DINCON 2017
CONFERÊNCIA BRASILEIRA DE DINÂMICA, CONTROLE E APLICAÇÕES
30 de outubro a 01 de novembro de 2017 – São José do Rio Preto/SP

Control of a Fourth-Order Fluid Level System Using Reinforcement Learning and Radial Basis Neural Networks

Lucas Guilhem de Matos[1]
Laboratório de Automação e Robótica, LARA, Brasília, DF,
Departamento de Engenharia Elétrica, UnB, Brasília, DF
Adolfo Bauchspiess[2]
Departamento de Engenharia Elétrica, UnB, Brasília, DF

Abstract. This work presents a proposal of an adaptive controller using reinforcement learning and neural networks in order to deal with nonlinearities and time-variance. To test the controller a fourth-order fluid level system was chosen because of tits high time constants and the possibility of varying the system parameters. Implementation was made on a computer connected to an Arduino as interface to the sensor and actuator. The controller is an adaptive PI with its gains defined by a Neural Network and its performance was better than the conventional PI controller used as reference and has shown adaptive features and improvement during execution. Also, the proposed controller needs no previous information of the system in order to be designed.

Key words. Reinforcement Learning, Neural Networks, Adaptive Control, Industrial Process

1 Introduction

Control is one of the core keys in the development of automation that is why developing control theory and control techniques is essential. Although control theory and techniques have stride largely since around the 18th century, there are still various causes that enhance the difficulty of the control problem such as non-linearities and time-variance. It can be seen, e.g., [4] and [6] the usefulness of adaptive control. In order to tackle the said problems, an adaptive controller based on reinforcement learning and radial basis networks will be designed based mostly on the algorithm proposed by [9] and theory by [7].

To test the efficiency of the designed controller the system developed in [1] and used in [5] was chosen because fluid level control is a very common process and used in large scale in a variety of industrial applications.

[1] lguilhem.matos@gmail.com
[2] adolfobs@unb.br

Sections 2 and 3 will provide some theoretical background, section 4 will describe the system used for experiments, section 5 will present the algorithm development and adjustment, section 6 will present the results, analysis and will close the document with conclusion and suggestions for upcoming work.

2 The Reinforcement Learning Problem

2.1 Concept and Elements

The Reinforcement Learning (RL) problem consists in an *Agent* or Individual interacting with a specific Environment with *actions* defined according to an *action policy*. The environment's response to the action is called *reward* and it defines the immediate quality of an action usually showing somehow if the agent got closer or further from its goal.

The objective of any RL problem is to achieve maximum accumulated reward, meaning that the agent is not only interested in instant high valued rewards, but in higher expected future rewards from a given state s_t, what is called the state's *value* or the *obtainable return* from that state, an estimate, defined as follows:

$$V(s_t) = \sum_{i=1}^{\infty} \gamma^i r_{t+i} \tag{1}$$

which is the Bellman Equation of the reward with γ as the discount factor. This equation defines how good it is for the agent to visit a given state in a long-term sense showing how much reward it can get from there.

The expansion and manipulation of equation (1) generates:

$$V(s_t) = r_{t+1} + \gamma \sum_{i=1}^{\infty} \gamma^i r_{t+i+1} \tag{2}$$

that gives,

$$V(s_t) = r_{t+1} + \gamma V(s_{t+1}) \tag{3}$$

which states that the sum of the discounted estimate of the next state value and the next state reward is the target to the actual estimate, as seen in reference [7].

The error between the target and the estimate of the value function is known as the Temporal Difference Error (δ_{td}), given by:

$$\delta_{td} = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{4}$$

The δ_{td} is the cost function to be minimized and it will be responsible for the updating of the value function estimator and the action policy estimator as well. This a method known as Temporal Difference Learning (TDL).

2.2 The Actor-Critic Method

The Actor-Critic is a TD method that has separated structures for the action policy, known as the Actor so it is the structure that defines which actions will be taken, that is, the control signal, and the value function estimator, known as the Critic which is the structure that evaluates the actor's decisions. In this specific case, it bootstraps the value function using the reward to generate the δ_{td}

which will be responsible for updating both structures. This procedure is depicted in Figure 1 from Reference [7].

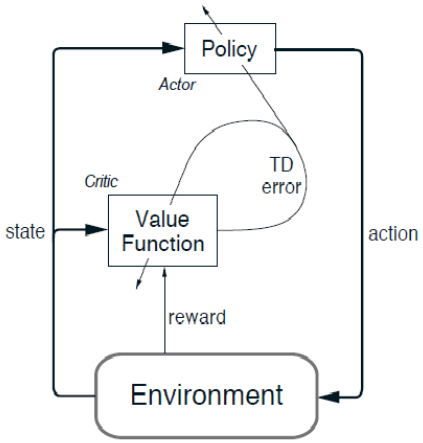

Figure 1: The Actor-Critic Structure, From reference [7].

2.3 The SARSA Algorithm

SARSA stands for 'state-action-reward-state-action' and is an TD algorithm which states that the value depends not solely on the state, but also in the chosen action to be taken from the given state, that is, the estimate is the sum of expected future rewards after taking a specific action in a given state. Instead of using the function $V(s_t)$, the Algorithm uses the function $Q(s_t, a_t)$ as the value function for the Critic. In this case, the δ_{td} defined in equation (4) becomes:

$$\delta_{td} = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \quad (5)$$

This is needed because the action taken will be also used to calculate the reward so it must be included as an input to the value function being approximated otherwise part of the reward's behavior will be uncorrelated to the function, and so, the estimation would most likely fail.

3 Radial Basis Neural Networks

Radial Basis Neural Networks (RBN) are Neural networks that use a Radial Basis Function (RBF) as the activation function of its neurons, whereRBF are functions whose value depend only on the distance from a central point while it decreases or increases monotonically with the distance from the center according to a dispersion parameter, as seen in [2].

RBFs are used to identify system's features as shown in [3], since they will respond differently to different system states showing which state they are more closely related to. The RBN is then a weighted sum of several RBF, that is, a weighted sum of several system features. The RBN classical structure is shown in Figure 2, from [2].

Since the centers of the radial basis represent specific system features, they are updated to be closer to inputs that are similar, this will be made by k-meaning clustering as seen in [2], also, the weights are also updated according to the importance of each feature using the δ_{td}.

The RBN will be used for the estimation of the action policy and the value function. This topology was chosen because of its simplicity and easiness to update.

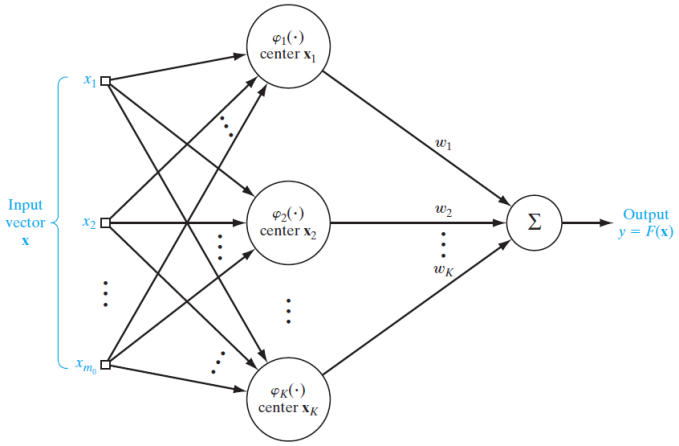

Figure 2: Radial Basis Neural Network Structure, [2].

4 Experimental Environment

To experiment on the controller's performance, the chosen system was a fourth-order fluid level system with sequenced tanks and regulable fluid passage between each pair of tank. The fluid input is located on the bottom of the first, or far left tank and the controlled level is the level of the fourth or the far right tank. The connection between 2 tanks is made by a gate valve and The level measurement is made with a pressure sensor with a tube to the bottom of the tank.

This system was built by [1] and was developed to have a range of time constants what makes the control task more challenging. Other interesting characteritics of this system is the considerable input delay, dead zones and saturation which poses challenge to the RL algorithm. The system has a non-linearity that is square root, and altough it is a weak non-linearity, also enhances the difficulty of the controll problem. Figure 3 shows the process set-up.

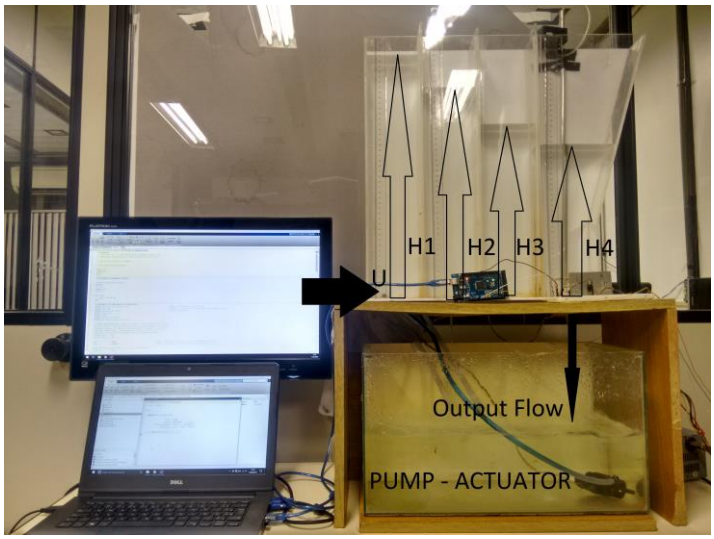

Figure 3: System used and Process Set-Up

5 Algorithm Development and Implementation

5.1 Calculating Reward in the Observed State

The reward as an immediate response to a given action, also it is given by the environment, not by the agent, so, the reward should be a function of the RL state variables. The variables that define the RL state where defined as the controlled tank level and the reference, also, the reward will address the controller output in order to avoid saturation, reward is given by:

$$\begin{gathered} r = 0 \\ if\ e(k) > \varepsilon, r = r - 0.1 \quad else\ if\ e(k) > e(k-1), r = r - 0.1 \\ if\ U(k) > 100\%\ or\ U(k) < 0\%\ r = r - 0.5 \end{gathered} \tag{7}$$

where ε is the tolerance margin, chosen as 1 cm, $e(k)$ is the error in instant k, $e(k-1)$ is the previous error signal and $U(k)$ is the control signal in % of power of the actuator.

5.2 Updating the RBN

The chosen radial basis function for the RBN neurons was the Gaussian function. The centers where updated via k-meaning clustering. The detailed explanation and mathematical development can be found in [2], chapter 5, pages 242-245. From Figure 2 it can be seen that the output of the RBN is given by:

$$y = \sum_{i=1}^{N} w_i\, \rho\left(||\boldsymbol{x} - \mu_i||\right) \tag{8}$$

N being the number of neurons, chosen to be 10^{inputs}, that means 100 for the Actor since it uses just the reference signal and the system output an 1000 to the critic, once it uses the controller's output as an input as well. ρ is the RBF and w the weights.

The weights where updated as seen in [8] and [9] using the gradient descent of the quadratic δ_{td}, resulting in the following expression for updating:

$$w_j(k+1) = w_j(k) + \eta_a \delta_{td} \rho\left(\left|\left|\boldsymbol{x} - \boldsymbol{\mu_j}\right|\right|\right) \tag{9}$$

where η_a is the learning rate.

5.3 Algorithm: Reinforcement Learning with Radial Basis Networks Control

1) Initialize s_t and all constant parameters that will be used along the process.
2) Define an action a_t using the Actor.
3) Calculate $Q(s_t, a_t)$ using the Critic.
4) Take action a_t, observe s_{t+1} and calculate the reward r_{t+1} as given in equation (7).
5) Define an action a_{t+1} using the Actor.
6) Calculate $Q(s_{t+1}, a_{t+1})$ using the Critic.
7) Calculate δ_{td} using equation (5).
8) Update both the Actor and the Critic using equations and (9) and [2], chapter 5.
9) Make $s_t = s_{t+1}$ and $a_t = a_{t+1}$
10) Repeat steps from 4 to 10

5.4 Algorithm Implementation

As seen in Figure 3, the implementation was made using a computer with MatLab® and an Arduino MEGA 2560. The algorithm was performed by the PC, only requesting and sending data through the Arduino to the process.

6 Experimental Results, Analysis and Conclusion

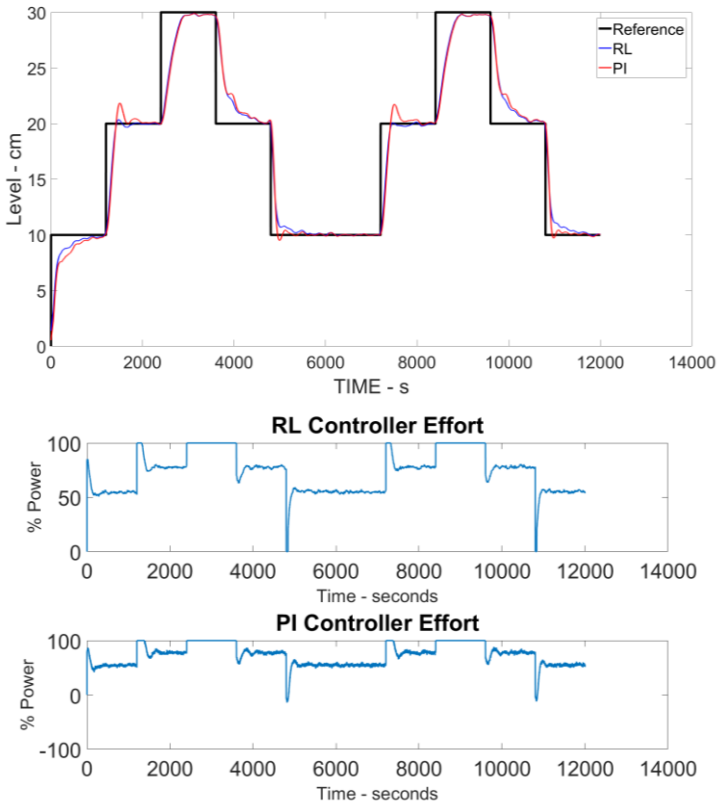

Figure 5: System Output and Control Effort

Figure 5 shows the results of the RL controller against a PI controller designed for the operating point of 20 cm for 10 min rising time (half of the open loop time) and acceptable overshoot of 2.5 cm, also, it shows the actuator's effort for the RL controlled against that of the PI.

As it can be seen, the designed controller behaves similarly to the PI in the specific operating point of 20 cm, for the 30 cm it can be seen that the system is dealing with saturation, since in both cases, the output does not reach the reference, and so, both answers are show to be almost equal. The more notable difference is to the 10 cm reference. In this point, the RL controller performs better, once it can change its gains to achieve better performance. One interesting information is that the PI, as expected,

maintains the same behavior during the second cycle, as the RL controller, also as expected, has a slightly change and improvement. This is important to observe, since it is the main goal of the controller. Regarding the control effort, it can be seen that the RL Controller has a wider variety of gains, which obviously happens because the gain update.

5.2 Conclusion and Further Work

The designed controller was able to perform better than the PI used for comparison and with no information about the controlled system and has demonstrated expected adaptive features, yet, some improvements can still be made. An idea is to use a predictor to enhance the RL algorithm and make the learning faster because since the system is very slow, some interaction represent no gain for the algorithm, also, as seen in reference [8] predicting a system model is useful to bypass delay and some noise.

References

[1] M. C. Bernardes, G. A. F. Melo, A. A. Freitas, G. A. Borges e A. Bauchspiess, Instrumentação e estimação de parâmetros de um sistema de nível de líquido com quatro tanques interligados. CBA, (2006).

[2] S. Haykin, Neural Networks and Learning Machines, Pearson, (2008).

[3] Y. He, X. Wang and J. Z. Huang, Fuzzy Nonlinear Regression Analysis Using a Random Weight Network, vol. 364-365, 222-240, (2016).

[4] C. Huang and C. Yu, *Senior Member, IEEE*, Global Adaptive Controller for Linear Systems with Unknow Input Delay, IEEE Transactions on Automatic Control, Issue 99, 1, (2017).

[5] J. C. P. Oliveira, Avaliação de Controle Neural a um Processo de Quatro Tanques Acoplados, Tese de Doutorado em Engenharia Elétrica, UnB, (2009).

[6] S. Papierok, A. Noglik and J. Pauli, Applications of Reinforcement Learning in a Real Environment Using and RBF Network, ERLARS, (2008)

[7] R. S. Sutton and A. G. Barto, Rainforcement Learning: An Introduction, MIT Press, (1998).

[8] T. Wang, H. Gao, *Fellow, IEEE,* and J. Qiu, *Senior Member, IEEE,* A Combined Adptive Neural Network and Nonlinear Model Predictive Control for Multirate Networked Industrial Process Control, IEEE Trans. on Neural Networks and Learning Systems, vol. 27, no. 2, (2016).

[9] X. Wang, Y. Cheng and W. Sun, A Proposal of Adaptive PID Controller Based on Reinforcement Learning, J. of China University of Mining & Technology, vol. 17, no 1, (2007).