Technical report 09-047

Approximate dynamic programming with a fuzzy parameterization*

L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška

If you want to cite this report, please use the following reference instead:
L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, "Approximate dynamic programming with a fuzzy parameterization," *Automatica*, vol. 46, no. 5, pp. 804–814, May 2010. doi:10.1016/j.automatica.2010.02.006

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/09_047.html

Approximate Dynamic Programming with a Fuzzy Parameterization [⋆]

Lucian Busoniu [a], Damien Ernst [c], Bart De Schutter [a,b], Robert Babuška [a]

[a] *Delft Center for Systems & Control, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands*

[b] *Marine & Transport Technology, Delft University of Technology*

[c] *Research Associate, FNRS; Institut Montefiore, Univ. Liège, Sart-Tilman, Bldg. B28, Parking P32, B-4000 Liège, Belgium*

Abstract

Dynamic programming (DP) is a powerful paradigm for general, nonlinear optimal control. Computing exact DP solutions is in general only possible when the process states and the control actions take values in a small discrete set. In practice, it is necessary to approximate the solutions. Therefore, we propose an algorithm for approximate DP that relies on a fuzzy partition of the state space, and on a discretization of the action space. This *fuzzy Q-iteration* algorithm works for deterministic processes, under the discounted return criterion. We prove that fuzzy Q-iteration asymptotically converges to a solution that lies within a bound of the optimal solution. A bound on the suboptimality of the solution obtained in a finite number of iterations is also derived. Under continuity assumptions on the dynamics and on the reward function, we show that fuzzy Q-iteration is consistent, i.e., that it asymptotically obtains the optimal solution as the approximation accuracy increases. These properties hold both when the parameters of the approximator are updated in a synchronous fashion, and when they are updated asynchronously. The asynchronous algorithm is proven to converge at least as fast as the synchronous one. The performance of fuzzy Q-iteration is illustrated in a two-link manipulator control problem.

Key words: approximate dynamic programming, fuzzy approximation, value iteration, convergence analysis.

1 Introduction

Dynamic programming (DP) is a powerful paradigm for solving optimal control problems, thanks to its mild assumptions on the controlled process, which can be nonlinear or stochastic [3, 4]. In the DP framework, a model of the process is assumed to be available, and the immediate performance is measured by a scalar reward signal. The controller then maximizes the long-term performance, measured by the cumulative reward. DP algorithms can be extended to work without requiring a model of the process, in which case they are usually called reinforcement learning (RL) algorithms [24]. Most DP and RL algorithms work by estimating an optimal value function, i.e., the maximal cumulative reward as a function of the process state and possibly also of the control action. Representing value functions exactly is only possible when the state-action space contains a relatively small number of discrete elements. In large discrete spaces and in continuous spaces, the value function generally has to be approximated. This is especially the case in automatic control, where the state and action variables are usually continuous.

Therefore, this paper proposes an algorithm for approximate DP that represents state-action value functions (called Q-functions in the sequel) using a fuzzy rule base with singleton consequents [18]. This algorithm works for deterministic problems, under the discounted return criterion. It is called *fuzzy Q-iteration*, because it combines the classical Q-iteration algorithm with a fuzzy approximator. The fuzzy rule base receives the state as input, and produces the Q-values of the discrete actions as outputs. The set of discrete actions is selected beforehand from the (possibly continuous) original action space. The membership functions of the fuzzy antecedents can also be seen as state-dependent basis functions or features [4].

We show that fuzzy Q-iteration asymptotically converges to an approximate Q-function that lies within a bounded distance from the optimal Q-function. The

[⋆] This paper was not presented at any IFAC meeting. Corresponding author Lucian Buşoniu (tel. +31 152788573, fax +31 152786679).

Email addresses: `i.l.busoniu@tudelft.nl` (Lucian Busoniu), `dernst@ulg.ac.be` (Damien Ernst), `b@deschutter.info` (Bart De Schutter), `r.babuska@tudelft.nl` (Robert Babuška).

suboptimality of the Q-function obtained after a finite number of iterations is also bounded. Both of these Q-functions lead to policies with a bounded suboptimality. We also show that fuzzy Q-iteration is consistent: under appropriate continuity assumptions on the process dynamics and on the reward function, the approximate Q-function converges to the optimal one as the approximation accuracy increases. These properties hold both when the parameters of the approximator are updated in a synchronous fashion, and when they are updated asynchronously. Additionally, the asynchronous algorithm is proven to converge at least as fast as the synchronous one. In a simulation example, fuzzy Q-iteration is used to control a two-link manipulator, and compared with the state-of-the-art fitted Q-iteration algorithm [11].

The remainder of this paper is structured as follows. Section 2 gives a brief overview of the literature related to our results. Section 3 describes Markov decision processes and the Q-iteration algorithm. Section 4 introduces fuzzy Q-iteration. This novel algorithm is analyzed in Section 5, and applied to a two-link manipulator example in Section 6. Section 7 concludes the paper and outlines some ideas for future work.

2 Related work

A rich body of literature concerns the analysis of approximate value iteration, both in the DP (model-based) setting [10, 14, 22, 23, 26] and in the RL (model-free) setting [1, 12, 25]. In many cases, convergence is ensured by using linearly parameterized approximators [14, 25, 26]. Our convergence analysis for synchronous fuzzy Q-iteration is related to the convergence analysis of approximate V-iteration for discrete state-action spaces in [14, 26]. We additionally consider continuous state-action spaces, introduce an explicit discretization procedure for the continuous actions, consider asynchronous fuzzy Q-iteration, and study the finite-time performance of the algorithm. While (exact) asynchronous value iteration is widely known [3, Sec. 1.3.2], asynchronous algorithms for approximate DP are not often studied. Many consistency results for model-based (DP) algorithms are found for discretization-based approximators [10, 23]. Such discretizations sometimes use interpolation schemes similar to fuzzy approximation. A different class of results analyzes the performance of approximate value iteration for stochastic processes, when only a limited number of samples are available [1, 12, 22].

Fuzzy approximators have typically been used in model-free (RL) techniques such as Q-learning [13, 15, 17] and actor-critic algorithms [2, 20]. Most of these approaches are heuristic in nature, and their theoretical properties have not been investigated yet. In this paper, we use fuzzy approximation with a *model-based* (DP) algorithm, and provide a detailed analysis of its convergence and consistency properties.

The present paper integrates and significantly extends the authors' earlier work on fuzzy Q-iteration [7, 8, 9], by removing some limiting assumptions: an originally discrete action space in [7, 9], and a restrictive bound on the Lipschitz constant of the process dynamics in [8]. Additionally, the solution obtained in a finite time is analyzed, and the suboptimality of the approximate solution is explicitly related to accuracy of the fuzzy approximator.

3 Markov decision processes and Q-iteration

This section introduces deterministic Markov decision processes (MDPs) and characterizes their optimal solution [3, 24]. Afterwards, exact and approximate Q-iteration are presented.

A deterministic MDP consists of the state space X, the action space U, the transition function $f : X \times U \to X$, and the reward function $\rho : X \times U \to \mathbb{R}$. As a result of the control action u_k applied in the state x_k, the state changes to $x_{k+1} = f(x_k, u_k)$ and a scalar reward $r_{k+1} = \rho(x_k, u_k)$ is generated, which evaluates the immediate effect of action u_k (the transition from x_k to x_{k+1}). The state and action spaces can be continuous or discrete. We assume that $\|\rho\|_\infty = \sup_{x,u} |\rho(x, u)|$ is finite. Actions are chosen according to the policy $h : X \to U$, which is a discrete-time state feedback $u_k = h(x_k)$.

The goal is to find an optimal policy, i.e., one that maximizes, starting from the current moment in time ($k = 0$) and from any initial state x_0, the discounted return:

$$R^h(x_0) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, h(x_k)) \qquad (1)$$

where the discount factor $\gamma \in [0, 1)$ and $x_{k+1} = f(x_k, h(x_k))$ for $k \geq 0$. Because the rewards are bounded, the infinite sum in (1) exists and is bounded. The task is therefore to maximize the long-term performance (return), while only using feedback about the immediate, one-step performance (reward).

Optimal policies can be conveniently characterized by the optimal Q-function $Q^* : X \times U \to \mathbb{R}$. For every pair (x, u), the optimal Q-function gives the discounted return obtained by first applying u in x, and then selecting actions optimally:

$$Q^*(x, u) = \rho(x, u) + \gamma \sup_h R^h(f(x, u)) \qquad (2)$$

where $x_1 = f(x, u)$ and $x_{k+1} = f(x_k, h(x_k))$ for $k \geq 1$. An optimal policy h^* can be found from Q^*, by ensuring that $h^*(x) \in \arg\max_u Q^*(x, u)$. Under mild technical assumptions, such a policy exists. In general, for a given Q, a policy h that satisfies $h(x) \in \arg\max_u Q(x, u)$ is called *greedy* in Q.

Let the set of all the Q-functions be denoted by $\mathbb{Q}$. Define the Q-iteration mapping $T : \mathbb{Q} \to \mathbb{Q}$:

$$[T(Q)](x,u) = \rho(x,u) + \gamma \sup_{u'} Q(f(x,u),u') \tag{3}$$

The optimal Q-function satisfies the Bellman optimality equation $Q^* = T(Q^*)$ [3, Sec. 6.4]. So, Q^* is a fixed point of T. The *Q-iteration* algorithm starts from an arbitrary Q-function Q_0 and in each iteration ℓ updates it by using:

$$Q_{\ell+1} = T(Q_\ell) \tag{4}$$

It is well-known that T is a contraction with factor $\gamma < 1$ in the infinity norm, i.e., for any pair of functions Q and Q', it is true that $\|T(Q)-T(Q')\|_\infty \leq \gamma\|Q-Q'\|_\infty$. This can be shown e.g., by extending the analogous result for V-functions given in [4, Sect. 2.3]. Therefore, Q^* is the *unique* fixed point of T, and Q-iteration converges to it as $\ell \to \infty$. Note that to implement (4), a model of the MDP is required, in the form of the transition and reward functions f, ρ.

In general, Q-iteration requires to store and update distinct Q-values for every state-action pair. This is only possible when the states and actions take values in a finite, discrete set. When the state or action variables are continuous, there are infinitely many state-action pairs, and the Q-function has to be represented approximately. Even when the number of state-actions pairs is finite but very large, exact Q-iteration might be impractical and approximation may be useful.

In this paper, we consider algorithms for *approximate Q-iteration* that parameterize the Q-function using a vector $\theta \in \mathbb{R}^n$. In addition to the Q-iteration mapping T (3), two other mappings are needed to formalize approximate Q-iteration. The *approximation* mapping $F : \mathbb{R}^n \to \mathbb{Q}$ produces an approximate Q-function $\widehat{Q} = F(\theta)$ from a parameter vector θ . So, θ is a finite representation of $\widehat{Q}$. The *projection* mapping $P : \mathbb{Q} \to \mathbb{R}^n$ computes a parameter vector θ such that $F(\theta)$ is as close as possible to a target Q-function Q (e.g., in a least-squares sense). We denote by $[F(\theta)](x,u)$ the value of the Q-function $F(\theta)$ for the state-action pair (x,u), and by $[P(Q)]_l$ the lth component in the parameter vector $P(Q)$.

Approximate Q-iteration starts with an arbitrary (e.g., identically 0) parameter vector θ_0 and at each iteration ℓ updates it using the composition of P, T, and F:[1]

$$\theta_{\ell+1} = P \circ T \circ F(\theta_\ell) \tag{5}$$

So, in principle, approximate Q-iteration first computes the approximate Q-function corresponding to the current parameter. It then updates this Q-function with the Q-iteration mapping, and finally computes an updated parameter vector that approximates the new Q-function. Of course, the results of F and T cannot be fully computed and stored. Instead, $P \circ T \circ F$ can be implemented, e.g., as a single entity, as will be the case for fuzzy Q-iteration. Once a satisfactory parameter vector $\theta^\dagger$ has been found, a policy $\widehat{h}^\dagger$ can be used, with:

$$\widehat{h}^\dagger(x) \in \arg\max_u [F(\theta^\dagger)](x,u) \tag{6}$$

assuming that the chosen approximator guarantees the existence of the maximum.

A similar derivation can be given for approximate V-iteration, which uses state-dependent V-functions instead of Q-functions. Approximate Q-iteration is not guaranteed to converge for arbitrary F and P. Counter-examples can be found for V-iteration e.g., in [26], but this problem appears in Q-iteration as well.

4 Fuzzy Q-iteration

In this section, the fuzzy Q-iteration algorithm is introduced. First, the fuzzy approximation and projection mappings are described, followed by synchronous and asynchronous fuzzy Q-iteration. The state space X and the action space U of the MDP may be either continuous or discrete, but they are assumed to be subsets of Euclidean spaces, such that the 2-norm of the states and actions is well-defined.

The proposed approximator relies on a fuzzy partition of the state space, and on a discretization of the action space. The fuzzy partition contains N fuzzy sets χ_i, each described by a membership function (MF) $\mu_i : X \to [0,1]$, $i = 1,\ldots,N$. A state x belongs to each set i with a degree of membership $\mu_i(x)$. The MFs can also be seen as state-dependent basis functions or features [4]. They do not necessarily have to be associated with meaningful linguistic labels. Nevertheless, if expert knowledge is available on the shape of the optimal Q-function, then MFs with meaningful linguistic labels can be defined. The following requirement is imposed on the MFs:

Requirement 1 *Each MF has its unique maximum at a single point, i.e., for every i there exists a unique x_i for which $\mu_i(x_i) > \mu_i(x)\ \forall x \neq x_i$. Additionally, $\mu_{i'}(x_i) = 0$ for all $i' \neq i$.*

Because all the other MFs take zero values in x_i, it can be assumed without loss of generality that $\mu_i(x_i) = 1$, which means that $\mu_i(x_i)$ is *normal*. The state x_i is then called the core of the ith set.

Example 1 (Triangular fuzzy partitions) A simple type of fuzzy partition that satisfies Requirement 1 can

[1] For simplicity, we write $P \circ T \circ F(\theta)$ instead of $(P \circ T \circ F)(\theta)$, and we use a similar notation for other composite mappings in the sequel.

be obtained as follows. For each state variable x_d, where $d = 1, \dots, D$ and $D = \dim(X)$, a number N_d of triangular MFs are defined as follows:

$$\begin{aligned}\phi_{d,1}(x_d) &= \max\left(0, \frac{c_{d,2} - x_d}{c_{d,2} - c_{d,1}}\right)\\ \phi_{d,i}(x_d) &= \max\left[0, \min\left(\frac{x_d - c_{d,i-1}}{c_{d,i} - c_{d,i-1}}, \frac{c_{d,i+1} - x_d}{c_{d,i+1} - c_{d,i}}\right)\right]\\ &\quad \text{for } i = 2, \dots, N_d - 1\\ \phi_{d,N_d}(x_d) &= \max\left(0, \frac{x_d - c_{d,N_d-1}}{c_{d,N_d} - c_{d,N_d-1}}\right)\end{aligned}$$

where $c_{d,1} < \cdots < c_{d,N_d}$ is the array of cores along dimension d, which fully determines the shape of the MFs, and $x_d \in [c_{d,1}, c_{d,N_d}]$. Adjacent functions always intersect at a 0.5 level. The product of each combination of (single-dimensional) MFs thus gives a pyramid-shaped D-dimensional MF in the fuzzy partition of X. □

Other types of MFs that satisfy Requirement 1 can be obtained, e.g., by using higher-order B-splines [5, Ch. 8] (triangular MFs are second-order B-splines), or Kuhn triangulations combined with barycentric interpolation [21]. Kuhn triangulations can have subexponential complexity in the number of state dimensions. In contrast, the number of MFs in triangular and B-spline partitions grows exponentially with the number of dimensions. Although our approach is not limited to triangular MFs, they will nevertheless be used in our examples, because they are the simplest MFs that satisfy the requirements for the convergence and consistency of fuzzy Q-iteration.

Requirement 1 can be relaxed using results in [26], so that other MFs can take non-zero values at the core x_i of a given MF i. If these values are small in a certain sense, fuzzy Q-iteration is still provably convergent, after some changes in the proofs. This relaxation allows other types of localized MFs such as Gaussian MFs. Note that we have indeed observed that fuzzy Q-iteration diverges when the other MFs take too large values at x_i.

Until now, approximation over the state space was discussed. To approximate over the (continuous or discrete) action space U, a discrete subset of actions U_d is chosen:

$$U_\mathrm{d} = \{u_j | u_j \in U, j = 1, \dots, M\} \tag{7}$$

The fuzzy approximator stores a parameter vector θ with $n = NM$ elements. Each pair (μ_i, u_j) of a MF and a discrete action is associated to one parameter $\theta_{[i,j]}$. The notation $[i, j]$ represents the scalar index corresponding to i and j, which can be computed as $[i, j] = i + (j - 1)N$. If the n elements of the vector θ were arranged into an $N \times M$ matrix, by first filling in the first column with the first N elements, then the second column with the subsequent N elements, etc., then the element at position $[i, j]$ of the vector would be placed at row i and column j of the matrix.

The Q-function is approximated using a multiple-input, multiple-output fuzzy rule base with singleton consequents. The rule base takes the (D-dimensional) state x as input, and produces M outputs $q_1, \dots, q_M$, which are the Q-values corresponding to each of the discrete actions $u_1, \dots, u_M$. The ith rule has the form:

$$R_i : \textbf{if } x \textbf{ is } \chi_i \textbf{ then } q_1 = \theta_{[i,1]}; q_2 = \theta_{[i,2]}; \dots ; q_M = \theta_{[i,M]} \tag{8}$$

To compute the Q-value of the pair (x, u), first the action u is discretized by selecting the discrete action $u_j \in U_\mathrm{d}$ that is closest to u. Then, the jth output of the rule base gives the approximate Q-value. This procedure is concisely written as the following *approximation mapping*:

$$\begin{aligned}\widehat{Q}(x, u) = [F(\theta)](x, u) = q_j = \sum_{i=1}^{N} \phi_i(x)\theta_{[i,j]}\\ \text{with } j \in \arg\min_{j'} \|u - u_{j'}\|_2\end{aligned} \tag{9}$$

where $\|\cdot\|_2$ is the Euclidean norm of the argument (here as well as in the sequel), and ϕ_i are the normalized MFs (degrees of fulfillment): [2]

$$\phi_i(x) = \frac{\mu_i(x)}{\sum_{i'=1}^{N} \mu_{i'}(x)}$$

Equation (9) describes a linearly parameterized approximator. To ensure that $F(\theta)$ is a well-defined function, any ties in the minimization of (9) have to be broken consistently. In the sequel we assume they are broken in favor of the smallest index that satisfies the condition. For a fixed x, such an approximator is constant over each subset of actions U_j given by:

$$u \in U_j \text{ if } \begin{cases} \|u - u_j\|_2 < \|u - u_{j'}\|_2 \text{ for all } j' \neq j \\ j < j' \text{ for any } j' \neq j \text{ such that} \\ \qquad \|u - u_j\|_2 = \|u - u_{j'}\|_2 \end{cases} \tag{10}$$

for $j = 1, \dots, M$, where the second condition is due to the way in which ties are broken. The sets U_j form a partition of U.

The *projection mapping* of fuzzy Q-iteration infers from a Q-function the approximator parameters according to:

$$\theta_{[i,j]} = [P(Q)]_{[i,j]} = Q(x_i, u_j) \tag{11}$$

[2] The MFs are already normal, see the discussion after Requirement 1. However, they may not yet be *normalized*; the sum of normalized MFs is 1 for any value of x.

Because $\phi_i(x_i) = 1$ and $\phi_{i'}(x_i) = 0$ for $i \neq i'$, the parameter vector θ in (11) is also the solution of the equation:

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\big([F(\theta)](x_i,u_j) - Q(x_i,u_j)\big)^2 = 0$$

The (synchronous) *fuzzy Q-iteration* is obtained by using the approximation mapping (9) and the projection mapping (11) in approximate Q-iteration (5). The algorithms starts with an arbitrary $\theta_0 \in \mathbb{R}^n$ and stops when $\|\theta_{\ell+1} - \theta_\ell\|_\infty \leq \varepsilon_{\text{QI}}$. Then, an approximately optimal parameter vector is $\widehat{\theta}^* = \theta_{\ell+1}$, and an approximately optimal policy can be computed with:

$$\hat{\hat{h}}^*(x) = u_{j^*}, \quad j^* \in \arg\max_j [F(\widehat{\theta}^*)](x,u_j) \tag{12}$$

This policy only chooses discrete actions, from the set U_d. However, because the Q-function $F(\widehat{\theta}^*)$ is constant in every region U_j (see (10)), the action u_{j^*} maximizes this Q-function over the entire action space. This means that $\hat{\hat{h}}^*$ is greedy in $F(\widehat{\theta}^*)$, and (12) is a special case of (6). The notation $\hat{\hat{h}}^*$ is used to differentiate from a policy $\widehat{h}^*$ that is greedy in $F(\theta^*)$, where θ^* is the parameter vector obtained asymptotically, as $\ell \to \infty$:

$$\widehat{h}^*(x) = u_{j^*}, \quad j^* \in \arg\max_j [F(\theta^*)](x,u_j) \tag{13}$$

It is also possible to obtain a continuous-action policy using the following heuristic. For any state, an action is computed by interpolating between the best local actions for every MF core, using the MFs as weights:

$$h(x) = \sum_{i=1}^{N}\phi_i(x)u_{j_i^*}, \;\; j_i^* \in \arg\max_j [F(\widehat{\theta}^*)](x_i,u_j) \tag{14}$$

The index j_i^* corresponds to a locally optimal action for the core x_i. For instance, when used with triangular MFs (cf. Example 1), the interpolation procedure (14) is well suited to problems where (near-)optimal policies are locally (piecewise) affine with respect to the state. Interpolated policies may, however, be a poor choice for other problems.

Because all the approximate Q-functions considered by fuzzy Q-iteration are constant inside every region U_j (see (10)), it suffices to consider only the discrete actions in U_d when computing the maximal Q-values in the Q-iteration mapping (3). This discrete-action version of the Q-iteration mapping is defined as follows:

$$[T_\text{d}(Q)](x,u) = \rho(x,u) + \gamma \max_j Q(f(x,u),u_j) \tag{15}$$

This property is useful in the practical implementation of fuzzy Q-iteration. For any θ, because $T \circ F(\theta) = T_\text{d} \circ F(\theta)$, we get $P \circ T \circ F(\theta) = P \circ T_\text{d} \circ F(\theta)$, and each iteration (5) can be implemented as:

$$\theta_{\ell+1} = P \circ T_\text{d} \circ F(\theta_\ell) \tag{16}$$

The maximization over U in the original T mapping is replaced with an easier maximization over the discrete set U_d in (7), which can be solved by enumeration. Furthermore, the norms in (9) do not need to be computed to implement (16).

Fuzzy Q-iteration using (16) can be written as Algorithm 1. To establish the equivalence between Algorithm 1 and the form (16), observe that the right-hand side in line 4 of Algorithm 1 corresponds to $[T_\text{d}(\widehat{Q}_\ell)](x_i,u_j)$, where $\widehat{Q}_\ell = F(\theta_\ell)$. Hence, line 4 can be written $\theta_{\ell+1,[i,j]} \leftarrow [P \circ T_\text{d} \circ F(\theta_\ell)]_{[i,j]}$ and the entire **for** loop described by lines 3–5 is equivalent to (16).

Algorithm 1 Synchronous fuzzy Q-iteration

```
1: $\theta_0 \leftarrow 0_{NM}$
2: repeat in every iteration $\ell = 0,1,2,\ldots$
3:     for $i = 1,\ldots,N, j = 1,\ldots,M$ do
4:         $\theta_{\ell+1,[i,j]} \leftarrow \rho(x_i,u_j) +$
               $\gamma \max_{j'} \sum_{i'=1}^{N} \phi_{i'}(f(x_i,u_j))\theta_{\ell,[i',j']}$
5:     end for
6: until $\|\theta_{\ell+1} - \theta_\ell\|_\infty \leq \varepsilon_{\text{QI}}$
Output: $\widehat{\theta}^* = \theta_{\ell+1}$
```

Algorithm 2 Asynchronous fuzzy Q-iteration

```
1: $\theta_0 \leftarrow 0_{NM}$
2: repeat in every iteration $\ell = 0,1,2,\ldots$
3:     $\theta \leftarrow \theta_\ell$
4:     for $i = 1,\ldots,N, j = 1,\ldots,M$ do
5:         $\theta_{[i,j]} \leftarrow \rho(x_i,u_j) +$
               $\gamma \max_{j'} \sum_{i'=1}^{N} \phi_{i'}(f(x_i,u_j))\theta_{[i',j']}$
6:     end for
7:     $\theta_{\ell+1} \leftarrow \theta$
8: until $\|\theta_{\ell+1} - \theta_\ell\|_\infty \leq \varepsilon_{\text{QI}}$
Output: $\widehat{\theta}^* = \theta_{\ell+1}$
```

Algorithm 1 computes the new parameters $\theta_{\ell+1}$ using the parameters θ_ℓ found at the previous iteration, which remain unchanged throughout the current iteration. Algorithm 2 is a different version of fuzzy Q-iteration that makes more efficient use of the updates: in each step of the computation, the latest updated values of the parameters are employed. Since the parameters are updated in an asynchronous fashion, this version is called *asynchronous fuzzy Q-iteration.* To differentiate between the two versions, Algorithm 1 is hereafter called *synchronous fuzzy Q-iteration.* In Algorithm 2 parameters are shown updated in sequence, but they can actually be updated in any order and our analysis still holds.

While in this paper fuzzy Q-iteration is described and analyzed for deterministic problems only, it can also be extended to stochastic problems. For instance, the asynchronous update in line 5 of Algorithm 2 becomes in the stochastic case:

$$\theta_{[i,j]} \leftarrow \mathrm{E}_{x' \sim \tilde{f}(x_i,u_j,\cdot)} \Big\{ \tilde{\rho}(x_i,u_j,x') + \gamma \max_{j'} \sum_{i'=1}^{N} \phi_{i'}(x')\theta_{[i',j']} \Big\}$$

where x' is sampled using the probability density function $\tilde{f}(x_i,u_j,\cdot)$ of the next state x', given x_i and u_j, and the dot stands for x'. In general, the expectation in this update cannot be computed exactly, but has to be estimated from a finite number of samples. In this case, our analysis does not apply. Approaches to analyze the effect of the finite-sampling errors have been proposed in [1, 12] for variants of approximate Q-iteration, and in [22] for approximate V-iteration. In some limited special cases, e.g., when there is a finite number of possible successor states, the expectation can be computed exactly, and then our results apply.

5 Analysis of fuzzy Q-iteration

In Section 5.1, we show that synchronous and asynchronous fuzzy Q-iteration are convergent and we characterize the suboptimality of their solution. In Section 5.2, we show that fuzzy Q-iteration is consistent, i.e., that its solution asymptotically converges to Q^* as the approximation accuracy increases. These results show that fuzzy Q-iteration is a theoretically sound algorithm. Section 5.3 examines the computational complexity of fuzzy Q-iteration.

5.1 Convergence

First, it is shown that there exists a parameter vector θ^* such that, for both synchronous and asynchronous fuzzy Q-iteration, $\theta_\ell \to \theta^*$ as $\ell \to \infty$. The parameter vector θ^* yields an approximate Q-function that is within a bound of the optimal Q-function, and a policy with a bounded suboptimality. Similar bounds are derived for the solution $\widehat{\theta}^*$ obtained in a finite number of iterations, by using a convergence threshold $\varepsilon_{\text{QI}} > 0$. Additionally, asynchronous fuzzy Q-iteration is shown to converge at least as fast as the synchronous version.

Theorem 1 (Convergence of synchronous fuzzy Q-iteration) *Synchronous fuzzy Q-iteration (Algorithm 1) converges.*

Proof: We show that $P \circ T \circ F$ is a contraction in the infinity norm with factor $\gamma < 1$, i.e., $\|P \circ T \circ F(\theta) - P \circ T \circ F(\theta')\|_\infty \le \gamma \|\theta - \theta'\|_\infty$, for any θ, θ'. Recall that T is a contraction with factor $\gamma < 1$. It is obvious from (11) that P is a non-expansion. Also, F is a non-expansion:

$$\begin{aligned} &|[F(\theta)](x,u) - [F(\theta')](x,u)| \\ &\quad= \left| \sum_{i=1}^{N} \phi_i(x)\theta_{[i,j]} - \sum_{i=1}^{N} \phi_i(x)\theta'_{[i,j]} \right| \\ &\qquad\qquad (\text{where } j \in \arg\min_{j'} \|u - u_{j'}\|_2) \\ &\quad\le \sum_{i=1}^{N} \phi_i(x) \left| \theta_{[i,j]} - \theta'_{[i,j]} \right| \\ &\quad\le \sum_{i=1}^{N} \phi_i(x) \|\theta - \theta'\|_\infty = \|\theta - \theta'\|_\infty \end{aligned}$$

Therefore, $P \circ T \circ F$ is a contraction with factor $\gamma < 1$. It follows that $P \circ T \circ F$ has a unique fixed point θ^*, and synchronous fuzzy Q-iteration converges to this fixed point as $\ell \to \infty$. □

This proof is similar to the proof of convergence for V-iteration with averagers [14] or with interpolative representations [26]. The fuzzy approximator can be seen as an extension of an averager or of an interpolative representation for approximating Q-functions, additionally using discretization to deal with large or continuous action spaces.

In the sequel, a concise notation of asynchronous fuzzy Q-iteration is needed. Recall that $n = NM$, and that $[i,j] = i + (j-1)N$. Define for all $l = 0, \dots, n$ recursively the mappings $S_l : \mathbb{R}^n \to \mathbb{R}^n$ as:

$$S_0(\theta) = \theta$$
$$[S_l(\theta)]_{l'} = \begin{cases} [P \circ T \circ F(S_{l-1}(\theta))]_{l'} & \text{if } l' = l \\ [S_{l-1}(\theta)]_{l'} & \text{if } l' \in \{1,\dots,n\} \setminus \{l\} \end{cases}$$

So, S_l for $l > 0$ corresponds to updating the first l parameters using approximate asynchronous Q-iteration, and S_n is a complete iteration of the algorithm.

Theorem 2 (Convergence of asynchronous fuzzy Q-iteration) *Asynchronous fuzzy Q-iteration (Algorithm 2) converges.*

Proof: We show that S_n is a contraction, i.e., $\|S_n(\theta) - S_n(\theta')\|_\infty \le \gamma \|\theta - \theta'\|_\infty$, for any θ, θ'. This can be done element by element. By the definition of S_l, the first element is only updated by S_1:

$$\begin{aligned} &|[S_n(\theta)]_1 - [S_n(\theta')]_1| = |[S_1(\theta)]_1 - [S_1(\theta')]_1| \\ &\quad= |[P \circ T \circ F(\theta)]_1 - [P \circ T \circ F(\theta')]_1| \le \gamma \|\theta - \theta'\|_\infty \end{aligned}$$

The last step is true because $P \circ T \circ F$ is a contraction. Similarly, the second element is only updated by S_2:

$$\begin{aligned}
|[S_n(\theta)]_2 - [S_n(\theta')]_2| &= |[S_2(\theta)]_2 - [S_2(\theta')]_2| \\
&= |[P \circ T \circ F(S_1(\theta))]_2 - [P \circ T \circ F(S_1(\theta'))]_2| \\
&\leq \gamma \|S_1(\theta) - S_1(\theta')\|_\infty \\
&= \gamma \max\{|[P \circ T \circ F(\theta)]_1 - [P \circ T \circ F(\theta')]_1|, \\
&\qquad |\theta_2 - \theta_2'|, \ldots, |\theta_n - \theta_n'|\} \\
&\leq \gamma \|\theta - \theta'\|_\infty
\end{aligned}$$

where the contraction property of $P \circ T \circ F$ is used twice. Continuing in this fashion, we obtain $|[S_n(\theta)]_l - [S_n(\theta')]_l| \leq \gamma \|\theta - \theta'\|_\infty$ for all l, and thus S_n is a contraction. Therefore, asynchronous fuzzy Q-iteration converges. This proof is similar to that for *exact* asynchronous V-iteration [3, Sec. 1.3.2]. $\square$

It can also be easily shown that synchronous and asynchronous fuzzy Q-iteration converge to the same parameter vector θ^*. We show next that asynchronous fuzzy Q-iteration converges at least as fast as the synchronous version. For that, we first need the following monotonicity lemma. In the sequel, vector and vector function inequalities are understood to be satisfied element-wise.

Lemma 1 (Monotonicity) *If $\theta \leq \theta'$, then $P \circ T \circ F(\theta) \leq P \circ T \circ F(\theta')$.*

Proof: It is well-known that T is monotonous, i.e., $T(Q) \leq T(Q')$ if $Q \leq Q'$. This can be shown e.g., by extending the analogous result for V-functions given in [4, Sec. 2.3]. We will show that F and P are monotonous. Given $\theta \leq \theta'$, it follows that $\sum_{i=1}^{N} \phi_i(x)\theta_{[i,j]} \leq \sum_{i=1}^{N} \phi_i(x)\theta'_{[i,j]}$, where $j \in \arg\min_{j'} \|u - u_{j'}\|_2$. This is equivalent to $[F(\theta)](x,u) \leq [F(\theta')](x,u)$, so F is monotonous. Given $Q \leq Q'$, it follows that for all for all i,j, $Q(x_i,u_j) \leq Q'(x_i,u_j)$. This is equivalent to $[P(Q)]_{[i,j]} \leq [P(Q')]_{[i,j]}$, so P is monotonous. Because P, T, and F are monotonous, so is $P \circ T \circ F$. $\square$

Asynchronous fuzzy Q-iteration converges at least as fast as the synchronous algorithm, in the sense that ℓ iterations of the asynchronous algorithm take the parameter vector at least as close to θ^* as ℓ iterations of the synchronous algorithm. This is stated formally as follows.

Proposition 1 (Convergence rate) *If a parameter vector θ satisfies $\theta \leq P \circ T \circ F(\theta) \leq \theta^*$, then:*

$$(P \circ T \circ F)^\ell(\theta) \leq S_n^\ell(\theta) \leq \theta^* \quad \forall \ell \geq 1$$

where $S_n^\ell(\theta)$ denotes the composition of $S_n(\theta)$ ℓ-times with itself, i.e., $S_n^\ell(\theta) = S_n \circ S_n \circ \cdots \circ S_n(\theta)$, and similarly for $(P \circ T \circ F)^\ell(\theta)$.

Proof: This follows from the monotonicity of $P \circ T \circ F$, and can be shown element-wise, in a similar fashion to the proof of Theorem 2. This result is an extension of the similar result on exact V-iteration in [3, Sec. 1.3.2]. $\square$

In the remainder of this section, in addition to examining the asymptotic properties of fuzzy Q-iteration, we also consider an implementation that stops when $\|\theta_{\ell+1} - \theta_\ell\|_\infty \leq \varepsilon_{\text{QI}}$, with a convergence threshold $\varepsilon_{\text{QI}} > 0$ (see Algorithms 1 and 2). This implementation returns the solution $\widehat{\theta}^* = \theta_{\ell+1}$.

Proposition 2 (Finite-time termination) *For any choice of the threshold $\varepsilon_{\text{QI}} > 0$ and any initial parameter vector $\theta_0 \in \mathbb{R}^n$, synchronous and asynchronous fuzzy Q-iteration stop in finite time.*

Proof: Consider synchronous fuzzy Q-iteration. Because the mapping $P \circ T \circ F$ is a contraction with factor $\gamma < 1$ and fixed point θ^*, we have: $\|\theta_{\ell+1} - \theta^*\|_\infty = \|P \circ T \circ F(\theta_\ell) - P \circ T \circ F(\theta^*)\|_\infty \leq \gamma \|\theta_\ell - \theta^*\|_\infty$. By induction, $\|\theta_\ell - \theta^*\|_\infty \leq \gamma^\ell \|\theta_0 - \theta^*\|_\infty$ for any $\ell > 0$. By the Banach fixed point theorem [16, Ch. 3], θ^* is bounded. Because the initial parameter vector θ_0 is also bounded, $\|\theta_0 - \theta^*\|_\infty$ is bounded. Using the notation $B_0 = \|\theta_0 - \theta^*\|_\infty$, it follows that B_0 is bounded and that $\|\theta_\ell - \theta^*\|_\infty \leq \gamma^\ell B_0$ for any $\ell > 0$. Therefore, we have: $\|\theta_{\ell+1} - \theta_\ell\|_\infty \leq \|\theta_{\ell+1} - \theta^*\|_\infty + \|\theta_\ell - \theta^*\|_\infty \leq \gamma^\ell(\gamma+1)B_0$. Using this inequality, we can choose for any $\varepsilon_{\text{QI}} > 0$ a number of iterations $L = \left\lceil \log_\gamma \frac{\varepsilon_{\text{QI}}}{(\gamma+1)B_0} \right\rceil$ that guarantees $\|\theta_{L+1} - \theta_L\|_\infty \leq \varepsilon_{\text{QI}}$. Therefore, the algorithm stops in at most L iterations. Here, $\lceil \cdot \rceil$ gives the smallest integer larger than or equal to the argument (ceiling). Because B_0 is bounded, L is finite. Since the computational cost of every single iteration is finite for finite N and M, synchronous fuzzy Q-iteration stops in finite time.

The proof for asynchronous fuzzy Q-iteration proceeds in the same way, because the asynchronous Q-iteration mapping S_n is also a contraction with factor $\gamma < 1$ and fixed point θ^*. $\square$

The following bounds on the suboptimality of the resulting approximate Q-function and policy hold.

Theorem 3 (Near-optimality) *Denote the set of Q-functions representable using the fuzzy approximator F by $\widehat{\mathbb{Q}} = \{F(\theta) \mid \theta \in \mathbb{R}^n\}$, where $\widehat{\mathbb{Q}} \subset \mathbb{Q}$, and take ε'_Q so that $\exists \widehat{Q} \in \widehat{\mathbb{Q}}$ with $\|Q^* - \widehat{Q}\|_\infty \leq \varepsilon'_Q$. The convergence point θ^* of asynchronous and synchronous fuzzy Q-iteration satisfies:*

$$\|Q^* - F(\theta^*)\|_\infty \leq \frac{2\varepsilon'_Q}{1-\gamma} \tag{17}$$

Additionally, the parameter vector $\widehat{\theta}^*$ obtained by asynchronous or synchronous fuzzy Q-iteration in a finite *time, by using a threshold ε_{QI}, satisfies:*

$$\|Q^* - F(\widehat{\theta}^*)\|_\infty \leq \frac{2\varepsilon'_Q + \gamma\varepsilon_{\text{QI}}}{1-\gamma} \tag{18}$$

Furthermore:

$$\|Q^* - Q^{\widehat{h}^*}\|_\infty \leq \frac{4\gamma\varepsilon'_Q}{(1-\gamma)^2} \tag{19}$$

$$\|Q^* - Q^{\hat{\widehat{h}}^*}\|_\infty \leq \frac{2\gamma(2\varepsilon'_Q + \gamma\varepsilon_{\text{QI}})}{(1-\gamma)^2} \tag{20}$$

where $Q^{\widehat{h}^*}$ is the Q-function of the policy $\widehat{h}^*$ that is greedy in $F(\theta^*)$ (13), and $Q^{\hat{\widehat{h}}^*}$ is the Q-function of the policy $\hat{\widehat{h}}^*$ that is greedy in $F(\widehat{\theta}^*)$ (12).

Proof: The proof proceeds in the same way for synchronous and asynchronous fuzzy Q-iteration, because it only relies on the contracting nature of their updates. After observing that $\widehat{\mathbb{Q}}$ is identical to the set of fixed points of the composite mapping $F \circ P$, the bound (17) can be proven in a similar way to the bound shown in [14, 26] for V-iteration. In order to obtain (18), a bound on $\|\widehat{\theta}^* - \theta^*\|_\infty$ is computed first. Let L be the number of iterations after which the algorithm stops, which is finite by Proposition 2. Therefore, $\widehat{\theta}^* = \theta_{L+1}$. We have:

$$\begin{aligned}\|\theta_L - \theta^*\|_\infty &\leq \|\theta_{L+1} - \theta_L\|_\infty + \|\theta_{L+1} - \theta^*\|_\infty \\ &\leq \varepsilon_{\text{QI}} + \gamma\|\theta_L - \theta^*\|_\infty\end{aligned}$$

where the last step follows from the convergence condition $\|\theta_{L+1} - \theta_L\|_\infty \leq \varepsilon_{\text{QI}}$ and from the contracting nature of the updates (see also the proof of Proposition 2). From the last inequality, it follows that $\|\theta_L - \theta^*\|_\infty \leq \frac{\varepsilon_{\text{QI}}}{1-\gamma}$ and therefore that $\|\theta_{L+1} - \theta^*\|_\infty \leq \gamma\|\theta_L - \theta^*\|_\infty \leq \frac{\gamma\varepsilon_{\text{QI}}}{1-\gamma}$, which is equivalent to:

$$\|\widehat{\theta}^* - \theta^*\|_\infty \leq \frac{\gamma\varepsilon_{\text{QI}}}{1-\gamma} \tag{21}$$

Using this inequality, the suboptimality of the Q-function $F(\widehat{\theta}^*)$ can be bounded with:

$$\begin{aligned}\|Q^* - F(\widehat{\theta}^*)\|_\infty &\leq \|Q^* - F(\theta^*)\|_\infty + \|F(\theta^*) - F(\widehat{\theta}^*)\|_\infty \\ &\leq \|Q^* - F(\theta^*)\|_\infty + \|\widehat{\theta}^* - \theta^*\|_\infty \\ &\leq \frac{2\varepsilon'_Q}{1-\gamma} + \frac{\gamma\varepsilon_{\text{QI}}}{1-\gamma} = \frac{2\varepsilon'_Q + \gamma\varepsilon_{\text{QI}}}{1-\gamma}\end{aligned}$$

where the second step is true because F is a non-expansion (which was shown in the proof of Theorem 1), and the third step follows from (17) and (21). So, (18) has been obtained.

The bounds (19) and (20), which characterize the suboptimality of the policies resulting from θ^* and $\widehat{\theta}^*$, follow from the following general inequality between the suboptimality of an arbitrary Q-function Q, and the suboptimality of the policy h that is greedy in this Q-function:

$$\|Q^* - Q^h\|_\infty \leq \frac{2\gamma}{(1-\gamma)}\|Q^* - Q\|_\infty$$

To obtain (19) and (20), this inequality is applied to the Q-functions $F(\theta^*)$ and $F(\widehat{\theta}^*)$, using their suboptimality bounds (17) and (18). □

Examining (18) and (20), it can be seen that the suboptimality of the finite-time solution is given by a sum of two terms. The second term depends linearly on the precision ε_{QI} with which the solution is computed, and is easy to control by setting ε_{QI} as close to 0 as needed. The first term in the sum depends linearly on ε'_Q, which is related to the accuracy of the fuzzy approximator, and is more difficult to control. This ε'_Q-dependent term also contributes to the suboptimality of the asymptotic solutions (17),(19). Ideally, one can find $\varepsilon'_Q = \min_{\widehat{Q} \in \widehat{\mathbb{Q}}} \|Q^* - \widehat{Q}\|_\infty$, which provides the smallest upper bounds in (17)–(20). Moreover, if the optimal Q-function Q^* is exactly representable by the chosen fuzzy approximator, i.e., $Q^* \in \widehat{\mathbb{Q}}$, one can take $\varepsilon'_Q = 0$, and fuzzy Q-iteration asymptotically converges to Q^*. Section 5.2 provides additional insight into the relationship between the suboptimality of the solution and the accuracy of the approximator.

5.2 Consistency

Next, we analyze the consistency of synchronous and asynchronous fuzzy Q-iteration. It is shown that the approximate solution $F(\theta^*)$ asymptotically converges to the optimal Q-function Q^*, as the largest distance between the cores of adjacent fuzzy sets and the largest distance between adjacent discrete actions decrease to 0. An explicit relationship between the suboptimality of $F(\theta^*)$ and the accuracy of the approximator is derived.

The state resolution step δ_x is defined as the largest distance between any point in the state space and the MF core that is closest to it. The action resolution step δ_u is defined similarly for the discrete actions. Formally:

$$\delta_x = \sup_{x \in X} \min_{i=1,\ldots,N} \|x - x_i\|_2 \tag{22}$$

$$\delta_u = \sup_{u \in U} \min_{j=1,\ldots,M} \|u - u_j\|_2 \tag{23}$$

where x_i is the core of the ith MF, and u_j is the jth discrete action. Smaller values of δ_x and δ_u indicate a higher resolution. The goal is to show that $\lim_{\delta_x \to 0,\ \delta_u \to 0} F(\theta^*) = Q^*$.

We assume that f and ρ are Lipschitz continuous, and require that the MFs are Lipschitz continuous as well as local and evenly distributed, as formalized next.

Assumption 1 (Lipschitz continuity) *The dynamics f and the reward function ρ are Lipschitz continuous, i.e., there exist finite constants $L_f \geq 0$, $L_\rho \geq 0$ so that:*

$$\begin{aligned}\|f(x,u) - f(\bar{x},\bar{u})\|_2 &\leq L_f(\|x-\bar{x}\|_2 + \|u-\bar{u}\|_2)\\ |\rho(x,u) - \rho(\bar{x},\bar{u})| &\leq L_\rho(\|x-\bar{x}\|_2 + \|u-\bar{u}\|_2)\\ &\forall x,\bar{x}\in X, u,\bar{u}\in U\end{aligned}$$

Requirement 2 *Every MF ϕ_i is Lipschitz continuous, i.e., for every i there exists a finite constant $L_{\phi_i} \geq 0$ so that:*

$$\|\phi_i(x) - \phi_i(\bar{x})\|_2 \leq L_{\phi_i}\|x-\bar{x}\|_2, \quad \forall x,\bar{x}\in X$$

Requirement 3 *Every MF ϕ_i has a bounded support, which is contained in a ball with a radius proportional to δ_x. Formally, there exists a finite $\nu > 0$ so that:*

$$\{x \mid \phi_i(x) > 0\} \subset \{x \mid \|x - x_i\|_2 \leq \nu\delta_x\}, \quad \forall i$$

Furthermore, for every x, only a finite number of MFs are non-zero. Formally, there exists a finite $\kappa > 0$ so that:

$$|\{i \mid \phi_i(x) > 0\}| \leq \kappa, \quad \forall x$$

where $|\cdot|$ denotes set cardinality.

Lipschitz continuity conditions such as those of Assumption 1 are typically needed to prove the consistency of algorithms for approximate DP. Requirement 2 is not restrictive. Requirement 3 is satisfied in many cases of interest. For instance, it is satisfied by convex fuzzy sets with their cores distributed on an (equidistant or irregular) rectangular grid in the state space, such as the triangular partitions of Example 1. In such cases, every point x falls inside a hyperbox defined by the two adjacent cores that are closest to x_d on each axis d. Some points will fall on the boundary of several hyperboxes, in which case we can just pick any of these hyperboxes. Given Requirement 1 and because the fuzzy sets are convex, only the MFs with the cores in the corners of the hyperbox can take non-zero values in the chosen point. The number of corners is 2^D where D is the dimension of X. Therefore, $|\{i \mid \phi_i(x) > 0\}| \leq 2^D$, and a choice $\kappa = 2^D$ satisfies the second part of Requirement 3. Furthermore, a MF will always extend at most two hyperboxes along each axis of the state space. By the definition of δ_x, the largest diagonal of any hyperbox is $2\delta_x$. Therefore, $\{x \mid \phi_i(x) > 0\} \subset \{x \mid \|x - x_i\|_2 \leq 4\delta_x\}$, and a choice $\nu = 4$ satisfies the first part of Requirement 3.

The next lemma bounds the approximation error introduced by every iteration of the synchronous algorithm. Since we are ultimately interested in characterizing the convergence point θ^*, which is the same for both algorithms, the final consistency result (Theorem 4) applies to the asynchronous algorithm as well.

Lemma 2 (Bounded error) *There exists a constant $\varepsilon_\delta \geq 0$, $\varepsilon_\delta = \mathrm{O}(\delta_x) + \mathrm{O}(\delta_u)$, so that any approximate Q-function $\widehat{Q}$ considered by synchronous fuzzy Q-iteration satisfies $\|F\circ P\circ T(\widehat{Q}) - T(\widehat{Q})\|_\infty \leq \varepsilon_\delta$.*

Proof: For any pair (x,u):

$$\begin{aligned}&\left|[F\circ P\circ T(\widehat{Q})](x,u) - [T(\widehat{Q})](x,u)\right|\\ &= \left|\left(\sum_{i=1}^N \phi_i(x)[T(\widehat{Q})](x_i,u_j)\right) - [T(\widehat{Q})](x,u)\right|\\ &\qquad \text{(where } j \in \arg\min_{j'}\|u-u_{j'}\|_2\text{)}\\ &= \Bigg|\left(\sum_{i=1}^N \phi_i(x)\left[\rho(x_i,u_j) + \gamma\max_{u'}\widehat{Q}(f(x_i,u_j),u')\right]\right)\\ &\qquad - \left[\rho(x,u) + \gamma\max_{u'}\widehat{Q}(f(x,u),u')\right]\Bigg|\\ &\leq \left|\left(\sum_{i=1}^N \phi_i(x)\rho(x_i,u_j)\right) - \rho(x,u)\right|\\ &+ \gamma\left|\left(\sum_{i=1}^N \phi_i(x)\max_{u'}\widehat{Q}(f(x_i,u_j),u')\right) - \max_{u'}\widehat{Q}(f(x,u),u')\right|\end{aligned} \tag{24}$$

Note that $\max_u \widehat{Q}(x,u)$ exists, because $\widehat{Q}(x,u)$ can take at most M distinct values for any fixed x. The first term on the right-hand side of (24) is:

$$\begin{aligned}&\left|\sum_{i=1}^N \phi_i(x)\left[\rho(x_i,u_j) - \rho(x,u)\right]\right|\\ &\quad\leq \sum_{i=1}^N \phi_i(x)L_\rho(\|x_i - x\|_2 + \|u_j - u\|_2)\\ &\quad= L_\rho\left[\|u_j - u\|_2 + \sum_{i=1}^N \phi_i(x)\|x_i - x\|_2\right]\\ &\quad\leq L_\rho(\delta_u + \kappa\nu\delta_x)\end{aligned} \tag{25}$$

where the Lipschitz continuity of ρ was used, and the last step follows from the definition of δ_u and Requirement 3. The second term in the right-hand side of (24) is:

$$\begin{aligned}&\gamma\left|\sum_{i=1}^N \phi_i(x)\left[\max_{u'}\widehat{Q}(f(x_i,u_j),u') - \max_{u'}\widehat{Q}(f(x,u),u')\right]\right|\\ &\leq \gamma\sum_{i=1}^N \phi_i(x)\left|\max_{j'}\widehat{Q}(f(x_i,u_j),u_{j'}) - \max_{j'}\widehat{Q}(f(x,u),u_{j'})\right|\\ &\leq \gamma\sum_{i=1}^N \phi_i(x)\max_{j'}\left|\widehat{Q}(f(x_i,u_j),u_{j'}) - \widehat{Q}(f(x,u),u_{j'})\right|\end{aligned} \tag{26}$$

The first step is true because $\widehat{Q}$ is of the form $F(\theta)$ given in (9) for some θ, and is therefore constant in each set U_j, for $j = 1, \dots, M$. The second step is true because the difference between the maxima of two functions of the same variable is at most the maximum of the difference of the functions. Writing $\widehat{Q}$ explicitly as in (9), we have:

$$\begin{aligned}
&\left|\widehat{Q}(f(x_i, u_j), u_{j'}) - \widehat{Q}(f(x, u), u_{j'})\right| \\
&= \left|\sum_{i'=1}^{N} \left[\phi_{i'}(f(x_i, u_j))\theta_{[i',j']} - \phi_{i'}(f(x, u))\theta_{[i',j']}\right]\right| \\
&\leq \sum_{i'=1}^{N} |\phi_{i'}(f(x_i, u_j)) - \phi_{i'}(f(x, u))|\,|\theta_{[i',j']}| \qquad (27)
\end{aligned}$$

Define $I' = \{i' \mid \phi_{i'}(f(x_i, u_j)) \neq 0 \text{ or } \phi_{i'}(f(x, u)) \neq 0\}$. Using Requirement 3, $|I'| \leq 2\kappa$. Denote $L_\phi = \max_i L_{\phi_i}$. Then, the right-hand side of (27) is equal to:

$$\begin{aligned}
&\sum_{i' \in I'} |\phi_{i'}(f(x_i, u_j)) - \phi_{i'}(f(x, u))|\,|\theta_{[i',j']}| \\
&\leq \sum_{i' \in I'} L_\phi L_f (\|x_i - x\|_2 + \|u_j - u\|_2)\|\theta\|_\infty \\
&\leq 2\kappa L_\phi L_f (\|x_i - x\|_2 + \|u_j - u\|_2)\|\theta\|_\infty \qquad (28)
\end{aligned}$$

Using (27) and (28) in (26) yields:

$$\begin{aligned}
&\gamma \sum_{i=1}^{N} \phi_i(x) \max_{j'} \left|\widehat{Q}(f(x_i, u_j), u_{j'}) - \widehat{Q}(f(x, u), u_{j'})\right| \\
&\leq \gamma \sum_{i=1}^{N} \phi_i(x) \max_{j'} 2\kappa L_\phi L_f (\|x_i - x\|_2 + \|u_j - u\|_2)\|\theta\|_\infty \\
&\leq 2\gamma\kappa L_\phi L_f \|\theta\|_\infty \left[\|u_j - u\|_2 + \sum_{i=1}^{N} \phi_i(x)\|x_i - x\|_2\right] \\
&\leq 2\gamma\kappa L_\phi L_f \|\theta\|_\infty (\delta_u + \kappa\nu\delta_x) \qquad (29)
\end{aligned}$$

where the last step follows from the definition of δ_u and Requirement 3. Finally, substituting (25) and (29) into (24) yields:

$$\begin{aligned}
&\left|[F \circ P \circ T(\widehat{Q})](x, u) - [T(\widehat{Q})](x, u)\right| \\
&\qquad \leq (L_\rho + 2\gamma\kappa L_\phi L_f \|\theta\|_\infty)(\delta_u + \kappa\nu\delta_x)
\end{aligned}$$

Given a bounded initial parameter vector θ_0, all the parameter vectors considered by the algorithm are bounded. This can be shown as follows. By the Banach fixed point theorem [16, Ch. 3], the optimal parameter vector θ^* (the unique fixed point of $P \circ T \circ F$) is finite. Also, we have $\|\theta_\ell - \theta^*\|_\infty \leq \gamma^\ell \|\theta_0 - \theta^*\|_\infty$ (see the proof of Proposition 2). Since $\|\theta_0 - \theta^*\|_\infty$ is bounded, all the other distances are bounded, and all the parameter vectors θ_ℓ are bounded. Let $B_\theta = \max_{\ell \geq 0} \|\theta_\ell\|_\infty$, which is bounded. Therefore, $\|\theta\|_\infty \leq B_\theta$ in (5.2), and the proof is complete with $\varepsilon_\delta = (L_\rho + 2\gamma\kappa L_\phi L_f B_\theta)(\delta_u + \kappa\nu\delta_x) = \mathrm{O}(\delta_x) + \mathrm{O}(\delta_u)$. □

Theorem 4 (Consistency) *Under Assumption 1 and if Requirements 2 and 3 are satisfied, synchronous and asynchronous fuzzy Q-iteration are consistent, i.e.,* $\lim_{\delta_x \to 0, \delta_u \to 0} F(\theta^*) = Q^*$, *and the suboptimality of the approximate Q-function satisfies* $\|F(\theta^*) - Q^*\|_\infty = \mathrm{O}(\delta_x) + \mathrm{O}(\delta_u)$.

Proof: It is easy to show by induction that $\|F(\theta^*) - Q^*\|_\infty \leq \varepsilon_\delta/(1 - \gamma)$ [4, Sec. 6.5.3]. From Lemma 2, $\lim_{\delta_x \to 0, \delta_u \to 0} \varepsilon_\delta = \lim_{\delta_x \to 0, \delta_u \to 0} (L_\rho + 2\gamma\kappa L_\phi L_f B_\theta)(\delta_u + \kappa\nu\delta_x) = 0$, and the first result is proven. Furthermore, using the same lemma, $\varepsilon_\delta = \mathrm{O}(\delta_x) + \mathrm{O}(\delta_u)$. □

In addition to guaranteeing consistency, Theorem 4 also relates the suboptimality of the Q-function $F(\theta^*)$ with the accuracy of the fuzzy approximator. Using Theorem 3, the accuracy can be further related with the suboptimality of the policy $\widehat{h}^*$ greedy in $F(\theta^*)$, and with the suboptimality of the solution obtained after a finite number of iterations (the Q-function $F(\widehat{\theta}^*)$ and the corresponding greedy policy $\hat{\widehat{h}}^*$).

5.3 Computational complexity

In this section, the time and memory complexity of fuzzy Q-iteration are analyzed. It is easy to see that each iteration of the synchronous and asynchronous fuzzy Q-iteration (Algorithms 1 and 2) requires $\mathrm{O}(N^2 M)$ time to run. Here, N is the number of MFs and M the number of discrete actions, leading to a parameter vector of length NM. The complete algorithms consist of L iterations and require $\mathrm{O}(LN^2 M)$ computation. Fuzzy Q-iteration requires $\mathrm{O}(NM)$ memory. The memory complexity is not proportional to L because, in practice, any $\theta_{\ell'}$ for which $\ell' < \ell$ can be discarded.

Next, we compare the complexity of fuzzy Q-iteration with the complexity of two representative RL algorithms, namely least-squares policy iteration (LSPI) [19] and fitted Q-iteration [11]. We focus on the case in which all algorithms compute *the same number of parameters*. LSPI approximates in each iteration the Q-function of the current policy (instead of the optimal Q-function), and then uses it to find an improved policy. Typically, the approximators employed in LSPI are structured similarly to our fuzzy approximator: they consist of N state-dependent basis functions (BFs) and M discrete actions, and have NM parameters. The time complexity of each iteration of the LSPI algorithm is $\mathrm{O}(N^3 M^3)$, due to solving a linear system of size NM. By solving the system incrementally, the complexity can be reduced, but will still be at least $\mathrm{O}(N^2 M^2)$. The memory complexity is $\mathrm{O}(N^2 M^2)$. So, LSPI requires more computation per iteration and more memory than fuzzy

Q-iteration. However, policy iteration methods such as LSPI often converge in a small number of iterations [24], usually smaller than the number of iterations required by value iteration methods such as fuzzy Q-iteration.

Fitted Q-iteration is a model-free variant of approximate Q-iteration. In every iteration, it solves a regression problem to find the updated Q-function. Many regression techniques can be employed, each incurring a different computational cost. Consider e.g., the kernel-based regression of regularized fitted Q-iteration [12]. For this algorithm, the number of parameters computed in an iteration is identical to the number of samples N_s employed in that iteration. The computational complexity of this iteration $\mathrm{O}({N_\mathrm{s}}^3)$, and its memory complexity is $\mathrm{O}({N_\mathrm{s}}^2)$. So, compared to fuzzy Q-iteration with $NM = N_\mathrm{s}$ parameters, regularized fitted Q-iteration is more computationally expensive.

These comparisons should be considered in the light of some important differences between fuzzy Q-iteration, on the one hand, and LSPI and fitted Q-iteration, on the other hand. The fact that all algorithms employ the same number of parameters means they employ *similarly*, but *not identically* powerful approximators: due to Requirements 1–3, the class of approximators considered by fuzzy Q-iteration is smaller, and therefore less powerful. These requirements also enable fuzzy Q-iteration to perform more computationally efficient updates, e.g., because the projection is reduced to an assignment (11).

Fuzzy Q-iteration is also related to interpolative V-iteration [26], which uses N state-dependent BFs to iteratively compute the optimal V-function $V^*(x)$. A discrete, finite action space is required, and the number of actions is denoted by $\bar{M}$. To make it comparable to fuzzy Q-iteration, interpolative V-iteration can be specialized for deterministic systems, and the BFs can be restricted in a similar way to Requirement 1. The resulting algorithm has a time complexity of $\mathrm{O}(N^2\bar{M})$ per iteration, and a memory complexity of $\mathrm{O}(N)$. So, if fuzzy Q-iteration uses all the discrete actions, i.e., $M = \bar{M}$, it has the same time complexity as interpolative V-iteration. The memory complexity of interpolative V-iteration is lower because the approximate V-function does not depend on the action.

6 Example: two-link manipulator

In this section, fuzzy Q-iteration is used to find a control policy for a two-link manipulator operating in a vertical plane (Figure 1). The control goal is to stabilize the manipulator pointing up. In order to create a challenging, highly nonlinear control problem, the torque of the first motor is limited to a value that is not sufficient to push the manipulator up in a single rotation. Instead, the first link needs to be swung back and forth (destabilized) to gather energy, prior to being pushed up and stabilized. The manipulator is a fourth-order nonlinear system with two control inputs, described by the model:

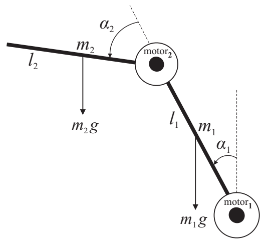

Fig. 1. Schematic drawing of the two-link manipulator.

$$M(\alpha)\ddot{\alpha} + C(\alpha, \dot{\alpha})\dot{\alpha} + G(\alpha) = \tau \tag{30}$$

where $\alpha = [\alpha_1, \alpha_2]^\mathrm{T}$, $\tau = [\tau_1, \tau_2]^\mathrm{T}$. The state vector contains the angles and angular velocities of the two links: $x = [\alpha_1, \dot{\alpha}_1, \alpha_2, \dot{\alpha}_2]^\mathrm{T}$, and the control vector is $u = \tau$. The angles wrap around in the interval $[-\pi, \pi)$ rad, and the velocities and torques are bounded as in Table 1. The sampling period is set to $T_\mathrm{s} = 0.05\,\mathrm{s}$, and the discrete-time dynamics f are obtained by a numerical integration of (30) between consecutive time steps. Table 1 gives the meaning and values (or domains) of the manipulator's variables. For the expressions of the mass matrix $M(\alpha)$, the Coriolis and centrifugal forces matrix $C(\alpha, \dot{\alpha})$, and the gravity matrix $G(\alpha)$, see [6].

Table 1
Physical ranges and parameter values of the manipulator.

Symbol	Domain or value	Units	Meaning
g	9.81	m/s^2	gravity
α_1; α_2	$[-\pi, \pi)$; $[-\pi, \pi)$	rad	link angles
$\dot{\alpha}_1$; $\dot{\alpha}_2$	$[-2\pi, 2\pi]$; $[-2\pi, 2\pi]$	rad/s	link velocities
τ_1; τ_2	$[-3, 3]$; $[-1, 1]$	Nm	motor torques
l_1; l_2	0.4; 0.4	m	link lengths
m_1; m_2	1.25; 0.8	kg	link masses
I_1; I_2	0.067; 0.043	kg m^2	link inertias
c_1; c_2	0.2; 0.2	m	centers of mass
b_1; b_2	0.08; 0.02	kg/s	joint dampings

The goal (stabilization around $\alpha = \dot{\alpha} = 0$) is expressed by the quadratic reward function:

$$\rho(x, u) = -x^\mathrm{T} Q_\mathrm{rew} x,\ Q_\mathrm{rew} = \mathrm{diag}(1, 0.05, 1, 0.05)$$

The discount factor is set to $\gamma = 0.98$. To apply fuzzy Q-iteration, triangular fuzzy partitions are defined for each state variable and then combined, as in Example 1. For the angles, a core is placed at the origin, and 6 logarithmically-spaced cores are placed on each side of the origin. For the velocities, a core is placed in the origin, and 3 logarithmically-spaced cores are used on

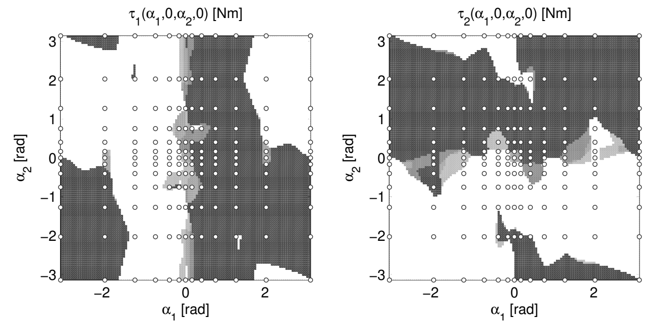

Fig. 2. A slice through the discrete-action policy, for $\dot{\alpha}_1 = \dot{\alpha}_2 = 0$ and parallel to the plane (α_1, α_2). Darker shades correspond to negative actions, lighter shades to positive actions. The fuzzy cores for the angle variables are represented as small white disks with dark edges.

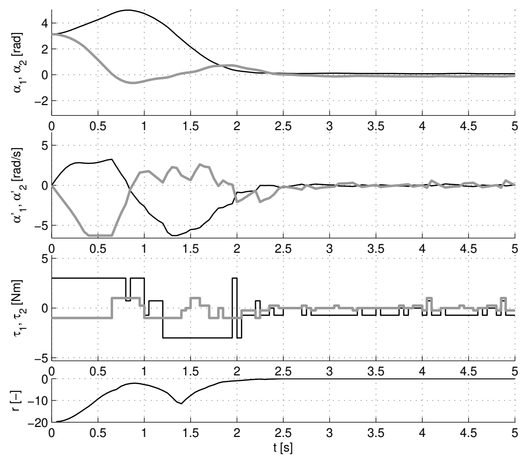

Fig. 3. A discrete-action controlled trajectory of the manipulator from the stable equilibrium (thin black line – link 1, thick gray line – link 2). Appropriate multiples of 2π have been added to the values of the angles, to eliminate any (artificial) discontinuities introduced by wrapping the angles in the interval $[-\pi, \pi)$.

each side of the origin. This leads to a total number of $(2 \cdot 6 + 1)^2 \cdot (2 \cdot 3 + 1)^2 = 8281$ MFs. The cores are spaced logarithmically to ensure a higher accuracy of the solution around the origin. Each torque variable is discretized using 5 logarithmically spaced values: $\tau_1 \in \{-3, -0.72, 0, 0.72, 3\}$ and $\tau_2 \in \{-1, -0.24, 0, 0.24, 1\}$. The convergence threshold is set to $\varepsilon_{\text{QI}} = 10^{-5}$. Synchronous fuzzy Q-iteration was applied, and it converged in 529 iterations.

Figure 2 presents a slice through the resulting discrete-action, near-optimal policy (12). Figure 3 presents a controlled trajectory of the manipulator, starting from the stable equilibrium. The controller successfully swings up and stabilizes the system in about 2.5 s. However, because only discrete actions are available, chattering is required to stabilize around the unstable equilibrium.

To avoid chattering, the interpolated policy (14) can be used. A slice through this policy is shown in Figure 4, and is clearly smoother than the discrete-action policy. Figure 5 presents the trajectory controlled with interpolated actions. Chattering has been eliminated. Note that, although the interpolated policy is continuous, the control signal sometimes varies significantly from one sampling instant to the next. This happens in state space regions where the policy is represented with a high resolution, and can therefore vary quickly.

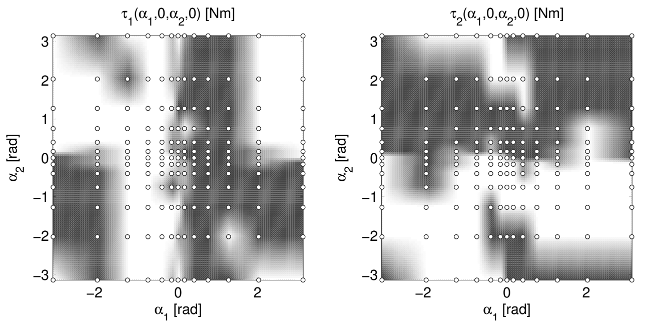

Fig. 4. The interpolated policy (zero-velocity slice).

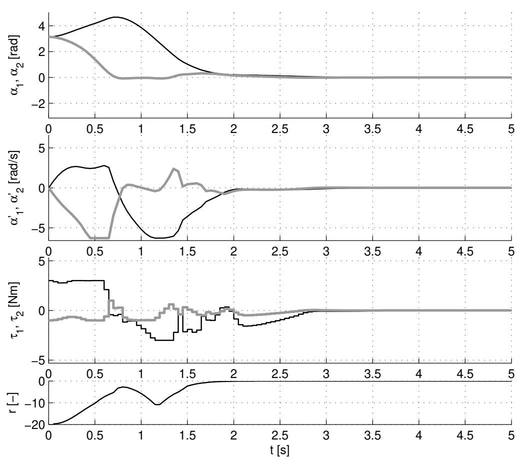

Fig. 5. A continuous-action controlled trajectory of the manipulator from the stable equilibrium.

Finally, we compare the solution of fuzzy Q-iteration with a solution obtained by the state-of-the-art fitted Q-iteration algorithm, employing ensembles of extremely randomized trees to approximate the Q-function [11]. Distinct ensembles are used to approximate the Q-function for each of the discrete actions, similarly to how the fuzzy approximator is working. Each ensemble consists of 50 trees, and the other tree construction parameters are set to their default values, namely $K = 4$, $n_{\min} = 2$ (see [11] for details). To enable a better comparison, fitted Q-iteration is supplied with the same samples as those employed by fuzzy Q-iteration, namely the cross product of the 8281 MF cores with the 25 discrete actions, leading to a total number of 207025 samples. This is close to the largest amount of samples that we could use, because the memory required to build the trees was already close to the capacity of our machine. Figure 6 illustrates the resulting policy.

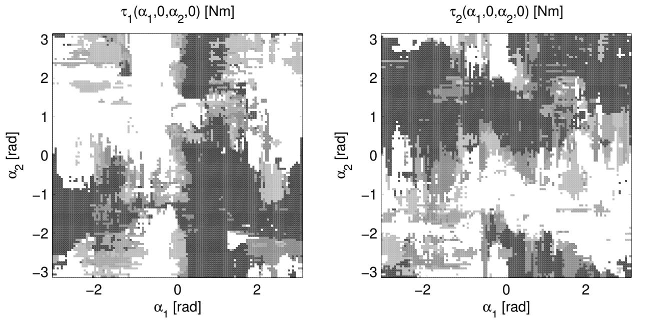

Fig. 6. The fitted Q-iteration policy (zero-velocity slice).

Although it resembles Figures 2 and 4, the fitted Q-iteration policy contains spurious (and probably incorrect) actions for many states. This policy is unable to swing the manipulator up from the stable equilibrium, so the resulting trajectory is not shown. We have also attempted tuning fitted Q-iteration (e.g., by supplying fewer discrete actions to simplify the learning problem, changing $n_{\min}$, and generating samples in other ways) but this did not improve the results. So, for this example, fuzzy Q-iteration outperforms fitted Q-iteration.

A possible explanation for the poorer results obtained by fitted Q-iteration in this example is the following. The tree construction algorithm includes random elements, which introduce variance in the approximate Q-values. If in some regions of the state space this variance is comparable to the differences between the Q-values of the discrete actions, the maximal Q-value may be incorrectly identified, which would lead to incorrect actions. An approximator variant that employs a single ensemble of trees, rather than a distinct ensemble for each discrete action, may alleviate such a problem – but would be less comparable with the fuzzy approximator.

7 Conclusions and future work

In this paper, we have considered fuzzy Q-iteration, an approximate dynamic programming algorithm that represents state-action value functions using a fuzzy partition of the state space and a discretization of the action space. The algorithm was shown to be convergent to a near-optimal solution, and consistent under continuity assumptions on the dynamics and the reward function. The performance of fuzzy Q-iteration was illustrated in an example involving the control of a two-link manipulator, in which fuzzy Q-iteration also compared favorably with fitted Q-iteration.

An important next step is extending fuzzy Q-iteration to stochastic problems. Section 4 indicated some possible approaches to perform this extension. The fuzzy approximator influences the computational complexity of fuzzy Q-iteration, as well as the accuracy of the solution. While we have considered in this paper that the MFs were given *a priori*, a technique to construct for a given accuracy an approximator with a small number of MFs would be very useful in practice. Furthermore, action-space approximators more powerful than discretization could be studied, e.g., approximators based on a fuzzy partition of the action space.

Acknowledgements

This research was financially supported by the BSIK-ICIS project (grant no. BSIK03024) and by the STW-VIDI project DWV.6188. We thank Pierre Geurts for allowing us the use of his code for building ensembles of regression trees.

References

[1] A. Antos, R. Munos, and Cs. Szepesvári. Fitted Q-iteration in continuous action-space MDPs. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 9–16. MIT Press, 2008.

[2] H. R. Berenji and D. Vengerov. A convergent actor-critic-based FRL algorithm with application to power management of wireless transmitters. *IEEE Transactions on Fuzzy Systems*, 11(4):478–485, 2003.

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 3rd edition, 2007.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

[5] M. Brown and C. Harris. *Neurofuzzy Adaptive Modeling and Control.* Prentice Hall, 1994.

[6] L. Buşoniu, B. De Schutter, and R. Babuška. Decentralized reinforcement learning control of a robotic manipulator. In *Proceedings 9th International Conference of Control, Automation, Robotics, and Vision (ICARCV-06)*, pages 1347–1352, Singapore, 5–8 December 2006.

[7] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška. Fuzzy approximation for convergent model-based reinforcement learning. In *Proceedings 2007 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE-07)*, pages 968–973, London, UK, 23–26 July 2007.

[8] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška. Consistency of fuzzy model-based reinforcement learning. In *Proceedings 2008 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE-08)*, pages 518–524, Hong Kong, 1–6 June 2008.

[9] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška. Continuous-state reinforcement learning with fuzzy approximation. In K. Tuyls, A. Nowé, Z. Guessoum, and D. Kudenko, editors, *Adaptive Agents and Multi-Agent Systems III*, volume 4865 of *Lecture Notes in Computer Science*, pages 27–43. Springer, 2008.

[10] C.-S. Chow and J. N. Tsitsiklis. An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control*, 36(8):898–914, 1991.

[11] D. Ernst, P. Geurts, and L. Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[12] A. M. Farahmand, M. Ghavamzadeh, Cs. Szepesvári, and S. Mannor. Regularized fitted Q-iteration for planning in continuous-space Markovian decision problems. In *Proceedings 2009 American Control Conference (ACC-09)*, pages 725–730, St. Louis, US, 10–12 June 2009.

[13] P. Y. Glorennec. Reinforcement learning: An overview. In *Proceedings European Symposium on Intelligent Techniques (ESIT-00)*, pages 17–35, Aachen, Germany, 14–15 September 2000.

[14] G. Gordon. Stable function approximation in dynamic programming. In *Proceedings 12th International Conference on Machine Learning (ICML-95)*, pages 261–268, Tahoe City, US, 9–12 July 1995.

[15] T. Horiuchi, A. Fujino, O. Katai, and T. Sawaragi. Fuzzy interpolation-based Q-learning with continuous states and actions. In *Proceedings 5th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE-96)*, pages 594–600, New Orleans, US, 8-11 September 1996.

[16] V. I. Istratescu. *Fixed Point Theory: An Introduction.* Springer, 2002.

[17] L. Jouffe. Fuzzy inference system learning by reinforcement methods. *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, 28(3):338–355, 1998.

[18] R. Kruse, J. E. Gebhardt, and F. Klowon. *Foundations of Fuzzy Systems.* Wiley, 1994.

[19] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[20] C.-K. Lin. A reinforcement learning adaptive fuzzy controller for robots. *Fuzzy Sets and Systems*, 137(3):339–352, 2003.

[21] R. Munos and A. Moore. Variable-resolution discretization in optimal control. *Machine Learning*, 49(2–3):291–323, 2002.

[22] R. Munos and Cs. Szepesvári. Finite time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9:815–857, 2008.

[23] M. S. Santos and J. Vigo-Aguiar. Analysis of a numerical dynamic programming algorithm applied to economic models. *Econometrica*, 66(2):409–426, 1998.

[24] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[25] Cs. Szepesvári and W. D. Smart. Interpolation-based Q-learning. In *Proceedings 21st International Conference on Machine Learning (ICML-04)*, pages 791–798, Bannf, Canada, 4–8 July 2004.

[26] J. N. Tsitsiklis and B. Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1–3):59–94, 1996.