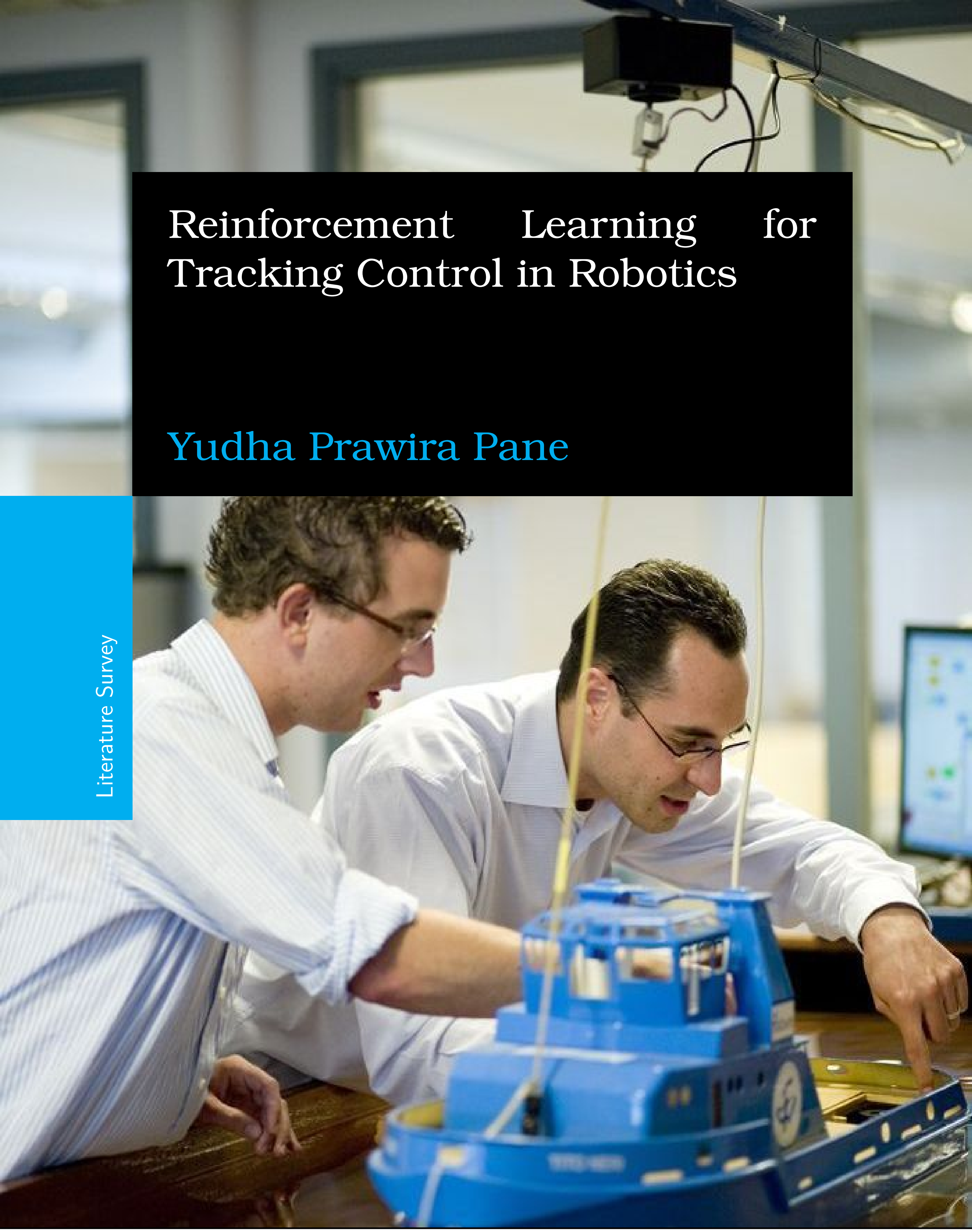

Reinforcement Learning for Tracking Control in Robotics

Yudha Prawira Pane

Literature Survey

TU Delft Delft University of Technology

Delft Center for Systems and Control

Reinforcement Learning for Tracking Control in Robotics

Literature Survey

Yudha Prawira Pane

February 3, 2015

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

The implementation work in this thesis was done at DCSC's robotics lab.

Abstract

This is an abstract.

Table of Contents

List of Figures

List of Tables

Preface

Acknowledgements

Delft, University of Technology
February 3, 2015

Yudha Prawira Pane

"It can scarcely be denied that the supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience"

— *Albert Einstein*

Chapter 1

Introduction

Reference or trajectory tracking is one of the building blocks to perform a complex task in robotics. Given a desired path/trajectory, the robot must be able to follow it as quickly as possible with minimum error. Capability to perform this precise tracking is crucial for robots that are to be deployed at manufacturing industries such as semiconductor, automotive, and recently, the emerging application of 3D printing.

Statistics by International Federation of Robotics (IFR) [1] shows that the global sales of industrial robots continues to increase steadily. In 2014, it is expected that the total number of industrial robots installed reaches 205,000 units, a rise of approximately 15 % from previous year. The survey points out that the mature markets such as automotive, electronics, and metal are responsible for such growth.

Meanwhile, there is also a growing interests in applying robots to relatively new applications such as 3D printing, architecture, and art. For instance, research done by Gramazio et. al [2] [3] aims to push the capability of industrial robots to make direct fabrication based on CAD model a reality. The advantage of using robots over conventional CNC machines lies on their flexibility, easy-to-adapt feature, and high degrees of freedom (DoF) – enabling execution of difficult configuration in 3-dimension (3D) space. These aforementioned applications demand high precision since a minuscule of error could lead to a defect product or even worse, a disaster. Therefore a precise, accurate reference tracking capability is inevitable.

In order to achieve this, a reference tracking control is needed. However, a robot brings along non-linearities, noises, and external disturbances that are difficult to model, let alone compensate. This unknown properties often hinders the controller to perform optimally. A class of controllers which solely depends on the system's model will surely suffer a poor tracking accuracy. The natural answer to this problem is to introduce a controller capable of adjusting its parameter overtime by comparing the reference to the actual trajectory. By doing so, the controller will have an extra degree of freedom to compensate for the unknown properties hence improving the tracking quality. The controller of such characteristic belongs to the class of adaptive controller.

In this thesis, a method to improve the performance of nominal controller by using Reinforcement Learning (RL) is proposed. Despite decades of extensive research on RL, its application to optimize tracking problem in robotics is still a relatively unexplored topic. Based on the literature, there are three potential approaches to address the tracking problem. The first one comes from the work of Lewis et. al. on RL for optimal control. Lewis and his group have been developing a comprehensive research on RL for solving the solution to adaptive optimal control. Their research has been extended for discrete [4] and continuous time [5], for linear [6] and non-linear system [7]. Furthermore, their technique could also be applied in Q-learning [4] and actor-critic structure [7]. The second approach is proposed by Bayiz et. al. in [8]. The paper discusses a slightly different approach by using RL to learn disturbance compensation for nonlinear system. This disturbance compensation acts as an additive input signal to the control signal. Finally, the third approach uses the notion of adaptive gain scheduling. Buchli et.al present an algorithm called Policy Improvement with Path Integral (PI^2) to vary the gain of a Proportional Derivative (PD) controller in order to achieve a desired terminal state [9] [10]. Having explained the motivation of this thesis, now we are ready to define the research problem.

1-1 Problem Definition

The fundamental problem in this literature study concerns the non-optimal performance of nominal controller with respect to reference tracking task. Hence the research question can be raised as follows.

"Is it possible to integrate Reinforcement Learning technique to a nominal controller in a certain structure such that reference tracking performance of the controlled system significantly improves?"

While conducting a research, it is often wise to restrict oneself to a simple context, but still captures the essential elements of the original problem [11]. Therefore, in answering this question, some simplifying assumptions are made.

1. The system to be controlled is fully actuated
2. The system to be controlled is observable. This assumption is necessary in order to satisfy Markov property [12].
3. Nominal, stabilizing controller is available
4. Identification reveals some information about the system, but alone is not adequate to design an accurate reference tracking controller.

1-2 Goal of the Thesis

The goal of this thesis are as follows:

1. To provide a general framework of improving tracking control using RL
2. To apply and compare existing method of RL for tracking application to the 3D printing robot setup
3. To come up with modification or improvement of previous methods

1-3 Literature Study Approach

In order to build a strong theoretical foundation for later implementation, the following literature approach is used. The order does not necessarily represent a sequential process.

1. To gather as many relevant papers as possible from reputable academic search engines. Relevant means papers which deal with RL and control system. Additional pointer to tracking problem is heavily considered. Examples of sources being used are Web of Science, IEEE Xplore and Google Scholar.
2. To discuss the detail of future experimental setup (UR5 3D printing robot) with Marco de Gier, who was working on the setup at the time this literature is written.
3. From the papers, extract existing methods which have the potential for application to the future experiments. So far, there are 3 different methods that are considered. These methods will be explained in detail in Chapter 3.
4. Create simple simulation programs showing how each method works

1-4 Outline

The structure of this literature review is arranged as follows. In the next chapter, an introductory materials of RL is presented. This covers the framework widely used in RL (Markov Decision Process), the principle of value and policy iteration, the formulation of RL for continuous space, and the actor-critic structure which suits the framework of control system. Chapter 3 provides the result of literature study being conducted. This includes the detailed explanation of methods found and their comparison. Furthermore, a new controller is proposed.

Chapter 2

Reinforcement Learning Preliminaries

This chapter is dedicated to present a concise theory of reinforcement learning. The first section will show how a certain goal can be formalized as a reward maximization – one of the ideas which serves as a basic foundation of Reinforcement Learning (RL). Section 2-2 explains the basics of Markov Decision Process (MDP), a general framework used in RL problem. The notion of value function will be discussed in Section 2-3. Subsequently, a method to solve RL, namely policy and value iteration will be developed in section 2-4. Finally, Section 2-5 will discuss the actor-critic structure which is an alternative solution to policy iteration.

2-1 Goal as Cost Minimization

The nature of RL is inspired by the way living organisms learn to reach their desired goals. Animals for instance, learn by first acting on the environment, observe the changes that occur, and improve their action iteratively. One example is a circus lion that is tasked to perform acrobatic show while its trainer observing the progress. If the lion successfully executes the task, it will be rewarded with foods. Conversely, punishment will be inflicted whenever it fails. The lion initially has no idea of how to perform the task. However through trial and error, it will follow its instinct to increase the frequency of receiving rewards while trying its best to avoid punishments. In a certain duration of training, the circus lion will be finally able to perform the task flawlessly.

Now we will formalize above illustration for robotics application. A robot can be described by its states x_k with subscript k denoting time instance. Applying an action u_k will bring the robot to state x_{k+1} with immediate reward r_{k+1}. Subsequently, at $k+1$ the robot applies u_{k+1} which yields state x_{k+2} and r_{k+2}. This action-state-update iteration is run for infinite time instances. The goal is defined as maximization of cumulative reward the robot receives. In control engineering, reward is usually replaced with cost. In that case the goal is defined as minimization problem. Starting from now, we will define goal as minimization of future cost J.

From the sequence of cost obtained over time, we can define a formalization of goal, called expected return. Return J_t is a function that maps the sequence of costs into real number. An example of return is the sum of the costs.

$$J_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T \tag{2-1}$$

2-2 Markov Decision Process

MDP is defined as a tuple $\langle X, U, f, \rho \rangle$ which satisfies Markov property [13]. The detailed explanation of Markov property can be found on [12] section 3.5 but the main idea is that to determine the probability of a state at certain time, it is sufficient to only know the state of previous time instance. The elements of the tuple are:

- X is the state space
- U is the action space
- $f : X \times U \rightarrow X$ is the state transition function (system dynamics)
- $\rho : X \times U \rightarrow \mathbb{R}$ is the reward function

In control engineering, f represents the system dynamics which is a transition function mapping a current state and action to the one-step ahead state up to a probability distribution. This probability distribution is mathematically denoted in Equation (2-2).

$$\Pr\{x_{t+1} = x', r_{t+1} = r | x_t, u_t\} \tag{2-2}$$

where x denotes state, u denotes action, and r denotes immediate reward obtained upon applying the input on the corresponding state.

2-3 Value Function

Value function describes how good a particular state or state-action pair under a certain policy. As previously explained, in this thesis we will stick to control engineering convention by seeing RL as cost minimization problem. Therefore, the smaller value function of a state x, the better it is. The optimum value function is denoted by $V^\pi(x)$ for state-value function and $Q^\pi(x, u)$ for action-value function. Furthermore, one can always find a policy which gives an optimal value function V^*. This optimal value function respects the Bellman optimality equation, which can be written as 2-3. The action-value function counterpart is denoted in Equation 2-4.

$$V^*(x) = \rho(x, u) + \gamma \min_u V^*(f(x, u)) \tag{2-3}$$

$$Q^*(x, u) = \rho(x, u) + \gamma \min_u Q^*(f(x, u), u') \tag{2-4}$$

Discount factor γ is introduced to avoid the value function goes to infinity. Once V^* is known, the optimal policy can be taken in a greedy way as in Equation 2-5. This concludes the formulation of RL problem. The subsequent sections will deal with two methods to solve for the solution.

$$\pi^* = \arg\max_{\pi} V^*(x) \tag{2-5}$$

2-4 Policy and value iteration

The optimal policy can be reached asymptotically by means of iteration. Let initial policy be given by $\pi_i(x,u)$. Then a new policy can be determined by first evaluating the value of π_i and recursively calculate the new policy π_{i+1}. This process can be casted as an iteration algorithm as follows.

Initialization: Start from admissible policy π_i
for *i = 1 to N* **do**
  Policy Evaluation:
  $V_{i+1}(x_t) = \rho(x_t, \pi_i(x_t)) + \gamma V_{i+1}(x_{t+1})$
  Policy Iteration:
  $\pi_{i+1}(x_t) = arg\min_{\pi}(\rho(x_t, \pi(x_t)) + \gamma V_{i+1}(x_{t+1}))$
end

Algorithm 1: Policy iteration algorithm

The prove of iteration above is provided in [14]. In order to increase computational efficiency, instead of evaluating value function V for all possible state x in every iteration, one can formulate a value function evaluation recursively. The policy iteration above is then modified as shown in Algorithm 2. It can be guaranteed that V_i will eventually converge to V^*.

Initialization: Start from admissible policy π_i
for *i = 1 to N* **do**
  Value Iteration:
  $V_{i+1}(x_t) = \rho(x_t, \pi_i(x_t)) + \gamma V_i(x_{t+1})$
  Policy Iteration:
  $\pi_{i+1}(x_t) = arg\min_{\pi}(\rho(x_t, \pi(x_t)) + \gamma V_{i+1}(x_{t+1}))$
end

Algorithm 2: Value iteration algorithm

This type of method to find solution to RL is called Dynamic Programming (DP). This method is closely related with a branch of control system, namely optimal control.

2-5 Actor Critic Methods

The second method for solving RL is by using temporal-difference learning. It is favored due to its model-free nature. In this section, we will discuss a class of temporal difference (TD) called actor-critic method. The idea of actor-critic structure is to separate policy and value function into called actor and critic entity respectively (see Figure 2-1). These actor ψ and critic θ function are parameterized by function approximators and updated using the temporal difference signal δ in every iteration. The actor-critic method is presented in Algorithm 3 (adapted from [13]). Note that $\tilde{u}$ denotes random exploration term.

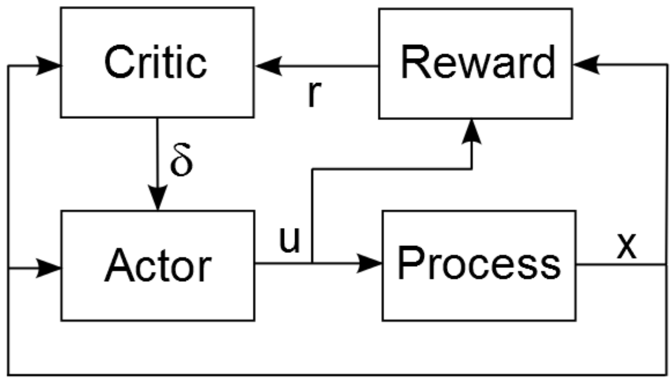

Figure 2-1: Actor critic structure

for *every trial* **do**
  Initialize x_0 and $u_0 = \tilde{u}_0$
  repeat
    apply u_k, measure x_{k+1}, receive r_{k+1}
    choose next action $u_{k+1} = \hat{\pi}(x_{k+1}, \psi_k) + \tilde{u}_{k+1}$
    $\delta_k = r_{k+1} + \hat{V}(x_{k+1}, \theta_k) - \hat{V}(x_k, \theta_k)$
    $\theta_{k+1} = \theta_k + \alpha_c \delta_k \frac{\partial \hat{V}(x,\theta)}{\partial \theta}\Big|_{x=x_k, \theta=\theta_k}$
    $\psi_{k+1} = \psi_k + \alpha_c \delta_k \frac{\partial \hat{V}(x,\psi)}{\partial \psi}\Big|_{x=x_k, \psi=\psi_k}$
  until *terminal state*;
end

Algorithm 3: Actor-critic algorithm

Chapter 3

Reinforcement Learning for Tracking Problem: A Survey

Despite the success of Reinforcement Learning (RL) in many robotics problem (e.g. learning to fly [15], walk [16] and navigate [17]), the application of RL for tracking control is not a widely explored topic. Over the spans of the literature survey, author finds several attempts to exploits RL for tracking problem, which can be categorized into 3 different approaches: dynamic tuning, RL for optimal control, and RL for nonlinear additive compensator.

This chapter covers the foundational theory of the 3 aforementioned approaches. The main idea, advantages, limitations and ease of implementation are the key issues which will be discussed. These issues will serve as the basis of the argument to choose one method for later implementation. The chapter starts in Section 3-1 by providing explanation about RL for optimal tracking control. Section 3-2 deals with the so called dynamic tuning – a class of gain scheduling which makes use of RL. The third method, presented in Section 3-3, is a relatively new approach which employs RL to learn additive input compensation.

Furthermore, author also finds it interesting to compare RL for reference tracking with a much more mature technique, namely Iterative Learning Control (ILC) [18]. As the name suggested, ILC is developed to improve tracking contaminated by repetitive error/disturbance. This method has been successfully implemented for various applications from CNC machining [19] to industrial robots [20]. Therefore, ILC will be the subject of evaluation in Section 3-4.

3-1 Reinforcement Learning for Optimal Tracking Control

This method is initiated and developed by Lewis et. al. which aims to solve the tracking by RL problem from dynamic programming perspective. The method uses optimal control, a branch of control theory whose root is closely related to dynamic programming [21]. The method starts from the downside of optimal tracking control which requires the solution of non-causal differential equation. It turns out that by assuming the reference to follow a certain dynamics and modifying the state of the optimal tracking, a causal representation can be obtained. Once a causality is in hand, we can then employ our favorite RL techniques to asymptotically solve for the solution.

To provide an easier comparison between the standard optimal control solution with RL-based one, this section starts by formulating the standard optimal tracking problem and deriving its solution. Next, the modified formulation of optimal control which allows the causal formulation of infinite horizon optimal control problem is discussed. Following is the Policy Iteration (PI) algorithms to solve the optimal control. In this section, only discrete-time Linear Quadratic Tracking (LQT) problem is considered [6]. Although the extension to non-linear and continuous time optimal tracking problem is not straightforward, the main steps are actually quite similar. The derivation presented in this section is based on work by Kiumarsi-Khomartash [6] with modifications to comply with the convention used in this report.

3-1-1 Standard LQT problem

The standard LQT problem is treated extensively in [22]. First, we formulize the linear time-invariant (LTI) discrete-time system as

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned} \tag{3-1}$$

where $x(j) \in \mathbb{R}^n$, $u(j) \in \mathbb{R}^m$ and $y(j) \in \mathbb{R}^l$ are the state, input and output at time instance j respectively. While A, B, C are the state matrices. For the sake of simplicity, we omit the feedthrough matrix D and consider a single-input single-output (SISO) system. The value of a certain state $x(k)$ and reference signal $r(k)$ can be formulized as the following infinite-horizon cost function

$$J = V(x(k), r(k)) = \frac{1}{2}\sum_{i=k}^{\infty}(Cx(i) - r(i))^T Q(Cx(i) - r(i)) + u(i)^T Ru(i) \tag{3-2}$$

where $Q \geq 0$ and $R > 0$. The goal of LQT is to obtain the optimal control input $u^*(k)$ which minimizes J. This control input is given as a combination of feedback and feedforward term

$$u(k) = -Kx(k) + K_v v(k+1) \tag{3-3}$$

where $v(k+1)$ can be obtained by solving a non-causal difference equation

$$v(k) = (A - BK)^T v(k+1) + C^T Qr(k) \tag{3-4}$$

The control gains K and K_v are

$$K = (B^T SB + R)^{-1} B^T SA \tag{3-5}$$

$$K_v = (B^T SB + R)^{-1} B^T \tag{3-6}$$

where $S = S^T > 0$ is a unique solution of the algebraic Riccati equation (ARE) as follows

$$\begin{aligned} S &= A^T S(A - BK) + C^T QC \\ &= A^T SA - A^T SB(B^T SB + R)^{-1} B^T SA + C^T QC \end{aligned} \tag{3-7}$$

Applying the optimal control input $u^*(k)$ gives us the minimal cost (optimal value) given by

$$J^* = V^*(x(k), r(k)) = \frac{1}{2} x(k)^T Sx(k) - x(k)^T v(k) + w(k) \tag{3-8}$$

where $w(k)$ is obtained from a backward recursion

$$w(k) = w(k+1) + \frac{1}{2} r(k)^T Qr(k) - \frac{1}{2} v(k+1)^T B(B^T SB + R)^{-1} B^T v(k+1) \tag{3-9}$$

Clearly, the drawback of standard optimal tracking control is the necessity to solve a non-causal difference equation (3-4). However, by assuming the reference trajectory to follow a certain dynamics, we can obtain a causal equation. This will be the subject of next subsection.

3-1-2 Causal Representation of the LQT

First, the necessary assumption is that the reference is generated by following stable difference equation

$$r(k+1) = Fr(k) \tag{3-10}$$

where F is hurwitz. By augmenting the state in (3-1), we obtain the following new state space system

$$\begin{aligned} \begin{bmatrix} x(k+1) \\ r(k+1) \end{bmatrix} &= \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & F \end{bmatrix} \begin{bmatrix} x(k) \\ r(k) \end{bmatrix} + \begin{bmatrix} B \\ \mathbf{0} \end{bmatrix} u(k) \\ X(k+1) &= TX(k) + B_1 u(k) \end{aligned} \tag{3-11}$$

Next, we assume that the candidate lyapunov function V for the augmented state space system to be

$$V(x(k), r(k)) = V(X(k)) = \frac{1}{2} X(k)^T PX(k) \tag{3-12}$$

where $P = P^T > 0$

Modifying the infinite-horizon cost function (3-2), we come up with a Bellman equation for LQT

$$\begin{aligned} V(x(k), r(k)) = &\frac{1}{2}(Cx(k) - r(k))^T Q(Cx(k) - r(k)) + u(k)^T Ru(k) + \\ &\frac{1}{2} \sum_{i=k+1}^{\infty} \Big[(Cx(i) - r(i))^T Q(Cx(i) - r(i)) + u(i)^T Ru(i) \Big] \end{aligned} \tag{3-13}$$

$$\begin{aligned}V(x(k), r(k)) =& \frac{1}{2}(Cx(k) - r(k))^T Q(Cx(k) - r(k)) + u(k)^T Ru(k) + \\ & V(x(k+1), r(k+1))\end{aligned} \tag{3-14}$$

Inserting the lyapunov equation (3-12), the LQT Bellman equation becomes

$$X(k)^T PX(k) = X(k)^T Q_1 X(k) + u(k)^T Ru(k) + X(k+1)^T PX(k+1) \tag{3-15}$$

where

$$Q_1 = \begin{bmatrix} C^T QC & -C^T Q \\ -QC & Q \end{bmatrix} \tag{3-16}$$

From the LQT Bellman equation, one can compute the time derivative (skipped here) to obtain the LQT ARE.

$$Q_1 - P + T^T PT - T^T PB_1(R + B_1^T PB_1)^{-1} B_1^T PT = 0 \tag{3-17}$$

Solving for P that satisfies (3-17), we finally obtain the optimal policy

$$u(k) = -K_1 X(k) \tag{3-18}$$

with

$$K_1 = (R + B_1^T PB_1)^{-1} B_1^T PT \tag{3-19}$$

Our next objective is to compute P of (3-17) in iterative manner using RL instead of direct computation which might be unfeasible.

3-1-3 RL for Solving the LQT ARE

In this subsection, we will employ iterative learning algorithms to solve for P. Before that, we need to derive for the lyapunov equation from the LQT Bellman equation (3-15) by inserting the optimal (3-18). This yields

$$X(k)^T PX(k) = X(k)^T Q_1 X(k) + X(k)^T K_1^T RK_1 X(k) + X(k)^T (T - B_1 K_1)^T P(T - B_1 K_1) X(k)$$

$$\Leftrightarrow P = Q_1 + K_1^T RK_1 + (T - B_1 K_1)^T P(T - B_1 K_1) \tag{3-20}$$

It turns out that by choosing a stabilizing initial policy $u^0 = -K_1^0 X(k)$, one can use policy evaluation and iteration to approximate P. In each iteration, the policy is guaranteed to be stable. The prove of this key theorem is given in [23].

By taking this theorem, one can design both offline and online PI algorithms to asymptotically approximate P. The offline PI improves P using the lyapunov function (3-20), while the LQT Bellman equation (3-15) is used for the online PI. These two algorithms are listed as follows.

Note that both require the knowledge of the system dynamics. If the model is not (fully) known, one can use Q-learning [4] or actor-critic RL [24] instead.

Initialization: Select an admissible (stable) gain K_1^0
for *j = 0 to N* **do**
  Policy evaluation:
  $P^{j+1} = Q_1 + (K_1^j)^T R K_1^j + (T - B_1 K_1^j)^T P^{j+1}(T - B_1 K_1^j)$
  Policy iteration:
  $K_1^{j+1} = (R + B_1^T P^{j+1} B_1)^{-1} B_1^T P^{j+1} T$
end

Algorithm 4: Offline Policy Iteration

Initialization: Select an admissible (stable) gain K_1^0
for *j = 0 to N* **do**
  Policy evaluation:
  $X(k)^T P^{j+1} X(k) = X(k)^T \left(Q_1 + (K_1^j)^T R K_1^j\right) X(k) + X(k+1)^T P^{j+1} X(k+1)$
  Policy iteration:
  $K_1^{j+1} = (R + B_1^T P^{j+1} B_1)^{-1} B_1^T P^{j+1} T$
end

Algorithm 5: Online Policy Iteration

3-1-4 Conclusion

We

3-2 Dynamic Tuning via Reinforcement Learning

In this second section, a class of method to improve tracking performance by dynamically tuned a controller's gain using RL is presented. To author's best knowledge, there are two prominent methods which serve this purpose. The first method is a relatively simple one which starts from an admissible controller e.g. PID, and tune the controller's gain according to the value function. The second method is a more complex approach which is based on a relatively new algorithm called Policy Improvement with Path Integral (PI^2). It is a model-free, sampling based learning method derived from the principle of optimal control [9]. This algorithm has been shown to work for a variable impedance control [10], [25], [26] to enable a manipulator performing task like flipping a light switch [25].

3-2-1 Direct Tuning of Nominal Controller

In many cases of reference tracking, a linear controller such as PID only performs well for a certain condition (e.g. a particular reference signal and a region of state) in which the gain is tuned. For different conditions, the performance is most likely degraded or even worse, unstable. Intuitively, one would call for a solution which adjusts the controller gain with respect to the current condition. This method, also known as gain scheduling, has been developed for quite some time. The most common techniques used are fuzzy logic [27] [28]

[29] and neural networks [30] [31] [32]. The main drawback of the two methods, however, lies on the scheduling mechanism which must be predefined. For instance with fuzzy logic, we need to define the fuzzy rules for the gain scheduling. For a system with a large number of states or a multi-input multi-output (MIMO) system, this could become a tedious task. For such cases, it is interesting to use RL to achieve an online gain scheduling. Surprisingly, the online gain scheduling by using RL is not a widely explored topic. [33] and [34] serve as relevant examples out of few search results.

In this literature report, author will refer to the work by Brujeni et. al. [34]. Although the paper's application is about chemical process, the technique presented is still considered suitable for robotics. The simplified block diagram of this method is shown in Figure 3-1. As the figure depicts, the idea of dynamic tuning is pretty general thus can be extended to a number of RL algorithms (e.g. actor critic, Q-learning) and controllers. However, in order to present a more concrete example, we will explain a specific method used in the paper – a class of temporal difference (TD) learning called state-action-reward-state-action (SARSA) with PID controller.

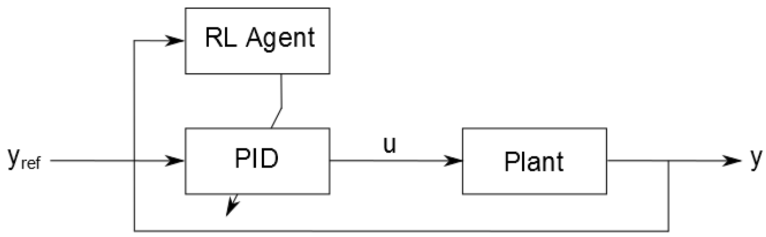

Figure 3-1: Gain scheduling of Nominal controller using RL. In this case, a PID controller is used

1. SARSA algorithm
   Consider a sequence of state and action as depicted in Figure 3-2. We start by applying control signal u_t at an initial state x_t, yielding a reward r_{t+1} and the next state x_{t+1}. Following the same policy π, we apply the next action u_{t+1}, hence the name SARSA. One of the objective is to learn the action-value function $Q^{\pi}(x, u)$ while following a fixed policy π over an episode. The pseudo-code of SARSA is given in Algorithm 6 where $\alpha, \gamma \in [0 \ldots 1]$ and r are learning rate, discount rate and immediate reward respectively [12]. Note that instead of updating Q by taking the optimal action at next step, SARSA sticks to the action resulting from the policy π (see line 8 of the algorithm). Therefore, SARSA is a type of on-policy RL algorithm.

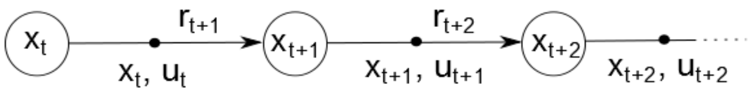

Figure 3-2: A sequence of state and action

Initialization: Initialize $Q(x, u)$ arbitrarily
repeat for each episode:
  Initialize x
  Choose u from x using policy derived from Q
  repeat for each step of episode:
    Take action u, observe r, x'
    Choose u' from x' using policy derived from Q
    $Q(x, u) \leftarrow Q(x, u) + \alpha[r + \gamma Q(x', u') - Q(x, u)]$
    $x \leftarrow x'$
    $u \leftarrow u'$
  until *s is terminal*;
until *episodes run out*;

Algorithm 6: SARSA algorithm

2. SARSA + PID controller
   To incorporate SARSA for gain scheduling purpose, we define the policy π as the gain modifier (see Figure 3-1) instead of control input generator. In other words, π will return the controller parameters e.g. K_p K_i and K_d gains for PID controller. In each iteration π will be improved by observing the most-updated value function Q. Furthermore, the performance of the RL-tuned controller needs to be evaluated in every N-steps to see if the method is actually improving the tracking performance. One possible measure for evaluation is the integral of squared errors (ISE)

$$ISE = \sum_{k=0}^{N} e(k)^2 = \sum_{k=0}^{N} (y_d(k) - y_m(k))^T (y_d(k) - y_m(k)) \tag{3-21}$$

   where y_d and y_m denotes desired and measured output respectively. It is also suggested to evaluate the value at each time step $Q(x(k), u(k))$. Now, we will proceed to a more concrete example using a PID controller for a linear discrete time system. The pseudo-code of combining SARSA with PID is presented in Algorithm 7.

3-2-2 Gain scheduling with PI^2

The second method of dynamic tuning is inspired by the sophisticated motor control of living animals. Biological motor control has shown superiority in terms of versatility and robustness to adapt to different task scenarios. Researchers have been trying to transfer the same capability to robots through variable impedance control. This task requires gain scheduling which, in one way, can be achieved by PI^2 algorithm. One of the main advantage of PI^2 is the scalability for robots with high degrees of freedom (DoF). Although variable

Initialization: Initialize Q
for *j = 1 to* $N_{episode}$ **do**
  Initialize x_0
  for $k = 0$ *to* $N_{steps} - 1$ **do**

    Compute PID gains, error, and control input
    $K_p, K_i, K_d = \pi(x(k))$
    $e(k) = y_d(k) - y_m(k)$
    $u(k) = K_p e(k) + K_i \sum_{i=0}^{k} e(k) + K_d[e(k) - e(k-1)]$

    Update state and output
    $x(k+1) = Ax(k) + Bu(k)$
    $y(k+1) = Cx(k+1) + Du(k+1)$

    Compute immediate reward
    $r(k+1) = \rho(x(k), u(k)) = \rho(x(k+1))$

    Compute PID gains, error, and control input for the next time instance
    $K_p, K_i, K_d = \pi(x(k+1))$
    $e(k+1) = y_d(k+1) - y_m(k+1)$
    $u(k+1) = K_p e(k+1) + K_i \sum_{i=0}^{k} e(k+1) + K_d[e(k+1) - e(k)]$

    Update value function
    $Q(x(k), u(k)) \leftarrow Q(x(k), u(k)) + \alpha[r(k+1) + \gamma Q(x(k+1), u(k+1)) - Q(x(k), u(k))]$

    modify π based on $Q(x(k), u(k))$
  end
end

Algorithm 7: PID gain scheduling with SARSA

impedance control is the only application of PI^2 for robotics so far [9], [10], [25], the method seems to be suitable for tracking application as well. Before moving on the motivation of such argument, we will summarize the PI^2 algorithm and its application for variable impedance control first.

Let a continuous-time (non)linear dynamics described as

$$\dot{x_t} = f(x_t) + G(x_t)(u_t + \epsilon_t) \tag{3-22}$$

where $G(x_t) \in \mathbb{R}^{n \times m}$ is the control matrix and $\epsilon_t \sim (0, \Sigma_\epsilon)$ is a zero-mean random variable. The key prerequisite before applying PI^2 is to transform the model-based stochastic optimal control problem into an approximation path integral problem. The goal of stochastic optimal control is to find an optimal input which minimizes a finite horizon cost function

$$J_{t_i} = V(X_{t_i}) = \min_{u_{t_i:t_N}} e_{\tau_i}[R(\tau_i)] \tag{3-23}$$

with

$$R(\tau_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t dt \tag{3-24}$$

where ϕ_{t_N} is the terminal reward received at time t_N and τ_i is a trajectory starts at time t_i and finishes at time t_N. The immediate reward can be formulized as

$$r_t = r(x_t, u_t) = q_t + \frac{1}{2} u_t^T R u_t \tag{3-25}$$

with $R > 0$ and $q_t = q(x_t)$ is an arbitrarily chosen function, providing a degree of freedom in specifying the cost. Next, we derive the Hamilton-Jacobi-Bellman (HJB) equation according to [35]

$$\partial_t V_t = q_t + (\partial_x V_t)^T f(x_t) - \frac{1}{2}(\partial_x V_t)^T G_t R^{-1} G_t^T (\partial_x V_t) + \frac{1}{2} trace((\partial_{xx} V_t) G_t \Sigma_\epsilon G_t^T) \tag{3-26}$$

where ∂_x and ∂_{xx} denotes jacobian and hessian respectively. Furthermore, we introduce assumptions that value function can be transformed into a logarithmic function $V_t = -\lambda \log \Psi_t$ and $\lambda G_t R^{-1} G_t^T = G_t \Sigma_\epsilon G_t^T = \Sigma(x_t) = \Sigma_t$, which give us

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + f(x_t)^T (\partial_x \Psi_t) + \frac{1}{2} trace((\partial_{xx} V_t) G_t \Sigma_\epsilon G_t^T) \tag{3-27}$$

In order to solve the so called Kolmogorov backward partial differential equation (PDE) (3-27), we need to use Feynman Kac formula which provides a numerical approximation of the solution. The detailed derivation can be seen in [36] and [37]. The solution of (3-27) becomes

$$\Psi_{t_i} = \lim_{dt \leftarrow 0} \int p(\tau_i | x_i) exp \left[-\frac{1}{\lambda} \left(\psi_{t_N} + \sum_{j=0}^{N-1} q_{t_j} dt \right) \right] d\tau_i \tag{3-28}$$

Equation (3-28) is called path integral problem. The optimal control input can be derived:

$$\begin{aligned} u_{t_i} &= \int P(\tau_i) u(\tau_i) d\tau_i \\ u(\tau_i) &= R^{-1} G_{t_i}^T (G_{t_i} R^{-1} G_{t_i}^T)^{-1} (G_{t_i} \epsilon_{t_i} - b_{t_i}) \end{aligned} \tag{3-29}$$

with $P(\tau_i)$ is the probability of trajectory τ_i and b_{t_i} is a complex notation which is explained in [37]. This concludes the problem formulation for the stochastic optimal control.

It turns out that the PI^2 algorithm can be casted into the stochastic optimal control problem with parameterized control policy expressed as follows

$$a_t = g_t^T (\theta + \epsilon_t) \tag{3-30}$$

One of the example of trajectory generator with parameterized policy is Dynamic Movement Primitive (DMP) [38]. The DMP generates desired trajectory with a point of attractor g and initial state q_0. The dynamics of DMP is given as follows

$$\frac{1}{\tau}\dot{v}_t = f_t + g_t^T(\theta + \epsilon_t) \tag{3-31}$$
$$\frac{1}{\tau}\dot{q}_{d,t} = v_t \tag{3-32}$$
$$f_t = \alpha(\beta(g - q_{d,t}) - v_t) \tag{3-33}$$
$$\frac{1}{\tau}\dot{s}_t = -\alpha s_t \tag{3-34}$$
$$[g_t]_j = \frac{w_j s_t}{\sum_{k=1}^{p} w_k}(g - q_0) \tag{3-35}$$
$$w_j = \exp(-0.5h_j(s_t - c_j)^2) \tag{3-36}$$
$$\tag{3-37}$$

The PI^2 algorithm will learn the optimal parameter θ which yields the optimal smooth trajectory so that the robot will go through a via-point g. The particular applications of such behavior are to enable robots performing task like swinging, catching, etc. The simulation and practical results presented on [9] and [10] provide an example of an intermediate goal g which the robot initially can not reach. After a number of iterations, the PI^2 algorithms finally manages to generate the optimal trajectory which enables to robot to reach g. The results also shows that PI^2 performs superior compared to standard RL algorithms. This property of PI^2 with DMP is interesting for tracking application if the points of attractor could be extended to a complete trajectory. To best of author's knowledge, there is still no paper which gives the application reference tracking. Therefore, this method is one of the possible solutions for the thesis problem.

3-3 Nonlinear Compensation for Tracking via Reinforcement Learning

This is third section.

3-4 Iterative Learning Control

This is third section.

Chapter 4

Simulation & Verification

4-1 Simulated Setup

This chapter will cover figures and math.

4-2 Simulation Result and Analysis

4-3 Discussion

Chapter 5

Future Work and Experiments Plan

Chapter 6

Conclusion

Appendix A

Appendix

Appendices are found in the back.

A-1 Simulation Program

A-1-1 A MATLAB listing

```
%
% Comment
%
n=10;
for i=1:n
    disp('Ok');
end
```

Bibliography

[1] I. F. of Robotics (IFR), "Industrial robot statistics," *World Robotics 2014 Industrial Robots*, 2014.

[2] V. Helm, J. Willmann, F. Gramazio, and M. Kohler, "In-situ robotic fabrication: Advanced digital manufacturing beyond the laboratory," *Springer Tracts in Advanced Robotics 2014*, 2014.

[3] E. Lloret, A. R. Shahabb, M. Linus, R. J. Flatt, F. Gramazio, M. Kohler, and S. Langenberg, "Complex concrete structures: Merging existing casting techniques with digital fabrication," *Computer-Aided Design*, 2014.

[4] B. Kiumarsi, F. L. Lewis, H. Modares, A. Karimpour, and M.-B. Naghibi-Sistani, "Reinforcement q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics," *Automatica*, vol. 50, no. 4, pp. 1167 – 1175, 2014.

[5] H. Modares and F. Lewis, "Online solution to the linear quadratic tracking problem of continuous-time systems using reinforcement learning," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 3851–3856, Dec 2013.

[6] B. Kiumarsi-Khomartash, F. Lewis, M.-B. Naghibi-Sistani, and A. Karimpour, "Optimal tracking control for linear discrete-time systems using reinforcement learning," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 3845–3850, Dec 2013.

[7] B. Kiumarsi and F. Lewis, "Actor-critic-based optimal tracking for partially unknown nonlinear discrete-time systems," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[8] Y. E. Bayiz and R. Babuska, "Nonlinear disturbance compensation and. reference tracking via reinforcement. learning with fuzzy approximators," *19th IFAC World Congress*, 2010.

[9] J. Buchli, E. Theodorou, F. Stulp, and S. Schaal, "Variable impedance control-a reinforcement learning approach," *Robotics: Science and Systems*, 2010.

[10] F. Stulp, J. Buchli, A. Ellmer, M. Mistry, E. Theodorou, and S. Schaal, "Reinforcement learning of impedance control in stochastic force fields," in *Development and Learning (ICDL), 2011 IEEE International Conference on*, vol. 2, pp. 1–6, Aug 2011.

[11] A. Einstein, "On the method of theoretical physics," vol. 1, pp. 163–169, Philosophy of Science, 1934.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 28. MIT press, 1998.

[13] R. Babuska, "Sc4081 knowledge-based control systems lecture slide,"

[14] D. Bertsekas, "Neuro-dynamic programming: An overview and recent results," in *Operations Research Proceedings 2006* (K.-H. Waldmann and U. Stocker, eds.), vol. 2006 of *Operations Research Proceedings*, pp. 71–72, Springer Berlin Heidelberg, 2007.

[15] A. Coates, P. Abbeel, and A. Ng, "Autonomous helicopter flight using reinforcement learning," in *Encyclopedia of Machine Learning* (C. Sammut and G. Webb, eds.), pp. 53–61, Springer US, 2010.

[16] J. Z. Kolter, P. Abbeel, and A. Y. Ng, "Hierarchical apprenticeship learning with application to quadruped locomotion," in *Advances in Neural Information Processing Systems 20* (J. Platt, D. Koller, Y. Singer, and S. Roweis, eds.), pp. 769–776, Curran Associates, Inc., 2008.

[17] S. Ross, B. Chaib-draa, and J. Pineau, "Bayesian reinforcement learning in continuous pomdps with application to robot navigation," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 2845–2851, May 2008.

[18] S. Arimoto, S. Kawamura, and F. Miyazaki, "Bettering operation of dynamic systems by learning: A new control theory for servomechanism or mechatronics systems," in *Decision and Control, 1984. The 23rd IEEE Conference on*, pp. 1064–1069, Dec 1984.

[19] D. Kim and S. Kim, "An iterative learning control method with application for cnc machine tools," in *Industry Applications Society Annual Meeting, 1993., Conference Record of the 1993 IEEE*, pp. 2106–2111 vol.3, Oct 1993.

[20] M. Norrlof and S. Gunnarsson, "Experimental comparison of some classical iterative learning control algorithms," *Robotics and Automation, IEEE Transactions on*, vol. 18, pp. 636–641, Aug 2002.

[21] R. Sutton, A. Barto, and R. J. Williams, "Reinforcement learning is direct adaptive optimal control," *Control Systems, IEEE*, vol. 12, pp. 19–22, April 1992.

[22] F. L. Lewis and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 1995.

[23] G. Hewer, "An iterative technique for the computation of the steady state gains for the discrete optimal regulator," *Automatic Control, IEEE Transactions on*, vol. 16, pp. 382–384, Aug 1971.

[24] H. Modares and F. L. Lewis, "Optimal tracking control of nonlinear partially-unknown constrained-input systems using integral reinforcement learning," *Automatica*, vol. 50, no. 7, pp. 1780 – 1792, 2014.

[25] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, 2011.

[26] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *The Journal of Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[27] J.-S. Jang and N. Gulley, "Gain scheduling based fuzzy controller design," in *Fuzzy Information Processing Society Biannual Conference, 1994. Industrial Fuzzy Control and Intelligent Systems Conference, and the NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic,*, pp. 101–105, Dec 1994.

[28] Z. Guo, J.-X. Xu, and T. H. Lee, "A gain-scheduling optimal fuzzy logic controller design for unicycle," in *Advanced Intelligent Mechatronics, 2009. AIM 2009. IEEE/ASME International Conference on*, pp. 1423–1428, July 2009.

[29] A. Hazzab, I. Bousserhane, M. Zerbo, and P. Sicard, "Real time implementation of fuzzy gain scheduling of pi controller for induction machine control.," in *Information and Communication Technologies, 2006. ICTTA '06. 2nd*, vol. 1, pp. 1416–1421, 2006.

[30] Y. Cheon, D. Lee, I.-B. Lee, and S. W. Sung, "A new pid auto-tuning strategy with operational optimization for mcfc systems," in *Control Conference (ASCC), 2013 9th Asian*, pp. 1–6, June 2013.

[31] C.-K. Chu, G.-R. Yu, E. Jonckheere, and H. Youssef, "Gain scheduling for fly-by-throttle flight control using neural networks," in *Decision and Control, 1996., Proceedings of the 35th IEEE Conference on*, vol. 2, pp. 1557–1562 vol.2, Dec 1996.

[32] J.-S. Chai, S. Tan, and C.-C. Hang, "Gain scheduling control of nonlinear plant using rbf neural network," in *Intelligent Control, 1996., Proceedings of the 1996 IEEE International Symposium on*, pp. 502–507, Sep 1996.

[33] P. Albertos and M. Olivares, "Online learning control of a gantry crane," in *Intelligent Control, 2000. Proceedings of the 2000 IEEE International Symposium on*, pp. 157–162, 2000.

[34] L. Brujeni, J. M. Lee, and S. Shah, "Dynamic tuning of pi-controllers based on model-free reinforcement learning methods," in *Control Automation and Systems (ICCAS), 2010 International Conference on*, pp. 453–458, Oct 2010.

[35] R. F. Stengel, *Optimal Control and Estimation.* Dover Publications, 1994.

[36] B. K. Oksendal, *Stochastic Differential Equations: An Introduction with Applications.* Springer, 2010.

[37] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2397–2403, May 2010.

[38] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," tech. rep., 2002.

Glossary

List of Acronyms

RL	Reinforcement Learning
MDP	Markov Decision Process
DoF	degrees of freedom
PI^2	Policy Improvement with Path Integral
PD	Proportional Derivative
PI	Policy Iteration
3D	3-dimension
DP	Dynamic Programming
TD	Temporal-Difference
LQT	Linear Quadratic Tracking
ILC	Iterative Learning Control
SISO	single-input single-output
MIMO	multi-input multi-output
LTI	linear time-invariant
ARE	algebraic Riccati equation
SARSA	state-action-reward-state-action
TD	temporal difference
ISE	integral of squared errors
HJB	Hamilton-Jacobi-Bellman

PDE partial differential equation

DMP Dynamic Movement Primitive