Data-Efficient Reinforcement Learning Control of Robotic Lower-Limb Prosthesis With Human in the Loop

by

Xiang Gao

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2019 by the
Graduate Supervisory Committee:

Jennie Si, Chair
He Helen Huang
Marco Santello
Antonia Papandreou-Suppappola

ARIZONA STATE UNIVERSITY

May 2020

ABSTRACT

Robotic lower limb prostheses provide new opportunities to help transfemoral amputees regain mobility. However, their application is impeded by that the impedance control parameters need to be tuned and optimized manually by prosthetists for each individual user in different task environments. Reinforcement learning (RL) is capable of automatically learning from interacting with the environment. It becomes a natural candidate to replace human prosthetists to customize the control parameters. However, neither traditional RL approaches nor the popular deep RL approaches are readily suitable for learning with limited number of samples and samples with large variations. This dissertation aims to explore new RL based adaptive solutions that are data-efficient for controlling robotic prostheses.

This dissertation begins by proposing a new flexible policy iteration (FPI) framework. To improve sample efficiency, FPI can utilize either on-policy or off-policy learning strategy, can learn from either online or offline data, and can even adopt exiting knowledge of an external critic. Approximate convergence to Bellman optimal solutions are guaranteed under mild conditions. Simulation studies validated that FPI was data efficient compared to several established RL methods. Furthermore, a simplified version of FPI was implemented to learn from offline data, and then the learned policy was successfully tested for tuning the control parameters online on a human subject.

Next, the dissertation discusses RL control with information transfer (RL-IT), or knowledge-guided RL (KG-RL), which is motivated to benefit from transferring knowledge acquired from one subject to another. To explore its feasibility, knowledge was extracted from data measurements of able-bodied (AB) subjects, and transferred to guide Q-learning control for an amputee in OpenSim simulations. This result again demonstrated that data and time efficiency were improved using previous knowledge.

While the present study is new and promising, there are still many open questions to be addressed in future research. To account for human adaption, the learning control objective function may be designed to incorporate human-prosthesis performance feedback such as symmetry, user comfort level and satisfaction, and user energy consumption. To make the RL based control parameter tuning practical in real life, it should be further developed and tested in different use environments, such as from level ground walking to stair ascending or descending, and from walking to running.

DEDICATION

To my wife Jane

and my parents,

whose support, encouragement and patience

enabled me to complete this research.

ACKNOWLEDGMENTS

First, I would like to thank my advisor, Dr. Jennie Si, for taking a chance and letting me pursue this line of reinforcement learning research and for her mentorship and guidance on my research projects and manuscript preparation. I would also like to thank Dr. Helen Huang, for her introduction to the field of rehabilitation robotics and human-in-the-loop control. Also many thanks to my committee members Dr. Marco Santello and Dr. Antonia Papandreou-Suppappola for their helpful suggestions.

I would also like to thank my collaborators from NCSU, Dr. Yue Wen and Minhan Li, for their insightful discussions about my research, and conducting the human-prosthesis experiments in their lab.

I would like to give my thanks to former and current lab members Dr. Weichao Ma, Ruofan Wu, Sijia Wang, Eric Munson and Dr. Hongwei Mao for their timely assistance on my research and life, and for making these years enjoyable.

Special thanks to Dr. Glen Abousleman and Dr. Brian Skromme for their guidance and providing me opportunities to working with them.

Thanks for the financial support in part by the National Science Foundation under Grants #1563921 and #1808752.

TABLE OF CONTENTS

CHAPTER Page

[1]This chapter is based on a co-first authored paper [70] with Minhan Li.

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

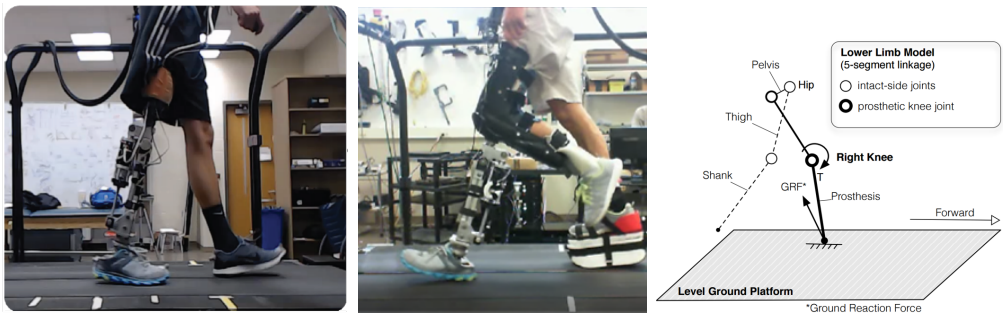
1.1 Background of Robotic Lower Limb Prosthetic Control

1.1.1 Robotic Knee Prosthesis

It is reported that there were over 600,000 lower-limb amputees lived in the US in 2005, and the number is expected to double due to the expected increase in diabetes in the coming years [1]. Amputees rely on lower limb prostheses prescribed by prosthetists to regain some function of the missing limb. These passive devices still cannot fully replicate intact leg behavior as they are incapable of providing positive net power during the gait cycle [2], which makes amputee's mobility and stability remain substantially limited [3].

Compared to traditional energy-passive prostheses, robotic lower limb prostheses can provide greater functionalities and more natural gait patterns [4, 5]. A robotic prosthesis and its wearer can be treated as a human-machine integrated system where the controller of the prosthesis may adapt to or fight against the human body. Such interaction or co-adaptation between the two adaptive systems, namely the human and the learning machine, has been rarely studied. Therefore, it is important yet challenging to design an adaptive optimal controller to improve the gait performance of the human-prosthesis system, and make it tolerant to external disturbances and environmental uncertainties.

The hardware design of the robotic knee prosthesis being used in this study is identical to [6]. It used a slider-crank mechanism, where the knee motion was driven by the rotation of the moment arm powered by the DC motor through the ball



(a) (b) (c)

Figure 1.1: Three Testing Scenarios in This Work. (a) An Amputee Subject Walking With Prosthesis; (b) An Able-Bodied Subject Walking With Prosthesis; (c) A Simulated Amputee Subject Walking with Prosthesis. Images Courtesy of NREL At NC State University.

screw. The prosthetic knee kinematics were recorded by a potentiometer embedded in the prosthesis. Some major gait events determining phase transitions in the finite state machine were detected by a load cell. The control system of the robotic knee prosthesis was implemented by LabVIEW and MATLAB on a desktop PC.

Before performing walking experiments on human subjects, we first validate our algorithm designs in a simulation platform OpenSim [7, 8]. OpenSim is a widely used simulator of human locomotion, here it is used for simulating walking patterns of the human-prosthesis system. In the OpenSim model [9], five rigid-body segments linked by one degree-of-freedom pin joints were used to represent the human body. The right knee was treated as a prosthetic knee and controlled by finite state impedance controller, while the other joints followed prescribed motions.

1.1.2 Finite State Impedance Control of Robotic Knee Prosthesis

The finite state impedance controller (FS-IC) [4, 6, 10, 11] is the most commonly used controller for lower limb prostheses. It prescribes different sets of impedance parameters to the robotic prosthesis corresponding to different gait phases. Based on the knee joint movements and ground-leg contact, a gait cycle was divided into four phases (phase $m = 1, 2, 3, 4$): stance flexion phase (STF, $m = 1$), stance extension phase (STE, $m = 2$), swing flexion (SWF, $m = 3$) and swing extension (SWE, $m = 4$). Transitions between phases were triggered by the ground reaction force (GRF), knee joint angle, and knee joint angular velocity measured from the prosthesis.

The learning controller is realized within a well established FSM platform. Specifically, an FSM partitions a gait cycle into four sequential gait phases based on knee joint kinematics and ground reaction force (GRF). These four gait phases are stance flexion (STF), stance extension (STE), swing flexion (SWF) and swing extension (SWE). In real-time experiments, transitions between phases are realized as those in [6] based on Dempster-Shafer theory (DST). For each phase, the prosthetic system mimicked a passive spring-damper-system with predefined impedance that matched the biological knee impedance. The predefined impedance parameters are selected by the finite state machine and outputted to the impedance controller as

$$I = [K, B, \theta_e]^T \in \mathbb{R}^3, \tag{1.1}$$

where K is stiffness, B is damping coefficient and θ_e is equilibrium position. In other words, for all four phases there are 12 impedance parameters to activate the knee joint which directly impact the kinematics of the robotic knee and thus the performance of the human-prosthesis system. During one of the gait phases, the knee joint torque $T \in \mathbb{R}$ is generated based on the impedance control law

$$T = K(\theta - \theta_e) + B\omega. \tag{1.2}$$

according to the three impedance parameters K, B and θ_e.

Traditionally, clinicians/prosthetists manually and heuristically tune prosthesis control parameters for an individual amputee in the clinic/laboratory by adjusting 1 or 2 parameters at a time. The clinician/prosthetist repeats these steps until the performance of the prosthetic knee reaches a desired level, which is time and labor intensive. To reduce the labor and time, researchers have tried to guide prosthetic control by measuring knee stiffness using intact leg models [12, 13, 14], the validity of which has not yet to be verified. Alternatively, researchers also tried to reduce the number of control parameters that need to be tuned [15, 16].

To overcome the disadvantages of biomechanical model-based methods, researchers also used general model-free optimization methods, such as response surface optimization [17] and cyber expert system [3], to configure wearable robot control with human in the loop. Either approach is infeasible because they do not scale well. Therefore, the prosthesis control problem calls for a design that can automatically adapt to different user and use environment.

1.1.3 Challenges in Human-in-the-loop Learning Control

- *Challenge 1: Control must ensure the stability of the prosthesis and amputee*

Improving stability and reducing the risk of falling are primary goals for designing control strategies for the human-prosthesis system. In this work, the meaning of stability is twofold. It is achieved not only by the stability of the states of the robotic system, i.e. the convergence properties of the control strategies, but also by the predictable behavior from the FS-IC based controller as they are passive within each gait phase.

- *Challenge 2: Control should learn from interacting with a human-prosthesis*

system that has perturbations and uncertainties

Unlike traditional robotic system, more variances and uncertainties are presented in a human-in-the-loop control problem, which makes learning a suitable control strategies more challenging.

- *Challenge 3: The form of the optimization goal remains an open question*

There are many performance measures for gait evaluation, including stability, gait symmetry, joint kinematics, metabolic energetic cost [18, 19] and human feedback [20]. In this work we used normative knee kinematics as an indicator of good performance because knee kinematics can provide real-time feedback for learning.

- *Challenge 4: Control must adapt to suit the needs of individual users, and adapt to changes in use environments as well*

Gait variations between individual users (inter-person variations) arise due to a number of factors including the amputee's limb-lengths, weight, strength, experience, personal preferences and etc. Gait variations within a single user (intra-person variations) arise due to fatigue, weight (e.g. with/without a backpack), walking speed and etc. Therefore the learning controller must automatically adapt to the variations in a user's gait.

- *Challenge 5: Learning control must be data and time efficient*

The learning controller should be data-based, i.e. learn from gait data directly without modeling the human-prosthesis system. To make the learning controller practical, it should be data and time efficient so that the learning process can be completed in a reasonable time frame.

1.2 Learning and Adaptive Methods for Robotic Control

1.2.1 Reinforcement Learning

To address the challenge of the adaptive and optimal control of lower limb prosthesis, reinforcement learning (RL) approach is a natural candidate. Recently, exciting new developments in deep RL, such as policy search methods and Deep Q-Network (DQN), have shed light on those challenges that were once thought nearly impossible for computers to solve. These methods found impressive success in robotics applications [21], Atari games [22] and the game of Go [23, 24]. However, the aforementioned deep RL methods are not yet applied and tested on those problems that training data is more expensive to collect. The learning controller design considered herein inherently involves continuous state and control space. Also, the learning controller needs to meet optimal performance objectives under system uncertainty, measurement noise, and unexpected disturbance. Put it together, deep learning approaches are not readily suitable for our amputee-prosthesis control problem. Approximate dynamic programming (ADP) [25, 26] is synonymous to RL especially in controls and operations research communities, and it has shown great promise to address the aforementioned challenges.

ADP is based on the idea of approximated solution to the Bellman equation of the optimal control problem. In recent years, ADP has demonstrated its scalability when dealing with large-scale and continuous state and control spaces, as well as its capability of learning from data measurements either in an online or offline manner [27]-[30]. ADP designs have demonstrated their success with many applications for continuous state and control problems [31, 32, 33, 34, 35].

In our previous work [36, 30, 37], we demonstrated the feasibility of ADP, specifically direct heuristic dynamic programming (dHDP), for personalizing powered pros-

thesis control. The dHDP is based on gradient descent, which is usually slow to converge [38]. Thus we aim to improve the time efficiency of an ADP algorithm for the application of prosthesis control optimization that involves amputee users. Similarly, an ADP training procedure should be data efficient and be limited to several minutes of amputees walking with the prosthesis during parameter tuning.

1.2.2 Knowledge Transfer in Reinforcement Learning

Most state-of-the-art RL algorithms are only good at dealing with one task. When facing a new task, the RL agents need to be retrained for the new task and the previously learned knowledge is abandoned. This greatly limits RL to achieve generalization across tasks and lowers the sample efficiency. Actually, it is very common to find such examples, that learning is performed under a new but similar environments, in our daily life. For instance, an agent who already knows how to drive a specific car is now asked to drive another bigger or smaller car, or an agent skilled in balancing the inverted pendulum now needs to deal with a shorter or longer pole. It is therefore of great interest to expand an RL agent's ability to adapt to a new and similar task, and the key challenge becomes how to learn and accumulate knowledge, and reuse such knowledge for a new task.

Transfer learning has established itself as a natural candidate for addressing this challenge, as it improves the learning in a new task through the transfer of knowledge from a related task that has already been learned [39]. Knowledge transfer in supervised learning scenarios, which also refereed as domain adaptation, has attracted much attentions [40, 41, 42]. Superiority of these approaches are verified by standard benchmarks. However, transfer in RL is still a relatively new topic and insufficiently explored [43].

Structural knowledge transfer perhaps has found most of its applications reported

in the literature. Barreto *et al.* [44] solved the problem where rewards change but environments remain the same using successor features, a value function representation that decouples the dynamics of the environment from the rewards. Asadi *et al.* [45] proposed a learning architecture which transfers control knowledge in the form of behavioral skills and representation hierarchy, which separates the subgoals so that a more compact state space can be learned. In [46], researchers demonstrated that Schema network is capable to perform zero-shot transfer between tasks where cause-effect relationship remains unchanged, such as learning to play the breakout game with different maps. In [47], target apprentice learning is proposed for cross-domain transfer, e.g. from balancing a cart-pole to balancing a bike.

To take advantage of previously learned knowledge and information, we consider building a representation for potentially transferable knowledge across subjects. We consider extracting knowledge from able-bodied (AB) subjects and use that for future RL control design for amputee subjects. It is known that transfer learning has attracted great attention in the machine learning field where it is typically considered for storing knowledge gained while solving one problem and applying it to a different but related problem [43]. In the context of general transfer learning in the literature, our prosthesis parameter tuning problem has the same state and action while the problem calls for gaining knowledge from tuning parameters for AB subjects (source task) and using that for tuning parameters for amputee subjects (target task).

1.3 Approaches and Contributions

In Chapter 2, we first propose a policy iteration based adaptive dynamic programming algorithm, namely the flexible policy iteration (FPI) algorithm, to learn the value function and the respective control policy along the trajectory of human-prosthesis gait pattern evolution. FPI can utilize either on-policy or off-policy data, as

well as the knowledge of an external critic, and guarantee nonincreasing convergence of value function to the optimal solution of the Bellman equation. We extensively evaluated the performance of our proposed algorithm in a well-established locomotion simulator, the OpenSim. Besides introducing a new and flexible reinforcement learning control approach, by utilizing this learning controller we have obtained new understanding of human-prosthesis system in terms of a quantitative description of the redundancy in the control parameter space.

In Chapter 3, we propose an offline policy iteration based reinforcement learning approach. We successfully designed a reinforcement learning controller realized by approximate policy iteration to control robotic lower limb prosthesis with human in the loop. This new prosthesis control design approach is data efficient as it was derived from offline data collected from interactions between human and prosthesis. We demonstrated this learning controller for tuning 12 prosthesis parameters to approach desired normal gait on real human subject.

In Chapter 4, we present a general reinforcement framework KG-RL that learns with both online experiences and transferred knowledge. First, structural knowledge represented as a function of state and action is extracted from a source task. Then a Q-learning based RL algorithm is presented to incorporate the knowledge in Q-value update. The convergence properties of the proposed method is analyzed. To implement KG-RL, a actor-critic structure is developed. Extensive experiments using three simulated classic RL problems verify that KG-RL outperforms its counterpart without transferred knowledge in terms of sample efficiency.

In Chapter 5, we develop a new KG-QL framework to integrate and transfer knowledge from AB subjects to OpenSim simulated amputee subjects with a common goal of optimizing impedance parameters for robotic knee prosthesis. Based on experimental measurements from two AB subjects, we established a knowledge representation

in the form of a regression model of the human-prosthesis dynamics, and a Q-value integration of this knowledge for transferring to the target task. We demonstrated the effectiveness of this KG-QL control framework. Our contribution is not limited to the demonstration of the feasibility of such transfer learning. It also includes our proposed RL control design framework that allows for flexible knowledge representation in the value function or system dynamics or both. In addition, we provided additional flexibility by allowing for a designer to determine how much information can be transferred from the source task to the target task.

Chapter 2

REINFORCEMENT LEARNING CONTROL OF ROBOTIC KNEE WITH HUMAN IN THE LOOP BY FLEXIBLE POLICY ITERATION

2.1 Abstract

This study is motivated by a new class of challenging control problems described by automatic tuning of robotic knee control parameters with human in the loop. In addition to inter-person and intra-person variances inherent in such human-robot systems, human user safety and stability, as well as data and time efficiency should also be taken into design consideration. Here by data and time efficiency we mean learning and adaptation of device configurations takes place within countable gait cycles or within minutes of time. As solutions to this problem is not readily available, we therefore propose a new policy iteration based adaptive dynamic programming algorithm, namely the flexible policy iteration (FPI). We show that the FPI solves the control parameters via (weighted) least-squares while it incorporates data flexibly and utilizes prior knowledge. We provide analyses on stable control policies, non-increasing and converging value functions to Bellman optimality, and error bounds on the iterative value functions subject to approximation errors. We extensively evaluated the performance of FPI in a well-established locomotion simulator, the OpenSim under realistic conditions. By inspecting FPI with three other comparable algorithms, we demonstrate the FPI as a feasible data and time efficient design approach for adapting the control parameters of the prosthetic knee to co-adapt with the human user who also places control on the prosthesis. As the proposed FPI algorithm does not require stringent constraints or peculiar assumptions, we expect this reinforcement

learning controller can potentially be applied to other challenging adaptive optimal control problems.

2.2 Introduction

Robotic knee is a type of wearable robot that assists individuals with lower limb amputations to regain the ability of walking. This human-robot system poses new challenges to the control of the robotic knee because of a human in the loop. Addressing these challenges needs to look beyond traditional control theory and engineering, as well as existing robotics theory and engineering.

Currently, the most advanced robotic knee control design approaches have several limitations. An intuitive idea would be to use the intact leg for the robotic knee to model after [12]. However, the validity of this approach is yet to be verified. Researchers also used response surface optimization [17] and cyber expert system [3] methods to configure wearable robot control parameters with human in the loop in order to overcome the lack of a human-robot system model. These methods are conceptually sound, however, they do not scale well for the robotic knee control design problem. It is therefore still an open question as for how to automatically configure the robotic knee control parameters. Additionally, the nature of the problem requires the control design to be data and time efficient to benefit prosthesis users.

The reinforcement learning (RL) based adaptive optimal control is naturally appealing to solve the above described challenges. As is well known, deep RL, including several policy search methods and Deep Q-Network (DQN), have shown unprecedented successes in solving difficult, sequential decision-making problems, such as those in robotics applications [21], Atari games [22], the game of Go [23, 24] and energy efficient data center [48]. Yet, it is not obvious that these successes can be extended to situations where there is no abundance of data and when the problems

involve continuous state and control variables. RL based adaptive optimal control approaches, or adaptive/approximate dynamic programming (ADP) [25, 26], is a promising alternative as they have demonstrated their capability of learning from data measurements in an online or offline manner in several realistic application problems including large-scale control problems, such as power system stability enhancement [27]-[29], and Apache helicopter control [49]-[51]. Note however, those problems do not have an explicit need of data and time efficiency during learning controller design.

At the heart of the ADP methods is the idea of providing approximating solutions to the Bellman equation of optimal control problems. In our previous work [36, 30, 37], we demonstrated the feasibility of ADP, specifically direct heuristic dynamic programming (dHDP) [52], for personalizing robotic knee control. The dHDP is an online RL algorithm based on stochastic gradient descent, which in its generic form, is not optimized for fast learning [38]. It is also worth mentioning that, the generic dHDP without imposing further conditions [53] have not shown its control law to be stable during learning. It is therefore necessary to take these limitations into design considerations especially for the current application.

The policy iteration (PI) ADP framework is potentially suitable for our applications as PI based ADP has been associated with important properties such as data efficiency [38, 54] and stable iterative control policies [55]-[58]. While the general PI based methods have improved data efficiency over stochastic gradient methods, they are still not specifically designed at the data level to be data efficient to incorporate previous data and prior knowledge in learning.

Experience replay (ER) [59] is a practically effective approach to improving sample efficiency for off-policy RL methods. In ER, past experiences (samples) generated under different behavior policies are stored in a memory buffer and selected repeatedly for evaluating the approximated value function. Advanced ER techniques such

as selective experience replay (SER) [60], prioritized experience replay (PER) [61, 62] and hindsight experience replay (HER) [63] are some of the effective ER techniques that have helped improve sample efficiency in deep RL. To prevent catastrophic forgetting, SER strategically selects which experiences will be stored. PER replays the important samples more frequently where the importance is measured by TD error. HER learns from failure by substituting the desired goal with the achieved goal and recomputing the reward function.

The ER idea has also been considered in ADP in different capacities [64]-[68]. It is shown in [64, 65] that ER can be implemented with Q-learning ADP to improve sample efficiency, yet neither of these works guarantees stable control policies or prioritizes samples. ER was also proposed in [66]-[68] to replace the persistence of excitation (PE) condition. However, the resulting sufficient condition is not practical as it requires the number of samples to be equal to the number of hidden neural network nodes, which is also a design parameter.

While there is room for efficient ADP algorithms to achieve data efficiency by innovative ER designs, prior knowledge should also be incorporated into the reinforcement learning process. This long existing idea of utilizing prior knowledge has until recently focused on specific problem domains. In a multi-agent reinforcement learning (MARL) setting [69, 70], prior knowledge such as value functions of each agent were shared to increase the learning speed. Typically, prior knowledge is represented in the form of policies [71, 72] or an initial value function [73, 74]. However, they still required expert knowledge in the process, which is difficult to interpret and encode [74]. Alternatively, previously learned value function can be used to initialize the RL algorithm. However, there is no analysis on whether the convergence of the RL algorithm is affected by such initialization.

In this paper, we propose a new, data efficient RL control method, namely the

flexible policy iteration (FPI). Compared to the existing works with ER discussed previously [64]-[68], FPI introduces a new approach that integrates the idea of prioritized sampling into policy evaluation with its solution obtained from weighted least squares. Compared to the similar works that incorporates prior knowledge [71]-[74], FPI provides a new and direct integration of a previous value into the Bellman equation. It avoids a straight forward use of previous information in the form of initial policy or initial value, the outcome of which have not been analytically assessed. Our approach instead lends itself to results with qualitative stability and convergence properties. In summary, the flexibility of FPI is demonstrated in three-folds. First, the way it collects and uses data for learning, i.e., data preparation (Table 2.1), is flexible, as it permits the agent learning from both samples generated from current policies and previous samples which are generated under different policies. Second, the way it deals with prior knowledge is flexible as it allows learning from prior knowledge in the form of an externally obtained value function using FPI from previous data collection experiments. With such a new FPI framework, we still can prove a set of qualitative properties as guidelines in the design of adaptive optimal controllers. Third, the implementation of FPI is flexible as the approximate value function can be obtained by a conventional least-square solution or by a weighted least-square solution with or without prioritized samples. All three aspects of flexibility can be customized to meet the user's needs.

This paper has three major contributions. First, we propose a new, data efficient and flexible PI method. Second, we prove the qualitative properties associated with the proposed FPI framework for its stabilizing control laws, convergence of the value function and achieving Bellman optimality approximately. Third, we provide results of applying this newly proposed FPI algorithm to an important, and also challenging problem of human-robot integration, the solution of which cannot be readily obtained

from well known control theory, control engineering or robotics engineering.

2.3 Human-Robot System

In this study, the RL controller aims at providing control torque adjustments to a robotic knee in order to help the wearer to regain mobility. We utilize a well-established finite state impedance control framework (FS-IC) which treats a gait cycle as four phases to represent different modes of stance and swing [6, 75, 76]: stance flexion phase (STF, $m = 1$), stance extension phase (STE, $m = 2$), swing flexion (SWF, $m = 3$) and swing extension (SWE, $m = 4$) (Fig. 2.2). Transitions between phases were triggered by the ground reaction force (GRF), knee joint angle, and knee joint angular velocity measured from the prosthesis. As a dynamic system, variance in a certain phase will affect the subsequent phases [77]. Fig. 2.1 shows an FS-IC based human-prosthesis system and how our proposed reinforcement learning control is integrated into the system. There are two control loops running at different frequencies. The impedance control (IC) loop generates knee joint torque T at 300 Hz following the impedance control law (2.2). In the FPI based parameter update loop, for each gait cycle k, state x_k is formed using peak knee angle P_k and gait phase duration D_k measures for each phase m as shown in Fig. 2.2.

2.3.1 Impedance Control Loop

During gait cycle k, for each FS-IC control phase m ($m = 1, 2, 3, 4$), the impedance control of the robotic knee involves three control parameters, namely stiffness $K_{m,k}$, damping coefficient $B_{m,k}$ and equilibrium position $(\theta_e)_{m,k}$. In vector form, the control parameters are represented as

$$I_{m,k} = [K_{m,k}, B_{m,k}, (\theta_e)_{m,k}]^T \in \mathbb{R}^3. \tag{2.1}$$

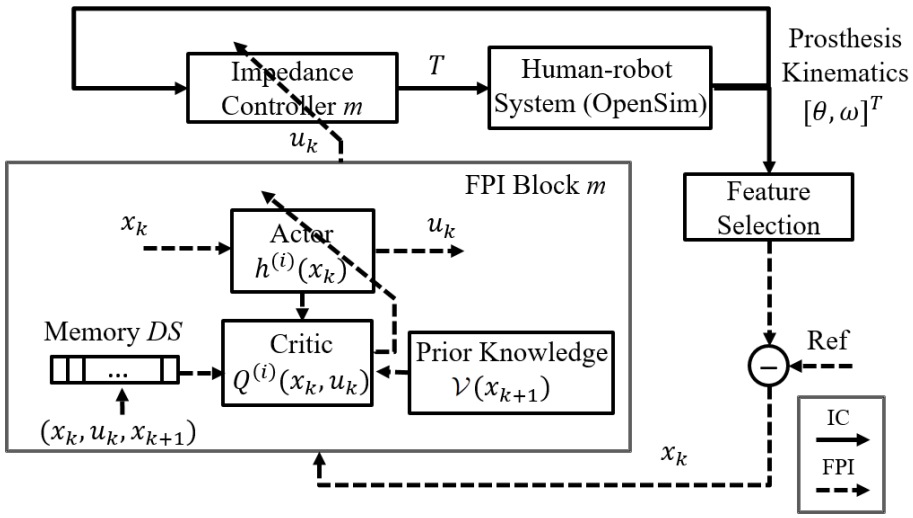


Figure 2.1: Block Diagram of the Human-Robot System With a RL Controlled Robotic Knee. The Impedance Control Loop (IC Loop) Generates Torque T According to (2.2). The FPI-Based Parameter Update Loop (FPI Loop) Adjusts Impedance Control Parameters for Each Phase m After Every Gait Cycle k. Four Identical RL Blocks ($m = 1; 2; 3; 4$) Are Needed for the Four IC Control Phases.

The prosthetic knee motor generates a knee joint torque $T \in \mathbb{R}$ from the knee joint angle θ and angular velocity ω according to the following impedance control law

$$T_k = K_k(\theta - (\theta_e)_k) + B_k\omega. \tag{2.2}$$

Without loss of generality, we drop the subscript m in the rest of the paper because all four impedance controllers and their respective FPI blocks share the same structure, although RL controller for each phase has its own parameters. The FPI controller then updates the IC parameters (2.1) for the next gait cycle $k + 1$ as

$$I_{k+1} = I_k + u_k, \tag{2.3}$$

where $u_k \in \mathbb{R}^3$ is the control output from the FPI block.

2.3.2 Parameter Update Loop by FPI

For each phase m during gait cycle k, the mth FPI controllers was enabled to update the IC parameters. After each gait cycle k, the peak knee angle $P_k \in \mathbb{R}$ and phase duration $D_k \in \mathbb{R}$ were selected by the feature selection module (Fig. 2.1). Specifically, peak knee angle P_k is the maximum or minimum knee angle in each phase, and phase duration D_k is the time interval between two consecutive peaks (Fig. 2.2). A reference trajectory of the knee joint that resembles a normal walking pattern is used in this study. Such a reference trajectory is frequently adopted in FS-IC designs [75, 78]. Subsequently we can also determine target peak angle $P_k^{'} \in \mathbb{R}$ and phase duration $D_k^{'} \in \mathbb{R}$ given the reference nominal trajectory (Fig. 2.2). For RL controller, its state variable x_k is defined using peak error $\Delta P_k \in \mathbb{R}$ and duration error $\Delta D_k \in \mathbb{R}$ as

$$x_k = [\Delta P_k, \Delta D_k]^T = [P_k - P_k^{'}, D_k - D_k^{'}]^T, \tag{2.4}$$

and its control u_k consists of increments to the IC parameters,

$$u_k = [\Delta K_k, \Delta B_k, (\Delta\theta_e)_k]^T. \tag{2.5}$$

2.4 Flexible Policy Iteration

Consider the human-robot, i.e., the amputee-prosthesis system as a discrete time nonlinear system with unknown dynamics,

$$x_{k+1} = F(x_k, u_k), k = 0, 1, 2, \ldots \tag{2.6}$$

where action u_k of the form described in (2.5) is determined according to policy h as

$$u_k = h(x_k). \tag{2.7}$$

In (2.6), the domain of $F(x_k, u_k)$ is denoted as $\mathcal{D} \triangleq \{(x, u)|x \in \mathcal{X}, u \in \mathcal{U}\}$, where $\mathcal{X}$ and $\mathcal{U}$ are compact sets with dimensions of N_x and N_u, respectively. A stage

cost function is defined in terms of x_k and u_k. In the human-robot system under consideration, F represents the kinematics of the robotic knee, which is affected by both the human wear and also the RL controller. Because of a human in-the-loop, an explicit mathematical model as (2.6) is intractable or impossible to obtain. Established biomechanical principles has provided sufficient conditions on the range of FS-IC control parameters as safety constraints on the knee joint angles and angular velocities [37]. Therefore, in our RL control designs, while the states and the controls are within the bounded set D, human subjects are guaranteed to be practically stable.

Our development of FPI requires the following assumption.

Assumption 1 The system is controllable; the system state $x_k = 0$ is an equilibrium state of system (2.6) under the control $u_k = 0$, i.e., $F(0, 0) = 0$; the feedback control $u_k = h(x_k)$ satisfies $u_k = h(x_k) = 0$ for $x_k = 0$; the stage cost function $U(x_k, u_k)$ in x_k and u_k is positive definite.

Assumption 1 is satisfied in the robotic knee control problem due to our construction of the system states and RL control (2.3) based on the biomechanics of human locomotion.

2.4.1 The Policy Iteration Framework

The RL control design objective is to derive an optimal control law via learning from observed data along the human-robot system dynamics. Consider a control policy $h(x_k)$, we define the state-action Q-value function or the total cost-to-go as

$$Q(x_k, u_k) = U(x_k, u_k) + \sum_{j=1}^{\infty} U(x_{k+j}, h(x_{k+j})). \tag{2.8}$$

Note that the $Q(x_k, u_k)$ value is a performance measure when action u_k is applied at state x_k and the control policy h is followed thereafter. It satisfies the following

Bellman equation,

$$Q(x_k, u_k) = U(x_k, u_k) + Q(x_{k+1}, h(x_{k+1})). \tag{2.9}$$

An optimal control is the one that stabilizes the system in (2.6) while minimizing the value function (2.8) according to Bellman optimality. The optimal value function is therefore of the form

$$Q^*(x_k, u_k) = U(x_k, u_k) + \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \tag{2.10}$$

or

$$h^*(x_k) = \arg\min_{u_k} Q^*(x_k, u_k), \tag{2.11}$$

$$Q^*(x_k, u_k) = U(x_k, u_k) + Q^*(x_{k+1}, h^*(x_{k+1})), \tag{2.12}$$

where $h^*(x_k)$ denotes the optimal control policy.

For our design approach to the optimal control problems, we need the control law to be admissible [55].

Definition 1 (Admissible Control) : A control policy $h(x)$ is admissible with respect to the value function $Q(x, u)$ (2.8) if $h(x)$ is continuous on $\mathcal{X}$, $h(0) = 0$ and it stabilizes system (2.6), and the corresponding value function $Q(x, u)$ (2.8) is finite for $\forall x \in \mathcal{X}$.

To assist our development of the proposed flexible policy iteration (FPI), we summarize the notation and the basic framework of a policy iteration algorithm for discrete time systems next. Consider an iterative value function $\check{Q}^{(i)}(x_k, u_k)$ and a control policy $\check{h}^{(i)}(x_k)$, the policy iteration algorithm proceeds by iterating the follow two steps:

Policy Evaluation

$$\check{Q}^{(i)}(x_k, u_k) = U(x_k, u_k) + \check{Q}^{(i)}(x_{k+1}, \check{h}^{(i)}(x_{k+1})). \tag{2.13}$$

The above policy evaluation step (2.13) is based on the Bellman equation (2.9).

Policy Improvement

$$\check{h}^{(i+1)}(x_k) = \arg\min_{u_k} \check{Q}^{(i)}(x_k, u_k), i = 0, 1, 2, \ldots \tag{2.14}$$

Motivated by the favorable properties of policy iteration in MDP problems, such as monotonically decreasing value, and demonstrated feasibility in solving realistic engineering problems [28, 29], we further develop the policy evaluation step to achieve data efficiency, easy implementation, and importantly, effectively solving realistic and complex problems.

2.4.2 Flexible Policy Iteration

We first consider a flexible use of prior information, which we expect to improve learning efficiency in data and time. Our approach entails a value function $\mathcal{V}(x_k)$ which can be obtained from an FPI solution based on past experience such as a robotic knee control experiment involving the same subject previously. Let $\mathcal{V}$ be positive definite in x_k. For $i = 0, 1, 2, \ldots$ we define a new cost-to-go $Q^{(i)}(x_k, u_k)$, which is an augmented value function constructed by $h^{(i)}(x_k)$,

$$\begin{aligned} Q^{(i)}(x_k, u_k) = & U(x_k, u_k) + \sum_{j=1}^{\infty} U(x_{k+j}, h^{(i)}(x_{k+j})) \\ & + \sum_{j=1}^{\infty} \alpha_i \mathcal{V}(x_{k+j}). \end{aligned} \tag{2.15}$$

where $0 < \alpha_{i+1} < \alpha_i < 1$, for example $\alpha_i = \gamma^i$, where $0 < \gamma < 1$. With such an augmented Q-value formulation, the policy evaluation based on the Bellman equation

(2.9) becomes:

Policy Evaluation with Augmented Information

$$Q^{(i)}(x_k, u_k) = U(x_k, u_k) + Q^{(i)}(x_{k+1}, h^{(i)}(x_{k+1})) + \alpha_i \mathcal{V}(x_{k+1}), i = 0, 1, 2, \ldots \tag{2.16}$$

Policy Improvement

$$h^{(i+1)}(x_k) = \arg\min_{u_k} Q^{(i)}(x_k, u_k), i = 0, 1, 2, \ldots \tag{2.17}$$

Remark 1. In (2.16), policy $h^{(i)}(x)$ is the policy being evaluated given the tuple (x_k, u_k, x_{k+1}), which means after control action u_k is applied at state x_k, the system reaches the next state x_{k+1}. The term $\mathcal{V}$ is a value function obtained from a previous experiment using FPI that represents prior knowledge. Note that both experiments must share the same cost function constructs.

Solving (2.16) and (2.17) to obtain closed-form optimal solutions $Q^*(x_k, u_k)$ and $h^*(x_k)$ are difficult or nearly impossible. A value function approximation (VFA) scheme replaces the exact value function in (2.16) with a function approximator such as neural networks. Such approximation based approaches to solving the Bellman equation, or RL approaches, usually utilize an actor-critic structure where the critic evaluates the performance of a control policy and the actor improves the control policy based on the critic's evaluation. Both the actor and the critic work together iteratively and learning takes place forward-in-time to approximately solve the Bellman equation.

Our next strategy to improve policy evaluation efficiency is to innovatively utilize experience replay.

2.4.3 Flexible Sampling with Experience Replay

In policy evaluation (2.16), the value function of $Q^{(i)}$ is to be evaluated with multiple samples of $s_k = (x_k, u_k, x_{k+1})$. How many samples to use and how to select the samples directly impact policy evaluation. We propose the following additional options to flexibly select the number of samples and/or prioritize the samples in order to improve policy evaluation.

Let $DS = \{s_k\}_N$ of size N be a memory buffer. When realizing experience replay without abundance of data, it would be natural to perform a policy evaluation of (2.16) using a newly available sample in conjunction with all those samples already in the memory buffer DS.

Next, samples in DS can be assigned with different priorities so that the important samples are more likely to be reused. In this work, the importance of sample s_k is measured by the TD error from a transition [61], which indicates how surprising or unexpected the transition is: specifically, how far the value is from its next-step bootstrap estimate.

Let $\delta_k^{(i)}$ be the TD error of sample s_k in DS under policy $h^{(i)}$, the rank $\zeta_k^{(i)}$ of sample s_k be obtained from sorting the memory buffer DS according to $|\delta_k^{(i)}|$ in a descending order with the largest TD error corresponding to a rank of $\zeta_k^{(i)} = 1$. Then each sample s_k is assigned a weight $\bar{\rho}_k^{(i)}$ as

$$\bar{\rho}_k^{(i)} = \frac{1}{\zeta_k^{(i)}}, \text{for } \forall k, \tag{2.18}$$

and $\bar{\rho}_k^{(i)}$ can be normalized as

$$\rho_k^{(i)} = \frac{\bar{\rho}_k^{(i)}}{\sum \bar{\rho}_k^{(i)}}, \text{for } \forall k, \tag{2.19}$$

where $0 < \rho_k^{(i)} < 1$.

2.4.4 Approximate Policy Evaluation in FPI

To implement the policy evaluation step (2.16), a function approximator $\hat{Q}^{(i)}(x_k, u_k)$ is needed for $Q^{(i)}(x_k, u_k)$. Here we use a linear-in-parameter function approximation structure which can readily deal with the prioritized samples described in the previous subsection:

$$\hat{Q}^{(i)}(x_k, u_k) = W^{(i)T}\phi(x_k, u_k) = \sum_{k=1}^{L} w_k^{(i)} \varphi_k(x_k, u_k) \tag{2.20}$$

where $W^{(i)} \in \mathbb{R}^L$ is a weight vector and $\phi(x_k, u_k) : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \to \mathbb{R}^L$ is a vector of the basis functions $\varphi_k(x_k, u_k), k = 1 \dots L$. The basis functions $\varphi_k(x_k, u_k)$ can be neural networks, polynomial functions, radial basis functions, etc.

The policy evaluation step (2.16) then becomes

$$\begin{aligned} &\hat{Q}^{(i)}(x_k, u_k) \\ &= U(x_k, u_k) + \hat{Q}^{(i)}(x_{k+1}, h^{(i)}(x_{k+1})) + \alpha_i \mathcal{V}(x_{k+1}). \end{aligned} \tag{2.21}$$

Substituting (2.20) into (2.21), we have

$$\begin{aligned} &[\phi(x_k, u_k) - \phi(x_{k+1}, h^{(i)}(x_{k+1}))]W^{(i)} \\ &\qquad = U(x_k, u_k) + \alpha_i \mathcal{V}(x_{k+1}). \end{aligned} \tag{2.22}$$

Equation (2.21) can be seen as an approximated policy evaluation step in terms of a weight vector that is to be determined from solving L linear equations. At iteration i, two column vectors $X^{(i)} \in \mathbb{R}^{N \times L}$ and $Y^{(i)} \in \mathbb{R}^N$, are formed by the term $\phi(x_k, u_k) - \phi(x_{k+1}, h^{(i)}(x_{k+1}))$ and $U(x_k, u_k) + \alpha_i \mathcal{V}(x_{k+1})$, respectively, in each row. In other words, (2.22) can be rewritten as

$$X^{(i)} W^{(i)} = Y^{(i)}. \tag{2.23}$$

The TD error $\delta_k^{(i)}$ can be computed as

$$\begin{aligned}\delta_k^{(i)} =& U(x_k, u_k) + \hat{Q}^{(i-1)}(x_{k+1}, h^{(i)}(x_{k+1})) + \alpha_{i-1}\mathcal{V}(x_{k+1}) \\ & - \hat{Q}^{(i-1)}(x_k, u_k), \text{for } i = 0, 1, 2, \ldots\end{aligned} \tag{2.24}$$

Then the weight $\rho_k^{(i)}$ of the samples can be obtained from (2.19). For $i = 0$, equal weights $\rho_k^{(0)} = 1$ will be assigned to all samples in DS. When the policy evaluation with function approximation (2.21) is carried out with sample $s_k = (x_k, u_k, x_{k+1})$, it can be weighted by $\rho_k^{(i)}$. Hence, the weight vector $W^{(i)}$ can be computed from (2.23) as a weighted least squares solution using N weighted samples

$$W^{(i)} = (X^{(i)^T}\Psi^{(i)}X^{(i)})^{\dagger}(X^{(i)^T}\Psi^{(i)}Y^{(i)})^T, \tag{2.25}$$

where $\Psi^{(i)} \in \mathbb{R}^N$ is a vector of $\rho_k^{(i)}$. Once $W^{(i)}$ is obtained, the approximated value function $\hat{Q}^{(i)}(x_k, u_k)$ can be obtained using (2.20).

Table 2.1: Data Preparation and Parameter Settings in Algorithm 1

Setting		Description
1	(A) N_b is fixed	Fixed
	(B) $N_b \leftarrow N_b + 5$	Adaptive
2	(A) $N \leftarrow N_b$	Batch mode
	(B) $N \leftarrow N + 1$	Incremental mode
3	(A) $\rho_k^{(i)} = 1$	No prioritization
	(B) $\rho_k^{(i)}$ from (2.19)	With prioritization
4	(A) $\alpha_i = 0$	No prior knowledge
	(B) $\alpha_i = 0.9^i$	With prior knowledge

Algorithm 2.1 Flexible Policy Iteration (FPI)

Initialization by

Random initial state $x_0 \in \mathcal{X}$, initial batch size N_b (if in batch mode), memory buffer $DS = \emptyset$, initially admissible control policy $h^{(0)}$.

Data Preparation

1a: (**Batch Data Collection**) Collect N_b samples $\{(x_k, u_k, x_{k+1})\}_{N_b}$ from system (2.6) following policy $\hat{h}^{(i)}$at gait cycle k, $N \leftarrow N_b$ (Setting 2(A) in Table 2.1).

1b: (**Incremental Data Collection**) Collect a sample (x_k, u_k, x_{k+1}) from system (2.6) following policy $\hat{h}^{(i)}$, and add it to DS, $N \leftarrow N + 1$ (Setting 2(B) in Table 2.1).

2: (**Set Batch Size**) Either use a fixed or adaptive N_b (Setting 1 in Table 2.1) if under batch mode (Setting 2(A) in Table 2.1).

3: (**Set Other Parameters**) Set $\rho_k^{(i)}$ (Setting 3 in Table 2.1) and α_i (Setting 4 in Table 2.1).

Policy Evaluation/Update for Iteration i

4: (**Policy Evaluation**) Evaluate policy $\hat{h}^{(i)}$ by solving (2.21) for $\hat{Q}^{(i)}$ using all samples in DS.

5: (**Policy Update**) Update policy $\hat{h}^{(i+1)}$ by (2.27) and (2.28).

2.4.5 Policy Improvement in FPI

After the approximated value function $\hat{Q}^{(i)}(x_k, u_k)$ is obtained, we can get the next policy $h^{(i+1)}(x_k)$ from (2.17) during policy improvement,

$$h^{(i+1)}(x_k) = \arg\min_{u_k} \hat{Q}^{(i)}(x_k, u_k). \tag{2.26}$$

We employ another linear-in-parameter function approximator $\hat{h}^{(i+1)}(x_k)$ for $h^{(i+1)}(x_k)$,

$$\hat{h}^{(i+1)}(x_k) = (\mathcal{K}^{(i+1)})^T \sigma(x_k), \tag{2.27}$$

where $\mathcal{K}^{(i+1)}$ is a weight vector and $\sigma(x_k)$ is a basis function vector. The weight vector $\mathcal{K}^{(i+1)}$ is updated iteratively using the gradient of the approximate value function $\hat{Q}^{(i)}(x_k, u_k)$,

$$\mathcal{K}_{j+1}^{(i+1)} = \mathcal{K}_j^{(i+1)} - l\frac{\partial \hat{Q}^{(i)}(x_k, (\mathcal{K}_j^{(i+1)})^T \sigma(x_k))}{\partial \mathcal{K}_j^{(i+1)}} \tag{2.28}$$

where l is the learning rate ($0 < l < 1$), the tuning index j is used for the policy update within a policy evaluation step.

2.4.6 Implementation of FPI

Algorithm 2.1 and Table 2.1 together describe our proposed FPI algorithm. The terminating condition in Algorithm 2.1 can be, for example, policy iteration index $i = i_{max}$ where i_{max} is some positive number, or $|\hat{Q}^{(i)}(x_k, u_k) - \hat{Q}^{(i-1)}(x_k, u_k)| < \varepsilon$ where ε is a small positive number. Note that there are four settings in Algorithm 1 (Table 2.1). FPI can run in batch mode or incremental mode (Setting 2). In batch mode, only samples (of length N_b) generated under the same policy are used in policy evaluation, thus no sample reuse is allowed in this mode. In incremental mode, previous samples that are generated under different policies can be reused to evaluate a new policy. In batch mode, an extra parameter batch size N_b need to be set (Setting 1 in Table 2.1), while such parameter is not required under incremental mode. In addition, Setting 3 describes how the priorities $\rho_k^{(i)}$ of the samples are assigned and Setting 4 describes how the prior knowledge is used at iteration i through the parameter of α_i .

Note that in batch mode, FPI can choose the number of samples for policy evaluation adaptively. FPI starts with a small N_b. A newly generated policy is tested with one or more gait cycles to determine if the policy can lower the stage cost. If not, a larger set of samples (e.g. $N_b \leftarrow N_b + 5$) is used.

This adaptive approach is based on our observations as follows. Given a continuous state and control problem such as the control of a robotic knee, we constructed a quadratic stage cost (x_k, u_k) in (2.51) which is common in control system design. As a decreasing stage cost can be viewed as necessary toward an improved value during each iteration, it thus becomes a natural choice for such a selection criterion. For example, Fig. 2.3 depicts stage cost for the uniformly sampled IC parameter space in our human-robot application, where the color of each sample point represents a stage cost. Fig. 2.4 was generated under the setting of (A)(A)(A)(A) in Table 2.1 and $N_b = 20$. Fig. 2.4 shows the trajectories of the IC parameters tuned by FPI starting from some random initial IC parameters. Apparently, the points with minimum stage cost in Fig. 2.3 coincides with the converging planes found by FPI in Fig. 2.4.

2.5 Qualitative Properties of FPI

For discrete-time nonlinear systems, policy iteration based RL has several important properties, such as stability, monotonicity of value function, and approaching approximate Bellman optimality [55, 58, 79]. As mentioned before, we introduce a value function term $\mathcal{V}(x_k)$ to capture prior knowledge. Specifically, we let $\mathcal{V}(x_k) = \min_{u_k} Q^*(x_k, u_k)$, where $Q^*(x_k, u_k)$ is a final converged value function obtained by applying FPI (Algorithm 1) in a previous experiment. Here we will show that, unlike previous results that demonstrated empirically the effect of utilizing prior knowledge, our new means of integrating prior knowledge $\mathcal{V}$ into a policy iteration framework allows us to obtain important stability and optimality related qualitative properties of FPI.

Lemma 1. *Let $i = 0, 1, ...$ be the iteration number and let $Q^{(i)}(x_k, u_k)$ and $h^{(i)}(x_k)$ be updated by* (2.16)-(2.17)*. Under Assumption 1, the iterative value function $Q^{(i)}(x_k, u_k)$, $i = 0, 1, ...$, is positive definite for x_k and u_k.*

Proof. For $i = 0$, according to Assumption 1, we have $h^{(0)}(x_k) = 0$ as $x_k = 0$. As $U(x_k, u_k)$ is positive definite for x_k and u_k, we have that $\sum_{j=0}^{\infty} U(x_{k+j}, h^{(0)}(x_{k+j})) = 0$ as $x_k = 0$, and $\sum_{j=0}^{\infty} U(x_{k+j}, h^{(0)}(x_{k+j})) > 0$ for any $x_k \neq 0$. Hence $\sum_{j=0}^{\infty} U(x_{k+j}, h^{(0)}(x_{k+j}))$ is a positive definite function for x_k. Since $\mathcal{V}(x_k)$ is also positive definite for x_k, according to (2.15), if $x_k = u_k = 0$, $Q^{(0)}(x_k, u_k) = 0$; if $|x_k| + |u_k| \neq 0$, $Q^{(0)}(x_k, u_k) > 0$, which proves that $Q^{(0)}(x_k, u_k)$ is positive definite for x_k and u_k. Based on this idea, we can prove that the iterative function $Q^{(i)}(x_k, u_k), i = 0, 1, ...,$ is positive definite for x_k and u_k.

Theorem 1. *Let Assumption 1 hold. Let $Q^{(i)}(x_k, u_k)$ and $h^{(i)}$ be updated by* (2.16)-(2.17)*, where $h^{(0)}$ is an admissible control policy. Then, for $i = 0, 1, 2, ...,$ $h^{(i)}$ stabilizes the system* (2.6)*.*

Proof. Consider the case when $x_k \neq 0$, we have $U(x_k, h^{(i)}(x_k)) > 0$ and $\alpha_i \mathcal{V}(x_{k+1}) \geq 0$. From (2.16), and $i = 0, 1, \ldots,$we can get

$$\begin{aligned} &Q^{(i)}(x_k, h^{(i)}(x_k)) - Q^{(i)}(x_{k+1}, h^{(i)}(x_{k+1})) \\ &= U(x_k, h^{(i)}(x_k)) + \alpha_i \mathcal{V}(x_{k+1}) > 0. \end{aligned} \tag{2.29}$$

Next, consider the case when $x_k = 0$, according to Assumption 1 we can get $h^{(i)}(x_k) = 0$ and $x_{k+1} = F(x_k, h^{(i)}(x_k)) = F(0, 0) = 0$. Hence we get $U(x_k, h^{(i)}(x_k)) = 0$ and $\alpha_i \mathcal{V}(x_{k+1}) = 0$, which imply $Q^{(i)}(x_k, h^{(i)}(x_k)) - Q^{(i)}(x_{k+1}, h^{(i)}(x_{k+1})) = 0$. According to Lemma 1 and Assumption 1, the function $Q^{(i)}(x_k, h^{(i)}(x_k))$ is positive definite for x_k. Then $Q^{(i)}(x_k, h^{(i)}(x_k))$ is a Lyapunov function. Thus $h^{(i)}$ stabilizes the system (2.6).

Remark 2. Theorem 1 shows that the Lyapunov stability can be guaranteed under iterative policy $h^{(i)}(x_k)$ under the augmented value evaluation of (2.15). Based on established physiological knowledge of human walking and the biomechanics of knee joints, we also are able to embed human practical stability into our RL controller design.

Theorem 2. *Let the value function $Q^{(i)}(x_k, u_k)$ and the control policy $h^{(i)}(x_k)$ be obtained from* (2.16) *and* (2.17)*, respectively. Then $Q^{(i+1)}(x_k, u_k) \leq Q^{(i)}(x_k, u_k)$ holds for $i = 0, 1, 2, \ldots$ and $\forall (x_k, u_k) \in \mathcal{D}$.*

Proof. For convenience, we will use the following short hand notations in the derivations, e.g. $U(x_k, h^{(i)})$ for $U(x_k, h^{(i)}(x_k))$. According to (2.15), we can define $V(x_k)$ as

$$V^{(i)}(x_k) = Q^{(i)}(x_k, h^{(i)}) = \sum_{j=k}^{\infty} U(x_j, h^{(i)}) + \alpha_i \sum_{j=k}^{\infty} \mathcal{V}(x_{j+1}). \tag{2.30}$$

Based on (2.17), we have

$$\begin{aligned} Q^{(i)}(x_k, h^{(i+1)}) &= \min_{u_k} Q^{(i)}(x_k, u_k) \\ &\leq Q^{(i)}(x_k, h^{(i)}). \end{aligned} \tag{2.31}$$

Based on (2.16) we have

$$\begin{aligned} V^{(i)}(x_k) &= Q^{(i)}(x_k, h^{(i)}) \\ &\geq Q^{(i)}(x_k, h^{(i+1)}) \\ &= U(x_k, h^{(i+1)}) + V^{(i)}(x_{k+1}) + \alpha_i \mathcal{V}(x_{k+1}) \\ &\geq U(x_k, h^{(i+1)}) + V^{(i)}(x_{k+1}) + \alpha_{i+1} \mathcal{V}(x_{k+1}). \end{aligned} \tag{2.32}$$

Hence

$$\begin{aligned} V^{(i)}(x_k) - V^{(i)}(x_{k+1}) &\geq U(x_k, h^{(i+1)}) + \alpha_{i+1} \mathcal{V}(x_{k+1}) \\ V^{(i)}(x_{k+1}) - V^{(i)}(x_{k+2}) &\geq U(x_{k+1}, h^{(i+1)}) + \alpha_{i+1} \mathcal{V}(x_{k+2}) \\ &\vdots \\ V^{(i)}(x_{k+N}) - V^{(i)}(x_{k+N+1}) &\geq U(x_{k+N}, h^{(i+1)}) + \alpha_{i+1} \mathcal{V}(x_{k+N+1}). \end{aligned} \tag{2.33}$$

Summing up the left and the right hand sides of (2.33) respectively,

$$\begin{aligned} V^{(i)}(x_k) - & V^{(i)}(x_{k+N+1}) \\ & \geq \sum_{j=k}^{k+N} U(x_j, h^{(i+1)}) + \alpha_{i+1} \sum_{j=k}^{k+N} \mathcal{V}(x_{j+1}), \end{aligned} \tag{2.34}$$

where N is a positive integer corresponding to gait cycles in this paper. Then, $V^{(i+1)}(x_{k+N+1}) \to 0$ as $h^{(i+1)}$ is an stablizing control policy as proved in Theorem 1, and $\lim_{N\to\infty} (\sum_{j=k}^{k+N} U(x_j, h^{(i)}) + \alpha_{i+1} \sum_{j=k}^{k+N} \mathcal{V}(x_{j+1})) = V^{(i+1)}(x_k)$. Hence, (2.34) yields

$$V^{(i)}(x_k) \geq V^{(i+1)}(x_k). \tag{2.35}$$

According to (2.16) and (2.35), we can obtain

$$\begin{aligned} Q^{(i+1)}(x_k, u_k) &= U(x_k, u_k) + V^{(i+1)}(x_{k+1}) \\ &\leq U(x_k, u_k) + V^{(i)}(x_{k+1}) \\ &= Q^{(i)}(x_k, u_k). \end{aligned} \tag{2.36}$$

Theorem 3. *Let Assumption 1 hold. Let $Q^{(i)}(x_k, u_k)$ and $h^{(i)}$ be updated by* (2.16)-(2.17)*, respectively, where $h^{(0)}$ is an admissible control policy that makes $Q^{(0)}(x_k, u_k)$ finite. Then for $i = 0, 1, 2, \ldots$, $h^{(i)}$ is an admissible control policy.*

Proof. From (2.15) and Theorem 2 we have

$$\begin{aligned} Q^{(0)}(x_k, u_k) &\geq Q^{(1)}(x_k, u_k) \\ &= U(x_k, u_k) + \sum_{j=1}^{\infty} U(x_{k+j}, h^{(1)}(x_{k+j})) \\ &\quad + \sum_{j=1}^{\infty} \alpha_1 \mathcal{V}(x_{k+j}). \end{aligned} \tag{2.37}$$

As $Q^{(0)}(x_k, u_k)$ is finite given $h^{(0)}$ is admissible for x_k, u_k, we have $Q^{(1)}(x_k, u_k)$ is also finite for x_k, u_k, and thus $\sum_{j=1}^{\infty} U(x_{k+j}, h^{(1)}(x_{k+j})) < \infty$. Given Assumption 1 and Theorem 1, we can conclude that $h^{(1)}$ is admissible. By mathematical induction, we can prove $h^{(i)}$ is admissible for $i = 0, 1, 2, \ldots$.

Theorem 4. *Let the iterative value function $Q^{(i)}(x_k, u_k)$ and the control policy $h^{(i)}(x_k)$ be obtained from* (2.16) *and* (2.17)*, respectively, and the optimal value function $Q^*(x_k, u_k)$ and the optimal policy be defined in* (2.10) *and* (2.11)*, respectively. Then $Q^{(i)}(x_k, u_k) \to Q^*(x_k, u_k)$ and $h^{(i)}(x_k) \to h^*(x_k)$ as $i \to \infty$, $\forall (x_k, u_k) \in \mathcal{D}$.*

Proof. By definition, $Q^*(x_k, u_k) \leq Q^{(i)}(x_k, u_k)$ holds for any i, and from Theorem 2 $\left\{Q^{(i)}(x_k, u_k)\right\}$ is a non-increasing sequence that is bounded by $Q^*(x_k, u_k)$. Hence $\left\{Q^{(i)}(x_k, u_k)\right\}$ must have a limit as $i \to \infty$. Denote this limit as $Q^{(\infty)}(x_k, u_k) \triangleq \lim_{i\to\infty} Q^{(i)}(x_k, u_k)$ and $h^{(\infty)}(x_k) \triangleq \lim_{i\to\infty} h^{(i)}(x_k)$. Note that $\lim_{i\to\infty} \alpha_i \mathcal{V}(x_{k+1}) = 0$, take the limits in (2.16) and (2.17) as $i \to \infty$,

$$Q^{(\infty)}(x_k, u_k) = U(x_k, u_k) + Q^{(\infty)}(x_{k+1}, h^{(\infty)}(x_k)), \tag{2.38}$$

$$h^{(\infty)}(x_k) = \arg\min_{u_k} Q^{(\infty)}(x_k, u_k). \tag{2.39}$$

The Bellman optimality equation for $V(x_k)$ is

$$V^*(x_k) = \min_{h(.)} \left[U(x_k, h(x_k)) + V^*(x_{k+1})\right]. \tag{2.40}$$

When $i \to \infty$, $u_k = h^{(\infty)}(x_k)$, so from (2.38) and (2.39) we can get

$$\begin{aligned} V^{(\infty)}(x_k) &= Q^{(\infty)}(x_k, h^{(\infty)}(x_k)) \\ &= \min_{u_k} \left[U(x_k, u_k) + Q^{(\infty)}(x_{k+1}, h^{(\infty)}(x_k))\right] \\ &= \min_{u_k} \left[U(x_k, u_k) + V^{(\infty)}(x_{k+1})\right]. \end{aligned} \tag{2.41}$$

Equation (2.41) satisfies the Bellman optimality equation (2.40), thus $V^{(\infty)}(x_k) = V^*(x_k)$. From (2.38) we can obtain

$$\begin{aligned} Q^{(\infty)}(x_k, u_k) &= U(x_k, u_k) + V^{(\infty)}(x_{k+1}) \\ &= U(x_k, u_k) + V^*(x_{k+1}) \\ &= Q^*(x_k, u_k). \end{aligned} \tag{2.42}$$

Therefore $h^{(\infty)}(x_k) = h^*(x_k)$ can be obtained from (2.39). The proof is complete.

Next, we consider the case of different types of errors that may affect the Q-function, such as value function approximation errors, policy approximation errors and errors from using N samples to evaluate the ith policy during policy iteration. We show an error bound analysis of FPI while taking into account approximation errors.

We need the following assumption to proceed.

Assumption 2. There exists a finite positive constant γ that makes the condition $\min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \leq \gamma U(x_k, u_k)$ hold uniformly on $\mathcal{X}$.

For most nonlinear systems, it is easy to find a sufficiently large number γ to satisfy this assumption as $Q^*(\cdot)$ and $U(\cdot)$ are finite.

Define a value function $\bar{Q}^{(i)}$ as

$$\bar{Q}^{(i)}(x_k, u_k) = U(x_k, u_k) + \hat{Q}^{(i-1)}(x_{k+1}, h^{(i)}(x_{k+1}))$$

for $i = 1, 2, \ldots$ and $\bar{Q}^{(0)} = Q^{(0)}$. Given the existence of universal approximators, the total approximation error can be considered finite during a single iteration, and therefore

$$\xi Q^{(i)} \leq \hat{Q}^{(i)} \leq \eta \bar{Q}^{(i)} \tag{2.43}$$

holds uniformly for i as well as x_k and u_k, where $0 < \xi \leq 1$ and $\eta \geq 1$ are constants, $\hat{Q}^{(i)}(x_k, u_k)$ is defined by (2.21) and $Q^{(i)}$ is defined by (2.15).

Theorem 5. *Let Assumptions 1 and 2 hold. Let $\hat{Q}^{(i)}(x_k, u_k)$ be defined by* (2.21) *and $Q^{(i)}$ be defined by* (2.15)*. Given $1 \leq \beta < \infty$ that makes $Q^* \leq Q^{(0)} \leq \beta Q^*$ hold uniformly for x_k, u_k. Let the approximate Q-function $\hat{Q}^{(i)}$ satisfies the iterative error condition* (2.43)*. If the following condition is satisfied*

$$\eta < \frac{\gamma + 1}{\gamma}, \tag{2.44}$$

then the iterative approximate Q-function $\hat{Q}^{(i)}$ *is bounded by*

$$\begin{aligned} &\xi Q^* \leq \hat{Q}^{(i)} \\ &\leq \left[\eta\beta(\frac{\eta\gamma}{1+\gamma})^i + (1-(\frac{\eta\gamma}{1+\gamma})^i)\frac{\eta}{1+\gamma-\eta\gamma}\right] Q^*. \end{aligned} \tag{2.45}$$

Moreover, as $i \to \infty$*, the approximate Q-function sequence* $\{\hat{Q}^{(i)}\}$ *approaches* Q^* *bounded by:*

$$\xi Q^* \leq \hat{Q}^{(\infty)} \leq \frac{\eta}{1+\gamma-\eta\gamma} Q^*. \tag{2.46}$$

Proof. The left-hand side of (2.45) can be easily obtained according to (2.43) and Theorem 3.

The right-hand side of (2.45) is proven by mathematical induction as follows.

First, for $i = 0$, $\hat{Q}^{(0)} \leq \eta\bar{Q}^{(0)} = \eta Q^{(0)} \leq \eta\beta Q^*$ holds according to (2.43) and the conditions in Theorem 5. Thus (2.45) holds for $i = 0$.

Assuming that (2.45) holds for $i \geq 0$, then for $i+1$ we have

$$\begin{aligned} &\bar{Q}^{(i+1)}(x_k, u_k) \\ &= U(x_k, u_k) + \hat{Q}^{(i)}(x_{k+1}, h^{(i+1)}(x_{k+1})) \\ &= U(x_k, u_k) + \min_{u_{k+1}} \hat{Q}^{(i)}(x_{k+1}, u_{k+1}) \\ &\leq U(x_k, u_k) + \min_{u_{k+1}} P_i Q^*(x_{k+1}, u_{k+1}), \end{aligned} \tag{2.47}$$

where

$$P_i = \eta\beta(\frac{\eta\gamma}{1+\gamma})^i + (1-(\frac{\eta\gamma}{1+\gamma})^i)\frac{\eta}{1+\gamma-\eta\gamma}. \tag{2.48}$$

According to Assumption 2, (2.47) yields

$$\begin{aligned}
&\bar{Q}^{(i+1)}(x_k, u_k) \\
&\leq (1+\gamma\frac{P_i-1}{\gamma+1})U(x_k,u_k) \\
&\quad + (P_i - \frac{P_i-1}{\gamma+1})\min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \\
&= \frac{1}{\eta}\left[\eta\beta(\frac{\eta\gamma}{1+\gamma})^{i+1} + (1-(\frac{\eta\gamma}{1+\gamma})^{i+1})\frac{\eta}{1+\gamma-\eta\gamma}\right] \\
&\quad \left[U(x_k,u_k) + \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1})\right] \\
&= \frac{1}{\eta}\left[\eta\beta(\frac{\eta\gamma}{1+\gamma})^{i+1} + (1-(\frac{\eta\gamma}{1+\gamma})^{i+1})\frac{\eta}{1+\gamma-\eta\gamma}\right] \\
&\quad \times Q^*(x_k, u_k).
\end{aligned} \tag{2.49}$$

On the other hand, according to (2.43), there is $\hat{Q}^{(i+1)} \leq \eta\bar{Q}^{(i+1)}$. Thus (2.45) holds for $i+1$. By mathematical induction, the proof for (2.45) is completed.

Considering (2.43) and (2.45), we can easily obtain

$$\hat{Q}^{(\infty)} \leq \frac{\eta}{1+\gamma-\eta\gamma} Q^* \tag{2.50}$$

as $i \to \infty$. Thus (2.46) holds. The proof is complete.

Remark 3. Condition (2.44) ensures that the upper bound in (2.46) is finite and positive. When $\xi = 1$ and $\eta = 1$, there is $Q^* \leq \hat{Q}^{(\infty)} \leq Q^*$ according to Theorem 5. Hence, $\hat{Q}^{(\infty)} = Q^*$. This means when $\xi = 1$ and $\eta = 1$, the sequence of $\hat{Q}^{(i)}$ converges to Q^* as $i \to \infty$.

2.6 Robotic Knee Impedance Control By FPI

We are now in a position to apply FPI to solving the robotic knee impedance control parameter tuning problem that originally motivated our development of the FPI. The results reported here are based on an OpenSim simulation of the human-prosthesis system where OpenSim (https://simtk.org/) is a widely accepted simulator

of human movements that was developed and maintained by the National Center for Simulation in Rehabilitation Research (NCSRR) under the support from the National Institute of Health. In OpenSim, five rigid-body segments linked by one degree-of-freedom pin joints were used to represent the human body. Segment lengths, masses, and other model settings were adopted from the lower limb OpenSim model. To simulate walking patterns of a unilateral above-knee amputee, the right knee was treated as a prosthetic knee and controlled by FS-IC, while the other joints in the model (left hip, right hip and left knee) were set to follow prescribed motions.

The dynamics in the OpenSim walking model are deterministic, which means identical gait performance can be obtained from the model if the conditions of the simulations are the same. In fact, the human sensorimotor system is inherently noisy and highly redundant. Therefore, it is necessary to add noise to the OpenSim model to realistically evaluate performance of different control algorithms. In Subsection 2.6.3, noise was either generated by a random number generator (the sensor noise and actuator noise cases in Table 2.3), or by gait-to-gait variances captured from two amputee subjects walking with prosthesis (case TF1 and TF2 in Table 2.3). For the latter case, data were collected from another study [80] where the experiments were approved by the Institutional Review Board at the University of North Carolina at Chapel Hill, and both amputee subjects provided written, informed consent. During the experiments, motion of intact joints (intact-side knee, intact-side hip, prosthesis-side hip) were collected using an 8-camera motion capture system (42 markers, 100 Hz, VICON, Oxford, UK) when amputee subjects were walking at a constant speed of 0.6 m/s on a treadmill. To applied real gait-to-gait variance in simulation, we first collected motion of the intact joints within 120 gait cycles from each subject of TF1 and TF2. Deviations to the average joint motions during gait cycles were calculated and applied to the prescribed joint motions in the OpenSim model accordingly when

simulating a gait cycle. Because the intact joints were controlled by human, introducing their variances to the OpenSim model can help represent the actual uncertainty of the human prosthesis system.

2.6.1 Algorithm and Experiment Settings

We summarize the parameters of the FPI in OpenSim simulations as follows. Algorithm 2.1 was applied to phases $m = 1, 2, 3, 4$ sequentially. The stage cost $U(x_k, u_k)$ is a quadratic form of state x_k and action u_k:

$$U(x_k, u_k) = x_k^T R_x x_k + u_k^T R_u u_k, \tag{2.51}$$

where $R_x \in \mathbb{R}^2$ and $R_u \in \mathbb{R}^3$ were positive definite matrices. Specifically, $R_x = diag(1, 1)$ and $R_u = diag(0.1, 0.2, 0.1)$ were used in our implementation. The minimum memory buffer size N_b was 20. During training, a small Gaussian noise (1% of the initial impedance) was added to the action output $u_k = h^{(i)}(x_k)$ to create samples to solve (2.16). The basis functions are $\phi(x_k, u_k) = [x(1)_k^2, x(1)_k x(2)_k, x(1)_k u(1)_k, x(1)_k u(2)_k, x(1)_k u(3)_k, x(2)_k^2, x(2)_k u(1)_k, x(2)_k u(2)_k, x(2)_k u(3)_k, u(1)_k^2, u(2)_k^2, u(3)_k^2, x(1)_k^2 x(2)_k, x(1)_k^2 u(1)_k, x(1)_k^2 u(2)_k]^T$,where $x(1)_k$ denotes the first element of x_k, and so on.

We define an experimental trial as follows. A trial started from gait cycle $k = 0$ until a success or failure status was reached. At the beginning of each trial, the FS-IC was assigned with random initial IC parameter I_0 as in (2.1). The adaptive optimal control objective for FPI is to make state x_k approach zero, i.e., the peak error ΔP_k and duration error ΔD_k for all four phases approach zero. We define upper bounds P^u and D^u and lower bounds P^l and D^l, and their values are identical to those in [30, Table I]. Specifically, upper bounds P^u and D^u are safety bounds for the robotic knee, i.e., $|\Delta P_k| \leq P^u$ and $|\Delta D_k| \leq D^u$ must hold during tuning. Lower bounds P^l

and D^l were used to determine whether a trial was successful: the current trial is successful if $|\Delta P_k| < P^l$ and $|\Delta D_k| < D^l$ hold for 10 consecutive gait cycles before reaching the limit of 500 gait cycles; otherwise it is failed. The maximum memory buffer size N in Algorithm 2.1 was 100. The results in Subsections 2.6.2 and 2.6.3 are based on 30 simulation trials. The success rate was the percentage of successful trials out of 30 trials.

We used two performance metrics in the experiments: the learning success rate as defined in Subsection 2.6.1, and tuning time measured by the number of gait cycles (samples) needed for a trial to meet success criteria. Tuning time also reflects on data efficiency.

2.6.2 FPI Batch Mode Evaluation

We first evaluated the performance of FPI under its simplest form, the batch mode where the entire batch (N_b samples) was generated under the policy to be evaluated (Setting 2(A) in Table 2.1), and neither PER nor prior knowledge was considered.

Table 2.2: FPI Tuner Performance under Batch Mode

N_b	Options*	Success Rate	Tuning Time (mean±sd)
20 (Fixed)	(A)(A)(A)(A)	76% (23/30)	93.4±13.6
40 (Fixed)		87% (26/30)	170.5±22.8
100 (Fixed)		100% (30/30)	428.6±52.2
20-40 (Ad.)	(B)(A)(A)(A)	93% (28/30)	107.6±12.4
40-100 (Ad.)		100% (30/30)	268.0±22.5

*refer to Table 2.1. Ad.: adaptive.

Table 2.3: Performance Comparisons of prosthesis control

	FPI		GPI [31]		NFQCA [81]		dHDP [30]	
	SR	TT	SR	TT	SR	TT	SR	TT
Noise free	93% (28/30)	107±12	53% (16/30)	384±33	47% (14/30)	213±48	73% (22/30)	323±136
Uniform 5% actuator	90% (27/30)	106±17	53% (16/30)	402±37	47% (14/30)	218±48	70% (21/30)	332±124
Uniform 10% actuator	83% (25/30)	112±19	53% (16/30)	401±36	40% (12/30)	226±54	73% (22/30)	348±141
Uniform 5% sensor	83% (25/30)	105±15	50% (15/30)	384±33	43% (13/30)	220±50	73% (22/30)	326±122
Uniform 10% sensor	80% (24/30)	128±21	43% (13/30)	421±28	33% (10/30)	223±51	70% (21/30)	342±138
TF Human Subject 1	77% (23/30)	147±22	43% (13/30)	459±32	30% (9/30)	225±47	70% (21/30)	350±126
TF Human Subject 2	80% (24/30)	142±17	40% (12/30)	456±41	36% (11/30)	245±53	70% (21/30)	361±129

FPI: proposed flexible policy iteration; GPI: generalized policy iteration; NFQCA: neural fitted Q with continuous actions; dHDP: direct heuristic dynamic programming; SR: Success rate for 30 trials; TT: Tuning Time, which is the number of gait cycles to success.

Table 2.2 summarizes the performance of FPI in batch mode with different batch sizes. In our experiments we observed that the both the success rate and tuning time rose as more samples (i.e. larger batch size N_b) are used for policy evaluation. Table 2.2 also shows that, under Setting 2(A), adaptive batch mode improves both success rates and tuning time over fixed batch mode.

Fig. 2.5 was generated under the setting of (A)(A)(A)(A) as in Table 2.1 and $N_b = 20$. Fig. 2.5 illustrates converging policies computed according to (2.26).

2.6.3 Comparisons with Other Methods

We now conduct a comparison study between FPI and three other popular RL algorithms. These RL algorithms include generalized policy iteration (GPI) [31], neural fitted Q with continuous action (NFQCA) [81] and our previous direct heuristic dynamic programming (dHDP) implementation [30]. GPI is an iterative RL algorithm that contains policy iteration and value iteration as special cases. To be specific, when the max value update index $N_i = 0$, it reduces to value iteration; when $N_i \to \infty$, it becomes policy iteration. NFQCA and dHDP are two configurations similar in the sense that both have features resemble SARSA and temporal difference (TD) learning. According to [81], NFQCA can be seen as the batch version of dHDP.

To make a fair comparison between FPI and the other three RL algorithms, we made FPI run under batch mode with neither PER nor prior knowledge involved. Specifically, results in Table 2.3 were based on an adaptive batch size N_b between 20 and 40 (i.e., Settings (B)(A)(A)(A) in Table 2.1), and results in Fig. 2.6 used a fixed N_b of either 20 or 40 (i.e., Settings (A)(A)(A)(A) in Table 2.1).

Before the comparison study, we first validated our implementations of GPI, NFQCA and dHDP using examples from [30, 81, 31], respectively. We were able to reproduce the reported results in those papers. For GPI, N and N_i were set equal

to p and N_i as described in [31], respectively. GPI's critic network (CNN) and the action network (ANN) were chosen as three-layer back-propagation networks with the structures of 2–8–1 and 2-8-3, respectively. For NFQCA, N was equivalent to the pattern set size $\#D$ in [81]. For both NFQCA and dHDP, CNN and ANN were chosen as 5-8-1 and 2-8-3 respectively. Notice that the number of neurons at the input layers are different, because NFQCA and dHDP approximate the state action value function $Q(x_k, u_k)$ while GPI approximates $V(x_k)$. To summarize, an effort was made to make the comparisons fair. For example, FPI's batch sample size N_b was equivalent to GPI's and NFQCA's N, thus the maximum N_b (FPI), N (GPI) and N (NFQCA) were all set to 40 gait cycles in Table 2.3.

Table 2.3 shows a systematic comparison of the four algorithms under various noise conditions. Artificially generated noise and noise based on variations of human subject movement profiles were used in the comparisons. To be specific, sensor noise and actuator noise are uniform noise that are added to the states x_k and actions u_k, respectively. In the last two rows, human variances collected from two amputee subjects TF1 and TF2 were introduced to the simulations, which would affect the states x_k. Under all noise conditions, FPI outperformed the other three existing algorithms in terms of both success rate and tuning time.

Fig. 2.6 compares the root-mean-square errors (RMSEs) between target knee angle profile and actual knee angle profile using FPI, GPI and NFQCA. Note that when we used a parameter setting of $(N = 40, N_i = 5)$ in GPI [31] which is in the typical range that has been tested, the RMSE increased after a few iterations. Also note from Fig. 2.6 that, GPI may achieve a similar performance as the FPI but it required a sample size of $N = 200$, which is much higher than FPI's case.

2.6.4 FPI Incremental Mode Evaluation

We now evaluate FPI under incremental mode to further study FPI's data and time efficiency. Both PER and learning from prior knowledge, two of the innovative features of FPI, can be employed in this mode.

To obtain prior knowledge $\mathcal{V}$ in (2.15) for the last row result in Table 2.4, we trained an FPI agent for just one trial in OpenSim under the same settings as those in the first row of Table (2.2) (Settings (A)(A)(A)(A) in Table 2.1 and $N_b = 20$). Then prior knowledge $\mathcal{V}$ is obtained from $\mathcal{V}(x_k) = \min_{u_k} \hat{Q}^*(x_k, u_k)$ where $\hat{Q}^*(x_k, u_k)$ the final approximate value function after Algorithm 1 is terminated.

Table 2.4: FPI Tuner Performance under Incremental Mode

Configuration	Options*	Success Rate	Tuning Time (mean±sd)
ER	(A)(B)(A)(A)	83% (25/30)	134.4±21.6
PER	(A)(B)(B)(A)	83% (25/30)	127.6±25.8
PER+Prior Knowledge	(A)(B)(B)(B)	90% (27/30)	103.3±15.1

*refer to Table 2.1. ER: Experience Replay; PER: Prioritized Experience Replay.

Table 2.4 summarizes the performance of FPI in incremental mode under three different configurations. ER or PER reutilized past samples from the current trial for policy iteration (Settings 2(B) in Table 2.1). The first configuration is the ER case without sample prioritization, i.e., $\rho_k^{(i)} = 1$ for all k. The second configurations prioritized the samples before performing the policy evaluation. In both the first and the second configurations (the first two rows in Table 2.4), no prior knowledge was used, i.e., $\mathcal{V}(x_k) = 0$ for all x_k. The third configuration (the third row in Table 2.4)

utilized both prioritized samples and prior knowledge. The prior knowledge $\mathcal{V}(x_k)$ was obtained from training FPI with a previous trial. In Table 2.4, the success rate increases from 83% to 90% as the algorithm gets more complex with PER and prior knowledge. The results also suggest that the introduction of sample prioritization and prior knowledge improves the data efficiency. Note that if the maximum number of gait cycles was extended from 500 to 1000, then the success rate of all simulation results in Table 2.4 will be 100%.

A statistical summary of a 30 randomly initialized trials based on the condition in row 1 of Table 2.4 is provided in Fig. 2.2 (bottom half panel). As shown, after tuning, the proposed FPI algorithm successfully reduced gait peak and duration errors.

2.7 Conclusion

We have proposed a new flexible policy iteration (FPI) algorithm aimed at providing data and time efficient parameter tuning for the control of a robotic knee with human in the loop. The FPI incorporates previous samples and prior knowledge during learning using PER and an augmented policy evaluation. Our results not only show qualitative properties of FPI as a stabilizing controller and that it approaches approximate optimal solution, but also include extensive simulation evaluations of control performance of FPI under different implementation conditions. We also compared FPI with other comparable algorithms, such as dHDP, NFQCA and GPI, which further demonstrates the efficacy of FPI as a data and time efficient learning controller. The FPI under batch mode performed better than other comparable algorithms, and FPI became more efficient when utilizing (prioritized) experience replay and previous knowledge. Even though our application does not render itself as a big data problem, but our results show that FPI has the capability of efficiently working with a tight data budget.

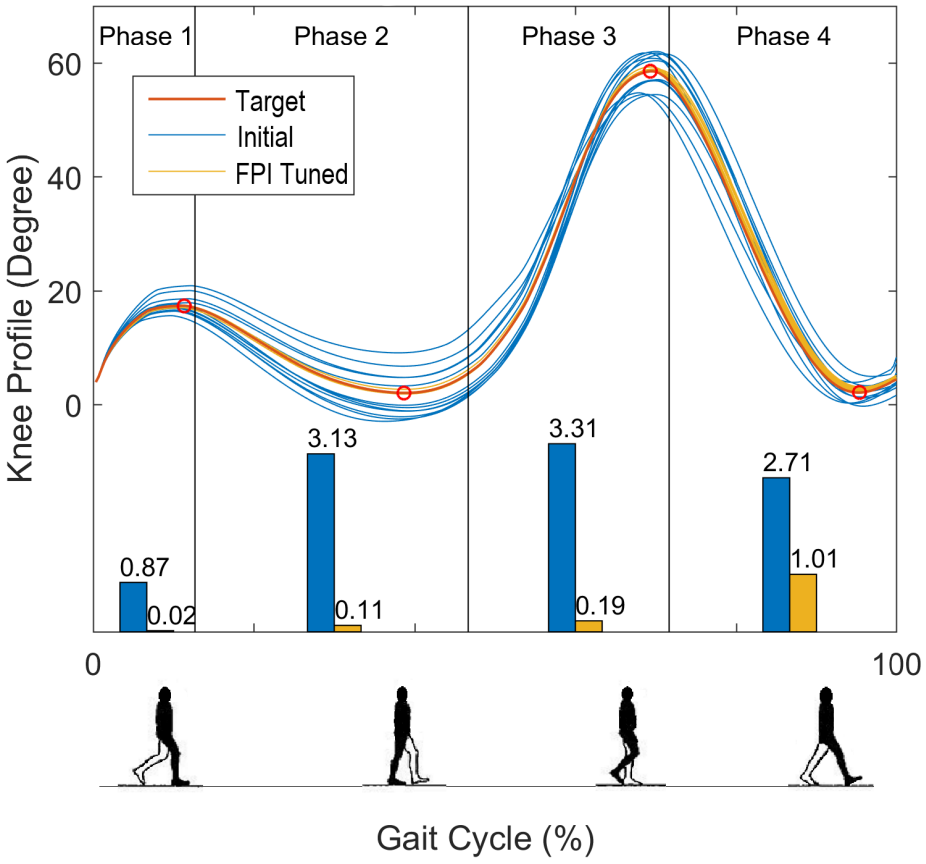
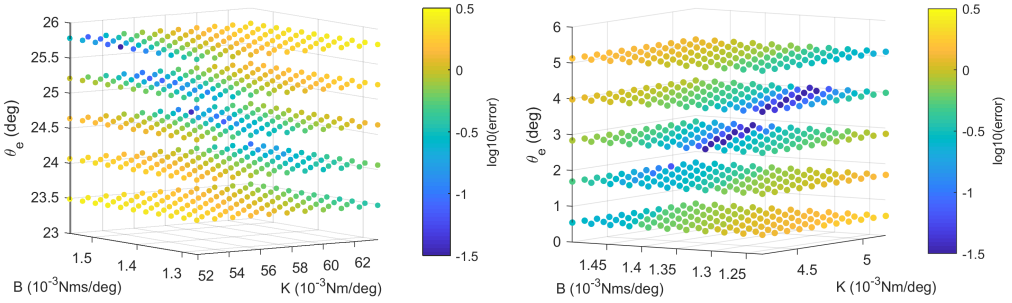
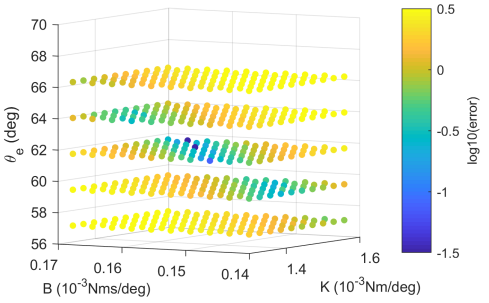


Figure 2.2: Top Half: Illustration of the Four Phases of a Gait Cycle: The Red Circles on the Target Profile (Red Curve) Indicate the Peak Angle Features of the Four Respective Phases (STF, STE, SWF, SWE). Bottom Half: Before-and-After FPI Tuning of Knee Profiles of 15 Randomly Selected Trials. The Blue Bars Are the RMSEs between the Initial Knee Angle Profiles and the Target Knee Profile, and the Yellow Bars Are the RMSEs between the FPI Tuned Knee Profiles and the Target Profile.

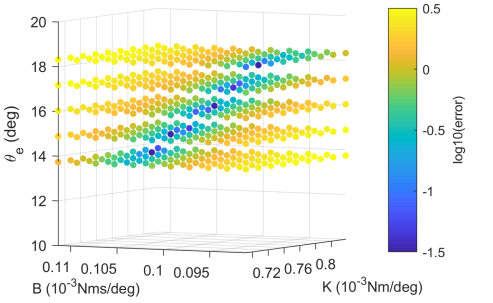


(a) Phase 1

(b) Phase 2

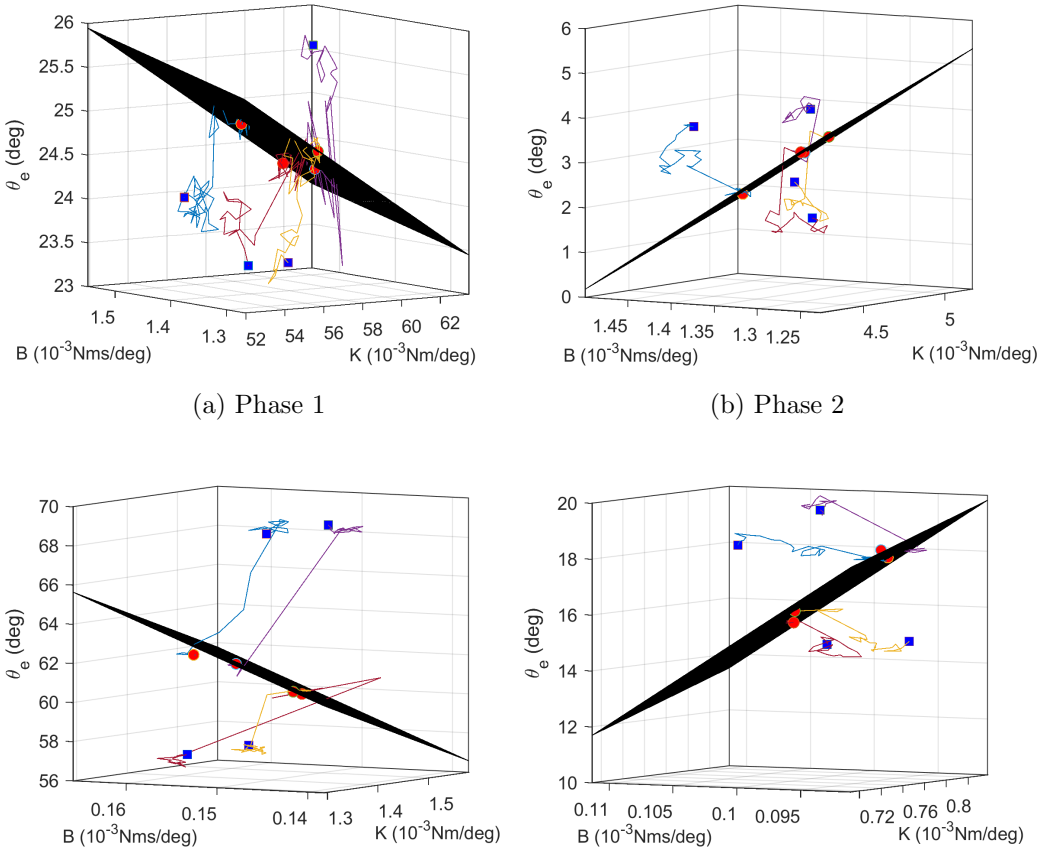


(c) Phase 3



(d) Phase 4

Figure 2.3: The Stage Cost in Peak Error and Duration Error As in Equation (2.51).



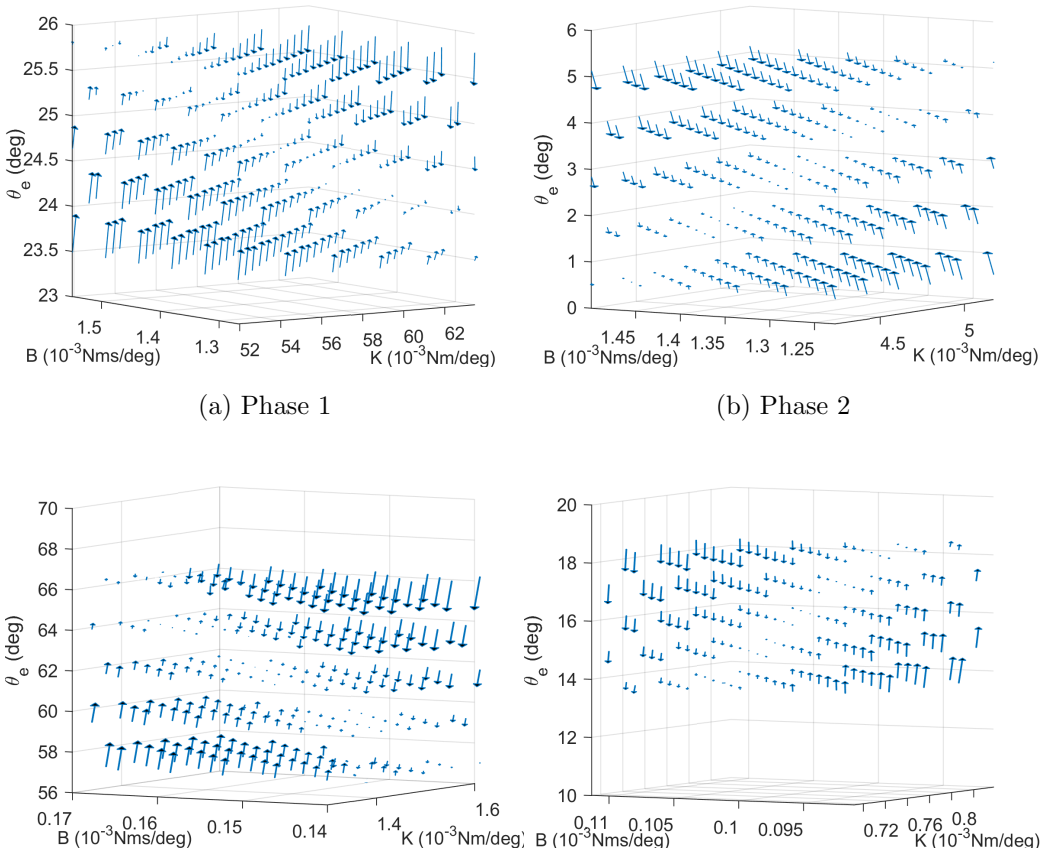
(a) Phase 1

(b) Phase 2

(c) Phase 3

(d) Phase 4

Figure 2.4: Illustration of the Converging Process of the IC Parameters During FPI Tuning: From Randomly Initialized IC Parameters (Four Trials for Illustration Here, Shown in Blue Squares) to the Final Parameters (Shown in Red Dots), Which Are Fitted With a Regression Response Surface.



(a) Phase 1

(b) Phase 2

(c) Phase 3

(d) Phase 4

Figure 2.5: Converging Policy Vector $[\Delta K, \Delta B, \Delta\theta_e]^T$.

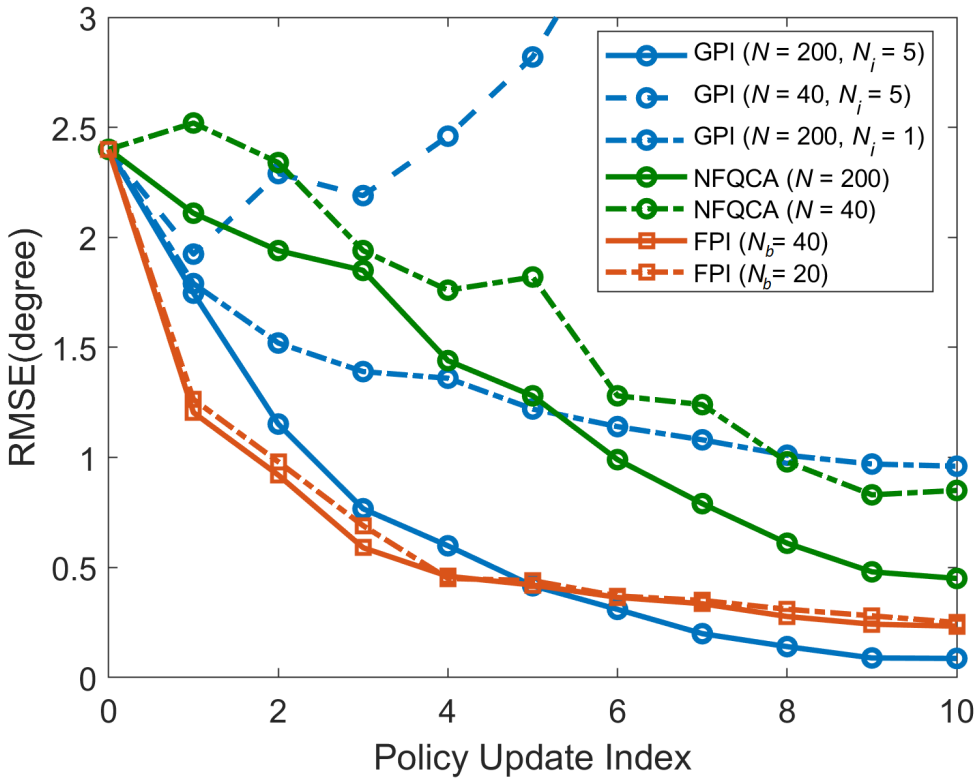


Figure 2.6: Comparison of the RMSEs Between Controlled Knee Profiles and Target Profiles Using FPI, GPI and NFQCA Under the Same Stage Cost (2.51).

Chapter 3

OFFLINE POLICY ITERATION BASED REINFORCEMENT LEARNING CONTROLLER FOR ONLINE ROBOTIC KNEE PROSTHESIS PARAMETER TUNING[1]

3.1 Abstract

This paper aims to develop an optimal controller that can automatically provide personalized control of robotic knee prosthesis in order to best support gait of individual prosthesis wearers. We introduced a new reinforcement learning (RL) controller for this purpose based on the promising ability of RL controllers to solve optimal control problems through interactions with the environment without requiring an explicit system model. However, collecting data from a human-prosthesis system is expensive and thus the design of a RL controller has to take into account data and time efficiency. We therefore propose an offline policy iteration based reinforcement learning approach. Our solution is built on the finite state machine (FSM) impedance control framework, which is the most used prosthesis control method in commercial and prototypic robotic prosthesis. Under such a framework, we designed an approximate policy iteration algorithm to devise impedance parameter update rules for 12 prosthesis control parameters in order to meet individual users' needs. The goal of the reinforcement learning-based control was to reproduce near-normal knee kinematics during gait. We tested the RL controller obtained from offline learning in real time experiment involving the same able-bodied human subject wearing a robotic lower limb prosthesis. Our results showed that the RL control resulted in good convergent

[1]THIS CHAPTER IS BASED ON A CO-FIRST AUTHORED PAPER [70] WITH MINHAN LI.

behavior in kinematic states, and the offline learning control policy successfully adjusted the prosthesis control parameters to produce near-normal knee kinematics in 10 updates of the impedance control parameters.

3.2 Introduction

The robotic prosthesis industry has experienced rapid advances in the past decade. Compared to passive devices, robotic prostheses provide active power to effciently assist gait in lower limb amptuees. Such active devices are potentially beneficial to amputees by providing the capability of decreased metabolic consumption during walking [83, 84], improved performance while walking on various terrains [85, 86], enhanced balance and stability [87], and improved adaptability to different walking speed [88]. In term of control for robotic prostheses, although several ideas [12, 89] have been proposed in recent years, the most commonly used approach in commercial and prototypic devices is still the finite state machine (FSM) impedance control [90, 5, 6].

The FSM impedance control framework requires customization of several impedance parameters for individual users in order to accommodate different physical conditions. This requirement currently poses a major challenge for broad adoption of the powered prosthesis devices because of the following reasons. For robotic knee prosthesis, the number of parameters to be configured is up to 15 [6, 11]. However, in clinical practice, only 2-3 parameters are practically feasible to be customized by prosthetists manually and heuristically. This procedure is time and labor intensive. Researchers have attempted alternative ways to manual tuning. To mimic the impedance nature of biological joint, intact leg models were studied to estimate the impedance parameters for the prosthetic knee joint [76, 91, 92]. Yet, the accuracy of these models have not been validated. Our group developed a cyber expert system approach to finding

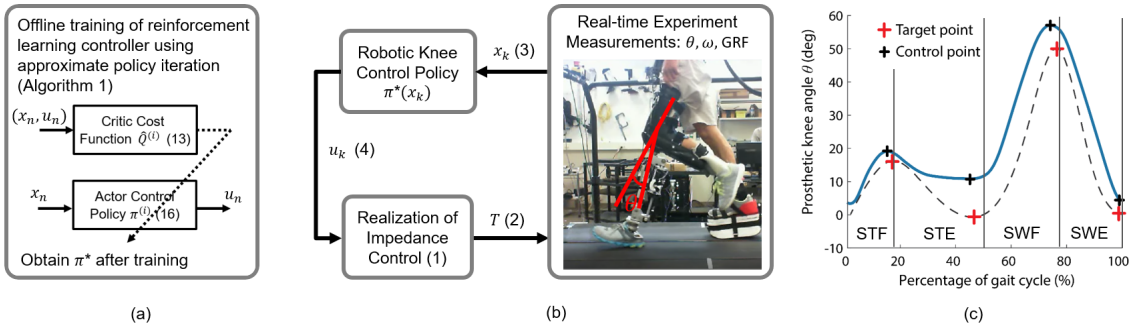


Figure 3.1: Overview of Offline Reinforcement Learning Controller Design and Online Human Subject Testing. (a) The Offline Training Process (Algorithm 1). (b) The Online Testing Process. (c) Target Knee Profile (Dash Curve) and Current Knee Profile (Blue Curve).

the impedance parameters [3]. This method is promising because of its model-free nature, however, its high demands for knowledge of experienced prosthesis tuning experts impedes its application in the real world. Most recently, some studies proposed to take into account the human's feedback in the optimization for the parameter configuration and demonstrated the promise. However, these methods still have some limitations, such as hard to extend for configuring high dimensional parameters [19] or imposing a prerequisite on the dataset which has to cover all users' preference [93].

In fact, the process of configuring impedance parameters can be formulated as a control problem of solving optimal sequential decisions. Because of the ability to autonomously learn an optimal behavior through interactions rather than explicitly formulate a detailed solution to a specific problem, the reinforcement learning (RL) based control design becomes a natural candidate when it comes to addressing the aforementioned challenges of configuring robotic knee prosthesis to meet individual needs. Recently, RL was successfully applied to solving robotic problems that involve sophisticated and hard-to-engineer behaviors. In most of these successful applica-

tions, policy search methods were at the center of the development [94, 95, 21, 96]. For example, Gu [96] developed an off-policy deep Q-function based RL algorithm to learn complex 7 DoF robotic arm manipulation policies from scratch for a door opening task. Vogt [97] presented a data-driven imitation learning system for learning human-robot interactions from human-human demonstrations. However, deep RL based methods may not be appropriate in some biomedical applications such as the human-prosthesis control problem under consideration. One primary reason is that training data involving human subjects are usually not easily acquired or expensive to collect. Additionally, experimental session involving human subjects usually cannot last more than one hour because of human fatique and safety considerations. Putting it together, we are in need of a reinforcement learning controller that can adapt to individual conditions in a timely and data efficient manner.

In our previous study [30, 37], we developed an actor critic RL controller, namely direct heuristic dynamic programming (dHDP) [52] to the robotic knee prosthesis parameter tuning problem. By interacting with the human-prosthesis system and under the same FSM impedance control framework, dHDP learned to reproduce near-normal knee kinematics. Though the dHDP showed its promise, it still took a relatively long time to complete the learning process. It took about 300 gait cycles or about 10 minutes of walking to achieve acceptable walking performance [37]. Moreover, because it is an online learning algorithm, it has not been developed to take advantage of existing offline data. Therefore, the problem calls for a more time efficient and data efficient solution.

To this end, we introduce an innovative, approximate policy iteration based reinforcement learning controller. Compared to the previous dHDP approach, it has several advantages. First, it enjoys several important properties of classic policy iteration algorithm such as convergent value functions and stable iterative control policies

[29, 58]. Second, it is reported that policy iteration has higher data and time efficiency than general gradient descent based methods [38]. Third, as we aim to show in this paper that our policy iteration based RL approach can learn from offline data to fully utilize historical data. As such, this learning controller can potentially be expanded to solve more complex problems that require an integration of both online and offline data.

The objective of this study is to develop and evaluate the feasibility of a policy iteration based learning control for personalizing a robotic prosthesis. In our previous study [98], we conducted a simulation study to indicate the potential of the proposed idea. Our approach is based on that in [29], which is further developed in this study to provide real time control for a real physical robotic prosthesis with human in the loop. The real human-prosthesis system is rich in unmodeled dynamics and uncertainties from environment and human. Especially, the human variances and consequent impact on the prosthetic knee and the human-prosthesis system have made controlling the robotic prosthesis more challenging than those problems encountered in humanoid robots or human-robot interactions to jointly perform a task such as picking up a box. This is because the human-prosthesis system interact and evolve seamlessly at an almost instantaneous time scale, i.e., a potentially out-of-control parameter adjustment in the prosthesis can result in system instability almost immediately, which is much less tolerant than a human-robot system.

In this chapter, for the first time, we successfully designed a reinforcement learning controller realized by approximate policy iteration to control robotic lower limb prosthesis with human in the loop. This new prosthesis control design approach is data efficient as it was derived from offline data collected from interactions between human and prosthesis. We demonstrated this learning controller for tuning 12 prosthesis parameters to approach desired normal gait on real human subject.

3.3 Human-Prosthesis Integrated System

3.3.1 Finite State Machine Framework

Fig. 5.1 illustrates reinforcement learning controlled prosthesis in a human-prosthesis integrated system. The learning controller is realized within a well established FSM platform. Specifically, an FSM partitions a gait cycle into four sequential gait phases based on knee joint kinematics and ground reaction force (GRF). These four gait phases are stance flexion (STF), stance extension (STE), swing flexion (SWF) and swing extension (SWE). In real-time experiments, transitions between phases are realized as those in [6] based on Dempster-Shafer theory (DST). For each phase, the prosthetic system mimicked a passive spring-damper-system with predefined impedance that matched the biological knee impedance. The predefined impedance parameters are selected by the finite state machine and outputted to the impedance controller as

$$I = [K, B, \theta_e]^T \in \mathbb{R}^3, \tag{3.1}$$

where K is stiffness, B is damping coefficient and θ_e is equilibrium position. In other words, for all four phases there are 12 impedance parameters to activate the knee joint which directly impact the kinematics of the robotic knee and thus the performance of the human-prosthesis system. The knee joint torque $T \in \mathbb{R}$ is generated based on the impedance control law

$$T = K(\theta - \theta_e) + B\omega. \tag{3.2}$$

The four target points (red markers) and four control points (black markers) in Fig. 5.1(c) provide state information for the learning controller to generate optimal control. The chosen points were the maximum or minimum points within each phase, so they could characterize basic knee movements. To approach the normal gait, target points were set to resemble the corresponding points in normative knee kinematics

measured in able-bodied individuals [99].

Specifically, one learning controller is designed for one phase under the FSM framework. Without loss of generality, our following discussion involves only one of the four phases. In each phase, peak error $\Delta P \in \mathbb{R}$ and duration error $\Delta D \in \mathbb{R}$ are defined as the vertical and horizontal distance between the corresponding pair of control point and target point. Then the state x of the RL controller are formed using $\Delta P \in \mathbb{R}$ and $\Delta D \in \mathbb{R}$ as

$$x = [\Delta P, \Delta D]^T. \tag{3.3}$$

Correspondingly, the action u is the impedance adjustment ΔI,

$$u = \Delta I. \tag{3.4}$$

Additional insights and construct on the FSM framework and the peak/duration errors can be found in [30].

3.4 Offline Reinforcement Learning Control Design

3.4.1 Problem Formulation

In this paper, we consider the integrated human-prosthesis system as a discrete-time nonlinear system (5.1),

$$x_{k+1} = F(x_k, u_k), k = 0, 1, 2, \ldots \tag{3.5}$$

$$u_k = \pi(x_k) \tag{3.6}$$

where k is the discrete time index that provides timing for each impedance control parameter update, $x_k \in \mathbb{R}^2$ is the state vector x at time k, $u_k \in \mathbb{R}^3$ is the action vector u at time k, F is the unknown system dynamics, and $\pi : \mathbb{R}^2 \to \mathbb{R}^3$ is the control policy.

To provide learning control of the prosthesis within system (5.1), we formulate an instantaneous cost function $U(x, u)$ in a quadratic form as

$$U(x, u) = x^T R_x x + u^T R_u u \tag{3.7}$$

where $R_x \in \mathbb{R}^{2\times 2}$ and $R_u \in \mathbb{R}^{3\times 3}$ are positive definite matrices. We use (3.7) to regulate state x and action u, as larger peak/duration error as in (3.3) and larger impedance adjustment as in (3.4) will be penalized with a larger cost.

The infinite horizon cost function $Q(x_k, u)$ is defined as

$$Q(x_k, u) = U(x_k, u) + \sum_{j=k+1}^{\infty} \gamma^{j-k} U(x_j, \pi(x_j)) \tag{3.8}$$

where γ is a discount factor. Note that the $Q(x_k, u)$ represents the cost function when action u is applied at state x_k, the system (5.1) then reaches x_{k+1} and follows the control policy π thereafter.

The optimal cost function $Q^*(x_k, u)$ satisfies the Bellman optimality equation

$$Q^*(x_k, u) = U(x_k, u) + \gamma Q^*(x_{k+1}, \pi^*(x_{k+1})) \tag{3.9}$$

where the optimal control policy $\pi^*(x_k)$ can be determined from

$$\pi^*(x_k) = \arg\min_u Q^*(x_k, u). \tag{3.10}$$

Policy iteration is used to solve the Bellman optimality equation (3.9) iteratively in this study. Policy iteration has several favorable properties such as convergence guarantee and high efficiency [29], which make it a good candidate for configuring a robotic knee with human in the loop. Starting from an initial admissible control $\pi^{(0)}(x_k)$, the policy iteration algorithm evolves from iteration i to $i + 1$ according to the following policy evaluation step and policy improvement step. Note that for offline training, a zero output policy is sufficient to be an initial admissible control.

Policy Evaluation

$$Q^{(i)}(x_k, u) = U(x_k, u) + \gamma Q^{(i)}(x_{k+1}, \pi^{(i)}(x_{k+1}))$$
$$i = 0, 1, 2, \ldots \tag{3.11}$$

Policy Improvement

$$\pi^{(i+1)}(x) = \arg\min_u Q^{(i)}(x, u), i = 0, 1, 2, \ldots \tag{3.12}$$

Equation (3.11) performs an off-policy policy evaluation, which means the action u need not to follow the policy being evaluated. In other words, $u \neq \pi^{(i)}(x_k)$ in general. This makes it possible to implement (3.11) and (3.12) in an offline manner using previously collected samples and thus achieve data efficiency. Solving (3.11) and (3.12) requires exact representations of both cost function and control policy, which is often not tractable in robotic knee configuration problem where continuous state and continuous control are involved. In Subsect. 3.4.2, we circumvent this issue by finding an approximated solution for (3.11) using offline data.

3.4.2 Offline Approximate Policy Iteration

For implementation of the policy evaluation equation (3.11), we used a quadratic function approximator to approximate the cost function $Q^{(i)}(x, u)$ in the ith iteration as

$$\hat{Q}^{(i)}(x, u) = \begin{bmatrix} x \\ u \end{bmatrix}^T S^{(i)} \begin{bmatrix} x \\ u \end{bmatrix} = \begin{bmatrix} x \\ u \end{bmatrix}^T \begin{bmatrix} S_{xx}^{(i)} & S_{xu}^{(i)} \\ S_{ux}^{(i)} & S_{uu}^{(i)} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \tag{3.13}$$

where $S^{(i)} \in \mathbb{R}^{5\times5}$ is a positive definite matrix and $S_{ux}^{(i)}, S_{xx}^{(i)}, S_{xu}^{(i)}$ and $S_{uu}^{(i)}$ are submatrices of $S^{(i)}$ with proper dimensions. The quadratic form of (3.13) corresponds to the instantaneous cost function $U(x, u)$ in (3.7).

To utilize offline data with the approximated cost function (3.13), samples are formulated as 3-tuples $(x_n, u_n, x'_n), n = 1, 2, 3 \ldots N$, where n is the sample index and

N is the total number of samples of the offline dataset. The 3-tuple (x_n, u_n, x'_n) means that after control action u_n is applied at state x_n, the system reaches the next state x'_n. In other words, $x_n \xrightarrow{u_n} x'_n$ is required to formulate a sample, but x'_n needs not to equal to x_{n+1} and u_n does not need to be on-policy, i.e. following a specific policy. Notice that k represents a sequential time evolution associated with gait cycle, but n does not need to follow such an order because offline sample n and $n+1$ may come from two different trials. Hence, collecting offline samples is much more flexible than collecting online learning samples. Having an offline dataset $D = \{(x_n, u_n, x'_n), n = 1, 2, 3 \dots N\}$, we can perform the following approximate policy evaluation step according to (3.11),

$$\hat{Q}^{(i)}(x_n, u_n) = U(x_n, u_n) + \gamma \hat{Q}^{(i)}(x'_n, \pi^{(i)}(x'_n)). \tag{3.14}$$

Solving (3.14) for $\hat{Q}^{(i)}(x_n, u_n)$ is equivalent to solving for $S^{(i)}$. In other words, based on (3.13), the policy evaluation (3.14) can be converted to the following convex optimization problem with respect to $S^{(i)}$,

$$\begin{aligned} &\text{minimize } \mu_n^T S^{(i)} \mu_n - \gamma (\mu'_n)^T S^{(i)} \mu'_n - U(\mu_n) \\ &\text{subject to } S^{(i)} \succ 0 \end{aligned} \tag{3.15}$$

where $\mu_n = [x_n^T, u_n^T]^T$ and $\mu'_n = [x'^T_n, \pi^{(i)}(x'_n)^T]^T$. After obtaining the $S^{(i)}$ and $\hat{Q}^{(i)}(x_n, u_n)$, we can update policy based on

$$\pi^{(i+1)}(x_n) = \arg\min_{u_n} \hat{Q}^{(i)}(x_n, u_n) \tag{3.16}$$

which is an approximate version of (3.12). In practice, constraints on actions are added to keep actions within a reasonable range (TABLE 3.1). As a result, policy update (3.16) can be converted to a quadratic programming problem,

$$\begin{aligned} &\text{minimize } \hat{Q}^{(i)}(x_n, u_n) \\ &\text{subject to } u_- \leqslant u_n \leqslant u_+ \end{aligned} \tag{3.17}$$

Table 3.1: Bounds on the Actions

Gait Phase	$K\ (N{\cdot}m/deg)$	$\theta_e\ (deg)$	$B\ (N{\cdot}m{\cdot}s/deg)$
STF	$[-0.1, 0.1]$	$[-1, 1]$	$[-0.001, 0.001]$
STE	$[-0.1, 0.1]$	$[-1, 1]$	$[-0.001, 0.001]$
SWF	$[-0.01, 0.01]$	$[-2, 2]$	$[-0.001, 0.001]$
SWE	$[-0.01, 0.01]$	$[-1, 1]$	$[-0.001, 0.001]$

where u_- and u_+ are the lower bound and upper bound of acceptable action, respectively. The values of u_- and u_+ can be found in TABLE 3.1. We used convex optimization [100] to solve (3.15) and (3.17).

Algorithm 5.1 summarizes the implementation of the offline approximate policy iteration algorithm.

Algorithm 3.1 Offline Approximate Policy Iteration

Input: training dataset $D = \{(x_n, u_n, x'_n), n = 1, 2, \ldots, N\}$

Output: optimal cost function $\hat{Q}^*(x, u)$ and policy $\pi^*(x_k)$

for $i = 1, 2, \ldots, i_{max}$

Get $S^{(i)}$ from (3.15) and $\hat{Q}^{(i)}(x, u)$ from (3.13)

Get policy $\pi^{(i+1)}(x)$ from (3.17)

end for

return $\hat{Q}^*(x, u) = \hat{Q}^{(i)}(x, u)$ and $\pi^*(x) = \pi^{(i+1)}(x)$

3.4.3 Implementation of Offline Policy Training

The offline training data including $N = 140$ pairs of the (x_n, u_n, x'_n) tuples came from two separate experiments invovling the same human subject using the same prosthesis device. The whole data collection process took 29 minutes to complete.

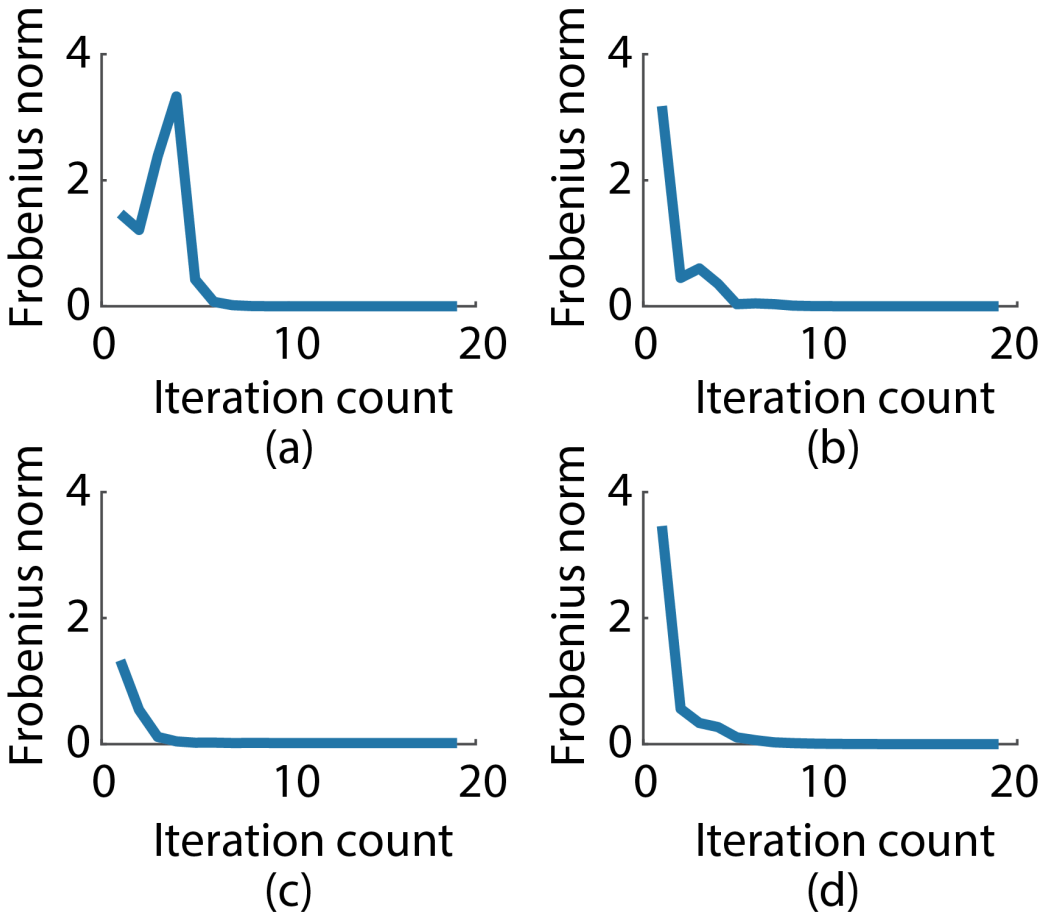


Figure 3.2: The Frobenius Norm of the Difference between Two Successive S Matrices. (a) STF. (b) STE. (c) SWF. (d) SWE.

During data collection, the prosthesis impedance parameters were controlled by the dHDP based RL approach that we investigated previously [30]. Note, however, that the dHDP was used to only provide some control to the prosthesis or in other words, dHDP was an enabler of the data collection session. That is to say that the data were drawn from the online learning process of the dHDP RL controller rather than generated by a well-learned policy. During data collection, the state x_n and next state x'_n in each pair of sampled tuples were averaged by 7 gait cycles conditioned on the same action u_n. In addition, prior to applying Algorithm 5.1, all samples were

normalized into the range between -1 and 1 to avoid ill-conditioning issues during application of convex optimization to achieve admissible control policies.

The discount factor γ was set to 0.8. The termination condition of the Algorithm 5.1 was set as a maximum of $i_{max} = 100$ iterations. The weight matrices of state and action were specified as $R_x = \text{diag}(10, 1)$ and $R_u = \text{diag}(1, 1, 1)$, respectively. They were specified to make the peak error dominating the cost. Because, compared to the duration error which is partially controlled by human behaviors (e.g. heel-strike or toe-off timing), the peak error is more sensitive to the parameter changes. Moreover, as a factor determining gait performance, the peak error is more important than the action taken in our settings. Yet, we still need to take the duration error as one of the monitored states in the controller, because the controller has to adjust parameters to keep the duration error in a reasonable range. Otherwise, human users cannot stabilize the duration error by themselves.

To evaluate the convergence of the trained policies, we investigated the changes of S matrix in the approximate cost function $\hat{Q}$ over the entire offline training process for each phase. As a measure of element-wise distance regarding two matrices, the Frobenius norm of the difference between two successive matrices $\left\|S^{(i+1)} - S^{(i)}\right\|_F$ was adopted to quantify the changes. As Fig. 3.2 shows, the norm value of the difference reduced fast when the training process started off for each phase, and they all approached zeros within 10 iterations. The result indicates that the approximated cost function as well as the policy was convergent and optimal given the training dataset. It took about 5 minutes to perform the offline training until reached the convergence.

3.5 Online Human Subject Testing Experiments

3.5.1 Experimental Protocol and Setup

The offline trained policy was implemented on the online able-bodied subject testing experiments. The male subject was the same one from whom we collected the offline training data. He was involved with informed consent. The experimental protocol was approved by the Institutional Review Board (IRB) of University of North Carolina at Chapel Hill. During the experiment, the subject wore a powered knee prosthesis and walked on a split-belt treadmill at a fixed speed of 0.6 m/s without holding handrails.

The entire experiment consisted of three sessions with different sets of initial impedance parameters for the prosthetic knee. The three sets of parameters were randomly selected, yet initially feasible to carry on policy iteration. The subject experienced 40 updates of the impedance control parameters for each phase of the FSM during a single experiment session. To reduce the influence of noises introduced by human variance during walking, the update period (i.e., the time index k in (5.1)) was set as 4 gait cycles (i.e., the states were obtained as an average of every 4 gait cycles). The proposed offline policy iteration based RL controller was used to automatically update impedance control parameters online such that actual knee kinematics approached predefined target points. At the beginning and at the end of each session, the subject had two stages of acclimation walking corresponding to the initial and final set of parameters, respectively. Each stage consisted of 20 gait cycles. The measured knee kinematics in the corresponding acclimation were averaged out to contrast the before-after effects of the proposed controller.

The robotic knee prosthesis used in this study was described in [6]. This prosthesis used a slider-crank mechanism, where the knee motion was driven by the rotation

of the moment arm powered by the DC motor through the ball screw. The prosthetic knee kinematics were recorded by a potentiometer embedded in the prosthesis. Some major gait events determining phase transitions in the finite state machine were detected by a load cell. The control system of the robotic knee prosthesis was implemented by LabVIEW and MATLAB in a desktop PC.

3.5.2 Performance Evaluations

Measures of knee kinematics were obtained at the beginning acclimation stage and at the ending acclimation stage during each session. These measurements reflect how the prosthetic knee joint moved when it interacted with the human subject before and after experiencing the control parameter update. By comparing the respective errors with respect to target points, the performance of the RL controller in a human-prosthesis system can be assessed.

While knee kinematic measures provide a quantitative evaluation of controller performance in terms of reaching desired gait target points, it is also necessary to consider an acceptable error range for the kinematic states. This is because the inherent human variance during walking. Our experiments indicate that when the peak errors and duration errors are within 2 degrees and 2 percent range of the target values, respectively, the human subject would not feel any discomfort or insecure while walking. Therefore, in our study, we adopted those error bounds.

3.5.3 Experimental Results

As Fig. 3.3 shows, the knee kinematics of the initial acclimation stages were different in three different sessions and distant from the target points, especially the peak angle errors. Clearly, after the impedance parameters were adjusted by the proposed RL controller, knee kinematics of the final acclimation stages approached

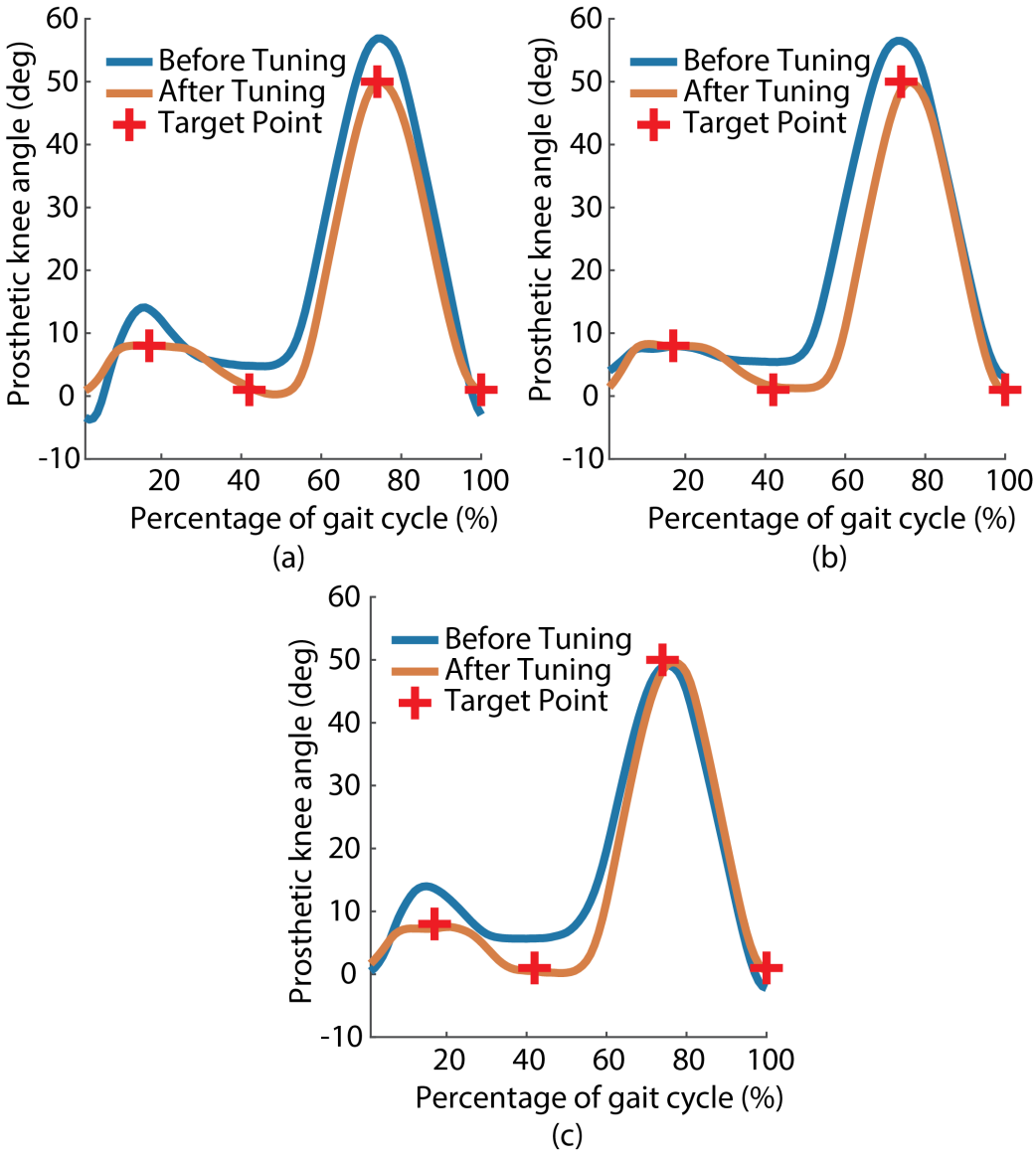


Figure 3.3: Comparisons of Knee Kinematics for Before and After Impedance Parameter Tuning Using Three Set of Initial Parameters.

the target points. Specifically, the averaged absolute values of the peak errors over the three sessions deceased from 4.18 ± 3.28 degrees to 0.56 ± 0.47 degrees for STF, from 4.33 ± 0.44 degrees to 1.11 ± 0.66 degrees for STE, from 4.92 ± 3.78 degrees to 0.14 ± 0.04 degrees for SWF and from 3.21 ± 1.23 degrees to 0.25 ± 0.23 degrees for SWE. The results indicate that offline policy iteration based RL controller is able to

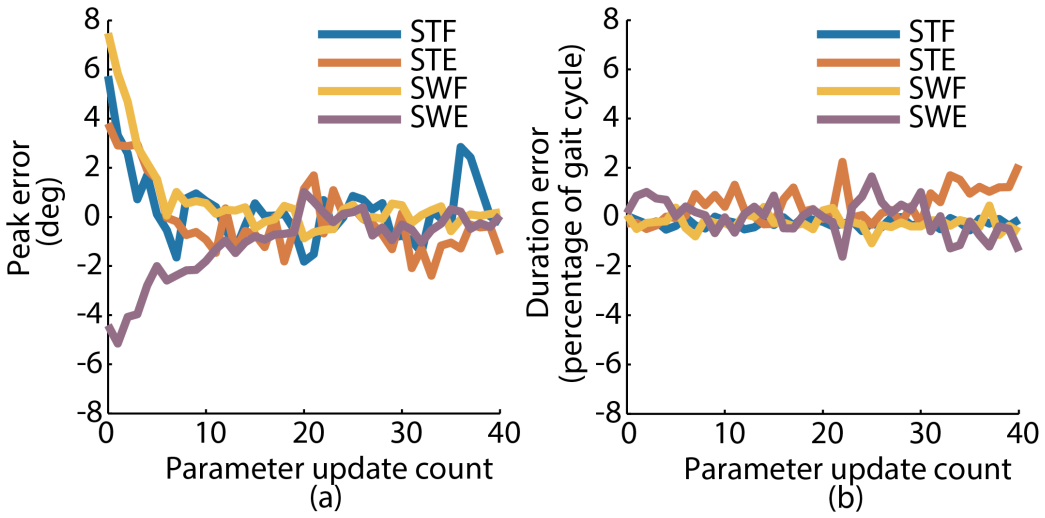


Figure 3.4: Evolution of States ((a) Peak Error and (b) Duration Error) As Impedance Parameters Were Updated.

reshape the prosthetic knee kinematics to meet the target points from different initial parameter settings.

Fig. 3.4 illustrates the evolution of peak errors and duration errors during the experimental session under the first set of initial parameters corresponding to the first result in Fig. 3.3. Since similar results were obtained from other experiment sessions, hereafter we only present the result from the first session as an example. All four phases experienced reduction in the peak angles errors at the end . Specifically, the peak error decreased from 5.8 degrees to -0.2 degrees for STF, from 3.8 degrees to -1.5 degrees in the STE phase. For SWF and SWE, they dropped from 7.4 degrees to 0.18 degrees and from -4.4 degrees to 0.05 degrees respectively.

The duration errors were insignificant, i.e., they were within the range of two percent of one gait cycle, and they remained within the range over the entire session. There are two considerations in this study. First, the duration time is controlled partially by human behavior, or in other words, the effect of controller on this state at the prosthetic knee is not the exclusively decisive factor. Second, given the previous

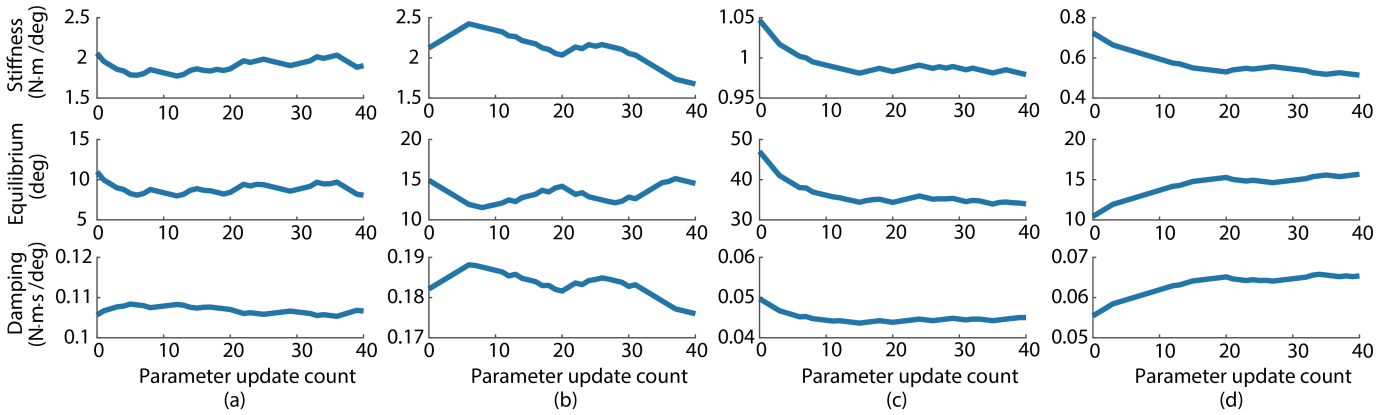


Figure 3.5: Evolution of the Impedance Parameters in Different Phases (a) STF (b) STE (c) SWF (d) SWE.

consideration, we placed more emphasis on the peak error than the duration error as reflected in the weighting matrix R_x in the quadratic cost measure.

The state errors at the final stage are mostly within the bounds of 2 degrees and 2 percent, respectively. These errors remained within bounds thereafter the first 10 parameter update cycles (40 gait cycles, about 1.3 minutes). Compared to the state errors achieved by dHDP [30], the offline policy iteration based RL controller achieved comparable performance with small errors (i.e. ± 2 degrees, ± 2 percent), but with less time to adjust the impedance control parameters. Specifically, it took dHDP 10 minutes of experiment (300 gait cycles) to achieve comparable state errors.

Note that the peak errors from the STF and the STE phases are usually associated with more oscillations than the other two swing phases as the state errors approach zeros (from the 10^{th} update to the 40^{th} update). In addition, as illustrated in Fig. 3.5, the impedance parameters exhibited different change patterns during the experimental sessions. It is apparent that the impedance parameters during swing phases converged in the first 20 updates and remained stationary thereafter. However, the impedance parameters exhibited somewhat oscillatory patterns during the stance phases. It is actually not surprising when we see the different patterns in the above. As can be understood, the stance phases involve direct interactions and thus directly affected by the ground, the human subject and the robotic prosthesis (for example, loading induced variation). Such varying interactions would introduce more perturbations to the prosthesis and result in oscillations. Whereas the swing phases are less likely to be affected by these factors and thus the state errors during these phases appear more stationary. Under the above discussed disturbances, the RL controller responded by making adjustments when it observed discrepancies between target and actual states. This unique phenomena is a result of us dealing with an inherently co-adapting human-prosthesis system.

3.6 Conclusion and Future Work

We developed a new data efficient and time efficient approximate policy iteration RL controller to optimally configure impedance parameters automatically for robotic knee prosthesis. The learning controller was trained offline using historical data and then the learned control policy was applied for online control of the prosthetic knee. Our experimental results validated this new approach and showed that it reproduced near-normal knee kinematics for the robotic knee prosthesis. Our results proved that the offline policy iteration based RL controller is a promising new tool to solve the challenging parameter tuning problems for the robotic knee prosthesis with human in the loop.

In this paper, we only collected one subject's data to train the offline policy and tested it on the same subject. Further studies need to be done to investigate whether the outcome of the proposed method can be generalized or transferred to other subjects. In addition, our future work will extend the current design to facilitate further online control policy adjustment. We believe such an integrated approach will facilitate even broader range of human-prosthesis integrated behavior to address changes in environment, task, and human condition.

Chapter 4

REINFORCEMENT LEARNING CONTROL WITH INFORMATION TRANSFER

4.1 Abstract

Recently, the class of temporal difference (TD) learning and value function based reinforcement learning (RL) algorithms are receiving much attention. However, these RL agents usually need to relearned from scratch when the model changes, or when a new task is presented, which greatly hinders their successful applications to our everyday life. In this paper, the problem of integrating structural transferred knowledge into RL is investigated. Here we develop a reinforcement learning control framework with information transfer (RL-IT) is developed by introducing data-based transferred knowledge into regular RL process. Realizing by regular Q-learning, the convergence of the algorithm is proved and the influence of the adaptive parameter is analyzed. The transferred knowledge is defined in a value function-like form which can be either obtained from previously learned agents or from data analysis and heuristics. Trade-off between the transferred knowledge and the current value function is achieved by an adaptive parameter. To implement the algorithm, a actor-critic neural network (NN) structure is developed. We evaluated the performance of the RL-IT algorithm using three classic problems, including the windy gridworld problem, the inverted pendulum cart-pole balancing problem and the human-prosthesis control problem. Results suggested that RL-IT outperformed the regular RL (without knowledge) in these three different tasks, and RL-IT is promising for learning for models with variations and for similar but new task.

4.2 Introduction

Machine learning is based on the idea that systems can learn knowledge directly from data, identify patterns by constructing statistical model and make decisions with minimal human intervention. Machine learning technologies have already achieved significant success in many applications including computer vision [101], game-playing AI [22, 24, 102] and robotic control [103, 104]. Traditional machine learning methods assume that training data and testing data are drawn from the same domain, such that the feature space and data distribution are the same [105, 106]. However, in many real-world scenarios, this assumption does not hold. For example, in indoor WiFi localization, it is very expensive to calibrate a localization model in a large-scale environment, as the WiFi signal strength may be varies with time, device or spaces. To reduce the re-calibration effort, we might want to adapt a localization model trained in one time period (the source domain) for a new time period (the target domain), or to adapt the localization model trained on one mobile device (the source domain) for a new mobile device (the target domain). However, the distributions of WiFi data collected over time or across devices may be very different, hence transfer learning (domain adaptation) is needed [107]. When it is expensive or impossible to re-collect the needed training data to rebuild the models, transfer learning between task domains would be much desirable.

Transfer learning (TL) attempts to improve on traditional machine learning by transferring knowledge learned in one or more source tasks and using it to improve learning in a related target task. TL is first established and widely used in supervised learning scenarios, where it also being referred as domain adaptation [40, 41, 42]. Pan and Yang [105] gave a systematic review of the works in TL under the paradigm of supervised learning (SL), especially for for classification, regression and clustering

problems that are related more closely to data mining tasks. According to [105], a *domain* consists of a feature space and a marginal probability distribution of the feature space; a *task* consists of a label space and an objective predictive function. As such, TL can be categorized under three sub-settings, *inductive TL*, *transductive TL* and *unsupervised TL*, based on different situations between the source and target domains and tasks. In the *inductive TL* setting, the target task is different from the source task, no matter when the source and target domains are the same or not. In the *transductive TL* setting, the source and target tasks are the same, while the source and target domains are different. Finally, in the *unsupervised TL* setting, similar to *inductive TL* setting, the target task is different from but related to the source task. However, the *unsupervised TL* focus on solving unsupervised learning tasks in the target domain, such as clustering, dimensionality reduction and density estimation . In this case, there are no labeled data available in both source and target domains in training.

Although TL has gain much success in supervised learning, transfer in RL is still a relatively new topic and insufficiently explored [43]. Traditional RL need to restart from scratch whenever the task changes, even when similar problems have been already solved or domain knowledge is available. The number of samples needed is often large and even prohibitive in real-world problems. To address these drawbacks, transfer RL has been proposed. There are several related names for transfer RL, such as lifelong RL [108] and reward shaping [69]. In general, transfer RL is motivated by reducing the number of samples needed in a new task via reutilizing previously learned knowledge, which is similar to transfer in SL. However, because of the principles of RL and SL are different, there are several important differences between transfer in RL and transfer in SL. In RL, state and action are the counterpart terms for feature in SL. As such, similar to SL where a *domain* consists of a feature space and its

distribution, the *domain* in RL can be defined as a combination of a state-action space and its distribution. Specifically, such a distribution can be expressed in the form of a system model, which relates both state and action. On the other hand, a *task* in SL consists of a label space and its prediction function. Similarly, a *task* in RL can be defined by a reward function as it relates to the goal of a RL agent, which is to maximize the accumulated rewards. However, according to the conventions of the transfer RL community, a *task* typically covers all components in RL, including state and action space, system function and reward function. Formal definitions of *task* and transfer learning under RL paradigm are given in Definitions 1 & 2. For transfer in single agent RL, past knowledge is transferred from previous task(s) to a similar task at hand, while for transfer for multi-agent RL (MARL), transferred knowledge not only comes from previously learned tasks, but also from more experienced agents in the same task [109, 110]. In this work,we will focus on transfer in single agent RL only.

Methods of transfer in RL can be categorized according to the forms of transferred knowledge, such as instance (sample), value function, policy, reward and environment (system model) [111]. Instance transfer is the most simple version of transfer algorithm, where the RL agent collects samples coming from different source tasks and reuses them in learning the target task. For example, the transfer of trajectory samples can be used to simplify the estimation of the action value function [112]. Policy shaping [113] is a special case of value function based transfer [114, 44], where human expert serves as an external critic to tell the RL agent whether its move is "right" or "wrong". Authors of [115] proposed a method that uses the past policies as a probabilistic bias where the learning agent faces three choices: the exploitation of the ongoing learned policy, the exploration of random unexplored actions, and the exploitation of past policies. Yet, many of these techniques involve the use of low level

information obtained in the source task by a RL agent, which may not be transferable to or incompatible with the agent learning in the new task, as the RL algorithms used in source and target task may differ in many ways.

Reward shaping methods provide prior knowledge to an agent through additional rewards [116, 117, 118, 119]. By incorporation of domain knowledge in RL via reward shaping, the training speed of a RL agent can be significantly improved [117]. However, to make the reward shaping signal become informative, expert knowledge and significant engineering effort is required [116]. To make the reward shaping process more autonomous, authors of [117, 118] designed a reward shaping method which has two separate representations: a Markov problem-space representation for reinforcement learning that differs for each task, and an agent-space representation that does not. The second representation is used to learn a shaping function that can provide value predictions for novel states across tasks. However, choosing an appropriate agent-space remains a difficult problem, and no analytical guarantee is provided whether or not it would not biased the final learning outcomes.

Unlike the above approaches, we propose a new knowledge transfer framework for the single task transfer problem where both the source task and target task have the same state and action variables. We built a knowledge representation via a value function obtaining by supervised learning into the RL update, and the knowledge transfer schedule results in a diminishing influence of previous knowledge which simultaneously allowing for increased attention to learning of the target task on hand. In this paper we realize our framework by a Q-learning algorithm with value function approximation. The main contributions of this paper are summarized as follows.

1. A flexible knowledge transfer framework for RL is proposed where the representation of the transferred knowledge can be either a value function or a regression model or both. It can be obtained in a data-driven manner or from

expert knowledge.

2. The amount of transferred knowledge into a new task can be programmed in a convenient way to address different applications needs.

3. Convergence analysis is provided for the proposed framework.

4. Comparisons and experiments with several classic RL problems are conducted, which verify the sample efficiency by our proposed RL-IT framework.

The structure of this paper is organized as follows: We describe the proposed framework of RL-IT in Section 4.3, and discuss its convergence properties in Section 4.4. Section 4.5 details the implementation. Experimental results with three classic RL problems are shown in Section 4.6, followed by discussion and conclusion in Sections 4.7 and 4.8.

4.3 Reinforcement Learning Control with Information Transfer

4.3.1 Transfer Reinforcement Learning

First, we give a definition of the *task* being discussed in this paper.

Definition 1. (Task) A task $\mathcal{T}$ consists of state space $\mathcal{X}$, action space $\mathcal{U}$, a joint probability distribution over the state-action space $P(\mathcal{X},\mathcal{U})$, a system function $F(\mathcal{X},\mathcal{U})$, a stage reward/cost function $R(\mathcal{X},\mathcal{U})$. A task $\mathcal{T}$ can be denoted by $\mathcal{T} = \{\mathcal{X},\mathcal{U}, P(\mathcal{X},\mathcal{U}), F(\mathcal{X},\mathcal{U}), R(\mathcal{X},\mathcal{U})\}$.

We now give a unified definition of transfer learning in the RL paradigm.

Definition 2. (Transfer Reinforcement Learning) Given a source task $\mathcal{T}_S$, a target task $\mathcal{T}_T$, transfer reinforcement learning aims to help improve the learning of the optimal policy h^* for $\mathcal{T}_T$ using the knowledge in $\mathcal{T}_S$, where $\mathcal{T}_S \neq \mathcal{T}_T$.

In the above definition, a task is a tuple $\mathcal{T} = \{\mathcal{X}, \mathcal{U}, P(\mathcal{X}, \mathcal{U}), F(\mathcal{X}, \mathcal{U}), R(\mathcal{X}, \mathcal{U})\}$. Thus the condition $\mathcal{T}_S \neq \mathcal{T}_T$ implies that there is at least one of the five components in $\mathcal{T}_S$ and $\mathcal{T}_T$ are different.

4.3.2 Reinforcement Learning Control for Nonlinear Discrete-Time Systems

Consider an nonlinear discrete-time system of the form

$$x_{k+1} = F(x_k, u_k), k = 0, 1, 2, \ldots \tag{4.1}$$

where $x_k \in \mathbb{R}^n$ is the state and $u_k \in \mathbb{R}^m$ is the control vector. Let x_0 be the initial state. The system function $F(x_k, u_k)$ is continuous for $\forall x_k, u_k$ and $F(0, 0) = 0$. Hence, $x = 0$ is an equilibrium state of system (4.1) under the control $u = 0$. Assume that the system (4.1) is stabilizable on a prescribed compact set $\Omega \in \mathbb{R}^n$.

Definition 3. (Stabilizable System) A nonlinear dynamical system is said to be stabilizable on a compact set $\Omega \in \mathbb{R}^n$, if for all initial states $x_0 \in \Omega$, there exists a control sequence $u_0, u_1, \ldots, u_i \in \mathbb{R}^m$, $i = 0, 1, \ldots$, such that the state $x_k \to 0$ as $k \to \infty$.

It is desired to find the control law $u_k = \pi(x_k)$ which minimizes the infinite horizon cost function given by

$$J(x_k, u_k) = \sum_{j=k}^{\infty} \gamma^{j-k} R(x_j, u_j), \tag{4.2}$$

where R is the stage cost function, $R(0, 0) = 0$, $R(x_j, u_j) \geq 0$ for $\forall x_j, u_j$, and γ is the discount factor with $0 < \gamma < 1$.

Assumption 1. The system is controllable, and the function $F(x_k, u_k)$ is Lipschitz continuous for $\forall x_k, u_k$; the system state $x_k = 0$ is an equilibrium state of system (4.1) under the control $u_k = 0$, i.e., $F(0, 0) = 0$; the feedback control $u_k = h(x_k)$ satisfies $u_k = h(x_k) = 0$ for $x_k = 0$; the stage cost function $U(x_k, u_k)$ in x_k and u_k is positive definite.

For optimal control problems, the designed feedback control must not only stabilize the system on Ω but also guarantee that (4.2) is finite, i.e., the control must be admissible.

Definition 4. (Admissible Control) A control $\pi(x)$ is said to be admissible with respect to (4.2) on Ω if $\pi(x)$ is continuous on a compact set $\Omega_\pi \in \mathbb{R}^m$, $\pi(0) = 0$, π stabilizes on Ω, and for $\forall x_0 \in \Omega$, $J(x_0, u_0)$ is finite.

Note that (4.2) can be written as

$$J(x_k, u_k) = R(x_k, u_k) + \gamma J(x_{k+1}, u_{k+1}). \tag{4.3}$$

According to Bellman's optimality principle, the optimal cost function $J^*(x_k, u_k)$ satisfies the action dependent discrete-time (DT) HJB equation

$$J^*(x_k, u_k) = R(x_k, u_k) + \gamma \min_{u_{k+1}} J^*(x_{k+1}, u_{k+1}). \tag{4.4}$$

Besides, the optimal control π^* can be expressed as

$$\pi^*(x_k) = \arg\min_{u_k} J^*(x_k, u_k). \tag{4.5}$$

By substituting (4.5) into (4.4), the DT HJB equation becomes

$$J^*(x_k, u_k) = R(x_k, u_k) + \gamma J^*(x_{k+1}, \pi^*(x_{k+1})). \tag{4.6}$$

where $J^*(x_k, u_k)$ is the value function corresponding to the optimal control policy $\pi^*(x_k)$. This equation reduces to the Riccati equation in the LQR case, which can be efficiently solved. In the general nonlinear case, the HJB cannot be solved exactly.

In the following subsections, we apply the Q-learning algorithm to solve for the value function $J^*(x_k, u_k)$ of the HJB equation (4.6).

4.3.3 Q-Learning

The Q-learning based value iteration algorithm is a method to solve the DT HJB online.

In the Q-learning algorithm, one starts with an initial value, e.g., $Q^{(0)}(x,u)=0$, and then solves for u_0 as follows:

$$u_0=\pi_0(x_k)=\arg\min_{u_k} Q_i(x_k,u_k).$$

Once the policy π_0 is determined, iteration on the value is performed by computing

$$Q^{(1)}(x_k,u_k)=R(x_k,u_k)+\gamma Q^{(0)}(x_{k+1},\pi_0(x_{k+1})).$$

The Q-learning value iteration scheme, therefore, is a form of incremental optimization that requires iterating between a sequence of action policies π_i determined by the greedy update

$$\pi_i(x_k)=\arg\min_{u_k} Q_i(x_k,u_k) \tag{4.7}$$

and a sequence $Q_i(x_k,u_k)\geq 0$ where

$$Q_{i+1}(x_k,u_k)=R(x_k,u_k)+\gamma Q_i(x_{k+1},\pi_i(x_{k+1})) \tag{4.8}$$

with initial condition $Q^{(0)}(x_k,u_k)=0$. Note that i is the value iteration index, whereas k is the time index. Combining (4.7) and (4.8), we have

$$Q_{i+1}(x_k,u_k)=R(x_k,u_k)+\gamma\min_{u_{k+1}} Q_i(x_{k+1},u_{k+1}). \tag{4.9}$$

Note that, as a value iteration algorithm, Q-learning does not require an initial stabilizing gain. This is important, as stabilizing gains are difficult to find for general nonlinear systems.

The Q-learning algorithm results in an incremental optimization that is implemented forward in time and online. Note that, unlike the case for policy iterations, the sequence $Q_i(x_k,u_k)$ is not a sequence of cost functions and is therefore not a sequence of Lyapunov functions for the corresponding policies $\pi_i(x_k)$ which are, in turn, not necessarily stabilizing.

4.3.4 Obtaining Prior Knowledge by Supervised Learning

The mathematical model of the system 4.1 is unknown. To proceed forward, we need to extract transferable knowledge from the task, such as system, task objective and etc. Here we introduce an data mining (supervised learning) approach to extract knowledge from prior task samples in the form of (x_k, u_k, x_{k+1}). We first obtain an regression model $\mathcal{F}$ given samples of (x_k, u_k, x_{k+1}):

$$x_{k+1} = \mathcal{F}(x_k, u_k), \tag{4.10}$$

where $x_{k+1} \in \mathbb{R}^n$ is the next state and $\mathcal{F}$ is the regression model. Note that $\mathcal{F}$ can be a linear regression model or an nonlinear regression model such as random forest or Gaussian regression model. Expert knowledge on system F (4.1) can help in choosing a proper regression model for $\mathcal{F}$.

After the regression model $\mathcal{F}$ is obtained, we can now formulate an value function of $Q'(x_k, u_k)$ based on $\mathcal{F}(x_k, u_k)$. The form of Q' is also related to the the stage reward or cost in RL. For example, Q' can be formulated as a quadratic form such that $Q' \geq 0$:

$$Q'_i(x_k, u_k) = \alpha_i x_{k+1}^2 = \alpha_i (\mathcal{F}(x_k, u_k))^2. \tag{4.11}$$

Note that here the form of Q' was manually defined and was not unique, and α_i is a ratio that satisfies $0 \leq \alpha_i \leq 1$, $\alpha_{i+1} \leq \alpha_i$ and $\alpha_i \to 0$ when $i \to \infty$. . As shown later, knowledge represented in Q' can be adopted by the designer at a preferred rate.

4.3.5 Reinforcement Learning Control with Information Transfer

Based on (4.7) and (4.8), we now propose a reinforcement learning control with information transfer which is initialized by $Q_0(x_k, u_k) = 0$.

$$\pi_i(x_k) = \arg\min_{u_k} [Q_i(x_k, u_k) + Q'_i(x_k, u_k)] \tag{4.12}$$

$$Q_{i+1}(x_k, u_k) = R(x_k, u_k) + \gamma[Q_i(x_{k+1}, \pi_i(x_{k+1})) + Q'_i(x_{k+1}, \pi_i(x_{k+1}))] \tag{4.13}$$

where Q'_i is a arbitrary positive semi-definite function that represents previously learned knowledge. Combining the above two equations, we have

$$Q_{i+1}(x_k, u_k) = R(x_k, u_k) + \gamma \min_{u_{k+1}}[Q_i(x_{k+1}, u_{k+1}) + Q'_i(x_{k+1}, u_{k+1})]. \tag{4.14}$$

Note that (4.7)-(4.9) are a special case for (4.12)-(4.14) when $Q'_i = 0$.

4.4 Properties of Reinforcement Learning with Information Transfer

4.4.1 Properties of the proposed algorithm without approximation error

The convergence analysis provided here is an extension of those given in [120, 121].

The QL update in (4.14) will generate a sequence of iterative Q-value function $\{Q_i(x_k, u_k)\}$. In this section, the convergence of the KG-QL algorithm will be proved by showing the sequence $\{Q_i(x_k, u_k)\}$ converges to the optimal value function $J^*(x_k, u_k)$. Before starting, the following Lemmas are required.

Lemma 1. *Let $\{\mu_i\}$ be be any arbitrary sequence of control laws and $\{\pi_i\}$ be the control laws as in* (4.7)*. Define Q_i as in* (4.8) *and define Λ_i as*

$$\Lambda_{i+1}(x_k, u_k) = R(x_k, u_k) + \gamma[\Lambda_i(x_{k+1}, \mu_i(x_{k+1})) + Q'_i(x_{k+1}, \mu_i(x_{k+1}))]. \tag{4.15}$$

If $Q_0(\cdot) = \Lambda_0(\cdot) = 0$, then $Q_{i+1}(x_k, u_k) \leq \Lambda_{i+1}(x_k, u_k), \forall i$.

Proof. It can be derived by noticing that Q_{i+1} is the result of minimizing the right-hand side of (4.8) with respect to the control input u_{k+1}, while Λ_{i+1} is a result of an arbitrary control input.

Lemma 2. *Let the sequence $\{Q_i\}$ be defined as in* (4.8)*. If the system is controllable on Ω and $Q'(0, 0) = 0$,there is an upper bound Y such that $0 \leq Q_i(x_k, u_k) \leq Y, \forall i$.*

Proof. Let $\eta(x_k)$ be any admissible control law, and let $Z_0(\cdot) = 0$, and Z_i is updated by

$$\begin{aligned} Z_{i+1}(x_k, u_k) = R(x_k, u_k) + \gamma[Z_i(x_{k+1}, \eta(x_{k+1})) \\ + Q_i'(x_{k+1}, \eta(x_{k+1}))] \end{aligned} \tag{4.16}$$

$$Z_1(x_k, u_k) = R(x_k, u_k) + \gamma Q_i'(x_{k+1}, \eta(x_{k+1})) \tag{4.17}$$

Let $Z_i(x_k, \eta) = Z_i(x_k, \eta(x_k))$ be a shorthand notation. Noticing the difference

$$\begin{aligned} & Z_{i+1}(x_k, u_k) - Z_i(x_k, u_k) \\ & = \gamma[Z_i(x_{k+1}, \eta) + Q_i'(x_{k+1}, \eta)] \\ & \quad - \gamma[Z_{i-1}(x_{k+1}, \eta) + Q_{i-1}'(x_{k+1}, \eta)] \\ & = \gamma[Z_i(x_{k+1}, \eta) - Z_{i-1}(x_{k+1}, \eta) + Q_i'(x_{k+1}, \eta) - Q_{i-1}'(x_{k+1}, \eta)] \\ & \leq \gamma(Z_i(x_{k+1}, \eta) - Z_{i-1}(x_{k+1}, \eta)) \\ & \leq \gamma^2(Z_{i-1}(x_{k+2}, \eta) - Z_{i-2}(x_{k+2}, \eta)) \\ & \vdots \\ & \leq \gamma^i(Z_1(x_{k+i}, \eta) - Z_0(x_{k+i}, \eta)) \\ & = \gamma^i Z_1(x_{k+i}, \eta), \end{aligned} \tag{4.18}$$

we can obtain

$$\begin{aligned}
&Z_{i+1}(x_k, u_k) \\
&\leq \gamma^i Z_1(x_{k+i}, \eta) + Z_i(x_k, u_k) \\
&\leq \gamma^i Z_1(x_{k+i}, \eta) + \gamma^{i-1} Z_1(x_{k+i-1},) + Z_{i-1}(x_k, u_k) \\
&= \gamma^i Z_1(x_{k+i}, \eta) + \gamma^{i-1} Z_1(x_{k+i-1}, \eta) \\
&\quad + \gamma^{i-2} Z_1(x_{k+i-2}, \eta) + Z_{i-2}(x_k, u_k) \\
&= \gamma^i Z_1(x_{k+i}, \eta) + \gamma^{i-1} Z_1(x_{k+i-1}, \eta) \\
&\quad + \gamma^{i-2} Z_1(x_{k+i-2}, \eta) + \ldots + \gamma Z_1(x_{k+1}, \eta) + Z_1(x_k, \eta), \\
&= \sum_{j=0}^{i} \gamma^j Z_1(x_{k+j}, \eta).
\end{aligned} \tag{4.19}$$

According to (4.17),

$$\begin{aligned}
&Z_{i+1}(x_k, u_k) \\
&\leq \sum_{j=0}^{i} \gamma^j [R(x_{k+j}, u_{k+j}) + \gamma Q_0'(x_{k+j}, \eta)] \\
&\leq \sum_{j=0}^{\infty} \gamma^j [R(x_{k+j}, u_{k+j}) + \gamma Q_0'(x_{k+j}, \eta)]
\end{aligned} \tag{4.20}$$

Since $\eta(x_k)$ is an admissible control input, i.e., $x_k \to 0$ and thus $Q_0'(x_k, \eta(x_k)) \to 0$ as $k \to \infty$,and the sequence of $\{R(x_{k+j}, u_{k+j})\}$and $\{Q_0'(x_{k+j}, \eta)\}$are bounded by some positive real numbers, i.e. $0 \leq R(x_{k+j}, u_{k+j}) \leq \mathcal{Y}_1, 0 \leq Q_0'(x_{k+j}, \eta) \leq \mathcal{Y}_2$,there exists finite Y_1 and Y_2 such that

$$\sum_{j=0}^{\infty} \gamma^j R(x_{k+j}, u_{k+j}) \leq \sum_{j=0}^{\infty} \gamma^j \mathcal{Y}_1 \leq Y_1, \forall i, \tag{4.21}$$

$$\sum_{j=0}^{\infty} \gamma^{j+1} Q_0'(x_{k+j}, \eta) \leq \sum_{j=0}^{\infty} \gamma^{j+1} \mathcal{Y}_2 \leq Y_2, \forall i. \tag{4.22}$$

Let $Y = Y_1 + Y_2$. By using Lemma 1, we get

$$Q_{i+1}(x_k, u_k) \leq Z_{i+1}(x_k, u_k) \leq Y, \forall i, \tag{4.23}$$

so the proof is completed.

Based on Lemmas 1 and 2, we now present the convergence proof of the Q-value function sequence.

Theorem 1. *Define the sequence* $\{Q_i\}$ *as in* (4.8) *with* $Q_0 = 0$*, and the control law sequence* $\{\pi_i\}$ *as in* (4.7)*. The learned knowledge* $Q'(x_k, u_k) \geq 0$*. Then, we can conclude that* $\{Q_i\}$ *is a nondecreasing sequence satisfying* $Q_i \leq Q_{i+1}, \forall i$*.*

Proof. Define a new sequence $\{\Phi_i\}$ as

$$\begin{aligned}\Phi_{i+1}(x_k, u_k) =& R(x_k, u_k) + \gamma[\Phi_i(x_{k+1}, \pi_{i+1}(x_{k+1})) \\ &+ Q'_{i+1}(x_{k+1}, \pi_{i+1}(x_{k+1}))]\end{aligned} \tag{4.24}$$

where $\Phi_0 = Q_0 = 0$. Now we show that $\Phi_i(x_k, u_k) \leq Q_{i+1}(x_k, u_k)$. Note that we use the shorthand notation in the following proof, e.g. $\Phi_i(x_{k+1}, \pi_{i+1}) = \Phi_i(x_{k+1}, \pi_{i+1}(x_{k+1}))$.

First, we prove that it holds for $i = 0$. Since

$$Q_1(x_k, u_k) = R(x_k, u_k) + \gamma Q'_0(x_{k+1}, \pi_0) \tag{4.25}$$

$$\begin{aligned}Q_1(x_k, u_k) - \Phi_0(x_k, u_k) &= R(x_k, u_k) + \gamma Q'_0(x_{k+1}, \pi_0) \\ &\geq 0,\end{aligned} \tag{4.26}$$

we have

$$\Phi_0(x_k, u_k) \leq Q_1(x_k, u_k). \tag{4.27}$$

Second, we assume that it holds for $i - 1$, i.e., $\Phi_{i-1}(x_k, u_k) \leq Q_i(x_k, u_k), \forall x_k$. Then, for i, from (4.13) and (4.24), we get

$$\begin{aligned}&Q_{i+1}(x_k, u_k) - \Phi_i(x_k, u_k) \\ &= \gamma[Q_i(x_{k+1}, \pi_i) - \Phi_{i-1}(x_{k+1}, \pi_i)] \geq 0\end{aligned} \tag{4.28}$$

i.e.,

$$\Phi_i(x_k, u_k) \leq Q_{i+1}(x_k, u_k). \tag{4.29}$$

Thus, the above equation is true for any i by mathematical induction.

Furthermore, according to Lemma 1, we know that $Q_i(x_k, u_k) \leq \Lambda_i(x_k, u_k)$. Combining with (4.29), we have

$$Q_i(x_k, u_k) \leq \Phi_i(x_k, u_k) \leq Q_{i+1}(x_k, u_k) \tag{4.30}$$

which completes the proof.

According to Lemma 2 and Theorem 1, we can obtain that $\{Q_i\}$ is a monotonically non-decreasing sequence with an upper bound, and therefore, its limit exists. Here, we define it as $\lim_{i\to\infty} Q_i(x_k, u_k) = Q_\infty(x_k, u_k)$ and present the following theorem.

Theorem 2. *Let the cost function sequence $\{Q_i\}$ be defined as in* (4.13)*. Then, its limit satisfies*

$$Q_\infty(x_k, u_k) = R(x_k, u_k) + \gamma \min_{u_{k+1}} Q_\infty(x_{k+1}, u_{k+1}) \tag{4.31}$$

Proof. For any u_{k+1} and i, according to (4.8), we can derive

$$\begin{aligned} Q_i(x_k, u_k) \leq & R(x_k, u_k) + \gamma[Q_{i-1}(x_{k+1}, u_{k+1}) \\ & + Q'_i(x_{k+1}, u_{k+1})] \end{aligned} \tag{4.32}$$

Combining with

$$Q_i(x_k, u_k) \leq Q_\infty(x_k, u_k), \tag{4.33}$$

which is obtained from (4.30), we have

$$\begin{aligned} Q_i(x_k, u_k) \leq & R(x_k, u_k) + \gamma[Q_\infty(x_{k+1}, u_{k+1}) \\ & + Q'_i(x_{k+1}, u_{k+1})], \forall i. \end{aligned} \tag{4.34}$$

Let $i \to \infty$, then $Q'_i \to 0$. We can obtain

$$Q_\infty(x_k, u_k) \leq R(x_k, u_k) + \gamma Q_\infty(x_{k+1}, u_{k+1}). \tag{4.35}$$

Note that in the above equation, u_{k+1} is chosen arbitrarily, thus, it implies that

$$Q_\infty(x_k, u_k) \leq R(x_k, u_k) + \gamma \min_{u_{k+1}} Q_\infty(x_{k+1}, u_{k+1}). \tag{4.36}$$

On the other hand, since the cost function sequence satisfies

$$\begin{aligned} Q_{i+1}(x_k, u_k) =& R(x_k, u_k) + \gamma \min_{u_{k+1}} [Q_i(x_{k+1}, u_{k+1}) \\ & + Q'_i(x_{k+1}, u_{k+1})] \end{aligned} \tag{4.37}$$

considering (4.33), we have

$$\begin{aligned} Q_\infty(x_k, u_k) \geq& R(x_k, u_k) + \gamma \min_{u_{k+1}} [Q_i(x_{k+1}, u_{k+1}) \\ & + Q'_i(x_{k+1}, u_{k+1})], \forall i. \end{aligned} \tag{4.38}$$

Let $i \to \infty$, then $Q'_i \to 0$. Then we can obtain

$$Q_\infty(x_k, u_k) \geq R(x_k, u_k) + \gamma \min_{u_{k+1}} Q_\infty(x_{k+1}, u_{k+1}). \tag{4.39}$$

Based on (4.36) and (4.39), we can conclude that (4.31) is true. The proof is complete.

Theorem 3. *Let* $\lim_{i\to\infty} Q_i(x_k, u_k) = Q_\infty(x_k, u_k) = J^*(x_k, u_k)$ *and* $\lim_{i\to\infty} \pi_i(x_k) = \pi_\infty(x_k) = \pi^*(x_k)$*, then the iterative Q-value and the policy convergent to the optimal value* J^* *and optimal policy* π^* *as defined in* (4.4) *and* (4.5)*,*

$$Q_\infty(x_k, u_k) = J^*(x_k, u_k), \tag{4.40}$$

$$\pi_\infty(x_k) = \pi^*(x_k). \tag{4.41}$$

Proof. According to Theorem 2 and the relationship between (4.12) and (4.13), we have

$$\begin{aligned} Q_\infty(x_k, u_k) &= R(x_k, u_k) + \gamma \min_{u_{k+1}} Q_\infty(x_{k+1}, u_{k+1}) \\ &= R(x_k, u_k) + \gamma Q_\infty(x_{k+1}, \pi_\infty(x_{k+1})) \end{aligned} \tag{4.42}$$

and

$$\pi_\infty(x_k) = \arg\min_{u_k} Q_\infty(x_k, u_k). \tag{4.43}$$

Observing (4.42) and (4.43), and then (4.4) and (4.5), we can find that (4.40) and (4.41) are true.

4.4.2 Properties of the proposed algorithm with approximation errors

The following analyses extends the results of [122, 123] to the action-dependent value function (Q-value function), discounted ($0 < \gamma < 1$), zero initial value function ($Q_0(x_k, u_k) = 0$) and with information transfer case.

Considering approximation errors in (4.12)(4.13), starting from $\hat{Q}_0(x_k, u_k) = 0$, for $i = 0, 1, 2, \ldots$, we have

$$\hat{\pi}_i(x_k) = \arg\min_{u_k}[\hat{Q}_i(x_k, u_k) + Q'_i(x_k, u_k)] + \rho_i(x_k) \tag{4.44}$$

$$\begin{aligned}\hat{Q}_{i+1}(x_k, u_k) = R(x_k, u_k) + \gamma[\hat{Q}_i(x_{k+1}, \pi_i(x_{k+1})) \\ + Q'_i(x_{k+1}, \pi_i(x_{k+1}))] + \varrho_i(x_k, u_k)\end{aligned} \tag{4.45}$$

where $\rho_i(x_k)$ and $\varrho_i(x_k, u_k)$ are finite approximation error functions of the iterative control and Q-value function, respectively. For convenience of analysis, for $\forall i = 0, 1, ...$, we assume that $\rho_i(x_k) = 0$ and $\varrho_i(x_k, u_k) = 0$ for $x_k = 0$ and $u_k = 0$.

Considering approximation errors, we generally have $\hat{\pi}_i(x_k) \neq \pi_i(x_k)$, $\hat{Q}_{i+1}(x_k, u_k) \neq Q_i(x_k, u_k)$, $i = 0, 1, ...$, and the convergence property in Theorem 1-3 for accurate case becomes invalid. Hence, in this subsection, a new convergence criteria will be established considering approximation errors in each iteration, which makes the iterative Q-value function converge to a finite neighborhood of the optimal one.

Define the target Q-value function as

$$\Gamma'_i(x_k) = \min_{u_k}\left\{U(x_k, u_k) + \gamma[\hat{Q}_{i-1}(x_{k+1}, u_{k+1}) + Q'_{i-1}(x_{k+1}, u_{k+1})]\right\} \tag{4.46}$$

and

$$\Gamma_i(x_k) = \min_{u_k} \left\{ U(x_k, u_k) + \gamma \hat{Q}_{i-1}(x_{k+1}, u_{k+1})] \right\}$$

where $\hat{Q}_0(x_k, u_k) = \Gamma_0(x_k, u_k) = 0$. For $i = 0, 1, 2, \ldots$,there exist finite constants $\upsilon \geq 1$ and $\tau \geq 1$ that makes

$$\hat{Q}_i(x_k, u_k) \leq \upsilon \Gamma_i'(x_k, u_k) \leq \upsilon\tau\Gamma_i(x_k, u_k). \tag{4.47}$$

hold uniformly. Let $\sigma = \upsilon\tau$, it becomes

$$\hat{Q}_i(x_k, u_k) \leq \sigma\Gamma_i(x_k, u_k). \tag{4.48}$$

Hence, we can give the following theorem.

Theorem 4. *Let $x_k \in \mathbb{R}^n$ be an arbitrary controllable state and Assumptions 1–4 hold. For $i = 0, 1, ...,$ let $\Gamma_i(x_k)$ be expressed as and $\hat{Q}_i(x_k, u_k)$ be expressed as* (4.45)*. Let $0 < \lambda < \infty$ and $1 \leq \delta < \infty$ be both constant that make*

$$J^*(x_{k+1}, u_{k+1}) \leq \lambda U(x_k, u_k) \tag{4.49}$$

hold uniformly. If there exists $1 \leq \sigma < \infty$ that makes (4.47) *hold uniformly, then we have*

$$\hat{Q}_i(x_k, u_k) \leq \sigma \left(1 + \sum_{j=1}^{i} \frac{\lambda^j \sigma^{j-1}(\sigma - 1)}{(\lambda + 1)^j} \right) J^*(x_k, u_k) \tag{4.50}$$

where $i, j = 0, 1, ...$

Proof. The theorem can be proved by mathematical induction. First, let $i = 0$. Then, (4.50) becomes

$$\hat{Q}_0(x_k, u_k) \leq \sigma J^*(x_k, u_k). \tag{4.51}$$

This can be obtained as $\hat{Q}_0(x_k, u_k) \equiv 0 \leq J^*(x_k, u_k) \leq \sigma J^*(x_k, u_k)$. Therefore, the conclusion holds for $i = 0$.

Next, let $i = 1$. According to (4.46), we have

$$\begin{aligned}
\Gamma_1(x_k, u_k) =& \min_{u_k} \left\{ U(x_k, u_k) + \gamma \hat{Q}_0(x_{k+1}, u_{k+1}) + \gamma Q_0'(x_{k+1}, u_{k+1}) \right\} \\
\leq& \min_{u_k} \{ U(x_k, u_k) + \gamma \sigma J^*(x_{k+1}, u_{k+1}) \} \\
\leq& \min_{u_k} \{ \left(1 + \lambda \frac{\sigma - 1}{\lambda + 1} \right) U(x_k, u_k) \\
& + \gamma \left(\sigma - \frac{\sigma - 1}{\lambda + 1} \right) J^*(x_{k+1}, u_{k+1}) \} \\
=& \left(1 + \lambda \frac{\sigma - 1}{\lambda + 1} \right) \min_{u_k} \{ U(x_k, u_k) + \gamma J^*(x_{k+1}, u_{k+1}) \} \\
=& \left(1 + \lambda \frac{\sigma - 1}{\lambda + 1} \right) J^*(x_k, u_k).
\end{aligned} \tag{4.52}$$

Note that in the derivation above, $\gamma J^*(x_{k+1}, u_{k+1}) \leq J^*(x_{k+1}, u_{k+1}) \leq \lambda U(x_k, u_k)$ holds given According to (4.47), we have

$$\hat{Q}_1(x_k, u_k) \leq \sigma \left(1 + \lambda \frac{\sigma - 1}{\lambda + 1} \right) J^*(x_k, u_k) \tag{4.53}$$

which shows that (4.50) holds for $i = 1$.

Assume that (4.50) holds for $i = l - 1$, where $l = 1, 2, \ldots$. Then, for $i = l$, let

$c = \sum_{j=1}^{l-1} \frac{\lambda^{j-1}\sigma^{j-1}(\sigma-1)}{(\lambda+1)^j}, d = \frac{\lambda^{l-1}\sigma^{l-1}(\sigma-1)}{(\lambda+1)^l}$, we have

$$\begin{aligned}
\Gamma_l\left(x_k, u_k\right) =& \min_{u_k}\left\{U\left(x_k, u_k\right) + \gamma\hat{Q}_{l-1}(x_{k+1}, u_{k+1})\right\} \\
\leq& \min_{u_k}\left\{U\left(x_k, u_k\right) + \gamma\sigma\left(1+\lambda c\right) J^*(x_{k+1}, u_{k+1})\right\} \\
\leq& \min_{u_k}\left\{U\left(x_k, u_k\right) + \gamma\sigma\left(1+\lambda c\right) J^*(x_{k+1}, u_{k+1})\right\} \\
\leq& \min_{u_k}\{\left(1+\lambda c+\lambda d\right) U\left(x_k, u_k\right) \\
& + \gamma\left[\sigma\left(1+\lambda c\right) - c - d\right] J^*(x_{k+1}, u_{k+1})\} \\
=& \left(1+\lambda\sum_{j=1}^{l-1}\frac{\lambda^{j-1}\sigma^{j-1}(\sigma-1)}{(\lambda+1)^j} + \lambda\frac{\lambda^{l-1}\sigma^{l-1}(\sigma-1)}{(\lambda+1)^l}\right) \\
& \min_{u_k}\left\{U\left(x_k, u_k\right) + \gamma J^*(x_{k+1}, u_{k+1})\right\} \\
=& \left(1+\sum_{j=1}^{l}\frac{\lambda^{j}\sigma^{j-1}(\sigma-1)}{(\lambda+1)^j}\right) J^*(x_k, u_k).
\end{aligned} \tag{4.54}$$

Then, according to (4.47), we can obtain (4.50), which proves the conclusion for $i = 0, 1, ...$The proof is complete.

Theorem 5. *Let $x_k \in \mathbb{R}^n$ be an arbitrary controllable state and Assumptions 1–4 hold. Suppose Theorem 4 holds for $\forall x_k \in \mathbb{R}^n$. If for $0 < \lambda < \infty$ the inequality*

$$1 \leq \sigma \leq \frac{\lambda+1}{\lambda} \tag{4.55}$$

holds, then as $i \to \infty$, the iterative performance index function $\hat{Q}_i(x_k, u_k)$ in (4.45) is uniformly convergent to a bounded neighborhood of the optimal performance index function $J^*(x_k, u_k)$, i.e.,

$$\lim_{i\to\infty}\hat{Q}_i(x_k, u_k) = \hat{Q}_\infty(x_k, u_k) \leq \sigma\left(1+\frac{\lambda(\sigma-1)}{1-\lambda(\sigma-1)}\right) J^*(x_k, u_k). \tag{4.56}$$

Proof. According to (4.54) in Theorem 4, we can see that for $j = 1, 2, \ldots.$ the

sequence $\left\{\frac{\lambda^j \sigma^{j-1}(\sigma-1)}{(\lambda+1)^j}\right\}$ is a geometric series. Then, (4.54) can be written as

$$\Gamma_i\left(x_k, u_k\right) = \left(1 + \frac{\frac{\lambda(\sigma-1)}{(\lambda+1)}\left(1 - (\frac{\lambda\sigma}{\lambda+1})^i\right)}{1 - \frac{\lambda\sigma}{\lambda+1}}\right) J^*(x_k, u_k). \tag{4.57}$$

As $i \to \infty$, if $1 \leq \sigma \leq \frac{\lambda+1}{\lambda}$, then (4.57) becomes

$$\lim_{i\to\infty} \Gamma_i\left(x_k, u_k\right) = \Gamma_\infty\left(x_k, u_k\right) \leq \left(1 + \frac{\lambda(\sigma-1)}{1-\lambda(\sigma-1)}\right) J^*(x_k, u_k). \tag{4.58}$$

According to (4.47), let $i \to \infty$, we have

$$\hat{Q}_\infty(x_k, u_k) \leq \sigma\Gamma_\infty(x_k, u_k). \tag{4.59}$$

From (4.58) and (4.59), we can obtain (4.56). The proof is complete.

Remark 1. Theorem 5 provides a convergent condition for the approximated Q-value function $\hat{Q}_i(x_k, u_k)$. Because of the existence of approximation errors, $\hat{Q}_i(x_k, u_k)$ is not necessary larger or smaller than $\Gamma_i\left(x_k, u_k\right)$. If there exists a $0 < \sigma < 1$ that satisfies (4.48), there must exist some $\sigma \geq 1$ that also satisfies (4.48). Since $0 < \sigma < 1$ is just a special case of $\sigma \geq 1$, here we will consider only the situation for $\sigma \geq 1$.

4.5 Implementation of Reinforcement Learning Control with Information Transfer

In this section, we discuss the implementation of the proposed knowledge guided QL algorithm for different types of applications based on whether the state-action space is discrete or continuous. For discrete state-action space such as finite state MDPs, Q-learning can be performed using exact Q-values which are stored in a Q-table. For continuous state-action space, actor-critic structure is developed to implement the adaptive QL algorithm. The actor and critic NNs are employed to approximate the Q-function and the control policy, respectively. The critic NN (CNN) weights are updated with a modified gradient decent method with additional function that captured the prior knowledge, and the actor NN (ANN) weights are updated based on the critic NN.

4.5.1 Exact Q-Learning using Q-Table

Consider the simple case for finite state MDP when the state and action spaces are small enough for the value functions to be represented as tables. In this tabular case, exact solutions, i.e. the optimal value function and the optimal policy, can often be found exactly by updating the values in the The proposed algorithm works similar to QL in this case, except that it follows a different equation (4.14) for updating the Q-values.

4.5.2 Actor-Critic Neural Networks

For the case where the state space is continuous, the proposed RL-IT method is implemented using an actor-critic structure.

Critic Neural Network

The CNN consisted of three layers of neurons, namely the input layer, the hidden layer and the output layer. For the input layer, it took the state $x_k \in \mathbb{R}4 \times 1$ and the action $u_k \in \mathbb{R}3 \times 1$ as inputs. The output of CNN, denoted by $\hat{Q}(x_k, u_k)$, approximates the the total cost-to-go value function $Q(x_k, u_k)$,

$$\hat{Q}^{(i)}(x_k, u_k) = W_{c2}^{(i)} \varphi(W_{c1}^{(i)} \begin{bmatrix} x_k \\ u_k \end{bmatrix}), \tag{4.60}$$

where $W_{c1}^{(i)}$ is the weight matrix between the input layer and the hidden layer, and $W_{c2}^{(i)}$ is the weight matrix between the hidden layer and the output layer. And $\varphi(\cdot)$ is the tangent sigmoid activation function,

$$\varphi(v) = \frac{1 - \exp(-v)}{1 + \exp(-v)}. \tag{4.61}$$

The hidden layer output is

$$h_{c1} = \varphi(v_{c1}) = \varphi(W_{c1}^{(i)} \begin{bmatrix} x_k \\ u_k \end{bmatrix}). \tag{4.62}$$

The prediction error $e_{c,k} \in \mathbb{R}$ of the CNN can be written as

$$\begin{aligned} e_{c,k}^{(i+1)} = R(x_k, u_k) + \gamma[Q_i(x_{k+1}, \pi^{(i)}(x_{k+1})) \\ + Q_i'(x_{k+1}, \pi^{(i)}(x_{k+1}))] - Q_{i+1}(x_k, u_k) \end{aligned} \tag{4.63}$$

To correct the prediction error, the weight update objective was to minimize the squared prediction error $E_{c,k}$, denoted as

$$E_{c,k}^{(i+1)} = \frac{1}{2}(e_{c,k}^{(i+1)})^2. \tag{4.64}$$

The weight update rule for the CNN was a gradient-based adaptation given by

$$W_c^{(i+1)} = W_c^{(i)} + \Delta W_c^{(i+1)} \tag{4.65}$$

The weight updates of the hidden layer matrix W_{c2} were

$$\Delta W_{c2}^{(i+1)} = l_c \left[-\frac{\partial E_{c,k}^{(i+1)}}{\partial W_{c2,k}^{(i+1)}} \right] = l_c \left[-\frac{\partial E_{c,k}^{(i+1)}}{\partial e_{c,k}^{(i+1)}} \frac{\partial e_{c,k}^{(i+1)}}{\partial Q_k^{(i+1)}} \frac{\partial Q_k^{(i+1)}}{\partial W_{c2,k}} \right]. \tag{4.66}$$

Here we used the short handed notation $Q_k^{(i+1)}$ for $Q_{i+1}(x_k, u_k)$.

The weight updates of the input layer matrix were W_{c1} were

$$\begin{aligned} \Delta W_{c1}^{(i+1)} &= l_c \left[-\frac{\partial E_{c,k}^{(i+1)}}{\partial W_{c1,k}^{(i+1)}} \right] \\ &= l_c \left[-\frac{\partial E_{c,k}^{(i+1)}}{\partial e_{c,k}^{(i+1)}} \frac{\partial e_{c,k}^{(i+1)}}{\partial Q_k^{(i+1)}} \frac{\partial Q_k^{(i+1)}}{\partial h_{c1}} \frac{\partial h_{c1}}{\partial v_{c1}} \frac{\partial v_{c1}}{\partial W_{c2,k}} \right] \end{aligned} \tag{4.67}$$

where $l_c > 0$ is the learning rate of the CNN.

Action Neural Network

The ANN consisted of three layers of neurons with two layers of weights, and it takes in the state $x_k \in \mathbb{R}4 \times 1$ from the system and output the actions $u_k \in \mathbb{R}$ to perform control actions to the system:

$$u_k = \varphi(W_{a2}^{(i)} * \varphi(W_{a1}^{(i)} x_k)), \tag{4.68}$$

where $W_{a1}^{(i)}$ and $W_{a2}^{(i)}$ are the weight matrices, and $\varphi(\cdot)$ was the tan-sigmoid activation function of the hidden layer and output layer. One major difference between the structures of ANN and CNN is that, there is no tan-sigmoid activation function on the output layer of CNN, while there is one on the output layer of ANN.

Under our problem formulation, the objective of adapting the ANN is to minimizing the absolute value of the approximated total cost-to-go $\hat{Q}^{(i)}(x_k, u_k)$ and make it close to 0. The weight update rule for the ANN was to minimize the following performance error:

$$E_{a,k}^{(i+1)} = \frac{1}{2}(\hat{Q}^{(i+1)}(x_k, u_k))^2. \tag{4.69}$$

Similarly, the weight matrix was updated based on gradient descent:

$$W_a^{(i+1)} = W_a^{(i)} + \Delta W_a^{(i+1)} \tag{4.70}$$

The weight updates of the hidden layer matrix W_{a2} are

$$\Delta W_{a2}^{(i+1)} = l_a \left[-\frac{\partial E_{a,k}^{(i+1)}}{\partial W_{a1,k}^{(i+1)}} \right]. \tag{4.71}$$

The weight updates of the input layer matrix W_{a1} are

$$\Delta W_{a1}^{(i+1)} = l_a \left[-\frac{\partial E_{a,k}^{(i+1)}}{\partial W_{a1,k}^{(i+1)}} \right]. \tag{4.72}$$

The above ANN and CNN weight updates and the memory guided QL implementation is summarized in Algorithm 1. The weights of both ANN and CNN were initialized with uniformly distributed random numbers between -1 and 1.

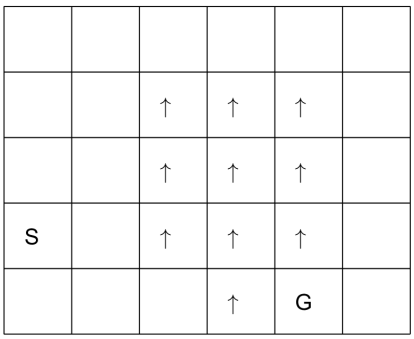
4.6 Experimental Results

4.6.1 Windy Gridworld

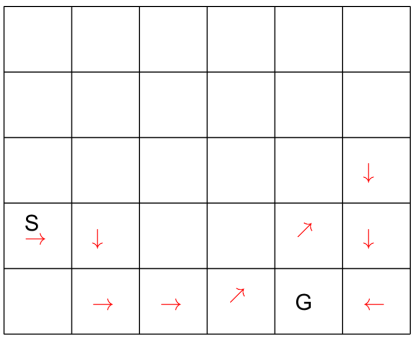
Fig. 4.1(a) shows a rectangular "windy gridworld" representation of a simple finite MDP, which is adopted from [124] and [125]. Each cell in the grid corresponds to a state. There are four actions can be taken at each cell: north, south, east, and west, which deterministically cause the agent to move one cell in the indicated direction. If the agent takes an action that would take it off the grid, its location will remain unchanged. All actions result in a reward of 0, except those that move the agent towards the goal state. Notice that there are arrows drawn in the "windy" states, where the agent experiences an extra "push" action that makes it move one step towards the indicated wind direction. For example, if the agent is in a windy state with south wind (pointing upward) and executes an action to right, the agent will end up move right one cell but also another one upward. As a result, the agent moves diagonally upward to the right. Fig. 4.1(b) shows an optimal policy (path) corresponds to the windy gridworld in Fig. 4.1(a), and the actual movements are affected by the wind.

We limited the maximum number of time steps in each episode to 30. At the beginning of each episode, the agent starts from the "Start" state and moves around according to some policy until it reaches a maximum of 30 steps or it reaches the goal state. Reward is -0.1 everywhere except that of the goal state is 10.

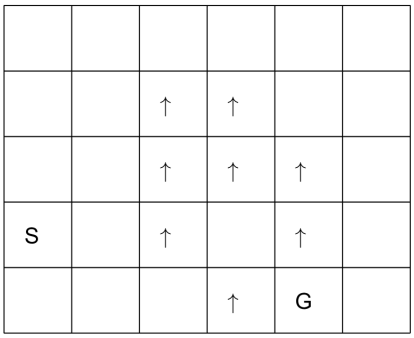
To test the idea of RL-IT we changed the wind as shown in Fig. 4.1(c), and changed the location of the goal state and size of the grid world as in Fig. 4.1(d). In the first case, the nine windy cells ranging from (2,3) to (4,5) each have a probability of 0.2 to become a wind-less cell (here (a, b) means ath row and bth column, counting from the top-right corner). In the second case, the size of the map was extended from



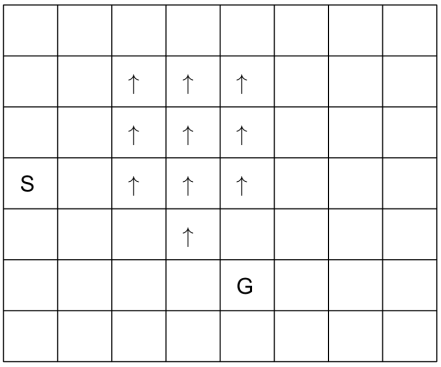
(a)



(b)



(c)



(d)

Figure 4.1: The Windy Gridworld Problem. (a) Original Gridworld, (b) Optimal Path for the Original Gridworld, (c) Change Wind, (d) Change Map Size and Goal.

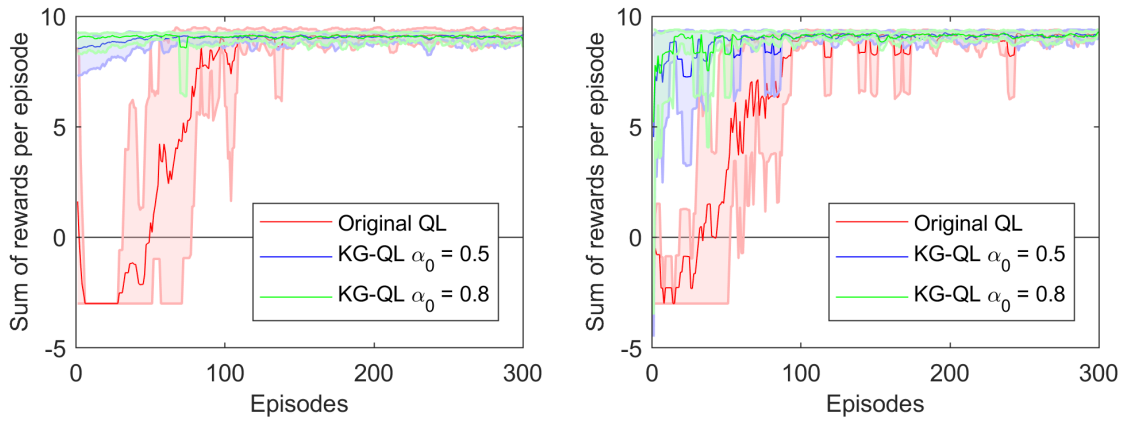
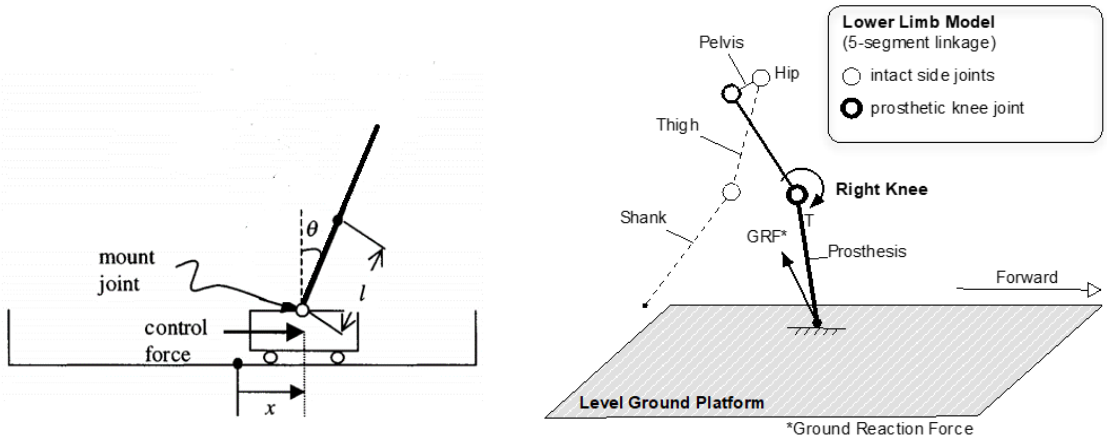


Figure 4.2: Comparison of Average Episode Reward of 10 Episodes. (a) Change Wind, (b) Change Size and Goal, the Shaded Area Indicates the Range of Data in the 10 Episodes.

a 5×6 map to a 7×8 map, meanwhile, the goal state was randomly chosen from cell (5,5), (5,6), (6,5), (6,6).

This wind grid-world problem can be solved by Q-learning. The Q-learning algorithm here used a tabular form to store the Q-values, and followed a ϵ-greedy exploration with $\epsilon = 0.1$. The learning rate was 0.1 and the discount rate $\gamma = 0.9$. The Q' values in this experiment was obtained from running the tabular form Q-learning on the original problem (Fig. 4.1(a)), then Q' table was used as the knowledge to guide the updates in the Q-values when a new task of "change wind" or "change size and goal" was presented.

The result in Fig. 4.2 shows the average episode reward of 10 episodes with changed wind or changed size and goal. Here the term episode reward means the total reward the agent actually collected in an episode. In both scenarios, with transferred knowledge, the Q-learning agents outperformed the ones without transferred



(a) (b)

Figure 4.3: Schematic Drawings of the Second and the Third Example Studied in This Work. (a) Inverted Pendulum Cart-Pole Balancing Problem. (b) Prosthesis Control Problem With Human in the Loop.

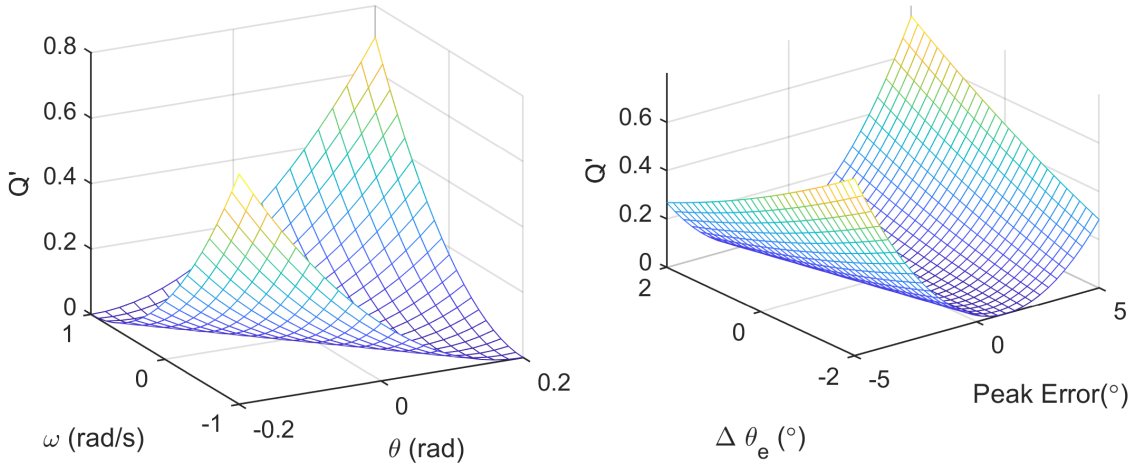
knowledge. Specifically, in the first scenario "changed wind", the average numbers of episodes experienced before reaching an episode reward of 8.5 are 22.4, 32.6 and 115.4 for original QL, RL-IT with $\alpha_0 = 0.5$ and RL-IT with $\alpha_0 = 0.8$ respectively. Correspondingly for the second scenario "changed size and goal", the three numbers are 53.2, 65.2 and 102.1 respectively. Notice that in Fig. 4.2 there are some cases where the episode reward are less than 0 at the early stage, and this implies that the agent failed to reach goal state during an episode.

4.6.2 Inverted Pendulum Cart-pole Balancing Problem

The inverted pendulum cart-pole model (Fig. 4.3(a)) is a classic problem in control theory and is often used as a benchmark for testing control strategies. The cart-pole system studied in this paper is the same as the one in [52], except that the control force that pushes or pulls the cart is continuous rather than binary.

This cart-pole model has four state variables: 1) position of the cart on the track; 2) angle of the pole with respect to the vertical position; 3) cart velocity; 4) angular velocity of the pole. We adopted the settings from [52] where a run consists of a maximum of 1000 consecutive episodes. It is considered successful if the last trial (trial number less than 1000) of the run has lasted 600 000 time steps. Otherwise, if the controller is unable to learn to balance the cart-pole within 1000 trials (i.e., none of the 1000 trials has lasted over 600 000 time steps), then the run is considered unsuccessful. In our simulations, we have used 0.02 s for each time step, and a trial is a complete process from start to fall. A pole is considered fallen when the pole is outside the range of $[-12^\circ, 12^\circ]$ and/or the cart is beyond the range of $[-2.4, 2.4]$ m in reference to the central position on the track.

In Fig. 4.3(a), the default value of the half-pole length l is 0.5 m. We first obtained the structural transferred knowledge Q' from the cart-pole model with default value of l using linear regression $\bar{x}_{k+1} = \mathcal{F}(\bar{x}_k, \bar{u}_k)$. Here $\bar{x}_k \in \mathbb{R}^2$ consisted of angle and angular velocity of the pole, in other words, two of the four state variables. The action variable $\bar{u}_k \in \mathbb{R}$ was the force applied on the cart. Sample pairs of $(\bar{x}_k, \bar{u}_k, \bar{x}_{k+1})$ were obtained by performing Monte-Carlo sampling on the cart-pole model, and the regression model $\mathcal{F}$ was built using these sample pairs. Then the transferred knowledge Q' was computed based on $\bar{x}_{k+1}$. Here we simply let $Q' = 0.01\bar{x}_{k+1}^2$ and its values corresponds to $\bar{x}_k$ while $\bar{u}_k$ are kept unchanged are shown in Fig. 4.4(a). Apparently, the Q' values are higher when the pole angle θ and pole angular velocity ω share the same sign. For example, when close to the point at $\theta = -0.2$ rad and $\omega = -1$ rad/s, and also the point at $\theta = 0.2$ rad and $\omega = 1$ rad/s, the corresponding Q' values are large, which indicates higher costs for these states. Note that the formulation of the transferred knowledge Q' relied on heuristics, and Q' is partial knowledge of the problem rather than complete knowledge as only partial state (two of the four state



(a) (b)

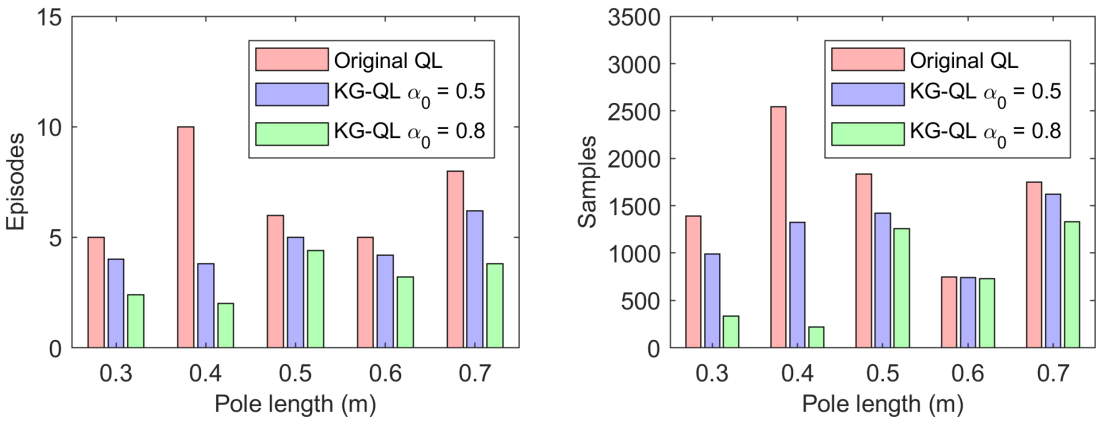
Figure 4.4: Regression Based Q' Values. (a) Inverted Pendulum Cart-Pole Balancing Problem. (b) SWF Phase of Prosthesis Control Problem With Human in the Loop.

variables) is considered in Q'.

With the regression result as a transferred knowledge Q', RL-IT was employed to find optimal solution for the target tasks with various half pole length, e.g. half pole length $l = 0.3$ m, 0.4 m, 0.5 m, 0.6 m, 0.7 m. Fig.4.5 shows that RL-IT needs fewer episodes and samples to find the optimal solution, compared to the original Q-learning algorithm.

4.6.3 Human-Prosthesis Simulation

We used OpenSim [126], a widely used human movements simulation platform, to simulate level-ground walking dynamics of the human-prosthesis system. Fig.4.3(b) shows the OpenSim lower limb walking model. In this model, five rigid-body segments linked by one degree-of-freedom pin joints were used to represent the human body. The right knee was treated as a prosthetic knee and controlled by finite-state



(a) (b)

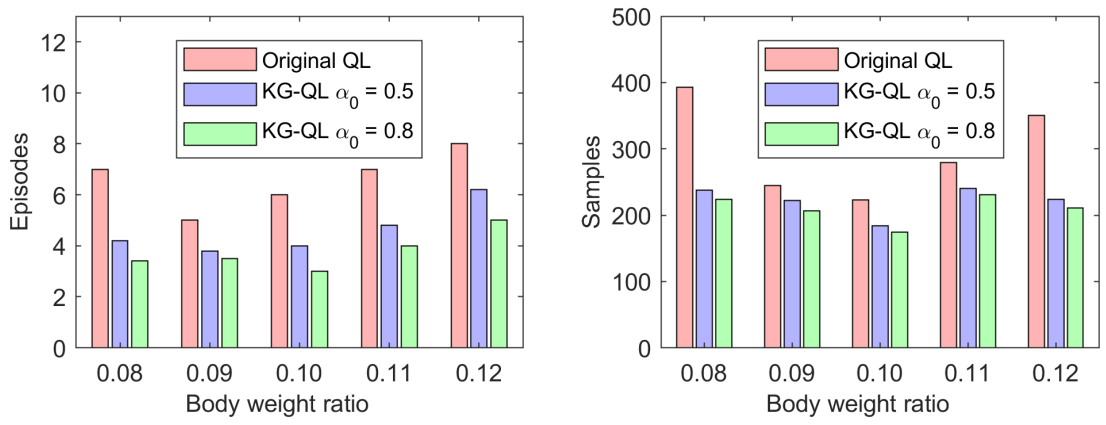
Figure 4.5: Comparison for the Cart-Pole Balancing Problem. (a) Number of Episodes Before Success. (b) Number of Samples Before Success.

impedance controller (FS-IC), while the other joints followed prescribed motions. The FS-IC divides a gait into four phases, in each of the four phases the torque at the prosthetic knee is determined by the impedance control law,

$$T = K(\theta - \theta_e) + B\omega. \tag{4.73}$$

where K, B and θ_e are the impedance parameters: K is stiffness, B is damping coefficient and θ_e, is equilibrium position. After each gait cycle, the differences (errors) between the measured knee angle profile and the target knee profile at the feature points are computed and treated as the state of the ADP controller. The adjustments to the three impedance parameters are the action. More details about this task, that a RL agent is used to tune the prosthetic knee impedance parameters, can be found in [30, 37].

Similar to the cart-pole balancing task, we first collected sample pairs of $(\bar{x}_k, \bar{u}_k, \bar{x}_{k+1})$ from the OpenSim model with a body weight ratio of 0.1. Here k is the index of gait cycle,$\bar{x}_k \in \mathbb{R}$ is peak angle at gait cycle k, and $\bar{u}_k \in \mathbb{R}^3$ is the adjust-



(a) (b)

Figure 4.6: Comparison for the OpenSim Human-Prosthesis Walking Problem. (a) Number of Episodes Before Success. (b) Number of Samples Before Success.

ments to the three impedance parameters. The regreesion model $\mathcal{F}$ can be found using regressions:$\bar{x}_{k+1} = \mathcal{F}(\bar{x}_k, \bar{u}_k)$. Then Q' can be computed based on heuristic formulation based on $\mathcal{F}$ as $Q'(\bar{x}_k, \bar{u}_k) = \mathcal{Q}(\bar{x}_{k+1}) = \mathcal{Q}(\mathcal{F}(\bar{x}_k, \bar{u}_k))$. Here we let $\mathcal{Q}(\bar{x}_{k+1}) = \bar{x}_{k+1}^2$. The values of $Q'(\bar{x}_k, \bar{u}_k)$ that is formulated in this way are visualized in Fig. 4.4(b).

With the regression result as a transferred knowledge Q', RL-IT was employed to find optimal solution for the target tasks with various body weight ratios, e.g. body weight ratio equals 0.08, 0.09, 0.10, 0.11 and 0.12. Fig.4.6 shows that RL-IT needs fewer episodes and samples to find the optimal solution, compared to the original Q-learning algorithm.

4.7 Discussion

In this paper, we developed a new RL-IT framework to perform reinforcement learning with online data as well as transfer knowledge from another similar task.

The transferred knowledge can be obtained either from offline historical data using regression, or from another trained RL agent. We tested our proposed method with three classic RL problems, including windy gridworld, inverted pendulum and a human-prosthesis simulation model. These three problems cover various scenarios ranging from discrete state/action space to continuous state/action space. The simulation results of the three problems validated this new approach and showed that it is more time and sample efficient compared to the naive learner.

Compared to the existing and related works such as reward shaping methods [117, 118] or experience replay [67, 127, 128], we provided a systematic framework RL-IT to utilize transferred knowledge as the regular reinforcement learning agent is learning simultaneously. To be specific, we provided a more complete analytical guarantee on the convergence of the proposed framework. It is showed that with the transferred knowledge, the new agent can still reach an optimal solution monotonically.

The flexibility of RL-IT is presented in three different levels. First, it allows for a flexible knowledge representation Q' in the value function or system dynamics or both. Second, additional flexibility can be achieved by a transfer ratio as designers are allowed to determine how much information can be transferred from the source task to the target task. Third, can be extend to other RL algorithms. Third, our RL-IT control framework focus transfer learning problems that involve the same states and controls. Thus, it can be integrated with other TD-based methods such as SARSA and value iteration, as well as their deep learning variants, to name a few.

The issue of negative transfer is remain an open question in the study of transfer learning. A successful transfer relies on the transferred knowledge is applicable in both source task and target task, otherwise noise will accumulate during learning iterations can lead to negative transfer which can adversely affect performance when more training data is used. We did not specify any specific technique to address

negative transfer, therefore, the results reported in this paper also relied on proper selection of source task and training task.

4.8 Conclusion

In this paper, we present general reinforcement framework RL-IT that learns with both online experiences and transferred knowledge. First, structural knowledge represented as a function of state and action is extracted from a source task. Then a Q-learning based RL algorithm is presented to incorporate the knowledge in Q-value update. The convergence properties of the proposed method is analyzed. To implement RL-IT, a actor-critic structure is developed. Extensive experiments using three simulated classic RL problems verify that RL-IT outperforms its counterpart without transferred knowledge in terms of sample efficiency. In our future work, we plan to extend the results to real robotic experiments with human in the loop.

Chapter 5

KNOWLEDGE-GUIDED REINFORCEMENT LEARNING CONTROL FOR ROBOTIC LOWER LIMB PROSTHESIS

5.1 Abstract

Robotic prostheses provide new opportunities to better restore the lost functions than passive prostheses for transfemoral amputees. But controlling a prosthesis device automatically for individual users in different task environments is an unsolved problem. Reinforcement learning (RL) is a naturally promising tool. For prosthesis control with a user in the loop, it is desirable that the controlled prosthesis can adapt to different task environments as quickly and smoothly as possible. However, most RL agents learn or relearn from scratch when the environment changes. To address this issue, we propose the knowledge-guided Q-learning (KG-QL) control method as a principled way for the problem. In this report, we collected and used data from two able-bodied (AB) subjects wearing a RL controlled robotic prosthetic limb walking on level ground. Our ultimate goal is to build an efficient RL controller with reduced time and data requirement and transfer knowledge from AB subjects to amputee subjects. Toward this goal, we demonstrate its feasibility by employing OpenSim, a well-established human locomotion simulator. Our results show the OpenSim simulated amputee subject improved control tuning performance over learning from scratch by utilizing knowledge transfer from AB subjects. Also in this paper, we will explore the possibility of information transfer from AB subjects to help tuning for the amputee subjects.

5.2 Introduction

The rapid development of robotic prostheses in both research and commercial products brings them closer to real-life scenarios. Compared to passive devices, robotic lower limb prostheses promise to provide better functions to restore natural gaits for amputees, such as decreased metabolic consumption [83], improved adaptation to various terrains [85, 86] or walking speed [88], and enhanced balance and stability [87]. In robotic lower limb prosthetics, finite state impedance control (FS-IC) [90, 6] is still the most common approach in both prototypes or commercial devices. However, in order to maximize the performance for each user, there are a large number of control parameters in these devices need to be tuned by experienced clinicians.

Reinforcement learning allows learning from interacting with the environment to generate suitable actions while maximizing a performance reward in a particular situation. Learning can take place under different formulations of a problem, including learning directly from data without requiring an explicit mathematical description of the environment and the interacting dynamics between the controller and the environment. This has given RL an expanded domain of control applications beyond the capacity of traditional control theory and practice. There have been several successful demonstrations of RL applications to solving challenging robotic control problems. Among those, deep RL methods attracted most attentions. For example, Nair *et al.* [129] employed deep deterministic policy gradients (DDPG) for a robotic arm block stacking task with sparse reward. The authors of [130] proposed deep latent policy gradient (DLPG) for learning locomotion skills. However, deep RL based methods may be not suitable for biomedical applications such as the human-prosthesis control problem being discussed in this paper, because training data involving amputee

subjects are usually difficult to acquire and expensive to collect. Additionally, experimental sessions involving human subjects usually cannot last more than one hour because of human fatigue and safety considerations. To tackle this challenge, we proposed several sample-efficient and easy-to-implement RL methods in our previous works [30]-[131] allowing for directly learning from data. In our application of prosthesis control, it is very common that the robotic prosthesis need to be adapted for a new user. However, these RL methods, as well as most existing RL methods, are designed to learn from scratch whenever a new task or new model is presented, and thus not readily capable of storing and transferring knowledge gained from one subject to another.

It is therefore of high priority that the RL agent is designed to be training time and sample efficient when tuned for a new user. To take advantage of previous knowledge and information, we consider building a representation for potentially transferable knowledge across subjects. In the current study, we consider extracting knowledge from able-bodied (AB) subjects and use that for future RL control design for amputee subjects. It is known that transfer learning has attracted great attention in the machine learning field where it is typically considered for storing knowledge gained while solving one problem and applying it to a different but related problem [43]. In the context of general transfer learning in the literature, our prosthesis parameter tuning problem has the same state and action while the problem calls for gaining knowledge from tuning parameters for AB subjects (source task) and using that for tuning parameters for amputee subjects (target task).

Structural knowledge transfer perhaps has found most of its applications reported in the literature. Barreto *et al.* [44] solved the problem where rewards change but environments remain the same using successor features, a value function representation that decouples the dynamics of the environment from the rewards. Asadi *et al.*

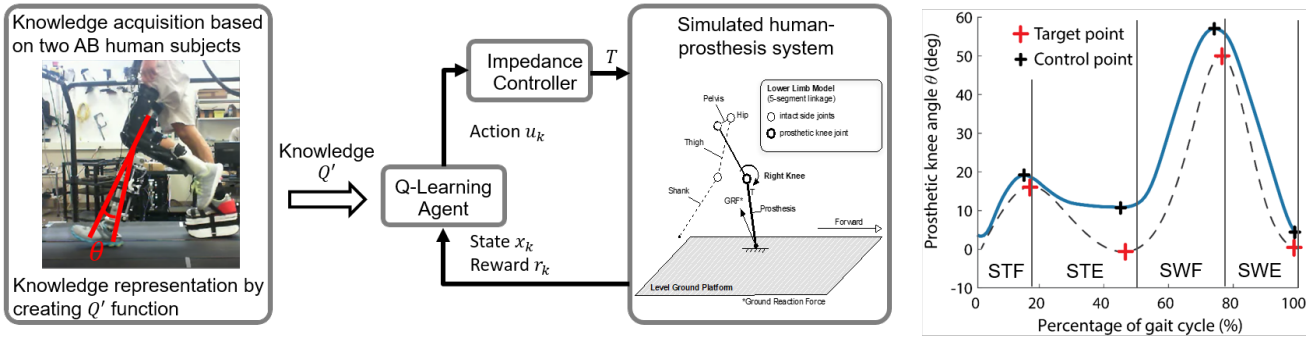


Figure 5.1: Schematic of Knowledge-Guided RL Control. (a) Knowledge Acquisition. (b) Online KG-QL Process. (c) Target and Measured Knee Kinematics.

[45] proposed a learning architecture which transfers control knowledge in the form of behavioral skills and representation hierarchy, which separates the subgoals so that a more compact state space can be learned. In [46], researchers demonstrated that Schema network is capable to perform zero-shot transfer between tasks where cause-effect relationship remains unchanged, such as learning to play the breakout game with different maps. In [47], target apprentice learning is proposed for cross-domain transfer, e.g. from balancing a cart-pole to balancing a bike.

Unlike the above approaches, we propose a new knowledge transfer framework for the class of problems that transfer from a source task to a target task while maintaining the same state and control problems. We built a knowledge representation from AB subjects into the Q-learning update, and the knowledge transfer schedule results in a diminishing influence of previous knowledge which simultaneously allowing for increased attention to learning of the target task on hand. Specifically, we first collected data from AB subjects, then we built regression models on these data, which then became transferred knowledge to guide a Q-learning process, namely our proposed knowledge guided Q-learning (KG-QL) process. Our method introduces two new advances from the existing transfer learning methods. First, we provided a flexible framework where the representation of the transferred knowledge can be either a value function or a regression model or both. Second, the amount of transferred knowledge into a new task can be programmed in a convenient way to address different applications needs.

The structure of this paper is organized as follows: Section II discusses the control framework of human-prosthesis system, the regression based transfer learning method, and the proposed framework of KG-QL for parameter tuning. Experimental results using OpenSim simulations are shown in Section III followed by discussion and future directions in Section IV.

5.3 Methods

5.3.1 Finite State Impedance Control of Human-Prosthesis System

From the perspective of a RL agent, the integrated human-prosthesis system can be treated as a nonlinear dynamic system of the form

$$x_{k+1} = F(x_k, u_k), k = 0, 1, 2, \ldots \tag{5.1}$$

where k is the discrete time index or gait cycle in this study, $x_k \in \mathbb{R}^2$ is the state vector, $u_k \in \mathbb{R}^3$ is the action or control vector, and F describes the intrinsic human-prosthesis system dynamics of how a new state at $k+1$ evolves from a current state and control at k. Specifically, state x_k is defined as the differences (errors) between the measured knee angle profile and the target knee profile at the feature points. The target knee profile is identical to those normative knee kinematics reported in [99]. In Fig. 5.1(c), for each of the four phases there is a pair of such feature points with black and red markers, where their vertical and horizontal differences are peak error $\Delta P_k \in \mathbb{R}$ and duration error $\Delta D_k \in \mathbb{R}$, respectively:

$$x_k = [\Delta P_k, \Delta D_k]^T. \tag{5.2}$$

The RL controller is realized within an established FS-IC platform. In FS-IC, a complete gait cycle is divided into four sequential gait phases based on knee joint kinematics and ground reaction force (GRF) by a finite state machine (FSM). These four gait phases are stance flexion (STF), stance extension (STE), swing flexion (SWF) and swing extension (SWE). In real-time experiments, phase transitions are realized as those in [6] based on Dempster-Shafer theory (DST). In each phase, the prosthetic system mimics a passive spring-damper-system with a group of three predefined impedance parameters as

$$I_k = [K_k, B_k, \theta_{e,k}] \in \mathbb{R}^3, \tag{5.3}$$

where K_k is stiffness, B_k is damping coefficient and $\theta_{e,k}$ is equilibrium position. In other words, for all four phases there are 12 impedance parameters in total with one RL controller being designed for one phase under the FS-IC framework. Without loss of generality, our following discussion on the RL controller design approach applies to all four phases, and therefore, we utilize only one of the four phases. The knee joint torque $T \in \mathbb{R}$ is generated based on the following impedance control law

$$T_k = K_k(\theta - \theta_{e,k}) + B_k\omega. \tag{5.4}$$

Correspondingly, the action u_k of the RL agent is defined as adjustments ΔI_k to the impedance parameters I_k,

$$u_k = \Delta I_k = [\Delta K_k, \Delta B_k, \Delta\theta_{e,k}]. \tag{5.5}$$

5.3.2 Human Gait Data Collection

To perform knowledge transfer from a source task to a target task, first we need to obtain the transferable knowledge, which can be represented in the form of raw data, policy, value function, or others. Here we define a function $Q'(x_k, u_k)$ to store such information for transfer. It takes state and action as input to generate the state-action value function or Q-value function $Q(x, u)$ as in the RL literature. Although Q' can be a previously learned Q-value function from a RL agent, it can also be represented in other forms. In this work, we construct Q' using regression model based on the source task data.

Source task data was collected from two AB participants (both male, 25-35 years old) while walking at a constant speed of 0.6 m/s on a split-belt treadmill with force platforms embedded within each belt. All participants provided written informed consent prior to participating according to protocols approved by the Institutional Review Board at North Carolina State University. A certified prosthetist aligned the

robotic knee prosthesis for each subject. The AB subjects used an L-shape adaptor (with one leg bent 90 degrees) to walk with the robotic knee prosthesis (Fig. 5.1(a)) [37].

The gait data used in this study includes a total of $N = 1120$ pairs of state-action tuples (x_k, u_k) from the two AB subjects (AB1: 480 pairs, AB2: 640 pairs) using the same prosthesis device. During data collection, the prosthesis impedance parameters were controlled by the dHDP based RL approach that we investigated previously [37]. Note that the dHDP was only to provide some control to the prosthesis instead of providing optimal control to achieve a performance measure. In other words, the data were drawn from the online learning process of the dHDP RL controller rather than generated by a well-learned policy to provide sufficient exploration of the control policy space. Actually, a RL controller is not unique for data collection. Any sampling method is acceptable as long as it sufficiently samples the control parameter space, and it maintains practical stability of the human-prosthesis system. Note that during data collection, the impedance parameters I_k were updated every seven gait cycles, and state x_k was averaged by the seven gait cycles conditioned on the same impedance parameters I_k. That is to say, to reduce step-by-step variability in feature measurements, the time index k here corresponds to every seven gait cycles.

5.3.3 Extracting Knowledge from Human Gait Data

We performed linear regression to establish a relationship between prosthesis impedance parameters and the human-prosthesis system kinematics as follows,

$$x_{k+1} = \mathcal{F}(x_k, u_k) = \mathcal{F}(z_k) = \beta_0 + \beta_1 z_k + e, \tag{5.6}$$

where $x_{k+1} \in \mathbb{R}^2$ is the next state, $\beta_0 \in \mathbb{R}$ is the intercept, $\beta_1 \in \mathbb{R}^{2\times 5}$ is the regression coefficient (or the slope), $z_k = [x_k, u_k] \in \mathbb{R}^5$ is the predictor variable formed by the

current state x_k and action u_k, and $e \in \mathbb{R}^2$ is the error term. Least-square solution of the coefficients β_0 and β_1 can be found using the (x_k, u_k, x_{k+1}) tuples. Equation (6) characterizes the human-prosthesis system qualitatively because when a controller enables the human-prosthesis system to generate improved locomotion performance, we generally observe that $|x_{k+1}| \leq |x_k|$.

After the regression model $\mathcal{F}$ is obtained, we can formulate $Q'(x_k, u_k)$ based on $\mathcal{F}(x_k, u_k)$. How Q' is formulated also relates to the the stage reward or cost in RL. In our work, we set the stage cost variable $r_k = 0$ for success and $r_k = 1$ for failure (see (5.9)), which implied that the goal for the RL agent was to minimize the total cost-to-go. Accordingly, inspired by LQR control objective function, Q' was formulated as a quadratic form such that $Q' \geq 0$,which was consistent with $r_k \geq 0$ and $Q_i \geq 0$ (Q_i is the the iterative Q-value function defined in (5.14) and (5.15)):

$$Q' = 0.02x_{k+1}^2 = 0.02(\mathcal{F}(x_k, u_k))^2. \tag{5.7}$$

Note that here the form of Q' was manually defined and was not unique. The ratio of 0.02 was set manually to make Q' within a comparable range of the stage cost r_k. As shown later, knowledge represented in Q' can be adopted by the designer at a preferred rate. Fig. 5.2 Illustrates kinematics and Q-values as knowledge representations.

5.3.4 Knowledge Guided Reinforcement Learning

We have introduced how the transferred knowledge Q' is obtained through regression. Now we can move onto the online learning process of the RL agent as shown in Fig. 5.1(b). We call this RL algorithm a knowledge-guided Q-learning algorithm (KG-QL) because when the Q-learning agent is determining a best action for the next step, its decision is guided and biased by the transferred knowledge Q'.

At time index k, the RL agent starts from state x_k and takes the action u_k. Then

Algorithm 5.1 Knowledge Guided Q-Learning (KG-QL) for prosthesis control with a human in the loop

Input: Transferred knowledge Q' from a source task

Initialization: Initialize actor NN and critic NN with random weights. Loop for each episode:

Random initial state x_0;

Loop for each step k:

Generate u_k from x_k according to actor NN (ϵ-greedy).

Take action u_k, observe cost r_k and next state x_{k+1}.

Update actor weights to minimize the actor prediction error(5.21)(5.22).

Update critic weights to minimize the critic prediction error (5.23)(5.24).

$x_{k+1} \leftarrow x_k$

Until x_k is terminal

it ends up at the next state x_{k+1}, and receives a cost r_k. This process repeats for $k = 1, 2, ...$ until a terminal state is reached. The total cost-to-go function or value function is defined as

$$J(x_k, u_k) = \sum_{j=k}^{\infty} \gamma^{j-k} r_k, \tag{5.8}$$

where $r_k = r(x_k, u_k)$ is the stage cost, and γ is the discount factor with $0 < \gamma < 1$. In our work, we defined r_k as

$$r_k = r(x_k, u_k) = \begin{cases} 0, & \text{if } x_{k+1} \text{ is a success state} \\ 1, & \text{if } x_{k+1} \text{ is a failure state} \\ 0.01, & \text{otherwise} \end{cases} \tag{5.9}$$

In this work, a success state is defined as when the state is in the target range, and a failure state is defined as the state is out of the safety range. Further details can be found in 5.4.1.

Equation (5.8) can be written as

$$J(x_k, u_k) = r_k + \gamma J(x_{k+1}, u_{k+1}). \tag{5.10}$$

According to Bellman's optimality principle [132], the optimal cost function J^* satisfies the action dependent discrete time Hamilton–Jacobi–Bellman (HJB) equation

$$J^*(x_k, u_k) = r_k + \gamma \min_{u_{k+1}} J^*(x_{k+1}, u_{k+1}). \tag{5.11}$$

Besides, the optimal control π^* can be expressed as

$$\pi^*(x_k) = \arg\min_{u_k} J^*(x_k, u_k). \tag{5.12}$$

By substituting (5.12) into (5.11), the discrete time HJB equation becomes

$$J^*(x_k, u_k) = r_k + \gamma J^*(x_{k+1}, \pi^*(x_{k+1})). \tag{5.13}$$

For a Q-learning agent, we have the following actor-critic structure,

$$\pi_i(x_k) = \arg\min_{u_k} Q_i(x_k, u_k), \tag{5.14}$$

$$Q_{i+1}(x_k, u_k) = r_k + \gamma Q_i(x_{k+1}, \pi_i(x_{k+1})), \tag{5.15}$$

where i is the iterative index, π_i and Q_i are the iterative control policy and iterative Q-value function, respectively. Combining (5.14) and (5.15), we have

$$Q_{i+1}(x_k, u_k) = r_k + \gamma \min_{u_{k+1}} Q_i(x_{k+1}, u_{k+1}). \tag{5.16}$$

Accordingly, the *knowledge-guided* form of Q-learning can be written as

$$\pi_i(x_k) = \arg\min_{u_k} [Q_i(x_k, u_k) + \alpha_i Q'(x_k, u_k)], \tag{5.17}$$

$$\begin{aligned} Q_{i+1}(x_k, u_k) = r_k + \gamma [&Q_i(x_{k+1}, \pi_i(x_{k+1})) \\ &+ \alpha_i Q'(x_{k+1}, \pi_i(x_{k+1}))], \end{aligned} \tag{5.18}$$

where Q' is an arbitrary positive semi-definite function that represents previously learned knowledge, and $0 \leq \alpha_i \leq 1$ is a weighting factor such that $\alpha_{i+1} \leq \alpha_i$, and $\alpha_i \to 0$ when $i \to \infty$. Here we simply let α_i be a uniformly decreasing sequence of 0.5,0.49,0.48,...,0 as i increases. Combining the above two equations, we have

$$\begin{aligned} Q_{i+1}(x_k, u_k) = r_k + \gamma \min_{u_{k+1}} [Q_i(x_{k+1}, u_{k+1}) \\ + \alpha_i Q'(x_{k+1}, u_{k+1})] \end{aligned} \tag{5.19}$$

5.3.5 *Actor-Critic Implementation*

The proposed KG-QL method was implemented by an actor-critic structure [125, 25]. Specifically, (5.17) was implemented by an actor, and (5.18) was implemented by a critic. Both actor and critic were feed-forward neural networks (NN) with one hidden layer (5-6-1 for the critic, and 2-6-3 for the actor). The critic has the state x_k and the action u_k as inputs, and outputs an approximation of the Q-value function, denoted by $\hat{Q}(x_k, u_k)$. The actor has state x_k as inputs, and outputs the control action u_k. The actor used a tangent sigmoid activation function $\varphi(v)$ in both the hidden layer and output layer,

$$\varphi(v) = \frac{1 - \exp(-v)}{1 + \exp(-v)} \tag{5.20}$$

where v is the input vector for the activation function. Note that $-1 < \varphi(v) < 1$. For the critic, it also used the same tan-sigmoid function $\varphi(v)$ in its hidden layer. But it used a linear activation function $\phi(v) = v$ in its output layer.

During training, the actor and critic back-propagated their performance error to updated their weights. The prediction error of the actor $e_{a,k} \in \mathbb{R}$ is to realize (5.17),

$$e_{a,k} = \hat{Q}_i(x_k, u_k) + \alpha_i Q'(x_k, u_k). \tag{5.21}$$

Then the squared error E_a for the actor is

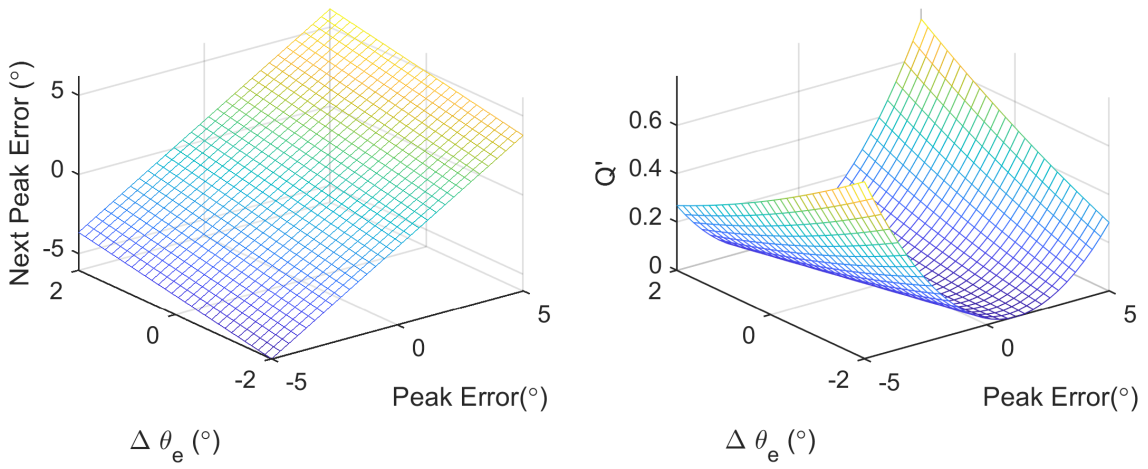
$$E_a = \frac{1}{2} e_{a,k}^2. \tag{5.22}$$

The prediction error of the critic $e_{c,k} \in \mathbb{R}$ is the temporal difference (TD) error of (5.18),

$$\begin{aligned} e_{c,k} = r_k + \gamma[\hat{Q}_i(x_{k+1}, \pi_i(x_{k+1})) \\ + \alpha_i Q'(x_{k+1}, \pi_i(x_{k+1}))] - \hat{Q}_{i+1}(x_k, u_k) \end{aligned} \tag{5.23}$$

which is the difference between the left-hand side and right-hand side of (5.18). To correct the prediction error, the weight update objective was to minimize the squared performance error E_c,

$$E_c = \frac{1}{2} e_{c,k}^2. \tag{5.24}$$

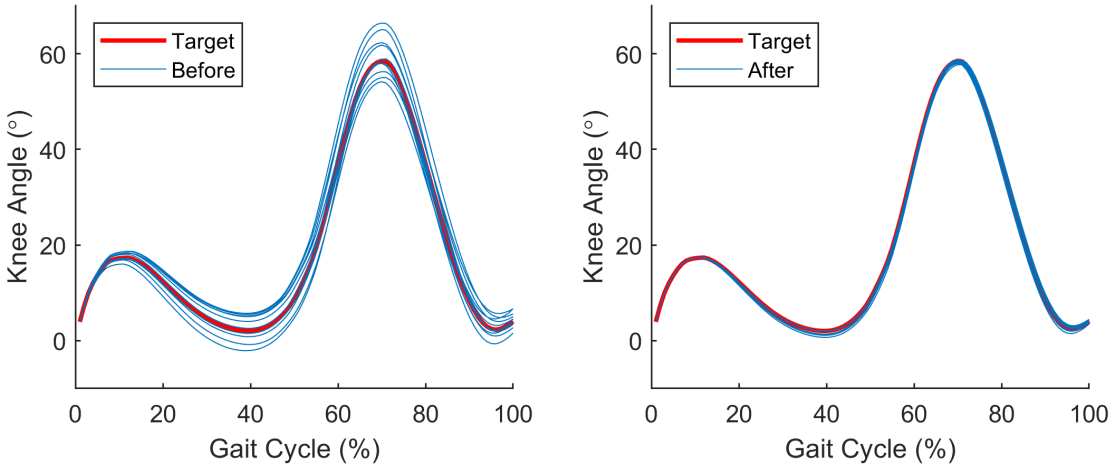


(a) (b)

Figure 5.2: Knowledge Extraction and Representation Based on AB Human Subjects. Data Shown Here is from the SWF Phase. (a) The Regression Model in (5.6). (b) Knowledge Representation in the Form of Q' in (5.7).

5.4 Results

Here we demonstrate how knowledge obtained from previous experiences at a source task can be transferred to a target task where in this study, we extracted



(a) (b)

Figure 5.3: Knee Angle Profiles. (a) Before Tuning (b) After Tuning

knowledge from AB subjects and transferred such knowledge to a target task, which is an OpenSim simulated amputee subject.

5.4.1 OpenSim Experiment Setup

The OpenSim lower limb walking model (Fig. 5.1(a)) used in this work is adopted from [9] and identical to the one in [30]. We defined a target range of $\pm 1^\circ$ and ± 0.01 s for peak error ΔP_k and duration error ΔD_k, respectively. Only if for all four phases both $|\Delta P_k| < 1^\circ$ and $|\Delta D_k| < 0.01$s were met then we said the state x_k was in the target range. If $|\Delta P_k|$ or $|\Delta D_k|$ are greater than some preset values, then state x_k was out of the safety range and the control system resets to the default position of initial impedance parameters to ensure human subject safety. More details about the target range and safety range can be found in our previous work [30, Table I]. An episode is the process from learning step $k = 0$ until termination which can either be that the state x_k enters the target range for 10 consecutive gait cycles or runs out

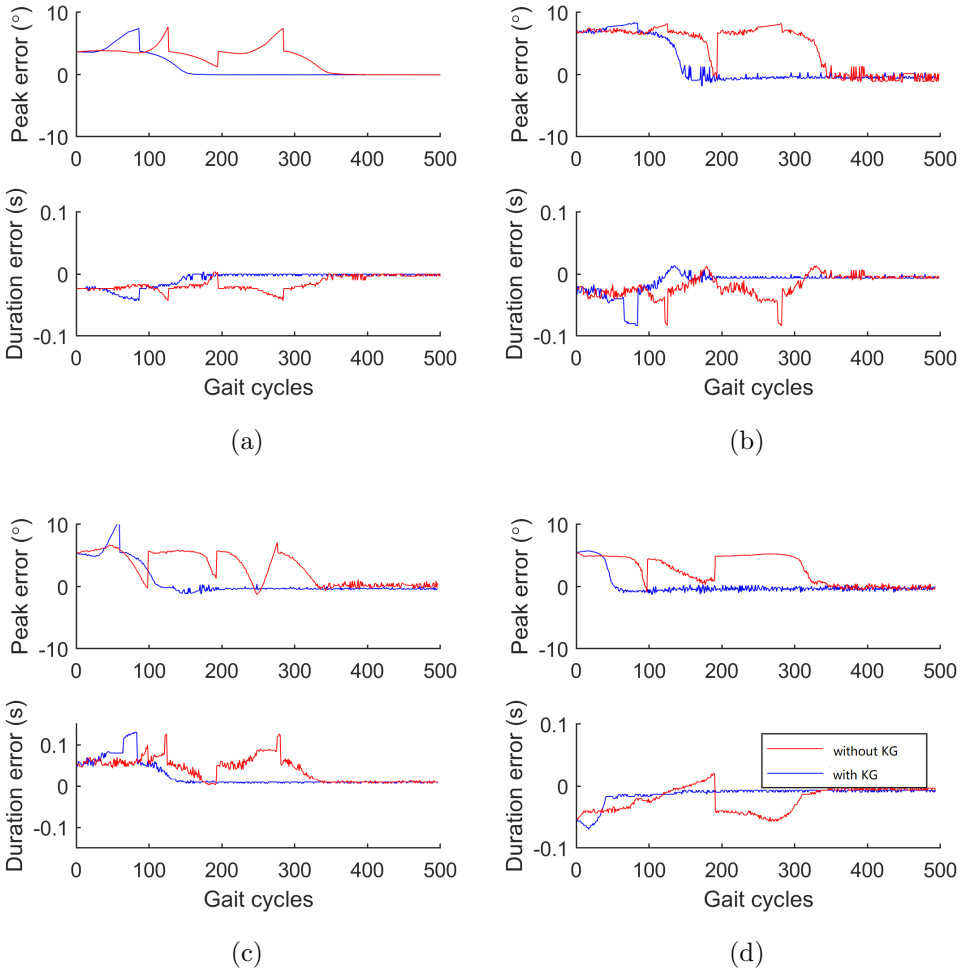


Figure 5.4: Evolution of States in the Four Gait Phases (a) Phase STF (b) Phase STE (c) Phase SWF (d) Phase SWE.

of safety range. If terminated, the state x_k was reset with the initial impedance and initial state as the next episode began. Each OpenSim session consisted of multiple episodes with a total of maximum 500 gait cycles.

The common parameters used in the OpenSim experiment are listed as follows except those mentioned elsewhere. The discount factor γ was set to 0.95, the initial NN weights for both actor and critic were uniformly distributed between -1 and 1.

5.4.2 Knowledge Representation Results

Fig. 5.2 depicts the regression results data from two AB subjects in the SWF phase. In Fig 5.2(a), the z-axis is the next peak error ΔP_{k+1}, which is the first element of the next state x_{k+1}. Its values were obtained from the linear regression model (5.6) by varying one of the state variable peak error ΔP_k and one of the action variable $\Delta\theta_{e,k}$, while other state and action variables remain unchanged. We can learn how the next peak angle θ_{k+1} may change from Fig 5.2(a). For example, suppose $\Delta P_k = -5^\circ$. If $\Delta\theta_{e,k} = -2^\circ$, then $\Delta P_{k+1} < -5^\circ$ according to Fig 5.2(a). Vice versa, if $\Delta\theta_{e,k} = 2^\circ$, then $\Delta P_{k+1} > -5^\circ$. So 2° may be a better choice than -2° for $\Delta\theta_{e,k}$ in this case, as it makes the deviation of the next peak error ΔP_{k+1} smaller. Fig 5.2(b) shows the values of Q' which are computed from (5.7). Q' has a minimum value 0 at $(0, 0)$. Larger Q' value indicates greater cost, which is unfavorable by the RL agent.

5.4.3 Results of Reinforcement Learning with Knowledge Transfer

Fig. 5.3 shows the knee kinematics with different initial impedance parameters in the 10 simulation sessions were distant from the target profile, especially the peak angle errors. Clearly, after the impedance parameters were adjusted by the proposed RL controller, knee kinematics of the final acclimation stages approached the target

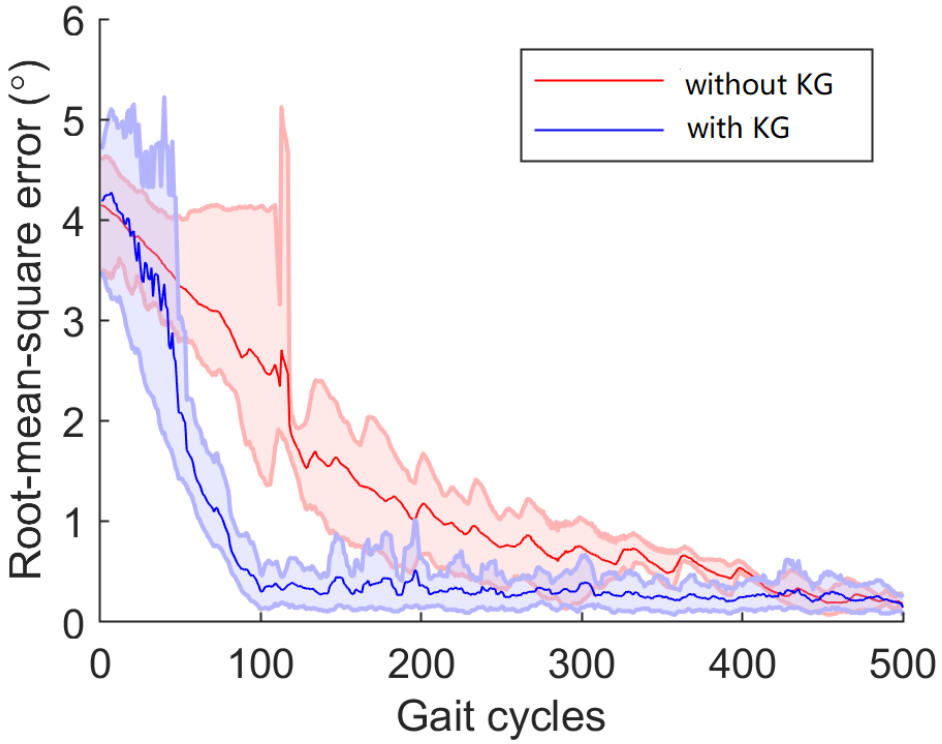


Figure 5.5: Comparison of Root-Mean-Square Error (RMSE) for the With Knowledge Guide Case and the Without Knowledge Guide Case.

points. Specifically, the averaged absolute values of the peak errors over the three sessions deceased from $1.23^\circ \pm 0.77^\circ$ to $0.36^\circ \pm 0.32^\circ$ for STF, from $3.13^\circ \pm 0.31^\circ$ to $0.52^\circ \pm 0.24^\circ$ for STE, from $5.53^\circ \pm 0.89^\circ$ to $0.63^\circ \pm 0.68^\circ$ degrees for SWF and from $2.72^\circ \pm 1.67^\circ$ to $0.12^\circ \pm 0.25^\circ$ for SWE. The results indicate that the proposed knowledge guided QL controller is able to adjust the prosthetic knee kinematics to reproduce the target knee profile under different initial conditions.

Fig. 5.4 illustrates the evolution of the state, i.e. peak errors ΔP_k and duration errors ΔD_k, during the experimental session under one of the sets of initial parameters. Since similar results were obtained from other experiment sessions, hereafter we only present the result from the first session as an example. Because all four phases

were tuned simultaneously, the parameter changes in one phase would affect its subsequent phases. In Fig. 5.4, notice that the sharp edges on the curves indicate the impedance parameters being reset to their initial values, because failure occurred. In this example episode, the KG-QL agent was able to reduce all peak errors and duration errors to zero after approximately 150 gait cycles.

Fig. 5.5 illustrates the averaged root-mean-square error (RMSE) of the gait knee profile over the 10 experimental sessions. With the transferred knowledge from AB subjects, the RMSE of the proposed KG-QL algorithm drops faster than the QL without knowledge transfer, i.e., learning with $\alpha_i = 0$ in (5.18). Our proposed KG-QL achieved a RMSE performance less than 1° after only 100 gait cycles, however, Q-learning without knowledge transfer can only achieved similar performance after about 400 gait cycles. Fig. 5.4 and Fig. 5.5 show that the target task time was significantly reduced with knowledge transfer.

5.5 Discussions and Conclusions

We developed a new KG-QL framework to integrate and transfer knowledge from AB subjects to OpenSim simulated amputee subjects with a common goal of optimizing impedance parameters for robotic knee prosthesis. The knowledge for transfer can be obtained offline using historical data, aka, from AB subjects in our study, to facilitate online reinforcement learning for amputee subjects. Our OpenSim simulation results validated this new approach and showed that our new scheme can help restore near-normal knee kinematics, in a time and sample-efficient manner compared to the naive learner. Our results suggested that the proposed KG-QL controller is a promising new framework when performing the cross-subject learning for the robotic knee prosthesis with human in the loop. Our demonstrated effectiveness of transfer learning from AB subjects to OpenSim simulated amputee subject may be due to

the fundamental principle guiding human gaiting. Or in other words, the underlying physiology and physics represented in the relationships from impedance parameters in the FS-IC to knee joint torque and further to locomotion, should be preserved in both AB subjects and the OpenSim simulated amputee subjects, where in the latter case, the forward dynamics model should capture such relationships.

Based on experimental measurements from two AB subjects, we established a knowledge representation in the form of a regression model of the human-prosthesis dynamics, and a Q-value integration of this knowledge for transferring to the target task. We demonstrated the effectiveness of this KG-QL control framework. Our results show that, with transferred knowledge, QL was able to reach a comparable performance in the target task of an OpenSim simulated subject, but saving at least 60% of the learning time.

Our contribution is not limited to the demonstration of the feasibility of such transfer learning. It also includes our proposed RL control design framework that allows for flexible knowledge representation in the value function or system dynamics or both. In addition, we provided additional flexibility by allowing for a designer to determine how much information can be transferred from the source task to the target task.

Our KG-RL control framework provides a principled way to solving transfer learning problems that involve the same states and controls. Thus, it can be integrated with other TD-based methods such as SARSA and value iteration, as well as their deep learning variants, to name a few. In our future work, we would like to test this method on human amputee subjects using knowledge from AB subjects.

Chapter 6

CONCLUDING REMARKS

This thesis focus on improving data efficiency of RL control on robotic lower limb prostheses. This goal is achieved from two aspects. 1) Flexible policy iteration based RL algorithm (Chapter 2) achieves data efficiency by reusing prior knowledge from the same task and by a batch learning scheme, and 2) when cross-subject knowledge is available, KG-RL can use such knowledge to guide a regular learning process. Results demonstrate the efficiency of both methods.

In the future, we will explore other options of performance goals, such as gait symmetry, stability, and user feedback. Taking these measures into considerations for the optimization of the robotic prosthesis, we may reach a better accepted optimization goal which reflects the performance of the whole human-prosthesis system, rather than a kinematic trajectory of the robotic knee.

References

[1] K. Ziegler-Graham, E. J. MacKenzie, P. L. Ephraim, T. G. Travison, and R. Brookmeyer, "Estimating the prevalence of limb loss in the united states: 2005 to 2050," *Archives of Physical Medicine and Rehabilitation*, vol. 89, pp. 422–429, Mar. 2008.

[2] N. Thatte, *Design and Evaluation of Robust Control Methods for Robotic Transfemoral Prostheses*. PhD thesis, Carnegie Mellon University, 2019.

[3] H. Huang, D. L. Crouch, M. Liu, G. S. Sawicki, and D. Wang, "A cyber expert system for auto-tuning powered prosthesis impedance control parameters," *Annals of Biomedical Engineering*, vol. 44, pp. 1613–1624, May 2016.

[4] F. Sup, H. A. Varol, J. Mitchell, T. J. Withrow, and M. Goldfarb, "Self-contained powered knee and ankle prosthesis: Initial evaluation on a transfemoral amputee," in *2009 IEEE International Conference on Rehabilitation Robotics, ICORR 2009*, 2009.

[5] B. E. Lawson, H. A. Varol, A. Huff, E. Erdemir, and M. Goldfarb, "Control of stair ascent and descent with a powered transfemoral prosthesis," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 21, pp. 466–473, Oct. 2013.

[6] M. Liu, F. Zhang, P. Datseris, and H. H. Huang, "Improving finite state impedance control of active-transfemoral prosthesis using dempster-shafer based state transition rules," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 76, pp. 461–474, Dec. 2014.

[7] S. Seth, A., Hicks J.L., Uchida, T.K., Habib, A., Dembia, C.L., Dunne, J.J., Ong, C.F., DeMers, M.S., Rajagopal, A., Millard, M., Hamner, S.R., Arnold, E.M., Yong, J.R., Lakshmikanth, S.K., Sherman,n M.A., Delp, "Opensim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement.," *Plos Computational Biology*, vol. 14, no. 7, 2018.

[8] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, D. G. Thelen, and D. Delp, S.L., Anderson, F.C., Arnold, A.S., Loan, P., Habib, A., John, C.T., Guendelman, E., Thelan, "Opensim: Open-source software to create and analyze dynamic simulations of movement," *IEEE Transactions on Biomedical Engineering*, vol. 55, pp. 1940–1950, Nov. 2007.

[9] Jennifer Hicks, "From the ground up: Building a passive dynamic walker model," 2014.

[10] N. Hogan, "Impedance control: An approach to manipulation: Part iii applications," *J. Dyn. Syst. Meas. Control*, vol. 107, p. 17, Mar. 1985.

[11] E. J. Rouse, L. M. Mooney, and H. M. Herr, "Clutchable series-elastic actuator: Implications for prosthetic knee design," *Int. J. Robot. Res.*, vol. 33, pp. 1611–1625, Oct. 2014.

[12] M. F. Eilenberg, H. Geyer, and H. Herr, "Control of a powered ankle-foot prosthesis based on a neuromuscular model," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 18, pp. 164–173, Apr. 2010.

[13] S. Pfeifer, H. Vallery, M. Hardegger, R. Riener, and E. J. Perreault, "Model-based estimation of knee stiffness," *IEEE Transactions on Biomedical Engineering*, vol. 59, pp. 2604–2612, Sept. 2012.

[14] K. Shamaei, G. S. Sawicki, and A. M. Dollar, "Estimation of quasi-stiffness of the human hip in the stance phase of walking," *PLoS ONE*, 2013.

[15] R. D. Gregg and J. W. Sensinger, "Towards biomimetic virtual constraint control of a powered prosthetic leg," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 1, pp. 246–254, 2014.

[16] A. M. Simon, K. A. Ingraham, N. P. Fey, S. B. Finucane, R. D. Lipschutz, A. J. Young, and L. J. Hargrove, "Configuring a powered knee and ankle prosthesis for transfemoral amputees within five specific ambulation modes," *PLoS One*, vol. 9, p. e99387, June 2014.

[17] W. Felt, J. C. Selinger, J. M. Donelan, and C. D. Remy, ""body-in-the-loop": Optimizing device parameters using measures of instantaneous energetic cost," *PLoS One*, vol. 10, pp. 1–21, Aug. 2015.

[18] J. Zhang, P. Fiers, K. A. Witte, R. W. Jackson, K. L. Poggensee, C. G. Atkeson, and S. H. Collins, "Human-in-the-loop optimization of exoskeleton assistance during walking," *Science*, vol. 1284, no. June, pp. 1280–1284, 2017.

[19] Y. Ding, M. Kim, S. Kuindersma, and C. J. Walsh, "Human-in-the-loop optimization of hip assistance with a soft exosuit during walking," *Sci. Robot.*, vol. 5438, no. February, pp. 1–9, 2018.

[20] R. Pinsler, R. Akrour, T. Osa, J. Peters, and G. Neumann, "Sample and feedback efficient hierarchical reinforcement learning from human preferences," in *Proceedings - IEEE International Conference on Robotics and Automation*, (Brisbane, Australia), pp. 596–601, 2018.

[21] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, pp. 1334–1373, Jan. 2016.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. a. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[23] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[24] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, and L. Sifre, "Article mastering the game of go without human knowledge," *Nature Publishing Group*, vol. 550, no. 7676, pp. 354–359, 2017.

[25] J. Si, A. G. Barto, W. B. Powell, and D. C. Wunsch, *Handbook of learning and approximate dynamic programming*. IEEE Press, 2004.

[26] L. Frank and D. Liu, eds., *Reinforcement learning and approximate dynamic programming for feedback control*. Piscataway, NJ: John Wiley & Sons, 2012.

[27] C. Lu, J. Si, and X. Xie, "Direct heuristic dynamic programming for damping oscillations in a large power system," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 38, pp. 1008–1013, Aug. 2008.

[28] W. Guo, F. Liu, J. Si, D. He, R. Harley, and S. Mei, "Approximate dynamic programming based supplementary reactive power control for dfig wind farm to enhance power system stability," *Neurocomputing*, vol. 170, pp. 417–427, Dec. 2015.

[29] W. Guo, F. Liu, J. Si, D. He, R. Harley, and S. Mei, "Online supplementary adp learning controller design and application to power system frequency control with large-scale wind energy integration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 8, pp. 1748–1761, 2016.

[30] Y. Wen, J. Si, X. Gao, S. Huang, and H. H. Huang, "A new powered lower limb prosthesis control framework based on adaptive dynamic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 2215–2220, Sept. 2017.

[31] D. Liu, Q. Wei, and P. Yan, "Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems," vol. 45, no. 12, pp. 1577–1591, 2015.

[32] B. Luo, H.-N. Wu, and T. Huang, "Off-policy reinforcement learning for h $$ control design," *IEEE Transactions on Cybernetics*, vol. 45, no. 1, pp. 65–76, 2015.

[33] H. Modares, F. L. F. Lewis, and M. B. Naghibi-Sistani, "Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, pp. 1513–1525, Oct. 2013.

[34] Q. Wei, B. Li, and R. Song, "Discrete-time stable generalized self-learning optimal control with approximation errors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 1226–1238, Apr. 2018.

[35] Z. Ni, H. He, D. Zhao, X. Xu, and D. V. Prokhorov, "Grdhp: A general utility function representation for dual heuristic dynamic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, pp. 614–627, July 2015.

[36] Y. Wen, M. Liu, J. Si, and H. H. Huang, "Adaptive control of powered transfemoral prostheses based on adaptive dynamic programming," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, vol. 2016-Octob, pp. 5071–5074, 2016.

[37] Y. Wen, J. Si, A. Brandt, X. Gao, and H. Huang, "Online reinforcement learning control for the personalization of a robotic knee prosthesis," *IEEE Trans. Cybern.*, pp. 1–11, Jan. 2019.

[38] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, Dec. 2004.

[39] E. Soria Olivas, J. D. Martin-Guerrero, M. Martinez, R. Magdalena, and A. J. Serrano, *Handbook of research on machine learning applications and trends : algorithms, methods and techniques*. Information Science Reference, 1 ed., 2010.

[40] P. Hao, G. Zhang, L. Martinez, and J. Lu, "Regularizing knowledge transfer in recommendation with tag-inferred correlation," *IEEE Transactions on Cybernetics*, vol. 49, pp. 83–96, Jan. 2019.

[41] J. Li, K. Lu, Z. Huang, L. Zhu, and H. T. Shen, "Transfer independently together: A generalized framework for domain adaptation," *IEEE Transactions on Cybernetics*, vol. 49, pp. 2144–2155, June 2019.

[42] G. Sun, C. Yang, J. Liu, L. Liu, X. Xu, and H. Yu, "Lifelong metric learning," *IEEE Transactions on Cybernetics*, vol. 49, pp. 3168–3179, Aug. 2019.

[43] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains : A survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[44] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. Van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017.

[45] M. Asadi and M. Huber, "Effective control knowledge transfer through learning skill and representation hierarchies," in *IJCAI International Joint Conference on Artificial Intelligence*, 2007.

[46] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George, "Schema networks: Zero-shot transfer with a generative causal model of physics intuitive," in *34th International Conference on Machine Learning, ICML 2017*, 2017.

[47] G. Joshi and G. Chowdhary, "Cross-domain transfer in reinforcement learning using target apprentice," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2018.

[48] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *Euromicro Int. Conf. Parallel, Distrib. Network-Based Process.*, pp. 500–507, IEEE, Feb. 2014.

[49] R. Enns and J. Si, "Helicopter flight control design using a learning control approach," in *Proc. 39th IEEE Conf. Decis. Control*, vol. 2, pp. 1754–1759, IEEE, 2000.

[50] R. Enns and J. Si, "Apache helicopter stabilization using neural dynamic programming," *J. Guid. Control. Dyn.*, vol. 25, pp. 19–25, Jan. 2002.

[51] R. Enns and J. Si, "Helicopter trimming and tracking control using direct neural dynamic programming," *IEEE Trans. Neural Networks*, vol. 14, pp. 929–939, July 2003.

[52] J. Si and Y. T. Wang, "On-line learning control by association and reinforcement," *IEEE Trans. Neural Networks*, vol. 12, pp. 264–276, Mar. 2001.

[53] F. Liu, J. Sun, J. Si, W. Guo, and S. Mei, "A boundedness result for the direct heuristic dynamic programming," *Neural Networks*, vol. 32, pp. 229–235, Aug. 2012.

[54] B. Scherrer BrunoScherrer, M. Ghavamzadeh MohammadGhavamzadeh, V. Gabillon, B. Lesner, M. Geist, B. Scherrer, M. Ghavamzadeh, and M. Geist Scherrer, "Approximate modified policy iteration and its application to the game of tetris," *Journal of Machine Learning Research*, vol. 16, pp. 1629–1676, Aug. 2015.

[55] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 3, pp. 621–634, 2014.

[56] Q. Wei and D. Liu, "A novel policy iteration based deterministic q-learning for discrete-time nonlinear systems," *Science China Information Sciences*, vol. 58, pp. 1–15, Dec. 2015.

[57] Q. Wei, D. Liu, Q. Lin, and R. Song, "Discrete-time optimal control via local policy iteration adaptive dynamic programming," *IEEE Transactions on Cybernetics*, vol. 47, pp. 3367–3379, Oct. 2017.

[58] W. Guo, J. Si, F. Liu, and S. Mei, "Policy approximation in policy iteration approximate dynamic programming for discrete-time nonlinear systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, pp. 2794 – 2807, July 2018.

[59] L. J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning*, 1992.

[60] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *AAAI Conf. Artif. Intell.*, pp. 3302–3309, 2018.

[61] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proceedings of International Conference on Learning Representations (ICLR)*, Nov. 2015.

[62] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver, "Distributed prioritized experience replay," in *Int. Conf. Learn. Represent.*, Mar. 2018.

[63] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Adv. Neural Inf. Process. Syst.*, 2017.

[64] S. Adam, L. Busoniu, and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 42, no. 2, pp. 201–212, 2012.

[65] B. Luo, Y. Yang, and D. Liu, "Adaptive q-learning for data-based optimal output regulation with experience replay," *IEEE Transactions on Cybernetics*, 2018.

[66] H. Modares, F. L. Lewis, and M. B. Naghibi-Sistani, "Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems," *Automatica*, 2014.

[67] D. Zhao, Q. Zhang, D. Wang, and Y. Zhu, "Experience replay for optimal control of nonzero-sum game systems with unknown dynamics," *IEEE Transactions on Cybernetics*, vol. 46, pp. 854–865, Mar. 2016.

[68] X. Yang and H. He, "Adaptive critic learning and experience replay for decentralized event-triggered control of nonlinear interconnected systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2019.

[69] Y. Hu, Y. Gao, and B. An, "Learning in multi-agent systems with sparse interactions by knowledge transfer and game abstraction," in *Proc. Int. Jt. Conf. Auton. Agents Multiagent Syst. AAMAS*, 2015.

[70] P. Mannion, S. Devlin, J. Duggan, and E. Howley, "Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning," *Knowl. Eng. Rev.*, 2018.

[71] K. R. Dixon, R. J. Malak, and P. K. Khosla, "Incorporating prior knowledge and previously learned information into reinforcement learning agents," tech. rep., 2000.

[72] D. L. Moreno, C. V. Regueiro, R. Iglesias, and S. Barro, "Using prior knowledge to improve reinforcement learning in mobile robotics," in *Towar. Auton. Robot. Syst.*, 2004.

[73] S. Koenig and R. G. Simmons, "The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms," *Mach. Learn.*, vol. 22, pp. 227–250, 1996.

[74] S. Gelly and D. Silver, "Combining online and offline knowledge in uct," in *ACM Int. Conf. Proceeding Ser.*, 2007.

[75] F. Sup, H. A. H. Varol, J. Mitchell, T. J. T. T. J. Withrow, and M. Goldfarb, "Preliminary evaluations of a self-contained anthropomorphic transfemoral prosthesis," *IEEE/ASME Transactions on Mechatronics*, vol. 14, pp. 667–676, Dec. 2009.

[76] E. J. Rouse, L. J. Hargrove, E. J. Perreault, and T. a. Kuiken, "Estimation of human ankle impedance during the stance phase of walking," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, pp. 870–878, July 2014.

[77] Y. Wen, A. Brandt, M. Liu, H. Helen, and J. Si, "Comparing parallel and sequential control parameter tuning for a powered knee prosthesis," *IEEE Int., Conf. Sys., Man and Cybern.*, pp. 1716–1721, 2017.

[78] E. C. Martinez-villalpando and H. Herr, "Agonist-antagonist active knee prosthesis: A preliminary study in level-ground walking," *J. Rehabil. Res. Dev.*, vol. 46, pp. 361–373, Feb. 2009.

[79] B. Luo, D. Liu, T. Huang, and D. Wang, "Model-free optimal tracking control via critic-only q-learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[80] A. Brandt, Y. Wen, M. Liu, J. Stallings, and H. H. Huang, "Interactions between transfemoral amputees and a powered knee prosthesis during load carriage," *Scientific Reports*, vol. 7, Nov. 2017.

[81] R. Hafner and M. Riedmiller, "Reinforcement learning in feedback control : Challenges and benchmarks from technical process control," *Mach. Learn.*, vol. 84, pp. 137–169, July 2011.

[82] M. Li, X. Gao, W. Yue, S. Jennie, and H. He, "Offline policy iteration based reinforcement learning controller for online robotic knee prosthesis parameter tuning," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2019.

[83] S. K. Au, J. Weber, and H. Herr, "Powered ankle-foot prosthesis improves walking metabolic economy," *IEEE Transactions on Robotics*, 2009.

[84] P. Malcolm, R. E. Quesada, J. M. Caputo, and S. H. Collins, "The influence of push-off timing in a robotic ankle-foot prosthesis on the energetics and mechanics of walking," *J. Neuroeng. Rehabil.*, vol. 12, Feb. 2015.

[85] S. Au, M. Berniker, and H. Herr, "Powered ankle-foot prosthesis to assist level-ground and stair-descent gaits," *Neural Netw.*, vol. 21, pp. 654–666, May 2008.

[86] A. H. Shultz and M. Goldfarb, "A unified controller for walking on even and uneven terrain with a powered ankle prosthesis," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2018.

[87] B. E. Lawson, H. A. Varol, and M. Goldfarb, "Standing stability enhancement with an intelligent powered transfemoral prosthesis," *IEEE Transactions on Biomedical Engineering*, 2011.

[88] D. Quintero, D. J. Villarreal, D. J. Lambert, S. Kapp, and R. D. Gregg, "Continuous-phase control of a powered knee-ankle prosthesis: Amputee experiments across speeds and inclines," *IEEE Transactions on Robotics*, 2018.

[89] R. D. Gregg, T. Lenzi, L. J. Hargrove, and J. W. Sensinger, "Virtual constraint control of a powered prosthetic leg: From simulation to experiments with transfemoral amputees," *IEEE Transactions on Robotics*, vol. 30, pp. 1455–1471, Dec. 2014.

[90] F. Sup, A. Bohara, and M. Goldfarb, "Design and control of a powered transfemoral prosthesis," *Int. J. Rob. Res.*, vol. 27, pp. 263–273, Feb. 2008.

[91] M. L. Handford, M. Srinivasan, M. Hulliger, and R. Zernicke, "Robotic lower limb prosthesis design through simultaneous computer optimizations of human and prosthesis costs," *Scientific Reports*, pp. 1–7, Feb. 2016.

[92] M. R. Tucker, C. Shirota, O. Lambercy, J. S. Sulzer, and R. Gassert, "Design and characterization of an exoskeleton for perturbing the knee during gait," *IEEE Transactions on Biomedical Engineering*, vol. 64, pp. 2331–2343, Oct. 2017.

[93] N. Thatte, H. Duan, and H. Geyer, "A sample-efficient black-box optimizer to train policies for human-in-the-loop systems with user preferences," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 993–1000, 2017.

[94] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[95] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," in *IEEE International Conference on Robotics and Automation*, pp. 156–163, 2015.

[96] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017.

[97] D. Vogt, S. Stepputtis, S. Grehl, B. Jung, and H. Ben Amor, "A system for learning continuous human-robot interactions from human-human demonstrations," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2017.

[98] X. Gao, J. Si, Y. Wen, M. Li, and H. He, "Reinforcement learning control for robotic knee with human-in-the-loop by flexible policy iteration," *Submitted to IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[99] M. P. Kadaba, H. K. Ramakrishnan, and M. E. Wootten, "Measurement of lower extremity kinematics during level walking," *Journal of Orthopaedic Research*, vol. 8, pp. 383–392, May 1990.

[100] C. Thurau, K. Kersting, M. Wahabzada, and C. Bauckhage, "Convex non-negative matrix factorization for massive datasets," *Knowl. Inf. Syst.*, vol. 29, no. 2, pp. 457–478, 2011.

[101] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[102] K. Shao, Y. Zhu, and D. Zhao, "Starcraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, pp. 73–84, Feb. 2019.

[103] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable deep reinforcement learning for robotic manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6244–6251, IEEE, May 2018.

[104] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, pp. 153–173, May 2017.

[105] S. J. Pan and Q. Yang, "A survey on transfer learning," 2010.

[106] K. Weiss, T. M. Khoshgoftaar, and D. D. Wang, "A survey of transfer learning," *J. Big Data*, 2016.

[107] S. J. Pan, V. W. Zheng, Q. Yang, and D. H. Hu, "Transfer learning for WiFi-based indoor localization," in *AAAI Work. - Tech. Rep.*, 2008.

[108] D. Abel, Y. Jinnai, G. Sophie Yue, G. Konidaris, and M. L. Littman, "Policy and Value Transfer in Lifelong Reinforcement Learning," in *Int. Conf. Mach. Learn.*, pp. 20–29, 2018.

[109] G. Boutsioukis, I. Partalas, and I. Vlahavas, "Transfer Learning in Multi-Agent Reinforcement Learning Domains," in *Eur. Work. Reinf. Learn.*, pp. 249–260, 2011.

[110] F. L. da Silva and A. H. R. Costa, "Transfer Learning for Multiagent Reinforcement Learning Systems," in *Proc. Twenty-Fifth Int. Jt. Conf. Artif. Intell.*, pp. 3982–3983, 2016.

[111] A. Lazaric, "Transfer in reinforcement learning: A framework and a survey," in *Reinforcement Learning - State of the art*, ch. 12, pp. 143–173, Springer, 2012.

[112] A. Lazaric, M. Restelli, and A. Bonarini, "Transfer of samples in batch reinforcement learning," in *Proc. 25th Int. Conf. Mach. Learn.*, pp. 544–551, 2008.

[113] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. Thomaz, "Policy shaping: Integrating human feedback with Reinforcement Learning," in *Adv. Neural Inf. Process. Syst.*, 2013.

[114] M. E. Taylor and P. Stone, "Behavior transfer for value-function-based reinforcement learning," in *Proc. Int. Conf. Auton. Agents*, 2005.

[115] F. Fernández and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in *Proc. Int. Conf. Auton. Agents*, 2006.

[116] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations : Theory and application to reward shaping," in *Int. Conf. Mach. Learn.*, pp. 278–287, 1999.

[117] G. Konidaris and A. Barto, "Autonomous shaping: knowledge transfer in reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning - ICML '06*, (New York, New York, USA), pp. 489–496, ACM Press, 2006.

[118] G. Konidaris and A. Barto, "Building portable options: Skill transfer in reinforcement learning," in *IJCAI International Joint Conference on Artificial Intelligence*, 2007.

[119] M. Grzes and D. Kudenko, "Plan-based reward shaping for reinforcement learning," in *Int. IEEE Conf. Intell. Syst.*, 2008.

[120] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof," *IEEE Tansactions Syst. Man, Cybern. Part B, Cybern.*, vol. 38, pp. 943–949, June 2008.

[121] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, "Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming," *Automatica*, vol. 48, no. 8, pp. 1825–1832, 2012.

[122] D. Liu and Q. Wei, "Finite-approximation-error-based optimal control approach for discrete-time nonlinear systems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 779–789, 2013.

[123] Q. Wei, F. Y. Wang, D. Liu, and X. Yang, "Finite-approximation-error-based discrete-time iterative adaptive dynamic programming," *IEEE Trans. Cybern.*, vol. 14, no. 12, pp. 2820–2833, 2014.

[124] Adil Ansari, "Windy-grid-world-q-learning," 2013.

[125] R. S. Sutton and A. G. Barto, *Reinforcement learning : an introduction*. Cambridge, MA: MIT Press, 2 ed., 2018.

[126] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, D. G. Thelen, and D. Delp, S.L., Anderson, F.C., Arnold, A.S., Loan, P., Habib, A., John, C.T., Guendelman, E., Thelan, "OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement," *IEEE Trans. Biomed. Eng.*, vol. 54, pp. 1940–1950, nov 2007.

[127] Z. Ni, N. Malla, and X. Zhong, "Prioritizing useful experience replay for heuristic dynamic programming-based learning systems," *IEEE Transactions on Cybernetics*, vol. 49, pp. 3911–3922, Nov. 2019.

[128] B. Luo, Y. Yang, and D. Liu, "Adaptive -learning for data-based optimal output regulation with experience replay," *IEEE Transactions on Cybernetics*, vol. 48, pp. 3337–3348, Dec. 2018.

[129] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2018.

[130] S. Choi and J. Kim, "Trajectory-based deep latent policy gradient for learning locomotion behaviors," in *IEEE International Conference on Robotics and Automation*, 2019.

[131] X. Gao, Y. Wen, M. Li, J. Si, and H. Huang, "Robotic knee parameter tuning using approximate policy iteration," in *Cognitive Systems and Signal Processing* (F. Sun, H. Liu, and D. Hu, eds.), (Singapore), pp. 554–563, Springer, Singapore, Nov. 2019.

[132] R. Bellman, *Dynamic programming*. Princeton University Press, 1957.