# Tuning Scaling Factors of Fuzzy Logic Controllers via Reinforcement Learning Policy Gradient Algorithms

Vahid Tavakol Aghaei
Sabanci university
Orhanli, Tuzla
Istanbul, Turkey
tavakolaghaei@sabanciuniv.edu

Ahmet Onat
Sabanci university
Orhanli, Tuzla
Istanbul, Turkey
onat@sabanciuniv.edu

## ABSTRACT
In this study a gain scheduling method for the scaling factors of the input variables to the fuzzy logic controller by means of policy gradient reinforcement learning algorithms has been proposed. The motivation for using PG algorithms is that they can scale RL problems into continuous high dimensional state-action spaces without the need for function approximation methods. Without incorporating any a-priori knowledge of the plant, the proposed method optimizes the cost function of the learning algorithm and tries to find optimal solutions for the scaling factors of the fuzzy logic controller. To show the effectiveness of the proposed method it has been applied to a PD type fuzzy controller along with a nonlinear model of an inverted pendulum. By performing different simulations, it is observed that the proposed method can find optimal solutions within a small number of learning iterations.

## CCS Concepts
• **Computing Methodologies → Policy Iteration**.

## Keywords
Reinforcement learning; policy gradients; fuzzy logic; fuzzy control; tuning scaling factors

## 1. INTRODUCTION
Reinforcement learning (RL) is among the most popular research topics in the field of machine learning and optimal control. Among the available RL methods policy gradient (PG) RL algorithms have attracted the most attention in the RL domain. [1, 2] can be considered among the first researches which used PG methods. These methods then have been utilized in different control and complex robotic problems such as [3, 4, 5].

This research focuses on policy search (PS) methods which usually work with a parameterized policy. Parameterized polices are beneficial since they scale RL problems into high-dimensional continuous state-action spaces. Several types of PS algorithms have been proposed and applied to real world systems such as studies carried out in [6, 7, 8, 9, 10].

In this work, we apply a model free PS method which uses stochastic trajectory generation via sampling from a real robot simulations without the need of a system model. This paper gives a general insight on the PG algorithms described by Peters [9] and extends the notion to fuzzy logic controllers (FLC).

Following the first fuzzy control application carried out by Mamdani [11] fuzzy control has become an alternative to conventional control algorithms to cope with complex processes and combine the advantages of classical controllers and human operator experience. The most common types of FLCs are Proportional Integral Derivative (PID) ones. Besides the existing classical gain scheduling methods, other types of tuning approaches can be found for both classical and FLCs such as fuzzy supervisors [12], genetic algorithms [13, 14] and the ant colony algorithms [15, 16].

To the best of the authors knowledge the possibility of applying PG RL methods in the FLC domain appears to be largely unexplored so far. Even though that there have been some attempts to make use of RL algorithms in the field of either parameter tuning of FLCs such as [17, 18, 19, 20] or extension of some value iteration based RL algorithms such as Q-learning in fuzzy environments [21, 22, 23]. We employ PG methods to tune the parameters of the FLCs which are beneficial because value function methods require filling the complete state-action space with data which turns out to be a very challenging problem in high-dimensional state-action spaces. This paper devotes its concentration to the subject of tuning the scaling factors of the FLCs by means of PG RL algorithms. Without loss of generality the proposed method minimizes the cost function of the RL algorithm during the learning process, which assesses the quality of the step response of a closed loop system consisting of a fuzzy controller and a nonlinear plant of an inverted pendulum. It is observed that without including any a-priori knowledge to the plant it can tune the scaling parameters of the FLC in a relatively small number of iterations and the resulting closed loop time response meets the desired specifications. The remainder of this paper is organized as follows: Section 2, briefly presents the fuzzy system. In section 3, the concept of RL and some PG algorithms are presented. In section 4, the proposed method is discussed and the simulation results will be incorporated in section 5. Finally, Section 6 gives concluding remarks and perspectives.

*ICMRE'17*, February 8–12, 2017, Paris, France.
Copyright 2017 ACM 978-1-4503-5280-2/17/02…$15.00.
DOI: http://dx.doi.org/10.1145/3068796.3068827

## 2. FUZZY SYSTEM AND CONTROL

The FLC is described by specifically determining the output for a given number of different input signal combinations. Each input signal combination is represented as a rule of the following form which defines how to best control the plant:

$$\text{If } x_1 \text{ is } A_1 \text{ ... and } x_n \text{ is } A_n \text{ then } O \text{ is } B\,. \tag{1}$$

where $x_i$, are crisp inputs, $A_i$ are fuzzy sets and "$O$" is output placed at center "$B$". Each rule has a firing strength (matching degree) which determines its applicability as:

$$\mu_k = \mu_{A_1} \times \ldots \times \mu_{A_n}\,. \tag{2}$$

where $\mu_k$ is the matching degree of the $k^{th}$ rule. We say that a rule is "on at time t" if its $\mu_k > 0$. Hence, the inference mechanism seeks to determine which rules are on to find out which rules are relevant to the current situation. Consider a FLC which its rule base has two inputs, the error "e", and the error change (derivative) "de", and one output, the control signal "u". In order to establish the structure of the FLC, for the inputs some fuzzy sets which can be Triangular, Trapezoidal or Gaussian membership functions (MF) can be selected with corresponding linguistic variables. (N as "Negative", Z as "Zero", P as "Positive").

Output can also be represented with either fuzzy sets or singletons. An example is shown in Figure 1. Defuzzification methods such as the center of gravity or weighted mean methods are used to obtain a crisp output.

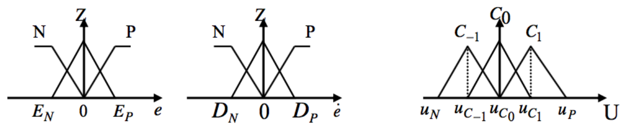

**Figure 1. Input output membership functions.**

where $e$ and $\dot{e}$ are the universe of discourses of the input and U is the universe of discourse of the output MF. It has been shown by Qiao [24] that for fuzzy controllers with product-sum inference method, center of gravity defuzzification method and triangular uniformly distributed MFs for the inputs and a crisp output, the relation between the input and the output variables of the FLC can be given by:

$$U = A + Pk_e e + Dk_d \dot{e}\,. \tag{3}$$

here $k_e$ and $k_d$ are scaling factors for error and change of error, respectively.

## 3. RL POLICY GRADIENT ALGORITHMS

### 3.1 Reinforcement Learning Formulation

A Markov decision process (MDP) can be defined by the tuple $(S, A, P, P(S_0), r)$ where $S$ is a set of $d$-dimensional continuous states, $A$ is a set of continuous actions, $P$ is the probabilistic transition function from current state $s_t$ to next state $s_{t+1}$ after taking action $a_t$ according to the density distribution $P(s_{t+1} \mid s_t, a_t)$. $P(s_0)$ is the probability of taking an initial state, $r(s_t, a_t, s_{t+1})$ is an immediate scalar reward for transition from $s_t$ to $s_{t+1}$ by taking action $a_t$. Let control policy be a stochastic parameterized policy denoted by $\pi(a \mid s, \theta)$ with $\theta \in \mathbb{R}^K$. The states and actions constitute a trajectory $\tau = [s_0, a_0, \ldots, s_T, a_T]$ with length $T$ which is also called a path, or rollout. Then one can judge the performance of a trajectory by discounted sum of future rewards which is called return of a path with a discount factor $\gamma \in (0,1]$:

$$R(\tau) = \sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t, s_{t+1})\,. \tag{4}$$

The objective of policy optimization in RL is to seek optimal policy parameters $\theta$ that optimizes the expected return:

$$J(\theta) = E[R(\tau)] = \int p(\tau \mid \theta) R(\tau) d\tau\,. \tag{5}$$

where the trajectory has the following distribution:

$$p(\tau \mid \theta) = p(s_0) \prod_{t=1}^{T} P(s_{t+1} \mid s_t, a_t) \pi(a_t \mid s_t, \theta)\,. \tag{6}$$

Typically, PG methods use the steepest ascent rule to update their parameters:

$$\theta_{h+1} = \theta_h + \alpha \nabla_\theta J(\theta)\,. \tag{7}$$

where $\alpha$ denotes a learning rate and $h$ is the number of update iterations. The main challenge in PG methods is to introduce approaches to produce a good estimate of gradient $\nabla_\theta J(\theta)$. Relevant algorithms will be briefly described in the following subsections.

### 3.2 Likelihood Ratio Policy Gradients

The REINFORCE algorithm introduced by Williams [25] has been deduced from the Likelihood-ratio methods:

$$\nabla_\theta J(\theta) = \int \nabla_\theta p(\tau \mid \theta) R(\tau) d\tau\,. \tag{8}$$

By using (6) as well as the likelihood "trick" which is represented as:

$$\nabla_\theta p(\tau \mid \theta) = p(\tau \mid \theta) \nabla_\theta \log\, p(\tau \mid \theta)\,. \tag{9}$$

the term $\nabla_\theta J(\theta)$ then can be written in the form of:

$$\nabla_\theta J(\theta) = \int p(\tau \mid \theta) \sum_{t=1}^{T} \nabla_\theta \log \pi(a_t \mid s_t, \theta) R(\tau) d\tau\,. \tag{10}$$

On account of lack of information about the trajectory distributions $p(\tau \mid \theta)$ the expectation is approximated by taking the average over whole trajectories:

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi(a_t{}^n \mid s_t{}^n, \theta) R(\tau^n)\,. \quad (11)$$

where $N$ is the total number of rollouts of length $T$ . Since the evaluation of the parameter $\theta$ is performed by Monte Carlo estimates, the resulting gradient estimates typically suffer from high variance. Without loss of generality, the resulting variance can be reduced by introducing a baseline $b \in \mathbb{R}$ for the trajectory reward as:

$$\nabla_\theta J(\theta) = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T}\nabla_\theta \log \pi(a_t{}^n \mid s_t{}^n, \theta)(R(\tau^n) - b)\,. \quad (12)$$

Since baseline can be chosen arbitrarily according to [26], it is selected to minimize the variance of the gradient estimate.

## 3.3 GPOMDP Algorithm

From (11) it is observed that REINFORCE uses the returns of whole episode to assess a single action performance. Due to the relatively large variance of the returns regarding trajectory length, the efficiency of the algorithm can get worse even by using the optimal baseline. According to this fact, a modified version of the REINFORCE algorithm namely called G(PO)MDP has been proposed by Baxter [26, 27]. Bearing the idea that instead of using the returns of whole episodes it would be better to incorporate the rewards of each individual time step in calculations of the optimal baseline and gradient which reveals the fact that past rewards do not depend on future actions.

## 3.4 Natural Policy Gradients

Natural gradient methods introduced by [28, 29] have evolved into several PG learning algorithms such as the Natural Actor-Critic algorithms (NAC) and episodic Natural Actor-critic (eNAC) which does not need complex parameterized baseline [4]. The basic idea behind this type of algorithms is that the information about the policy parameters $\theta$ contained in the observed paths $\tau$ is given by the Fisher information $F(\theta)$ defined as:

$$F(\theta) = E\{\nabla_\theta \log p(\tau \mid \theta) \nabla_\theta \log p(\tau \mid \theta)^T\}\,. \quad (13)$$

This definition of the Fisher information reveals that it is equivalent to the variance of the path derivatives. If we deviate the policy by a sufficiently small amount of $\delta\theta$, an information loss will occur which can be seen as the size of the deviation in path distribution. Therefore, searching for the policy change $\delta\theta$ which maximizes the expected return $J(\theta + \delta\theta)$ for a constant information loss, is seeking for the highest values around $\theta$ and go in the direction of these highest values.

## 4. TUNING SCALING FACTORS OF FLC VIA PG ALGORITHMS

In this work, we intend to employ the PG methods that we briefly described in the previous sections to tune the scaling factors of a FLC and investigate their effectiveness. For this purpose, a PD type FLC with a constant structure for its input-output MFs has been considered. This FLC controls a nonlinear plant with continuous state space representation. The procedure constitutes running the FLC for a specified period and collecting the relevant data regarding state transitions of the plant, control signal and reward. This process continues until a predetermined number of episodes is reached. Then REINFORCE, GPOMDP or eNAC used to calculate the incremental value that is needed to update the scaling factors of the FLC. This procedure is illustrated schematically in Fig. 2.

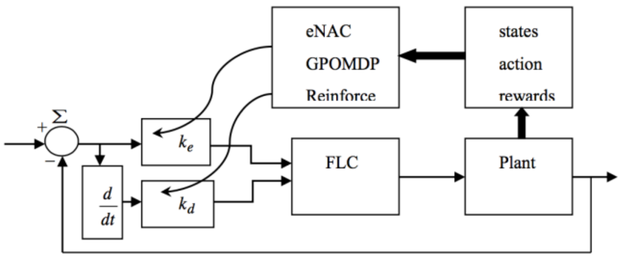

**Figure 2. Schematic of the proposed tuning mechanism.**

One typical symmetrical rule base that can be used for most of the FLC rule bases is summarized in Table.1

**Table 1. A typical symmetrical rule base of a FLC.**

| **error** | **error derivative** | | |
|---|---|---|---|
| | | | |
| | **N** | **Z** | **P** |
| **N** | $C_1$ | $C_1$ | $C_0$ |
| **Z** | $C_1$ | $C_0$ | $C_{-1}$ |
| **P** | $C_0$ | $C_{-1}$ | $C_{-1}$ |

The output of the PD type FLC is:

$$U = A + Pk_e e + Dk_d \dot{e} + \varepsilon\,. \quad (14)$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian distribution with zero mean and standard deviation $\sigma$ . The goal is to optimize the parameter vector $[k_e, k_d]$ so we need a parameterized policy that can model the action generation procedure given the parameters and states as (14). By considering (14) it can be observed that we can take $[k_e, k_d]$ as the parameters of the policy. On the other hand, $[e, \dot{e}]^T$ is the state vector of the plant. By considering these facts, the model that best suits for our objective and can be considered as an equivalent model to (14), is a Gaussian policy whose parameter vector is $\theta = [\mu, \sigma]$ where $\mu$ is the mean vector and σ is its standard deviation. Then the corresponding parameterized policy would be:

$$\pi(a \mid s, \theta) = \frac{1}{\sigma\sqrt{2pi}} \exp(-\frac{(a - \mu s)^2}{2\sigma^2})\,. \quad (15)$$

Here $s$ and $a$ are the continuous state and action, respectively, where for this problem the action is actually the control signal. Now we can relate $[k_e, k_d]$ to the mean vector $\mu$ of the parameterized policy $\pi(a\,|\,s,\theta)$. Once the parameterized policy is determined, as discussed in previous sections it is required to calculate the gradient of the logarithm of the parameterized policy with respect to its parameters and use (7) to update them. This gradient can be calculated as:

$$\nabla_{\mu}\log\,\pi(a\,|\,s,\theta)=\frac{a-\mu^T s}{\sigma^2}s\,. \qquad (16)$$

$$\nabla_{\sigma}\log\pi(a\,|\,s,\theta)=\frac{(a-\mu^T s)^2-\sigma^2}{\sigma^3}\,. \qquad (17)$$

## 5. SIMULATIONS AND RESULTS

The plant to be controlled under a PD type FLC is a nonlinear inverted pendulum with two continuous states consisting of the angle and angular velocity of the pendulum i.e., $[\omega,\dot{\omega}]$. The system dynamics of the pendulum are given in [30] which are defined as following:

$$\ddot{\omega}=\frac{-3ml\dot{\omega}^2\sin\omega\cos\omega+6(M+m)g\sin\omega-6u+6b\dot{\omega}\cos\omega}{4l(M+m)-3ml\cos\omega}$$

$$(18)$$

$$\ddot{x}=\frac{-2ml\dot{\omega}^2\sin\omega+3mg\sin\omega\cos\omega+4u-4b\dot{\omega}}{4(M+m)-3m\cos\omega}\,.$$

where $g=9.8(ms^{-2})$, friction coefficient $b=0.1N(ms)^{-1}$, length of the pole $l=0.6(m)$, mass of the cart $M=0.5(kg)$, mass of the pole $m=0.5(kg)$. The control objective is to stabilize the pendulum at the upright position. The inputs to the fuzzy controller are error and change of error whose corresponding fuzzy sets are taken as three equidistant triangular MFs. The output of the FLC which is control signal is defined with symmetrical triangular MFs, as well. Input-output MFs are illustrated in Fig. 1.

Here we take the universe of discourses of the input MFs to be $E_n=\frac{-pi}{2}$, $E_p=\frac{pi}{2}$, $D_n=\frac{-pi}{4}$ and $D_p=\frac{pi}{4}$. The parameters of the output MFs are taken as $u_n=-20$, $u_{C-1}=-10$, $u_{C0}=0$, $u_{C1}=10$ and $u_p=20$. The deffuzzified output obeys the center of gravity method producing a crisp control signal as:

$$\frac{\sum_{i=1}^{M} b_i\int\mu_i}{\sum_{i=1}^{M}\int\mu_i}\,. \qquad (19)$$

with totally $M$ rules and $b_i$ is the center of the MF of the consequent of the $i^{th}$ rule.

During the simulations two reward functions introduced namely called as "Interval based" and "Absolute value based" rewards which are written in the following form:

$$\begin{cases} if\ -8(\deg)\le\omega\le 8(\deg); & r=0 \\ otherwise; & r=-10 \end{cases}\,. \qquad (20)$$

$$r=-w_{\omega}\,|\,\omega_d-\omega\,|-w_{\dot{\omega}}\,|\,\dot{\omega}_d-\dot{\omega}\,|-w_u\,|\,u\,|\,. \qquad (21)$$

Here terms $\omega_d$ and $\dot{\omega}_d$ stand for the desired values of pendulum angle and its angular velocity and $w_{\omega}$, $w_{\dot{\omega}}$ and $w_u$ are the weights on pendulum angle, angular velocity and force applied to the cart, respectively. These values are taken as $w_{\omega}=3$, $w_{\dot{\omega}}=0.85$ and $w_u=0.1$. Consider that in (21) if pendulum leaves its accepted vicinity the simulation will be stopped and it will receive a reward value of $-1000$. During the experiments discount factor is $\gamma=0.9$. For simulating the nonlinear plant using MATLAB, "Fourth-Order Runge-Kutta" method has been used.

In this study for the sake of simplicity, the standard deviation for the parameterized policy was assumed to be fixed $\sigma=2$ therefore in calculating the gradient of the logarithm of the policy only equation (16) was considered. In all experiments during the learning, number of episodes are $N=100$ and in each individual episode the inverted pendulum runs for $T=1000$ time steps with a sampling time of $0.01(\sec)$. Performance of the algorithms are tested after every $100$ episodes with starting from random initial states for the pendulum's angle between $-8(\deg)$ and $8(\deg)$.

For testing process, the Gaussian noise of the policy set to 0. It is worth mentioning that this problem is challenging since it starts to learn without any a-priori knowledge of the system i.e., both parameters of the scaling factors of the FPD controller are set to zero. To show the performance of the individual algorithms we averaged each experiment over 20 times. In the experiments REINFORCE method could only find optimal solution for the parameters with "Absolute value reward" and when $\sigma=0.001$ therefore we just incorporated time response plot for REINFORCE algorithm. The resulting figures for the eNAC and GPOMDP performances are depicted in Fig. 3 with a confidence interval representation. Note that average rewards are normalized in the interval $[0,1]$ to have a fair comparison between both reward structures.

As can be seen from Fig. 3 in case of comparing eNAC and GOPMDP, GPOMDP converged in less iterations than eNAC. GPOMDP exhibited a very similar convergence performance in both types of rewards. If we also consider the performance of the algorithms in case of the time response, we notice that an early convergence of the GPOMDP algorithm is due to getting stuck in a local optimum whereas eNAC with "Absolute reward" struggled to search an optimal solution and ended up with a satisfactory solution with a relatively high standard deviation of the average return and converged in more iterations in comparison with others.

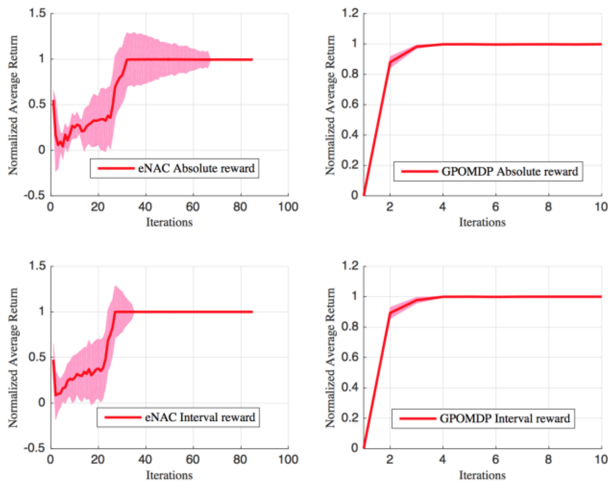

**Figure 3. Performance of eNAC and GPOMDP algorithms.**

In Fig. 4 and Fig. 5 time responses of the corresponding algorithms with both types of RL rewards have been illustrated. From the figures, it is apparent that eNAC algorithm with the "Absolute value reward" structure outperforms its counterpart in closed loop specifications by inheriting satisfactory settling time and less overshoot in its response. eNAC with "Interval based reward" performs better in case of settling time but it expresses overshoot in its responses. In the case of GPOMDP, "Absolute value based reward" displayed a better settling time by almost three times less than that of "Interval based reward".

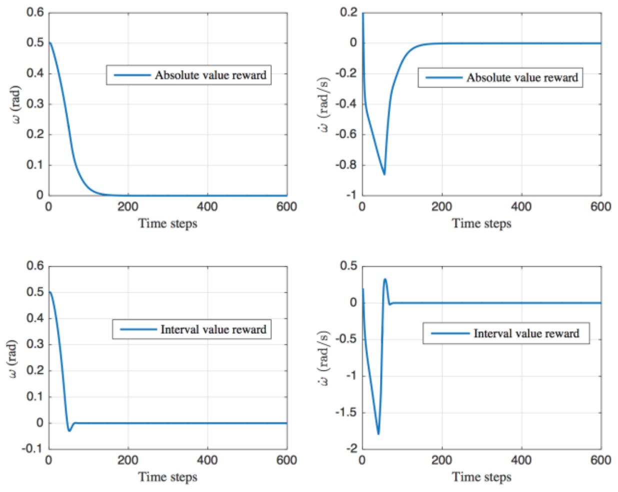

**Figure 4. Time responses of eNAC algorithm**

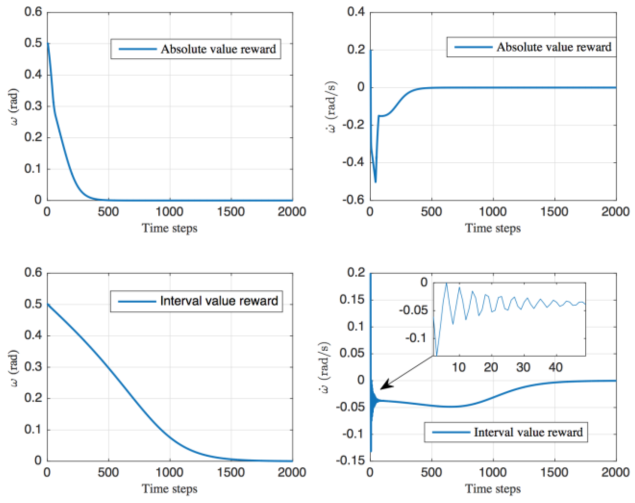

**Figure 5. Time responses of GPOMDP algorithm.**

In Fig. 6 time responses of the REINFORCE algorithm based on the "Absolute value reward" is depicted and it is obvious that the settling time is larger than that of related to eNAC and GPOMDP.

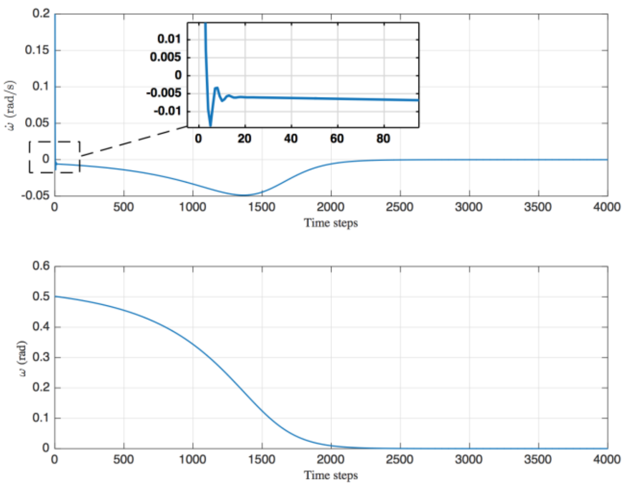

**Figure 6. Time responses of REINFORCE algorithm.**

## 6. CONCLUSIONS AND FUTURE WORK

In this research, we utilized PG RL algorithms: REINFORCE, GPOMDP and eNAC methods to tune the scaling factors of the FLCs. To show the effectiveness of the proposed method we applied it to a nonlinear inverted pendulum model which is being controlled by a fuzzy PD controller with two scaling factors for error and change of error. For the reward function of the RL algorithms we described two structures namely called here as "Interval based" and "Absolute value based" rewards.

By investigating different simulations, we found out that for this problem eNAC and GPOMDP algorithms can find optimal values for FPD scaling factors with a reasonable time response specifications while REINFORCE showed a weak performance.

To improve the performance of the FLC system, it is important to realize that the scaling factors are not the only parameters that can be tuned. Indeed, sometimes it is the case that for a given rule-base and MFs you cannot achieve the desired performance by tuning only the scaling factors. Often, what is needed is a more careful consideration of how to specify additional rules or better MFs. In future studies, we will strive to apply PG methods to modify the universe of discourses of the input-output MFs.

## 7. REFERENCES

[1] Gullapalli. V, J. Franklin, and H. Benbrahim, “Aquiring robot skills via reinforcement learning,” *IEEE Control Systems*, vol. -, no. 39, 1994.

[2] Benbrahim. H and J. Franklin, “Biped dynamic walking using reinforcement learning,” Robotics and Autonomous Systems, vol. 22, pp. 283–302, 1997.

[3] Kimura. H and S. Kobayashi, “Reinforcement learning for continuous action using stochastic gradient ascent,” in *the 5th International Conference on Intelligent Autonomous Systems*, 1998.

[4] Peters. J, S. Vijayakumar, and S. Schaal, “Natural actor-critic,” in *Proceedings of the European Machine Learning Conference (ECML)*, 2005.

[5] Mitsunaga. N, C. Smith, T. Kanda, H. Ishiguro, and N. Hagita, “Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, 2005, pp. 1594–1601.

[6] Daniel. C, G. Neumann, and J. Peters, “Hierarchical relative entropy policy search,” in *Proceedings of the International Conference of Artificial Intelligence and Statistics*, (N. Lawrence and M. Girolami, eds.) pp. 273–281.2012.

[7] Deisenroth. M.P, C. E. Rasmussen, and D. Fox, “Learning to control a low-cost manipulator using data-efficient reinforcement learning,” in *Proceedings of the International Conference on Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[8] Kober. J and J. Peters, “Policy search for motor primitives in robotics,” *Machine Learning*, pp. 1–33, 2010.

[9] Peters. J and S. Schaal, “Policy gradient methods for robotics,” in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robotics Systems*, pp. 2219–2225, Beijing, China, 2006.

[10] Vlassis. N, M. Toussaint, G. Kontes, and S. Piperidis, “Learning model-free robot control by a Monte Carlo EM algorithm,” *Autonomous Robots*, vol. 27, no. 2, pp. 123–130, 2009.

[11] Mamdani, Ebrahim H. "Application of fuzzy algorithms for control of simple dynamic plant." *Proceedings of the Institution of Electrical Engineers.* Vol. 121. No. 12. IET Digital Library, 1974.

[12] Zheng L, Yamatake-Honeywell Co. A practical guide to tune of proportional and integral (PI) like fuzzy controllers. In: *Proc. IEEE international conference on fuzzy systems.* 1992, p. 633–40.

[13] Adams. JM, Rattan KS. A genetic multi-stage fuzzy PID controller with a fuzzy switch: In*: Proc. IEEE Int. Conference On Systems, Man, And Cybernetics*, vol. 4. 2001. p. 2239–44.

[14] Ko C-N, Lee T-L, Fan H-T, Wu C-J. Genetic auto-tuning and rule reduction of fuzzy PID controllers. *In: Proc. IEEE International Conference On Systems, Man, And Cybernetics*. 2006. p. 1096–101.

[15] Duan. H.B, D.-B Wang, X.-F. Yu, Novel approach to nonlinear PID parameter optimization using ant colony optimization algorithm. *Journal of Bionic Engineering*, 3 (2006), pp. 73–78.

[16] Varol HA, Bingul Z. A new PID tuning technique using ant algorithm. *In: Proc. American Control Conference*. 2004. p. 2154–9.

[17] Berenji. H.R and P. Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on Neural Networks*, 3(5), 1992.

[18] Rezine. H, Louali Rabah, J èrome Faucher and Pascal Maussion "An Approach to Tune PID Fuzzy Logic Controllers Based on Reinforcement Learning" , *Automation and Control, Book* ISBN 978-953-7619-18-3, pp. 494, October 2008, I-Tech, Vienna, Austria.

[19] Boubertakh. H , Mohamed Tadjine Pierre-Yves Glorennec Salim Labiod Tuning fuzzy PD and PI controllers using reinforcement learning, *ISA Transactions* Volume 49, Issue 4, October 2010, Pages 543–551.

[20] Tavakol Aghaei. V, Ahmet Onat, Ibrahim Eksin, and Mujde Guzelkaya "Fuzzy PID Controller Design Using Q-Learning Algorithm with a Manipulated Reward Function" *2015 European Control Conference (ECC)* July 15-17, 2015. Linz, Austria.

[21] Berenji. H.R. Fuzzy (&Learning: A new approach for Fuzzy Dynamic Programming problems. *In Third IEEE, International conference on Fuzzy Systems*, Orlando, FL, June 1994.

[22] Glorennec. P.Y and J. Jouffe, “Fuzzy Q-learning,” *in Proc. 6th IEEE Int Conf. Fuzzy Systems*, 1997.

[23] Busoniu, Lucian, Robert Babuska, Bart De Schutter and Damien Ernst. Reinforcement Learning and Dynamic Programming Using Function Approximators. CRC Press, Taylor and Francis Group, 2010.

[24] Qiao. W.Zh , Masaharu Mizumoto, PID type fuzzy controller and parameters adaptive method, *Fuzzy Sets and Systems*, Volume 78, Issue 1, 26 February 1996.

[25] Williams R.J, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, vol. 8, pp. 229–256, 1992.

[26] Baxter.J and P. Bartlett, “Direct gradient-based reinforcement learning: gradient estimation algorithms,” *Technical report.* 1999.

[27] Baxter.J and P. L. Bartlett, “Infinite-horizon policy-gradient estimation,” *Journal of Artificial Intelligence Research,* 2001.

[28] Amari. S. Natural gradient works efficiently in learning. *Neural Computation*, 10, 1998.

[29] Kakade. S.A. Natural policy gradient. *Advances in Neural Information Processing Systems* 14, 2002.

[30] Dann. Ch, Gerhard Neumann, Jan Peters Policy Evaluation with Temporal Differences: A Survey and Comparison *Journal of Machine Learning Research* 15 (2014) 809-88