Available online at www.sciencedirect.com
ScienceDirect
Engineering Applications of Artificial Intelligence 20 (2007) 767–782
Engineering Applications of
ARTIFICIAL
INTELLIGENCE
www.elsevier.com/locate/engappai

Model-free learning control of neutralization processes using reinforcement learning

S. Syafiie[a,*], F. Tadeo[a], E. Martinez[b]

[a]*Department of Systems Engineering and Automatic Control, Science Faculty, University of Valladolid, Prado de la Magdalena s/n., 47011 Valladolid, Spain*
[b]*Consejo Nacional de Investigaciones Científicas y Técnicas, Avellaneda 3657 3000, Santa Fe, Argentina*

Received 1 July 2005; received in revised form 26 October 2006; accepted 27 October 2006
Available online 8 February 2007

Abstract

The pH process dynamic often exhibits severe nonlinear and time-varying behavior and therefore cannot be adequately controlled with a conventional PI control. This article discusses an alternative approach to pH process control using model-free learning control (MFLC), which is based on reinforcement learning algorithms. The MFLC control technique is proposed because this algorithm gives a general solution for acid–base systems, yet is simple enough to be implemented in existing control hardware without a model. Reinforcement learning is selected because it is a learning technique based on interaction with a dynamic system or process for which a goal-seeking control task must be performed. This "on-the-fly" learning is suitable for time varying or nonlinear processes for which the development of a model is too costly, time consuming or even not feasible. Results obtained in a laboratory plant show that MFLC gives good performance for pH process control. Also, control actions generated by MFLC are much smoother than conventional PID controller.

Keywords: Learning control; Goal-seeking control; Process control; Intelligent control; Online learning; Neutralization process; pH control

1. Introduction

pH control in neutralization processes is a ubiquitous problem encountered in the chemical and biotechnological industries. For example, the pH value is controlled in such chemical processes as fermentation, precipitation, oxidation, flotation and solvent extraction processes. Also, control of pH in food and beverage production (such as in bread, liquor, beer, soy sauce, cheese, and milk production) is an important issue because the enzymatic reactions are affected by the pH value of the process and each has its optimum pH critical to the yield. Other pH control applications in industry, for example, are in the decomposition section of the Sucono/UOP phenol production 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).

In chemical processes, the pH is a measure of the hydrogen ion concentration, which determines the acidity/alkalinity of a solution. It can be defined as (for 25 °C, 1 atm)

$$\mathrm{pH} = -\log[\mathrm{H}^+], \tag{1}$$

where $[\mathrm{H}^+]$ denotes the hydrogen ion concentration in mol/l (more precisely, it should be the activity of the hydrogen ion). The pH of a solution is used as a measure of $[\mathrm{H}^+]$ by means of a potential difference in an electrolytic cell.

The dissociation of weak acids and bases or their salts involved in the solution determines the number of hydrogen ions. All weak species have the property, called buffering, to resist change in the pH. A weak acid, for example, is not completely dissociated, so it can absorb hydrogen ions by converting them to undissociated acid molecules (Kalafatis et al., 2005).

In most pH neutralization processes, the control of pH is not only a control problem but also comprises chemical equilibrium, kinetics, thermodynamics and mixing

*Corresponding author. Tel.: +34 983 184647; fax: +34 983 423161.
E-mail address: syam@autom.uva.es (S. Syafiie).

0952-1976/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2006.10.009

problems all of which must be considered (Gustafsson et al., 1995). These inherent characteristics of pH processes are an interesting and challenging one to be solved. An important problem is that the process buffer capacity varies with time, which is unknown and dramatically changes process gain, thus making it difficult for controller design. For example, if either the concentration in the inlet flows or the composition of the feed changes, the shape of the titration curve will be altered. This means that the process nonlinearity becomes time dependent and the system switch over several titration curves. Also, due to the nonlinear dependence of the pH value on the amount of titrated reactant, the process will be inherently nonlinear. As a result, it is difficult to develop an appropriate mathematical model of the pH process for designing a controller that can exhibit a good performance over a wide rage of operating conditions.

Many researchers have proposed control strategies for pH problems based on the titration curve. For example, Shinskey (1973) designed an adaptive control strategy by resorting to three regions in the titration curve. A general dynamic model for fast acid–base reaction was presented by Gustafsson and Waller (1983), using invariance reactions. Wright and Kravaris (1991) defined an alternative control objective using a strong acid equivalent of a mixture of electrolytes, which is linear in state, using a linear control law. As the reaction invariance is unmeasured online and the linear system is unobservable, Henson and Seborg (1994) presented an indirect adaptive nonlinear controller. The controller was designed by augmenting the state feedback controller and combining an input–output linearizing controller with reduced-order, open loop observer which provides online estimates of the reaction invariance.

As alternative methods to overcome the nonlinearities and time-varying characteristics of pH processes; Sung and Lee (1995) proposed online identification, using a setpoint change for PID autotuning; Norquay et al. (1998) used a Wiener model for representing nonlinear process behavior and designed a controller using a model predictive control law; the linearization of a pH system using the Wiener model was also proposed by Kalafatis et al. (2005), followed by the application of linearizing feedback pH control.

Alternative strategies based on intelligent control have been proposed by some researchers; applying fuzzy control, neural networks or different combinations of intelligent and model-based methods. For example, fuzzy logic (Sabharwal and Chen, 1996; Biasizzo et al., 1997) and neural networks (Ramirez and Jackson, 1999; Loh et al., 1995) have been implemented on pH control. 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. Neural networks and adaptive controllers (Krishnapura and Jutan, 2000), PID using linearization through neural networks (Chen and Huang, 2004), and genetic algorithms combined with internal model control (Mwembeshi et al., 2004) have been reported to address the problem of proper control the pH in a chemical process.

As has been discussed in the above referenced works, tight and robust pH control is often difficult to achieve due to the inherent uncertain, nonlinear and time-varying characteristics of pH neutralization processes. Unfortunately, the above-mentioned literature on pH control approaches presents some weaknesses, such as

- complexity of the control structures (which could be difficult to implement on existing control systems),
- conservativeness (the controller takes a long time to reject disturbances and reach the desired reference),
- difficulty of tuning, which makes it a time-consuming task (some of these controllers have many tuning parameters, or require many experiments before they can be applied to the process).

This paper discusses an alternative approach to solve the pH control problem by applying *model-free learning control* (MFLC), based on *reinforcement learning* algorithms (Sutton and Barto, 1998), and *hierarchical reinforcement learning* (Sutton et al., 1999). These control algorithms are based on learning directly from the closed-loop behavior of the plant.

Compared to other control techniques based on learning, reinforcement learning has some clear advantages:

- It can optimize the control signal upon choosing actions during the online interaction between an *agent* (controller) and an *environment* (process/systems). This can be seen in Fig. 1.
- It is possible to include previous process knowledge in the controller design.
- The control algorithm is quite simple from a computational point of view, so it can be implemented using low-cost hardware.
- It is possible to understand reinforcement learning from an optimal control point of view, which makes it attractive for control and plant engineers.

The first part of this paper examines the reinforcement learning idea in detail. It is followed by the *Q*-learning

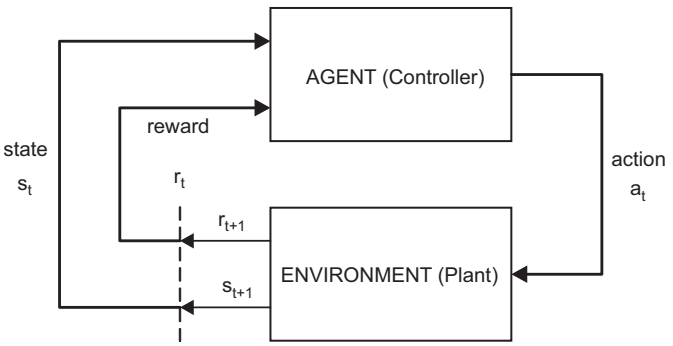

Fig. 1. Cycle of reinforcement learning.

algorithm and an illustrative simulated example of the pH control problem. The second part introduces hierarchical reinforcement learning in order to speed up learning. The hierarchical reinforcement learning idea is based on *temporal extended action*, in the form of *multi-step action* (MSA) and *macro-actions* algorithms. Finally, for comparison of the proposed algorithms, online results on a pilot plant are shown and compared to the PID controller.

2. Model-free learning control algorithm

One of the main advantages of using reinforcement learning for automatic control is that the controller modifies its control policy as it gathers experience online, which makes it adequate for time-varying or nonlinear systems. After a revision of reinforcement learning algorithms, the proposed MFLC is presented.

2.1. Reinforcement learning algorithms

The learning algorithm in reinforcement learning emphasizes the *interaction* between an active decision-making agent (intelligent controller) and its target dynamic system (Fig. 1). In the latter, a desired behavior or control goal is permanently sought despite imperfect 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 (which corresponds to the controller in conventional process control) interacts with its environments (the process or plant). They interact on a continuous basis: the agent selects an action and then the environment responds to the executed action and presents a new situation to the agent. These responses of the environment are communicated to the agent through a scalar reinforcement signal, which indicates that the action chosen by the agent in the current state is *good* or *bad*. This is depicted in Fig. 1, where the dashed line represents a unit delay.

There are four essential factors for dealing with the reinforcement learning problem;

- A *policy* defines the agent's behavior of *what to do*, i.e. what action to take at each state.
- A *reward* function specifies the overall goal of the agent that gives the clue concerning *what is good to do* and what is not a desirable outcome following an action.
- A *value* function is the value of a state or a state-action which indicates how good a controller's behavior is from the point of view of the control goal.
- The *model of the environment* gives a predictive capability for state to state transitions depending on the applied actions.

Based on this, reinforcement learning can be defined as an on-line learning ('*learning what to do by doing*') approach to find an optimal decision policy in multi-stage decision problems, i.e. how to map perceptions of process states or histories to control actions, so as to maximize an externally provided scalar *reward* signal. Compared to other learning approaches, in simple terms, it is possible to describe reinforcement learning (Sutton and Barto, 1998) as an automatic learning method based on the use of "critic" (instead of a "teacher", as in other learning methods). The only feedback provided by the "critic" is a scalar reinforcement signal, which can be thought of as a reward or a punishment.

2.2. Q-learning algorithm

From the different proposed algorithms of reinforcement learning, this paper proposes the application of MFLC based on the Q-learning algorithm for pH process control as this algorithm is guaranteed to converge to the correct Q-values, with the probability one if the environment is stationary, and depends on the current state and the action taken in it. In Q-learning, the learning task is based on estimating the *cumulative future reward* or *value*, which makes it interesting for process control. This predicted value is used for selecting the action from those available in each visited state. The value of the reinforcement at each time reflects the control objectives, which might involve cost, errors, or profits (Sutton and Barto, 1998).

In the proposed algorithm, the agent should take future decisions into account to assess the goodness of the current decision or action. If the sequence of actions is infinite, discounted return criteria will be used: a discount factor (γ, $0 \leqslant \gamma \leqslant 1$) is introduced, to weight more heavily near-term reinforcements. In addition, the objective of the agent is to maximize the return function ($R(t)$), which represents the expected total value at time t applying a given action in the present state. It can be evaluated from the reinforcements at time $t+k$ (denoted by r_{t+k}) as follows:

$$R(t) = \sum_{k=0}^{\infty} \gamma^k r_{t+k}. \tag{2}$$

To clearly distinguish between the effect on R_t of the action at time t (denoted a_t), from the effect on R_t of decisions to be taken later in sequel, the action-value function $Q(s_t, a_t)$ 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:

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

This is the central part of the algorithm: the estimation of the so-called Q-function gives the benefit of applying action a_t when the system is in state s_t. Eq. (3) can be rewritten as an immediate reinforcement plus a sum of

future reinforcements.

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

The agent observes the present state, s_t, and selects and executes an action, a_t, according to the evaluation of the return that it makes at this stage, the value function is updated. The benefits, both now and in the future, must take into account: when action a_t has been selected and applied to the plant, the system moves to the next state, s_{t+1}, and receives a next reinforcement signal, r_{t+1}. Hence, the equation is updated by substituting the sum of future reinforcements with the estimated value function. The expectation of taking action, a_t, in the next state, s_{t+1}, is replaced by a fraction of the difference. The update equation can be written as

$$Q_\pi(s_t, a_t) \leftarrow Q_\pi(s_t, a_t) + \alpha_t[r_{t+1} + \gamma \max_{b \in A_{s_{t+1}}} Q_\pi(s_{t+1}, b) - Q_\pi(s_t, a_t)], \quad (5)$$

where

- $A_{s_{t+1}}$ is the set of possible actions in the next state.
- The *learning rate*, $0 \leqslant \alpha \leqslant 1$, is a tuning parameter, that can be used to optimize the speed of learning (Although too small learning rates might induce slow learning, while too large learning rates might induce oscillations).
- The *discount factor*, γ, is used to weight near term reinforcements more heavily than distant future reinforcements: If γ is small, the agent learns to behave only for short-term reward; the closer γ is to 1 the greater the weight assigned to long-term reinforcements.

The Eq. (5) is known as the Q-learning algorithm which is used to improve optimal control.

The agent knows exactly neither the optimal value function nor the correct estimation of the dynamic environment. If the agent knows this value correctly, the policy can select a greedy action that maximizes the value function at each state. If this estimation and prediction are good enough, therefore, a good policy is greedy action; this is called *exploitation*. However, the agent does not know the correct optimal value function. In order to know the optimal value functions, the agent should execute trial actions, i.e. actions that are not optimal with respect to the current value function; this is called *exploration*.

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

It is noteworthy to clarify what on-line learning is about. As new experience is acquired and incorporated into the control policy, the controller behavior is continuously adapting to the process dynamics. Hence, the MFLC is learning "on-the-fly" how to act better to achieve the control goal (defined by rewards).

2.3. MFLC algorithm

The proposed MFLC algorithm is an online learning algorithm based on the value function. The value function, which is a mapping of history of visiting states and executing actions, gives a clue for the agent to select an action in given state. The agent takes into account that taking action in the current state predicts it will be giving the maximum cumulative future reward. This predicted value is used by policy for selecting an action from those available in each visited state. The policy implicitly observes the future reward upon choosing an action in state, s_t. Choosing an action in s_t will be criticized as "best or worse", based on observation of the future state and reward. This MFLC approach can be seen in Fig. 2. The value of the reinforcement at each time reflects the control objectives, which might involve cost, errors, or profits (Sutton and Barto, 1998).

The algorithm developed for the learning system is as follows:

1. Observe the state s_t based on tracking error.
2. Select an action a_t, from the set of available actions in state s_t.

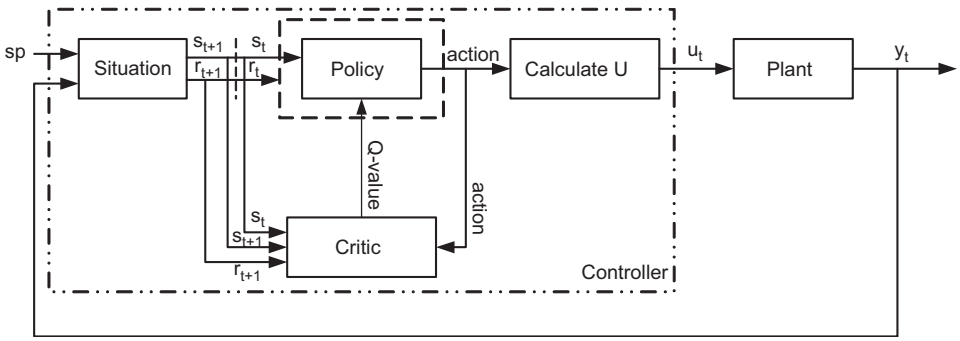

Fig. 2. MFLC architecture based on Q-Learning.

3. Calculate control signal, u_t, and send to actuator.
4. Receive immediate reward r_{t+1} and sense the next state.
5. Update the entry in the Q-value table for state s_t and action a_t, and save it at $Q(s_t, a_t)$ using Eq. (5).

2.4. Illustrative simulated example

This illustrative example study uses the mathematical model proposed by Kwok et al. (2003). The dynamic model of the process is only used to study, via simulation, how the proposed control algorithm performs in controlling a pH process, whereas the learning control algorithms are model-free. The dynamic of the pH neutralization process can be described by a quasi-linear equation. This model assumes that the system is in an ideal condition without any pollutant influence. It is represented by the following differential equation:

$$V\frac{dG}{dt} = -(F_A + F_B)G + C_A F_A - C_B F_B, \quad (6)$$

where G is the distance from neutrality and is given by $G = [H^+]$–$[OH^-]$, V is the tank volume, C_A and C_B are the concentration of acid and base, respectively and F_A and F_B are the flow rate of acid and base, respectively. The value of G is zero at neutral point, pH = 7. The measurement equation is derived as follows:

$$pH = -\log_{10}(G + \sqrt{G^2 + 4K_W}) + \log_{10} 2, \quad (7)$$

where $K_w = 10^{-14}$ (mol/l)^2 is taken. The high nonlinearity is introduced by this output equation between the measurement pH value and state G. To make the simulation behave as a real plant, a *white noise* of pH measurement is introduced based on Gaussian distributions. The distribution is weighted by 0.1.

From the above model, the operating conditions used in the simulation are listed in Table 1. The manipulated flow is varied from 0.00 to 0.21 l/m and the acid process flow continuously supplied into the continuous stirred tank reactor (CSTR) is 0.11 l/m.

The titration curve of the strong acid–strong base system generated from Eqs. (6) and (7) is presented in Fig. 3. It clearly shows that the system is highly nonlinear when the pH varies from 2 to 12. Therefore, controlling the system in the whole range is difficult.

Table 1
Operating parameters

Strong acid flow	F_A	0.11 l/min
Strong base flow	$F_{B\ min}$	0.001 l/min
	$F_{B\ max}$	0.21 l/min
Acid normality	CA	0.001 mol/l
Base normality	C_B	0.001 mol/l
Tank volume	V	1 l

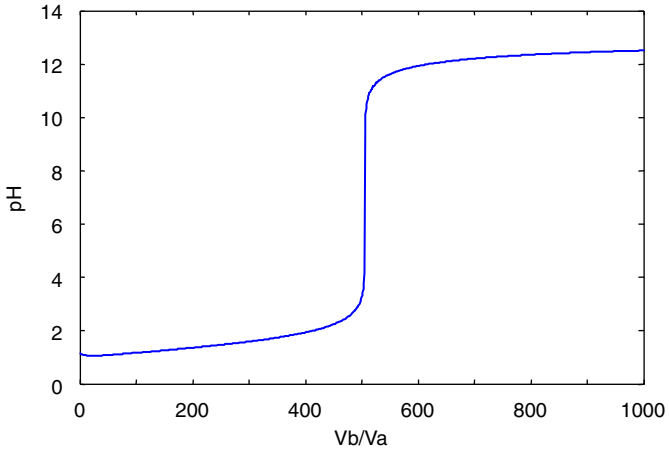

Fig. 3. Titration curve strong acid–strong base system of the model.

2.5. The environment definition

The environment is described as follows: the symbolic states are defined to be 5, in which state 3 is a goal state. State 1 is when the pH is higher sp + 0.5, state 2 is smaller than sp + 0.5 and higher than sp + 0.2, whereas state 3 is smaller than sp + 0.2 and higher than sp−0.2, and states 4 and 5 follow the rule of states 2 and 1, respectively. Every state has 2 actions except for the goal state which has only 1 action. The positive reward is introduced when the system reaches goal state and negative reward 1 when the system is outside the goal state. Details of the environment definition will be discussed later, in the online application.

2.6. The agent description

The agent defined for the illustrative example is as follows: The value of the *meta-parameters for the agent* were selected to be: discount factor, $\gamma = 0.90$ and learning rate, $\alpha = 0.1$. The agent selects an optimum action based on $(1-\varepsilon)$, where ε is 0.1, to allow the agent to explore other actions which have no maximum value function.

The selected action is weighted to have a control signal, which is calculated as

$$u_t = u_{t-1} + k(a_t - a_w), \quad (8)$$

where a_t is the action chosen by the agent and a_w is the *wait action* which the action is defined to have no manipulation of 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. In this case, action two is called wait action. The controller gain, $k = 1 \times 10^{-7}$, is a tuning parameter that can be selected to weight how much to increase or decrease previous control signal upon chosen action. The overall available actions are defined to be 5.

The responses of the limited available actions in the simulated process are presented in Fig. 4, which clearly shows that the agent quickly learns to reach the goal state.

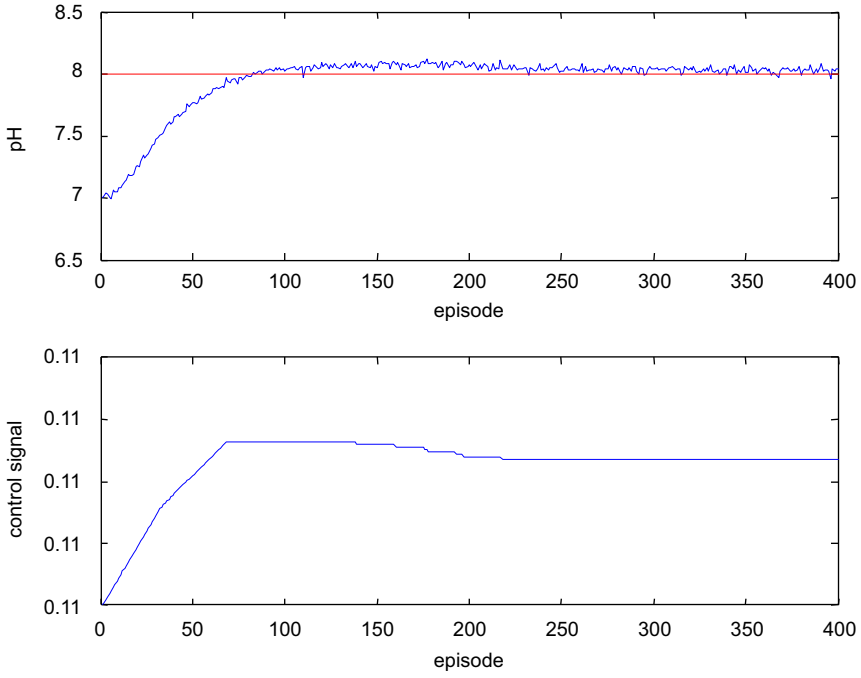

Fig. 4. Responses and control signal of strong acid–strong base system.

The process, after learning, remains in the goal state and close to the desired setpoint. More detailed of the MFLC application will be discussed later.

2.7. Hierarchical reinforcement learning

In standard reinforcement learning frameworks discussed above, a learning agent interacts with an environment at some discrete time scale ($t = 1,2,3,\ldots$). At each time step, t, the environment is in some state, s_t. In the current state, s_t, the agent selects an action, a_t, and executes it, the environment responds to the action and presents the agent the state transition, s_{t+1}, and the reward, r_{t+1}. State transition is depend on the preceding state and action, also may depend on it in a stochastic fashion. The mapping from state to action called *policy* is to learn to maximize discounted return by the agent.

Standard reinforcement learning algorithm, as in Q-learning, is difficult to implement in real complex problems: these algorithms scale slowly 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, hierarchical approaches based on *temporal abstraction* have been proposed to keep tractable the number of decision to be taken to reach the goal state. Temporal abstraction can enable the agent to improve performance more rapidly and to use this solution to reduce the number of trials required to obtain acceptable performance. 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 a *primitive action*. Semi Markov decision processes (SMDPs) is the theory used to deal with temporal abstraction as a minimal extension of the reinforcement learning framework. SMDPs is an appropriate MDP for modeling continuous-time discrete-event systems.

Several reinforcement learning algorithms resorting to hierarchical temporal abstraction approaches have recently been proposed: hierarchy of abstract machine (HAM) (Parr, 1998); MaxQ (Dietterich, 1997) and multi-step-actions (MSA) (Riedmiller, 1998). The first two methods are based on the notion that the whole task is decomposed into subtasks each of which corresponds to a subgoal. MSA is a method that enables the agent to learn to experience multiple-fixed-time-scale, for example for m-time-step termination condition (Riedmiller, 1998). The *macro-actions* used in this paper refers to the temporal extended action proposed by Sutton et al. (1999), in which they use the term of *Option*. Options may be either multiple step policies or primitive actions while macro-actions are restricted to temporally extended actions (McGovern and Sutton, 1998).

The discussion below focuses on MSA and macro-actions algorithms applied to the pH process. Both algorithms have no decomposition in subprograms, also decomposition in the process to be controlled (pH process) is unknown in advance.

2.7.1. Multi step actions

The concept of MSA (Schoknecht and Riedmiller, 2003) is applied to pH control because it is suited to systems where no decomposition in subproblems 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 enables an intelligent control to learn a control policy by using multiple time scales simultaneously. The MSA consists of several identical actions on the primitive time scale. This algorithm is possible to increase in responsiveness and add flexibility to the controller behavior. In addition, 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 problems, such as pH control problem.

The idea of MSA is based on a set of all multiple step actions of degree m, defined as $A^{(m)} = \{a^m | a \in A^{(1)}\}$, 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. This can be seen in Fig. 5. The selected action, a_t, will be applied for m time-steps. The next state, s_m, after the execution of action, a_t, for m time-steps, will be used by the agent to select a new action.

The concept of MSA can be integrated into learning algorithms, such as Q-learning. For example, when the agent executes action a^m of degree m in state s, the environment makes transition to state s_n after m time steps. The state-action value can be updated as follows:

$$Q(s_t, a^m) \leftarrow Q(s_t, a^m) + \alpha[r_{sa^m} + \gamma^m \max_{a_n \in A} Q(s_n, a_n^m) - Q(s_t, a^m)], \quad (9)$$

where

$$r_{sa^m} = \sum_{\tau=i}^{i+m-1} \gamma^{\tau-i} r_{s_\tau a},$$

where $Q(s_t, a^m)$ is Q-value for state-action in time t, and $Q(s_n, a_n{}^m)$ is Q-value for next state, α is learning rate, and γ is discount factor. When action a^m with degree m is selected in s_i, the environment makes transition to s_{i+m} with reward $r_{s_i a^m}$. When executing a^m, all actions a_i, $i = 1, 2, \ldots, m-1$ are executed implicitly. The transition from s_i to state s_{i+m} contains all information necessary to update the Q-values for those lower-level actions at all intermediate states.

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

1. Observe the state s_t
2. Select an action a_t, this action is chosen from state s_t using ε-greedy policy
3. Apply the selected action a_t for n time steps
4. Do until terminating condition $m = n$
   4.1. Read the resulting state s_{t+m}
   4.2. Update Q-value using Eq. (9)

2.7.2. Macro-actions

Macro-actions are policies with termination conditions. Each macro action is specified by a closed-loop policy, which determines the primitive action when the macro actions are in force, and by a completion function, which determines when the macro action ends. At each time step, the agent can choose either a macro action or a primitive action, unless it is already executing a macro action. Once the agent has chosen a specific macro action, it selects the primitive actions in accordance with the macro-action's policy until the macro-action's termination condition is satisfied. The idea is drawn in Fig. 6. The Selected macro action, a_t, will be executed until the termination condition is satisfied. After that, the agent observes the next state, s_m, and selects either primitive or macro actions.

To provide for learning when selecting macro-actions, the notion of the optimal action-value function is extended to Q^*, to include macro-actions. This extended action-value function can be defined as $Q^*(s, a_m)$ for each state s and macro-action a_m, as the maximum expected return given that the agent start macro-actions a_m in state s. This definition naturally leads to update rule: upon each termination of a macro action, its value is updated using the cumulative discounted reward received while executing the macro-actions and the maximum value at the resulting state. More precisely, after a multi-step transition from state s_t to state s_n using macro action a_m, the approximate action value $Q(s_n, a_m)$ is updated by

$$Q(s_t, a_m) \leftarrow Q(s_t, a_m) + \alpha\left[r + \gamma^n \max_{a \in A} Q(s_n, a) - Q(s_t, a_m)\right], \quad (10)$$

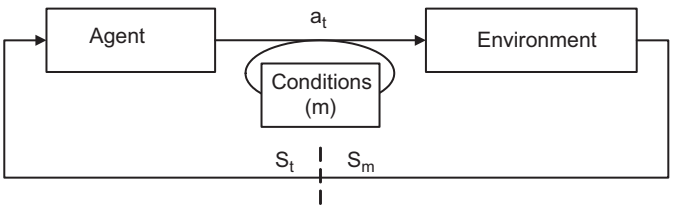

Fig. 5. Multi-step actions idea.

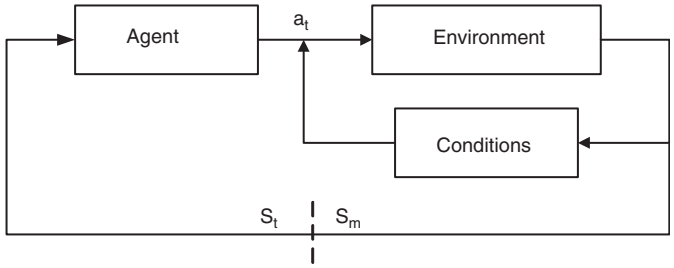

Fig. 6. Macro actions idea.

where the max is taken over both action and macro-actions, and

$$r = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n}$$

α is positive step size parameter or learning rate and γ is a discounted factor.

The macro-actions algorithm developed for the learning system is as follows:

1. Observe the state s_t
2. Select either primitive or macro action
3. If macro action is selected, apply the selected one until termination condition is satisfied
4. Update Q-value using Eq. (10)

Compared to standard reinforcement learning, these hierarchical reinforcement learning algorithms are proposed for the pH process because it can extract more training examples from the same experiences of taking action and applying it until the termination condition is reached. By experiencing a sequence identical actions applying for pH process, the agent can speed up learning and planning to maintain the process in the desired pH value.

MFLC uses the zero initial condition of Q-function. This value is updated by time upon taking actions and the process behavior. When the environment changes, for instance, due to setpoint changes, the action-value function is immediately reset to its initial condition. Resetting Q-value to the initial condition makes it possible for the agent to learn new environment without any influence from learning the past environment.

3. Application to neutralization process

This section describes the sequence of steps proposed in the implementation of the MFLC algorithm to a general neutralization process.

3.1. States and reward

In a neutralization process, the main control objective is to maintain the pH inside a band of $\pm\delta$ around the desired setpoint (the width of this band is defined by measurement noise in the process and the allowed tolerance). Therefore, the desired reference may be within this range and the tolerance could be allowed within this range. This band is defined as the goal state. In MFLC, the rest of the states corresponds to values of the pH uniformly distributed outside this band, as depicted in Fig. 7.

To classify the pH measurement where the process is in current time and to select an action available in each state, based on practical experience, 21 symbolic states are proposed, where the *goal state* (the state towards which the dynamics must converge) is state number 11, which corresponds to the desired pH band. The actions in every state are defined as shown in Fig. 8 and in Table 2, where

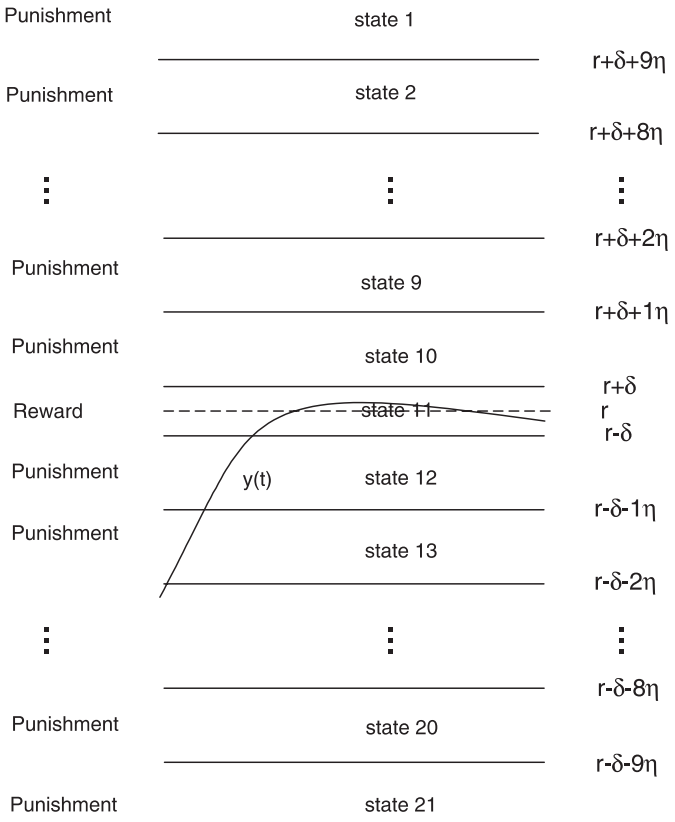

Fig. 7. State definition and control objective and definition of states: dash line presents a reference.

the goal state has only one possible action, namely no control action. When the system is in the goal band, the agent does not need to learn the environment, therefore, the agent just has one possible action. In other words, when the process is in the goal band, the agent successfully learns the environment. This selection between a number of states and actions corresponds to a trade-off between controller complexity and control accuracy.

The probability of the system moving 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 agent chooses action 2, the process may move either to state 2 or to another state, or stay in state 1.

States 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 Fig. 7. The goal of the control task is to maintain the process in the goal state, or return it to the goal state, despite the occurrence of any disturbance. To achieve this, maximum reward is introduced in the goal state. When the system is outside the goal band, the controller is punished by a negative reward. This reward function is applied in each state as a single number, as shown in Eq. (11).

$$\text{reward} = \begin{cases} 1 & \text{if pH is in goal band} \\ -1 & \text{otherwise} \end{cases}. \tag{11}$$

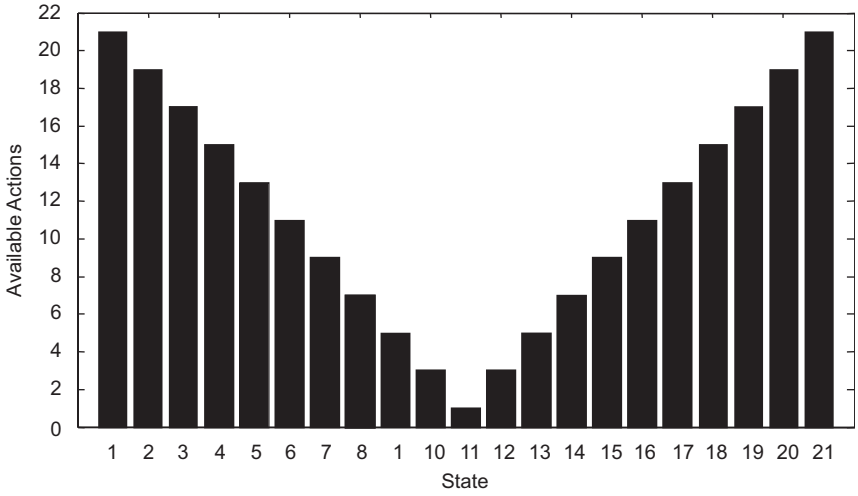

Fig. 8. Available actions in every state.

Table 2
Available actions in every state

State	Actions
1	21
2	19
3	17
4	15
5	13
6	11
7	9
8	7
9	5
10	3
11	1
12	3
13	5
14	7
15	9
16	11
17	13
18	15
19	17
20	19
21	21

3.2. Termination conditions

The MSA termination condition is applied after applying a sequence of identical actions for m time steps. In this study, the identical actions will be applied for 10, 20 and 30 times for every executing action.

The termination conditions for macro-actions application are when the process reaches, over or goes far away from the goal band. The idea is shown in Fig. 9. Therefore, in this application, there are 3 termination conditions introduced. For instance, when the process is in state 8, the agent can either select macro actions or primitive action. If the agent selects macro actions, the agent will execute this until the termination conditions are reached. The idea for termination condition is if the system, for example, is in state 8 the agent selects macro actions and makes transition to the next state and receives a reward. If the next state is goal state the macro actions are terminated or if the next state is pass goal state, for example the next state is state 12, the macro actions are terminated. Also, if the next state goes far away from goal state, for example next state is state 5, the macro actions terminate. However, if the next state is neither goal state nor passes goal state nor goes far away from goal state, the agent continues executing macro actions. Therefore, there is no macro actions in the goal state, because the agent has already learned the system.

4. Experimental results

This section describes the plant setup, discusses the application of MFLC to various pH process controls on a laboratory plant and discusses some online results.

4.1. Description of the experimental setup

The experimental setup, shown in Figs. 10 and 11, consists of a CSTR where a process stream, such as diluted sodium acetate ($NaCH_3COO$) to be maintained at a certain pH value, is titrated with a solution of hydrochloric acid (HCl). The solution of sodium acetate is prepared and stored in a storage tank. Different concentrations and pH values of the process stream sodium acetate can be achieved by adding varying amounts into this storage tank. This process stream is fed from the storage tank using a pump. The reaction occurs in the CSTR, which has overflows, therefore the volume of liquid in the tank can be considered constant (1 l).

The control variable u_t is the flowrate of the titrating stream (normalized to the maximum value), which is applied using a peristaltic pump (ISMATEC MS-1 REGLO/6-160).

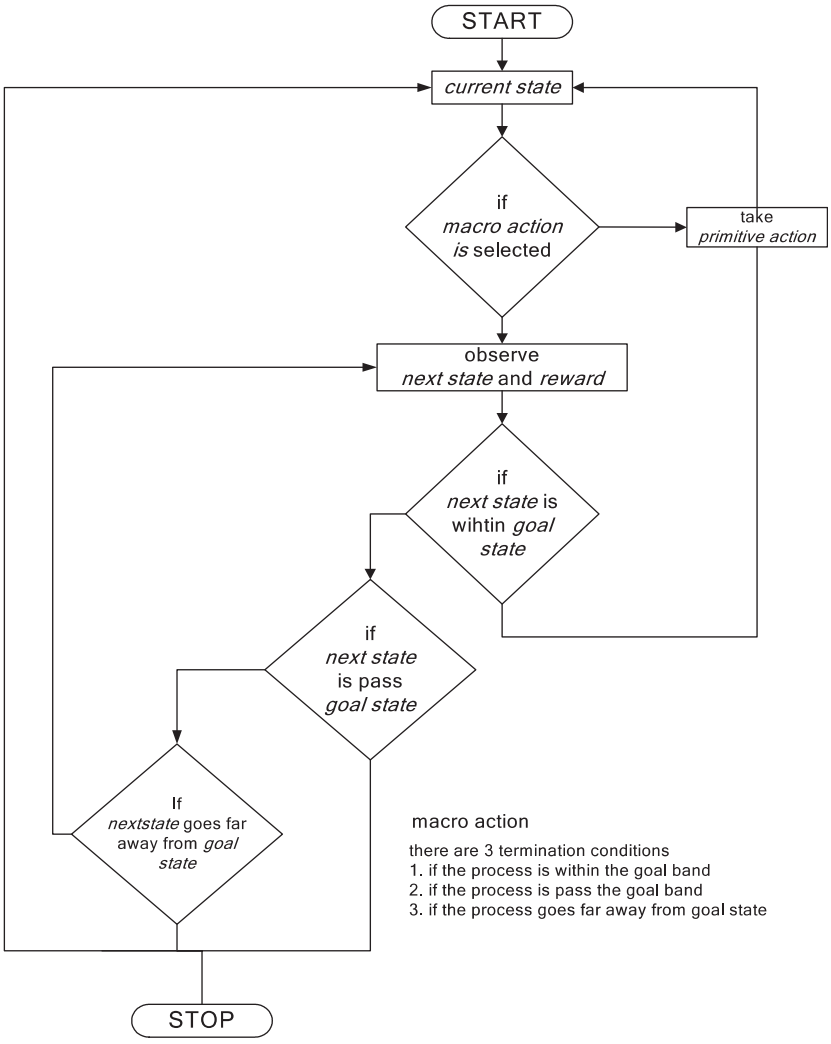

Fig. 9. Implementation of macro action and definition of termination conditions.

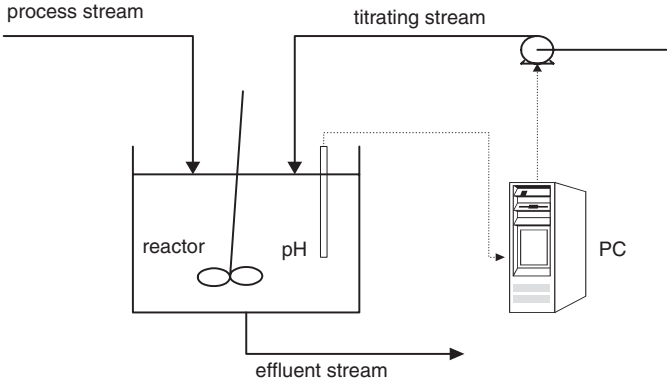

Fig. 10. pH neutralization process plant.

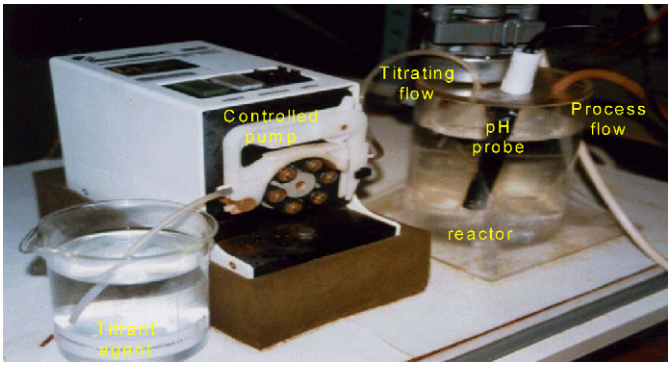

Fig. 11. pH neutralization real laboratory plant.

The output variable, y_t, is the logarithmic hydrogen ion concentration (pH) in the reactor. 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. The pH value in the mixture is measured using an Ag–AgCl electrode (Crison 52-00) and transmitted using a pH-meter (Kent EIL9143). The electrode dynamic response exhibits appreciable and asymmetric inertia. The pH measurement and the control signals are transmitted through an A/D interface (ComputerBoards CIO-AD16, 0–5 V). The plant is controlled and monitored from a personal computer, using Matlab and the Real-Time Toolbox for online control.

4.2. Parameter selection

In this study, the solution of sodium acetate is prepared with variable concentrations (to check the robustness of the controllers) that change during the operation. The titrating solution of hydrochloric acid is prepared for concentrations of $\pm 1\%$ in volume.

As mentioned above, the defined system has 21 states. In this implementation, the parameters δ and η are selected to be $\delta = 0.05$ and $\eta = 0.1$, based on the level of measurement noise and the desired pH operating range. From the parameters δ and η, it can be defined that the agent is in state 1 when the measured pH is higher than $r+\delta+9\eta$. State 2 is defined when the pH is lower than $r+\delta+9\eta$ and higher than $r+\delta+8\eta$. The rest of the states are defined accordingly (see Fig. 7).

In MFLC, the ε-greedy policy, is applied for choosing an action in every visited state of the pH process. The parameter ε used in the ε-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 maximum action-value) will be selected with a probability of 1 out of 10, which represents a good compromise for the plant, given its time-varying and nonlinear characteristics (less experience would be necessary if the plant were linear and the concentration less uncertain).

The value of the *meta—parameters* selected for the agent are: discount factor (γ) of 0.98 and learning rate (α) of 0.1. These learning parameters chosen for the agent are tuned to weight rewards and speed up learning, based on compact dealing of slow learning, oscillations, short and long-term reinforcement.

4.3. Control actions

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

$$u_t = u_{t-1} + k(a_w - a_t), \tag{12}$$

where a_t is the optimal action (chosen by the agent from those available actions in every visited state), and a_w is a numerical value of the wait action (where there is no variation of the previous control signal). As the process stream is a base, from Eq. (8) wait action and chosen action switch places as shown in Eq. (12).

From the defined actions, the agent selects an action in each visited state. Following Eq. (12), the variation of numerical value of the chosen action is then weighted by a gain to increase (or decrease) the previous control signal. The controller gain, k, of Eq. (12) is selected in different experiments to be 1×10^{-5}, 2×10^{-5} and 3×10^{-5} to study the effect of gain. This control signal is then bounded on the range 0.04–1, which corresponds to the range where the actuator operates correctly.

4.4. Experimental results and discussion

4.4.1. MFLC algorithms for $NaCH_3COO$–HCL system

Application of the proposed MFLC controller to the laboratory plant showed good result. The responses of the plant to some changes in setpoint and comparison with a constant PID controller can be seen in Fig. 12a for the sodium acetate–hydrochloride acid system. The PID controller was tuned based on normal conditions at pH = 5, where correction gain and proportional gain are chosen to be 0.01 and 0.001, respectively. Derivative time and integral time are selected to be 1. The comparison shows that the responses of the proposed MFLC algorithm settle in reference faster than the PID controller. The responses of the plant show that MFLC controller based on reinforcement learning algorithms is much closer to the reference signals while the PID controller has higher overshoots. The control signal, Fig. 12b shows that the MFLC controller manipulates the actuator in a smoother way than a PID controller. Since the MFLC allows a tolerance error of the process being on the band, the control signal is smoother when the process is close to the reference and within the goal band.

Another set of experiments are done by alternative gain selections, k, as mentioned above. The different gains (1×10^{-5}, 2×10^{-5} and 3×10^{-5}) are applied to study the effect of the gain influence. The responses plotted in Fig. 12c show that even very small changes in the gain give slightly different responses of the plant due to the inherent nonlinearity of the process. Among the gains considered during the experiment, it shows that with the MFLC controller at gain $k = 1\times10^{-5}$ the responses of the process are found to be fairly satisfactory. This gain shows that the responses of the plant settle on the goal state faster than when using the other two gains. This can be seen in detail in Fig. 12.

From the control signal (shown in Fig. 12b), it can also be seen that the agent learns to maintain the pH in the goal band around the reference by adequately increasing and decreasing the control signal sent to the actuator. The agent increases or decreases the control signal when the process is outside the goal band. When the system is in the goal band, the control signal is maintained, allowing the process to remain within the tolerance error (the width of the goal band).

4.4.2. MSA algorithms for $NaCH_3COO$–HCL system

From the defined system, which has 21 states and 205 actions, the MSA algorithms were developed and applied to pH control. The online responses of the applied algorithms for $NaCH_3COO$–HCl system can be seen in Fig. 13. To study the effect of simultaneously taking and executing the identical actions, the MSA algorithm is applied for 10, 20 and 30 identical actions.

The online comparison responses of applied identical actions to the pH process can be seen in Fig. 13a. It clearly shows that the responses are good enough and lay close

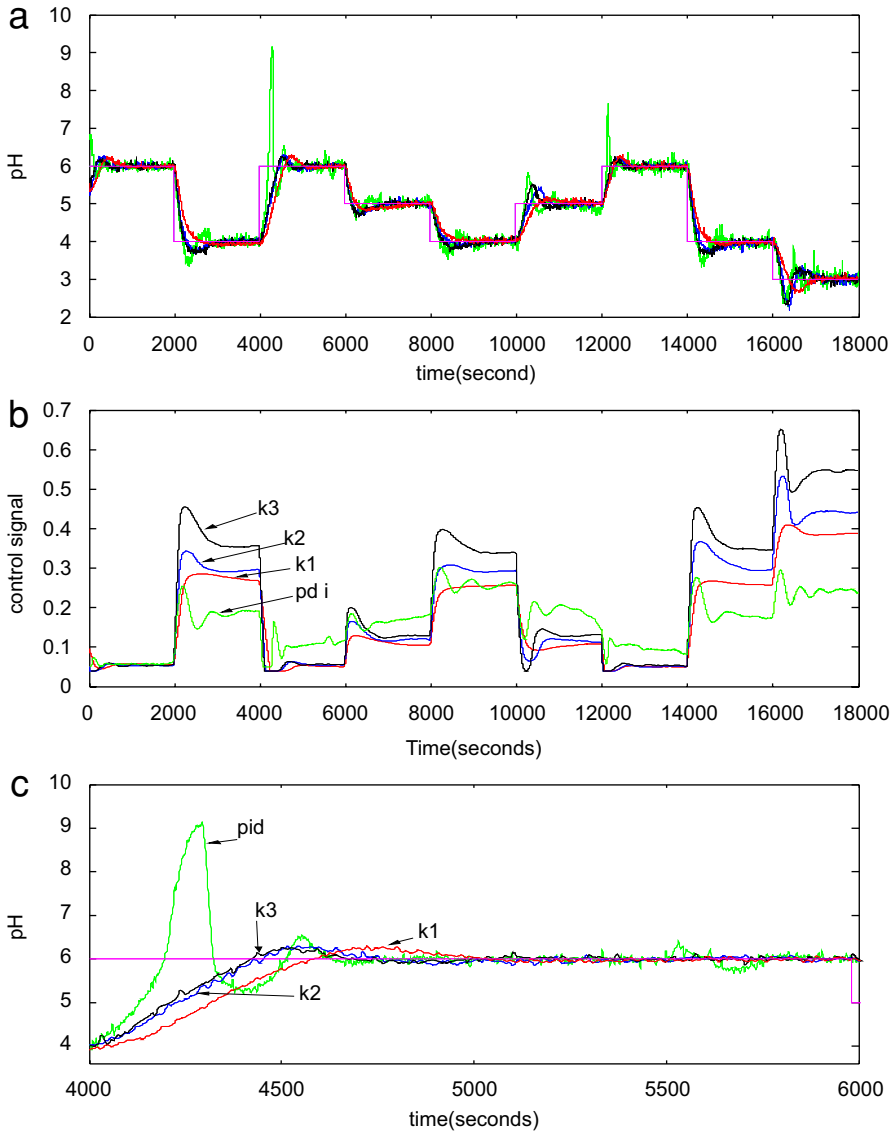

Fig. 12. Online application MFLC for $NaCH_3COO$–HCl systems, Q-learning standard for $k_1 = 1 \times 10^{-5}$; $k_2 = 2 \times 10^{-5}$; $k_3 = 3 \times 10^{-5}$ and compare to PID: (a) online responses, (b) control signal and (c) systems for 4000–6000 s.

to the reference. The MSA responses of applying 30 step-actions show that the peak is a little higher in some cases, and lower in other cases, but that overall it is reasonably well. However, the responses of appliying 20 step-actions are suitable for these systems and better than others.

The pattern of the control signal for these MSA algorithms is similar (Fig. 13b). The algorithms manipulate the controller smoothly.

4.4.3. Macro-actions algorithms for $NaCH_3COO$–HCL system

The application of macro actions with three termination conditions have been applied to pH process. The responses of the MFLC application for controlling pH process are shown in Fig. 14 comparing to *Q*-learning standard and MSA algorithms. The responses show that the online results of macro actions reach the goal band faster, but produce oscillations around reference. This is because the control keeps on increasing or decreasing the control signal with the same increment from the first time macro actions are selected even though the system is close to the goal band.

Due to the present of titrant (acid) when the system reaches the goal band is higher or lower than it is needed for reaction, the process behaves to response to the titrant and presents the situation, and the controller acts to control the situation by selecting actions (Figs. 14a and 15). It clearly shows that the controller sharply increases or decreases the control signal until the system reaches the goal band. By the moment, the process responses to the present of reactant, the controller acts to maintain the process by adequately reducing and increasing the control signal.

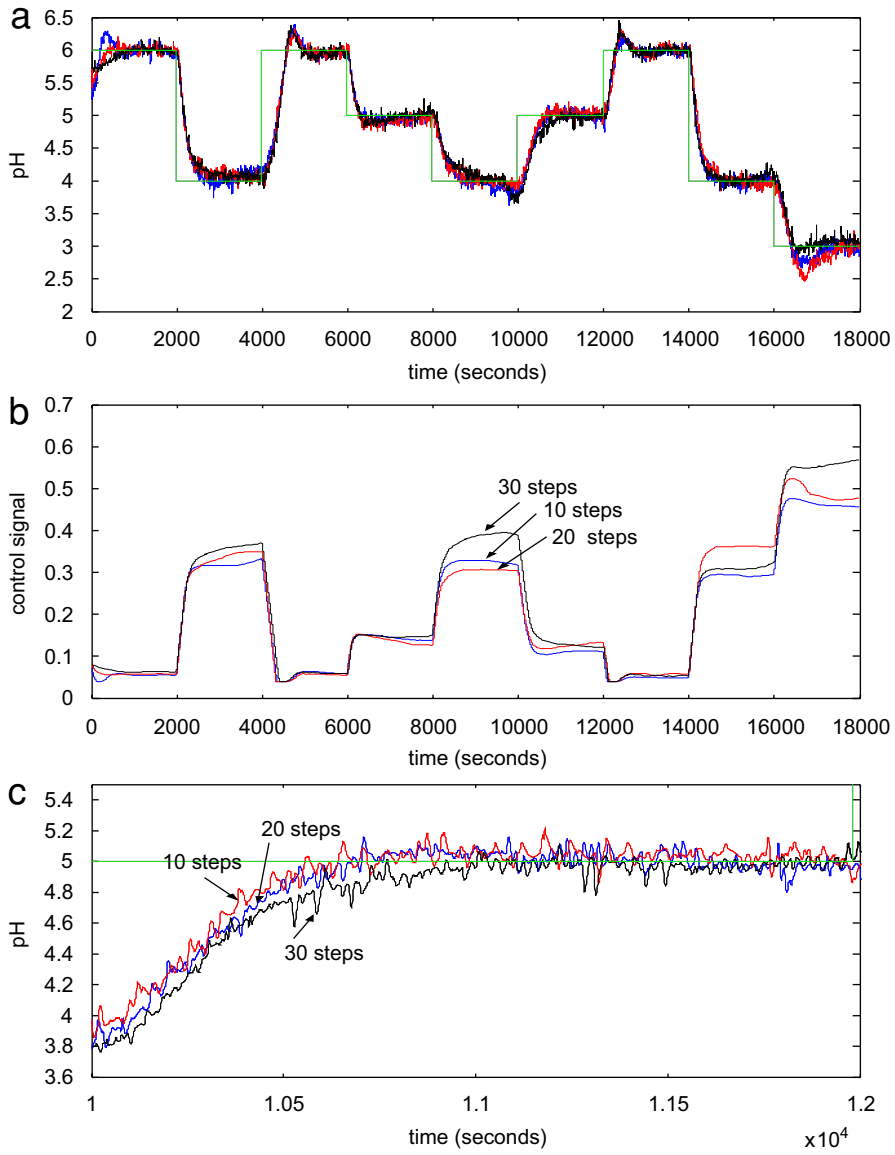

Fig. 13. Online application MFLC for $NaCH_3COO$–HCl systems, MSA for 10, 20 and 30 step-actions: (a) online responses, (b) control signal and (c) systems for 10000–12000 s.

4.4.4. MFLC algorithms for sodium hydroxide (NaOH)–HCL system

To check the robustness of the MFLC controller, when applied to other neutralization processes, the algorithm has also been applied to neutralize a process stream for a strong acid–strong base system. The process stream of NAOH is titrated with HCl. This solution (NaOH) is prepared with an unknown concentration and the titrating stream (HCl) is prepared for $\pm 1\%$ in volume. This process has a very deep titration curve, which makes it quite difficult to control if the concentrations are not exactly known.

The parameters of the MFLC Controller algorithm remain the same as before, except that the nontuning parameter, the controller gain, k, is decreased and set of 5×10^{-7} (to compensate for the higher gain of the plant). The responses for the NaOH–HCl system are presented in Fig. 16a. The corresponding control signal is shown in Fig. 16b. It can be seen in Fig. 16a that oscillations in the responses are observed around the reference, but the controller manipulates the control signal to maintain the process within the control band. Since the process is highly nonlinear, the responses of the plant, due to small-manipulated variable changes, exhibit small oscillations around the setpoint. The controller tracks the reference correctly.

5. Conclusions

Since the MFLC algorithm is based on learning directly from the closed-loop behavior of the pH plant, this approach gives a general solution for acid–base systems, simple to implement in existing control hardware and easy to design without a detailed understanding of plant dynamics.

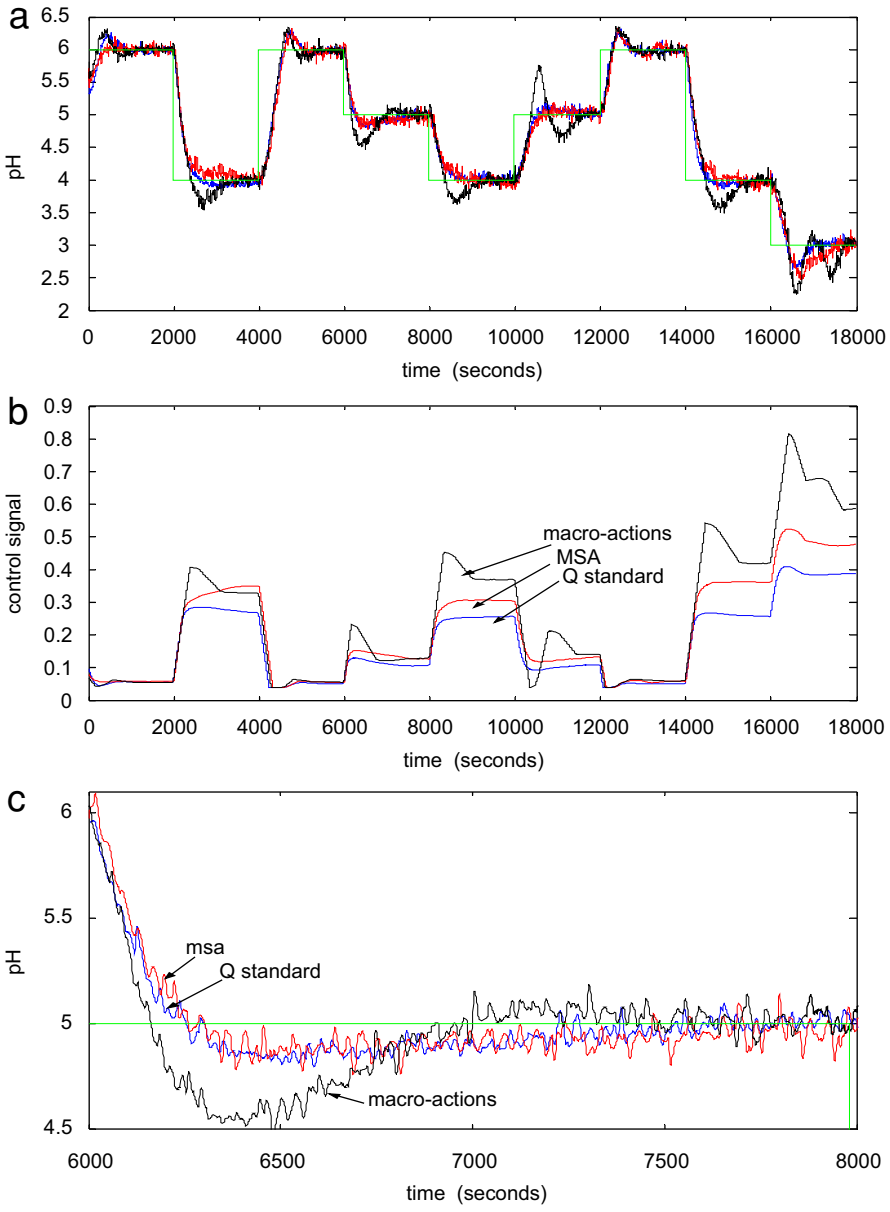

Fig. 14. Online application MFLC for $NaCH_3COO$–HCl systems, macro-actions: (a) online responses, (b) control signal and (c) systems for 6000–8000 s.

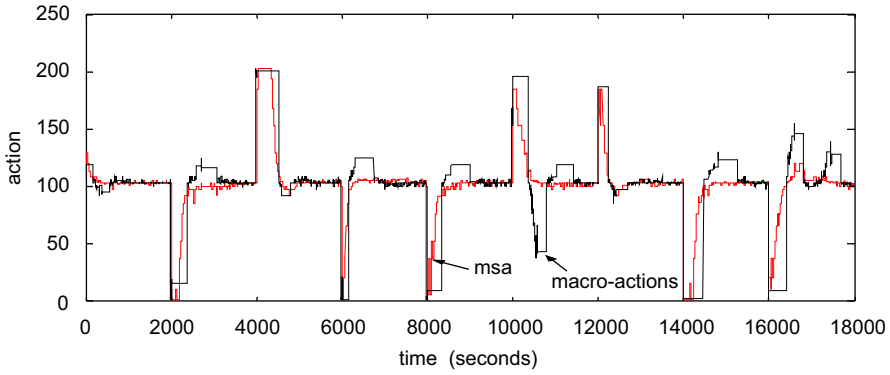

Fig. 15. Actions execute for MSA and macro actions.

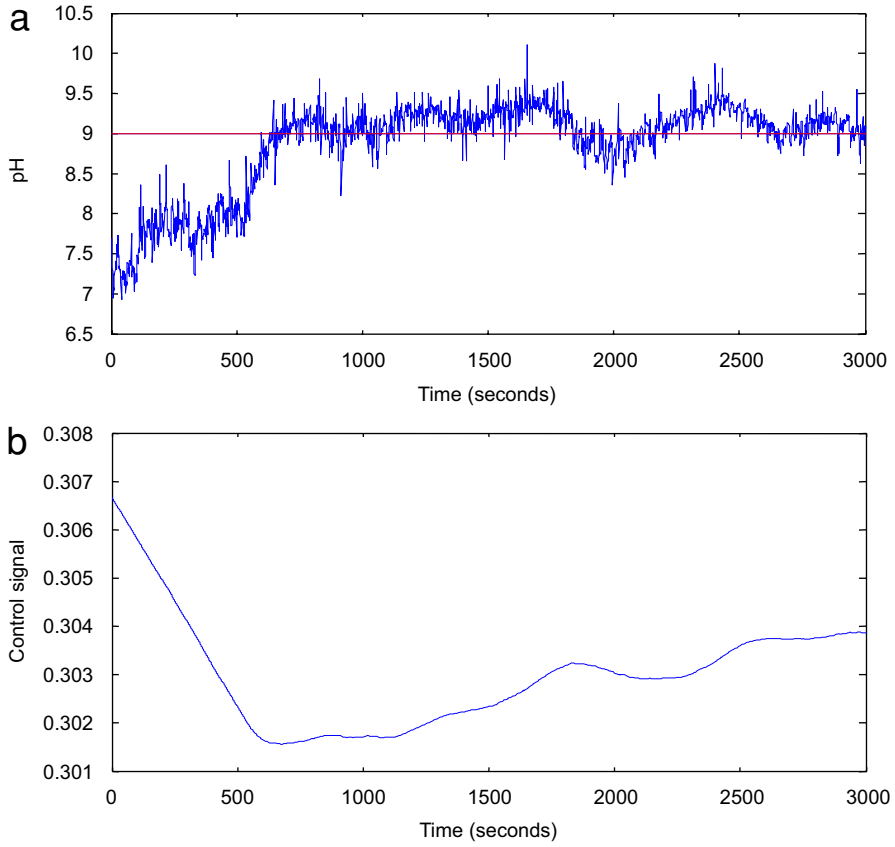

Fig. 16. Online application of MFLC for NaOH–HCl systems: (a) responses and (b) control signal.

The MFLC controller algorithm, based on the one-step ahead Q-learning look-up table, has been presented. The MFLC has been developed for general pH neutralization processes. The optimal control actions are selected using the ε-greedy policy. Tuning parameters are provided in the algorithm that have a clear meaning (representing the variation in the plant, desired degree of exploration, etc.).

The behavior of the MFLC algorithm applied to a laboratory pH plant gives a good performance in a wide range of pH values, and for different processes. The controller shows that in some cases the controller acts aggressively when the pH is far from the control band but get smoother as it approaches. In general, the controller performs reasonably well.

Acknowledgements

This work was funded by MCYT-CICYT (DPI2004-07444-C04-02). The first author would also like to thank the MAE-AECI and UVa for financial support.

References

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

Biasizzo, K.K., Skrjanc, I., Matko, D., 1997. Fuzzy predictive control of highly nonlinearity pH process. Computers and Chemical Engineering 21, s613–s618.

Chen, J., Huang, T.C., 2004. Applying neural networks to on-line updated PID controller for nonlinear process control. Journal of Process Control 14, 211–230.

Dietterich, T.G., 1997. Hierarchical reinforcement learning with the MAXQ value function decomposition. Technical Report, Department of Computer Science, Oregon State University.

Edgar, C.R., Postlethwaite, B.E., 2000. MIMO fuzzy internal model control. Automatica 34, 867–877.

Gustafsson, T.K., Waller, K.V., 1983. Dynamic modeling and reaction invariant control of pH. Chemical Engineering Science 38 (3), 389–398.

Gustafsson, T.K., Skrifvars, B.O., Sandström, K.V., Waller, K.V., 1995. Modeling of pH for control. Industrial Engineering Chemical Research 34, 820–827.

Henson, M.A., Seborg, D.E., 1994. Adaptive nonlinear control of a pH neutralization process. IEEE Transactions on Control Systems Technology 2 (3), 169–182.

Kalafatis, A.D., Wang, L., Cluett, W.R., 2005. Linearizing feedforward–feedback control of pH process based on Wiener model. Journal of Process Control 15, 103–112.

Krishnapura, V.G., Jutan, A., 2000. A neural adaptive controller. Chemical Engineering Science 55, 3803–3812.

Kwok, D.P., Deng, Z.D, Li, C.M.K., Leung, T.P., Sun, Z.Q., Wong, J.C.K., 2003. Fuzzy neural control of systems with unknown dynamics using Q-learning strategies. In: Proceeding of The 12th IEEE International Conference on Fuzzy Systems, vol. 1, St. Louis, MO, USA, 25–28 May, pp. 482–487.

Loh, A.P., Looi, K.O., Fong, K.F., 1995. Neural network modeling and control strategies for a pH process. Journal of Process Control 6, 355–362.

McGovern, A., Sutton, R.S., 1998. Macro-actions in reinforcement learning: an empirical analysis. Amberst Technical Report No. 98-70.

Mwembeshi, M.M., Kent, C.A., Salhi, S., 2004. A genetic algorithm based approach to intelligent modelling and control of pH in reactor. Computer and Chemical Engineering 28 (9), 1743–1757.

Norquay, S.J., Palazoglu, A., Romagnoli, J.A., 1998. Model predictive control based on Wiener models. Chemical Engineering Science 53 (1), 75–84.

Parr, R., 1998. Hierarchical Control and learning for Markov decision processes. Ph.D. Thesis, University of California at Berkeley.

Precup, D., 2000. Temporal abstraction in reinforcement learning. Ph.D. Dissertation, Department of Computer Science, University of Massachusetts, Amherst.

Ramirez, N., Jackson, H., 1999. Application of neural networks to chemical process control. Computers and Chemical Engineering 37, 387–390.

Riedmiller, M., 1998. High quality thermostat control by reinforcement learning—a case study. In: Proceedings of the Conald Workshop 1998. Carnegie Mellon University.

Sabharwal, J., Chen, J., 1996. Intelligent pH control using fuzzy invariant clustering. In: Proceedings of the 28th Southeastern Symposium on System Theory, Baton Rouge, LA, USA, pp. 514–518.

Schmidt, R.J., 2005. Industrial catalytic process—phenol production. Applied Catalysis A: General 280, 89–103.

Schoknecht, R., Riedmiller, M., 2003. Learning to control at multiple time scales. In: Proceeding of ICANN 2003, Istanbul, Turkey, 26–29 June, pp. 479–487.

Shinskey, F.G., 1973. pH and pIon Control in Process and Waste Stream. Wiley, New York.

Sung, S., Lee, I., 1995. pH control using a simple set point change. Industrial and Engineering Chemistry Research 34 (50), 1730–1734.

Sutton, R.S., Barto, A.G., 1998. Reinforcement Learning: An Introduction. The MIT Press, Cambridge, MA.

Sutton, R.S., Precup, D., Singh, S., 1999. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. Artificial Intelligence 112 (1–2), 181–211.

Wright, R.A., Kravaris, C., 1991. Nonlinear control of pH processes using the strong acid equivalent. Industrial and Engineering Chemistry Research 30 (7), 1561–1572.