Reinforcement Learning for Controlling a Coupled Tank System Based on the Scheduling of Different Controllers

Anthony A.R. Diniz, Paulo R. M. Pires, Jorge D. de Melo, and Adrião D. D. Neto
Dep. de Engenharia de Computação e Automação
Universidade Federal do Rio Grande do Norte
Natal, Brasil
e-mails: anthony_andrey@yahoo.com, prmotta@gmail.com, jdmelo@dca.ufrn.br, adriao@dca.ufrn.br

Armando J. J. L. Filho and Sérgio M. Kanazava
EMA – Departamento de Energia e Meio Ambiente
Telvent Brasil S. A.
Natal, Brasil
e-mail: armandolemos1@hotmail.com, sergiocompras@yahoo.com.br

***Abstract*—Reinforcement Learning has been an approach successfully applied for solving several problems available in literature. It is usually employed for solving complex problems, such as the ones involving systems with incomplete knowledge, time variant systems, non-linear systems, etc., but it does not mean that it cannot be applied for solving simple problems. Therefore, this paper proposes an alternative application, where RL could be applied to switch among controllers with a fixed tuning, in a system with a known non-linear dynamics, aiming to optimize its time response. It was shown, after the online training and test of the RL agent that it could take advantage of the best characteristics of the available controllers to improve the response of the coupled tank system.**

Keywords-reinforcement learning; Q-learning; intelligent control

I. INTRODUCTION

Control system design approaches deal with different kinds of information that can lead to different controllers [1]. According to [2], the RL approach based control strategies have been successfully employed for several difficult problems such as control of inverted pendulum, playing backgammon, adaptive control of the walking machine, improving elevator performance, etc.

When the process model is deterministic, linear and the control objectives are stated as either some desirable static and dynamic performance indexes or by the minimization of an objective function, a large variety of design methodologies is available [1]. On the other hand, dealing with complex systems usually involves building a simplified problem model and starts foreseeing a progressive improvement in its knowledge, trying to get a better control which, in some cases, could be directly obtained but paying a much greater effort in modeling, computation, and time wasting than with learning intelligent control [1].

This paper proposes a combined approach, where there is a simple process with a well known dynamics, but there are three controllers pre-tuned, whose parameters cannot be changed. The goal is designing an agent, applying RL, whose function is switch among these "black box" controllers to learn the optimal policy that improves the system's time response.

II. COUPLED TANK SYSTEM AND CONTROLLERS

Our problem was developed using the educational kit manufactured by the Quanser Consulting Inc., composed by two coupled tanks, supplied by a pump connected to an external reservoir, as illustrated in the Fig. 1 [3]. The pump could be controlled applying a voltage signal from -3 to +3 volts, where the negative range would do it drain water from the upper tank and the positive range would do it pump water to it.

The level of each tank was measured by a pressure sensor, positioned in the lowest part of the tank, and configured to indicate the level directly in volts. Each tank was 25cm high, but their height could be measured in a range from 0 to 4 volts.

All the programs were implemented using the MATLAB©, which is compatible with the Quanser kit, and can send instructions to control the pump and read the sensors signal.

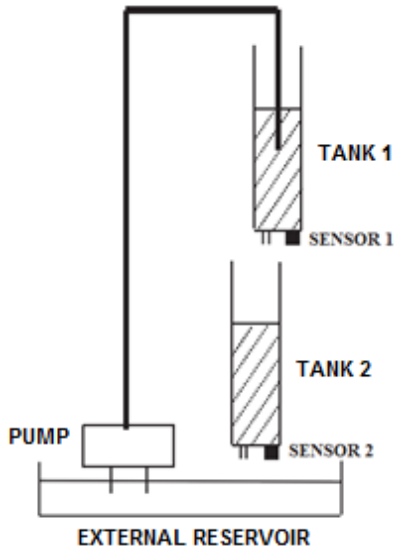

Figure 1. Schematic of the coupled tanks system

978-0-7695-4210-2/10 $26.00 © 2010 IEEE
DOI 10.1109/SBRN.2010.44

The tanks level measured during the experiment will be shown in graphs with two drawings, where the "H Tank" line represents the level of the upper tank and the "L Tank" line represents it for the lower tank. During the experiment, the setpoint was defined for the lower tank and it was introduced a voltage limit (nonlinearity) to pump liquid to the upper tank, when it was close to its highest allowed limit, as a way of preventing its overflow.

The system also comprised three controllers, whose tuning could not be modified, to the purposes of this paper, that the agent could manage (switch them). From this point, the setpoint will be considered 2 volts for the lower tank level that was chosen for all the tests. The outputs of the system connected to each controller are shown in Fig. 2.

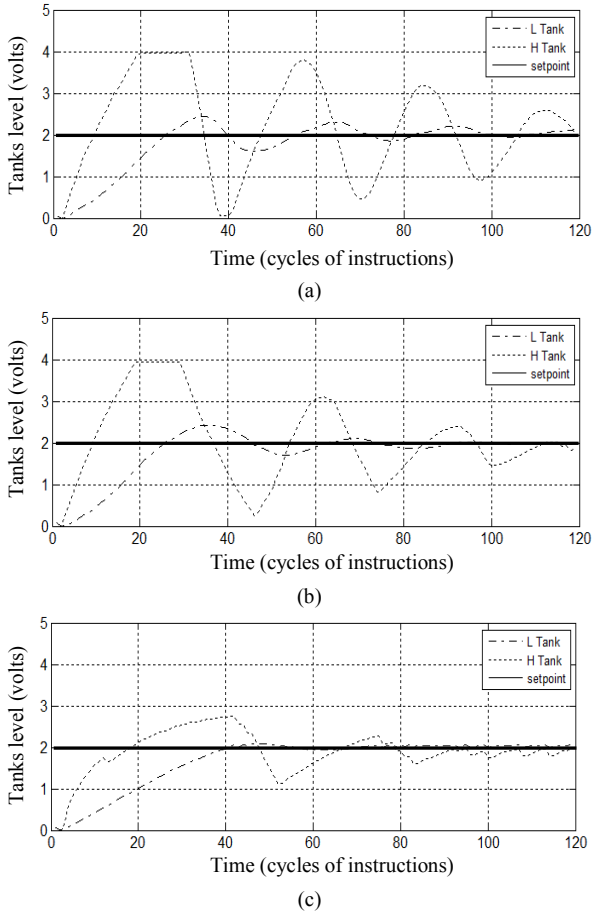

Figure 2. System response to: (a) controller 1; (b) controller 2 and (c) controller 3.

The desired system configuration is shown in Fig. 3. The goal of this paper was designing an RL agent to manage this system, switching among these three controllers, seeking for an improved time response.

When we think about the nature of learning, one of the first things that come to mind is that we learn interacting with the environment. According to this concept, [4] define reinforcement learning (RL) as learning what to do – as mapping situations to actions – in a way that maximizes a numerical reward.

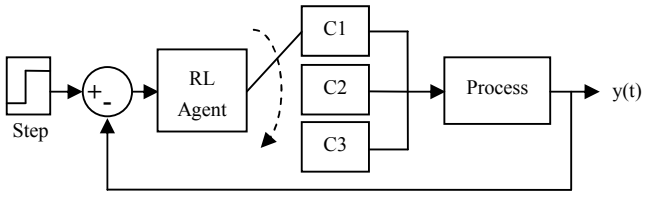

Figure 3. System configuration with the agent.

III. REINFORCEMENT LEARNING

The objective of RL is orientating an agent to take actions that take him to maximize (or minimize) the sum of all the reinforcement signals (numerical reward or punishment) received through the time, named as expected total reward, what not even results in maximize the immediate reward received [5]. The sum of all the reward signals in a finite horizon is expressed by (1) and (2).

$$R_t = r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \cdots \quad (1)$$

$$R_t = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \quad (2)$$

where R_t represents the return (sum of all reinforcements) received through the time, r_{t+k} represents the signal of immediate return and γ is the discount rate, defined within the interval $0 \leq \gamma \leq 1$ to assure that R_t be finite.

The behavior that the agent should follow to reach the maximization (or minimization) of its return function is known as "policy" and can be defined as π. According to [5], a policy $\pi(s,a)$ is a mapping from states s to actions a, taken at that state, and represents the probability of selecting each one of the possible actions, in such a way that the best actions correspond to the highest probabilities of choice.

To evaluate the quality of the actions taken by the agent is applied the concept of the "action-value function for policy π", that represents an estimation of the total return expected, i. e., the quality of the action taken by the agent when it's following some policy π. This function represents the value of the expected total return to the state $s_t = s$ (current state) when the action a is chosen and it follows, from that state, the policy π, as shown in (3).

$$Q^{\pi}(s,a) = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} \quad (3)$$

[4] presents two main questions to be answered by the application of RL:

- Given a policy $\pi(s,a)$, what's the best way of

estimating $Q(s,a)$?

- Knowing an affirmative answer for the previous question, in what way can this policy be changed to make $Q(s,a)$ the optimal value of this function and, by this way, make it matches the optimal policy?

Literature includes many algorithms developed to answer these questions, but this research applied the Q-learning algorithm developed by [6] that among its advantages is the fact that it approximates directly the optimal value of $Q(s,a)$, independent from the policy chosen. The values of $Q(s,a)$ are updated according to (4).

$$Q(s,a) = Q(s,a) + \alpha[r_{t+1} + \gamma \cdot \max_a Q(s_{t+1},a) - Q(s,a)] \quad (4)$$

where α is the learning rate.

Given that the convergence of the algorithm is assured only if all the pairs state-action have been visited infinite times, the policy chosen to be applied to Q-learning has to guarantee that the probability of visiting all the pairs be different from zero, what can be reached by applying an □-greedy policy, defined by (5).

$$\pi(s,a) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|A(s)|}, \text{se } a = a^* = \arg\max_a Q(s,a) \\ \frac{\varepsilon}{|A(s)|}, a \neq a^* \end{cases} \quad (5)$$

The algorithm implementing Q-learning is presented in Fig. 4.

For the use of RL it was defined that the states could be error values (difference between the levels set and measured in the tank bellow). Even though a continuous system like this one has infinite states between -4 and +4 volts, the error signal was discretized with a resolution of 0,2 volts and it was designed a table that could have 41 discrete states. By this way, state mapping was given by (6).

$$s(k) = e(k) \cdot 5 + 21 \quad (6)$$

where $s(k)$ was the state at the time k, and $e(k)$ was the error between the desired level and the measured level at the tank bellow at the time k.

Initialize $Q(s,a)$ randomly;
Repeat (for each episode)
  Initialize s
  Repeat (for each period of time)
    Choose a for s using the policy π;
    Given the action a, watch r, s';
      $Q(s,a) = Q(s,a) + \alpha[r_{t+1} + \gamma \cdot \max_a Q(s_{t+1},a) - Q(s,a)]$
    $s \rightarrow s'$;
  Until the stop condition is reached.

Figure 4. Q-learning algorithm.

The agent was implemented to manage three different controllers, as mentioned before. The code included an index for activating each one (choice), depending on the action defined as choosing any of them. It was adopted an □-greedy policy, with 90% chance of choosing the action with the highest estimative.

The return function (reward) had to be defined in such a way that the agent could be trained to perform the optimal behavior. In that case, as the input function would be a step, it was desired that the agent could be trained to manage the controllers available to perform a response as close as possible to the step function. Thus, it was concluded that an appropriate return function (reward) had to involve the error measure and it could be expressed by (7-8).

$$R_t = \frac{|e(k) + e(k-1)|}{2} \cdot t_s \quad (7)$$

$$e(k) = r(k) - y(k) \quad (8)$$

where $r(k)$ is the setpoint, $y(k)$ is the level measured at the time k and t_s is the sampling time.

That return function was chosen because, while the output is different from the setpoint, there is an area between two sampling times that can be approximated by a trapezium, described by R_t. By this way, trying to minimize this reward function, the agent would manage the system controllers seeking the optimal policy that would result in the lowest value for this return function and the best output signal.

Defined the states, actions and return function, the next step was training the agent, because initially it does not know the consequence of choosing among the possible actions. That is why it has to choose each action and watch the results into the environment. For a while, the agent explores many actions seeking for decreasing its reward and gradually tends to repeat them (exploration). After that, he acquires knowledge from the actions and eventually learns to repeat the ones that result in lowest rewards (exploitation).

It was chosen the online training method, because the process (educational kit) was available for testing and it was known that all the results obtained with that training method would be better than the ones acquired from the offline training using models and simulations of the system. The training was defined to be performed in 250 episodes, each one with 120 cycles of iterations (loops), controlled by an internal index created to be increased at each sampling time. Each episode took a little less than 2 minutes to be performed, because 1 minute and 31 seconds was taken training the agent (the episode itself) and between 13 seconds and 20 seconds was taken setting up the system for the next episode. The next topic will present all the results achieved.

IV. RESULTS

After implementing the agent, the system started working as the schematic presented in Fig. 3. The agent started choosing one controller and waiting the sampling time, after

that, it could check the results and update the Q-table, choosing a controller for the next sampling time that could be the same or another one, according to the application of the □-greedy policy. The time sampling defined to perform the training and tests was 0.5 seconds, considering that the tanks system has slow dynamics.

During the online training, the agent kept switching among the three different controllers and improving its Q-table (policy) according to the algorithm shown in Fig. 4. Fig. 5a-d shows some episodes gathered during the agent training.

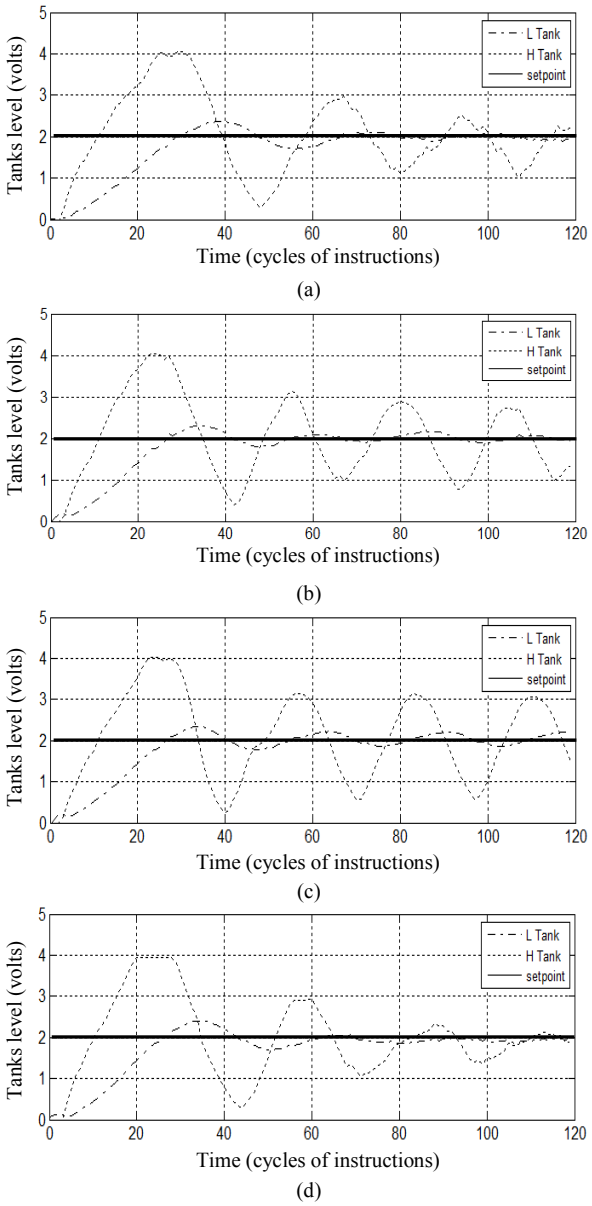

Figure 5. System response: (a) at the 1st episode; (b) at the 50th episode; (c) at the 100th episode and (d) at the 250th episode

Table I shows the improvement of the agent policy during its training, presenting some policies obtained after the 1st, 100th and 250th episode. As a consequence of defining the setpoint as 2 volts and reaching an overshoot of only 60% of this value, even having 41 possibilities (discrete states), it was necessary using only 14 of these states (lines), as showed on the table. For the states not mentioned on the Table I, it was kept the same original probability of choosing each controller (33.3%), as a result of not visiting any of that states during the training.

There are three columns for each policy showed on the Table I, referring to each controller applied on this research. For each state (line), Table I identifies the best choice indicated by the agent after the corresponding episode. For example, after the 250th episode, it concluded that when the output is equals to the setpoint ($e(k) = 0$), the best choice is switching for the controller C3.

TABLE I. POLICY IMPROVENT OF THE AGENT

	Policy 1			Policy 100			Policy 250		
Error (v)	C1	C2	C3	C1	C2	C3	C1	C2	C3
-0,6	0,33	0,33	0,33	0	0,50	0,50	0	1	0
-0,4	0	1	0	1	0	0	0	1	0
-0,2	0	0	1	1	0	0	0	1	0
0,0	1	0	0	0	1	0	0	0	1
0,2	0	1	0	1	0	0	1	0	0
0,4	0	0	1	0	0	1	0	1	0
0,6	0,50	0	0,50	0	1	0	1	0	0
0,8	0	1	0	1	0	0	0	1	0
1,0	0	1	0	0	0	1	0	1	0
1,2	1	0	0	0	1	0	1	0	0
1,4	1	0	0	1	0	0	1	0	0
1,6	1	0	0	1	0	0	0	0	1
1,8	0	0	1	0	0	1	0	1	0
2,0	1	0	0	0	1	0	1	0	0

Concluded the training and obtained the optimal policy for 250 episodes, this policy was tested by setting the agent to read this table and repeat 120 iterations. The results are shown in Fig. 6.

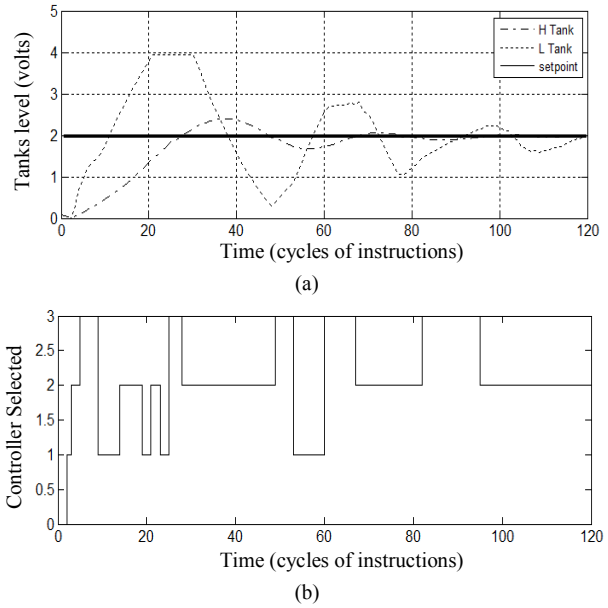

Figure 6. (a) system response to the agent aplying the optimal policy and (b) the switching sequence among the three controllers

Fig. 6a shows the system response obtained when the agent switches among the three controllers obeying the optimal policy established after the 250th episode and Fig. 6b shows the switching sequence, where C1, C2 and C3 are referred by their specific index indicated on the left axis.

Fig. 7 shows the error curves obtained for each individual controller and the RL agent that makes possible comparing their performance.

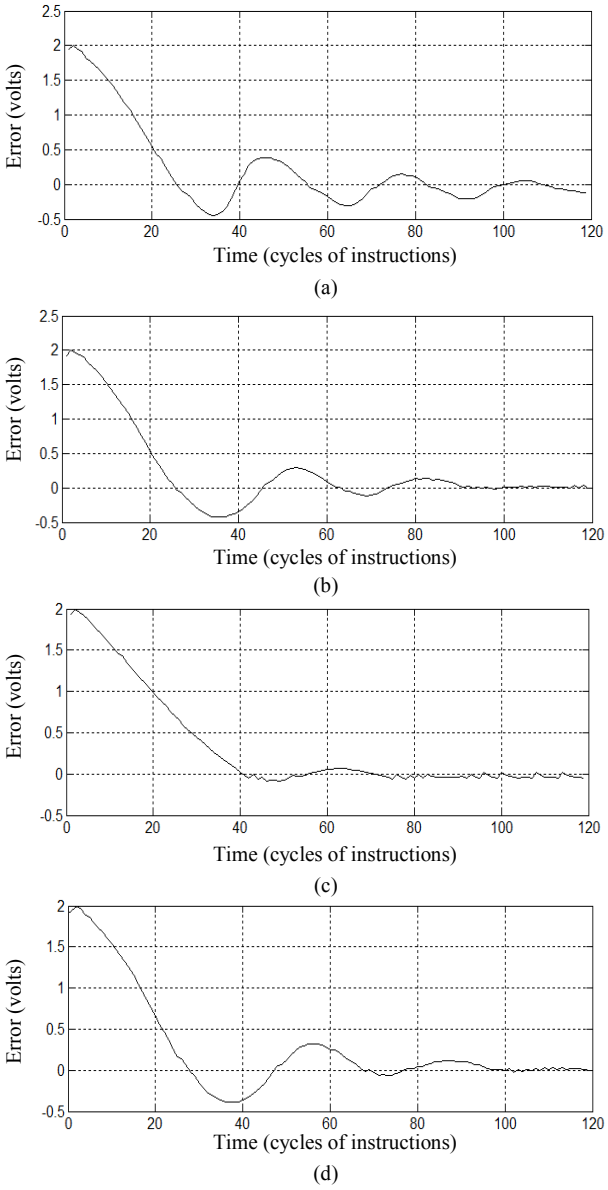

Figure 7. Error curves to the system applying: (a) C1; (b) C2; (c) C3 and (4) RL agent

Analyzing these error curves and the switching sequence, it is possible realizing that the RL agent starts applying more C1 and C2 at the output rising, because they are faster than C3. When the system response exceeds the setpoint, it tends to switch more between C2 and C3, because they are less oscillatory than C1, what indicates that it tries to take advantage of the best characteristics of each individual controller.

V. CONCLUSION

Through this experiment was possible concluding that reinforcement learning can be effective to implement a policy of switching among different controllers, as a way of taking advantage of their best characteristics to improve the system response of a physical system.

REFERENCES

[1] P. Albertos, J. Picó, and A. Sala, "Learning Control Systems: An overview," presented at the European Science Foundation Workshop on Control of Complex Systems (COSY), Rome, 1995.

[2] Y. M. Wang, Q. J. Liu, and T. Yu. "A Reinforcement Learning Approach to Dynamic Optimization of Load Allocation in AGC System". Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=05275778

[3] H. Pan, H. Wong, V. Kapila, M. S. Queiroz. "Experimental Validation of a Nonlinear Backstepping Liquid Level Controller for a State Coupled Two Tank System," Control Engineering Practice 13, pp. 27-40, 2005.

[4] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An introduction*. 4th Printing, The MIT Press, 2002.

[5] M. L. Lima Jr., J. D. Melo and A. D. Dória Neto. "Utilização de Sistemas Inteligentes Baseados em Aprendizagem por Reforço para a Otimização do Problema do Gerencimanto de Sondas de Produção Terrestre," presented at the 3º Congresso Brasileiro de P&D em Petróleo e Gás, IBP, Salvador, BA, 2004.

[6] C. J. C. H. Watkins and P. Dayan. "Q-learning," Machine Learning, no. 8, pp. 279-292, 1992.