Iranian Journal of Fuzzy Systems

Volume 22, Number 2, (2025), pp. 59-80

Original paper

University of Sistan and Baluchestan

Fuzzy radial basis function least square policy iteration: A novel critic-only reinforcement learning framework

O. Mehrabi [1], A. Fakharian [2], M. Siahi [3] and A. Ramezani [4]

[1,3] *Department of Mechanical, Electrical and Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran*

[2] *Department of Electrical Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

[4] *Department of Electrical Engineering, Tarbiat Modares University, Tehran, Iran*

o.mehrabi62@gmail.com, ahmad.fakharian@iau.ac.ir, mehdi.siahi@srbiau.ac.ir, ramezani@modares.ac.ir

Abstract

In this paper, a new form of critic-only Reinforcement Learning (RL) algorithm for continuous state space control problems is proposed. Our approach, called Fuzzy-RBF Least Square Policy Iteration (FRLSPI), dynamically tunes the weight parameters of a Takagi–Sugeno type neuro-fuzzy network online. FRLSPI is developed by integrating Least Squares Policy Iteration (LSPI) with the fuzzy-RBF network as a function approximator. Using this approach, we overcome two difficulties: determining state-action basis functions in LSPI algorithms and learning rate dependency in most RL algorithms. Additionally, we present positive theoretical results concerning an error bond between the optimal and the approximated Action Value Function (AVF) for FRLSPI. Our proposed method has suitable features such as positive mathematical analysis, learning rate independency and, comparatively good convergence properties. Simulation studies validate the effectiveness and performance of our learning framework, suggesting that our approach surpasses existing reinforcement learning algorithms.

Keywords: Neuro-fuzzy reinforcement learning, fuzzy-RBF network, generalization, least square policy iteration.

1 Introduction

A self-learning, model-free controller eliminates the need for complex model development and tuning, making it more efficient and adaptable to changing conditions in industrial processes. By utilizing machine learning algorithms, this type of controller can continuously learn from data and improve its performance over time without the need for manual adjustments [26]. This approach would be particularly beneficial for complex nonlinear systems where traditional controllers struggle to maintain optimal performance. The ability to scale well with multiple inputs and outputs would make this controller versatile and applicable to a wide range of industrial processes. Overall, a self-learning model-free controller has the potential to revolutionize control systems in the industry by offering a more flexible, efficient, and adaptive solution [1, 26]. Reinforcement learning (RL) has emerged as a powerful tool in various fields due to its ability to learn control strategies directly from data. Model-free RL is particularly popular for adapting to nonlinear processes without requiring an explicit process model. This is especially advantageous when dealing with complex systems where accurate models may be difficult to obtain. By eliminating the need for model identification, RL continuously enhances performance through re-training in response to process shifts or noise. In contrast, traditional model-based control methods rely on parametric models that may degrade over time when faced with unexpected disturbances or changes in the process dynamics [22]. Applying RL to control problems with large state and action spaces presents two major challenges. One of them is well known as "curse of dimensionality": Since even comparatively simple control problems can have huge state spaces that have to be explored, the learning process results in time-consuming and low efficient

Corresponding Author: A. Fakharian

Received: March 2024; Revised: January 2025; Accepted: April 2025.

https://doi.org/10.22111/ijfs.2025.48345.8503

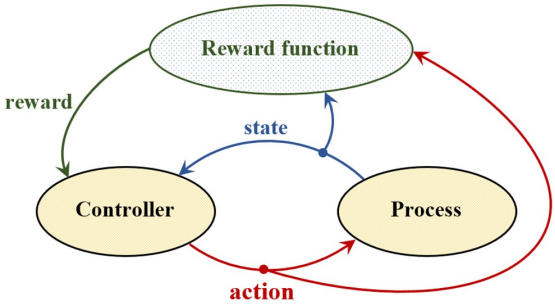


Figure 1: The flow of interaction in RL

learning. The other is the "generalization problem": assuming that limited experience does not provide sufficient data for each and every state. Hence, it's crucial to generalize the observed data to unseen states. Both these issues are addressed by the function approximation approach [3, 8, 11, 17].

Fuzzy systems and neural networks have attracted extensive research interests in the area of function approximation due to their ability to approximate nonlinear functions. As such, some authors [3, 8, 11, 15, 18, 20, 32] have offered continues fuzzy/Neural Reinforcement Learning (FRL/NRL), and employed FRL/NRL to tune a fuzzy/neural controller.

1.1 Related work

Continues algorithms that uses fuzzy systems or neural network as a function approximator can be classified into three well-known architectures in general, which are actor-only, critic-only, and actor-critic methods [11]. Most of the research done in this field has an actor-critic architecture, in which a policy function and the expected value are learned simultaneously [15, 16, 18, 20, 32]. However, even with these significant contributions, the main limitation of the aforementioned algorithms is the lack of proper exploration [8, 13]. Contrary to the mentioned architecture, critic-only methods are formed on the idea of first finding the optimal value function and then extracting an optimal policy from this value function [8, 11]. In a critic-only based FRL/NRL algorithms, the Action Value Function (AVF) is approximated by a fuzzy system or neural network as a critic part, and the action is produced with probability proportional to this function.

Such an action selection strategy improves balance between exploration and exploitation [8]. However, despite learning schemes like Fuzzy Sarsa Learning (FSL), Deep Q-learning [28], Deep deterministic policy gradient algorithm (DDPG) [25] and others. continues methods usually suffer from two major challenges: 1- In contrast with standard RL algorithms, the experimental results of combining successful RL algorithms with fuzzy/neural Function Approximators (FAs) generally lack reliable theoretical analysis. 2- Dependence on learning rate: Tuning the learning rate parameter for each problem separately in continuous reinforcement learning algorithms is challenging. Considering that most of the presented methods are an extension of the discrete algorithms dependent on the learning rate, therefore the root of the problem can be regarded as a way in the basic algorithm used.

Least Square Policy Iteration (LSPI) based approaches are an important family of value-based algorithms that approximate AVF instead of calculating its exact value [16]. These algorithms work by iteratively evaluating and improving the control policies, are sample-based, and have suitable features such as learning rate independency and comparatively good learning speed. The original form of LSPI is a model-free, off-policy algorithm that generates a sequence of AVF using the least square temporal difference (LSTD) algorithm at the policy evaluation phase. Extensions to approximation-based LSPI are studied in detail by Bušoniu et al. [6], and its online version (OLSPI) is derived in Bušoniu et al. that performs policy improvements once every few transitions have been completed, and updates its policy based on an incomplete evaluation of its current policy [5]. Bayesian least-squares policy iteration (BLSPI) is another off-policy, model-free, policy iteration algorithm that uses the Bayesian least-squares temporal-difference (BLSTD) learning algorithm instead of LSTD to evaluate policies [27]. An online variant of BLSPI, called randomized BLSPI (RBLSPI), that improves its policy based on an incomplete policy evaluation step is also presented in [27]. Although the presented methods, offering a RL method with better properties in convergence, stability, and sample complexity than previous RL algorithms, the approximation structure in these methods may lead to degraded performance when improperly selecting features. In [29], a kernel-based LSPI (KLSPI) algorithm was presented for MDPs with large or continuous state spaces and in [30] the hierarchical version of KLSPI (HKLSPI) is presented but the final policies

Table 1: Comparison of FRLSPI with other related works

Approach		On-Line	Continuous		Admiss. System dimensionality	Kernel trick	Mathematical Analysis	Initial Policy required
Ref.	Algorithm		State	Action				
Lagoudakis and Parr (2003)	LSPI	×	✓	×	Low	×	×	×
Xu et al. (2007)	KLSPI	×	✓	×	Mid	✓	×	✓
Thiery and Scherrer (2010)	LSλPI	×	✓	×	Low	×	✓	×
Buşoniu et al. (2010)	OLSPI	✓	✓	×	Low	×	×	×
Buşoniu et al. (2010)	OLSPI with cont. action	✓	✓	✓	Low	×	×	×
Jakab and Csato (2015)	KRLSTD	×	✓	×	Low	✓	×	✓
Cui et al. (2017)	KDPP	×	✓	×	High	✓	×	×
Ghorbani, et al. (2017)	FLSPI	✓	✓	✓	Mid	✓	✓	✓
Tziortziotis, et al. (2018)	BLSPI	×	✓	×	Mid	✓	×	×
Friedrich, et al. (2019)	OKLSPI	✓	✓	×	Mid	✓	×	×
Our Approach	FRLSPI	✓	✓	×	Mid	✓	✓	✓

of these algorithms are greatly influenced by the approximation precision of value functions and the choice of basis functions. An overview on different LSPI algorithms is presented in Table 1 [7, 11, 14, 29, 30, 31].

To address these challenges, we employ a hybrid approach that combines LSPI with a fuzzy RBF network. We introduce Fuzzy-Radial Basis Function Least Squares Policy Iteration (FRLSPI), an on-policy, model-free algorithm with critic-only architecture for control problems with large or continuous state spaces. We use fuzzy-RBF network as our general approximator, to approximate AVF in LSPI or from another perspective, LSPI is used to tune the weights of a fuzzy-RBF network. The main innovation of this work is theoretical and focused on the problem of establishing a performance bond for FRLSPI over all iterations.

The reminder of this paper is structured as follows: First, we briefly review some basic concepts and terms in RL that will be used in the rest of the article. FRLSPI algorithm is presented in Section 3, and then we try to provide some positive analytical results in Section 4. Section 5 demonstrates the performance of our algorithm on two standard RL testing domains. Finally, conclusions are drawn in Section 6.

2 Background

In this section, some background material, which will be referenced throughout the remainder of the paper, is reviewed. Table 2 lists the key symbols used in this article.

2.1 Preliminaries and problem statement

Notation Environments are formalized as Markov Decision Processes (MDP) and can be defined as a 5-tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, r, \gamma)$. The model evolves in discrete time and is specified by state space $\mathcal{S} = \left\{s_1, s_2, \ldots, s_{|s|}\right\}$ and action space $\mathcal{A} = \left\{a_1, a_2, \ldots, a_{|\mathcal{A}|}\right\}$, the transition probabilities $P_{s,a}(s') = (P(s'|s,a))_{s,s' \in \mathcal{S},\, a \in \mathcal{A}}$ describing the (stochastic) system behavior ($P_{s,a}(s') \geq 0, \sum_{s' \in \mathcal{S}} P_{s,a}(s') = 1 \forall s \in \mathcal{S}, a \in \mathcal{A}$) and an immediate reward function $r(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. In RL, at each time step, $t = 0, 1, 2, \ldots$ the agent (controller) observes the state $s_t \in \mathcal{S}$, containing information about its current state and generates an action $a_t \in \mathcal{A}(s_t)$. This action causes the environment to move into a new state $s_{t+1} = s'$, with probability $P_{s,a}(s')$ the agent then receives a scalar reward signal r_{t+1} that evaluates the quality of this transition, and the whole cycle repeats. This flow of interaction is represented in Figure 1. The scalar discount factor $\gamma \in [0, 1)$ determines the importance of immediate rewards compared with future rewards. The agent's objective is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$ mapping states to actions that maximize the return $G_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+k+1}$ [8, 26]. The central idea of value-based methods is to maximize the return for every possible state by taking actions in each step t that are expected to lead to higher values. The value of state s, $V^{\pi}(s) : \mathcal{S} \to \mathbb{R}$ is determined as follows [26]:

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|s_t = s\right\}, \tag{1}$$

where $E_{\pi}\{\}$ indicates the expectation over trajectories assuming that the agent follows policy π. We call the function V^{π} the state-value function for policy π. It can be easily shown that the following consistency condition holds between the value of state s and the value of its possible successor states [26]:

$$V^{\pi}(s) = \sum_{a \in A} \pi(s,a) \sum_{s' \in S} p^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')\right). \tag{2}$$

Table 2: List of main symbols and notation.

Symbol	***Description***
t, r	*Discrete time step, reward*
γ, V, Q	*Discount factor, value function, Action value function*
e, λ	*Eligibility trace and decay parameter*
$\pi, \mathcal{T}_\pi$	*Policy and bellman operator under policy* π
$\hat{Q}$	*Approximate action value function*
a	*Action,* $a \in [a_1, a_2, \ldots, a_m]$
s_i, s	i^{th} *element of state vector,* $i = 1, 2, \ldots, n$ *and state vector* $s = [s_1, s_2, \ldots, s_n]$
x_i, x	i^{th} *dimension input* $i = 1, 2, \ldots, n+1$ *and input vector* $x = [s, a]^T$
ϕ, Φ	*Basis functions and state-action space matrix*
n_i	*Number of gaussian fuzzy sets composes the partition of* i^{th} *input*
h, w	*Number of fuzzy rules and vector of tunable weights*
φ_{ij}	*The* i^{th} *membership function in the* j^{th} *rule neuron*
c_{ij}, σ_{ij}	*The center and width of the membership function* φ_{ij}
$\|.\|$, $\|.\|_\infty$	*Generic norm, infinity norm*

By defining:

$$p_{ss'} = \sum_{a \in A} \pi(s,a)\, p^a_{ss'}, \tag{3}$$

$$\mathcal{R}_s = \sum_{a \in A} \sum_{s' \in s} \pi(s,a)\, p^a_{ss'} \mathcal{R}^a_{ss'} = \sum_{a \in A} \pi(s,a)\, \mathcal{R}^a_s. \tag{4}$$

The relation (2) can be represented in the matrix form as:

$$\begin{bmatrix} V^\pi(s_1) \\ V^\pi(s_2) \\ \vdots \\ V^\pi(s_n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}^\pi(s_1) \\ \mathcal{R}^\pi(s_2) \\ \vdots \\ \mathcal{R}^\pi(s_n) \end{bmatrix} + \gamma \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix} \begin{bmatrix} V^\pi(s_1) \\ V^\pi(s_2) \\ \vdots \\ V^\pi(s_n) \end{bmatrix}. \tag{5}$$

More compactly:

$$V^\pi(s) = \mathcal{R}^\pi + \gamma P_\pi V^\pi. \tag{6}$$

An expected Bellman operator $\mathcal{T} : \mathbb{R}^{|\mathcal{S}|} \longrightarrow \mathbb{R}^{|\mathcal{S}|}$ can be defined as $\mathcal{T}^\pi(V) = \mathcal{R}^\pi + \gamma P_\pi V^\pi$ for any $V \in \mathbb{R}^{|\mathcal{S}|}$. Similarly, for the AVF under policy π, the Bellman equation can be defined as [8, 26]:

$$\begin{aligned} Q^\pi(s,a) &= E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|s_t = s\ ,\ a_t = a \right\} \\ &= \sum_{s' \in S} p^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma V^\pi(s') \right) \\ &= \sum_{s' \in S} p^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma Q^\pi(s', \pi(s')) \right). \end{aligned} \tag{7}$$

The AVF for policy π can be characterized as a unique fixed point of operator $\mathcal{T}$ and can be written in the following matrix notation:

$$Q^\pi = \mathcal{R} + \gamma P \Pi_\pi Q^\pi = \mathcal{T}_\pi Q, \tag{8}$$

where Q^π and $\mathcal{R}$ are $|\mathcal{S}|\,|\mathcal{A}|$ dimensional vectors, $P \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|}$ is the transition matrix induced by the selection of an action and policy π right after, and $\Pi_\pi \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}||\mathcal{A}|}$ is a stochastic matrix that describes policy π [16]. The goal of RL is to find the optimal policy π^* that satisfies $Q^*(s,a) = \sup_\pi Q^\pi(s,a)$ for each $(s,a) \in \mathcal{S} \times \mathcal{A}$.

Now, here we state some important properties of the operators $\mathcal{T}$ and $\mathcal{T}_\pi$ that will be of use in the rest of the article. $\mathcal{T}_\pi$ is a contraction mapping under supremum norm if for all $Q_1, Q_2 \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and every policy π there exists a $\gamma \in [0,1)$ such that [4]:

$$\|\mathcal{T}Q_1 - \mathcal{T}Q_2\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty, \tag{9}$$

$$\|\mathcal{T}_\pi Q_1 - \mathcal{T}_\pi V Q_2\|_\infty \leq \gamma \|Q_1^\pi - Q_2^\pi\|_\infty. \tag{10}$$

Consider the identity matrix $e \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, the following relations hold [4]

$$\mathcal{T}(Q + ce)(s,a) = \mathcal{T}Q(s,a) + \gamma ce, \tag{11}$$

$$\mathcal{T}_\pi(Q + ce)(s,a) = \mathcal{T}_\pi Q(s,a) + \gamma ce. \tag{12}$$

Operators $\mathcal{T}$ and $\mathcal{T}_\pi$ satisfy the monotonicity property, i.e., for every Q_1 , $Q_2 \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$, we have [4]:

$$Q_1 \leq Q_2 \Longrightarrow \mathcal{T}Q_1 \leq \mathcal{T}Q_2, \quad \mathcal{T}_\pi Q_1 \leq \mathcal{T}_\pi Q_2. \tag{13}$$

2.2 Least-squares policy iteration (LSPI)

Policy iteration (PI) is an iterative algorithm used to learn the optimal policy for a given MDP [4]. It starts with an arbitrary policy π_0, typically randomly chosen, and generates a sequence of improving policies along with their corresponding AVFs. This sequence can be represented as follows:

$$\pi_0 \xrightarrow{eval.} Q^{\pi_0} \xrightarrow{impr.} \pi_1 \xrightarrow{eval.} Q^{\pi_1} \xrightarrow{impr.} \pi_2 \xrightarrow{eval.} \ldots \xrightarrow{impr.} \pi^* \xrightarrow{eval.} Q^*$$

Each iteration m involves two successive phases: policy evaluation and policy improvement. In policy evaluation, the AVF of given policy Q^{π_m} is computed for the current policy π_m:

$$Q^{\pi_m}(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in S} P(s,a,s') \sum_{a' \in A} \pi(s',a') Q^{\pi_m}(s',a'). \tag{14}$$

Afterward, in policy improvement, the new policy is made greedy with respect to the new AVF Q^{π_m}:

$$\pi_{m+1}(s) = arg\max_{a \in A} Q^{\pi_m}(s,a). \tag{15}$$

This iterative procedure continues until there is no further improvement (i.e., $\pi_m = \pi_{m-1}$). After the convergence of policy iteration, the optimal policy may be obtained $\pi^* = \pi_k$, commonly in a finite number of iterations.

Despite its advantages, in classic policy iteration, Value functions and policies are tabular and accurate. But many real-world control problems have large or continuous state-action spaces, making the explicit representation of AVF impossible in general. To deal with the trouble, function approximation techniques are commonly used in the policy evaluation phase to estimate AVF instead of explicitly computing it. This kind of policy iteration is widely known as approximate policy iteration. It is common to approximate the AVF $Q(s,a)$ by the combination of a set of features or basis functions $\Gamma = \{\phi_i(s,a) : S \times A \to \mathbb{R}, i = 1, \ldots, L\}$ as follows:

$$\begin{aligned} Q^\pi(s,a) \approx \widehat{Q}^\pi(s,a;\theta) &= \sum_{i=1}^{L} \phi_i(s,a)\theta_i = \phi^T(s,a)\theta, \\ \phi(s,a) &= [\phi_1(s,a), \ldots, \phi_L(s,a)]^T, \end{aligned} \tag{16}$$

where $\theta \in \mathbb{R}^L$ represents a parameter vector and $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^L$ is a function that maps state-action pairs to a feature vector of L components. In this manner, the number of parameters ($L \ll |\mathcal{S}|\,|\mathcal{A}|$) that need to be estimated is significantly reduced compared to what is required in the case of an exact representation, and therefore this allows for faster convergence and scalability to larger spaces. In matrix form, the above formula can be written as $\widehat{Q}^\pi = \Phi\theta$ where Φ is an $|\mathcal{S}|\,|\mathcal{A}| \times L$ dimensional feature matrix:

$$\Phi = \begin{bmatrix} \phi^T(s, a_1) \\ \cdots \\ \phi^T(s_i, a_i) \\ \cdots \\ \phi^T\left(s_{|s|}, a_{|A|}\right) \end{bmatrix}. \tag{17}$$

The standard approach to find an approximate AVF is to solve $\widehat{Q}^\pi = \Phi\,\theta$ such that, $\Phi\,\theta_\pi = \Pi\,\mathcal{T}^\pi\,\Phi\,\theta_\pi$ where Π is the orthogonal projection onto the subspace spanned by the columns of Φ.

As function approximation used in (16), $\widehat{Q}^\pi$ lies in the space spanned by the column space of Φ; it is possible that $\mathcal{T}^\pi\widehat{Q}^\pi$ does not lie in the feature space. Let $\left(\Phi\big(\Phi^T\Phi\big)^{-1}\Phi^T\right)$ be the orthogonal projection to the space which minimizes the L_2 norm; the following fix point equation is solved:

$$\widehat{Q}^\pi = \Phi\big(\Phi^T\Phi\big)^{-1}\Phi^T\left(\mathcal{T}^\pi\widehat{Q}^\pi\right) = \Phi\big(\Phi^T\Phi\big)^{-1}\Phi^T\left(\mathcal{R} + P\Pi_\pi\widehat{Q}^\pi\right). \tag{18}$$

Where $\widehat{Q}^\pi$ is parameterized by θ. By manipulation of above formula, we get:

$$\Phi\big(\Phi^T\Phi\big)^{-1}\Phi^T\left(\mathcal{R} + P\Pi_\pi\widehat{\mathcal{Q}}^\pi\right) = \widehat{\mathcal{Q}}^\pi \xrightarrow{\widehat{\mathcal{Q}}^\pi = \Phi\theta^\pi} \underbrace{\Phi^T\left(\mathcal{R} + P\Pi_\pi\Phi\right)}_{k\times k}\underbrace{\theta^\pi}_{k\times 1} = \underbrace{\Phi^T\mathcal{R}}_{k\times 1}. \tag{19}$$

Thus, for every Π_π, we have:

$$\theta^\pi = \left(\Phi^T\left(\mathcal{R} + P\Pi_\pi\Phi\right)\right)^{-1}\Phi^T\mathcal{R}. \tag{20}$$

By defining the notations $A = \left(\Phi^T\left(\mathcal{R} + P\Pi_\pi\Phi\right)\right)^{-1}$ and $b = \Phi^T\mathcal{R}$, we can express the above formula as $\theta^\pi = A^{-1}b$.

The original version of LSPI is an offline algorithm that relies on a batch of samples gathered from the agent's interactions with the environment. However, since most real-world control problems are online, we consider an online variant of LSPI. This version performs policy improvements after a few transitions and updates its policy based on an incomplete evaluation of its current policy.

To solve the projected Bellman equation through TD learning, OLSPI constructs a matrix A and vector b from consecutive interactions according to [25]:

$$\hat{A} \longleftarrow \hat{A} + \phi(s_i, a_i)\left(\phi(s_i, a_i) - \gamma\phi(s_i', a_i')\right)^T. \tag{21}$$

$$\hat{b} \longleftarrow \hat{b} + \phi(s_i, a_i)\, r_i. \tag{22}$$

In each iteration, LSPI reconstructs these matrices from the beginning, while OLSPI keeps updating A and b as long as it remains in interaction with the environment.

The process involves approximating the AVF $Q_\theta^{\pi_{k-1}}$ for policy π_{k-1} in the policy evaluation phase, followed by generating an improved greedy policy π_k in the policy improvement phase. These two steps are repeated iteratively until we reach an optimal policy convergence.

3 Fuzzy-RBF least square policy iteration (FRLSPI)

Function approximation plays a crucial role in real-world control problems where precise models are not feasible or hard to compute, providing a way to estimate and understand complex systems. The fuzzy-RBF network- a hybrid structure constituted by combining Takagi-Sugeno fuzzy rule inference system with a normalized RBF neural network- is an efficient, popular function approximator. It has the advantages of excellent universal approximation property and a quick and robust learning rule. In this paper, we proposed a five-layer fuzzy-RBF network for approximating AVF as shown in Figure 2. The specific interpretation of each layer is outlined below:

(1) The first layer: Input layer

Each node in input layer corresponds a crisp input variable x_i where i is an input variable index. input vector denotes a system state-action variable $x = [s, a]^T = [x_1, x_2, \ldots, x_{n+1}]^T$ which is constructed from $s = [s_1, s_2, \ldots, s_n]$, as well as possible actions $a \in [a_1, a_2, \ldots, a_m]$ at any time step. Input vector $x \in \mathbb{R}^{n+1}$ is fed into the next layer directly.

(2) The second layer: fuzzification layer

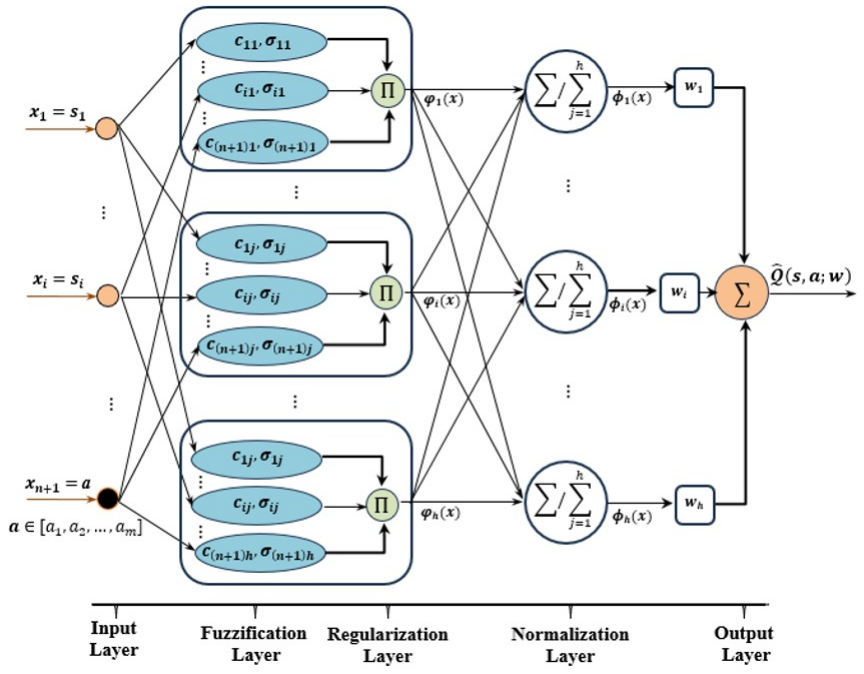


Figure 2: Topology of proposed fuzzy-RBF network model

The linguistic variables are represented by nodes of fuzzification layer, with Gaussian functions used as fuzzy membership functions to fuzzify the input variables.

$$\varphi_{ij} = exp\left(-\frac{\|x_i - c_{ij}\|_2^2}{2{\sigma_{ij}}^2}\right), i = 1, 2, \ldots, n+1, j = 1, 2, \ldots, h. \tag{23}$$

(3) The third layer: regularization layer

Each node corresponds to a specific fuzzy rule, and at time step t, its activation level can be determined through the process of product reasoning:

$$\varphi_j(x_t) = \prod_{i=1}^{n+1} \varphi_{ij}(x_i) = \exp\left(-\sum_{i=1}^{n+1} \frac{\|x_{it} - c_{ij}\|_2^2}{2{\sigma_{ij}}^2}\right) \in [0, 1], j = 1, 2, \ldots, h. \tag{24}$$

$$h = \prod_{i=1}^{n+1} n_i. \tag{25}$$

where, x_{it} is the i^{th} input variable within the input vector x_t at time step t. n_i is the number of gaussian fuzzy sets that composes the partition of i^{th} input and h represents the number of fuzzy rules of the model. The two layers of fuzzification and regularization can be considered as a hidden layer together.

(4) The fourth layer: normalization layer

The output $\phi_j(x_t) = \phi_j(s_t, a_t)$ of the j^{th}node in this layer represents the degree of membership to the fuzzy rule:

$$\phi_j(x_t) \stackrel{\text{def}}{=} \frac{\varphi_j(x_t)}{\sum_{k=1}^{h} \varphi_k(x_t)}, \left\langle \forall x, \sum_{j=1}^{h} \phi_j(x) = 1 \right\rangle. \tag{26}$$

(5) The fifth layer: Output layer

The output of the layer $\widehat{Q} : \mathbb{R}^{n+1} \to \mathbb{R}$ implements a weighted sum of hidden-unit outputs and can be presented by the following parametric model:

$$\widehat{Q}_t(s_t, a_t; w) = \sum_{j=1}^{h} \phi_j(x_t) w_j(t), \tag{27}$$

where $w = (w_1, w_2, \ldots, w_h)$, is the vector of tunable weights and $\phi_1, \ldots, \phi_h$ correspond to the fixed state-action basis functions on the state-action space [8].
With this notation, the approximation can also be stated in the vector form as follows:

$$\widehat{Q}_t(s_t, a_t; w) = \phi^T(s, a)\, w_t, \text{for all}\, a \in A, \tag{28}$$

where $\phi^T(s,a) = [\phi_1(s,a), \phi_2(s,a), \ldots \phi_h(s,a)]$. Defining the state-action space matrix $\Phi \in \mathbb{R}^{|S|.|A| \times h}$ as:

$$\Phi = \begin{bmatrix} \phi_1(s_1,a_1) & \ldots & \phi_h(s_1,a_1) \\ \phi_1(s_1,a_2) & \ldots & \phi_h(s_1,a_2) \\ & \vdots & \\ \phi_1(s_1,a_{|A|}) & \ldots & \phi_h(s_1,a_{|A|}) \\ & \vdots & \\ & \vdots & \\ \phi_1(s_{|S|},a_1) & \ldots & \phi_h(s_{|S|},a_1) \\ \phi_1(s_{|S|},a_2) & \ldots & \phi_h(s_{|S|},a_2) \\ & \vdots & \\ \phi_1(s_{|S|},a_{|A|}) & \ldots & \phi_h(s_{|S|},a_{|A|}) \end{bmatrix}, \tag{29}$$

$$= \begin{bmatrix} | & & | \\ \phi_1(s,a) & \ldots & \phi_h(s,a) \\ | & & | \end{bmatrix} = \begin{bmatrix} \phi^T(s_1,a_1) \\ \vdots \\ \phi^T(s_{|S|},a_{|A|}) \end{bmatrix}, \tag{30}$$

where $\phi_j(s_i, a_k)$ is the normalized output of j^{th} node of fuzzy-RBF network. The entire approximate AVF $\widehat{Q}_t$ with weights w can be stated in the following compact form:

$$\widehat{Q}_t = \Phi\, w_t. \tag{31}$$

A significant challenge in developing an effective fuzzy-RBF network is selecting optimal values for the network's centers and widths. In this paper, these parameters are determined in advance rather than by training. The center vector $c_i = [c_{i1}, c_{i2}, \ldots, c_{in_i}]$ of gaussian membership functions defined for input x_i are set as follows within the interval $[x_{imin}, x_{imax}]$:

$$c_{i1} = x_{i_{\min}},\ i = 1, 2, \ldots, n+1, \tag{32}$$

$$c_{ir} = c_{i(r-1)} + \Delta c_i\ ,\ r = 2, 3, \ldots,\ n_i, \tag{33}$$

where Δc_i is the distance between two adjacent centers and is determined as follows:

$$\Delta c_i = \frac{x_{i_{\max}} - x_{i_{\min}}}{n_i - 1},\ i = 1, 2, \ldots, n+1, \tag{34}$$

with this description, in every problem $h = n_1 \times n_2 \times \ldots \times n_{n+1}$ basis functions on the *n+1*-dimensional state-action space is used to approximate the AVF. We also utilize a scalar standard deviation parameter σ_i for each input x_i calculated as:

$$\sigma_i = \frac{x_{i_{\max}} - x_{i_{\min}}}{\sqrt{n_i}},\ i = 1, 2, \ldots, n+1. \tag{35}$$

The training objective is to update the weight values of the fuzzy-RBF network w to achieve the best approximation of the Action Value Function (AVF). The weight parameters of fuzzy-RBF network are updated by the following relation:

$$\frac{1}{l-1} w_l = \frac{1}{l-1} A_t^{-1} b_t. \tag{36}$$

Where l is a positive integer [11], matrix A and vector b updated as follows:

$$A_{t+1} = A_t + \vec{e}_{t+1}(\phi(s_t, a_t) - \gamma\phi(s_{t+1}, \pi(s_{t+1})))^T, \tag{37}$$

$$b_{t+1} = b_t + \vec{e}_{t+1} r_t. \tag{38}$$

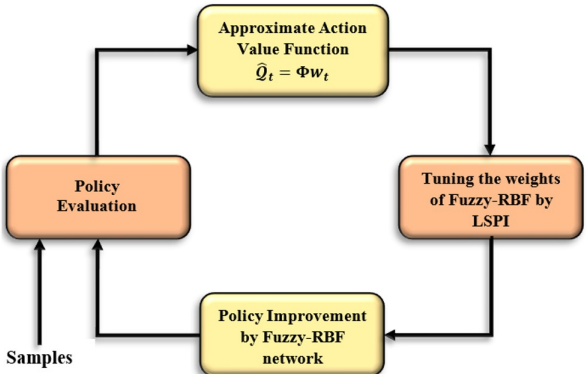


Figure 3: Fuzzy-RBF least square policy iteration

In this formula, $\vec{e}_t$ is the eligibility trace and calculated for FRLSPI according to:

$$\begin{aligned}\vec{e}_t &= \sum_{k=1}^{t} (\lambda\gamma)^{t-k} \frac{\partial Q(s_t, a_t, w_t)}{\partial w(k)} \\ &= \sum_{k=1}^{t-1} (\lambda\gamma)^{t-k} \frac{\partial Q(s_t, a_t, w_t)}{\partial w(k)} + \frac{\partial Q(s_t, a_t, w_t)}{\partial w(t)} \\ &= \gamma\lambda\vec{e}_{t-1} + \frac{\partial Q(s_t, a_t, w_t)}{\partial w(t)} = \gamma\lambda\vec{e}_{t-1} + \phi_j(x_t).\end{aligned} \tag{39}$$

The following relation can be directly concluded from the above formula:

$$\vec{e}_{t+1} = \gamma\lambda\vec{e}_t + \phi_j(x_{t+1}). \tag{40}$$

Where λ is a trace decay parameter.

By utilizing the Sherman-Morrison equation, the FRLSPI algorithm can efficiently update the inverse of the matrix A in one step, thereby reducing the volume of calculations required.

Definition 3.1. *Suppose $M \in \mathbb{R}^{n\times n}$ is an invertible square matrix and $u, v \in \mathbb{R}^n$ are column vectors such that $1 + v^T M^{-1} u \neq 0$, then then **Sherman-Morrison** formula states that* [23]*:*

$$\left(M + uv^T\right)^{-1} = M^{-1} - \frac{M^{-1}uv^T M^{-1}}{1 + v^T M^{-1} u}. \tag{41}$$

Let us denote:

$$u = \phi(s_t, a_t), \tag{42}$$

$$v = \vec{e}_{t+1}\left(\phi^T(s_t, a_t) - \gamma\phi^T(s_{t+1}, a_{t+1})\right). \tag{43}$$

Consequently, the inverse update formula for matrix A can be expressed as follows [19]:

$$A_{t+1}^{-1} = A_t^{-1} - \frac{A_t^{-1}uv^T A_t^{-1}}{1 + v^T A_t^{-1} u}. \tag{44}$$

The block diagram of FRLSPI is depicted in Figure 3, and the general farmwork of on-line FRLSPI algorithm is summarized in Algorithm 1.

It should also be noted that updating the weight vector of fuzzy-RBF corresponds to the update of approximated AVF in FRLSPI.

Algorithm 1 Online fuzzy-RBF Least Square Policy Iteration

Input:
$\mathcal{R}$: Reward function, γ : Discount factor, π_0 : Initial policy, $k = a$ fix number. T_{max} = maximum episode
Initialization:
$A_0 \leftarrow \delta I_{n\times n}; b_0 \leftarrow 0_n; e_0 \leftarrow 0_n; w_0 = 0_n; t = 1, l = 1$
Observe initial state s_0
Loop
Choose a suitable action considering approximate AVFs using decayed ε-greedy policy.
Take action a_t, observe next state s_{t+1} and receive reward r_{t+1}
$A_{t+1} = A_t + \vec{e}_{t+1}\left(\phi\left(s_t, a_t\right) - \gamma\phi\left(s_{t+1}, \pi\left(s_{t+1}\right)\right)\right)^T$
$b_{t+1} = b_t + \vec{e}_{t+1} r_t$
$\vec{e}_{t+1} = \gamma\lambda\vec{e}_t + \phi_j\left(x_{t+1}\right)$
if $l = kt,\ k \in N$ **then**
Solve
$\frac{1}{l-1} W_l = \frac{1}{l-1} A_t^{-1} b_t$
$t \leftarrow t + 1$
end if
$t \leftarrow t + 1$
until the end of episode
$t \leftarrow 1$
until the termination criterion is satisfied.
Output:
policy π (w: weight vector)

4 Theoretical results

In this section, we first prove that a linear combination of the Gaussian basis functions employed in FRLSPI has the ability to uniformly approximate any real continuous function with arbitrary degree of accuracy on a compact space by Stone-Weierstrass theorem [19]. Although it has been proven that fuzzy systems and RBF networks are general approximators, our network model is different from theirs. Then we drive an error bond between the optimal and the approximated AVF over all iterations for FRLSPI.

Theorem 4.1 (Stone-Weierstrass theorem)**.** *Let Ω be a set of real continuous functions on a convex set T satisfying the following criteria:*

1. The set Ω is algebra; that is, the set Ω is closed under multiplication, addition, and scalar multiplication;
2. The set Ω separates points of T, i.e.

$$\forall\, t_1, t_2 \in \mathrm{T}, t_1 \neq t_2 \exists\, \delta\left(t\right) \in \Omega : \delta\left(t_1\right) \neq \delta\left(t_2\right).$$

3. The set Ω vanishes at no point of T, that is,

$$\forall\, t \in \mathrm{T}; \exists\, \delta\left(t\right) \in \Omega : \delta\left(t\right) \neq 0.$$

Then, for any real continuous function $\delta\left(t\right)$ on T and arbitrary $\varepsilon \geq 0$, there exists a function $f \in \Omega$ such that:

$$\sup_{t\epsilon T} \left|f\left(t\right) - \delta\left(t\right)\right| < \varepsilon.$$

Where $\varepsilon > 0$ is a small arbitrary number representing the desired approximation error.

Lemma 4.2 (Universal approximation theorem)**.** *lets Q be a continuous real function on the convex set* T *in* $\Re$*, then for each arbitrary $\varepsilon > 0$, there exists a function in the form of $\widehat{Q}_t\left(s, a; w\right) = \sum_{j=1}^{h} \phi_j\left(x\right) w_j\left(t\right)$ such that:*

$$\sup_{(s,a)} \left|\widehat{Q}\left(s, a; w\right) - Q\left(s, a; w\right)\right| < \varepsilon.$$

Proof. Consider Ω as the space of all functions $\widehat{Q} = \Phi w$, where w is a parameter vector, and Φ is determined by relation [9].

$$\widehat{Q} : \mathrm{T} \to Y,\ y = \widehat{Q}\,(s, a; w)\,,\ \mathrm{T} \subseteq \mathbb{R}^{n+m}, Y \subseteq \mathbb{R}.$$

Now, consider a fuzzy-RBF network in the form defined in section (3) and also consider y as follows:

$$y = \widehat{Q}\,(s, a; w) = \sum_{j=1}^{h} \phi_j\,(x)\, w_j\,(t).$$

Where w, is the vector of tunable weights and $\phi_j\,(x) = \frac{\varphi_j(x)}{\sum_{k=1}^{h} \varphi_k(x)}$ is the normalized output of j^{th} hidden node where $\varphi_j\,(x) = \prod_{i=1}^{n+1} \varphi_{ij}(x)$ and $\varphi_{ij}(x)$ is the i^{th} membership function in the j^{th} rule neuron. So, we can rewrite Eq. (26) and Eq. (27) as:

$$\phi_j\,(x) = \frac{\prod_{i=1}^{n+1} \varphi_{ij}(x)}{\sum_{k=1}^{h} \prod_{i=1}^{n+1} \varphi_{ik}(x)}, \tag{45}$$

$$\begin{aligned} \widehat{Q}\,(s, a; w) &= \sum_{j=1}^{h} \frac{\prod_{i=1}^{n+1} \varphi_{ij}(x)}{\sum_{k=1}^{h} \prod_{i=1}^{n+1} \varphi_{ik}(x)} w_j\,(t) \\ &= \frac{\sum_{j=1}^{h} w_j\,(t) \prod_{i=1}^{n+1} \varphi_{ij}(x)}{\sum_{j=1}^{h} \prod_{i=1}^{n+1} \varphi_{ij}(x)}. \end{aligned} \tag{46}$$

Consider two arbitrary functions, $\widehat{Q}_1$ and $\widehat{Q}_2$, approximated by the fuzzy-RBF network with h_1 and h_2 hidden nodes, respectively:

$$\begin{aligned} \widehat{Q}_1\,(s, a; w) &= \frac{\sum_{j_1=1}^{h_1} w_{j_1}\,(t) \prod_{i=1}^{n+1} \varphi_{ij_1}(x)}{\sum_{j_1=1}^{h_1} \prod_{i=1}^{n+1} \varphi_{ij_1}(x)} \\ &= \frac{\sum_{j_1=1}^{h_1} w_{j_1}\,(t) \prod_{i=1}^{n+1} exp\left(-\frac{\left\|x - c_{ij_1}\right\|^2}{2{\sigma_{ij_1}}^2}\right)}{\sum_{j_1=1}^{h_1} \prod_{i=1}^{n+1} exp\left(-\frac{\left\|x - c_{ij_1}\right\|^2}{2{\sigma_{ij_1}}^2}\right)}, \end{aligned}$$

$$\begin{aligned} \widehat{Q}_2\,(s, a; w) &= \frac{\sum_{j_2=1}^{h_2} w_{j_2}\,(t) \prod_{i=1}^{n+1} \varphi_{ij_2}(x)}{\sum_{j_2=1}^{h_2} \prod_{i=1}^{n+1} \varphi_{ij_2}(x)} \\ &= \frac{\sum_{j_2=1}^{h_1} w_{j_2}\,(t) \prod_{i=1}^{n+1} exp\left(-\frac{\left\|x - c_{ij_2}\right\|^2}{2{\sigma_{ij_2}}^2}\right)}{\sum_{j_2=1}^{h_1} \prod_{i=1}^{n+1} exp\left(-\frac{\left\|x - c_{ij_2}\right\|^2}{2{\sigma_{ij_2}}^2}\right)}, \end{aligned}$$

thus,

$$\begin{aligned} \widehat{Q}_1\,(s, a; w) + \widehat{Q}_2\,(s, a; w) &= \frac{\sum_{j_1=1}^{h_1} w_{j_1}\,(t) \prod_{i=1}^{n+1} \varphi_{ij_1}(x)}{\sum_{j_1=1}^{h_1} \prod_{i=1}^{n+1} \varphi_{ij_1}(x)} + \frac{\sum_{j_2=1}^{h_2} w_{j_2}\,(t) \prod_{i=1}^{n+1} \varphi_{ij_2}(x)}{\sum_{j_2=1}^{h_2} \prod_{i=1}^{n+1} \varphi_{ij_2}(x)} \\ &= \frac{\sum_{j_1=1}^{h_1} \sum_{j_2=1}^{h_2} \left(w_{j_1} + w_{j_2}\right) \left(\prod_{i=1}^{n+1} \left(\varphi_{ij_1}\,(x) + \varphi_{ij_2}(x)\right)\right)}{\sum_{j_1=1}^{h_1} \sum_{j_2=1}^{h_2} \prod_{i=1}^{n+1} \left(\varphi_{ij_1}\,(x) + \varphi_{ij_2}(x)\right)} \\ &= \frac{\sum_{j=1}^{h_1+h_2} w_j\,(t) \prod_{i=1}^{n+1} \varphi_{ij}(x)}{\sum_{j=1}^{h_1+h_2} \prod_{i=1}^{n+1} \varphi_{ij}(x)} \\ &= \sum_{j=1}^{h_1+h_2} w_j\,(t) \phi_j\,(x) = \phi^T\,(s, a)\, w_t. \end{aligned} \tag{47}$$

Where,

$$\phi^T(s,a) = \left[\phi_1^1(s,a), \ldots, \phi_{h_1}^1(s,a), \phi_1^2(s,a), \ldots, \phi_{h_2}^2(s,a)\right],$$
$$w = \left[w_1^1, w_2^1, \ldots, w_{h_1}^1, w_1^2, w_2^2, \ldots, w_{h_2}^2\right].$$

Hence, $\widehat{Q}_1(s,a;w) + \widehat{Q}_2(s,a;w) \in \Omega$ and Ω is closed under summation. Similarly:

$$\begin{aligned}\widehat{Q}_1(s,a;w).\widehat{Q}_2(s,a;w) &= \frac{\sum_{j_1=1}^{h_1} w_{j_1}(t)\prod_{i=1}^{n+1}\varphi_{ij_1}(x)}{\sum_{j_1=1}^{h_1}\prod_{i=1}^{n+1}\varphi_{ij_1}(x)} \cdot \frac{\sum_{j_2=1}^{h_2} w_{j_1}(t)\prod_{i=1}^{n+1}\varphi_{ij_2}(x)}{\sum_{j_2=1}^{h_2}\prod_{i=1}^{n+1}\varphi_{ij_2}(x)} \\ &= \frac{\sum_{j_1=1}^{h_1}\sum_{j_2=1}^{h_2}(w_{j_1}.w_{j_2})\prod_{i=1}^{n+1}\left(\varphi_{ij_1}(x)\varphi_{ij_2}(x)\right)}{\sum_{j_1=1}^{h_1}\sum_{j_2=1}^{h_2}\prod_{i=1}^{n+1}\left(\varphi_{ij_1}(x)\varphi_{ij_2}(x)\right)}. \end{aligned} \tag{48}$$

Since $\varphi_{ij_1}(x)$ and $\varphi_{ij_2}(x)$ are Gaussian in form, their product $\varphi_{ij_1}(x)\varphi_{ij_2}(x)$ is also Gaussian in form; hence, (48) is in the same form as (46). so $\widehat{Q}_1(s,a;w).\widehat{Q}_2(s,a;w) \in \Omega$ that indicates that Ω is closed under multiplication. Finally, for an arbitrary $\rho \in \mathcal{R}$, we have:

$$\rho.\widehat{Q}(s,a;w) = \rho.\frac{\sum_{j=1}^{h} w_j(t)\prod_{i=1}^{n+1}\varphi_{ij}(x)}{\sum_{j=1}^{h}\prod_{i=1}^{n+1}\varphi_{ij}(x)} = \rho.\frac{\sum_{j=1}^{h} w_j(t)\prod_{i=1}^{n+1} exp\left(-\frac{\|x-c_{ij}\|^2}{2{\sigma_{ij}}^2}\right)}{\sum_{j=1}^{h}\prod_{i=1}^{n+1} exp\left(-\frac{\|x-c_{ij}\|^2}{2{\sigma_{ij}}^2}\right)}. \tag{49}$$

It is clear that $\rho.\widehat{Q}(s,a;w) \in \Omega$ that signifies that Ω is closed with respect to scalar multiplication. So, according to Eqs. (47) - (49), it can be concluded that Ω is an algebra, and the first condition of theorem is satisfied for Ω.

In order to verifying the second condition, let $t_1, t_2 \in \mathrm{T}$ be two arbitrary vectors such that $t_1 \neq t_2$; consider a fuzzy-RBF network with two nodes in hidden layer, then $\widehat{Q}$ can be expressed as,

$$\widehat{Q}(s,a;w) = \frac{\sum_{j=1}^{2} w_j(t)\prod_{i=1}^{n+1}\varphi_{ij}(x)}{\sum_{j=1}^{2}\prod_{i=1}^{n+1}\varphi_{ij}(x)} = w_1(t)\frac{\prod_{i=1}^{n+1}\varphi_{ij}(x)}{\sum_{j=1}^{2}\prod_{i=1}^{n+1}\varphi_{ij}(x)} + w_2(t)\frac{\prod_{i=1}^{n+1}\varphi_{ij}(x)}{\sum_{j=1}^{2}\prod_{i=1}^{n+1}\varphi_{ij}(x)}.$$

Thus, $\widehat{Q}(t_1) \neq \widehat{Q}(t_2)$ if $t_1 \neq t_2$. So, Ω separates various points of T, and the second condition of theorem is also verified.

To demonstrate that Ω does not vanish at any point of T, we can easily see that any function in the form of Eq. (46) with $w_j = l \neq 0$ possesses the property of:

$$\forall\, t \in \mathrm{T}, \widehat{Q}(x) = l \neq 0.$$

Therefore, since Ω does not vanish at any point of T, it can be concluded that the three conditions of theorem are met. So, the result follows by Stone-Weierstrass theorem. □

Lemma 4.3. *Let $\{\pi_m\}$ be the sequence generated by FRLSPI algorithm and $\delta_m > 0$ be a positive scalar such that:*

$$\sup_{(s,a)}\left|\left(Q^{\pi_m}(s,a;w) - \widehat{Q}^{\pi_m}(s,a;w)\right)\right| \geq \delta_m.$$

Then,

$$Q^{\pi_{m+1}}(s,a;w) < Q^{\pi_m}(s,a;w) + \frac{2\gamma}{1-\gamma}\,\delta_m, \forall\,(s,a).$$

Proof. Let ε_m be defined according to:

$$\varepsilon_m = \sup_{(s,a)}\left|\left(Q^{\pi_{m+1}}(s,a;w) - Q^{\pi_m}(s,a;w)\right)\right|.$$

From the above equation, we can conclude that:

$$\left|\left(Q^{\pi_{m+1}}(s,a;w) - Q^{\pi_m}(s,a;w)\right)\right| \leq \varepsilon_m, \forall\ s,a.$$

Due to the fact that if $|a-b| < c$ then $b < a+c$ and $a < b+c$, we obtain:

$$Q^{\pi_{m+1}}(s,a;w) \leq Q^{\pi_m}(s,a;w) + \varepsilon_m, \forall\ s,a.$$

By applying $\mathcal{T}_{\pi_{m+1}}$ to both sides of inequality, and considering that operator $\mathcal{T}_\pi$ is monotone (relation 13), we get:

$$\begin{aligned} T_{\pi_{m+1}} Q^{\pi_{m+1}}(s,a;w) &\leq T_{\pi_{m+1}}\left(Q^{\pi_m}(s,a;w) + \varepsilon_m\right) \\ &= T_{\pi_{m+1}} Q^{\pi_m}(s,a;w) + \gamma\varepsilon_m, \end{aligned}$$

since $T_{\pi_{m+1}} Q^{\pi_{m+1}}(s,a;w) = Q^{\pi_{m+1}}(s,a;w)$ then we have:

$$Q^{\pi_{m+1}}(s,a;w) \leq T_{\pi_{m+1}} Q^{\pi_m}(s,a;w) + \gamma\varepsilon_m.$$

Given that, in FRLSPI, we don't need to represent policy approximation, and all errors of actor are considered to be zero. So, for every m, we have: $\mathcal{T}_{\pi_{m+1}} Q^{\pi_m} = \mathcal{T}_{\pi_m} Q^{\pi_m}$. By subtracting $Q^{\pi_m}(s,a;w)$ from both sides of above inequality and then subtracting and adding $\mathcal{T}_{\pi_{m+1}} \widehat{Q}^{\pi_m}(s,a;w)$ to the result, we will have:

$$\begin{aligned} Q^{\pi_{m+1}}(s,a;w) - Q^{\pi_m}(s,a;w) &\leq \mathcal{T}_{\pi_{m+1}} Q^{\pi_m}(s,a;w) + \gamma\varepsilon_m - Q^{\pi_m}(s,a;w) \\ &= \mathcal{T}_{\pi_{m+1}} Q^{\pi_m}(s,a;w) - \mathcal{T}_{\pi_{m+1}} \widehat{Q}^{\pi_m}(s,a;w) \\ &\quad + \mathcal{T}_{\pi_{m+1}} \widehat{Q}^{\pi_m}(s,a;w) - Q^{\pi_m}(s,a;w) + \gamma\varepsilon_m, \end{aligned}$$

thus,

$$\begin{aligned} &\left\|Q^{\pi_{m+1}}(s,a;w) - Q^{\pi_m}(s,a;w)\right\|_\infty \leq \\ &\left\|\mathcal{T}_{\pi_{m+1}} Q^{\pi_m}(s,a;w) - \mathcal{T}_{\pi_{m+1}} \widehat{Q}^{\pi_m}(s,a;w)\right\|_\infty + \left\|\mathcal{T}_{\pi_{m+1}} \widehat{Q}^{\pi_m}(s,a;w) - Q^{\pi_m}(s,a;w) + \gamma\varepsilon_m\right\|_\infty. \end{aligned} \tag{50}$$

Now, we quantify the two terms on the right-hand side of (50) respectively. First, catch on to the fact that in FRLSPI $\mathcal{T}_{\pi_{m+1}} Q^{\pi_m} = \mathcal{T}_{\pi_m} Q^{\pi_m}$. hence,

$$\left\|\mathcal{T}_{\pi_{m+1}} Q^{\pi_m} - \mathcal{T}_{\pi_{m+1}} \widehat{Q}^{\pi_m}\right\|_\infty = \left\|\mathcal{T}_{\pi_m} Q^{\pi_m} - \mathcal{T}_{\pi_m} \widehat{Q}^{\pi_m}\right\|_\infty \leq \gamma\left\|Q^{\pi_m} - \widehat{Q}^{\pi_m}\right\|_\infty. \tag{51}$$

Meanwhile, by the definition of the operator $\mathcal{T}_{\pi_m}$ ($\mathcal{T}_{\pi_m} Q^{\pi_m} = Q^{\pi_m}$), we have:

$$\left\|\mathcal{T}_{\pi_{m+1}} \widehat{Q}^{\pi_m} - Q^{\pi_m} + \gamma\varepsilon_m\right\|_\infty = \left\|\mathcal{T}_{\pi_m} Q^{\pi_m} - \mathcal{T}_{\pi_m} \widehat{Q}^{\pi_m}\right\|_\infty + \gamma\varepsilon_m \leq\ \gamma\left\|Q^{\pi_m} - \widehat{Q}^{\pi_m}\right\|_\infty + \gamma\varepsilon_m, \tag{52}$$

plugging (51) and (52) into (50), we obtain,

$$\begin{aligned} \left\|Q^{\pi_{m+1}}(s,a;w) - Q^{\pi_m}(s,a;w)\right\|_\infty &\leq \gamma\left\|Q^{\pi_m}(s,a;w) - \widehat{Q}^{\pi_m}(s,a;w)\right\|_\infty + \gamma\left\|Q^{\pi_m}(s,a;w) - \widehat{Q}^{\pi_m}(s,a;w)\right\|_\infty + \gamma\varepsilon_m \\ &= 2\gamma\left\|Q^{\pi_m}(s,a;w) - \widehat{Q}^{\pi_m}(s,a;w)\right\|_\infty + \gamma\varepsilon_m. \end{aligned} \tag{53}$$

Since $\left\|Q^{\pi_{m+1}} - Q^{\pi_m}\right\|_\infty = \sup_{(s,a)} \left|Q^{\pi_{m+1}} - Q^{\pi_m}\right|$ we can rewrite (53) as:

$$\sup_{(s,a)} \left|Q^{\pi_{m+1}}(s,a;w) - Q^{\pi_m}(s,a;w)\right| \leq 2\gamma \sup_{(s,a)} \left|\left(Q^{\pi_m}(s,a;w) - \widehat{Q}^{\pi_m}(s,a;w)\right)\right| + \gamma\varepsilon_m. \tag{54}$$

By assumption, $\sup_{(s,a)} \left|\left(Q^{\pi_m}(s,a;w) - \widehat{Q}^{\pi_m}(s,a;w)\right)\right| \leq \delta_m$ thus, it can be concluded that,

$$\sup_{(s,a)} \left|Q^{\pi_{m+1}}(s,a;w) - Q^{\pi_m}(s,a;w)\right| \leq 2\gamma\delta_m + \gamma\varepsilon_m.$$

And consequently,

$$\varepsilon_m \leq 2\gamma\delta_m + \gamma\varepsilon_m \Rightarrow \varepsilon_m \leq \frac{2\gamma}{1-\gamma}\delta_m.$$

Since $Q^{\pi_{m+1}}(s,a;w) \leq Q^{\pi_m}(s,a;w) + \varepsilon_m$ and $\varepsilon_m \leq \frac{2\gamma}{1-\gamma}\delta_m$ then can conclude $Q^{\pi_{m+1}}(s,a;w) \leq Q^{\pi_m}(s,a;w) + \frac{2\gamma}{1-\gamma}\delta_m$ and we get the desired result. Now we are ready to present the main theorem. □

Theorem 4.4 (Performance bond for FRLSPI)**.** *Let $\pi_1, \pi_2, \ldots, \pi_m$ and $\widehat{Q}^{\pi_1}, \widehat{Q}^{\pi_2}, \ldots, \widehat{Q}^{\pi_m}$ be the sequence of policies and corresponding approximate AVFs produced by FRLSPI algorithm, respectively. Let $\delta \in (0, \infty)$, be an arbitrary real number that limits the errors between the approximate and the true AVFs throughout all iterations. if*

$$\forall\ m = 0, 1, 2, \ldots \left\| \widehat{Q}^{\pi_m} - Q^{\pi_m} \right\|_\infty \leq \delta_m.$$

Then, the sequence of policies generated by FRLSPI is guaranteed to be lied within a bounded distance from the optimal performance eventually.

$$\lim_{m \to \infty} \sup \left\| \widehat{Q}^{\pi_m} - Q^* \right\|_\infty < \frac{1 + \gamma^2}{(1-\gamma)^2} \delta.$$

Where $\delta = lim_{m \to \infty} \sup \delta_m$ and Q^* corresponds the optimal AVF.

Proof. Consider:

$$g_m = \sup_{(s,a)} \left| Q^{\pi_m}(s, a; w) - Q^*(s, a; w) \right| .$$

Thus,

$$Q^{\pi_m}(s, a; w) \leq Q^*(s, a; w) + g_m\ ,\ \forall (s, a) .$$

By applying operator $\mathcal{T}$ to both sides of inequality, we have:

$$\mathcal{T} Q^{\pi_m}(s, a; w) \leq \mathcal{T} \left(Q^*(s, a; w) + g_m \right) = \mathcal{T} Q^*(s, a; w) + \gamma g_m,$$

considering the point that $TQ^* = Q^*$ we get:

$$\mathcal{T} Q^{\pi_m}(s, a; w) \leq Q^*(s, a; w) + \gamma g_m. \tag{55}$$

Using the assumption, we have:

$$Q^{\pi_m}(s, a; w) \leq \widehat{Q}^{\pi_m}(s, a; w) + \delta_m,$$

applying $\mathcal{T}_{\pi_{m+1}}$ to both sides of the above inequality result in:

$$\begin{aligned}
\mathcal{T}_{\pi_{m+1}} Q^{\pi_m}(s, a; w) &\leq \mathcal{T}_{\pi_{m+1}} \left(\widehat{Q}^{\pi_m}(s, a; w) + \delta_m \right) \\
&= T_{\pi_{m+1}} \widehat{Q}^{\pi_m}(s, a; w) + \gamma \delta_m \\
&= T_{\pi_m} \widehat{Q}^{\pi_m}(s, a; w) + \gamma \delta_m \\
&\leq T_{\pi_m} \left(Q^{\pi_m}(s, a; w) + \delta_m \right) + \gamma \delta_m \\
&= T_{\pi_m} Q^{\pi_m}(s, a; w) + \gamma \delta_m + \gamma \delta_m.
\end{aligned} \tag{56}$$

By combining (55) and (56) we can conclude:

$$\mathcal{T}_{\pi_{m+1}} Q^{\pi_m}(s, a; w) \leq Q^*(s, a; w) + \gamma g_m + 2\gamma \delta_m. \tag{57}$$

Moreover,

$$\begin{aligned}
Q^{\pi_{m+1}}(s, a; w) &= T_{\pi_{m+1}} Q^{\pi_{m+1}}(s, a; w) \\
&\leq T_{\pi_{m+1}} \left(Q^{\pi_m}(s, a; w) + \varepsilon_m \right) \\
&= T_{\pi_{m+1}} Q^{\pi_m}(s, a; w) + \gamma \varepsilon_m,
\end{aligned} \tag{58}$$

using (57),

$$T_{\pi_{m+1}} Q^{\pi_m}(s, a; w) + \gamma \varepsilon_m \leq Q^*(s, a; w) + \gamma g_m + 2\gamma \delta_m + \gamma \varepsilon_m.$$

Thus,

$$Q^{\pi_{m+1}}(s, a; w) \leq Q^*(s, a; w) + \gamma g_m + 2\gamma \delta_m + \gamma \varepsilon_m.$$

By subtracting $Q^*(s, a; w)$ from both sides of above inequality, we get:

$$Q^{\pi_{m+1}}(s, a; w) - Q^*(s, a; w) \leq \gamma g_m + 2\gamma \delta_m + \gamma \varepsilon_m\ ,\ \forall (s, a) .$$

By taking supremum on both sides, we can conclude:

$$g_{m+1} \leq \gamma g_m + \gamma \varepsilon_m + 2\gamma\delta_m.$$

Taking the limit as $m \to \infty$ on both sides, we have the following:

$$\lim_{m\to\infty} \sup g_{m+1} \ \leq \gamma \lim_{m\to\infty} \sup g_m \ + \gamma \lim_{m\to\infty} \sup \varepsilon_m + 2\gamma\delta.$$

Since $lim_{m\to\infty} \sup g_{m+1} = lim_{m\to\infty} \sup g_m$ and also according to Lemma 2, $lim_{m\to\infty} \sup \varepsilon_m \leq \frac{2\gamma}{1-\gamma}\delta$ then we can rewrite the above inequality as:

$$\lim_{m\to\infty} \sup g_m \ \leq \gamma \lim_{m\to\infty} \sup g_m \ + \gamma\frac{2\gamma}{1-\gamma}\delta + 2\gamma\delta.$$

Simplifying yields,

$$(1-\gamma) \lim_{m\to\infty} \sup g_m \ \leq \gamma\frac{2\gamma}{1-\gamma}\delta + 2\gamma\delta = \frac{2\gamma}{1-\gamma}\delta \Longrightarrow \lim_{m\to\infty} \sup g_m \leq \frac{2\gamma}{(1-\gamma)^2} \ \delta,$$

or equivalently,

$$\lim_{m\to\infty} \sup \|Q^{\pi_m} - Q^*\|_\infty \ < \frac{2\gamma}{(1-\gamma)^2}\delta,$$

in the following we establish an upper bond for $\widehat{Q}^{\pi_m} - Q^*$.

$$\left\|\widehat{Q}^{\pi_m} - Q^*\right\|_\infty = \left\|\widehat{Q}^{\pi_m} - Q^{\pi_m} + Q^{\pi_m} - Q^*\right\|_\infty.$$

Thus,

$$\lim_{m\to\infty} \sup \left\|\widehat{Q}^{\pi_m} - Q^*\right\|_\infty \ = \lim_{m\to\infty} \sup \left\|\widehat{Q}^{\pi_m} - Q^{\pi_m} + Q^{\pi_m} - Q^*\right\|_\infty.$$

By using the triangle inequality:

$$\lim_{m\to\infty} \sup \left\|\widehat{Q}^{\pi_m} - Q^*\right\|_\infty \ \leq \lim_{m\to\infty} \sup \left\|\widehat{Q}^{\pi_m} - Q^{\pi_m}\right\|_\infty \ + \lim_{m\to\infty} \sup \|Q^{\pi_m} - Q^*\|_\infty \ .$$

Since $lim_{m\to\infty} \sup \left\|\widehat{Q}^{\pi_m} - Q^{\pi_m}\right\|_\infty = lim_{m\to\infty} \delta_m = \delta$ therefore,

$$\lim_{m\to\infty} \sup \left\|\widehat{Q}^{\pi_m} - Q^*\right\|_\infty \ \leq \delta + \frac{2\gamma}{(1-\gamma)^2}\delta. \tag{59}$$

Then, our bound in the inequality in (59) can be rewritten as the following:

$$\lim_{m\to\infty} \sup \left\|\widehat{Q}^{\pi_m} - Q^*\right\|_\infty \ < \frac{1+\gamma^2}{(1-\gamma)^2}\delta, \tag{60}$$

and the proof is complete. □

5 Simulation

In order to evaluate and compare the performance and properties of described algorithm, two experiments on mountain-car problem and acrobot swing-up task are performed.

5.1 Mountain-car problem:

Mountain-car is a standard typical benchmark in the reinforcement control literature that has been widely studied by researchers and used in RL competitions to evaluate new algorithms [19, 21, 23]. The goal is to control the acceleration of an underpowered car from the lowest point of a valley to drive up a steep mountain road, as suggested by Figure 4. Since the car's engine is too weak to overcome the gravitational force and traverse the steep slope in one attempt. Thus, it needs to reverse up the left-hand side of the valley to build sufficient potential energy. This means that the agent must choose a counterintuitive, backward trajectory, receiving additional punishment, to find the solution. The

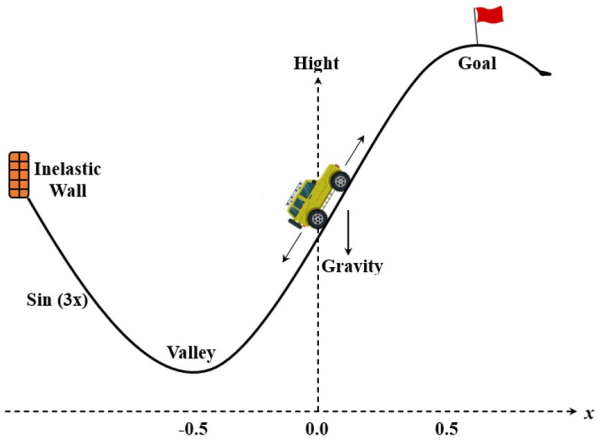


Figure 4: Schematic of the mountain car environment

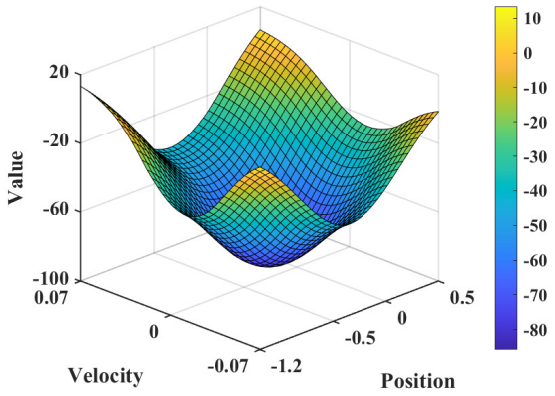


Figure 5: The value function of mountain car

Table 3: Performance comparisons in terms of average return in the mountain-car domain

Method	FRLSPI	LSPI	KLSPI	RPI	HKLSPI	RBLSPI	REPS	DDPG	DDQN
Mean Return	−99.8	−202.3	−179.1	−173.8	−131.8	−110	−275.6	−288.4	−135.7

movement of the car is described by its's position $x \in [-1.2, 0.5]$ and velocity $v \in [-0.07, 0.07]$, and there are three possible actions $a \in \{+1, 0, -1\}$ - throttle forward, throttle backward, and no throttle. The topology of the mountain road where the car moves is governed by $H = \sin(3x)$. The state transition is defined by the following Eqs. [21]:

$$\begin{aligned} v_{t+1} &= \min\{0.07, \max(-0.07, v_t + 0.001 \times a_t + 0.0025 \times \cos(3x_t))\}, \\ x_{t+1} &= \min\{0.5, \max(-1.2, x_t + v_{t+1})\}, \end{aligned} \tag{61}$$

with the extra condition that the position $x_t = -1.2$, is an inelastic wall: if the car reaches this point, its speed is set to zero.

5.1.1 Learning details

For this problem, the inputs of fuzzy-RBF network $\boldsymbol{x} = [s, a]^T$ constructed from the car's current position and velocity $s = [x, v]$, as well as $a \in \mathcal{A} = [-1, 0, +1]$. The experimental results are obtained by averaging over ten independent runs. Each run includes 300 episodes. Each episode starts from the lowest point of the valley ($x = -0.5$) and terminates when either the car reaches the goal ($x \geq 0.5$) or 300 movements are performed. The reward is always $r(s, a) = -1.0$ until the car reaches the goal position.

In our work, for all experiments, a discount factor of $\gamma = 0.99$ is used, and also, decayed ε-greedy action selection policy is used to balance between exploration and exploitation. The algorithm initially starts with $\varepsilon = 0.01$ in every iteration and decreases the exploration rate in a controlled way throughout the execution by $\varepsilon = \varepsilon \times 0.98$.

5.1.2 Simulation results

Figure 5 illustrates, the approximated AVF of FRLSPI on mountain-car domain. As it is seen, the approximated AVF effectively generates a good policy and satisfactorily captures the main topology of the true AVF. Table 3 reports the results of online FRLSPI in terms of average return over all training iterations in comparison with LSPI, RPI, KLSPI, HKLSPI and DDPG reported by refs. [9, 16, 29, 30], based on 300 iterations training trials. As the results show, FRLSPI could outperform other methods in terms of average return. Also, in Figure 6, the performance of four algorithms from the LSPI family is compared with the FRLSPI algorithm. As it is demonstrated, FRLSPI performed considerably better than other methods in terms of convergence speed and learning quality. In Figure 7, the weight value changes corresponding to the first four normalized weights of FRLSPI are depicted. As it is seen, weight values completely converge after a few iterations, which correspond to finding the optimal solution.

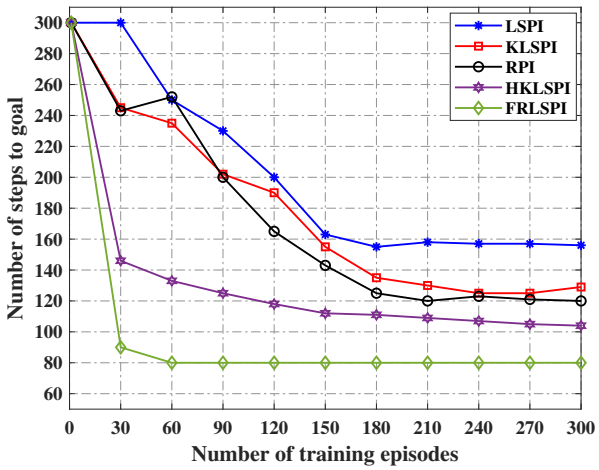


Figure 6: Performance comparisons between LSPI, KLSPI, RPI, HKLSPI and FRLSPI

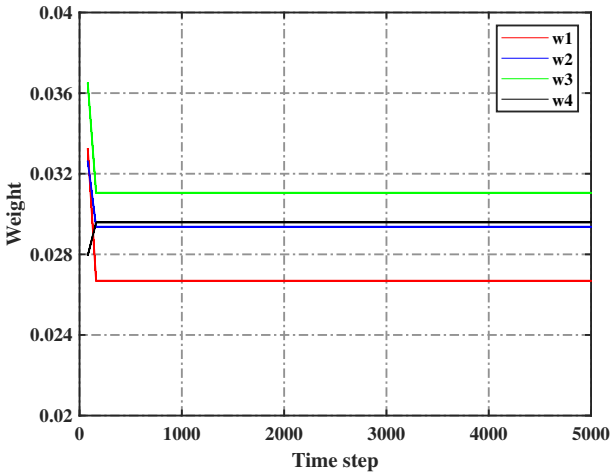


Figure 7: The changes of the first four normalized weights of FRLSPI (w_1, w_2, w_3, w_4) for $\varepsilon_0 = 0.01$ and $\gamma = 0.99$

5.2 The swing-up acrobot

In the following, FRLSPI as an on-line learning control approach is examined on acrobot swing-up task. The acrobot, a two-link planar robot, consists of un-actuated systems is quite challenging to control because it is nonlinear, non-differentially flat, not feedback linearizable, and continuous system [24]. It has been roughly compared to a gymnast swinging on a horizontal bar (Figure 8). The first link is suspended from a point and the second is able to exert torque. The control objective is to swing up the free end from its hanging position until it reaches a designated target height in as few steps as possible (we used a height of 1). Since the acrobot is under-actuated, it cannot do this directly; instead, the successful strategy consists in swing back and forth to gain momentum. The system is described by four continuous observation variables, which comprise two joint positions and two joint velocities. There are no limitations on the joint positions, but the angular velocities are restricted to $\dot{\theta}_1 \in [-4\pi, 4\pi]$ and $\dot{\theta}_2 \in [-9\pi, 9\pi]$. The equations of motion of the system used in the current study are the same as the one in [24]:

$$\ddot{\theta}_1 = -{d_1}^{-1}\left(d_2\,\ddot{\theta}_2 + \phi_1\right),$$
$$\ddot{\theta}_2 = \left(m_2{l_{c2}}^2 + I_2 - \frac{{d_2}^2}{d_1}\right)^{-1}\left(\tau + \frac{d_2}{d_1}\dot{\phi}_1 - \dot{\phi}_2\right). \tag{62}$$

Where

$$\begin{aligned}
d_1 &= m_1{l_{c1}}^2 + m_2\left({l_1}^2{l_{c2}}^2 + 2l_1l_2\cos\theta_2\right) + I_1 + I_2,\\
d_2 &= m_2\left({l_{c2}}^2 + l_1l_{c2}\cos\theta_2\right) + I_2,\\
\dot{\phi}_1 &= -m_2l_1l_{c2}\dot{\theta}_2^2\sin\theta_2 - 2m_2l_1l_{c2}\dot{\theta}_2\dot{\theta}_1\sin\theta_2 + \ (m_1l_{c1} + m_2l_1)\,g\cos\left(\theta_1 - \pi/2\right) + \dot{\phi}_2,\\
\dot{\phi}_2 &= m_2l_{c2}g\cos\left(\theta_1 + \theta_2 - \pi/2\right).
\end{aligned} \tag{63}$$

It is noteworthy that the above equations are used solely for simulation purposes, and from the perspective of the agent (the control component), the dynamics of the system remain unknown.

5.2.1 Learning details

Each training episode begins with $(\theta_1,\theta_2,\dot{\theta}_2,\dot{\theta}_2) = (0,0,0,0)$, which corresponds to vertically downward position and motionless and terminates either when the goal is reached, or when 500 movements are performed. In order to encourage the learning agent to reach the goal in minimum time steps, the reward function is defined as follows:

$$r_t = \begin{cases} 0 & \text{if} \quad s = s_T \\ -1 & \text{otherwise} \end{cases} \tag{64}$$

where s_T specifies the goal state.

The learning procedure, discount factor, and exploration rate are set similarly to those of the previous example. The inputs into the fuzzy-RBF network $x = [s, a]^T = [x_1, x_2, x_3, x_4, x_5]^T$ are the angles θ_1, θ_2 and the angular velocities $\dot{\theta}_1$, $\dot{\theta}_2$ where the subscripts correspond to upper (1) and lower (2) links as well as $x_5 = a \in [-1, 0, +1]$. In the simulation a time step of $t_i = 0.05$ seconds was used, and actions were selected after every four-time steps ($t_c = 0.2$). Table 4 defines the parameters used in obtaining the results.

Table 4: Parameters of Simulated Acrobot

Model Parameter	***Symbol***	***Value***	***Unit***
Link lengths	l_1, l_2	1.0	m
Joint to mass center	lc_1, lc_2	0.5	m
Link masses	m_1, m_2	1.0	kg
Link inertias	I_1, I_2	1.0	$kg.m^2$
Torque range	τ	$\{-1, 0, 1\}$	$N.m$
Gravitational force	g	9.	$m\backslash s^2$
Integration time step	t_i	0.05	s
Control time step	t_c	0.2	s

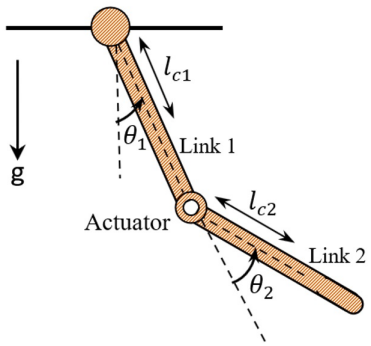


Figure 8: Acrobot Schematic

5.2.2 Simulation results

Figs. 9-12 show the angular changes of θ_1 and θ_2 for the acrobot controlled by the policies of FRLSPI and KLSPI, respectively. As it is seen, FRLSPI converges and stabilizes to near-optimal policy very soon, and also, the learned policy in FRLSPI exhibits significantly higher quality compared to KLSPI, enabling the acrobot to be swung up in just 37-time steps, surpassing KLSPI's 52 steps and outperforming the near-optimal policies (around 70 steps) obtained by previous RL algorithms [9], [10]. And also, it is so much better than DDQN, H_2, H_∞, and $H_{\infty,f}$ with 98, 146, 111 and 99 time steps, respectively [28], as well as LSPI with 315 time steps reported by [2].

The changes of the first three weight values of FRLSPI are depicted in Figure 13. As it is demonstrated, the weight values in FRLSPI have been converged to stationary points surprisingly fast in the acrobot problem.

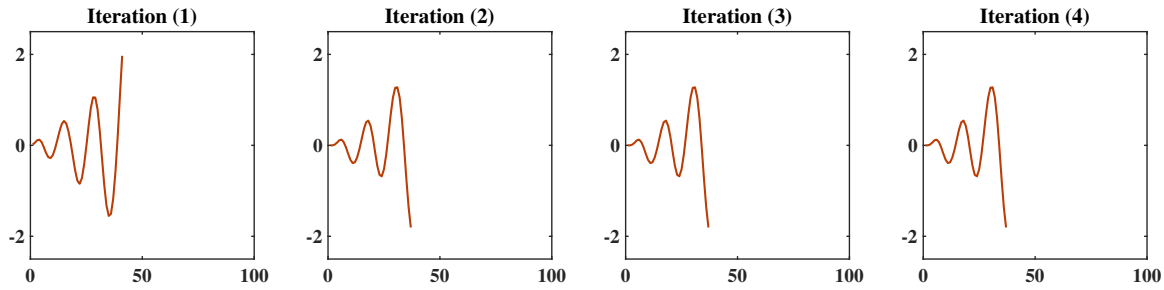


Figure 9: Angular changes of first link (θ_1) for the acrobot controlled by FRLSPI

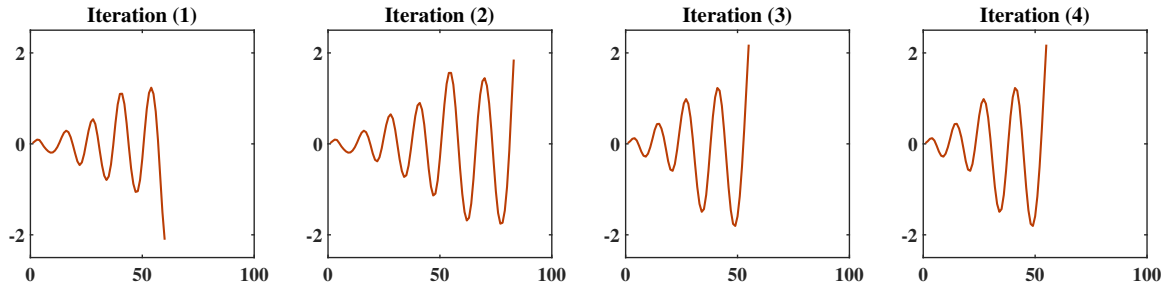


Figure 10: Angular changes of first link (θ_1) for the acrobot controlled by KLSPI

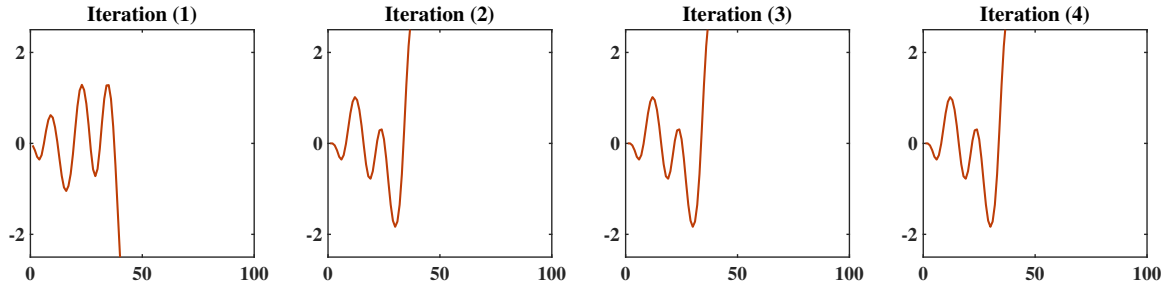


Figure 11: Angular changes of second link (θ_2) for the acrobot controlled by FRLSPI

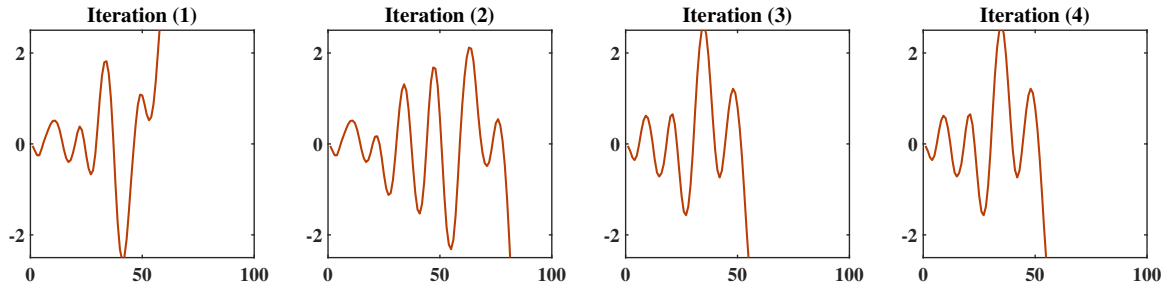


Figure 12: Angular changes of second link (θ_2) for the acrobot controlled by KLSPI

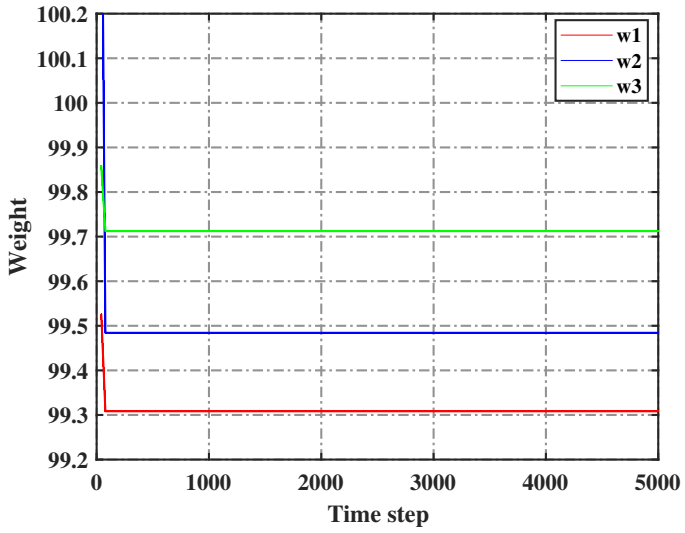


Figure 13: The changes of the first three weight values of FRLSPI (w_1, w_2, w_3) for $\varepsilon_0 = 0.01$ and $\gamma = 0.99$

6 Conclusion

In the present work, we introduced a novel critic-only Takagi-Sugeno type neuro-fuzzy Reinforcement Learning algorithm, called FRLSPI. It is achieved by combining LSPI with a fuzzy-RBF network as a function approximator. The most notable accomplishment of this paper is the positive theoretical analysis concerning an error bond between the optimal and the approximated AVF over all iterations. Moreover, we proved that linear combination of the Gaussian basis functions used in FRLSPI are able to uniformly approximate any continuous complex function as closely as desired over a compact interval. These results are important from different points of view. First, to our knowledge, the theoretical analysis presented for FRLSPI is the first reliable guarantee of near-optimality for policies generated by critic-only neuro-fuzzy algorithms in solving optimal control problems, where policies are adjusted after a few state transitions. Second, FRLSPI is an implementation of LSPI with fuzzy-RBF network. With this implementation, we addressed the two obstacles outlined in the introduction: determining state-action basis functions while satisfy the conditions specified by LSPI theorem and learning rate dependency. Third, proving that the proposed algorithm finally generates policies that are within a limited distance from the optimal performance guarantees to return a near-optimal policy as it was demonstrated experimentally in two continuous state-space environments. Fourth, due to employing LSPI to tune the weight values of fuzzy-RBF network, the proposed method has comparatively fast convergence speed.

References

[1] D. Allahverdy, A. Fakharian, M. B. Menhaj, *Back-stepping integral sliding mode control with iterative learning control algorithm for quadrotor UAVs*, Journal of Electrical Engineering and Technology, **14**(6) (2019), 2539-2547. https://doi.org/10.1007/s42835-019-00257-z

[2] B. André, C. Anderson, *Restricted gradient-descent algorithm for value-function approximation in reinforcement learning*, Artificial Intelligence, **172**(4-5) (2008), 454-482. https://doi.org/10.1016/j.artint.2007.08.001

[3] A. Barakat, P. Bianchi, J. Lehmann, *Analysis of a target-based actor-critic algorithm with linear function approximation*, arXiv preprint arXiv:2106.07472, (2021). https://doi.org/10.48550/arXiv.2106.07472

[4] D. P. Bertsekas, J. N. Tsitsiklis, *Neuro-dynamic programming*, Belmont, MA: Athena Scientific, 1996.

[5] L. Buşoniu, D. Ernst, B. De Schutter, R. Babuška, *Online least-squares policy iteration for reinforcement learning control*, Proceedings of the 2010 American Control Conference, Baltimore, MD, USA, (2010), 486-491. https://doi.org/10.1109/ACC.2010.5530856

[6] L. Buşoniu, A. Lazaric, M. Ghavamzadeh, R. Munos, R. Babuška, B. De Schutter, *Least-squares methods for policy iteration*, In: Wiering, M., van Otterlo, M. (Eds.), Reinforcement Learning: State-of-the-Art. In: Adaptation,

Learning, and Optimization, **12**, Springer, Heidelberg, Germany, (2012), 75-109. https://doi.org/10.1007/978-3-642-27645-3-3

[7] Y. Cui, T. Matsubara, K. Sugimoto, *Kernel dynamic policy programming: Applicable reinforcement learning to robot systems with high dimensional states*, Neural Networks, **94** (2017), 13-23. https://doi.org/10.1016/j.neunet.2017.06.007

[8] V. Derhami, V. J. Majd, M. N. Ahmadabadi, *Fuzzy sarsa learning and the proof of existence of its stationary points*, Asian Journal of Control, (2008), 535-549. https://doi.org/10.1002/asjc.54

[9] Y. Duan, X. Chen, R. Houthooft, J. Schulman, P. Abbeel, *Benchmarking deep reinforcement learning for continuous control*, arXiv preprint arXiv:1604.06778, (2016). https://doi.org/10.48550/arXiv.1604.06778

[10] K. Främling, *Light-weight reinforcement learning with function approximation for real-life control tasks*, In Proceedings of 5th International Conference on Informatics in Control, Automation and Robotics, Funchal, Madeira, Portugal, (2008), 127-134. https://doi.org/10.5220/0001484001270134

[11] F. Ghorbani, V. Derhami, M. Afsharchi, *Fuzzy least square policy iteration and its mathematical analysis*, International Journal of Fuzzy Systems, (2016), 1-14. https://doi.org/10.1007/s40815-016-0270-1

[12] R. A. Howard, *Dynamic programming and Markov processes*, New York: Wiley, 1960.

[13] K. S. Hwang, S. W. Tan, M. C. Tsai, *Reinforcement learning to adaptive control of nonlinear systems*, IEEE Transactions on Systems, Man, and Cybernetics-Part B, **33**(3) (2003), 514-521. https://doi.org/10.1109/TSMCB.2003.811112

[14] H. S. Jakab, L. Csató, *Sparse approximations to value functions in reinforcement learning*, In: Koprinkova-Hristova, P., Mladenov, V., Kasabov, N. (eds) Artificial Neural Networks. Springer Series in Bio-/Neuroinformatics, **4**, Springer, Cham, (2015). https://doi.org/10.1007/978-3-319-09903-3-14

[15] Y. Jia, X. Y. Zhou, *Policy gradient and actor-critic learning in continuous time and space: Theory and algorithms*, Journal of Machine Learning Research, **23**(275) (2022), 12603-12652. https://doi.org/10.2139/ssrn.3969101

[16] M. G. Lagoudakis, R. Par, *Least-squares policy iteration*, Journal of Machine Learning Research, (2003), 1107-1249. https://doi.org/10.1162/1532443041827907

[17] Y. J. Liu, L. Tang, S. Tong, C. P. Chen, D. J. Li, *Reinforcement learning design-based adaptive tracking control with less learning parameters for nonlinear discrete-time MIMO systems*, IEEE Transactions on Neural Networks and Learning Systems, **26**(1) (2015), 165-176. https://doi.org/10.1109/TNNLS.2014.2360724

[18] I. Nishikawa, K. Matsunaga, *An unsupervised learning of a layered network and its application to a motion acquisition*, Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290), Honolulu, HI, USA, **2** (2002), 1667-1672. https://doi.org/10.1109/IJCNN.2002.1007768

[19] W. Rudin, *Principles of mathematical analysis*, 3rd ed. New York, NY, USA: McGraw-Hill Education, 1976.

[20] B. Saglam, C. Cicek, F. Multu, S. Kozat, *Off-Policy correction for actor-critic algorithms in deep reinforcement learning*, arXiv, (2022). https://doi.org/10.48550/arXiv.2208.00755

[21] Second Annual Reinforcement Learning Competition. http://rl-competition.org.

[22] A. Sheikhlar, A. Fakharian, *Online policy iteration-based tracking control of four wheeled omni-directional robots*, Journal of Dynamic Systems, Measurement, and Control, **140**(8) (2018), 081017. https://doi.org/10.1115/1.4039287

[23] J. Sherman, W. J. Morrison, *Adjustment of an inverse matrix corresponding to a change in one element of a given matrix*, The Annals of Mathematical Statistics, **21**(1) (1950), 124-127. https://doi.org/10.1214/aoms/1177729893

[24] N. Snehal, W. Pooja, K. Sonam, S. R. Wagh, N. M. Singh, *Control of an acrobot system using reinforcement learning with probabilistic policy search*, 2021 Australian and New Zealand Control Conference (ANZCC), Gold Coast, Australia, (2021), 68-73. https://doi.org/10.1109/ANZCC53563.2021.9628194

[25] E. H. Sumiea, S. J. Abdulkadir, H. Alhussian, S. M. Al-Selwi, A. Alqushaibi, M. G. Ragab, S. M. Fati, *Deep deterministic policy gradient algorithm: A systematic review*, Heliyon, **10** (2024). https://doi.org/10.1016/j.heliyon.2024.e30697

[26] R. S. Sutton, A. G. Bareto, *Reinforcement learning: An introduction*, Second Edition, MIT Press, Massachusetts, 2017.

[27] N. Tziortziotis, C. Dimitrakakis, M. Vazirgiannis, *Randomised bayesian least-squares policy iteration*, arXiv, (2019). https://doi.org/10.48550/arXiv.1904.03535

[28] B. Varga, B. Kulcsár, M. H. Chehreghani, *Deep Q-learning: A robust control approach*, International Journal of Robust and Nonlinear Control, **33**(1) (2023), 526-554. https://doi.org/10.1038/nature14236

[29] X. Xu, D. Hu, X. Lu, *Kernel-based least squares policy iteration for reinforcement learning*, In IEEE Transactions on Neural Networks, **18**(4) (2007), 973-992. https://doi.org/10.1109/TNN.2007.899161

[30] X. Xu, C. Liu, S. X. Yang, D. Hu, *Hierarchical approximate policy iteration with binary-tree state space decomposition*, In IEEE Transactions on Neural Networks, **22**(12) (2011), 1863-1877. https://doi.org/10.1109/TNN.2011.2168422

[31] S. Yahyaa, B. Manderick, *Knowledge gradient for online reinforcement learning*, In: Duval, B., van den Herik, J., Loiseau, S., Filipe, J. (Eds.), Agents and Artificial Intelligence. In: ICAART 2014 LNCS, **8946**, Springer, Cham, (2014), 103-118. https://doi.org/10.1007/978-3-319-25210-0-7

[32] M. Zaki, A. Mohan, A. Goplan, S. Manner, *Actor-critic based improper reinforcement learning*, arXiv, (2022). https://doi.org/10.48550/arXiv.2207.09090