*Chemical Product and Process Modeling*

*Volume* 2, *Issue* 1 2007 *Article* 7

# Learning to Control pH Processes at Multiple Time Scales: Performance Assessment in a Laboratory Plant

S. Syafiie* F. Tadeo†

E. Martinez‡

*Universidad de Valladolid, Spain, syam@autom.uva.es

†Universidad de Valladolid, Spain, fernando@autom.uva.es

‡National Research Council of Argentina, ecmarti@ceride.gov.ar

# Learning to Control pH Processes at Multiple Time Scales: Performance Assessment in a Laboratory Plant*

S. Syafiie, F. Tadeo, and E. Martinez

**Abstract**

This article presents a solution to pH control based on model-free learning control (MFLC). The MFLC technique is proposed because the algorithm gives a general solution for acid-base systems, yet is simple enough for implementation in existing control hardware. MFLC is based on reinforcement learning (RL), which is learning by direct interaction with the environment. The MFLC algorithm is model free and satisfying incremental control, input and output constraints. A novel solution of MFLC using multi-step actions (MSA) is presented: actions on multiple time scales consist of several identical primitive actions. This solves the problem of determining a suitable fixed time scale to select control actions so as to trade off accuracy in control against learning complexity. An application of MFLC to a pH process at laboratory scale is presented, showing that the proposed MFLC learns to control adequately the neutralization process, and maintain the process in the goal band. Also, the MFLC controller smoothly manipulates the control signal.

**KEYWORDS:** learning control, goal-seeking control, process control, intelligent control, online learning, neutralization process, pH control

---

*This work was funded by mcyt-CICYT project DPI2004-07444-C04-02. The first author was also funded by AECI-BecasMAE and UVa.

## 1. INTRODUCTION

Control of pH in neutralization processes is a ubiquitous problem encountered in chemical and biotechnological industries. For example, pH value is controlled in chemical processes such as fermentation, precipitation, oxidation, flotation and solvent extraction processes. Also, controlling the pH in food and beverage production is an important issue as the growth and activity of anaerobic microorganisms are largely pH dependent. Also, in the decomposition section of Sucono/UOP Phenol Process: The acid catalyst that is added in the decomposition section must be neutralized to prevent yield loss due to side reactions and protect against corrosion in the fractionation section (Schmidt, 2005).

Control of pH also involves the chemical equilibrium, kinetic, thermodynamic and mixing problems. In designing controllers, these characteristics must be considered (Gustafsson *et al.*, 1995). By considering them, it makes difficulty to design a controller. In other hand, the process buffer capacity varies with time, which is unknown and dramatically changes process gain. This can be understood as, for example, if either the concentration in the inlet stream or the composition of the feed changes, the shape of the titration curve will be drastically altered. Therefore, the process nonlinearity becomes time-varying and the system moves among several titration curves. Consequently, it is difficult to develop appropriate mathematical models of the pH process for control design.

Intelligent control strategies have been proposed for pH control applying a wide array of techniques such as fuzzy control (Fuente *et al.*, 2006) and neural networks (Ramirez and Jackson, 1999). Fuzzy self tuning PI control (Babuska *et al.*, 2002) and fuzzy internal model control (Edgar and Postlethwaite, 2000) have also been implemented to control pH processes.

The approaches cited above have several difficulties for practical applications, and also are difficult when tackling control system design. The resulting control structures are complex and difficult to supervise. They might be conservative or may have many tuning parameters. Thus, tight and robust pH control is often difficult to achieve due to the inherent uncertain, nonlinear and time varying characteristics of pH neutralization processes.

This paper discusses an alternative approach to solve the pH control problem by applying *Model-Free Learning Control* (MFLC) (Syafiie *et al.*, 2004; 2005; 2006a; 2006b), based on the *reinforcement learning* (RL) framework (Sutton and Barto, 1998).

In standard **RL** algorithms, like *Q*-learning, there is a difficulty for Process Control implementations: these algorithms scale very badly with increasing problem size, granularity of states or control actions. Among others, one intuitive reason for this is that the number of decisions from the start state to the goal state

increase exponentially.

According to the problem size, to keep tractable the number of decision to be taken to reach the goal state, hierarchical approaches based on *temporal abstraction* have been proposed. Temporal abstraction can be defined as an explicit representation of *extended actions*, as policies together with a termination condition (Precup, 2000). The original one-step action is called *primitive action*. Semi Markov Decision Processes (SMDPs) is the theory used to deal with the temporal abstraction as a minimal extension of **RL** frameworks. SMDPs is a Markov Decision Processes (MDP) appropriate for modeling continuous-time discrete-event systems.

Several **RL** algorithms resorting to hierarchical temporal abstraction approaches have been proposed: Options (Sutton *et al.*, 1999); Hierarchy of Abstract Machine (HAM) (Parr, 1998); MaxQ (Dietterich, 1997) and Multi-step actions (MSA) (Riedmiller, 1998). The first three methods are based on the notion that the whole task is decomposed into subtasks each of which corresponds to a subgoal. The last one, using MSA, is very convenient for process control application where typically the most appropriate time scale for defining control actions is unknown in advance.

The paper is organized as follows. A short introduction to reinforcement learning is given in section 2. The architecture of model-free learning control (MFLC) for chemical process is discussed in section 3, followed by MSA approach in section 4. In sections 5 and 6, the application to real laboratory plant and discussion are given. Section 7 gives some conclusions.

## 2. REINFORCEMENT LEARNING

Reinforcement Learning (RL, for short) can be defined as '*learning what to do by doing'*, i.e. how to map perceptions of process states to control actions, so as to maximize an externally provided scalar reward signal. According to this definition, the learning agent (controller) is not instructed to act under the tutelage of an exemplar teacher, as in most forms of supervised learning, but instead must try control actions seeking out those that provide the maximum cumulative reward. These algorithms are based on online learning directly from the closed-loop behavior of the plant.

The learning algorithm in **RL** emphasizes the *interaction* between an active decision-making agent (or an intelligent controller) and its target dynamic system or plant (see Figure 1). In the latter, a desired behavior or control *goal* is permanently sought despite lacking *a priori* knowledge about system dynamics and the influence of external disturbances, including other controllers. The *reward function* can also incorporate information on one or more *preference* indices. These preferences define the most desirable ways for achieving a control goal (or

objective) and are the basis for assigning rewards (or penalties) to a learning controller.

An *agent* (called controller) interacts with its *environments* (controlled system or plant). They interact continually: the agent selects an action and then the environment responds to the action and presents a new state or situation to the agent. The agent uses features from the new situation as clues for decision making in the next time step, can be seen in Figure 1.

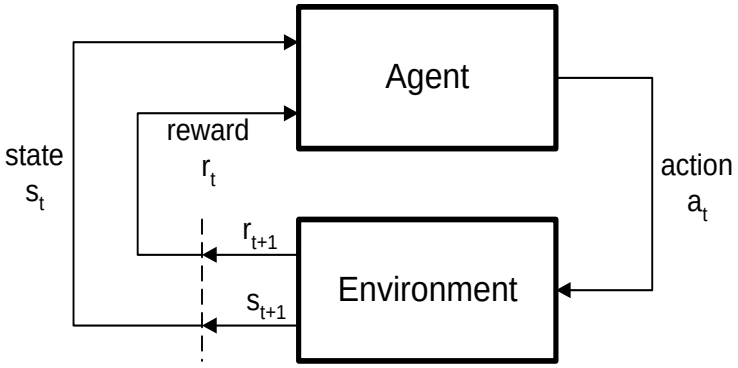


**Figure 1**. An agent interacts with its environment: dash line represents delay.

Achievement of the control goal and preference optimization demands foresight to account for indirect, delayed consequences of control actions. This is particularly critical for a chemical plant where material recycles gives rise to process dynamics slowly unfolds over time. This goal-oriented perspective of control actions gives rise to a central problem of **RL**: to devise a computational approach with the capability of apportioning rewards over a sequence of actions, taking into account the control goal to be achieved and the preferences to be optimized. A key concept in this regard is that of an *action-value function*, which has permitted an important breakthrough in the analysis and design of **RL** algorithms (Sutton and Barto, 1998).

When resorting to controller/process interactions for learning, four typical components of an **RL** algorithm can be identified: a *control policy*, a *reward function*, a *value function*, and a *learning algorithm*. The control policy or feedback law is a dynamic relationship (i.e. changes with interaction) that defines which action to take at a given state bearing in mind the achievement of the goal and optimizing the preferences. The other components serve as means to learn and improve the control law; i.e. their existence is only justifiable on the grounds of being components of learning algorithms for the control law.

When the desired goal has been reached (reward) or when the system has failed (punishment) a signal indicates the success or failure after a sequence of

action has been taken. This signal is called *delayed reinforcement signal*.

## 2.1 Reward function

The reward function translates the goal and preferences into single numbers indicating the short-term benefit or reward $r_{t+1}$ of taking the action $a_t$ at a given state or situation $s_t$. Note that the reward is a single number that varies from one decision step to another. However, the very purpose of the control law is to maximize the cumulative reward that can be obtained over the next $h$ steps in the sequence of rewards that follow the time step $t$ denoted by $r_{t+1}$, $r_{t+2}$, $r_{t+3}$,..., $r_h$. The cumulative reward $R_t$ that can be obtained from time $t$ on is just the return, defined as:

$$R_t \equiv r_{t+1}+r_{t+2}+r_{t+3}+...+r_h. \tag{1}$$

To make definition (1) more mathematically tractable for long-sequence of controls (i.e. when h $\rightarrow \infty$) it is better to geometrically discount rewards using a recency factor $\gamma$, (where $0 \leq \gamma \leq 1$):

$$R_t = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{2}$$

The discount factor $\gamma$ is used to weight near term reinforcements more heavily than distant future reinforcements. If $\gamma$ is small, the agent learns to behave only for short-term reward. The closer $\gamma$ is to 1 the greater the weight assigned to long-term reinforcements.

Using the return $R_t$, it is possible to assess *how good* (or bad) is to take the action $a_t$ at the state $s_t$, from the point of view of the control goal and the chosen preferences. This forms the basis for defining the action-value function. To clearly distinguish between the effect on $R_t$ of $a_t$ from the effect on $R_t$ of decisions to be taken later in sequel, the action-value function is defined as follows. At time step *t,* the action-value function approximates the expected value of $R_t$ upon executing $a_t$ when $s_t$ is observed and acting *optimally* thereafter. This function is called an *action-value function* and is mathematically defined by

$$Q(s_t, a_t) = E\{R_t | s_t = s, a_t = a\}. \tag{3}$$

The main issue to be solved during learning is the dilemma of *exploration* versus *exploitation*. To exploit what it is already known, good estimates of the actual value of actions at different states are needed. For this, exploration of

actions with apparently lower values must be tried. To exploit more after learning it is necessary to explore more during learning.

### 2.2 *Q*-Learning Algorithm

A policy, $\pi(s_t, a_t)$, is defined via the action value function, with represent how much the future rewards the agent would get by taking the action $a_t$ at state $s_t$ and following the current policy in subsequent steps. Equation 3 can be rewritten as an immediate reinforcement, plus a sum of future reinforcements:

$$Q_\pi(s_t, a_t) = E_\pi\left\{R(s_t, a_t) + \sum_{k=1}^{T} \gamma^k R(s_{t+k}, a_{t+k})\right\}. \tag{4}$$

The equation can be updated by substituting the sum of future reinforcements with the estimated value function:

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + E_\pi\{R(s_t, a_t) + \gamma Q_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t)\}, \tag{5}$$

where the expected discounted reinforcement of taking action $a_t$ is taken over the next state, $s_{t+1}$, given that the current state is $s_t$. A model of state transition probabilities is needed. If it does not exist, a Monte Carlo approach can be used in which a single sample replaces the expectation, and the value function is updated by a fraction of the difference (Anderson *et al*, 1997):

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha_t[R(s_t, a_t) + \gamma Q_\pi(s_{t+1}, a_{t+1}) - Q_\pi(s_t, a_t)], \tag{6}$$

where the *learning rate*, $0 \le \alpha \le 1$, is tuned to maximize the speed of learning, as small learning rates induce slow learning, and large learning rates induce oscillations. Value iteration is applied to increase the action-selection policy and achieve optimal control. This dynamic programming method combines steps of policy evaluation with policy improvement. The update corresponds to the equation:

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha_t\left[R(s_t, a_t) + \gamma \max_{b \in A} Q_\pi(s_{t+1}, b) - Q_\pi(s_t, a_t)\right] \tag{7}$$

This equation is known as *Q-learning* algorithm, which is used to improve optimal control.

The agent knows neither exactly the optimal value function nor the correct estimation of the dynamic environment. If the agent knows and estimates this value correctly, the policy can select a greedy action in each time step. This is

called *exploitation*. However, the agent should execute trial actions in order to know the optimal value function. This is called *exploration*.

In this study, the $\varepsilon$-greedy policy explores and exploits the available actions with $\varepsilon$ probability of choosing an apparently nonoptimal action. The action which has maximum $Q$-value will be selected with (1-$\varepsilon$) probability and the rest will explore and exploit nonmaximum-$Q$-value-action. The exploration of choosing nonmaximum-$Q$-value-action is chosen based on uniform distribution.

## 3. MODEL-FREE LEARNING CONTROL

The proposed Model-Free Learning Control (MFLC) architecture can be seen in Figure 2. First, the "*Situation*" block preasents the next state and the corresponding reward, based on the desired setpoint and measured output. This presentation is based on *Boolean* or *fuzzification* functions. In current state, the agent uses the "*Policy*" to select an action from those available. Then, control signal $u_t$ is calculated in "*Calculate U*" block and actuated to the plant. The selected action is then criticized by using "*Critic*" block as "good" or "bad" one based on evaluation what happens in the plant after applying the selected action, which is based on measurement in the next step.

The proposed MFLC in this study is based on this architecture. The *policy* block on Figure 2 can be any strategy or algorithm to select actions. For one step learning, every action chosen by the agent in this application is criticized time to time. Whereas, for temporal abstraction (extended actions), the agent applies the selected action for several time steps until termination condition is satisfied. Undoubtedly, when the agent selects temporal abstraction, the policy will be hold until termination condition. Soon after the termination condition is reached, and then the policy will be used to explore and select the optimal action.

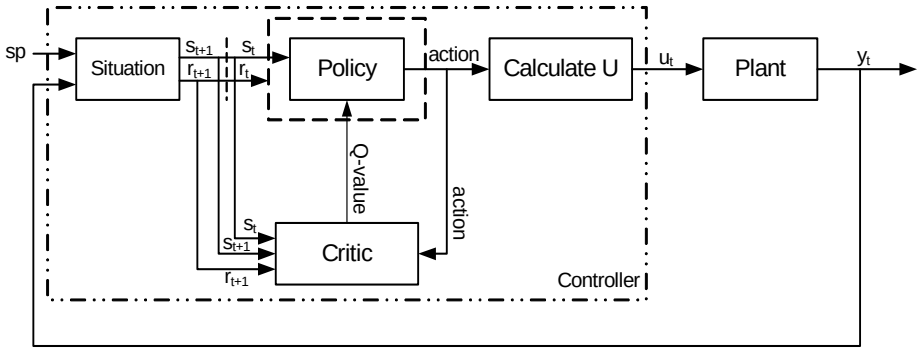


**Figure 2**. Model-Free Learning Control Architecture.

Compared to other control techniques based on learning, the proposed

MFLC has some clear advantages:

- It is possible to put in the design of the controller a priori knowledge about the plant.
- The control algorithm is quite simple from a computational point of view, so it is feasible to implement using low-cost hardware.
- It is possible to derive and obtain a feedback control law from an optimal control point of view based on actual experience rather than a process model, which makes it attractive for Control and Plant Engineers.

### 3.1 State definition in MFLC

In MFLC, symbolic states are defined as the length of error from desired setpoint. The symbolic state is represented as in Figure 3.

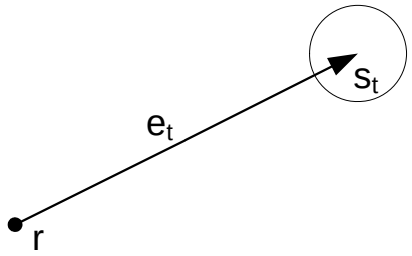


**Figure 3**. State is the length of error from the reference in current time.

It is introduced a band in desired control objective to allow some tolerance error, which makes room to allow noise. Therefore, the objective can be augmented to have the output of the system being in the bound, $-\eta \le r \le \eta$, all time. This band is called the *goal band*, and it is defined as *goal state*. The other symbolic states are defined as how far from this band that the controller has to explore finding optimal actions to achieve the goal band. For example, the *outer state* is when the error of system is bigger or smaller than $r \pm \eta \pm n$, respectively. This means that when the system in this band, the controller provides a control signal by exploring in this area.

Usually, the available states are defined based on the length of $n$ from the goal state, which is depicted in Figure 4. Generally, the longer $n$ is from the goal state, the more states are available for the controller to be learned and explored the environment. The parameter $n$, is a unique solution, can be selected as a region where the controller explores to provide actions to learn the environment.

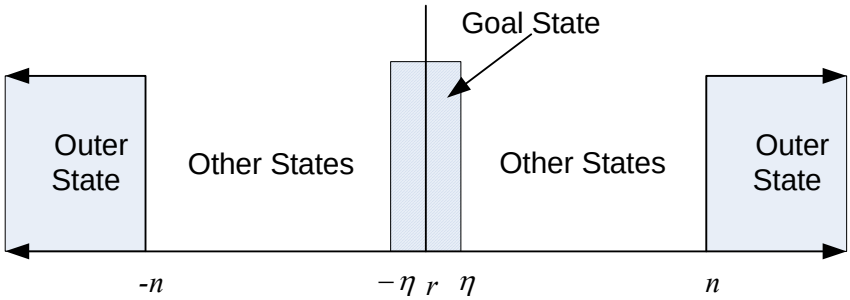


**Figure 4**. States definition from desired point.

### 3.2 Action Definition

Applying **RL** for process control, the number available actions leading to control signals will be a huge space. However, the number of actions in MFLC can be reduced by bounding the incremental control signal as a function of the state. This function is as follows:

$$\varsigma(s_i) \le \Delta u(s_i) \le \xi(s_i), \tag{8}$$

where $\zeta(s_i)$ and $\xi(s_i)$, respectively, are lower and upper bound constraints of the agent can manipulate previous control signal when the system is in state $s_i$. This means that increase or decrease previous control signal is allowed on the limitation, when the system is within the state $s_t$ in time $t$. In the goal state, a *wait action*, which is for keeping previous control signal, is defined.

With respect to the constraints (9), the control signal can be calculated as follows:

$$\Delta u = k\left(a_{op}^{s_i} - a_w\right), \tag{9}$$

where $0<k<1$ is gain, $a_{op}^{s_i}$ is optimal actions selected by the agent and $a_w$ is wait action. Wait action means that when the system is in the goal state, $s_g$, the agent selects action, $a_{op}^{s_g}$, that is equal to $a_w$. Therefore, the right hand side of the equation (10) is zero.

The action, $a_b^{s_i}$, satisfying lower bound constraint of incremental control signal is calculated in state $s_i$ as follows:

$$a_b^{s_i} = \frac{\zeta(s_i)}{k} + a_w. \tag{10}$$

The action, $a_p^{s_i}$, satisfying upper bound constraint of incremental control signal in state $s_i$ is calculated as

$$a_p^{s_i} = \frac{\xi(s_i)}{k} + a_w. \tag{11}$$

The distance between these actions, $d\left(a_b^{s_i}, a_p^{s_i}\right)$, in state $s_i$ is

$$d\left(a_b^{s_i}, a_p^{s_i}\right) = \frac{|\zeta(s_i) - \xi(s_i)|}{k}. \tag{12}$$

It stresses out that $k$, $\zeta(s_i)$, $\xi(s_i)$ are chosen such that $a_b^{s_i}, a_p^{s_i}$ are integers, so that the distance between this actions is also going to be integer.

The actions in state $s_i$ between $a_b^{s_i}, a_p^{s_i}$ are:

$$a_j^{s_i} = a_b^{s_i} + d, \quad j \in [1,2,\cdots,d+1],\ d \in [0,1,\cdots,d]. \tag{13}$$

In state 1, the action satisfying the lower bound incremental constraint is action 1. Therefore, wait-action is calculated from Equation 10, as follows:

$$a_w = 1 - \frac{\zeta_1}{k}. \tag{14}$$

The distance between lower and upper constraints can be a fixed value for all available states, or can be also chosen different values as a function of states. Usually, in the goal state the distance is chosen smaller or even to keep previous control signal. If the fixed value is chosen, it means that the agent has the same space in every given states to explore in order to provide optimal control signal. Therefore, the available actions are the same in every visited state. Difference values give different spaces for the agent to learn to present optimal control signal

## 4. MFLC (MSA)

In this paper the concept of MSA (Schoknecht and Riedmiller, 2003) is applied to pH control because it is best suited for systems where no decomposition in subgoals is known in advance. As in the general framework defined by Sutton *et al.* (1999), MSA is a special type of semi-Markov option. A Markov option would require a state-dependent termination condition. In the MSA algorithms, the

termination condition is applied after executing a sequence of *m* primitive actions.

The MSA method is a method enabling an intelligent control to learn a control policy by using multiple time scales simultaneously. The MSA consists of several identical actions in the primitive time scale. This algorithm is possible to increase responsiveness and add flexibility to the controller behavior. Also, giving a learning controller the possibility of using MSA to reach the goal can improve the speed of learning and reduce control efforts. This approach has been successfully applied in a simple thermostat control (Riedmiller, 1998; Schoknecht and Riedmiller, 2003). Thus, we think that the algorithm can be extended to complex and highly nonlinear problem, such as the pH control problem.

MFLC based on MSA seems to be a promising approach to overcome the pH problem as mentioned above because the control law can easily adapt to varying scenarios by online learning. By experiencing a sequence of identical actions applying for pH process, the agent can speed up learning and planning, to maintain the process in the desired pH value. In order to explore the set of possible actions and acquire experience through the reinforcement signals, the actions are selected using an exploration/exploitation policy. In this study *ε-greedy policy* is applied to select one of the available actions in visited state and experience it for a multiple time steps of the plant. The ε-greedy policy has been selected because it gives better performance for pH process than *softmax* policy (Syafiie *et al.*, 2004).

This proposed solution can be used for general-purpose system control in acid-base process, providing the user-friendly and smart control system that users demand. The advantages of MFLC based on MSA are:

- no precise quantitative knowledge of the process is available,
- no process identification mechanism (identifier is included in the system, which is an online learning), no controlled design for a specific process is needed,
- simple manual tuning of controller parameter is required and stability analysis criteria are available to guarantee the closed-loop system stability,
- compared to the standard **RL**, this *Q*-learning-modification (MSA) algorithm is proposed for pH process because it can extract more training examples from the each experimental run. The agent executes a primitive action and applies it for *m* time steps.

The idea of MSA is based on a set of all multiple step actions of degree *m,* defined as $A^{(m)} = \{a^m \mid a \in A^{(1)}\}$, where $a^m$ denotes the MSA that arises if action *a* is executed in *m* consecutive time steps (Schoknecht and Riedmiller, 2003). The next action will be executed after the whole MSA has been applied. Thus, the MSA has a time-dependent termination condition after *m* primitive time steps.

The concept of MSA can be integrated in learning algorithms, such as *Q*-learning. For example, when the controller executes action $a^m$ of degree *m* in state

$s_e$ the environment makes transition to state $s_n$ after $m$ time steps. The state-action value can be updated as follows:

$$Q\left(s_e, a_e^m\right) \leftarrow Q\left(s_e, a_e^m\right) + \alpha\left[r\left(s_e, a_e^m\right) + \gamma^m \max_{b \in A} Q\left(s_n, b\right) - Q\left(s_e, a_e^m\right)\right] \tag{15}$$

where

$$r\left(s_e, a_e^m\right) = \sum_{\tau=i}^{i+m-1} \gamma^{\tau-i} r_{s_\tau a}$$

where $Q\left(s_e, a_e^m\right)$ is $Q$-value for state-action where MSA is executed in state $s_e$, and $Q(s_n, b)$ is $Q$-value for next state, $\alpha$ is learning rate, and $\gamma$ is discount factor. When action $a^m$ with degree $m$ is selected in $s_e$, the environment makes a transition to $s_{e+m}$ with reward $r\left(s_e, a_e^m\right)$. When executing $a^m$, all actions $a_i$, $e = 1, 2, ..., m - 1$ are executed implicitly. The transition from $s_e$ to state $s_{e+m}$ contains all information necessary to update the $Q$-values for those lower-level actions at all intermediate states.

This paper also proposes to introduce additional closed-loop termination conditions in order to study flexibility and speed up learning and planning. Besides the $m$ time steps termination condition (Syafiie *et al.*, 2005), two new termination conditions are introduced: the MSA execution finishes if the process reaches the goal band and/or if the control goal has changed (for example the desired setpoint). If in time $t$ the agent executes MSA for $m$ time step, but in time $t+j$ ($m>j$), the control goal has been changed. Therefore, the MSA should terminate on time $t+j$.

## 5. APPLICATION TO PH PROCESSES

This section describes the proposed MFLC (MSA) implementation to a pH process.

### 5.1 States and Reward

The main control objective is to maintain the pH inside a band of ±δ around the desired setpoint (the width of this band is defined by measurement noise in the process and allowed tolerance). This band is defined as the goal state. Other states are defined in accordance to values of the pH outside this band, as depicted in Figure 5.

To classify the reading of pH and to select an action available in each

visited state, this study uses 11 symbolic states, where the goal state is 6 (corresponds to the desired pH band). The number of states is selected to satisfy the error of the pH process. For example, the states are defined for $-1 \le (y - r) \le 1$.

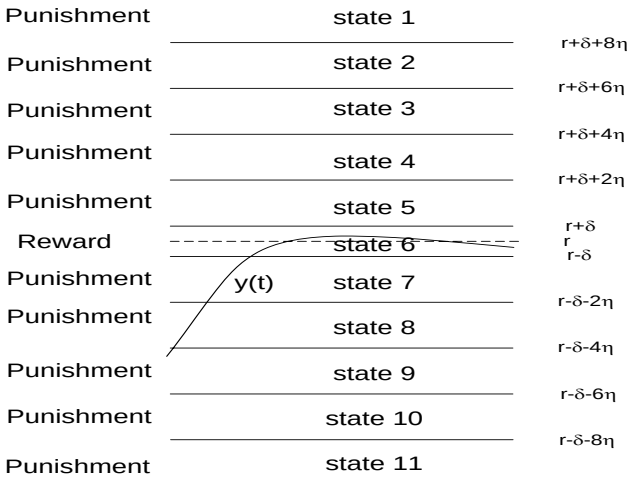


**Figure 5**. Control objective and definition of states.

## 5.2 Control Actions

The available actions in every state are defined satisfying incremental constraint:

$$\Delta u(s_i) = \begin{cases} 0 < \Delta u(s_i \neq s_g) \le \mu \\ \Delta u(s_i = s_g) = 0 \end{cases} \tag{16}$$

where $s_g$ is goal state, and $\mu$ is the upper limit incremental constraints in state $s_i$. From this defined incremental constraint, the available actions are calculated as in *Syafiie et al.* (2006a).

In MFLC, the agent selects an action and executes it in current time and receives the next reward. From the chosen action, the control signal, $u_t$, is calculated as follows:

$$u_t = u_{t-1} + k(a_w - a_t) \tag{17}$$

where $a_t$ is the optimal action chosen by the controller from those available actions in every visited state, and $a_w$ is the *wait action* where there is no change to the previous control signal. For example, if the system has three actions, where action one is to increase the previous control signal, action two is to maintain the previous control signal and action three is to decrease it. Therefore, action two is

called wait action. The controller gain, $k$, is a tuning parameter that can be selected to weight how much to increase or decrease previous control signal.

The number of actions in every state is calculated as follows:

$$N(s_i) = \frac{\mu}{k} + 1. \tag{18}$$

It is stress out that $\mu$ and $k$ are chosen such that $\mu/k$ is integer, so that the number of actions $N$ is also integer.

The probability that the system moves to a new state from the current state depends on the system behavior following the execution of the chosen action. For instance, if the process is in state 1, and the controller chooses action 1, the process may move either to state 2 or to another state or stay in state 1.

These states and reward are defined by a parameter that refers to the setpoint, $r$, as a desired output. The goal state is restricted by boundary values: upper, $r+\delta$, and lower, $r-\delta$ as shown in equation 16. The maximum reward is introduced in the goal state. When the system is outside the desired band, the controller is punished by negative rewards. This reward function is applied in each state as a single number:

$$reward = \begin{cases} 1 & if\ r-\delta < pH < r+\delta, \\ -1 & else \end{cases} \tag{19}$$

MFLC uses zero initial condition for the values of the (tabular) $Q$-function. This value entry in $Q$ is updated by each time when following an action the process situation changes. When the environment changes, for example, the setpoint changes, the action-value function $Q$ is immediately reset to an initial condition. Resetting $Q$-values to their initial values makes possible for the agent to learn new environment without any influence from the past learning onto the new environment.

### 5.3 MFLC (MSA) Control Algorithm

The MSA algorithm developed for the learning system is as follows:

1. Observe/Read the state $s_t$
2. Select an action $a_t$, this action is chosen from state $s_t$ using $\varepsilon$-greedy policy
3. Apply the selected action $a_t$ for $n$ time steps
4. Do until terminating condition $m=n$
    1. Read the resulting state $s_{t+m}$
    2. Update $Q$-value using Equation (8)

## 6. RESULTS AND DISCUSSION

The experimental results section describes and discusses the application of MFLC based on **RL** to a pH process control on the laboratory plant.

### 6.1 Description of the Experimental Setup

The experimental setup consists of a continuous stirred tank reactor (CSTR) where a process stream (sodium acetate) should be maintained at a certain pH value for which it is titrated with a solution of hydrochloric acid (HCl). The solution of sodium acetate ($CH_3COONa$) is prepared using various concentration levels. The hydrochloric acid is prepared 1% of volume. It is assumed that the mixing in the tank is homogeneous; therefore, the concentration in the effluent stream is similar to the concentration in the reactor.

For the experimental process, the zero initialized $Q$-value is used. The value of the *meta – parameter* for the agent are selected to be: discount factor, $\gamma$=0.98 and learning rate, $\alpha$=0.1, which were determined to be a good values from previous work (Syafiie *et al.*, 2004).

As mentioned above, the defined system has eleven states. In this implementation, the parameters $\delta$ and $\eta$ are selected to be $\delta=0.1$ and $\eta=0.1$, based on the level of measurement noise and the desired operating range of pH. From the parameters $\delta$ and $\eta$, it can be defined that state 1 is when the measured pH is higher than $r+\delta+8\eta$. It means that the agent is in state 1 if the pH is higher than $r+\delta+8\eta$. State 2 is defined when the pH is lower than $r+\delta+8\eta$ and higher than $r+\delta+6\eta$. The rest of the states are defined following Figure 5.

In this application, the gain of MFLC is chosen $2\times10^{-5}$ and the upper limit of incremental constraint is selected to be $8\times10^{-5}$. Therefore, each state has 5 possible actions, except in the goal state that has only 1 action, which is the so-called *wait action*. The wait action is chosen to be 22, which is no manipulation of previous control signal. In this MSA application, 3 identical primitive actions are executed in every multiple time scale.

In MFLC, the $\varepsilon$-greedy policy is applied for choosing an action in every visited state of the pH process. Parameter $\varepsilon$ used in the $\varepsilon$-greedy policy is selected to be $\varepsilon$=0.1, to leave space for the agent to explore the available actions. This means that exploration (choosing an action that does not have the estimated maximum action-value) will be selected with a probability of 1 out of 10, which represents a good compromise for controller performance, given its time-varying and nonlinear characteristic (less experience would be necessary if the plant were more linear and the concentration less uncertain).

### 6.2 Experimental Results and Discussions

The application of the proposed MFLC (MSA) controller to the laboratory plant shows that the responses of the process are reasonably well. It can be seen in the responses of the plant by applying MFLC (MSA) controller for some changes in setpoint for the $NaCH_3CHOO - HCl$ system and comparison to a PID controller can be seen in Figure 6. Whereas, the application of MFLC (standard) can be seen in *Syafiie et al.*(2005).

The comparison shows that the responses of the application the proposed MFLC (MSA) controller settle in reference faster than the PID controller. Also, the responses of the plant show that MFLC controller based on RL algorithms lay closer to the references, whereas the PID controller has higher peak of oscillations.

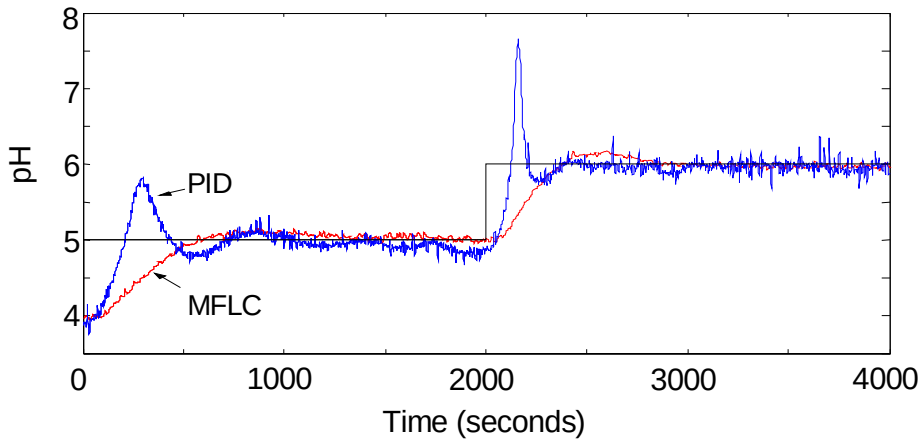


**Figure 6**. pH Responses of the application of MFLC (MSA) and PID controllers.

The control signal (Figure 7) shows that the MFLC (MSA) controller manipulates, after learning, the actuator much smoother than a PID controller. Since MFLC allows a tolerance error of the process whenever the pH is within the control band, the control signal is smoother when the process is closer or within the pH band.

Control signal produced by MFLC (MSA) is smoother than PID controller because the MFLC controller can only manipulate control signal satisfying incremental control. Also, the MFLC acts passively when the process in the goal band. It means that the agent resorts to the wait action.

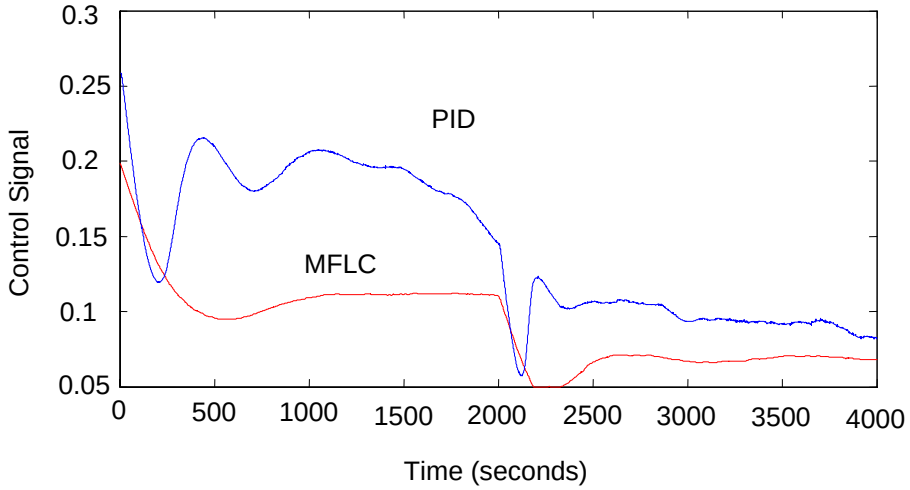


**Figure 7**. Control Signal by using of MFLC (MSA) and PID controllers.

6.3 MFLC (MSA) Termination Condition Study

In this section, it is reported the study of the effect for simultaneously taking and executing identical actions. The proposed MFLC (MSA) algorithm was applied to the laboratory pH plant, for 10, 20 and 30 steps of identical actions. All parameters chosen are as in that section previous section and $k$=$10^{-5}$ is selected.

The responses for all steps actions are plotted in Figure 8 and control signals are shown in Figure 9.

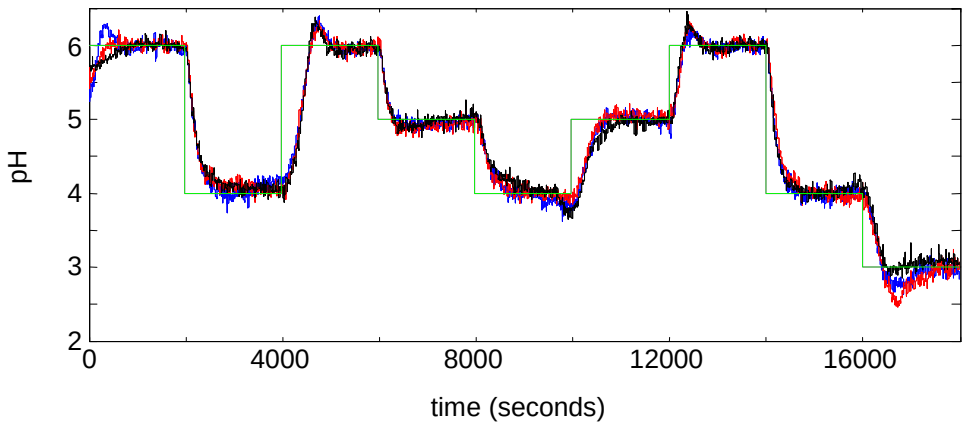


**Figure 8**. Online Responses of MFLC (MSA) for $NaCH_3COO$-HCl system, for n=10, 20 or 30 steps.

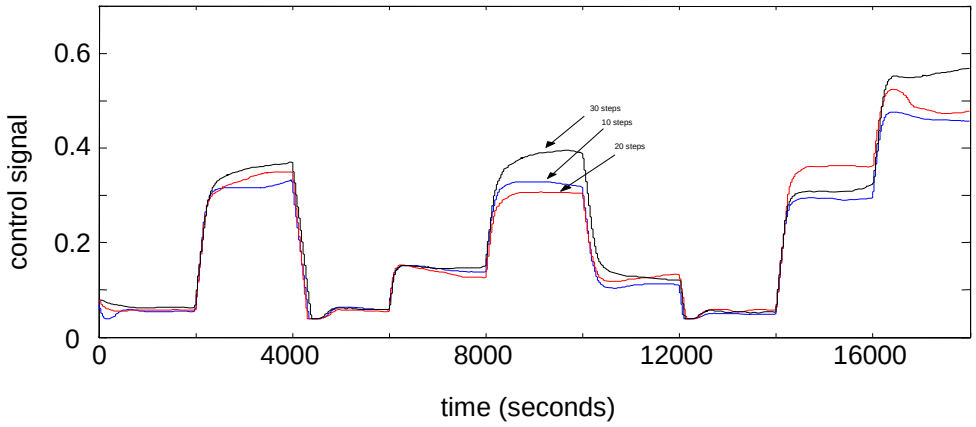


**Figure 9**. Control signal of MFLC (MSA) for $NaCH_3COO$-HCl system, for n=10, 20 or 30 steps.

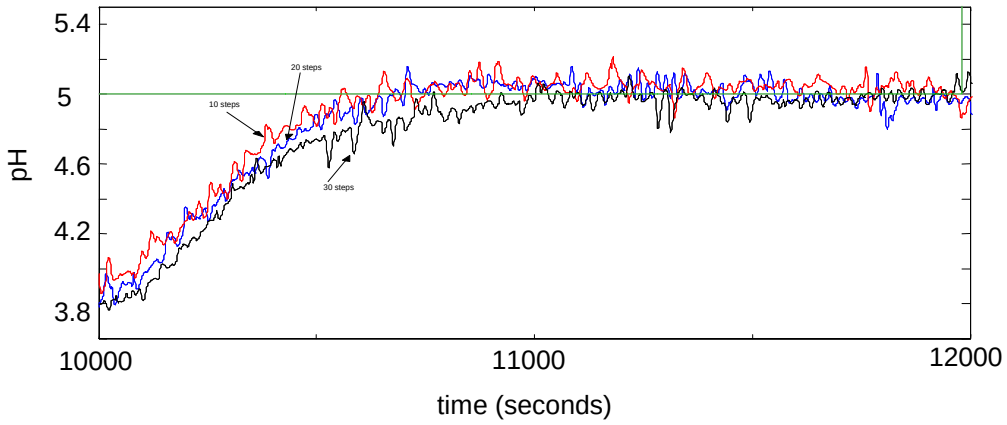


**Figure 10**. Detail of Responses from 10000 to 12000 secs.

The online comparison responses of applying identical actions to the laboratory pH plant can be seen in Figure 8, which clearly shows that the responses are good enough and lie close to the reference. The responses of applying 10, 20 and 30 step-actions of MFLC (MSA) show quite similar. There are no significant differences that can be seen from the responses of multiple time steps of action courses (Figure 8). This is because the agent has the same space for exploring and providing actions.

However, the agent reduces significantly the exploration time for applying 20 step-actions. However, the responses of 30 step-actions are a little slower than others, because the agent has a little time to try other actions. The pattern of the control signal for these MFLC (MSA) algorithms, given in Figure 9, is similar.

## 7. CONCLUSION

In this paper, the algorithm with multiple time scale MFLC (MSA) has been proposed for pH control problems and applied on a pH control process at laboratory scale. It has been shown that the main advantage of designing control systems using MFLC (MSA) is that a priori model of the process is not necessary. Also a simple algorithm is used that takes into account different constraints. The proposed algorithm, MFLC (MSA), performs reasonably well when controlling in real-time a pH laboratory plant. The responses of the process applying the proposed MFLC (MSA) show that the process regulated adequately well. Also, the MFLC agent manipulates smoothly the control signal.

## REFERENCES

[1] Anderson C. W., D. C. Hittle, A. D. Katz and R. M. Kretchmar, Synthesis of reinforcement learning, neural network and PI control applied to a simulated heating coil, *Journal of Artificial Intelligence in Engineering*, **1997**; 11; 4; 423 – 431.

[2] Babuska R., Oosterhoff J., Oudshoorn A., and Bruijn P. M., "Fuzzy self tuning PI control of pH in fermentation", *Journal of Engineering Application of Artificial Intelligence*. **2002**; 15; 3 – 15.

[3] Camacho E.F. and C. Bordóns., *Model Predictive Control in the Process Industry*, **2004**; Springer-Verlag. London.

[4] Dietterich T. G., Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition, *Technical report*, 1997, Department of Computer Science, Oregon State University.

[5] Edgar, C. R., and Postlethwaite B. E., MIMO fuzzy internal model control, *Automatica*, **2000**; 34; 867 – 877.

[6] Fuente M. J., Robles C., Casado O., Syafiie S. and Tadeo F., Fuzzy Control of a Neutralization Process, in press (*Engineering Applications of Artificial Intelligence*)

[7] Gustafsson, T.K., Skrifvars B. O., Sandström K.V., Waller K. V., Modeling of pH for Control, *Industrial Engineering Chemical Research*, **1995**; 34; 820 – 827.

[8] Parr, R., Hierarchical Control and learning for Markov decision processes, *PhD thesis,* **1998**, University of California at Berkeley.

[9] Precup D., Temporal Abstraction in Reinforcement Learning, Ph.D. Dissertation, **2000**, Department of Computer Science, University of Massachusetts, Amherst.

[10] Ramirez, N., and Jackson H., Application of neural networks to chemical process control, *Computers and Chemical Engineering*, **1999**; 37; 387-390.
[11] Riedmiller M., High quality thermostat control by reinforcement learning-A case study, in *Proceedings of the Conald Workshop 1998*, Carnegie Mellon University.
[12] Schmidt R. J., Industrial catalytic process-phenol production, *Applied Catalysis A: General*, **2005**; 280; 89-103
[13] Schoknecht, R. and M. Riedmiller, Learning to control a multiple time scales, *Proceeding of ICANN 2003*, Istanbul, Turkey, June 26 – 29, pp. 479 - 487
[14] Sutton, R. S., and Barto A. G., *Reinforcement Learning: an Introduction*, **1998**,The MIT Press, Cambridge, MA
[15] Sutton, R.S., D. Precup, S. Singh, "Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning". *Artificial Intelligence,* 1999; 112; 1 - 2; 181-211.
[16] Syafiie S., Tadeo F. and Martinez E., Softmax and ε-Greedy Policies Applied To Process Control, *IFAC Workshop on Adaptive and Learning Systems (ALCOSP04)*, August 2004, Yokohama, Japan
[17] Syafiie S., Tadeo F., and Martinez E., Model free intelligent control using reinforcement learning and temporal abstraction-applied to pH control, *IFAC Workshop Congress*, Prague, Czech, July 2005.
[18] Syafiie S., Tadeo F., and Martinez E., Model-Free Learning Control for Processes with Constrained Incremental Control, *Proceedings of IEEE International Conference on Control Applications, International Symposium on Computer-Aided Control Systems Design, International Symposium on Intelligent Control* (CCA/CACSD/ISIC), Munich, Germany, 2006a; pp. 826-831.
[19] Syafiie S., Tadeo F., and Martinez E., Model Free Learning control using of Neutralization Process using Reinforcement Learning, 2006b, Accepted to publish in *Engineering Application of Artificial Intelligent*.