Fuzzy Approximation for Convergent Model-Based Reinforcement Learning

Lucian Buşoniu, Damien Ernst, Bart De Schutter, and Robert Babuška

Abstract— **Reinforcement learning (RL) is a learning control paradigm that provides well-understood algorithms with good convergence and consistency properties. Unfortunately, these algorithms require that process states and control actions take only discrete values. Approximate solutions using fuzzy representations have been proposed in the literature for the case when the states and possibly the actions are continuous. However, the link between these mainly heuristic solutions and the larger body of work on approximate RL, including convergence results, has not been made explicit. In this paper, we propose a fuzzy approximation structure for the Q-value iteration algorithm, and show that the resulting algorithm is convergent. The proof is based on an extension of previous results in approximate RL. We then propose a modified, serial version of the algorithm that is guaranteed to converge at least as fast as the original algorithm. An illustrative simulation example is also provided.**

I. Introduction

Learning controllers can tackle problems where preprogrammed solutions are difficult or impossible to design. Reinforcement learning (RL) is a popular learning paradigm, mainly because it requires only mild assumptions on the process to be controlled, and is able to work without an explicit model [1]–[3]. A RL controller measures directly the process state, and receives feedback on the control performance in the form of a scalar reward signal. The learning objective is to maximize the cumulative reward signal. Well-understood algorithms with good convergence and consistency properties are available for solving the RL task, both when a model of the controlled process is available and when it is not. However, these algorithms require that the controller inputs (process states) and outputs (control actions) take values in a relatively small discrete set. When the state and / or action spaces are continuous or contain a large number of elements, approximate solutions must be used.

Approximation schemes have been proposed for model-based RL [4]–[6], as well as for model-free or model-learning RL [7]–[13].[1] Unfortunately, in general, approximate RL is not guaranteed to converge [4], [14]. One type of approximators for which many RL algorithms converge are linear basis functions, also known as kernel functions, averagers, and interpolative representations [4], [5], [7], [8].

Fuzzy approximation for RL is also popular in the literature, mainly for model-free RL. Fuzzy approximators are combined e.g., with Q-learning [15], yielding fuzzy Q-learning [16]–[18], or with actor-critic algorithms [1], yielding fuzzy actor-critic architectures [18]–[23]. For fuzzy Q-learning, Takagi-Sugeno fuzzy rule-bases are typically used. Actor-critic algorithms use fuzzy rule-bases for the actor element, and either fuzzy or other approximators (e.g., neural networks) for the critic element. Typically, fuzzy RL approaches are heuristic, and their convergence has not been studied, with the exception of the actor-critic algorithms in [20], [21]. These algorithms use special rulebase structures and parameter update rules in order to guarantee convergence. The results on convergence in the larger body of work in approximate RL have not been employed for fuzzy RL.

In this work, we propose a fuzzy approximator similar to others previously used for Q-learning [16], [18], but we combine it with the model-based Q-iteration algorithm (see e.g., [8]). This approximator works for continuous states and discrete actions; however, continuous actions can be handled by discretization. We show that the resulting fuzzy Q-iteration algorithm converges. We then propose an asynchronous, serial version of fuzzy Q-iteration, which converges at least as fast as the original algorithm. The modified algorithm has not, to the authors' best knowledge, been studied yet in approximate RL, although *exact* serial value iteration is widely used [3].

The remainder of this paper is structured as follows. Section II introduces the necessary RL elements, and Section III describes approximate model-based RL. Section IV describes the proposed fuzzy approximation structure. The properties of approximate Q-iteration using this structure are analyzed in Section V. Section VI illustrates the proposed algorithms on a simulated example. Finally, Section VII outlines ideas for future work and concludes the paper.

II. Background: Reinforcement Learning

In this section, we briefly introduce the RL task and characterize its optimal solution, following [1]–[3].

Consider a deterministic Markov decision process with the state space X, the action (control) space U, the state transition function $f : X \times U \to X$, and the reward function $\rho : X \times U \to \mathbb{R}$.[2] As a result of the control action u_k

Lucian Buşoniu, Bart De Schutter, and Robert Babuška are with the Center for Systems and Control of the Delft University of Technology, The Netherlands (email: i.l.busoniu@tudelft.nl, b@deschutter.info, r.babuska@tudelft.nl). Bart De Schutter is also with the Marine and Transport Technology Department of TU Delft. Damien Ernst is with Supélec, Rennes, France (email: damien.ernst@supelec.fr).

[1] Some authors use the term 'model-based RL' when referring to algorithms that build a model from interaction with the process. We use the term 'model-learning' for such techniques, and reserve the name 'model-based' for algorithms that rely on an *a priori* model of the process.

[2] A stochastic formulation is possible, where the state transitions are probabilistic. In that case, expected returns under these probabilistic transitions must be considered, and the results discussed still hold.

applied in state x_k, the state changes to $x_{k+1} = f(x_k, u_k)$. The controller receives feedback on its performance in the form of the scalar reward signal $r_{k+1} = \rho(x_k, u_k)$. This reward evaluates the immediate effect of action u_k, but says nothing directly about the long-term effects of this action. The controller chooses actions given the current state, according to its policy $h : X \to U$: $u_k = h(x_k)$.

The learning goal is the maximization, starting from the current moment in time ($k = 0$), of the discounted return:

$$\sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, u_k) \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor. The discounted return compactly represents the reward accumulated by the controller in the long-run. The learning task is therefore to maximize long-term performance, while only receiving feedback about immediate, one-step performance. This can be achieved by computing the optimal action-value function.

An action-value function (Q-function), $Q^h : X \times U \to \mathbb{R}$, gives the return of each state-action pair under a policy h:

$$Q^h(x, u) = \rho(x, u) + \sum_{k=1}^{\infty} \gamma^k \rho(x_k, h(x_k)) \tag{2}$$

where $x_1 = f(x, u)$ and $x_{k+1} = f(x_k, h(x_k))$, $\forall k$. The optimal action-value function is defined as $Q^*(x, u) = \max_h Q^h(x, u)$. Any policy that picks for every state the action with the highest optimal Q-value:

$$h^*(x) = \arg\max_u Q^*(x, u) \tag{3}$$

is then optimal (i.e., it maximizes the return (1)).

A central result in RL is the *Bellman optimality equation*:

$$Q^*(x, u) = \rho(x, u) + \gamma \max_{u' \in U} Q^*(f(x, u), u') \ \forall x, u \tag{4}$$

This equation states that the optimal value of action u taken in state x is the expected immediate reward plus the expected (discounted) optimal value attainable from the next state.

Let the set of all Q-functions be denoted by $\mathcal{Q}$. The Q-iteration mapping $T : \mathcal{Q} \to \mathcal{Q}$ is the right-hand side of the Bellman equation for any Q-function:

$$[T(Q)](x, u) = \rho(x, u) + \gamma \max_{u' \in U} Q(f(x, u), u') \tag{5}$$

Using this notation, the Bellman optimality equation (4) states that Q^* is a fixed point of T, i.e., $Q^* = T(Q^*)$. The following result is also well-known (see e.g., [24]).

Theorem 1: T is a contraction with factor γ in the infinity norm, i.e., for any pair of functions Q, Q', $\|T(Q) - T(Q')\|_\infty \leq \gamma \|Q - Q'\|_\infty$.

The Q-value iteration (Q-iteration) algorithm starts from an arbitrary Q-function Q_0 and at each iteration ℓ updates the Q-function using the formula $Q_{\ell+1} = T(Q_\ell)$. From Theorem 1, it follows that T has a unique fixed point, and from (4), this point is Q^*. Therefore, Q-iteration converges to Q^* as $\ell \to \infty$.

The standard Q-iteration uses an *a priori* model of the task (in the form of the transition and reward functions f, ρ). There are also algorithms that learn a model from experience, and others that do not use an explicit model at all [1], [2].

III. Function Approximation for Q-iteration

In general, the practical implementation of RL algorithms requires that Q-values are stored and updated explicitly for each state-action pair. This can only be realized when the number of state and action values is small. When the state and / or action spaces contain a large or infinite number of elements (e.g., they are continuous), approximate solutions must be used instead.

Parametric approximators use a parameter vector θ as a finite representation of the Q-function $\widehat{Q}$. The following mappings are defined in order to formalize parametric approximate Q-iteration (the notation follows [10]).

1) The *Q-iteration* mapping T, defined by equation (5).
2) The *approximation* mapping $F : \mathbb{R}^n \to \mathcal{Q}$, which for a given value of the parameter vector $\theta \in \mathbb{R}^n$ produces an approximate Q-function $\widehat{Q} = F(\theta)$.
3) The *projection* mapping $P : \mathcal{Q} \to \mathbb{R}^n$, which given a target Q-function Q computes the parameter vector θ such that $F(\theta)$ is as close as possible to Q (e.g., in a least-squares sense).

The notation $[F(\theta)](x, u)$ refers to the value of the Q-function $F(\theta)$ for the state-action pair (x, u). The notation $[P(Q)]_l$ refers to the l-th parameter in the parameter vector $P(Q)$.

Approximate Q-iteration starts with an arbitrary parameter vector θ_0 and at each iteration ℓ updates it using the composition of the mappings P, T, and F:

$$\theta_{\ell+1} = PTF(\theta_\ell) \tag{6}$$

Unfortunately, the approximate Q-iteration is not guaranteed to converge for an arbitrary approximator. Counter-examples can be found e.g., in [4], [14] for the related value-iteration algorithm, and those results apply directly to Q-iteration as well. One particular case in which approximate Q-iteration converges is when the composite mapping PTF can be shown to be a contraction [4], [5]. This property will be used below to show that fuzzy Q-iteration converges.

IV. Fuzzy Q-iteration

In this section, we propose a fuzzy approximation similar to others previously used for Q-learning [16], [18], but we combine it with the model-based Q-iteration algorithm. In the sequel, it is assumed that the action space is discrete, denoted by $U_0 = \{u_j | j = 1, \ldots, M\}$. This discrete set can be obtained from the discretization of an originally continuous action space. The state space can be either continuous or discrete. In the latter case, fuzzy approximation is useful when the number of discrete states is large.

The proposed approximation architecture relies on a fuzzy partition of the state space into N sets $\mathcal{X}_i$, each described by a membership function $\mu_i : X \to [0, 1]$. A state x belongs to each set i with a degree of membership $\mu_i(x)$. In the sequel the following assumptions are made:

1) The fuzzy partition is normalized, i.e., $\sum_{i=1}^{N} \mu_i(x) = 1$, $\forall x \in X$.
2) The fuzzy sets in the partition are normal, i.e., for every i there exists an x_i for which $\mu_i(x_i) = 1$ (and consequently, $\mu_{\underline{i}}(x_i) = 0$ for all $\underline{i} \neq i$ by Assumption 1). The state value x_i is called the core of set $\mathcal{X}_i$. This second assumption is made here for brevity in the description and analysis of the algorithms; it can be relaxed using results of [4].

For an example of a partition that satisfies the above conditions, see Figure 2 of Section VI.

The Q-function is approximated using a Takagi-Sugeno rule-base with singleton consequents. The rule-base has one input, the state x, and M outputs $q_1, \ldots, q_M$, the Q-values corresponding to each of the discrete actions $u_1, \ldots, u_M$. The i-th rule in this rule-base has the form:

$$R_i : \quad \textbf{if } x \textbf{ is } \mathcal{X}_i \textbf{ then } q_1 = \theta_{i,1}; q_2 = \theta_{i,2}; \ldots ; q_M = \theta_{i,M}$$

The parameters of this approximator are the singleton consequent values appearing in the rule-base. They are arranged in an $N \times M$ matrix θ, one row for each rule i and one column for each output j.[3] The logical expression 'x **is** $\mathcal{X}_i$' holds true with degree $\mu_i(x)$, the membership degree of x in $\mathcal{X}_i$. The fuzzy rule-base outputs the weighted sum of the consequent values $\theta_{i,j}$ in each rule, where the weight factor of a particular rule corresponds to the degree of fulfillment of its logical expression. Thus, the approximator takes as input the state-action pair (x, u_j) and outputs the Q-value:

$$\widehat{Q}(x, u_j) = [F(\theta)](x, u_j) = \sum_{i=1}^{N} \mu_i(x) \theta_{i,j} \tag{7}$$

This is a basis-functions form, with the basis functions only depending on the state. The approximator (7) can be regarded as M distinct approximators, one for each of the M discrete actions.

The projection mapping infers from a Q-function the values of the approximator parameters according to the relation:

$$\theta_{i,j} = [P(Q)]_{i,j} = Q(x_i, u_j) \tag{8}$$

This is a particular case of the least-squares solution:

$$P(Q) = \arg\min_{\theta} \sum_{(x,u) \in X_0 \times U_0} [Q(x,u) - F(\theta)(x,u)]^2$$

when the set of samples $X_0 \times U_0$ is the Cartesian product of the set of cores $X_0 = \{x_1, \ldots, x_M\}$ and the discrete action space U_0. Note that when the set of samples is different from this, least-squares projection no longer guarantees convergence [4].

The approximator specified in this way is a special case of several types of approximators previously considered for RL: interpolative representations [4], averagers [5], and representative-state techniques as described in [13]. It also shares similarities with barycentric interpolation [6]. The analysis in Section V will rely on theoretical properties of these approximators.

Algorithm 1 Parallel fuzzy Q-iteration

```
1: $\ell \leftarrow 0$; $\theta_0 \leftarrow 0$ (or arbitrary values)
2: repeat
3:     for $i = 1, \ldots, N, j = 1, \ldots, M$ do
4:         $\theta_{\ell+1,i,j} \leftarrow \rho(x_i, u_j) +$
               $\gamma \max_{\underline{j}} \sum_{\underline{i}=1}^{N} \mu_{\underline{i}}(f(x_i, u_j)) \theta_{\ell,\underline{i},\underline{j}}$
5:     end for
6:     $\ell \leftarrow \ell + 1$
7: until $\|\theta_\ell - \theta_{\ell-1}\|_\infty \leq \delta$
```

Algorithm 2 Serial fuzzy Q-iteration

```
$\ell \leftarrow 0$; $\theta_0 \leftarrow 0$ (or arbitrary values)
repeat
    $\theta \leftarrow \theta_\ell$
    for $i = 1, \ldots, N, j = 1, \ldots, M$ do
        $\theta_{i,j} \leftarrow \rho(x_i, u_j) + \gamma \max_{\underline{j}} \sum_{\underline{i}=1}^{N} \mu_{\underline{i}}(f(x_i, u_j)) \theta_{\underline{i},\underline{j}}$
    end for
    $\theta_{\ell+1} \leftarrow \theta$; $\ell \leftarrow \ell + 1$
until $\|\theta_\ell - \theta_{\ell-1}\|_\infty \leq \delta$
```

An explicit form of the approximate Q-value iteration algorithm using the approximator (7) and projection (8) is given in Algorithm 1. To establish the equivalence between Algorithm 1 and the approximate Q-iteration in the form (6), observe that the right-hand side in line 4 of Algorithm 1 corresponds to $[T(\widehat{Q}_\ell)](x_i, u_j)$, where $\widehat{Q}_\ell = F(\theta_\ell)$. Hence, line 4 can be written $\theta_{\ell+1,i,j} \leftarrow [PTF(\theta_\ell)]_{i,j}$ and the entire **for** loop described by lines lines 3–5 is equivalent to (6).

In Algorithm 1, only the parameters θ_ℓ at the end of the previous iteration are used in the computation of the updated values $\theta_{\ell+1}$. Algorithm 2 is an alternative version, which uses the updated parameters as soon as they are available. Since the parameters are updated in serial fashion, this version is called *serial* Q-iteration. Although the exact counterpart of this algorithm is widely used [1], [3], approximate serial Q-iteration has not, to the authors' best knowledge, been studied yet. To differentiate between the two versions, we hereafter call Algorithm 1 *parallel* fuzzy Q-iteration.

V. Analysis

In this section, the convergence of parallel and serial fuzzy Q-iteration is established. It is shown that there exists a parameter vector θ^* such that for both algorithms, $\theta_\ell \rightarrow \theta^*$ as $\ell \rightarrow \infty$. The consistency of the algorithms, i.e., the convergence to the optimal Q-function Q^* as the maximum distance between the cores of adjacent fuzzy sets goes to 0, is not studied here and is a topic for future research. It can be shown, however, that under certain conditions, $F(\theta^*)$ is within a given bound of the Q^* [4], [5].

Proposition 1: Fuzzy Q-iteration (Algorithm 1) converges.

[3] The matrix arrangement is adopted for convenience of notation only. For the theoretical study of the algorithms, the collection of parameters is still regarded as a vector, leading e.g., to $\|\theta\|_\infty = \max_{i,j} |\theta_{ij}|$.

Proof: The proof follows from the convergence proof of value iteration with averagers [5], or with interpolative representations [4]. This is because fuzzy approximation is an averager by the definition in [5], and an interpolative representation by the definition in [4]. For these types of approximator, P and F are nonexpansions, making PTF a contraction with factor γ, i.e., $\|PTF(\theta) - PTF(\theta')\|_\infty \leq \gamma \|\theta - \theta'\|_\infty$, for any θ, θ'. ■

Similarly to the convergence proof for exact serial value iteration in [3], it is shown below that the *approximate* serial Q-iteration Algorithm 2 converges.

Proposition 2: Serial fuzzy Q-iteration (Algorithm 2) converges.

Proof: Denote $n = N \cdot M$, and rearrange the matrix θ into a vector in $\mathbb{R}^n$, placing first the elements of the first row, then the second etc. The element at row i and column j of the matrix is now the l-th element of the vector, with $l = (i-1) \cdot M + j$.

Define for all $l = 0, \ldots, n$ recursively the mappings $S_l : \mathbb{R}^n \to \mathbb{R}^n$ as:

$$S_0(\theta) = \theta$$

$$[S_l(\theta)]_{\underline{l}} = \begin{cases} [PTF(S_{l-1}(\theta))]_{\underline{l}} & \text{if } \underline{l} = l \\ [S_{l-1}(\theta)]_{\underline{l}} & \underline{l} \in \{1, \ldots, n\} \setminus l \end{cases}$$

In words, S_l corresponds to updating the first l parameters using approximate serial Q-iteration, and S_n is a complete iteration of the approximate serial algorithm. Now we show that S_n is a contraction, i.e., $\|S_n(\theta) - S_n(\theta')\|_\infty \leq \gamma \|\theta - \theta'\|_\infty$, 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:

$$\begin{aligned} |[S_n(\theta)]_1 - [S_n(\theta')]_1| &= |[S_1(\theta)]_1 - [S_1(\theta')]_1| \\ &= |[PTF(\theta)]_1 - [PTF(\theta')]_1| \\ &\leq \gamma \|\theta - \theta'\|_\infty \end{aligned}$$

The last step follows from the contraction mapping property of PTF.

Similarly, the second element is only updated by S_2:

$$\begin{aligned} |[S_n(\theta)]_2 - [S_n(\theta')]_2| &= |[S_2(\theta)]_2 - [S_2(\theta')]_2| \\ &= |[PTF(S_1(\theta))]_2 - [PTF(S_1(\theta'))]_2| \\ &\leq \gamma \|S_1(\theta) - S_1(\theta')\|_\infty \\ &= \gamma \max\{|[PTF(\theta)]_1 - [PTF(\theta')]_1|, \\ &\qquad |\theta_2 - \theta'_2|, \ldots, |\theta_n - \theta'_n|\} \\ &\leq \gamma \|\theta - \theta'\|_\infty \end{aligned}$$

where $\|S_1(\theta) - S_1(\theta')\|_\infty$ is expressed by direct maximization over its elements, and the contraction mapping property of PTF 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, serial fuzzy Q-iteration converges. ■

This proof is actually more general, showing that approximate serial Q-iteration converges for any approximation F and projection P for which PTF is a contraction.

In the same way as exact serial value iteration [3], serial fuzzy Q-iteration can be shown to converge at least as quickly as Algorithm 1.

The following bound on the suboptimality of the computed Q-function follows from [5], but applies only when the action space of the *original* problem is discrete (i.e., no discretization is necessary prior to fuzzy Q-iteration).

Proposition 3: If the original action space is discrete and $\min_{\underline{Q}} \|Q^* - \underline{Q}\|_\infty = \varepsilon$ where $\underline{Q}$ is any fixed point of the composite mapping $FP : \mathcal{Q} \to \mathcal{Q}$, then fuzzy Q-iteration converges to θ^* such that:

$$\|Q^* - F(\theta^*)\|_\infty \leq \frac{2\varepsilon}{1-\gamma} \tag{9}$$

For example, any Q-function which satisfies $Q(x, u_j) = \sum_{i=1}^{N} \mu_i(x) Q(x_i, u_j) \;\; \forall x, j$ is a fixed point of FP. In particular, if the optimal Q-function has this form, i.e., is exactly representable by the chosen fuzzy approximator, the algorithm will converge to it (since in this case $\varepsilon = 0$).

In this section, we have established the parallel and serial fuzzy Q-iteration as theoretically sound algorithms for approximate RL in continuous-state tasks. When the original action space is discrete, bounds on the derived Q-function and policy were also shown to hold.

VI. Simulation Example

As an illustrative example, fuzzy Q-iteration is applied in simulation to the minimum-time stabilization of a two-link manipulator.

A. Two-link Manipulator Model

The two-link manipulator, depicted in Figure 1, is described by the fourth-order nonlinear model:

$$M(\alpha)\ddot{\alpha} + C(\alpha, \dot{\alpha})\dot{\alpha} + G(\alpha) = \tau \tag{10}$$

where $\alpha = [\alpha_1, \alpha_2]^{\mathrm{T}}$, $\tau = [\tau_1, \tau_2]^{\mathrm{T}}$. The system has two control inputs, the torques in the two joints, τ_1 and τ_2, and four measured outputs – the link angles, α_1, α_2, and their angular speeds $\dot{\alpha}_1, \dot{\alpha}_2$.

In the sequel, it is assumed that the manipulator operates in a horizontal plane, leading to $G(\alpha) = 0$. The mass matrix $M(\alpha)$ and the Coriolis and centrifugal forces matrix $C(\alpha, \dot{\alpha})$ have the following form:

$$M(\alpha) = \begin{bmatrix} P_1 + P_2 + 2P_3 \cos\alpha_2 & P_2 + P_3 \cos\alpha_2 \\ P_2 + P_3 \cos\alpha_2 & P_2 \end{bmatrix} \tag{11}$$

$$C(\alpha, \dot{\alpha}) = \begin{bmatrix} b_1 - P_3 \dot{\alpha}_2 \sin\alpha_2 & -P_3(\dot{\alpha}_1 + \dot{\alpha}_2) \sin\alpha_2 \\ P_3 \dot{\alpha}_1 \sin\alpha_2 & b_2 \end{bmatrix} \tag{12}$$

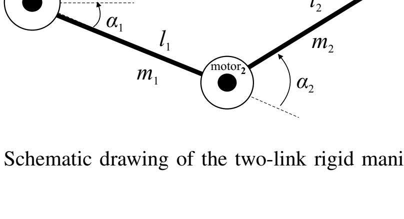

Fig. 1. Schematic drawing of the two-link rigid manipulator.

TABLE I

PHYSICAL PARAMETERS OF THE MANIPULATOR

Symbols and values	Meaning
$l_1 = l_2 = 0.4\,\mathrm{m}$	link lengths
$m_1 = 1.25\,\mathrm{kg}$, $m_2 = 0.8\,\mathrm{kg}$	link masses
$I_1 = 0.066\,\mathrm{kgm}^2$, $I_2 = 0.043\,\mathrm{kgm}^2$	link inertias
$c_1 = c_2 = 0.2\,\mathrm{m}$	centers of mass for the links
$b_1 = 0.08\,\mathrm{kg/s}$, $b_2 = 0.02\,\mathrm{kg/s}$	dampings in the joints
$\tau_{1,\max} = 1.5\,\mathrm{Nm}$, $\tau_{2,\max} = 1\,\mathrm{Nm}$	maximum motor torques
$\dot{\alpha}_{1,\max} = \dot{\alpha}_{2,\max} = 2\pi$ rad/s	maximum angular velocities

The meaning and values of the physical parameters of the system are given in Table I. Using these, the rest of the parameters in (10) can be computed by:

$$\begin{aligned} P_1 &= m_1 c_1^2 + m_2 l_1^2 + I_1 \qquad P_2 = m_2 c_2^2 + I_2 \\ P_3 &= m_2 l_1 c_2 \end{aligned} \tag{13}$$

B. Setup of the RL Algorithm

The input of the RL controller (the process state) is $x = [\alpha^{\mathrm{T}}, \dot{\alpha}^{\mathrm{T}}]^{\mathrm{T}}$, and its output (the command signal) is $u = \tau$. The discrete time step is set to $T_{\mathrm{S}} = 0.05$ and the discrete-time dynamics f are obtained by numerical integration of (10) between consecutive time steps.

The control goal is the stabilization of the system around $\alpha = \dot{\alpha} = 0$ in minimum time, with a tolerance of $\pm 5 \cdot \pi/180$ rad for the angles, and ± 0.1 rad/s for the angular speeds. The reward function chosen to express this goal is:

$$\rho(x,u) = \begin{cases} 0 & \text{if } |\alpha_p| \le 5 \cdot \pi/180 \text{ rad} \\ & \text{and } |\dot{\alpha}_p| \le 0.1 \text{ rad/s}, \; p = 1,2 \\ -1 & \text{otherwise} \end{cases} \tag{14}$$

where $[\alpha_1, \alpha_2, \dot{\alpha}_1, \dot{\alpha}_2]^{\mathrm{T}} = f(x,u)$ (the next state).

Each torque signal τ_p, $p = 1,2$ takes continuous values in the corresponding interval $[-\tau_{p,\max}, \tau_{p,\max}]$. To apply fuzzy Q-iteration, three discrete values are chosen for each torque: $-\tau_{p,\max}$ (maximal torque clockwise), 0, and $\tau_{p,\max}$ (maximal torque counter-clockwise).

Separately for each state component, a normal, complete triangular fuzzy partition is defined. Such a partition is completely determined by the core coordinates of the fuzzy sets. For $\dot{\alpha}_1$ and $\dot{\alpha}_2$, the interval is partitioned into 7 fuzzy sets, with their cores at $\{-360, -180, -30, 0, 30, 180, 360\} \cdot \pi/180$ rad/s. This partition is depicted as an example in Figure 2. For α_1 and α_2, 12 sets are used, with their cores at $\{-180, -130, -80, -30, -15, -5, 0, 5, 15, 30, 80, 130\} \cdot \pi/180$ rad. There is no fuzzy set with core π, because this is identical with the first set, having the core $-\pi$ (the angles evolve on a circle manifold $[-\pi, \pi)$).

The fuzzy partition of the state space is then defined as follows. One fuzzy set is computed for each combination $(i_1, \ldots, i_4)$ of individual sets for the four state components α_1, α_2, $\dot{\alpha}_1$, $\dot{\alpha}_2$. Such a fuzzy set has the following membership function:

$$\mu(x) = \mu_{\alpha_1,i_1}(\alpha_1) \cdot \mu_{\alpha_2,i_2}(\alpha_2) \cdot \mu_{\dot{\alpha}_1,i_3}(\dot{\alpha}_1) \cdot \mu_{\dot{\alpha}_2,i_4}(\dot{\alpha}_2) \tag{15}$$

This way of building the state space partition can be thought of as a conjunction of one-dimensional concepts corresponding to the fuzzy partitions of the individual state variables. The fuzzy partition computed in this way still satisfies Assumptions 1 and 2. It contains $(12 \cdot 7)^2 = 7056$ sets.

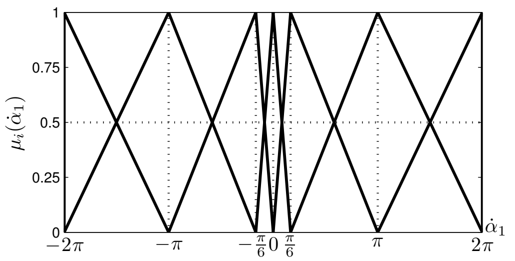

Fig. 2. The triangular fuzzy partition for the state variable $\dot{\alpha}_1 \in [-2\pi, 2\pi]$ (identical to the partition for $\dot{\alpha}_2$). Core values are in $\{-2\pi, -\pi, \frac{-\pi}{6}, 0, \frac{\pi}{6}, \pi, 2\pi\}$.

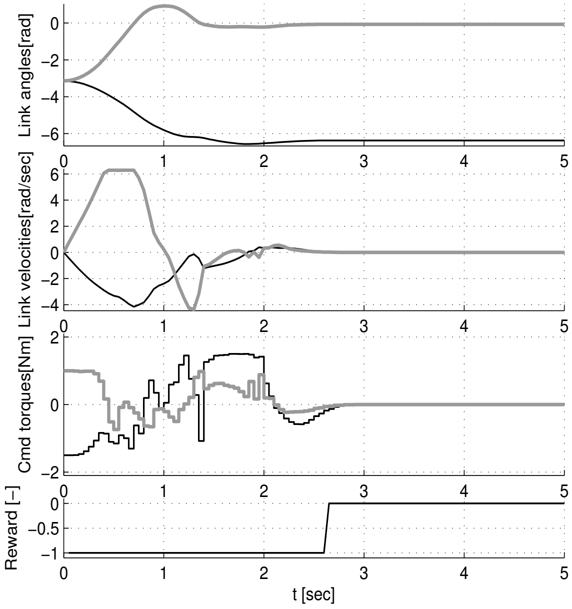

Fig. 3. State, command, and reward signals for RL control (thin black line – link 1, thick gray line – link 2). The initial state is $x_0 = [-\pi, -\pi, 0, 0]^{\mathrm{T}}$.

An approximate optimal action-value function is computed with serial and parallel fuzzy Q-iteration. The discount factor is set to $\gamma = 0.98$.

C. Results

Figure 3 presents a controlled trajectory starting from the initial state $x_0 = [-\pi, -\pi, 0, 0]^{\mathrm{T}}$, together with the corresponding command and reward signals. In order to obtain a continuous policy from the computed Q-function, the following heuristic is used. For any state value, an action is computed by interpolating between the best local actions, using the membership degrees as weights: $h(x) = \sum_{i=1}^{N} \mu_i(x) u_{j_i^*}$, where j_i^* is the index of the best local action for the core state x_i, $j_i^* = \arg\max_j \widehat{Q}^*(x_i, u_j) = \arg\max_j \theta_{i,j}^*$.

The controller successfully stabilizes the system in about 2.7 s. Because the control actions were originally continuous and had to be discretized prior to running the fuzzy Q-iteration, the bound (9) does not apply.

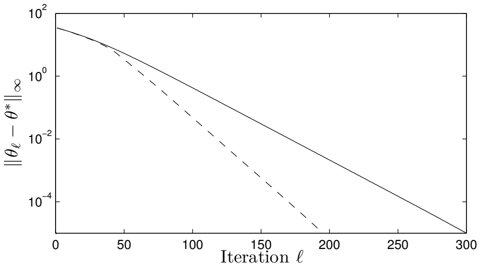

Fig. 4. Convergence (solid line – fuzzy Q-iteration, dashed line – serial fuzzy Q-iteration). The vertical axis represents the distance to the optimal parameter vector.

Figure 4 compares the convergence rate of the two algorithms. As expected from Section V, serial fuzzy Q-iteration converges more quickly. It gets within 10^{-5} of the optimum in under 200 iterations, as opposed to 300 for the parallel version. Because the computation time required by one iteration is nearly the same for the two algorithms, this translates directly into a decrease of the computation time required for serial fuzzy Q-iteration.

VII. Conclusions and Future Work

In this work, fuzzy approximation was combined with the model-based Q-iteration algorithm, and it was shown that the resulting algorithm is convergent. A serial version of the algorithm was then proposed that updates parameters more efficiently, and that converges at least as fast as the original, parallel version. The algorithms exhibited good performance on a nonlinear control problem with four continuous states and two continuous actions.

A first direction for future work is the study of the consistency of fuzzy Q-iteration, i.e., the asymptotic convergence to the optimal Q-function as the distance between the cores of the fuzzy sets shrinks to 0. Another important step is the study of online (model-learning or model-free) fuzzy RL algorithms that have good learning speed and low computational complexity. Possibly, results from kernel-based or interpolation-based RL apply to this case [7], [10]. It would also be interesting to extend our approach such that it can handle without discretization complex or continuous action spaces.

Finally, it would be very useful to find a method of adapting the structure of the fuzzy approximator (e.g., the cores of the triangular fuzzy sets in the partitions) during learning, using the current Q-function estimate, in order to minimize the distance to the optimum. It should be noted however that changing the approximator structure while learning an approximate Q-function could lead to convergence problems.

Acknowledgment

This research is financially supported by Senter, Ministry of Economic Affairs of the Netherlands within the BSIK-ICIS project "Interactive Collaborative Information Systems" (grant no. BSIK03024), by the NWO Van Gogh grant VGP 79-99, and by the STW-VIDI project "Multi-Agent Control of Large-Scale Hybrid Systems" (DWV.6188).

References

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

[2] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[3] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Athena Scientific, 2001, vol. 2.

[4] J. N. Tsitsiklis and B. Van Roy, "Feature-based methods for large scale dynamic programming," *Machine Learning*, vol. 22, no. 1–3, pp. 59–94, 1996.

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

[6] R. Munos and A. Moore, "Variable-resolution discretization in optimal control," *Machine Learning*, vol. 1, pp. 1–31, 2001.

[7] D. Ormoneit and S. Sen, "Kernel-based reinforcement learning," *Machine Learning*, vol. 49, pp. 161–178, 2002.

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

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

[10] C. Szepesvári and W. D. Smart, "Interpolation-based Q-learning," in *Proceedings 21st International Conference on Machine Learning (ICML-04)*, Bannf, Canada, July 4–8 2004.

[11] L. Baird and A. Moore, "Gradient descent for general reinforcement learning," in *Advances in Neural Information Processing Systems 11 (NIPS-98)*, Denver, US, 30 November – 5 December 1998, pp. 968–974.

[12] S. P. Singh, T. Jaakkola, and M. I. Jordan, "Reinforcement learning with soft state aggregation." in *Advances in Neural Information Processing Systems 7*, Denver, Colorado, USA, 1994, pp. 361–368.

[13] D. Ernst, "Near optimal closed-loop control. Application to electric power systems," Ph.D. dissertation, University of Liège, Belgium, March 2003.

[14] J. Boyan and A. Moore, "Generalization in reinforcement learning: Safely approximating the value function," in *Advances in Neural Information Processing Systems 7 (NIPS-94)*, Denver, Colorado, US, 1994, pp. 369–376.

[15] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

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

[17] 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)*, New Orleans, US, 8-11 September 1996, pp. 594–600.

[18] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews*, vol. 28, no. 3, pp. 338–355, 1998.

[19] H. R. Berenji and P. Khedkar, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 724–740, 1992.

[20] 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*, vol. 11, no. 4, pp. 478–485, 2003.

[21] D. Vengerov, N. Bambos, and H. R. Berenji, "A fuzzy reinforcement learning approach to power control in wireless transmitters," *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, vol. 35, no. 4, pp. 768–778, 2005.

[22] A. Bonarini, "Evolutionary learning, reinforcement learning, and fuzzy rules for knowledge acquisition in agent-based systems," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1334–1346, 2001.

[23] C.-K. Lin, "A reinforcement learning adaptive fuzzy controller for robots," *Fuzzy Sets and Systems*, vol. 137, pp. 339–352, 2003.

[24] S. Jodogne, "Closed-loop learning of visual control policies," Ph.D. dissertation, University of Liège, Belgium, December 2006.