Reinforcement Learning Control of Robotic Knee with Human in the Loop by Flexible Policy Iteration

Xiang Gao, Jennie Si, Fellow, IEEE, Yue Wen, Minhan Li, and He (Helen) Huang, Senior Member, IEEE

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.

Index Terms—Reinforcement Learning (RL), flexible policy iteration (FPI), adaptive optimal control, data and time efficient learning, robotic knee, human-in-the-loop

I. 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 [1]. However, the validity of this approach is yet to be verified. Researchers also used response surface optimization [2] 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 [4], Atari games [5], the game of Go [6], [7] and energy efficient data center [8]. 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) [9], [10], 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 [11]-[13], and Apache helicopter control [14]-[16]. 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 [17]-[19], we demonstrated the feasibility of ADP, specifically direct heuristic dynamic programming (dHDP) [20], 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 [21]. It is also worth mentioning that, the generic dHDP without imposing further conditions [22] 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 [21], [23] and stable iterative control policies [24]-[28]. 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) [29] 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) [30], prioritized experience replay (PER) [31], [32] and hindsight experience replay (HER) [33] 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.Â

The ER idea has also been considered in ADP in different capacities [34]-[38]. It is shown in [34], [35] 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 [36]-[38] 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 [39], [40], 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 [41], [42] or an initial value function [43], [44]. However, they still required expert knowledge in the process, which is difficult to interpret and encode [44]. 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 [34]-[38], 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 [41]-[44], 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 I), 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.

II. 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 [45]-[47]; 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). 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 [48]. Fig. 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). 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.

A. 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

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

(1)

The prosthetic knee motor generates a knee joint torque \(T \in \mathbb{R}\) from the knee joint angle \(\theta\) and angular velocity \(\omega\) according to the following impedance control law

\[
T_k = K_k(\theta - (\theta_e)_k) + B_k\omega.
\]  

(2)
knee angle in each phase, and phase duration $D_k$ is the time
interval between two consecutive peaks (Fig. 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 [46], [49]. 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). 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
\begin{equation}
    x_k = [\Delta P_k, \Delta D_k]^T = [P_k - P_k', D_k - D_k']^T,
\end{equation}
and its control $u_k$ consists of increments to the IC parameters,
\begin{equation}
    u_k = [\Delta K_k, \Delta B_k, (\Delta \theta_k)_k]^T.
\end{equation}

### III. Flexible Policy Iteration

Consider the human-robot, i.e., the amputee-prosthesis sys-

tem as a discrete time nonlinear system with unknown dyna-

mics,
\begin{equation}
    x_{k+1} = F(x_k, u_k), k = 0, 1, 2, \ldots
\end{equation}
where action $u_k$ of the form described in (5) is determined
according to policy $h$ as
\begin{equation}
    u_k = h(x_k).
\end{equation}

In (6), the domain of $F(x_k, u_k)$ is denoted as $D \triangleq \{(x, u) | x \in \mathcal{X}, u \in \mathcal{U}\}$, 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 math-

etical model as (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 [19].
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 (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 prob-
lem due to our construction of the system states and RL control
(3) based on the biomechanics of human locomotion.

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

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,

\[
Q(x_k, u_k) = U(x_k, u_k) + Q(x_{k+1}, h(x_{k+1})). \tag{9}
\]

An optimal control is the one that stabilizes the system in (6) while minimizing the value function (8) according to Bellman optimality. The optimal value function is therefore of the form

\[
Q^*(x_k, u_k) = U(x_k, u_k) + \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1}) \tag{10}
\]

or

\[
h^*(x_k) = \arg\min_{u_k} Q^*(x_k, u_k), \tag{11}
\]

\[
Q^i(x_k, u_k) = U(x_k, u_k) + Q^i(x_{k+1}, h^*(i)(x_{k+1})), \tag{12}
\]

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 [24].

**Definition 1.** (Admissible Control) A control policy \( h(x) \) is admissible with respect to the value function \( Q(x, u) \) (8) if \( h(x) \) is continuous on \( \mathcal{X} \), \( h(0) = 0 \) and it stabilizes system (6), and the corresponding value function \( Q(x, u) \) 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 \( \tilde{Q}^{i}(x, u) \) and a control policy \( \tilde{h}^{i}(x) \), the policy iteration algorithm proceeds by iterating the following two steps:

**Policy Evaluation:**

\[
\tilde{Q}^{i}(x, u) = U(x, u) + \tilde{Q}^{i}(x_{k+1}, \tilde{h}^{i}(x_{k+1})). \tag{13}
\]

The above policy evaluation step (13) is based on the Bellman equation (9).

**Policy Improvement:**

\[
\tilde{h}^{i+1}(x) = \arg\min_{u_k} \tilde{Q}^{i}(x, u_k), i = 0, 1, 2, \ldots \tag{14}
\]

Motivated by the favorable properties of policy iteration in MDP problems, such as monotonically decreasing value, and demonstrated feasibility in solving realistic engineering problems [12], [13], we further develop the policy evaluation step to achieve data efficiency, easy implementation, and importantly, effectively solving realistic and complex problems.

**B. 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 \( V(x_k) \) 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 \( V \) 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 \( h^{i}(x_k) \),

\[
Q^{i}(x_k, u_k) = U(x_k, u_k) + \sum_{j=1}^{\infty} U(x_{k+j}, h^{i}(x_{k+j})) + \sum_{j=1}^{\infty} \alpha_i V(x_{k+j}). \tag{15}
\]

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 (9) becomes:

**Policy Evaluation with Augmented Information:**

\[
Q^{i}(x_k, u_k) = U(x_k, u_k) + Q^{i}(x_{k+1}, h^{i}(x_{k+1})) + \alpha_i V(x_{k+1}), i = 0, 1, 2, \ldots \tag{16}
\]

**Policy Improvement:**

\[
h^{i+1}(x_k) = \arg\min_{u_k} Q^{i}(x, u_k), i = 0, 1, 2, \ldots \tag{17}
\]

**Remark 1.** In (16), policy \( \tilde{h}^{i}(x) \) 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 \( V \) 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 (16) and (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 (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.

**C. Flexible Sampling with Experience Replay**

In policy evaluation (16), the value function of \( Q^{i}(x) \) 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 (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 [31], 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 \( \hat{\rho}_k^{(i)} \) as

\[
\hat{\rho}_k^{(i)} = \frac{1}{\zeta_k^{(i)}}, \quad \text{for } \forall k, \tag{18}
\]

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

\[
\rho_k^{(i)} = \frac{\hat{\rho}_k^{(i)}}{\sum \hat{\rho}_k^{(i)}}, \quad \text{for } \forall k, \tag{19}
\]

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

D. Approximate Policy Evaluation in FPI

To implement the policy evaluation step (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:

\[
\hat{Q}^{(i)}(x_k, u_k) = W^{(i)^T} \phi(x_k, u_k) = \sum_{k=1}^{L} w_k^{(i)} \varphi_k(x_k, u_k) \tag{20}
\]

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

The policy evaluation step (16) then becomes

\[
\hat{Q}^{(i)}(x_k, u_k) = U(x_k, u_k) + \hat{Q}^{(i)}(x_{k+1}, h^{(i)}(x_{k+1})) + \alpha_i \mathcal{V}(x_{k+1}). \tag{21}
\]

Substituting (20) into (21), we have

\[
[\phi(x_k, u_k) - \phi(x_{k+1}, h^{(i)}(x_{k+1}))] W^{(i)} = U(x_k, u_k) + \alpha_i \mathcal{V}(x_{k+1}). \tag{22}
\]

Equation (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 \( X^{(i)} \in \mathbb{R}^{Na\times L} \) 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, (22) can be rewritten as

\[
X^{(i)} W^{(i)} = Y^{(i)}. \tag{23}
\]

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

\[
\delta_k^{(i)} = U(x_k, u_k) + \hat{Q}^{(i-1)}(x_{k+1}, h^{(i)}(x_{k+1})) + \alpha_{i-1} \mathcal{V}(x_{k+1}) - \hat{Q}^{(i-1)}(x_k, u_k), \quad \text{for } i = 0, 1, 2, \ldots \tag{24}
\]

Then the weight \( \rho_k^{(i)} \) of the samples can be obtained from (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 (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 (23) as a weighted least squares solution using \( N \) weighted samples

\[
W^{(i)} = (X^{(i)^T} \Psi^{(i)} X^{(i)})^{-1} (X^{(i)^T} \Psi^{(i)} Y^{(i)})^T, \tag{25}
\]

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 (20).

**Algorithm 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 \( \{x_k, u_k, x_{k+1}\} \) from system (6) following policy \( h^{(i)} \) at gait cycle \( k \), \( N \leftarrow N_b \) (Setting 2(A) in Table I).

1b: (Incremental Data Collection) Collect a sample \( (x_k, u_k, x_{k+1}) \) from system (6) following policy \( h^{(i)} \), and add it to \( DS \), \( N \leftarrow N + 1 \) (Setting 2(B) in Table I).

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

3: (Set Other Parameters) Set \( \rho_k^{(i)} \) (Setting 3 in Table I) and \( \alpha_i \) (Setting 4 in Table I).

**Policy Evaluation/Update for Iteration \( i \)**

4: (Policy Evaluation) Evaluate policy \( \hat{h}^{(i)} \) by solving (21) for \( \hat{Q}^{(i)} \) using all samples in DS.

5: (Policy Update) Update policy \( \hat{h}^{(i+1)} \) by (27) and (28).

**TABLE I**

| Setting | Description |
|---------|-------------|
| 1       | (A) \( N_b \) is fixed  
(B) \( N_b \leftarrow N_b + 5 \) | Fixed  
Adaptive |
| 2       | (A) \( N \leftarrow N \)  
(B) \( N \leftarrow N + 1 \) | Batch mode  
Incremental mode |
| 3       | (A) \( \rho_k^{(i)} = 1 \)  
(B) \( \rho_k^{(i)} \) from (19) | No prioritization  
With prioritization |
| 4       | (A) \( \alpha_i = 0 \)  
(B) \( \alpha_i = 0.9^i \) | No prior knowledge  
With prior knowledge |

E. 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 (17) during policy improvement,

\[
h^{(i+1)}(x_k) = \arg \min_{u_k} \hat{Q}^{(i)}(x_k, u_k). \tag{26}
\]
We employ another linear-in-parameter function approximator $\hat{h}^{(i+1)}(x_k)$ for $h^{(i+1)}(x_k)$,

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

(27)

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

$$K_{j+1}^{(i+1)} = K_j^{(i+1)} - l \frac{\partial \hat{Q}^{(i)}(x_k, (K_j^{(i+1)})^T \sigma(x_k))}{\partial K_j^{(i+1)}}$$

(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.

F. Implementation of FPI

Algorithm 1 and Table I together describe our proposed FPI algorithm. The terminating condition in Algorithm 1 can be, for example, policy iteration index $i = i_{\text{max}}$ where $i_{\text{max}}$ is some positive number, or $|\hat{Q}^{(i)}(x_k, u_k) - \hat{Q}^{(i-1)}(x_k, u_k)| < \varepsilon$ where $\varepsilon$ is a small positive number. Note that there are four settings in Algorithm 1 (Table I). FPI can run in batch mode or incremental mode (Setting 1 in Table I), while such parameter is not required under incremental mode. In addition, Setting 3 describes how the priorities $p_k^{(i)}$ of the samples are assigned and Setting 4 describes how the prior knowledge is used at iteration $i$ through the parameter of $\alpha_i$.

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 (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. 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. 4 was generated under the setting of (A)/(A)/(A)/(A) in Table I and $N_b = 20$. Fig. 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. 3 coincides with the converging planes found by FPI in Fig. 4.

IV. 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 [24], [28], [50]. As mentioned before, we introduce a value function term $V(x_k)$ to capture prior knowledge. Specifically, we let $V(x_k) = \min Q^*(x_k, u_k)$, 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 $V$ 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 (16)-(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 $V(x_k)$ is also positive definite for $x_k$, according to (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 $Q^{(i)}(x_k, u_k)$, $i = 0, 1, ..., i$ 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 (16)-(17), where $h^{(0)}$ is an admissible control policy. Then, for $i = 0, 1, 2, ..., h^{(i)}$ stabilizes the system (6).

Proof: Consider the case when $x_k \neq 0$, we have $U(x_k, h^{(i)}(x_k)) > 0$ and $\alpha_i V(x_{k+1}) \geq 0$. From (16), and
Based on (17), we have
\[ Q^{(i)}(x_k, h^{(i+1)}) = \min_{u_k} Q^{(i)}(x_k, u_k) \leq Q^{(i)}(x_k, h^{(i)}). \] (31)

Based on (16) we have
\[ V^{(i)}(x_k) = Q^{(i)}(x_k, h^{(i)}) \geq U(x_k, h^{(i)}) + \alpha_i V(x_{k+1}) \]
\[ \geq U(x_k, h^{(i+1)}) + V^{(i)}(x_{k+1}) + \alpha_{i+1} V(x_{k+1}). \] (32)

Hence
\[ V^{(i)}(x_k) - V^{(i)}(x_{k+1}) \geq U(x_k, h^{(i+1)}) + \alpha_{i+1} V(x_{k+1}) \]
\[ V^{(i)}(x_{k+1}) - V^{(i)}(x_{k+2}) \geq U(x_{k+1}, h^{(i+1)}) + \alpha_{i+1} V(x_{k+2}) \]
\[ \vdots \]
\[ V^{(i)}(x_{k+N}) - V^{(i)}(x_{k+N+1}) \geq U(x_{k+N}, h^{(i+1)}) + \alpha_{i+1} V(x_{k+N+1}). \] (33)

Summing up the left and the right hand sides of (33) respectively,
\[ V^{(i)}(x_k) - V^{(i)}(x_{k+N+1}) \geq \sum_{j=k}^{k+N} U(x_j, h^{(i+1)}) + \alpha_{i+1} \sum_{j=k}^{k+N} V(x_{j+1}), \] (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 stabilizing control policy as proved in Theorem 1, and \( \lim_{N \to \infty} (\sum_{j=k}^{k+N} U(x_j, h^{(i+1)}) + \alpha_{i+1} \sum_{j=k}^{k+N} V(x_{j+1})) = V^{(i+1)}(x_k). \) Hence, (34) yields
\[ V^{(i)}(x_k) \geq V^{(i+1)}(x_k). \] (35)

According to (16) and (35), we can obtain
\[ Q^{(i+1)}(x_k, u_k) = U(x_k, u_k) + V^{(i+1)}(x_{k+1}) \]
\[ \leq U(x_k, u_k) + V^{(i)}(x_{k+1}) \]
\[ = Q^{(i)}(x_k, u_k). \] (36)

**Theorem 3.** Let Assumption 1 hold. Let \( Q^{(i)}(x_k, u_k) \) and \( h^{(i)} \) be updated by (16)-(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 (15) and Theorem 2 we have
\[ Q^{(i)}(x_k, u_k) \geq Q^{(i)}(x_k, u_k) \]
\[ = U(x_k, u_k) + \sum_{j=1}^{\infty} U(x_{k+j}, h^{(i)}(x_{k+j})) \]
\[ + \sum_{j=1}^{\infty} \alpha_1 V(x_{k+j}). \] (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 (16) and (17), respectively, and the optimal value function \( Q^*(x_k, u_k) \) and the optimal policy be defined in (10) and (11), respectively. Then \( Q^{(i)}(x_k, u_k) \rightarrow Q^*(x_k, u_k) \) and \( h^{(i)}(x_k) \rightarrow h^*(x_k) \) as \( i \rightarrow \infty \), \( \forall (x_k, u_k) \in D \).

Proof: By definition, \( Q^*(x_k, u_k) \leq Q^{(i)}(x_k, u_k) \) holds for any \( i \), and from Theorem 2 \( \{Q^{(i)}(x_k, u_k)\} \) is a non-increasing sequence that is bounded by \( Q^*(x_k, u_k) \). Hence \( \{Q^{(i)}(x_k, u_k)\} \) must have a limit as \( i \rightarrow \infty \). Denote this limit as \( Q^\infty(x_k, u_k) \equiv \lim_{i \to \infty} Q^{(i)}(x_k, u_k) \) and \( h^\infty(x_k) \equiv \lim_{i \to \infty} h^{(i)}(x_k) \). Note that \( \lim_{i \to \infty} \alpha_i \mathcal{V}(x_{k+1}) = 0 \), take the limits in (16) and (17) as \( i \to \infty \),

\[
Q^\infty(x_k, u_k) = U(x_k, u_k) + Q^\infty(x_{k+1}, h^\infty(x_k)), \quad (38)
\]

\[
h^\infty(x_k) = \arg \min_{u_k} Q^\infty(x_k, u_k). \quad (39)
\]

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

\[
V^*(x_k) = \min_{h_k} \left[ U(x_k, h_k(x_k)) + V^*(x_{k+1}) \right]. \quad (40)
\]

When \( i \to \infty \), \( u_k = h^\infty(x_k) \), so from (38) and (39) we can get

\[
V^\infty(x_k) = Q^\infty(x_k, h^\infty(x_k)) = \min_{u_k} \left[ U(x_k, u_k) + Q^\infty(x_{k+1}, h^\infty(x_k)) \right] = \min_{u_k} \left[ U(x_k, u_k) + V^\infty(x_{k+1}) \right]. \quad (41)
\]

Equation (41) satisfies the Bellman optimality equation (40), thus \( V^\infty(x_k) = V^*(x_k) \). From (38) we can obtain

\[
Q^\infty(x_k, u_k) = U(x_k, u_k) + V^\infty(x_{k+1}) = U(x_k, u_k) + V^*(x_{k+1}) = Q^*(x_k, u_k). \quad (42)
\]

Therefore \( h^\infty(x_k) = h^*(x_k) \) can be obtained from (39).■

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 \( i \)th 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 \( \gamma \) that makes the condition \( \min_{u_{k+1}} Q^\infty(x_{k+1}, u_{k+1}) \leq \gamma U(x_k, u_k) \) hold uniformly on \( \mathcal{X} \).

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

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

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

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

\[\xi Q^{(i)} \leq \bar{Q}^{(i)} \leq \eta Q^{(i)} \quad (43)\]

holds uniformly for \( i \) as well as \( x_k \) and \( u_k \), where \( 0 < \xi \leq 1 \) and \( \eta \geq 1 \) are constants, \( \bar{Q}^{(i)}(x_k, u_k) \) is defined by (21) and \( Q^{(i)} \) is defined by (15).

Theorem 5. Let Assumptions 1 and 2 hold. Let \( \hat{Q}^{(i)}(x_k, u_k) \) be defined by (21) and \( Q^{(i)} \) be defined by (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 (43). If the following condition is satisfied

\[\eta < \frac{\gamma + 1}{\gamma}, \quad (44)\]

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

\[\xi Q^{(i)} \leq \hat{Q}^{(i)} \leq \frac{\eta}{\gamma + 1 - \gamma} Q^* \quad \xi (45)\]

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

\[\xi Q^* \leq \hat{Q}^{(i)} \leq \frac{\eta}{\gamma + 1 - \gamma} Q^*. \quad (46)\]

Proof: The left-hand side of (45) can be easily obtained according to (43) and Theorem 3.

The right-hand side of (45) is proven by mathematical induction as follows.

First, for \( i = 0 \), \( \bar{Q}(0) \leq \eta \bar{Q}(0) = \eta Q(0) \leq \beta Q^* \) holds according to (43) and the conditions in Theorem 5. Thus (45) holds for \( i = 0 \).

Assuming that (45) holds for \( i \geq 0 \), then for \( i + 1 \) we have

\[
\hat{Q}^{(i+1)}(x_k, u_k) = U(x_k, u_k) + \hat{Q}^{(i)}(x_{k+1}, h^{(i+1)}(x_{k+1}))
\]

\[= U(x_k, u_k) + \min_{u_{k+1}} \bar{Q}^{(i)}(x_{k+1}, u_{k+1}) \quad (47)
\]

\[\leq U(x_k, u_k) + \min_{u_{k+1}} P_i Q^*(x_{k+1}, u_{k+1})
\]

where

\[P_i = \eta \beta \left( \frac{\eta \gamma}{\gamma + 1} \right)^i + \frac{\eta}{\gamma + 1 - \gamma} \eta . \quad (48)\]

According to Assumption 2, (47) yields

\[
\hat{Q}^{(i+1)}(x_k, u_k) \leq (1 + \gamma \frac{P_i - 1}{\gamma + 1}) U(x_k, u_k)
\]

\[+ (P_i - \frac{P_i - 1}{\gamma + 1}) \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1})
\]

\[= \frac{1}{\eta} \left[ \eta \beta \left( \frac{\eta \gamma}{\gamma + 1} \right)^{i+1} + (1 - \frac{\eta \gamma}{\gamma + 1}) \right] \frac{\eta}{1 + \gamma - \gamma \eta} \]

[\hat{U}(x_k, u_k) + \min_{u_{k+1}} Q^*(x_{k+1}, u_{k+1})] \]
where

\[
\eta = \frac{1}{\gamma} \left[ \eta \beta \frac{\eta^\gamma}{1 + \gamma} T^{i+1} + \left( 1 - \frac{\eta^\gamma}{1 + \gamma} T^{i+1} \right) \frac{\eta}{1 + \gamma - \eta^\gamma} \right] \times Q^*(x_k, u_k).
\]

(49)

On the other hand, according to (43), there is \( \hat{Q}^{(i+1)} \leq \eta \hat{Q}^{(i+1)} \). Thus (45) holds for \( i+1 \). By mathematical induction, the proof for (45) is completed.

Considering (43) and (45), we can easily obtain

\[
\hat{Q}^{(\infty)} \leq \frac{\eta}{1 + \gamma - \eta^\gamma} Q^*,
\]

(50)
as \( i \to \infty \). Thus (46) holds.

Remark 3. Condition (44) ensures that the upper bound in (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 \( i \to \infty \).

V. 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 (NCSRRI) 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 V-C, noise was either generated by a random number generator (the sensor noise and actuator noise cases in Table III), or by gait-to-gait variances captured from two amputee subjects walking with prosthesis (case TF1 and TF2 in Table III). For the latter case, data were collected from another study [51] 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 apply 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.

A. Algorithm and Experiment Settings

We summarize the parameters of the FPI in OpenSim simulations as follows. Algorithm 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 \( u_k \):

\[
U(x_k, u_k) = x_k^T R x_k + u_k^T R u_k,
\]

(51)

where \( R_x \in \mathbb{R}^2 \) and \( R_u \in \mathbb{R}^3 \) were positive definite matrices. Specifically, \( R_x = \text{diag}(1, 1) \) and \( R_u = \text{diag}(0.1, 0.2, 0.1) \) 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 (16). The basis functions are \( \phi(x_k, u_k) = [x(1)^2, x(1) k x(2) k, x(1) u(1) k, x(1) k u(2) k, x(1) k u(3) k, x(2)^2, x(2) k u(1) k, x(2) k u(2) k, x(2) k u(3) k, u(1)^2, u(2)^2, u(3)^2, x(1) k x(2) k, x(1) k u(1) k, x(1) k u(2) k]^T \), 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 \( l_0 \) as in (1). The adaptive optimal control objective for FPI is to make state \( x_k \) approach zero, i.e., the peak error \( \Delta P_k \) and duration error \( \Delta D_k \) 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 [18, 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 1 was 100. The results in Subsections V-B and V-C 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 V-A, 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.

B. 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 I), and neither PER nor prior knowledge was considered. Table II summarizes the performance of FPI in batch mode.
Phase 1

Phase 2

Phase 3

Phase 4

Fig. 5. Illustration of the converging process of the policy vector $[\Delta K, \Delta B, \Delta \theta_e]^T$.

Fig. 6. Comparison of the RMSEs between controlled knee profiles and target profiles using FPI, GPI and NFQCA under the same stage cost (51).

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 II also shows that, under Setting 2(A), adaptive batch mode improves both success rates and tuning time over fixed batch mode.

Fig. 5 was generated under the setting of (A)(A)(A)(A) as in Table I and $N_b = 20$. Fig. 5 illustrates converging policies computed according to (26).

C. 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) [52], neural fitted Q with continuous action (NFQCA) [53] and our previous direct heuristic dynamic programming (dHDP) implementation [18].

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 [53], 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 III were based on an adaptive batch size $N_b$ between 20 and 40 (i.e., Settings (B)(A)(A)(A) in Table I), and results in Fig. 6 used a fixed $N_b$ of either 20 or 40 (i.e., Settings (A)(A)(A)(A) in Table I).

Before the comparison study, we first validated our implementations of GPI, NFQCA and dHDP using examples from [18], [53], [52], 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 [52], 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 [53]. 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 $N$, while the maximum $N_b$ (FPI), $N$ (GPI) and $N$ (NFQCA) were all set to 40 gait cycles in Table III.

Table III 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. 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 $(N = 40, N_l = 5)$ in GPI [52] which is in the typical range that has been tested, the RMSE increased after a few iterations. Also note from Fig. 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.

D. 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 (15) for the last row result in Table IV, we trained an FPI agent for just one trial in OpenSim under the same settings as those in the first row of Table II (Settings (A)(A)(A)(A) in Table I and $N_l = 20$). Then prior knowledge $\mathcal{V}$ is obtained from $\mathcal{V}(x_k) = \min_{u_k} Q^*(x_k, u_k)$ where $Q^*(x_k, u_k)$ the final approximate value function after Algorithm 1 is terminated.

### Table IV

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

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

Table IV 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 I). The first configuration is the ER case without sample prioritization, i.e., $\rho_{i,k}^{(i)} = 1$ 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 IV), no prior knowledge was used, i.e., $\mathcal{V}(x_k) = 0$ for all $x_k$. The third configuration (the third row in Table IV) 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 IV, 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 IV will be 100%.

A statistical summary of a 30 randomly initialized trials based on the condition in row 1 of Table IV is provided in Fig. 2 (bottom half panel). As shown, after tuning, the proposed FPI algorithm successfully reduced gait peak and duration errors.

VI. 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.

### References

[1] 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, no. 2, pp. 164–173, Apr. 2010.

[2] 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, no. 8, pp. 1–21, Aug. 2015.

[3] H. Huang, D. L. Crouch, M. Liu, G. S. Sawicki, and D. Wang, “A cyber expert system for auto-tuning powered prosthetic impedance control parameters,” Ann. Biomed. Eng., vol. 44, no. 5, pp. 1613–1624, May 2016.

[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” J. Mach. Learn. Res., vol. 17, no. 1, pp. 1334–1373, Jan. 2016.
