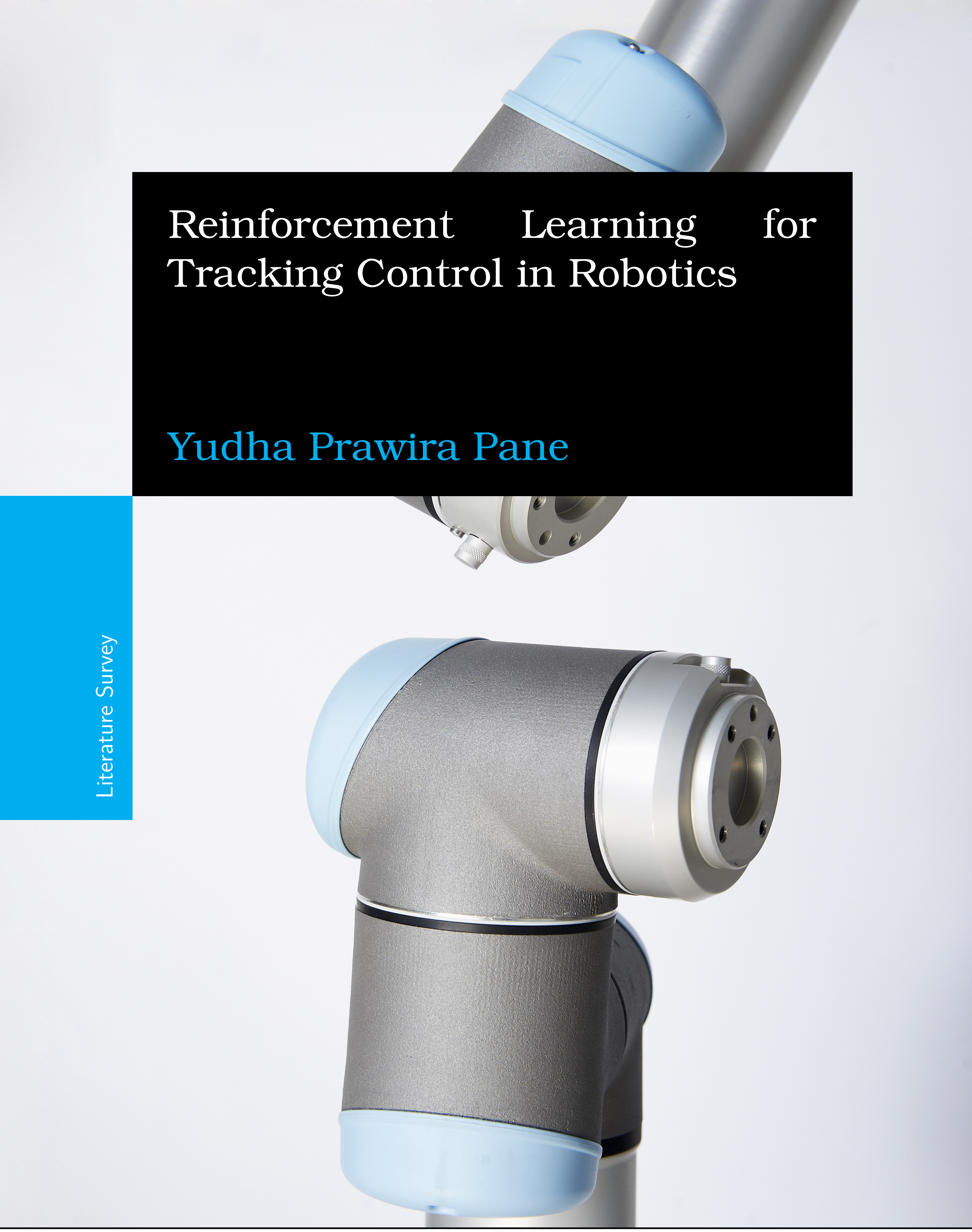


Reinforcement Learning for Tracking Control in Robotics

Yudha Prawira Pane

Literature Survey

Delft University of Technology

Delft Center for Systems and Control

Reinforcement Learning for Tracking Control in Robotics

Literature Survey

Yudha Prawira Pane

February 26, 2015

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

The implementation part of the thesis is conducted at DCSC's robotics lab. The cover is courtersy of Universal Robots

Abstract

Reference or trajectory tracking is one of the requirements in order to carry out a complex robotic task. Capability to perform a precise tracking with minimum possible error is crucial for the robots that are to be deployed at manufacturing industries such as semiconductor, automotive and recently, an emerging application of 3-dimension (3D) printing.

The approach used in the past has been to design model based controllers which involve feedback and feedforward control or more recently, a predictive control. The drawback of such scheme, however, lies on the requirements of system model as a slight model mismatch could lead to poor tracking performance. For a repetitive control error, researchers have designed the so called iterative learning control (ILC). In this literature study, a new method to optimize the tracking performance of nominal controller using reinforcement learning (RL) is proposed.

Throughout the literature study, the existing work for RL-based tracking control clusters into 3 approaches: RL for optimal tracking (Kiumarsi et al. [7]), RL-based dynamic tuning (Brujeni et al. [9]), and nonlinear compensator via RL (Bayiz et al. [12]). The advantages, limitations and practical challenges of the 3 approaches are discussed. These criterion serves as a basis to select one method which will be developed and implemented during the thesis. Furthermore, the testbed for the thesis which is a UR5 3D printing robot is also presented. Finally, the literature study concludes with the research plan and discussion.

Table of Contents

List of Figures

List of Tables

Preface

This thesis came into existence after a long wander to search for a topic which suits my vision. I have always wanted to start my own robotics company some day. So the first thing that I decided was that my master thesis should have a strong practical work. To meet this goal, I set up an appointment with Subbu in his office where he explained to me about the topics which were available at the Delft Center for Systems and Control (DCSC) robotics lab. At first, my topic was going to be "RL for underactuated robot". Sounds novel and interesting. However, in terms of feasibility, this topic might become hard since there is only one underactuated robot in the lab, the slacklining robot, and it was not in a ready-to-program state. This means that I would have to spend much time making sure the robot work before I can apply my controller. Prof. Babuška then offers me a slightly similar topic but with totally different test bed. This is when I came into RL for tracking control. This is a quite new topic in control field since only few people have conducted researches on it. After an hour of discussion, it is decided that this will be my thesis topic. The topic offers both theoretical and practical aspects since I will get a chance to apply my solution to a real robotic setup, UR5 from Universal Robots. Now that I am finishing my literature survey, I hope that I would deliver a satisfying result at the end of the thesis.

Delft, University of Technology
February 26, 2015

Yudha Prawira Pane

Acknowledgements

I would like to thank my supervisor Prof. Dr. Ir. R. Babuška for his guidance in defining my thesis topic and his valuable advices during the literature study.

I would also like to thank Subramanya Nageshrao (Subbu) for being a patient and resourceful daily supervisor. His advices help me a lot while encountering problems.

Delft, University of Technology
February 26, 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) [2] 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 would reach 205,000 units, a rise of approximately 15 % from the 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 [3] [4] 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 error could lead to a defect in 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 being a physical system is stymied by non-linearities, noises, and external disturbances that are difficult to model, let alone compensate. These unknown properties often hinder 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 with self-adjusting characteristic are called adaptive controller.

In this thesis, a method to improve the performance of nominal controller by using RL is proposed. Despite decades of extensive research on RL, its application to optimize tracking problem in robotics is still relatively unexplored. Based on the literature, there are three potential approaches to address this 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 [5] and continuous time [6], for linear [7] and non-linear system [8]. Furthermore, their technique could also be applied to Q-learning [5] and actor-critic structure [8]. The second approach uses the notion of adaptive gain scheduling which further bifurcates into two methods: directly learning the controller's gains with RL [9] and applying policy improvement with path integral (PI^2) algorithm to dynamic movement primitive (DMP) for optimizing the robot's trajectory [10] [11] . The latter was not designed for tracking in the first place, but rather to optimize gain scheduling for variable impedance control task. Nevertheless, it has some properties which makes it interesting for tracking application. Finally, the third approach is a relatively new method by using RL to learn a disturbance compensation signal for nonlinear system. This method, first proposed by Bayiz et. al. [12], would provide an additive signal to the control input. 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 a 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 [13]. Therefore, in answering this question, some simplifying assumptions are made.

1. The system to be controlled is fully actuated
2. Nominal, stabilizing controller is available
3. 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 to improve the 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 and improvement of the 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. Analyze the feasibility, advantages and limitations for all the methods found. Based on this, select one which is most suitable to answer the thesis problem.
5. Organize the research planning and future steps to be taken during the remaining thesis time.

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 markov decision process (MDP) framework, the principle of policy and value iteration and the actor-critic structure which suits the control framework. Chapter 3 provides the result of literature study being conducted, including the main ideas and brief derivation of the methods found. Chapter 4 explains the UR5 3D printing robot setup along with the result of the previous works. Finally, the last chapter deals with the selection process of the 3 methods followed by an explanation about the research plan.

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 RL. Section 2-2 explains the basics of 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

markov decision process (MDP) is defined as a tuple $\langle X, U, f, \rho \rangle$ which satisfies Markov property [14]. The detailed explanation of Markov property can be found on [1] 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
- $f : X \times U \to X$ is the state transition function (system dynamics)
- $\rho : X \times U \to \mathbb{R}$ 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 as

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

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 value function is denoted by $V^\pi(x)$ for state-value function and $Q^\pi(x, u)$ for action-value function under policy π. It can be written in terms of immediate reward and the value function of the next state by following derivation

$$\begin{aligned}
V^\pi(x_t) &= \rho(x_t, u_t) + \gamma\rho(x_{t+1}, u_{t+1}) + \gamma^2\rho(x_{t+2}, u_{t+2}) + ... + \gamma^\infty\rho(x_\infty, u_\infty) \\
V^\pi(x_t) &= \rho(x_t, u_t) + \gamma\left(\rho(x_{t+1}, u_{t+1}) + \gamma^2\rho(x_{t+2}, u_{t+2}) + ... + \gamma^\infty\rho(x_\infty, u_\infty)\right) \\
V^\pi(x_t) &= \rho(x_t, u_t) + \gamma V^\pi(x_{t+1})
\end{aligned} \tag{2-3}$$

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

$$V^*(x_t) = \rho(x,u) + \gamma \min_u V^*(x_{t+1}) \tag{2-4}$$

Similarly, the action-value function is

$$Q^*(x_t,u_t) = \rho(x,u) + \gamma \min_u Q^*(x_{t+1},u_{t+1}) \tag{2-5}$$

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

$$\pi^* = \arg\max_\pi V^*(x) \tag{2-6}$$

This concludes the formulation of RL problem. The subsequent sections will deal with two methods to solve for the solution.

2-4 Policy and value iteration

policy iteration (PI) and value iteration (VI) belong to a class of elementary solution to the RL called dynamic programming [1]. They are characterized by the requirement of system model f. These methods are closely related with a branch of control system, namely optimal control [15].

PI is a two steps algorithm, consists of policy evaluation and policy improvement. Let the initial policy at certain state x be given by π. A better policy can be determined by first evaluating the old policy's value V^π, search for the optimal action at state x greedily, and replace the old policy with the optimal action. This process can be casted as Algorithm 1. Note that the policy evaluation step is actually a Bellman equation (2-4) turned into assignment. The *policy-stable* variable is to indicate that the policy does not change anymore i.e. converged.

VI is similar to, but more efficient version than PI. In order to increase computational efficiency, instead of evaluating value function V for all possible state x in every iteration, one can evaluate the value function in greedy way, which results in less iteration. Once the value function converges to V^*, the optimal policy can be directly obtained as the control input which minimizes V. This algorithm is shown in Algorithm 2.

2-5 Actor Critic Method

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 method is to separate policy and value function into entities called actor and critic respectively (see Figure 2-1). The critic evaluates and criticizes the actor performance by feeding a temporal difference signal δ to the actor. This signal is basically the difference between right and left hand side of Bellman equation, in other words, the Bellman equation error.

Algorithm 1: Policy iteration algorithm

Initialization:
Start from an admissible policy π
Initialize $V^{\pi}(x)$ arbitrarily, e.g. $V^{\pi}(x) = 0 \;\; \forall x \in \mathcal{X}$
repeat
  Policy Evaluation:
  repeat
    $\Delta \leftarrow 0$
    For each $x \in \mathcal{X}$:
      $v \leftarrow V^{\pi}(x)$
      $V^{\pi}(x) \leftarrow \rho(x, \pi(x)) + \gamma V^{\pi}(x')$
      $\Delta = \max(\Delta, |v - V^{\pi}(x)|)$
  until $\Delta < \varepsilon$ *(a small positive number)*;
  Policy Improvement:
  policy-stable $\leftarrow$ false
  For each $x \in \mathcal{X}$:
    $b \leftarrow \pi(x)$
    $\pi(x) = \arg\min_{u} \rho(x, u) + \gamma V^{\pi}(x')$
    if $b = \pi(x)$ **then**
      policy-stable $\leftarrow$ true
    endif
until *policy-stable = true*;

Algorithm 2: Value iteration algorithm

Initialization:
Start from an admissible policy π
Initialize $V^{\pi}(x)$ arbitrarily, e.g. $V^{\pi}(x) = 0 \;\; \forall x \in \mathcal{X}$
repeat
  repeat
    $\Delta \leftarrow 0$
    For each $x \in \mathcal{X}$:
      $v \leftarrow V^{\pi}(x)$
      $V^{\pi}(x) \leftarrow \min_{u} \rho(x, \pi(x)) + \gamma V^{\pi}(x')$
      $\Delta = \max(\Delta, |v - V^{\pi}(x)|)$
  until $\Delta < \varepsilon$ *(a small positive number)*;
  Obtain a deterministic policy:
  $\pi(x) = \arg\min_{u} \rho(x, u) + \gamma V^{\pi}(x')$
until *policy converges to* π^*;

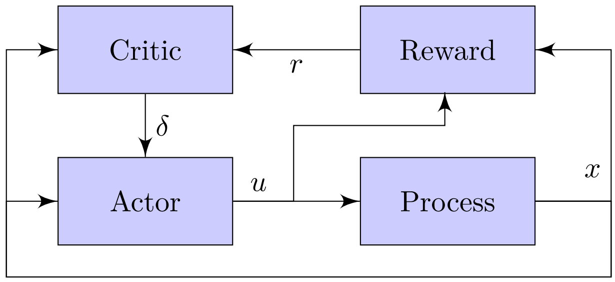


Figure 2-1: Actor critic structure (diagram reproduced from [14])

In order to deal with continuous state space, the actor ψ and critic θ functions are parameterized by function approximators. Examples of function approximators are fuzzy, neural networks and tile coding. The actor-critic method is presented in Algorithm 3 (adapted from [14]). Note that $\tilde{u}$ denotes random exploration term which is needed to avoid getting stuck at local optimum.

Algorithm 3: Actor-critic algorithm

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

Chapter 3

Reinforcement Learning for Tracking Problem: A Survey

Despite the success of RL in many robotics problem (e.g. learning to fly [16], walk [17] and navigate [18]), 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 in the next chapter. 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.

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 [15]. 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 tracking 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 tracking which allows the causal formulation of infinite horizon optimal tracking problem is discussed. Following is the PI algorithm to solve the optimal tracking. In this section, only discrete-time linear quadratic tracking (LQT) problem is considered [7]. 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 [7] 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 [19]. 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 tracking 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 tracking input $u^*(k)$ gives us the minimal cost (optimal value) given by

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

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

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

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

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

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

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

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

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

where $P = P^T > 0$

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

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

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

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 [20].

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 in Algorithm 4 and 5. We can see that for the offline PI, we can directly obtain P. Meanwhile, for the online PI, P needs to be solved with a least square method. Note that both require the knowledge of the system dynamics. If the model is not (fully) known, one can use Q-learning [5] or actor-critic RL [21] instead.

Algorithm 4: Offline Policy Iteration

Initialization: Select an admissible (stable) gain K_1^0

repeat

  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 improvement:

  $K_1^{j+1} = (R + B_1^T P^{j+1} B_1)^{-1} B_1^T P^{j+1} T$

until P *converges*;

Algorithm 5: Online Policy Iteration

Initialization: Select an admissible (stable) gain K_1^0

repeat

  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 improvement:

  $K_1^{j+1} = (R + B_1^T P^{j+1} B_1)^{-1} B_1^T P^{j+1} T$

until P *converges*;

3-1-4 RL for LQT with unknown system dynamics

As the main goal of this thesis is to improve reference tracking by compensating for the unknown dynamics using RL, the previously descibed PI method which requires full information about the system dynamics is no longer relevant. In order to relax this requirement, we will turn to TD learning. The solution to optimal tracking problem for a partially unknown system is given in [7] [8] and [21]. Although the method no longer requires drift dynamics T, input dynamics B_1 is still necessary. For a fully unknown system dynamics, a method proposed in [5] is the solution.

We start by first defining the Q function as

$$Q(X(k), u(k)) = \frac{1}{2} X(k)^T P X(k) \tag{3-21}$$

Then, multiplying LQT Bellman equation (3-15) by $\frac{1}{2}$ we obtain

$$\begin{aligned} Q(X(k), u(k)) &= \frac{1}{2} X(k)^T Q_1 X(k) + \frac{1}{2} u(k)^T R u(k) + \frac{1}{2} \gamma X^T(k+1) P X(k+1) \\ &= \frac{1}{2} X(k)^T Q_1 X(k) + \frac{1}{2} u(k)^T R u(k) + \frac{1}{2} \gamma (T X(k) + B_1 u(k))^T P (T X(k) + B_1 u(k)) \\ &= \frac{1}{2} \begin{bmatrix} X(k) \\ u(k) \end{bmatrix}^T \begin{bmatrix} Q_1 + \gamma T^T P T & \gamma T^T P B_1 \\ \gamma B_1^T P T & R + \gamma B_1^T P B_1 \end{bmatrix} \begin{bmatrix} X(k) \\ u(k) \end{bmatrix} \end{aligned} \tag{3-22}$$

Furthermore, we define the kernel matrix $H = H^T$ as

$$\begin{aligned} H &= \begin{bmatrix} Q_1 + \gamma T^T PT & \gamma T^T PB_1 \\ \gamma B_1^T PT & R + \gamma B_1^T PB_1 \end{bmatrix} \\ &= \begin{bmatrix} H_{XX} & H_{Xu} \\ H_{uX} & H_{uu} \end{bmatrix} \end{aligned} \tag{3-23}$$

The the quadratic cost function reaches minimum when $\frac{\partial Q(X(k),u(k))}{\partial u(k)} = 0$. The input which satisfies this condition is

$$u(k) = -H_{uu}^{-1} H_{uX} X(k) \tag{3-24}$$

Hence, by learning the value of H online, we can obtain the optimal tracking input u without the need of the system model. By modifying the infinite horizon cost function (3-22), we write the Q function in Bellman equation format as

$$Q(X(k), u(k)) = \frac{1}{2} X(k)^T Q_1 X(k) + \frac{1}{2} u(k)^T Ru(k) + \gamma Q(X(k+1), u(k+1)) \tag{3-25}$$

Define $Z(k) = \left[X(k)^T u(k)^T \right]^T$, the Q function can be written as

$$Q(X(k), u(k)) = \frac{1}{2} Z(k)^T HZ(k) \tag{3-26}$$

Combining equation (3-25) with (3-26) yields

$$Z(k)^T HZ(k) = X(k)^T Q_1 X(k) + u(k)^T Ru(k) + Z(k+1)^T HZ(k+1) \tag{3-27}$$

Finally, we can once again apply PI to learn matrix H online. This PI is shown in Algorithm 6. Once again, H can be calculated using a least square method after sufficient time instances.

Algorithm 6: Model-free Policy Iteration

Initialization: Select an initial admissible (stable) control input $u = -K_1^0 X_0$

repeat

  Policy evaluation:

  $Z(k)^T H^{j+1} Z(k) = X(k)^T Q_1 X(k) + (u(k)^j)^T Ru(k)^j + Z(k+1)^T HZ(k+1)$

  Policy improvement:

  $u^{j+1}(k) = -(H_{uu}^{-1})^{j+1} H_{uX}^{j+1} X(k)$

until H *converges*;

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 the best of author's knowledge, there are two prominent methods which serve this purpose. The first method is relatively simple – it starts from an admissible controller e.g. proportional-integral-derivative (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 PI^2. It is a model-free, sampling based learning method derived from the principle of optimal control [10]. This algorithm has been shown to work for a variable impedance control [11], [22], [23] to enable a manipulator performing task like flipping a light switch [22].

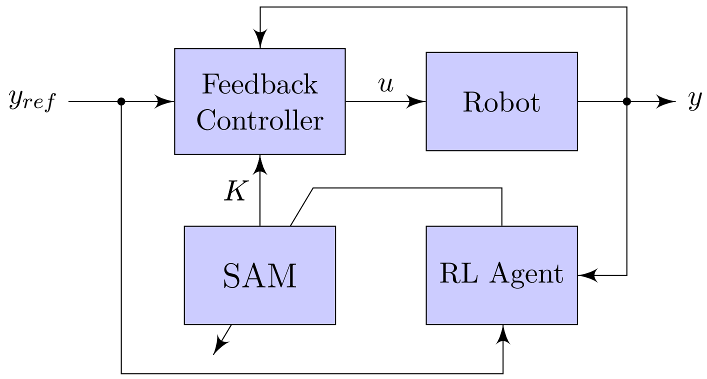


Figure 3-1: Gain scheduling of a feedback nominal controller using RL

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, the response becomes 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 [24] [25] [26] and neural networks [27] [28] [29]. 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. [30], [31] and [9] serve as relevant examples out of the search results.

In this literature report, author will refer to the work by Brujeni et. al. [9] and Howell [31]. Although the papers' applications are not related to robotics, the techniques presented are still considered relevant. The simplified block diagram of RL-based dynamic tuning is shown in Figure 3-1. The stochastic action modifier (SAM) block acts as a random gains generator which samples from the probability distribution shaped by the RL algorithm. 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 TD learning called state-action-reward-state-action (SARSA) with PID controller.

SARSA algorithm

Consider a sequence of state and action as depicted in Figure 3-2. We start by applying control signal $u(k)$ at an initial state $x(k)$, yielding a reward $r(k+1)$ and the next state $x(k+1)$. Following the same policy π, we apply the next action $u(k+1)$, 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 7 where $\alpha, \gamma \in [0 \dots 1]$ and r are learning rate, discount rate and immediate reward respectively [1]. 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.

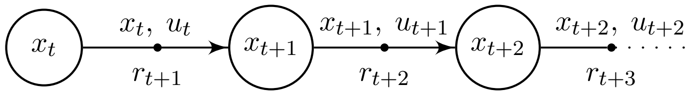


Figure 3-2: A sequence of state and action (diagram reproduced from[1])

Algorithm 7: SARSA algorithm

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*;

SARSA + PID controller

To incorporate SARSA for gain scheduling purpose, we define the policy π as the SAM modifier (see Figure 3-1) instead of the control input generator. This modification can be, for instance, in the form of the gains' probability density mean [32]. The SAM will subsequently generates 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)

$$ISE = \sum_{k=0}^{N} e(k)^2 = \sum_{k=0}^{N} (y_d(k) - y_m(k))^T (y_d(k) - y_m(k)) \tag{3-28}$$

where y_d and y_m denotes desired and measured output respectively. It is also suggested to evaluate the value at each time step $Q(x(k), u(k))$. A more concrete example of a PID controller for a linear discrete time system is presented in Algorithm 8.

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

Algorithm 8: PID gain scheduling with SARSA

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

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 DoF. Although variable impedance control is the only application of PI^2 for robotics so far [10], [11], [22], 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

$$\dot{x_t} = f(x_t) + G(x_t)(u_t + \epsilon_t) \tag{3-29}$$

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

$$J_{t_i} = V(X_{t_i}) = \min_{u_{t_i:t_N}} e_{\tau_i}[R(\tau_i)] \tag{3-30}$$

with

$$R(\tau_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t dt \tag{3-31}$$

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

$$r_t = r(x_t, u_t) = q_t + \frac{1}{2} u_t^T R u_t \tag{3-32}$$

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 [33]

$$\partial_t V_t = q_t + (\partial_x V_t)^T f(x_t) - \frac{1}{2}(\partial_x V_t)^T G_t R^{-1} G_t^T (\partial_x V_t) + \frac{1}{2} trace((\partial_{xx} V_t) G_t \Sigma_\epsilon G_t^T) \tag{3-33}$$

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

$$-\partial_t \Psi_t = -\frac{1}{\lambda} q_t \Psi_t + f(x_t)^T (\partial_x \Psi_t) + \frac{1}{2} trace((\partial_{xx} V_t) G_t \Sigma_\epsilon G_t^T) \tag{3-34}$$

In order to solve the so called Kolmogorov backward partial differential equation (PDE) (3-34), we need to use Feynman Kac formula which provides a numerical approximation of the solution. The detailed derivation can be seen in [34] and [35]. The solution of (3-34) becomes

$$\Psi_{t_i} = \lim_{dt \leftarrow 0} \int p(\tau_i | x_i) exp \left[-\frac{1}{\lambda} \left(\psi_{t_N} + \sum_{j=0}^{N-1} q_{t_j} dt \right) \right] d\tau_i \tag{3-35}$$

Equation (3-35) is called path integral problem. The optimal control input can be derived:

$$\begin{aligned} u_{t_i} &= \int P(\tau_i) u(\tau_i) d\tau_i \\ u(\tau_i) &= R^{-1} G_{t_i}^T (G_{t_i} R^{-1} G_{t_i}^T)^{-1} (G_{t_i} \epsilon_{t_i} - b_{t_i}) \end{aligned} \tag{3-36}$$

with $P(\tau_i)$ is the probability of trajectory τ_i and b_{t_i} is a complex notation which is explained in [35]. 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

$$a_t = g_t^T (\theta + \epsilon_t) \tag{3-37}$$

One of the example of trajectory generator with parameterized policy is DMP [36]. The DMP generates desired trajectory with a point of attractor g and initial state q_0. The dynamics of

DMP is given as follows

$$\frac{1}{\tau}\dot{v}_t = f_t + g_t^T(\theta + \epsilon_t) \tag{3-38}$$

$$\frac{1}{\tau}\dot{q}_{d,t} = v_t \tag{3-39}$$

$$f_t = \alpha(\beta(g - q_{d,t}) - v_t) \tag{3-40}$$

$$\frac{1}{\tau}\dot{s}_t = -\alpha s_t \tag{3-41}$$

$$[g_t]_j = \frac{w_j s_t}{\sum_{k=1}^{p} w_k}(g - q_0) \tag{3-42}$$

$$w_j = \exp(-0.5 h_j (s_t - c_j)^2) \tag{3-43}$$

$$\tag{3-44}$$

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 [10] and [11] 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 the 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 Input Compensation via Reinforcement Learning

The third method is a relatively new solution originated from DCSC. As the name suggested, the idea is to learn an additive compensator to the reference signal by means of RL. The proposed method is slightly different from the one presented in [12] in the sense that the compensator is added directly to the reference signal q_{ref} instead of the controller output u. Furthermore, we specifically choose standard actor-critic RL instead of model learning actor-critic (MLAC) since we are not interested in learning the system model online for the sake of safety. The simplified block diagram of the control scheme is shown in Figure 3-3.

3-3-1 Actor-critic formulation

First, we need to parameterize the actor and critic using local linear regression (LLR) approximation [37]. The actor and critic becomes $\pi(x_k, \vartheta_{k-1})$ and $V(x_k, \theta_k)$ respectively. As previously explained in Algorithm 3, the actor-critic will update the parameters θ and ϑ at each iteration. The fact that it uses partial derivative of value function makes actor-critic belongs to the so called policy gradient methods. Secondly, we propose the cost function as

$$r_k = \rho(y_m(k), y_d(k), u(k)) = (y_d(k) - y_m(k))^T Q (y_d(k) - y_m(k)) + u(k)^T R u(k) \tag{3-45}$$

with $Q \geq 0$ and $R > 0$ which is similar to that of LQT. The possibility for a different cost function is very likely since choosing the suitable cost function itself is not a trivial problem.

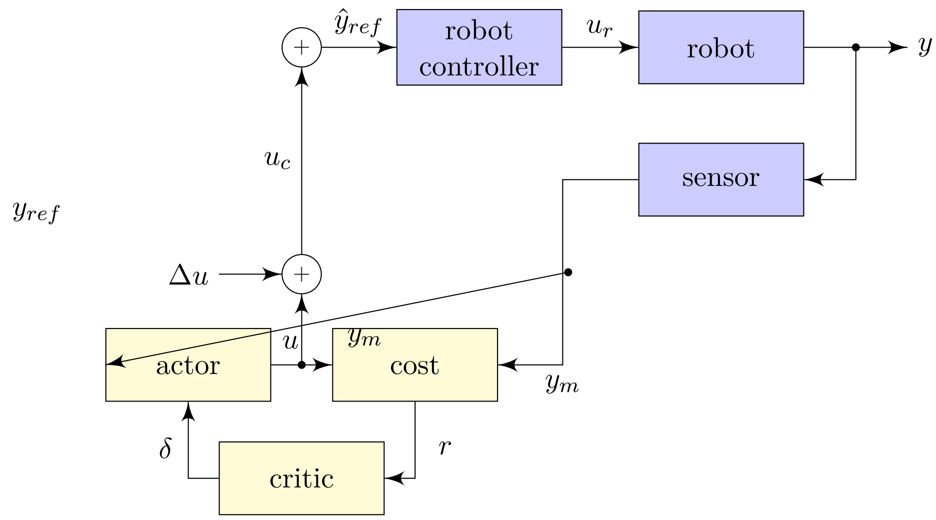


Figure 3-3: Block diagram of robot with RL block acting as an additive compensator

3-3-2 LLR Function Approximator

The function approximator is needed since we are dealing with continuous state space. Popular examples of function approximator include fuzzy [12], neural networks [8] and LLR [37]. In this literature study, we will consider the latter due to its relatively simple and intuitive algorithm compared those of, for instance, neural network. The idea of LLR is to approximate a non-linear function by predicting an output $\hat{y}_q$ to a certain query x_q through a local fitting. This local fitting is performed by means of linear regression with respect to points X which are close to x_q. The measure of distance is done by assigning weights into each point x_i in the data base.

A sorting algorithm can be employed to select K number of closest neighbors from the database. Once these neighbor samples are selected, the input and output samples are said to be related with a simple linear function

$$Y = \beta X \tag{3-46}$$

where

$$\begin{aligned} Y &= \begin{bmatrix} y_1 & y_2 & \dots & y_K \end{bmatrix} \\ X &= \begin{bmatrix} x_1 & x_2 & \dots & x_K \\ 1 & 1 & \dots & 1 \end{bmatrix} \end{aligned} \tag{3-47}$$

with the last row of X is meant for bias term. The parameter β is then computed using pseudo-inverse

$$\beta = YX^T(XX^T)^{-1} \tag{3-48}$$

Now the predicted output y_q can be calculated as

$$\hat{y}_q = \beta x_q \tag{3-49}$$

The more samples in X and Y, in other words the denser the neighborhood, the more accurate the prediction would be.

Chapter 4

Experimental Setup

The purpose of this chapter is to present the 3D printing robot system used as the testbed. The robotic system consists of a UR5 robot manipulator, 3D print head, and a laser scanner – each will be described in Section 4-1. In Section 4-2, we will discuss the previous works done on the robotic system. This includes the system identification result and the model predictive control (MPC) controller. Furthermore, based on the errors, we will also formalize the hypothesis.

4-1 The 3D Printing Robot System

4-1-1 UR5 Robot Arm

UR5 is a lightweight, flexible industrial robot from Universal Robots. The robot is widely used in academia and industry due to its human-safe operation with a quite good repeatability of 0.1 mm [38] [39]. The robot is a serial link 6 DoF manipulator with internal controller to take care of the gravity compensation and most of the non-linearities. In general, the robot can be controlled like a system of 6 decoupled servos, although this is only correct to some degree. We will see later why this is the case. Since the internal controller can not be "seen", let alone modified, the UR5 and the controller are viewed as one system.

The RL controller will be implemented in MATLAB which communicates with the UR5 using a TCP/IP protocol on 125 Hz frequency. There is a number of ways to send control command to the robot:

1. Tool position commmand
   The tool position means the position of the robot's end effector in Cartesian coordinate. The origin of the Cartesian frame is exactly at the robot's base. Refer to Figure 4-1 to see the full posture of the UR5 robot. With this type of control input, one can command the end effector to move to a specified tool position $s = (x, y, z)$ in m. This type of command results in a smooth motion if applied in a long sampling period ($>$ 1 second). For the 125 Hz communication rate, however, it suffers from a poor jitter.

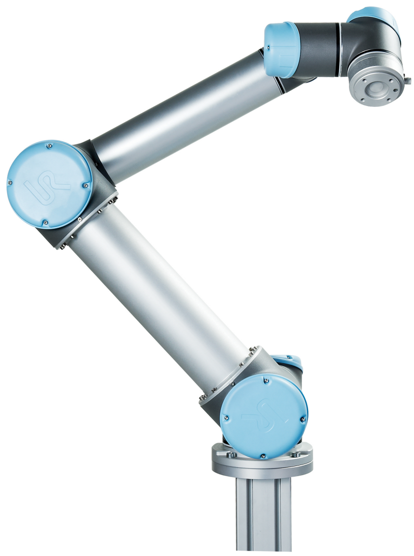


Figure 4-1: The UR5 Robot (photo courtesy of Universal Robots)

2. Tool speed command
   The tool speed command is in the same coordinate as the position command, but now the control input is the velocity of the end effector in m/s. This command results in a much smoother motion compared to position command.

3. Joint position command
   This command controls the joint position of individual robot's joint to a specified angle in radian. Similar to tool position command, it results in a jerky motion.

4. Joint speed command
   This command drives the individual joint into the desired joint velocity in the unit of radian/s. This results in the smoothest motion compared to others. Due to this reason, this command will be used to drive the robot throughout the thesis.

4-1-2 Laser Scanner

The laser scanner is a scanCONTROL 2700-25 series laser manufactured by Micro-Epsilon and mounted on the robot's end effector. It offers a 100 Hz profile frequency, up to 64,000 measuring points per second and 4 μm scanning resolution. The laser scanner is used to generate the 3D point cloud of the surface we would like to print on and to measure the tracking accuracy. Figure 4-2 shows the 3D view of the laser scanner.

4-1-3 3D Print head

The 3D print head is a Cobalt C29 inkjet printhead developed by Océ Technologies B.V. Since the print head does not contribute to the controller design and implementation, it will

Figure 4-2: scanCONTROL 2700 series laser scanner (photo courtesy of Micro-Epsilon)

not be used throughout the thesis. Figure 4-3 shows the robot's end effector with print head attached [40].

4-2 Previous Work

In this section, we will present a short overview of the previous works done by de Gier from DCSC, specifically on the system identification and the MPC controller [41].

4-2-1 System Identification

An important assumption for the identification is that the robot acts as six decoupled servos along with their internal controller. Additionally, it is also assumed that each joint is a LTI system. This implies that there will be six individuals LTI model to identify. Each model is a single-input multi-output (SIMO) system with the joint velocity reference $\dot{\theta}_r$ as the input and joint position θ and velocity $\dot{\theta}$ as the outputs.

For this purpose, a subspace identification method is used. According to the significant singular values, the order of each model is determined to be either 4 or 5 [42]. After the models are obtained, each joint model is then validated using a square input signal while the other joint angles are held constant. The variance accounted for (VAF) values of each joint for the validation test is shown in Table 4-1. Although the result shows a quite good VAF score, it is still not good enough to achieve a perfect tracking. To give an example of the performance, the model validation plot of joint 1 and 5 is shown in Figure 4-4.

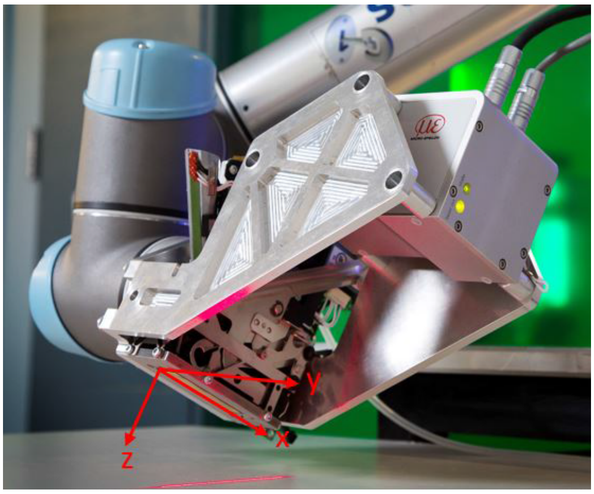
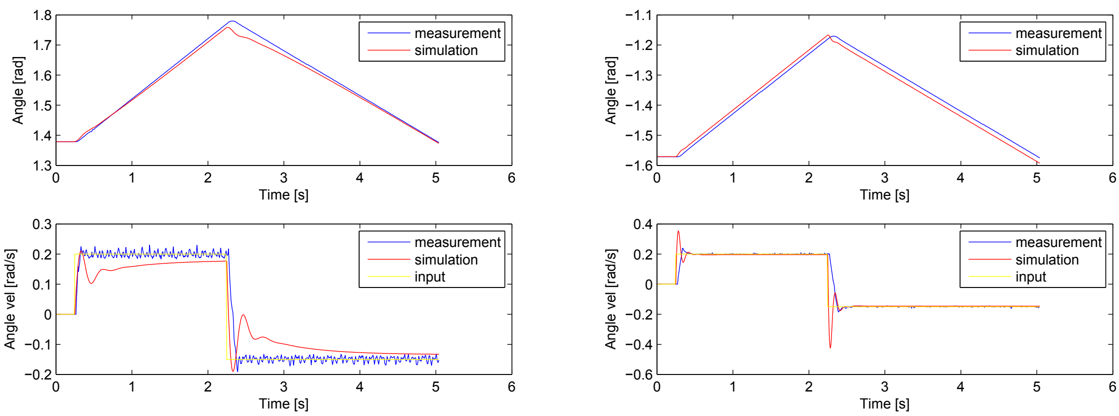


Figure 4-3: The end effector of the UR5 robot with 3D print head and laser scanner attached (photo courtesy of Sunniva Ipenburg)

Table 4-1: VAF scores of the simulated outputs for all joints

Joint	Position	Velocity
1	98.64	87.33
2	98.05	88.33
3	98.55	88.47
4	98.97	89.50
5	99.46	90.32
6	98.87	85.13



(a) **(b)**

Figure 4-4: Model validation of joint 1 and 5 for joint position and velocity

4-2-2 MPC Controller

The previous work involves a design and implementation of MPC controller. As the comparisons, we will control the robots using 3 of the 4 methods described in Subsection 4-1-1: tool position, tool velocity and joint velocity command. We will call these three control methods the default controllers. The experiment is to make the robot following a straight line along X-axis with constant values on Y and Z axes. In order to see the effect of robot's speed to the errors, the experiment is repeated with three different speeds for the default controllers. As for the MPC, only one speed is possible due to the difficulty to modify the provided MPC program. The plots for Y and Z trajectories are shown in Figure 4-5 and 4-6 respectively.

As we can see, there is a shaky behavior in both axes trajectories. To have an idea of the amplitude of the jitter, we calculate the difference between maximum and minimum values of each trajectory as shown in Table 4-2 and 4-3. The plot shows that the Y-axis trajectory of the default controllers are slowly drifting from the desired value. This is expected since the default controllers are basically open loop controllers. This causes the jitter's amplitude to be much higher than the MPC (see Table 4-2). Furthermore, for different speeds, the jitters of each default controller behave similarly. If we run the controller for several iterations at constant speed, the plot (not shown here) confirms that the jitter indeed shows a repeating behavior.

Table 4-2 has shown that the MPC controller manages to reduce the jitter significantly – it almost reaches the robot repeatability. However, for The Z-axis trajectory, the jitter is much larger. The MPC controller no longer manages to suppress the trembling motion as effective as in the Y-axis case. One of the possible factors which might introduce this behavior is the imperfect gravity compensation. Other non-linearities such as friction could also take part. Therefore we will formulate the hypothesis for the experiment as follows:

Hypothesis: *"Current controller (MPC) relies heavily on the identified model which is not perfect. Therefore, the model-mismatch induced by the unknown dynamics is responsible for the non-optimal performance of the MPC controller"*

This hypothesis justifies the use of RL-based controller.

Table 4-2: The difference between maximum and minimum values of Y-axis trajectory with different control methods

Control method	speed	jitter's amplitude (mm)
Tool Position	$\times 1$	0.635678
	$\times 2$	0.670974
	$\times 4$	0.624456
Tool Velocity	$\times 1$	0.529104
	$\times 2$	0.376090
	$\times 4$	0.320906
Joint Velocity	$\times 1$	0.550691
	$\times 2$	0.522474
	$\times 4$	0.490630
MPC	$\times 1$	0.120889

Table 4-3: The difference between maximum and minimum values of Z-axis trajectory with different control methods

Control method	speed	jitter's amplitude (mm)
Tool Position	×1	0.879843
	×2	0.805172
	×4	0.800554
Tool Velocity	×1	0.904581
	×2	0.909465
	×4	1.007834
Joint Velocity	×1	0.793960
	×2	0.795276
	×4	0.975885
MPC	×1	0.795315

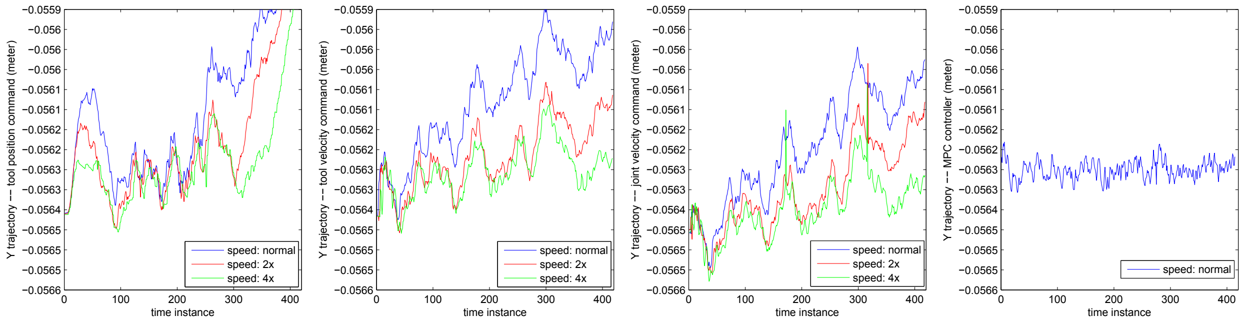


Figure 4-5: the Y trajectories of the robot with different controllers. From left to right: tool position command, tool velocity command, joint velocity command and MPC

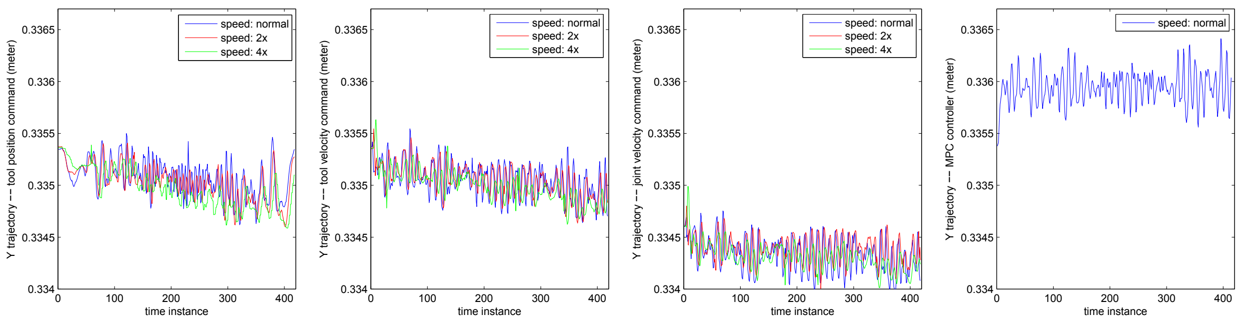


Figure 4-6: the Z trajectories of the robot with different controllers. From left to right: tool position command, tool velocity command, joint velocity command and MPC

Chapter 5

Research Direction and Discussion

This chapter presents the result of the literature study. It starts in Section 5-1 by presenting the review and analysis of the three RL approaches explained in Chapter 3. The review consists of parameters that are taken into considerations – advantages, limitations and practical challenges for implementation. Section 5-2 presents the research plan which will be carried out during the thesis. In the end of the chapter, a discussion is covered in Section 5-3

5-1 Review and Analysis of RL Approaches

5-1-1 RL for Optimal Tracking Control

From the mathematical perspective, the optimal tracking approach is more rigorous compared to the other two. The proof of convergence, for instance, has been provided in literature [20]. Another advantage is since the approach is based on lyapunov function, system's stability is guaranteed. Furthermore, researchers have successfully extended this approach to different control condition: discrete-continuous time, linear-nonlinear system, known-(partially)unknown model. From all of these benefits, the optimal tracking approach seems to be a suitable choice. However, there are some limitations which must be overcome in order to implement this method for a robotic tracking application.

As previously stated in the motivation, the goal of the thesis is to push the envelope of current tracking performance using RL. As for the case-specific, it is hypothesized in Chapter 4 that the low tracking performance is due to unknown non-linearities and disturbance in the robot. Therefore, a model-based standard optimal tracking is no longer a relevant solution. Although [5] has shown that using RL, one can obviate the need of system model for linear system, the solution for an unknown non-linear system is not yet available. This is one crucial limitation to this method for now since we actually want to improve tracking of potentially non-linear system. Even if the solution exists, a practical problem will most probably arise due to the necessity of persistence of excitation in order to achieve a converging control policy [5] [43].

For the UR5 robot testbed, this could potentially damage the robot since the motors must be excited with a persistently exciting signal such as pseudo-random noise.

The second limitation of the approach is that it is still not known how to integrate the available but inaccurate model to the optimal tracking. Literatures have shown that although the models are not perfect, it can still help in speeding up the RL convergence time [9] [37]. Therefore, model integration would be a useful, though not necessary factor. Due to these factors, optimal tracking RL is not the most suitable method for the thesis requirements.

5-1-2 Dynamic Tuning via RL

Direct Tuning of Nominal Controller

The advantage of the direct tuning by RL is its simple, intuitive scheme and rather direct implementation. However, direct tuning RL only works with a standard state/output feedback controller such as PID. State/output feedback controller is known for its nature which is always "late" since it has to wait for the error to appear and then compensate for it. Throughout the literature survey, author could not find a paper which integrates an RL-based dynamic tuning with more sophisticated controller such as optimal tracking control or MPC. Due to this bottleneck in performance, the direct-tuning is not the best choice to answer the research's goal.

Gain scheduling for DMP

As previously explained in Chapter 3, PI^2 learns the optimum trajectory to enable robotic manipulator passing through a certain intermediate point described by DMP. In order to use PI^2 DMP for reference tracking, one would need to extend the point of attractor g into a full trajectory $g(k)$. The answer of this problem, assuming that the question itself is relevant, is not present at this time.

In order to provide an answer to above problem, a thorough understanding of the PI^2 algorithm and DMP approach is needed. This is a particularly challenging problem since to derive the PI^2 itself involves a rather complex procedures such as stochastic optimal control which can easily be misunderstood if not treated carefully [10]. Due to the needs of extensive theoretical study, duration constraint of the thesis and the risk that this method is ultimately irrelevant, it is decided that PI^2 DMP is not the most suitable solution.

5-1-3 Nonlinear Input Compensation via RL

To the best of author's knowledge, there is no any literature yet which addresses the implementation of this method to a full reference tracking task. The only existing work is presented in [12] which compensates an unknown gravity for a proportional derivative (PD)-controlled 1-DoF robot arm acting on a simple step command. Thus, this method, if successfully implemented, can be considered a novel solution. This algorithm is also modular in the sense that the compensator acts as an additive signal which does not influence the controller's output. This means the better the nominal controller performs, the smaller tracking errors needs to be compensated. Furthermore, the compensator also does not require any information about the system model to operate.

The drawback of this method is the lack of mathematical formalization to prove the common control criterion: stability, convergence and robustness. From the practical point of view, some kind of mechanism must be introduced to ensure that the compensator signal is not too large since the nominal controller has already performed near optimally. Nevertheless, due to the model-free characteristic, this method is chosen to be the solution which will be developed and implemented for the rest of the thesis.

5-2 Research Plan

The post-literature survey research plan is organized as follows. Before commissioning the implementation on the UR5, a simple simulation of a 1-DoF manipulator will be programmed. The purpose of the simulation is to verify that the proposed method is actually working. If the RL-based optimizer indeed works, the next step would be to further develop the controller for implementation on the UR5 robot. If the controller performs as expected, improvement will be carried out until the end of the thesis. Parallel to these steps, the thesis report will be written as well. The research plan is summarized in a flowchart as shown in Figure 5-1.

5-3 Discussion

Since Chapter 4 has shown that for a simple straight line trajectory the jitter appears to be repetitive, author finds it interesting to compare RL for reference tracking with a much more mature technique, namely ILC [44]. 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 [45] to industrial robots [46]. Therefore, ILC could serve as a benchmark during later verification.

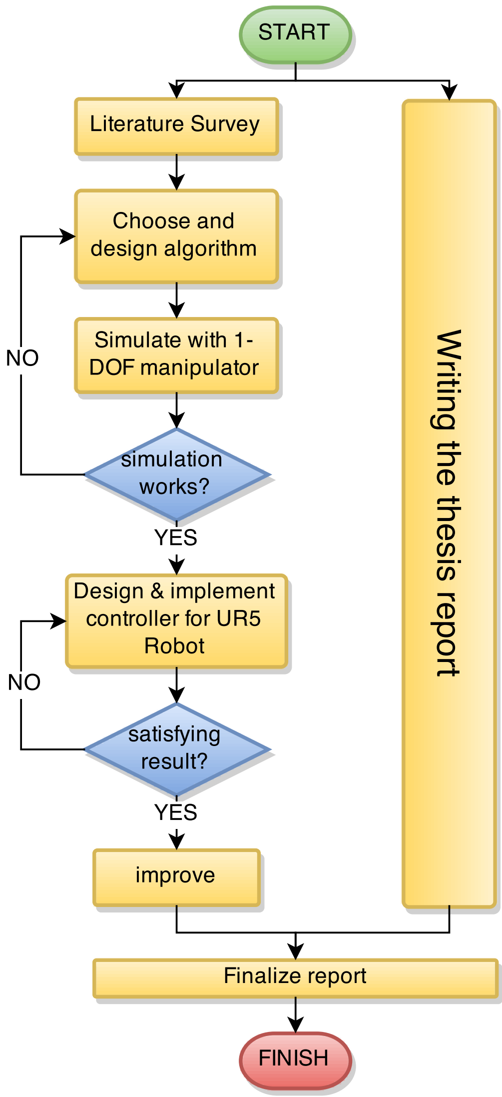


Figure 5-1: Research plan flowchart

Chapter 6

Conclusion

Despite having used for various applications for decades, the application of RL in control still has much space to explore. In this thesis, a subset of that space, namely reference tracking problem is addressed. The search for existing works results in 3 methods which must be analyzed carefully in order to obtain a strong motivation to finally choose one of them. The RL-based optimal tracking is mathematically more convincing than the other two. Not only that it will surely converge, a stability can also be guaranteeed. In the other hand, the dynamic tuning methods have problems with performance and feasibility. The direct tuning is intuitive and relatively easy to implement, but it will perform less since it depends on a feedback controller which will always be late in compensating error. The PI^2 method has successful track-record for variable impedance control task, but rather unconvincing for tracking problem. Finally, the additive tracking problem is chosen since it strongest represents the nature of RL which does not depend on the system model. This method enables an additional degree of freedom to optimize the controller, independent from the nominal controller itself. Furthermore, it is considered interesting to compare the RL-based tracking controller with a more widely known ILC.

Bibliography

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

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

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

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] M. Howell, T. Gordon, and M. Best, "The application of continuous action reinforcement learning automata to adaptive pid tuning," in *Learning Systems for Control (Ref. No. 2000/069), IEE Seminar*, pp. 2/1–2/4, 2000.

[32] M. Sedighizadeh and A. Rezazadeh, "Adaptive pid controller based on reinforcement learning for wind turbine control," in *World Academy of Science, Engineering and Technology Vol:2*, 2008.

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

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

[35] 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.

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

[37] I. Grondman, M. Vaandrager, L. Busoniu, R. Babuska, and E. Schuitema, "Efficient model learning methods for actor critic control," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, pp. 591–602, June 2012.

[38] "Ur5 technical specification." http://files.mediastorage.dk/UR_Tech_Spec/UR5_EN.pdf. Accessed: 2015-02-11.

[39] Q. Lei and M. Wisse, “Fast grasping of unknown objects using force balance optimization,” in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 2454–2460, Sept 2014.

[40] S. van Ipenburg, *On-Surface 3D Printing Method using a Robotic Arm.* Thesis Report, MSc BioMechanical Engineering, TU Delft, 2013.

[41] M. de Gier, *UR5 MPC Program.* DCSC, TU Delft, 2014.

[42] M. Verhaegen and V. Verdult, *Filtering and system identification: a least squares approach.* Cambridge university press, 2007.

[43] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, “Model-free q-learning designs for linear discrete-time zero-sum games with application to h-infinity control,” *Automatica*, vol. 43, no. 3, pp. 473 – 481, 2007.

[44] 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.

[45] 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.

[46] 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.

Glossary

List of Acronyms

3D	3-dimension
ARE	algebraic Riccati equation
DCSC	Delft Center for Systems and Control
DMP	dynamic movement primitive
DoF	degrees of freedom
HJB	Hamilton-Jacobi-Bellman
ILC	iterative learning control
ISE	integral of squared errors
LLR	local linear regression
LQT	linear quadratic tracking
LTI	linear time-invariant
MDP	markov decision process
MIMO	multi-input multi-output
MLAC	model learning actor-critic
MPC	model predictive control
PD	proportional derivative
PDE	partial differential equation
PI	policy iteration
PI^2	policy improvement with path integral

PID	proportional-integral-derivative
RL	reinforcement learning
SAM	stochastic action modifier
SARSA	state-action-reward-state-action
SIMO	single-input multi-output
SISO	single-input single-output
TD	temporal-difference
VAF	variance accounted for
VI	value iteration