Delta t = dt$ the optimal trajectory will equal the approximate trajectory through state space.

2.3.1 Bounding the Single-step Error

The goal of bounding the total path error can be obtained by first quantifying the error for a single step of holding a control for a given sample time. Given Theorem 2.3, we know that the optimal continuous and the discrete trajectories should converge to some point along the reachable region. Therefore, the single-step error is bounded by the dynamics of the system.

Theorem 2.4 *The error introduced by a single discrete step $d(t)$ is bounded above by*

$$d(t) \leq \left\| t \left\| \mathbf{BK} \right\| e^{\mu(\mathbf{A})+\|\mathbf{BK}\|t} - \left[e^{\mathbf{A}t} - \mathbf{I} \right] \left[\Delta t \left\| \mathbf{BK} \right\| e^{\mu(\mathbf{A})+\|\mathbf{BK}\|\Delta t} \right] \right\| \left\| x(0) - x_d \right\|$$

In order to derive the error bound mathematically, we start with the fact that we know the optimal control policy for the infinite-horizon, continuous-time Linear Quadratic Regulator (LQR) objective function

$$J = \int_0^{t_f} \mathbf{w}(t)^T \mathbf{Q}\mathbf{w}(t) + \mathbf{u}(t)\mathbf{R}\mathbf{u}(t)\, dt \tag{2.5}$$

which is

$$u = -\mathbf{K}w = -\mathbf{K}(x - x_d) = -\mathbf{K}x + \mathbf{K}x_d \tag{2.6}$$

where $\mathbf{K} = \mathbf{B}^T\mathbf{P}$ and $\mathbf{P}$ is the solution to the Continuous Time Algebraic Riccati Equation (CARE). Plugging into the dynamics we get

$$\dot{w} = \mathbf{A}w + \mathbf{A}x_d + \mathbf{B}\mathbf{U} - \mathbf{B}\mathbf{K}w$$

$$\dot{w} = (\mathbf{A} - \mathbf{BK})w$$

$$w = e^{(\mathbf{A}-\mathbf{BK})t} w(0)$$

$$x - x_d = e^{(\mathbf{A}-\mathbf{BK})t}(x(0) - x_d)$$

meaning the optimal trajectory is defined as

$$x(t) = \sigma^*(t) = e^{(\mathbf{A}-\mathbf{BK})t}(x(0) - x_d) + x_d \tag{2.7}$$

The discrete time trajectory is defined as (from [14])

$$w(t) = e^{\mathbf{A}t} w(0) + \mathbf{A}^{-1}[e^{\mathbf{A}t} - \mathbf{I}]\mathbf{BL}$$

$$x(t) - x_d = e^{\mathbf{A}t}(x(0) - x_d) + \mathbf{A}^{-1}[e^{\mathbf{A}t} - \mathbf{I}]\mathbf{BL}$$

$$x(t) = \sigma(t) = e^{\mathbf{A}t} x(0) - e^{\mathbf{A}t} x_d + \mathbf{A}^{-1}[e^{\mathbf{A}t} - \mathbf{I}]\mathbf{BL} + x_d \tag{2.8}$$

where $\mathbf{L}$ is a constant input that causes the system to end in the same state as the optimal trajectory after Δt. $\mathbf{L}$ can be derived by

$$\sigma^*(\Delta t) = \sigma(\Delta t)$$

$$e^{(\mathbf{A}-\mathbf{BK})\Delta t}(x(0) - x_d) + x_d = e^{\mathbf{A}\Delta t}(x(0) - x_d) + \mathbf{A}^{-1}[e^{\mathbf{A}\Delta t} - \mathbf{I}]\mathbf{BL} + x_d$$

$$e^{(\mathbf{A}-\mathbf{BK})\Delta t}(x(0) - x_d) - e^{\mathbf{A}\Delta t}(x(0) - x_d) = \mathbf{A}^{-1}[e^{\mathbf{A}\Delta t} - \mathbf{I}]\mathbf{BL}$$

$$\left[\mathbf{A}^{-1}[e^{\mathbf{A}\Delta t} - \mathbf{I}]\mathbf{B}\right]^{-1}\left[e^{(\mathbf{A}-\mathbf{BK})\Delta t}(x(0) - x_d) - e^{\mathbf{A}\Delta t}(x(0) - x_d)\right] = \mathbf{L}$$

$$\mathbf{L} = \left[\mathbf{A}^{-1}[e^{\mathbf{A}\Delta t} - \mathbf{I}]\mathbf{B}\right]^{-1}\left[e^{(\mathbf{A}-\mathbf{BK})\Delta t}(x(0) - x_d) - e^{\mathbf{A}\Delta t}(x(0) - x_d)\right] \tag{2.9}$$

With these definitions, we can define the difference between trajectories as

$$d(t) = \sigma^*(t) - \sigma(t)$$

$$d(t) = \left[e^{(\mathbf{A}-\mathbf{BK})t}(x(0)-x_d)+x_d\right] - \left[e^{\mathbf{A}t}(x(0)-x_d)+\mathbf{A}^{-1}[e^{\mathbf{A}t}-\mathbf{I}]\mathbf{BL}+x_d\right]$$

$d(t)$ is related to the error in the value function via the triangle inequality in that the distance between both trajectories is bounded above by the distance to the goal from the optimal trajectory. That is,

$$\varepsilon = V_{\Delta t} - V^*$$

$$= \int_0^\infty V_{\Delta t}(x(t)) - \int_0^\infty V(x^*(t))\,dt$$

$$= \int_0^\infty V_{\Delta t}(x(t)) - V(x^*(t))\,dt$$

$$= \int_0^\infty (x(t)-x_d)^2 - (x^*(t)-x_d)^2\,dt$$

By the triangle inequality,

$$\leq \int_0^\infty (x(t)-x^*(t))^2\,dt$$

$$\varepsilon \leq \int_0^\infty (d(t))^2\,dt$$

Continuing by plugging in $\mathbf{L}$, we get,

$$= \left[e^{(\mathbf{A}-\mathbf{BK})t} - e^{\mathbf{A}t}\right](x(0)-x_d) -$$

$$\left[\mathbf{A}^{-1}[e^{\mathbf{A}t}-\mathbf{I}]\mathbf{B}\right]\underbrace{\left[\mathbf{A}^{-1}[e^{\mathbf{A}\Delta t}-\mathbf{I}]\mathbf{B}\right]^{-1}}_{\mathbf{C}}\underbrace{\left[e^{(\mathbf{A}-\mathbf{BK})\Delta t} - e^{\mathbf{A}\Delta t}\right]}_{\mathbf{D}}(x(0)-x_d)$$

$$= \left[e^{(\mathbf{A}-\mathbf{BK})t} - e^{\mathbf{A}t}\right](x(0)-x_d) - \left[\mathbf{A}^{-1}[e^{\mathbf{A}t}-\mathbf{I}]\mathbf{B}\right]\mathbf{CD}(x(0)-x_d)$$

$$d(t) = \left[\underbrace{\left[e^{(\mathbf{A}-\mathbf{BK})t} - e^{\mathbf{A}t}\right]}_{\text{part1}} - \underbrace{\left[\mathbf{A}^{-1}[e^{\mathbf{A}t}-\mathbf{I}]\mathbf{B}\right]\mathbf{CD}}_{\text{part2}}\right](x(0)-x_d) \tag{2.10}$$

Mathematically, the error bound can be derived by first defining the logarithmic norm,

$$\mu(A) = \lim_{h\to 0+} \frac{(\|I+hA\|-1)}{h}$$

In contrast to the previous definition, we need to redefine $d(t)$. Specifically, we need to take the norm in the definition of $d(t)$,

$$d(t) = \left\| [e^{(\mathbf{A}-\mathbf{BK})t} - e^{\mathbf{A}t}] - [e^{\mathbf{A}t} - \mathbf{I}][e^{\mathbf{A}\Delta t} - \mathbf{I}]^{-1}[e^{(\mathbf{A}-\mathbf{BK})\Delta t} - e^{\mathbf{A}\Delta t}] \right\| \|x(0) - x_d\|$$

We can then continue to derive a bound by approximating

$$[e^{\mathbf{A}t} - \mathbf{I}][e^{\mathbf{A}\Delta t} - \mathbf{I}]^{-1} \leq [e^{\mathbf{A}t} - \mathbf{I}]$$

$$d(t) \leq \left\| [e^{(\mathbf{A}-\mathbf{BK})t} - e^{\mathbf{A}t}] - [e^{\mathbf{A}t} - \mathbf{I}][e^{(\mathbf{A}-\mathbf{BK})\Delta t} - e^{\mathbf{A}\Delta t}] \right\|$$

A convenient bound for the perturbation of exponential matrices is defined as [15]

$$\left\| e^{(\mathbf{A}-\mathbf{BK})t} - e^{\mathbf{A}t} \right\| \leq t \, \|\mathbf{BK}\| \, e^{\mu(\mathbf{A})+\|\mathbf{BK}\|t}$$

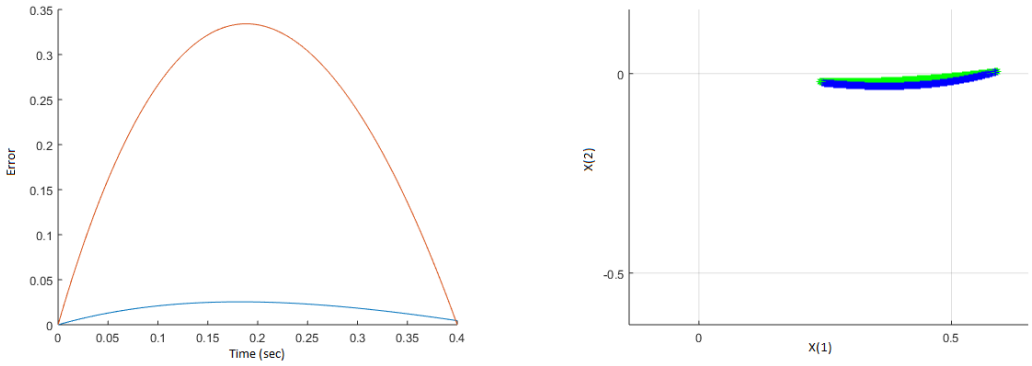
where $\mu(\mathbf{A})$ is the logarithmic matrix norm as defined above. Using this bound we can conclude that

$$d(t) \leq \left\| t \, \|\mathbf{BK}\| \, e^{\mu(\mathbf{A})+\|\mathbf{BK}\|t} - \left[e^{\mathbf{A}t} - \mathbf{I}\right] \left[\Delta t \, \|\mathbf{BK}\| \, e^{\mu(\mathbf{A})+\|\mathbf{BK}\|\Delta t}\right] \right\| \|x(0) - x_d\|$$

An example of this bound can be seen in Figure 2.6 using the following dynamics,

$$\mathbf{A} = \begin{bmatrix} -1 & 2 \\ -1 & -4 \end{bmatrix} \mathbf{B} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \mathbf{P} = \begin{bmatrix} 0.2917 & 0.0833 \\ 0.0833 & 0.1667 \end{bmatrix} \mathbf{K} = \begin{bmatrix} 0.5 & 0 \end{bmatrix}$$

with a $\Delta t = 0.63$



(a) (b)

Figure 2.6: The bound (red) on the error vs. the true error (blue) for a single step (sample time). The second figure shows the actual deviation between the discrete (blue) path and the optimal (green) paths.

2.3.2 Bounding the Total Path Error

Theorem 2.5 *The error introduced by a Δt (ε) over the whole path to the goal is bounded above by*

$$\varepsilon \leq \frac{\Psi \left\| (A - BK) \right\| C \left\| (x(0) - x_d) \right\| \Delta t}{1 - e^{-\alpha \Delta t}}$$

The single time step path error,

$$d(t) \leq \left\| t \left\| \mathbf{BK} \right\| e^{\mu(\mathbf{A}) + \left\| \mathbf{BK} \right\| t} - \left[e^{\mathbf{A}t} - \mathbf{I} \right] \left[\Delta t \left\| \mathbf{BK} \right\| e^{\mu(\mathbf{A}) + \left\| \mathbf{BK} \right\| \Delta t} \right] \right\| \left\| x(0) - x_d \right\|$$

can be used to derive a bound on the total error ε. The maximum of the first norm in $d(t)$ can be defined as

$$\Psi = \left\| \Delta t \left\| BK \right\| e^{(\mu(A) + \left\| BK \right\| \Delta t)} \right\|$$

which leaves us with

$$d(t) \leq \Psi \left\| x(0) - x_d \right\|$$

$$\varepsilon \leq \sum_{i=0}^{\infty} d(\Delta t \cdot i)$$

$$\varepsilon \leq \sum_{i=0}^{\infty} \Psi \left\| x(\Delta t \cdot i) - x(\Delta t \cdot (i+1)) \right\|$$

Assuming that the system is stable, the maximum difference between $\|x(\Delta t \cdot i) - x(\Delta t \cdot (i+1))\|$ is bounded above by $\dot{x}(\Delta t \cdot i)\Delta t$ because $x(i)$ will be decreasing exponentially toward $\mathbf{0}$ in w space as $i \to \infty$. Using this we get

$$\varepsilon \leq \Psi \sum_{i=0}^{\infty} \left\| (A - BK) e^{(A-BK)\Delta t i} (x(0) - x_d) \Delta t \right\|$$

where $x(0)$ and x_d are the overall start position and the overall goal for the whole path, respectively.

One important property of matrix norms is

$$\|AB\| \leq \|A\| \, \|B\|$$

for square matrices A and B. Using that property we can modify the definition of ε,

$$\varepsilon \leq \Psi \sum_{i=0}^{\infty} \|(A - BK)\| \left\| e^{(A-BK)\Delta t i} (x(0) - x_d) \Delta t \right\|$$

$$\varepsilon \leq \Psi \, \|(A - BK)\| \sum_{i=0}^{\infty} \left\| e^{(A-BK)\Delta t i} (x(0) - x_d) \Delta t \right\|$$

$$\varepsilon \leq \Psi \, \|(A - BK)\| \sum_{i=0}^{\infty} \left\| e^{(A-BK)\Delta t i} \right\| \|(x(0) - x_d)\| \, \Delta t$$

One final issue that we must deal with is the infinite sum in the definition of ε. First, the inner term of the norm

$$\left\| e^{(A-BK)\Delta t i} \right\|$$

is bounded by

$$\left\| e^{(A-BK)\Delta t i} \right\| \leq \|P\| \left\| e^{J} \right\| \left\| P^{-1} \right\| \leq C e^{-\alpha \Delta t i}$$

where $C = \|P\| \left\| P^{-1} \right\|$ and $\alpha = -\max_{\lambda_i} \Re \operatorname{eig}(A - BK)$.

After substituting we see that this can be formulated as a geometric series,

$$\sum_{i=0}^{\infty} C e^{-\alpha \Delta t i} = \sum_{i=0}^{\infty} C \left(\frac{1}{e^{-\alpha \Delta t}} \right)^{i} = \frac{C}{1 - \left(\frac{1}{e^{\alpha \Delta t}} \right)}$$

Therefore, we arrive at the final bound

$$\varepsilon \leq \frac{\Psi \|(A - BK)\| C \|(x(0) - x_d)\| \Delta t}{1 - e^{-\alpha \Delta t}}$$

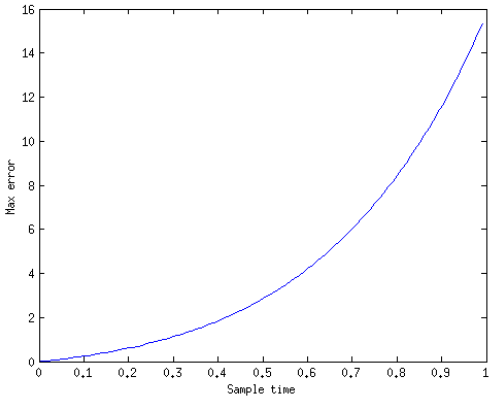


Figure 2.7: The relationship between Δt and the maximum error of the example problem.

Therefore, it can be seen that the maximum amount of error introduced is exponentially related to the Δt used to learn a control for a given system. This can be seen in Figure 2.7 where the dynamics of the system for the plot are defined by

$$\mathbf{A} = \begin{bmatrix} -1 & 2 \\ -1 & -4 \end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$$

3 EXPERIMENTS

3.1 Verification

In order to test the bound and the theory in this work two main systems are defined: a simple theoretical system that adheres to the assumptions of stability and controllability and a real-world system that can be linearized in order to run in the simulator discussed below. With the systems defined, the first focus is to show that the single-step error bound holds by running single steps in both the real-world and example systems. Therefore, a learner will be trained until convergence on each system and the error bounds for each single step along the path will be compared to the perceived error. Building on the single-step error is the next step, the full-path error will be analyzed for both the example and real-world systems in a similar fashion to the single-step error. In addition, for both of the systems, it will be important to show that in the limit as $\Delta t \to 0$ the value function learned approaches the optimal: $V^{\Delta t} \to V^*$. Therefore, a random initial state can be chosen and the policy for each Δt can run until the goal is reached. The accumulated cost for following each policy will then be plotted in order to show the relationship between Δt and the value for using a specific policy.

The learning algorithm used in these examples is the adaptive policy iteration algorithm proposed by Bradtke in his work on applying reinforcement learning to LQR problems [7]. By following this algorithm, it is possible to learn to control linear systems without knowledge of the model. The number of policy evaluation stages was adjusted in

order to give a good approximation of the Q-function for a particular policy depending on the sample time being used and there were at most 50 policy updates depending on whether or not the policy had converged which is defined as 5 policy updates in a row with a change in weights less than 1×10^{-3} .

A simulator was built in order to test the ideas from the theory in this work. The simulator is written in MATLAB due to its plethora of mathematical tools, especially for matrix manipulation. The simulation of the evolution of the linear systems is done through fourth-order Runge-Kutta methods [16]. In order to approximate the evolution of the system two notions of Δt are necessary. First, a base Δ_{dt} is used as the absolute smallest amount of time in the simulation and this is chosen to be at least an order of magnitude smaller than the other notion of discrete time Δt_{ct} which is the time a control must be held before a new one can be selected. Δt_{ct} is the same Δt that is presented in throughout the theory in this work.

3.1.1 Example System

The example system is an arbitrary, stable, and controllable LTI system. The dynamics are defined as

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} -1 & 2 \\ -1 & -4 \end{bmatrix}}_{\mathbf{A}} \mathbf{x}(t) + \underbrace{\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}}_{\mathbf{B}} \mathbf{u}(t)$$

3.1.2 Real-world System

The real-world system being tested in this work is that of an aircraft (Commander 700) cruising at a constant speed with the goal of controlling the longitudinal dynamics.

Figure 3.1 describes the problem visually. The longitudinal dynamics can be approximated as described in [17]. First, we define the state variables

$$x = \begin{bmatrix} U \\ \alpha \\ q \\ \theta \end{bmatrix}$$

where

$U =$ Velocity of the plane (ft/sec)

$\alpha =$ Angle of attack (radians)

$q = \dot{\theta} =$ Pitch rate (rad/sec)

$\theta =$ Pitch angle (rad)

The system can be modeled as a LTI system as

$$\dot{\mathbf{x}}(t) = \underbrace{\begin{bmatrix} -0.0246 & 6.847 & 0 & -32.17 \\ -0.0012 & -1 & 1 & 0 \\ 0 & -3.535 & -2.245 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{A}} \mathbf{x}(t) + \underbrace{\begin{bmatrix} 0 \\ -0.0915 \\ -8.574 \\ 0 \end{bmatrix}}_{\mathbf{B}} \mathbf{u}(t)$$

Figure 3.1: Depiction of the real-world longitudinal control problem.

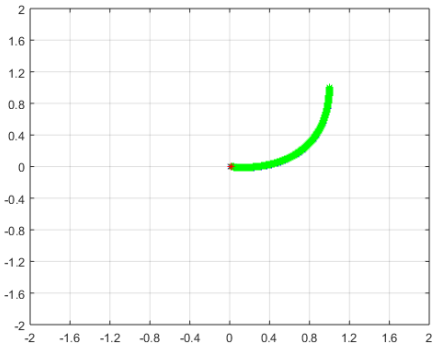
3.2 Effect of Discrete Time

The section will explore the quality of policy produced by the reinforcement learner as a function of Δt. In order to show that the quality increases as $\Delta t \to 0$, the systems will

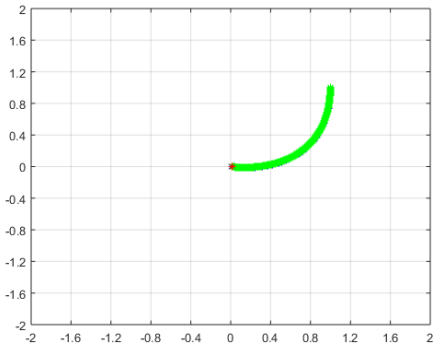
be trained until convergence for a number of different Δt's. Convergence for this work is defined as policy updates with a change in weights less than 1×10^{-3}. The policies will then be evaluated by placing the systems in a consistent starting position then measuring the cost to get to the goal using the policy. Plots will be generated from the cost-to-goal for a discrete-time policy against the optimal cost-to-go from that starting position.

3.2.1 Example System

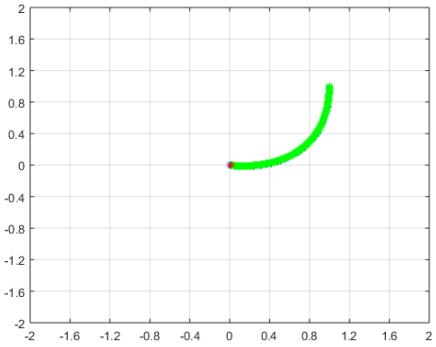
Using the dynamics defined in Section 3.1.1, a learner was run until it converged to a policy. The system was then set in the initial position of: $x = [1;\ 1]$ and the learned policy was followed until the state was close enough to the goal which was defined as $(x - x_d) < 1 \times 10^{-3}$. Sample times of $[0.001, 0.01, 0.05, 0.1, 0.33, 0.5,$ and $1]$ were used.



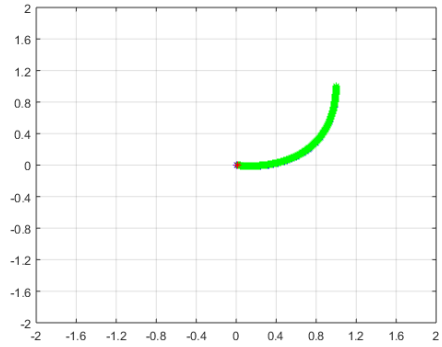
(a) $\Delta t = 0.001$



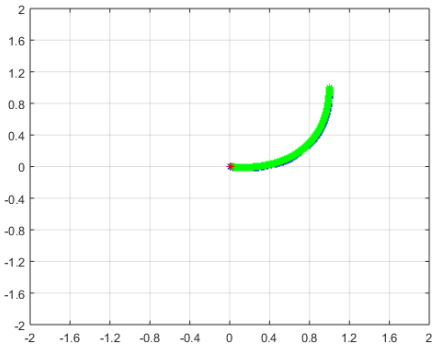
(b) $\Delta t = 0.01$



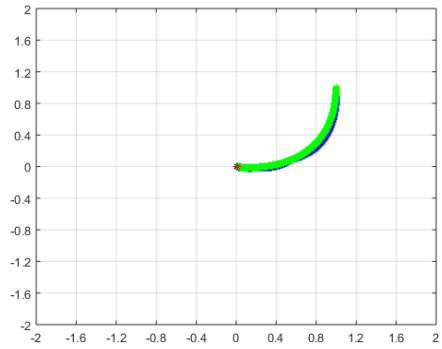
(c) $\Delta t = 0.05$



(d) $\Delta t = 0.1$

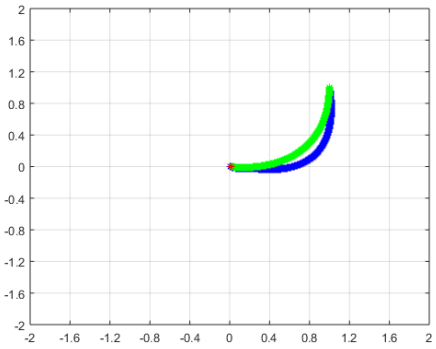


(e) $\Delta t = 0.33$



(f) $\Delta t = 0.5$

Figure 3.2: The learned control makes jumps along the optimal trajectory introducing more error as the sample time increases.



(g) $\Delta t = 1.0$

Figure 3.2 continued...

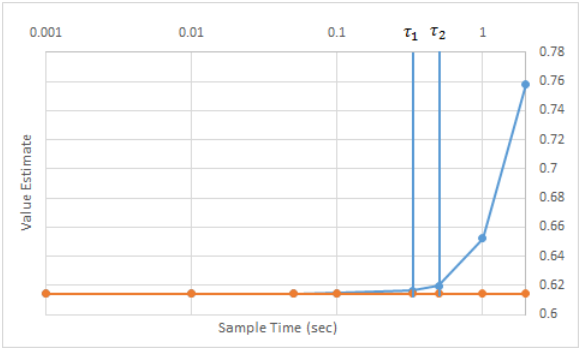


Figure 3.3: The value estimates from the learned policies as a function of Δt for the example system. The red line represents the optimal value and the blue line is the value estimate for the learned policy for a given Δt.

From Figure 3.3, we can clearly see that the quality of the policy increases as $\Delta t \to 0$. We see that in the limit as $\Delta t \to 0$ the value approaches the optimal value which is denoted by the red asymptote. In addition, from Figure 3.2 we can see that the policies make jumps along the optimal trajectory on the way to the goal.

3.2.2 Real-world System

The Commander 700 longitudinal flight controls example with the dynamics as de-

scribed in Section 3.1.2 was tested in the same manner as the example system above. The system was started with the initial conditions of $\alpha = 5.25°$, $U = 200$ ft/sec, $q = 0$ deg/sec, $10°$. Sample times of $[0.001, 0.005, 0.01, 0.05, 0.1, 0.15, 0.2, 0.5, 1.0,$ and $2.0]$ were used.

$$X(0) = \begin{bmatrix} 200 \\ 0.0916 \\ 0 \\ 0.1745 \end{bmatrix} \quad X_d = \begin{bmatrix} 200 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

As was done with the last system the learned policy was followed until the state was close enough to the goal which was defined as $(x - x_d) < 1 \times 10^{-3}$.

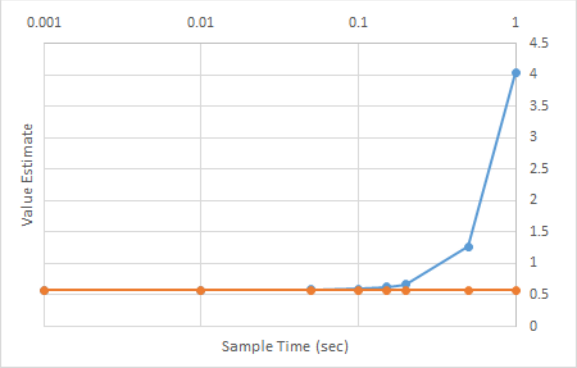


Figure 3.4: The value estimates from the learned policies as a function of Δt for the Commander 700 system. The red line represents the optimal value and the blue line is the value estimate for the learned policy for a given Δt.

This example is of particular interest because the system is real and the control is somewhat complicated. First, the only available control is the elevator of the aircraft and speed is included in the state-space description. Therefore, any change in speed must be accomplished through attitude changes. If the learner had the ability to control the thrust of the aircraft, then the optimal policy might not need to trade altitude changes for speed. Figure 3.5 shows the progression of states for the Commander 700 example from one of

the controllers that was learned. The controller first levels off the aircraft, but the initial angle causes the aircraft to lose velocity. Therefore, the controller must trade off altitude for velocity in order to reach the desired goal. Figure 3.6 shows the angle of attack over time by following one of the learned controllers. Figure 3.4 shows the value estimates for following the learned policies for different Δts to the goal. We can clearly see that as $\Delta t \rightarrow 0$ we get closer to the optimal value.

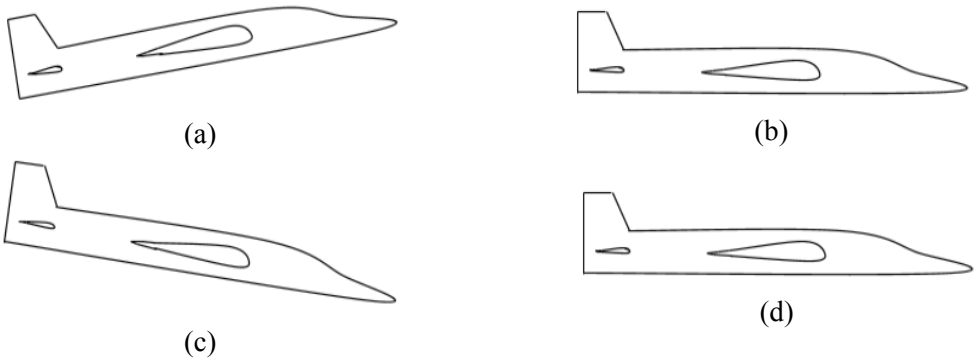


Figure 3.5: The learned control must make changes in altitude in order to reach the goal state. (a) shows the initial state of the aircraft. (b) shows aircraft at a level angle, but its velocity is lower than the goal. (c) shows control beginning to trade altitude in order to gain velocity. (d) shows the aircraft returning back to a horizontal flight path and reaching it's goal.

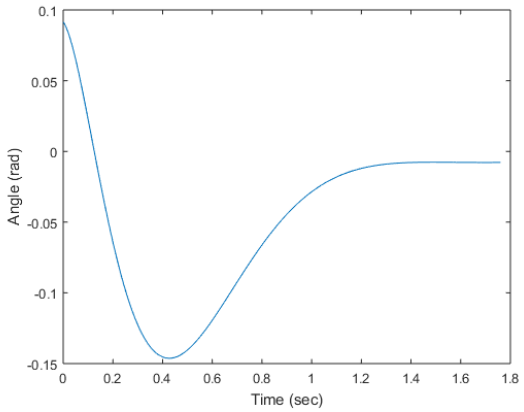


Figure 3.6: The AOA of the Commander 700 over following the learned policy.

3.3 Single Step Error Bounds

This sections discusses the first step toward deriving a bound on the error caused by the introduction of a sample time (Δt).

3.3.1 Example System

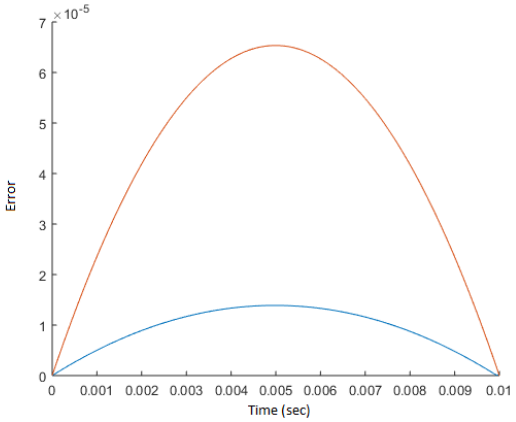


Figure 3.7: The single step error (blue) and its bound (red) for the example system when the end state is exactly equal to a point on the optimal trajectory.

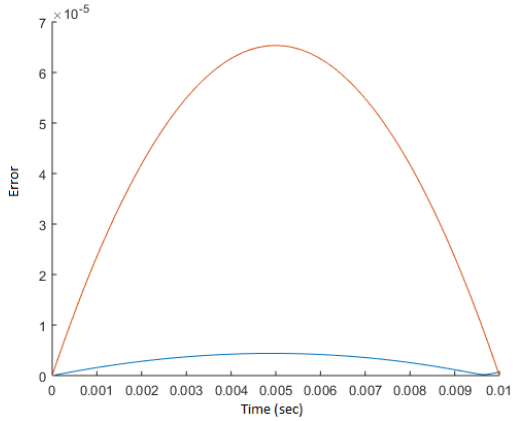


Figure 3.8: The single step error (blue) and its bound (red) for the example system when the end state is close to a point on the optimal trajectory.

Figure 3.7 shows the bound vs the perceived error for a single time step. It can be seen that the bound for the example is relatively tight. Figure 3.8 shows that the perceived error is slightly higher than the bound near $t = \Delta t$. This is due to the learned controller not being exactly equal to the optimal controller. In order to guarantee the learned controller is exactly equal to the optimal controller, we would need to run the policy iteration algorithm for an infinite amount of time. For any finite stopping time, some error will exist [7].

3.3.2 Real-world System

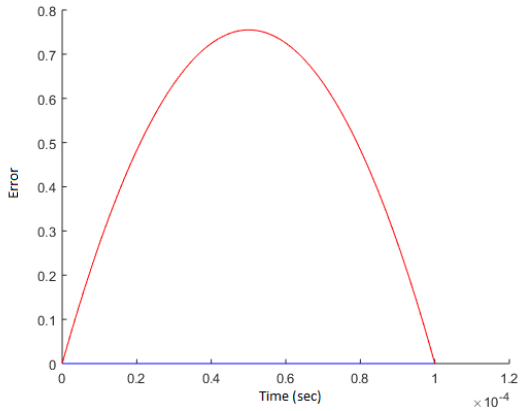


Figure 3.9: The single step error (blue) and its bound (red) for the Commander 700 example.

Figure 3.9 shows the bound vs the perceived error for a single time step for the Commander 700 example. It can be seen that the bound for the example is not tight. This is a consequence of the spectral norm, $\|BK\|$, being very large for this example.

3.4 Total Path Error Bounds

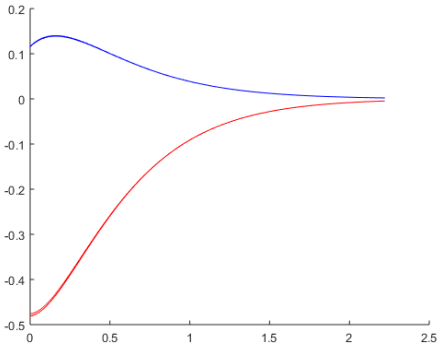
This section shows the full path error which is an expansion of the experiments from the previous section. The figures will show that an approximation to the optimal control policies is made and that the error bound holds for all systems that were tested.

3.4.1 Example System

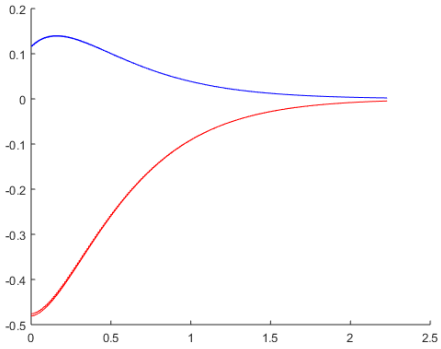
Table 3.1 shows the maximum error bound and the actual errors perceived between following the optimal policy and following the learned policies for different sample times. We can see that the full-path error bound holds for the example problem. In addition, Figure 3.10 shows the learned policies being better approximations to the optimal policy as $\Delta t \to 0$. A plot of the perceived errors for the example problem is shown in Figure 3.11. We can see that the error is exponential in the sample time used as the bound predicts.

Maximum Error		
Sample Time	**Max Error**	**Actual Error**
0.001	0.0067	1.0197×10^{-7}
0.01	0.0687	1.0201×10^{-5}
0.05	0.3820	2.5640×10^{-4}
0.1	0.8706	0.0010
0.33	5.1471	0.0114
0.5	11.7779	0.0271
1.0	72.8085	0.1159

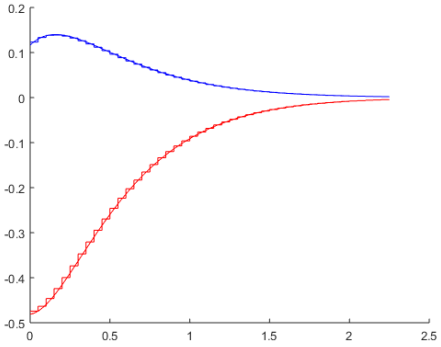
Table 3.1: Error bound for different sample times and the actual perceived errors for the example problem.



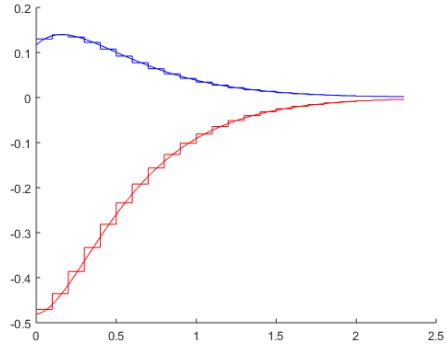
(a) $\Delta t = 0.001$



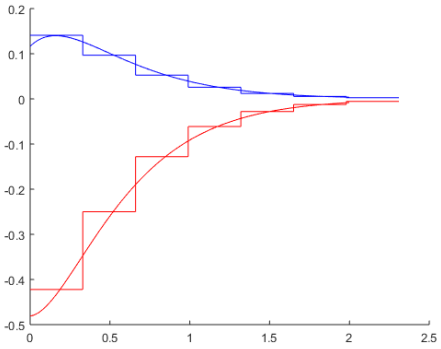
(b) $\Delta t = 0.01$



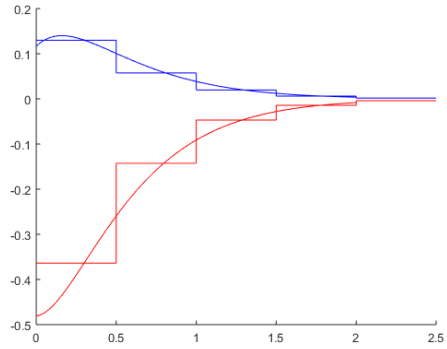
(c) $\Delta t = 0.05$



(d) $\Delta t = 0.1$

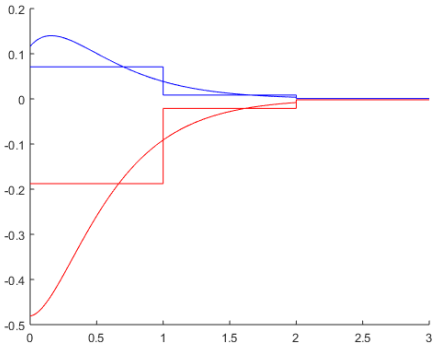


(e) $\Delta t = 0.33$



(f) $\Delta t = 0.5$

Figure 3.10: The learned control is approximated with different sample times. There are two control inputs for the example problem. The blue line shows the first input and the red line shows the second. The larger the sample time the worse approximation of the true continuous-time optimal value function.



(g) $\Delta t = 1.0$

Figure 3.10 continued...

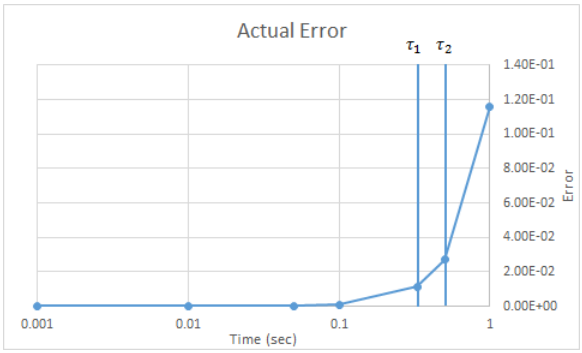


Figure 3.11: The perceived errors plotted vs. sample time for the example problem.

3.4.2 Real-world System

Table 3.2 shows the maximum error bound and the actual errors perceived between following the optimal policy and following the learned policies for different sample times. We can see that the full-path error bound holds for the Commander 700 control problem. In addition, Figure 3.13 shows the learned policies being better approximations to the optimal policy as $\Delta t \to 0$. A plot of the perceived errors for the Commander 700 control problem is shown in Figure 3.12. We can see, once again, that the error is exponential in the sample time used as the bound predicts.

Maximum Error		
Sample Time	**Max Error**	**Actual Error**
0.001	1.0780×10^{9}	1.9475×10^{-4}
0.01	2.8771×10^{10}	0.0192
0.05	1.1289×10^{13}	0.4454
0.1	5.2716×10^{15}	1.6038
0.15	1.8457×10^{18}	3.3053
0.2	5.7426×10^{20}	5.3301
0.5	2.3024×10^{35}	34.6373
1.0	2.1272×10^{59}	111.8434

Table 3.2: Error bound for different sample times and the actual perceived errors for the Commander 700 example.

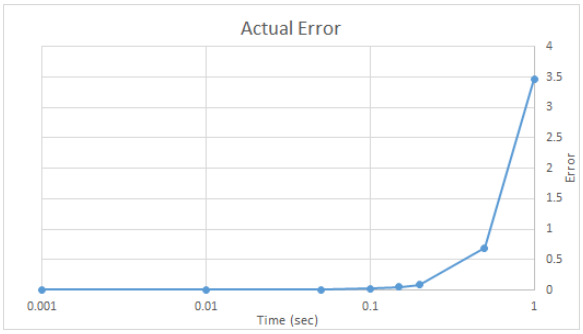
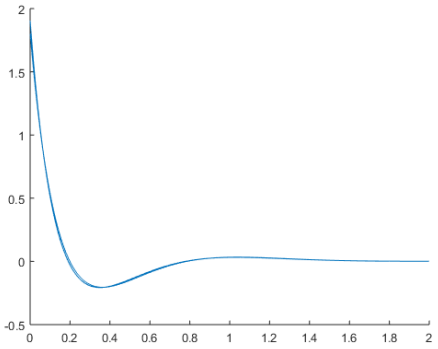
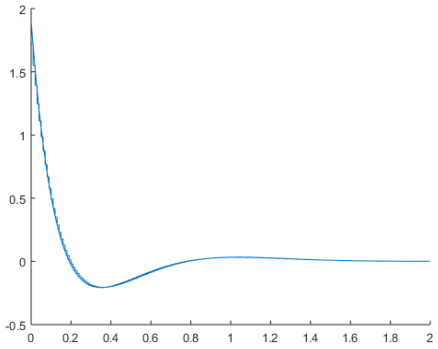


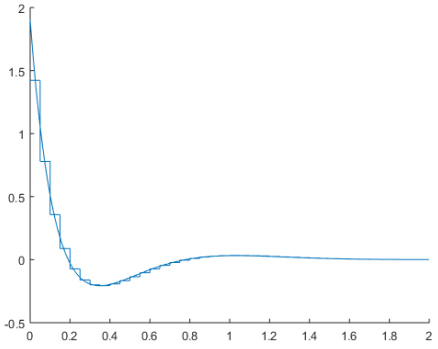
Figure 3.12: The perceived errors plotted vs. sample time for the Commander 700 problem.



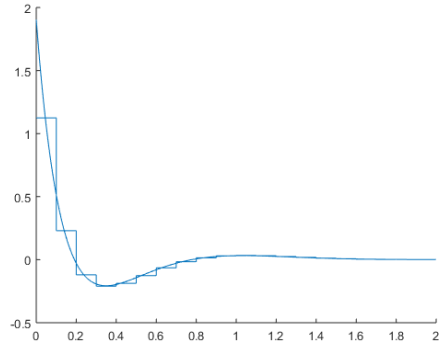
(a) $\Delta t = 0.001$



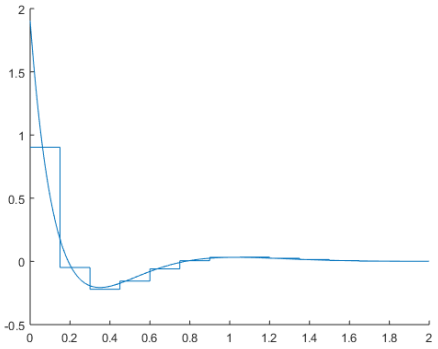
(b) $\Delta t = 0.01$



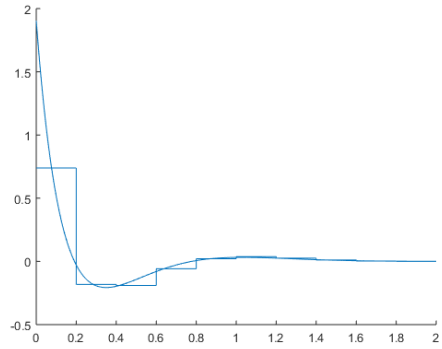
(c) $\Delta t = 0.05$



(d) $\Delta t = 0.1$

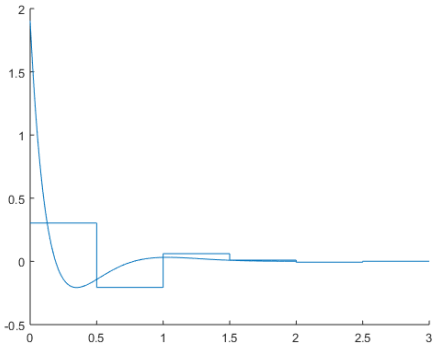


(e) $\Delta t = 0.15$

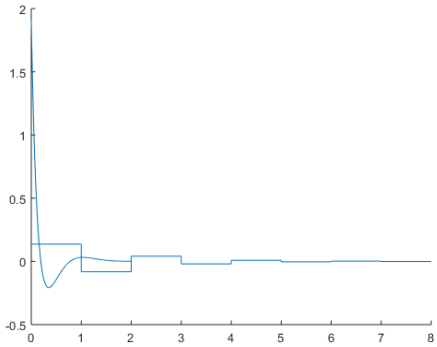


(f) $\Delta t = 0.2$

Figure 3.13: The learned control is approximated with different sample times. There are control input shown is the angle of the elevator on the Commander 700. The larger the sample time the worse approximation of the true continuous-time optimal value function.



(g) $\Delta t = 0.5$



(h) $\Delta t = 1.0$

Figure 3.13 continued...

4 SUMMARY AND CONCLUSIONS

It has been shown that a LTI system's behavior can be decomposed into a long-term and a transient response. The quickness with which the system converges to the long-term response is dependent on the time constants of the LTI system. It has also be shown that controls for systems defined in terms of the LQR problem can be successfully learned by a reinforcement learner in continuous action and state spaces.This was accomplished without the need to discretize either the state or action spaces. The key in doing this was using a weighted basis set representation where the set of basis functions ϕ_i were the quadratic combinations of the state and action variables from the state space representation. We have also seen that an increase in sample time negatively correlates with the quality of control learned and that $V^{\Delta t} \rightarrow V^*$ as $\Delta t \rightarrow 0$. In addition, a bound on the maximum error for a learner that follows the discrete points along the optimal trajectory was derived. The adaptive policy iteration algorithm proposed by Bradtke can then be used to adjust the weights on the basis functions.

This work also sheds light on the ability of reinforcement learners to learn to control arbitrary linear systems. It is not a trivial issue due to the complexity involved in the dynamics of some linear systems because although we call them linear systems the internal behavior in response to perturbations in the control input produce non-linear behavior. This work shows that even for higher order systems a reinforcement learner can learn to control in an optimal or near-optimal way. However, learning must be done in a specific way

in order to avoid the pitfall of converging to a non-optimal representations of value or Q functions.

The other importance of this work is an understanding of the amount of error being inserted into real-world control systems. Many times the sample time for continuous linear systems is discretized arbitrarily. For instance, it might be the smallest possible sample time for a given hardware implementation or a time that is convenient for other parts of the control system. However, it is important to note that whenever these continuous linear systems are discretized, some error is inserted into the quality of control. This work also gives some indication to designers or implementers of non-reinforcement learning control systems how much error is being inserted into their ability to control for discretized sample times as the error bound is still relevant in those cases.

Originally, the goal of this work was to study the effect of Δt on first-order linear systems. That is, systems which are defined only by first derivatives. However, as was shown in the Commander 700 example, a reinforcement learner can learn to control higher order systems and the bound is still applies. The only portion of theory affected by the limitation to first-order systems is the system behavior decomposition presented in Section 2.1. It is affected because the transient behavior in higher order systems involves complex eigenvalues meaning the state evolution is no longer an exponential convergence to a new equilibrium point and instead it is possible for ringing and oscillations to exist.

The fact that a reinforcement learner can learn optimal or near-optimal controls for linear systems is a very important result in CPS and other engineering fields. It means

that a control can be learned for an arbitrary system regardless of the order and without the need to know the true model of the system dynamics. Moreover, as the selection of $\Delta t \to 0$ the value function learned approaches the optimal continuous time value function and, as presented, the error is exponentially correlated with the selection of Δt. These results provide scientists and engineers with new insights into both the use of reinforcement learners on linear systems and the selection of a sample time for such a reinforcement learner. For example, it is obvious that the sample time should be chosen to be less than the smallest time constant of system. This is due to the fact that a controller must be able to control the transient dynamics of a system in order to perform optimally.

One major improvement that could be made to this work would be to tighten the error bound. The current bound was observed to be loose for certain experiments, such as the Commander 700 example. This could be due to using the maximum possible error over a step or the bound on the perturbation of matrix exponentials being loose itself. There are some possible paths to take for future work on this. First, instead of assuming the maximum possible error over a step, one could try to use the actual exponential definition of the single step error in the summation along the full path. Second, there is a promising approach that requires looking at the problem from the view of the difference in value functions instead of the distance between trajectories to the goal. In this view, it is necessary to study the affects of the discretization of the system dynamics on the solutions to the DARE which could then be compared with the solution from the CARE in order to find an error bound.

The study should find that

$$V_{\Delta t} = x^T [F\Delta t]x \approx V_* = x^T Px$$

One important topic that is left to study is the affect on the value function, control policies, and error introduced by Δt for systems where actions are limited to a discrete set. In many applications, the state may be evolving in continuous time, but the actions available are limited to a discrete set. For example, consider a cruise control system which attempts to match a target speed. If we assume that a controller only has the ability to apply the accelerator and the brakes as discrete intervals (e.g. 5% from 0% to 100%), how would the overall value function be affected and how would the introduction of a Δt affect the error? The first step to approach this problem would be to study when such a system is even controllable or stabilizable. The ideas from control theory on controllability and stabilizability could likely be adapted for this application. Next, a theory of the optimal action to take for the relevant systems must be developed. Pontryagin's minimum principle can be used to derive an optimal action in cases where the set of actions is discrete. Finally, the error introduced by Δt would becomes even more complicated in this setting. There is no longer a guarantee that an action exists that will intersect with the optimal trajectory. Therefore, it is likely that an optimal policy for a given Δt will perform some sort of course correction around the continuous optimal time trajectory if such an action even exists in the set. For example, consider an example of a car attempting to maintain a straight line in the middle of an uneven road. If we assume that the actions available to a controller are discrete angles on the steering wheel then for a larger Δt the optimal action might overshoot

the middle line and at the next time point a course correction must be made. From this example, it is clear that oscillations are possible and thus if there is no damping affect on these oscillations the error is unbounded. Therefore, a study of when a set of actions would cause the error to be bounded is necessary. It is clear that the logical extension of this work is an error bound on such systems.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[2] K. Kirkpatrick and J. Valasek, "Approximation of agent dynamics using reinforcement learning," in *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, (Grapevine, TX), p. 875, January 2013.

[3] K. Kirkpatrick, *Reinforcement Learning Control with Approximation of Time-dependent Agent Dynamics*. PhD thesis, Texas A&M University. College Station, TX, 2013.

[4] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1 ed., 1957.

[5] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[6] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, Dec. 2003.

[7] S. J. Bradtke, "Reinforcement learning applied to linear quadratic regulation," *Advances in Neural Information Processing Systems*, pp. 295–295, 1993.

[8] W. L. Brogan, *Modern Control Theory (3rd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1991.

[9] M. Johnson, R. Kamalapurkar, S. Bhasin, and W. E. Dixon, "Approximate-player nonzero-sum game solution for an uncertain continuous nonlinear system," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 8, pp. 1645–1658, 2015.

[10] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos, *Reinforcement Learning and Optimal Adaptive Control*, pp. 461–517. John Wiley & Sons, Inc., 2012.

[11] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[12] E. Nelson and T. Ioerger, "Learning to control first order linear systems with discrete time reinforcement learning," in *Proceedings of the 2016 International Conference on Embedded Systems, Cyber-physical Systems, & Applications*, pp. 155–161, 2016.

[13] E. D. Sontag, *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, vol. 6. Springer Science & Business Media, 2013.

[14] H. Saadat, *Computational Aids in Control Systems Using MATLAB*. New York, NY, USA: McGraw-Hill, Inc., 1st ed., 1993.

[15] B. Kågström, "Bounds and perturbation bounds for the matrix exponential," *BIT Numerical Mathematics*, vol. 17, no. 1, pp. 39–57, 1977.

[16] J. C. Butcher, "A history of runge-kutta methods," *Applied Numerical Mathematics*, vol. 20, no. 3, pp. 247–260, 1996.

[17] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control*. New York, NY: Hemisphere, 1975.