Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Nonlinear Disturbance Compensation and Reference Tracking via Reinforcement Learning with Fuzzy Approximators

Y. Efe Bayiz and Robert Babuška

Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, 2628CD, Delft, The Netherlands, r.babuska@tudelft.nl

Abstract: Reinforcement Learning (RL) algorithms can learn optimal control laws for nonlinear dynamic systems without relying on a mathematical model of the system to be controlled. While RL can in principle discover control laws from scratch, by solely interacting with the process, in practice this does not yield any significant advantages. Learning control laws from scratch is lengthy and may lead to system damage due to the trial and error nature of the learning process. In this paper, we adopt a different and largely unexplored approach: a nominal control law is used to achieve reasonable, yet suboptimal, performance and a RL agent is trained to act as a nonlinear compensator whose task is to improve upon the performance of the nominal controller. The RL agent learns by means of an actor-critic algorithm using a plant model acquired on-line, alongside the critic and actor. Fuzzy approximators are employed to represent all the adjustable components of the learning scheme. One advantage of fuzzy approximators is the straightforward way in which they allow for the inclusion of prior knowledge. The proposed control scheme is applied to a reference tracking problem of 1-DOF robot arm influenced by an unknown payload disturbance due to gravity. The nominal controller is a PD controller, which is unable to properly compensate the effect of the disturbance considered. Simulation results indicate that the novel method is able to learn to compensate the disturbance for any reference angle varying throughout the experiment.

Keywords: reinforcement learning, nonlinear disturbance compensation, fuzzy modeling, model-based learning control

1. INTRODUCTION

Reinforcement Learning (RL) can be used to optimize the control performance for a large class of nonlinear dynamic systems. The user sets a certain goal by specifying a suitable reward function for the RL controller, and the controller then learns to maximize the cumulative reward received over time. While this method can in principle discover control laws from scratch by solely interacting with the process, in practice this does not yield any significant advantages. Direct applications of RL to a control problem typically starts from scratch and the performance only gradually improves throughout the learning trials. The desired performance is generally achieved after a long period of unpredictable and potentially damaging behavior. This is not acceptable in practical applications, especially if a reasonable controller already exists.

The use of RL as a compensator cooperating with a readily available controller has not been much investigated in the literature, see (Brujeni et al., 2010) for a notable exception. This paper presents such a control scheme, using a nominal controller in combination with a compensator learned through the actor-critic RL. The latter component of the scheme learns to deal with disturbances or nonlinearities that cannot be rejected by the nominal controller. We employ fuzzy function approximators to approximate the actor, the critic and the process model involved in the MLAC (model-learning actor critic) RL method (Grondman et al., 2012). Through a hierarchical structure of the critic's rule base, the scheme can handle varying references. For other approaches to reference tracking in the context of RL see (Kiumarsi-Khomartash et al., 2013; Modares and Lewis, 2013).

As the learning task is interacting with a well performing nominal controller, the initial performance of the control scheme is comparable with that of the nominal controller and the performance improves over the course of time. Although, we do not formally prove the convergence of the learning, experimental results indicate that with reasonable settings of the learning parameters, the closed-loop performance of the proposed control scheme never becomes worse than the performance of nominal controller alone.

The use of fuzzy approximators enables the use of prior knowledge within the learning task (Berenji and Vengerov, 2003). For instance, *a priori* information on the form of the nonlinearities can be incorporated in the fuzzy approximators in terms of constraints on their parameters (Abonyi et al., 2000). Consequently, the complexity of the learning problem can be reduced, which leads to a significant acceleration on the learning speed.

The rest of the paper is organized as follows. Section 2 gives the preliminaries of RL, actor-critic algorithms and fuzzy modeling. Then, the novel control scheme is pre-

Copyright © 2014 IFAC

sented and the algorithm is explained in Section 3. Finally, the results of simulation experiments with 1-DOF robot arm reference-tracking are presented in Section 4. Conclusions are drawn in Section 5.

2. PRELIMINARIES

2.1 Reinforcement Learning

Reinforcement learning can find optimal policies for systems modeled as the Markov Decision Process (MDP). For the sake of brevity, and without the loss of generality, we present the deterministic MDP framework. An MDP is defined by the tuple $M(X, U, \mathbf{f}, \rho)$ where X is the state space, U is the action space, $\mathbf{f} : X \times U \to X$ is the state transition function and $\rho : X \times U \to \mathbb{R}$ is the reward function.

The process to be controlled is described by the state transition function

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k). \tag{1}$$

which returns the state $\mathbf{x}_{k+1}$ that the process has reached from state $\mathbf{x}_k$ after applying action $\mathbf{u}_k$, where k denotes discrete time. After each transition, the controller receives a scalar reward $r_{k+1} \in \mathbb{R}$, calculated by the reward function

$$r_{k+1} = \rho(\mathbf{x}_k, \mathbf{u}_k).$$

The goal in RL is: given the above reward function $\rho(\cdot, \cdot)$ and the transition samples $(\mathbf{u}_k, \mathbf{x}_k, \mathbf{x}_{k+1})$ generated by process (1), find an optimal control policy $\boldsymbol{\pi}\colon X \to U$, such that the discounted sum of rewards received while following that policy is maximized.

The discounted sum of rewards is called the return J. To estimate the return for a given state $\mathbf{x}$ and a given policy $\boldsymbol{\pi}$, define the value function $V^{\boldsymbol{\pi}} : X \to \mathbb{R}$ as follows:

$$V^{\boldsymbol{\pi}}(\mathbf{x}) = \sum_{j=0}^{\infty} \gamma^j \rho(\mathbf{x}_j, \mathbf{u}_j), \text{ with } \mathbf{x}_0 = \mathbf{x},\ \mathbf{u}_j = \boldsymbol{\pi}(\mathbf{x}_j) \tag{2}$$

with $\gamma \in [0, 1)$ the discount factor. The value function satisfies the Bellman equation

$$V^{\boldsymbol{\pi}}(\mathbf{x}) = \rho(\mathbf{x}, \boldsymbol{\pi}(\mathbf{x})) + \gamma V^{\boldsymbol{\pi}}(\mathbf{f}(\mathbf{x}, \boldsymbol{\pi}(\mathbf{x}))), \tag{3}$$

forming the basis upon which RL improves the policy $\boldsymbol{\pi}$, as explained next.

2.2 Model-Learning Actor-Critic Algorithm

Reinforcement learning in continuous state and action spaces requires the use of function approximators. Actor-critic RL algorithms employ two separate function approximators to represent the policy (actor) and the value function (critic). The critic is parameterized by a parameter vector $\boldsymbol{\theta}^c \in \mathbb{R}^q$ and from here on denoted as $V(\mathbf{x}; \boldsymbol{\theta}^c)$.

The critic parameter vector is updated by the following TD(λ) update rule:

$$\delta_k = r_{k+1} + \gamma V(\mathbf{x}_{k+1}; \boldsymbol{\theta}^c_k) - V(\mathbf{x}_k; \boldsymbol{\theta}^c_k) \tag{4}$$

$$\boldsymbol{\theta}^c_{k+1} = \boldsymbol{\theta}^c_k + \alpha_c \delta_k \boldsymbol{\zeta}_k \tag{5}$$

$$\boldsymbol{\zeta}_{k+1} = \lambda\gamma\boldsymbol{\zeta}_k + \nabla_{\boldsymbol{\theta}^c} V(\mathbf{x}_k; \boldsymbol{\theta}^c_k) \tag{6}$$

where $\alpha_c \in (0, 1]$ is the critic learning rate, ζ the eligibility trace with λ its associated forgetting factor, δ the temporal difference, and ∇ denotes the gradient.

The actor is parameterized by $\boldsymbol{\theta}^a \in \mathbb{R}^r$ and denoted as $\boldsymbol{\pi}(\mathbf{x}, \boldsymbol{\theta}^a)$. The actor parameter vector is updated by:

$$\boldsymbol{\theta}^a_{k+1} = \boldsymbol{\theta}^a_k + \alpha_a \nabla_{\boldsymbol{\theta}^a} J_k \tag{7}$$

where α_a is the actor learning rate and $\nabla_{\boldsymbol{\theta}^a} J_k$ is the gradient of the return with respect to the policy parameter $\boldsymbol{\theta}^a$.[1] A common way to approximate $\nabla_{\boldsymbol{\theta}^a} J_k$ is to use the following heuristic estimate:

$$\nabla_{\boldsymbol{\theta}^a} J_k \approx \Delta\mathbf{u}_k \nabla_{\boldsymbol{\theta}^a} \boldsymbol{\pi}(\mathbf{x}_k; \boldsymbol{\theta}^a_k) \tag{8}$$

where $\Delta\mathbf{u}_k$ is an exploration term drawn from a zero-mean normal distribution and added to control value calculated by the policy $\boldsymbol{\pi}(\mathbf{x}_k, \boldsymbol{\theta}^a_k)$:

$$\mathbf{u}_k = \boldsymbol{\pi}(\mathbf{x}_k; \boldsymbol{\theta}^a_k) + \Delta\mathbf{u}_k\,.$$

In this paper, we use a more accurate estimate of the policy gradient (8), as introduced in the Model Learning Actor-Critic (MLAC) method (Grondman et al., 2012). In addition to the actor and the critic, the MLAC algorithm learns a process model

$$\hat{\mathbf{x}}_{k+1} = \hat{\mathbf{f}}(\mathbf{x}_k, \mathbf{u}_k; \boldsymbol{\theta}^p_k)$$

with $\hat{\mathbf{x}}_{k+1}$ the estimated next state for the $(\mathbf{x}_k, \mathbf{u}_k)$ pair and $\boldsymbol{\theta}^p \in \mathbb{R}^{n \times s}$ an adjustable parameter matrix (n is the dimension of the state). The model parameters are updated by the following first-order gradient descent rule:

$$\boldsymbol{\theta}^p_{k+1} = \boldsymbol{\theta}^p_k + \alpha_p(\mathbf{x}_{k+1} - \hat{\mathbf{x}}_{k+1}) \nabla_{\boldsymbol{\theta}^p} \hat{\mathbf{f}}(\mathbf{x}_k, \mathbf{u}_k; \boldsymbol{\theta}^p_k) \tag{9}$$

where α_p is the learning rate of the process model.

The learned process model allows us to calculate the Jacobian of the process model $\hat{\mathbf{f}}$ with respect to $\mathbf{u}$ and consequently to approximate the policy gradient by using the chain rule as follows:

$$\nabla_{\boldsymbol{\theta}^a} J_k \approx \nabla_{\mathbf{x}} V(\mathbf{x}_{k+1}; \boldsymbol{\theta}^c_k) \nabla_{\mathbf{u}} \hat{\mathbf{f}}(\mathbf{x}_k, \mathbf{u}_k; \boldsymbol{\theta}^p_k) \nabla_{\boldsymbol{\theta}^a} \boldsymbol{\pi}(\mathbf{x}_k; \boldsymbol{\theta}^a_k)$$

Note that contrary to (8), the policy gradient of the MLAC method does not use the exploration term $\Delta\mathbf{u}$. However, exploration is still needed to make the value function estimate cover a sufficiently large portion of the state space.

2.3 Fuzzy Rule-Based Parameterization

The MLAC algorithm relies on three function approximators which are in this paper defined as singleton and Takagi-Sugeno (TS) fuzzy models (Takagi and Sugeno, 1985). A generic function $\mathbf{y} = \mathbf{h}(\mathbf{x})$ with input $\mathbf{x} \in \mathbb{R}^n$ and output $\mathbf{y} \in \mathbb{R}^m$ is approximated by the following rule base:

$$\textbf{If } \mathbf{z} \text{ is } \mathcal{A}_i \ \textbf{ then } \mathbf{y}_i = \boldsymbol{\beta}_i^\top \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}, \quad i = 1, 2, \ldots, K$$

where $\mathbf{z} \in \mathbb{R}^{n_z}$ denotes the vector of antecedent variables which are typically some selected elements of $\mathbf{x}$, $\mathcal{M}_i$ is a fuzzy set (linguistic term) of rule i, $\boldsymbol{\beta}_i \in \mathbb{R}^{m \times (n+1)}$ is the consequent parameter matrix for rule i and K is the number of rules in the rule base. The rule antecedent can also be defined by using the individual elements of $\mathbf{z}$, e.g., in the conjunctive form:

$$\begin{aligned} &\textbf{If } z_1 \text{ is } \mathcal{A}_{i,1} \textbf{ and } \ldots \textbf{ and } z_{n_z} \text{ is } \mathcal{A}_{i,n_z} \\ &\textbf{then } \mathbf{y}_i = \boldsymbol{\beta}_i^\top \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}, \quad i = 1, 2, \ldots, K\,. \end{aligned} \tag{10}$$

[1] In the RL literature called the *policy gradient*.

This fuzzy rule-based structure allows for the use of prior knowledge in the selection of the antecedent variables $\mathbf{z}$ and the associated membership functions $\mu_{\mathcal{A}_{i,j}}$.

The output of the fuzzy model is calculated by using the weighted-mean interpolation

$$\mathbf{y} = \sum_{i=1}^{K} \omega_i(\mathbf{z})\mathbf{y}_i = \sum_{i=1}^{K} \omega_i(\mathbf{z})\boldsymbol{\beta}_i^\top \begin{pmatrix}\mathbf{x}\\ 1\end{pmatrix}$$

where $\omega_i(\mathbf{z})$ is the i^{th} antecedent's normalized degrees of fulfillment, for the conjunctive form (10) given by:

$$\omega_i(\mathbf{z}) = \frac{\prod_{j=1}^{n_z} \mu_{\mathcal{A}_{i,j}}(\mathbf{z})}{\sum_{i'=1}^{K}\prod_{j=1}^{n_z} \mu_{\mathcal{A}_{i',j}}(\mathbf{z})}$$

In matrix notation, $\mathbf{y}$ can be expressed as

$$\mathbf{y} = \boldsymbol{\Omega}(\mathbf{z})^\top \mathbf{B}^\top \begin{pmatrix}\mathbf{x}\\ 1\end{pmatrix}$$

where the degrees of fulfillment are collected in matrix $\boldsymbol{\Omega}(\mathbf{z})$ and the consequent parameter matrices are collected in matrix $\mathbf{B}$ defined by

$$\boldsymbol{\Omega}(\mathbf{z}) = \begin{pmatrix}\omega_1(\mathbf{z})\mathbf{I}_m\\ \omega_2(\mathbf{z})\mathbf{I}_m\\ \vdots\\ \omega_K(\mathbf{z})\mathbf{I}_m\end{pmatrix}, \qquad \mathbf{B} = (\boldsymbol{\beta}_1\ \boldsymbol{\beta}_2\ \dots\ \boldsymbol{\beta}_K)$$

with $\mathbf{I}_m$ the $m \times m$ identity matrix. In this paper, the antecedent membership functions are fixed, while the consequent parameters are adjustable.

3. NONLINEAR DISTURBANCE WITH REINFORCEMENT LEARNING

We propose a control scheme which combines a nominal controller, typically a linear controller designed by standard control methods, with an actor-critic RL algorithm to learn to compensate for nonlinear deterministic disturbances and other effects due to the process nonlinearity. These effects are supposed to be input additive and consequently the compensatory control action of the RL actor is added to the control input calculated by the nominal controller, see the diagram in Figure 1.

3.1 Fuzzy Rule Base Representation of Actor

Following the generic structure presented in Section 2.3, the actor rule base has the form:

$$\textbf{If } \mathbf{z}^a \text{ is } \mathcal{A}_i^a \textbf{ then } \mathbf{u}_i = \boldsymbol{\beta}_i^{a\top} \begin{pmatrix}\mathbf{x}\\ 1\end{pmatrix}, \quad i = 1, 2, \dots, K^a$$

where the superscript a denotes the parameters of the actor. For the update, the columns of the parameter matrix

$$\mathbf{B}^a = (\boldsymbol{\beta}_1^a\ \boldsymbol{\beta}_2^a\ \dots\ \boldsymbol{\beta}_{K^a}^a)$$

are stacked into one vector $\boldsymbol{\theta}^a \in \mathbb{R}^{m(n+1)K^a}$, with m the input signal dimension, n the state dimension and K^a the number of rules in the actor rule base.

Note that the notational distinction between the antecedent variables $\mathbf{z}$ and the consequent variables $\mathbf{x}$ is explicit, as they can generally be different. Typically, variables responsible for nonlinear behavior are included in the antecedent, while they do not necessarily have to be a part of the consequent. Similarly, the consequent can include

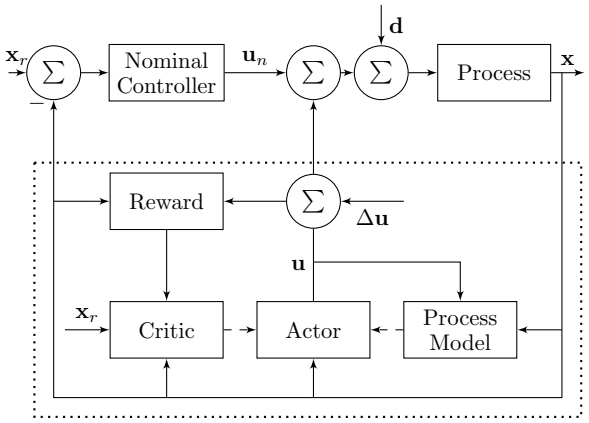

Fig. 1. A block diagram of the proposed control scheme. The solid lines indicate actual signals whereas the dashed lines indicate the use of the gradient from a particular block. Signal $\mathbf{d}$ is the additive disturbance that is to be compensated. The dotted box encloses the learning compensator.

variables in which the output is linear and these variables are then not present in the antecedent. This holds also for the fuzzy approximators of the process model and of the critic. The example in Section 4 illustrates the concept on a practical problem.

3.2 Fuzzy Rule Base Representation of Process Model

The process model is approximated in the same manner as the actor:

$$\textbf{If } \mathbf{z}^p \text{ is } \mathcal{A}_i^p \textbf{ then } \hat{\mathbf{f}}_i(\mathbf{x},\mathbf{u}) = \boldsymbol{\beta}_i^{p\top} \begin{pmatrix}\mathbf{x}\\ \mathbf{u}\\ 1\end{pmatrix}, \quad i = 1, 2, \dots, K^p$$

with the superscript p denoting the parameters of the process model. For the update, the columns of the parameter matrix

$$\mathbf{B}^p = (\boldsymbol{\beta}_1^p\ \boldsymbol{\beta}_2^p\ \dots\ \boldsymbol{\beta}_{K^p}^p)$$

are stacked into one vector $\boldsymbol{\theta}^p \in \mathbb{R}^{n(n+m+1)K^p}$, with n the state dimension, m the input dimension and K^p the number of rules in the process model rule base. Note that in the scheme in Figure 1, the process model approximates the closed-loop system including the process and the nominal compensator.

3.3 Fuzzy Rule Base Representation of Critic

The learning scheme works for arbitrary asymptotically constant state references, which is an improvement over the RL solutions found in the literature. However, to guarantee the Markov property, the critic must include the state reference $\mathbf{x}_r$ as one of its inputs, see Figure 1. To this end, we employ the two-layer hierarchical rule-base structure shown in Figure 2.

The individual value functions $W_j(\mathbf{x} - \mathbf{c}_j)$ are defined for constant, a priori selected, references $\mathbf{c}_j$ and are approximated with the following rule base with the total of K_j^c rules:

$$\textbf{If } \mathbf{z}_j^c \text{ is } \mathcal{C}_{i,j} \textbf{ then } W_{i,j}(\mathbf{x} - \mathbf{c}_j) = \boldsymbol{\beta}_{i,j}^{c\top} \begin{pmatrix}\mathbf{x} - \mathbf{c}_j\\ 1\end{pmatrix},$$
$$i = 1, 2, \dots, K_j^c$$

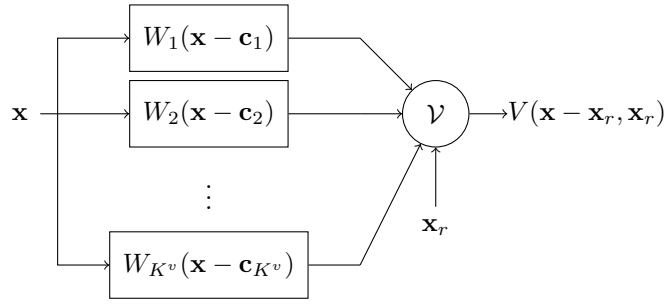

Fig. 2. Hierarchical structure of the critic. $\mathcal{V}$ indicates the fuzzy mean aggregation used to approximate $V(\mathbf{x}-\mathbf{x}_r,\mathbf{x}_r)$.

with the antecedent variables $\mathbf{z}_j^c$ being some selected elements of $\mathbf{x}$, $\mathcal{C}_{i,j}$ the fuzzy terms of rule i. The rules involved in $\mathcal{V}$, approximating the overall value function $V(\mathbf{x}-\mathbf{x}_r,\mathbf{x}_r)$, are given as follows:

$$\textbf{If } \mathbf{x}_r \text{ is } \mathcal{V}_j \textbf{ then } V_j(\mathbf{x}-\mathbf{x}_r,\mathbf{x}_r)=W_j(\mathbf{x}-\mathbf{c}_j)$$
$$j=1,2,\ldots,K^v$$

where $\mathcal{V}_j$ is the antecedent fuzzy term of rule j. For the update, the columns of the parameter matrix

$$\mathbf{B}_j^c=\left(\boldsymbol{\beta}_{1,j}^c\ \boldsymbol{\beta}_{2,j}^c\ \ldots\ \boldsymbol{\beta}_{K_j^c,j}^c\right)$$

are stacked into one vector $\boldsymbol{\theta}_j^c\in\mathbb{R}^{(n+1)K_j^c}$ with K_j^c the number of rules in the jth critic rule base.

To compensate for disturbances and to achieve a good reference-tracking performance, the reward function must also be modified. For each W_j, the corresponding reward is defined as

$$r_j=\rho(\mathbf{x}-\mathbf{c}_j,\mathbf{u})$$

The $\rho(\cdot)$ function is constructed such that it attains its maximum when $\mathbf{x}-\mathbf{c}_j$ is 0.

3.4 Parameter Update Laws

The individual value functions are updated as follows:

$$\delta_{j,k}=r_{j,k+1}+\gamma W_j(\mathbf{x}_{k+1}-\mathbf{c}_j)-W_j(\mathbf{x}_k-\mathbf{c}_j)$$
$$\boldsymbol{\theta}_{j,k+1}^c=\boldsymbol{\theta}_{j,k}^c+\alpha_c\delta_{j,k}\boldsymbol{\zeta}_{j,k}$$
$$\boldsymbol{\zeta}_{j,k+1}=\lambda\gamma\boldsymbol{\zeta}_{j,k}+\nabla_{\boldsymbol{\theta}_j^c}W_j(\mathbf{x}_k-\mathbf{c}_j)$$

where, k is the time index, $\boldsymbol{\zeta}_{j,k}$ the eligibility trace and $r_{j,k+1}=\rho(\mathbf{x}_{k+1}-\mathbf{c}_j,\mathbf{u}_k)$ the reward. The update rules for the actor and process model remain the same as in Section 2.2:

$$\boldsymbol{\theta}_{k+1}^a=\boldsymbol{\theta}_k^a+\alpha_a\nabla_{\mathbf{x}}V(\mathbf{x}_{k+1}-\mathbf{x}_r,\mathbf{x}_r)\nabla_{\mathbf{u}}\hat{f}(\mathbf{x}_k,\mathbf{u}_k)\nabla_{\boldsymbol{\theta}^a}\boldsymbol{\pi}(\mathbf{x})$$

and

$$\boldsymbol{\theta}_{k+1}^p=\boldsymbol{\theta}_k^p+\alpha_p(\mathbf{x}_{k+1}-\hat{\mathbf{x}}_{k+1})\nabla_{\boldsymbol{\theta}^p}\hat{f}(\mathbf{x}_k,\mathbf{u}_k)$$

The entire model-learning actor-critic (MLAC) algorithm for nonlinear disturbance rejection is summarized in Algorithm 1. Note that other RL algorithms, such as the standard actor-critic, can be applied in this setting as well.

4. REFERENCE TRACKING FOR 1-DOF ROBOT ARM

The proposed control scheme is applied a 1-DOF robot arm operating under the influence of gravity as shown in Fig. 3. The equation of motion is:

Algorithm 1 Nonlinear disturbance rejection via MLAC with fuzzy approximators

Input: γ, λ, and learning rates α
1: Initialize $\boldsymbol{\zeta}_{j,0},\forall j$, and fuzzy function approximators
2: Apply $\mathbf{u}_{n,0}+\Delta\mathbf{u}_0$
3: $k\leftarrow 0$
4: **loop**
5: Measure $\mathbf{x}_{k+1}$, $r_{j,k+1},\forall j$ and $\mathbf{x}_r$
6: Choose $\mathbf{u}_{n,k+1}$ according to nominal control law
7: ***% Choose compensation action and update actor***
8: $\mathbf{u}_{k+1}\leftarrow$ output of actor rule base
9: $\boldsymbol{\theta}_{k+1}^a\leftarrow\boldsymbol{\theta}_k^a+$
$\alpha_a\nabla_{\mathbf{x}}V(\mathbf{x}_{k+1}-\mathbf{x}_r,\mathbf{x}_r)\nabla_{\mathbf{u}}\hat{f}(\mathbf{x}_k,\mathbf{u}_k)\nabla_{\boldsymbol{\theta}^a}\boldsymbol{\pi}(\mathbf{x})$
10: ***% Update process model***
11: $\boldsymbol{\theta}_{k+1}^p\leftarrow\boldsymbol{\theta}_k^p+\alpha_p(\mathbf{x}_{k+1}-\hat{\mathbf{x}}_{k+1})\nabla_{\boldsymbol{\theta}^p}\hat{f}(\mathbf{x}_k,\mathbf{u}_k)$
12: ***% Update critic***
13: **for** $\forall j\in[1,2,\ldots,K^v]$ **do**
14: $\delta_{j,k}\leftarrow r_{j,k+1}+\gamma W_j(\mathbf{x}_{k+1}-\mathbf{c}_j)-W_j(\mathbf{x}_k-\mathbf{c}_j)$
15: $\boldsymbol{\theta}_{j,k+1}^c\leftarrow\boldsymbol{\theta}_{j,k}^c+\alpha_c\delta_{j,k}\boldsymbol{\zeta}_{j,k}$
16: $\boldsymbol{\zeta}_{j,k+1}=\lambda\gamma\boldsymbol{\zeta}_{j,k}+\nabla_{\boldsymbol{\theta}_j^c}W_j(\mathbf{x}_k-\mathbf{c}_j)$
17: **end for**
18: Choose $\Delta\mathbf{u}_{k+1}\sim\mathcal{N}(0,\sigma^2)$
19: Apply $\mathbf{u}_{n,k+1}+\mathbf{u}_{k+1}+\Delta\mathbf{u}_{k+1}$
20: $k\leftarrow k+1$
21: **end loop**

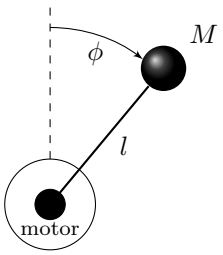

Fig. 3. 1-DOF robot arm.

$$J\ddot{\phi}=Mgl\sin(\phi)-\left(b+\frac{K^2}{R}\right)\dot{\phi}+\frac{K}{R}v \qquad (11)$$

with ϕ the arm angle measured clockwise from the upright position and v the control voltage, limited to $v\in[-10,10]$ V. The model parameters are given in Table 1.

Table 1. Robot arm parameters

Model parameter	Symbol	Value	Units
Arm inertia	J	$1.91\cdot10^{-4}$	kgm^2
Arm mass	M	$5.50\cdot10^{-2}$	kg
Arm length	l	$4.20\cdot10^{-2}$	m
Gravity acceleration	g	9.81	m/s^2
Damping	b	$3\cdot10^{-6}$	Nms
Torque constant	K	$5.36\cdot10^{-2}$	Nm/A
Rotor resistance	R	9.50	Ω

The fully measurable state $\mathbf{x}$ consists of the angle ϕ and the angular velocity $\dot{\phi}$:

$$\mathbf{x}=\begin{bmatrix}\phi\\ \dot{\phi}\end{bmatrix}$$

The reference state, $\mathbf{x}_r$, is given as;

$$\mathbf{x}_r = \begin{bmatrix} \phi_r \\ 0 \end{bmatrix}$$

with ϕ_r the reference angle. The nominal controller is a regular PD controller:

$$u_n = K_p(\phi_r - \phi) - K_d(\dot{\phi})$$

A continuous reward function $\rho(\cdot)$ is defined as follows:

$$\rho(\epsilon) = 0.1\exp(-\epsilon^2) + 0.9\exp(-\epsilon^4/0.06)$$

This function attains its maximum of 1 when the error ϵ is 0. The reward for jth individual value function is calculated by:

$$r_{k+1}^j = \rho(\phi - c_j)$$

where c_j is the constant reference angle for the jth individual value function. Additionally, the performance of the control scheme in experiments is measured by accumulating the reward

$$r_{k+1} = \rho(\phi - \phi_r)\,.$$

The RL controller proposed is tested in a *learning experiment* consisting of 400 consecutive trials, each trial lasting 1.8 seconds. The trials start in the downward position with zero angular velocity: $\mathbf{x}_0 = [\pi\ 0]^T$.

In defining the fuzzy approximators, we used the following prior knowledge. As the gravity-induced nonlinearity only depends on the angle (11), the actor has a single input, the angle ϕ. Equidistant Gaussian membership functions are used and the fuzzy system is of the singleton type (constant consequent parameters). Similarly, the process model also uses equidistant membership functions defined for the angle ϕ only. However, this fuzzy system is of the TS type, using local linear consequent models to capture the second-order process dynamics. Finally, the critic rule bases are of the TS type with the full state $\mathbf{x}$ as its input and use Gaussian membership functions. The individual critic rule bases are aggregated using Gaussian membership functions depending on the reference angle, see Fig. 2.

The membership functions at the edge of the domains have their centers at $\pm\pi$ for the angle ϕ, 0 and π for the reference ϕ_r and $\pm 8\pi$ for the angular velocity $\dot{\phi}$. The membership functions parameters are presented in Table 2 in the following format:

$$[N_\phi, N_{\dot{\phi}}, N_{\phi_r}]\,.$$

Since all membership functions are Gaussian functions

$$e^{-(x-s_1)^2/2s_2}$$

the width parameter is given as s_2 in Table 2.

Figure 4 shows a comparison between the mean learning curves of the RL control scheme and of the PD controller for random reference angles ϕ_r, averaged over 10 learning experiments. Note that the control performance significantly improves with the RL disturbance compensator compared to the plain PD controller and that the improvement is on average monotonic with time. This is a very favorable property with respect to potential practical applications of this method.

Figure 5 shows the closed-loop step responses for three different setpoints. While the PD control response results in steady-state errors (due to the gravity load disturbance) the RL disturbance compensator is able to almost completely eliminate the error.

Table 2. Parameters used in the experiments

RL parameters		
sample period	T_s	0.03
reward discount rate	γ	0.97
eligibility discount rate	λ	0.65
exploration variance	σ^2	9
Linear Controller		
Proportional Gain	K_p	5
Derivative Gain	K_d	0.5
Process model parameters		
learning rate	α_p	0.008
width of membership functions	s_2^p	0.1
number of membership functions	N^p	[9,– ,–]
Critic parameters		
learning rate	α_c	0.08
width of membership functions (ϕ)	$s_2^{c\phi}$	0.1
width of membership functions (ϕ_r)	$s_2^{c\phi_r}$	0.08
width of membership functions ($\dot{\phi}$)	$s_2^{c\dot{\phi}}$	10
number of membership functions	N^c	[9,11,5]
Actor parameters		
learning rate	α_a	0.008
width of membership functions	s_2^a	0.1
number of membership functions	N^a	[9,– ,–]

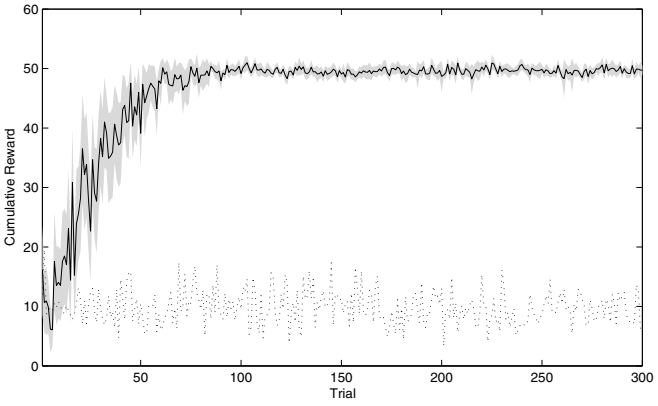

Fig. 4. Comparison of the average cumulative reward collected per trial for the PD controller (dashed line) and the fuzzy RL controller (solid line) for random references. The results are the mean of 10 experiments with the gray area showing the 95% confidence interval.

5. CONCLUSIONS AND FUTURE WORK

This paper introduced a new control method combining a nominal linear controller with a nonlinear disturbance compensator tuned by means of actor-critic reinforcement learning. The method is suitable for compensating nonlinear disturbances in reference tracking tasks with arbitrary asymptotically constant references. We have verified empirically that the closed loop control performance improves monotonically from the baseline performance of the nominal controller and throughout the learning experiment never becomes worse than the nominal performance. In simulation experiments with a 1-DOF robot arm the convergence times were satisfactory – the optimal performance was achieved in 50 to 100 trials.

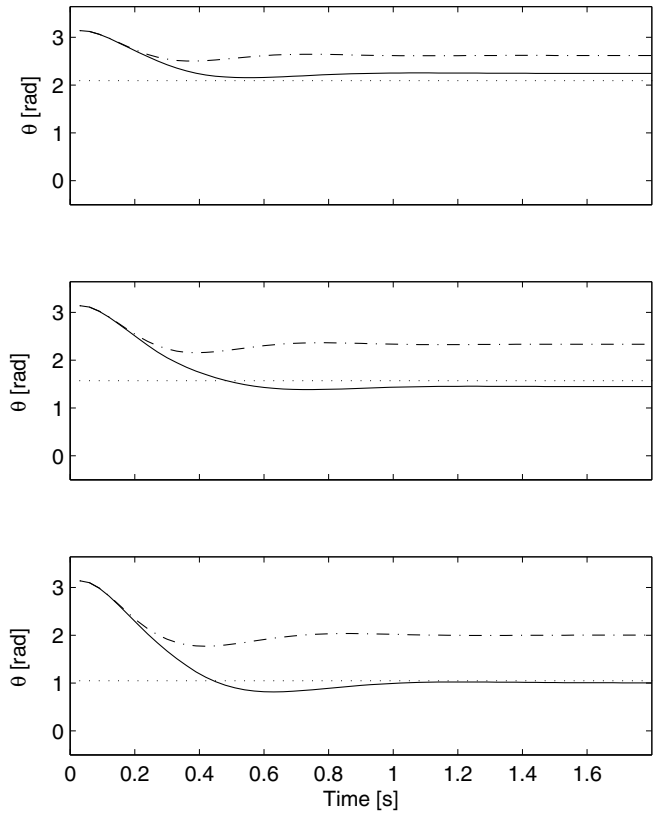

Fig. 5. Closed-loop step responses. The dotted line shows the reference angle, the dashed-dotted line the PD control response and the solid line the fuzzy RL response.

The rule-base structure of the fuzzy approximators employed in the learning scheme makes it easy to implement a priori information about the process. The approach is therefore not completely black box and the learning task can easily be simplified, for instance, by providing information on what variables are responsible for the nonlinear behavior.

Proper tuning of the learning parameters remains a challenge in RL and also in this research we do not have any guarantees that these parameters we chosen optimally. This remains a subject for future research.

REFERENCES

Abonyi, J., Babuška, R., Verbruggen, H., and Szeifert, F. (2000). Incorporating prior knowledge in fuzzy model identification. *International Journal of Systems Science*, 31(5), 657–667.

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

Brujeni, L., Lee, J.M., and Shah, S. (2010). Dynamic tuning of PI-controllers based on model-free reinforcement learning methods. In *Control Automation and Systems (ICCAS), 2010 International Conference on*, 453 –458.

Grondman, I., Vaandrager, M., Buşoniu, L., Babuška, R., and Schuitema, E. (2012). Efficient model learning methods for actor–critic control. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 42(3), 591–602.

Kiumarsi-Khomartash, B., Lewis, F.L., Naghibi-Sistani, M.B., and Karimpour, A. (2013). Optimal tracking control for linear discrete-time systems using reinforcement learning. In *Proceedings 52nd IEEE Annual Conference on Decision and Control (CDC)*, 3845–3850. Firenze, Italy.

Modares, H. and Lewis, F.L. (2013). Online solution to the linear quadratic tracking problem of continuous-time systems using reinforcement learning. In *Proceedings 52nd IEEE Annual Conference on Decision and Control (CDC)*, 3851–3856. Firenze, Italy.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control. 15(1), 116–132.