Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 49-1 (2016) 089–094

MODEL-FREE PREDICTIVE CONTROL OF NONLINEAR PROCESSES BASED ON REINFORCEMENT LEARNING

Dr. Hitesh Shah*, Dr. M. Gopal**
***Professor, Department of Electronics & Communication Engineering,**
G H Patel College of Engineering & Technology
Vallabh Vidyanagar (Gujarat) 388120, INDIA
E-mail: iitd.hitesh@ gmail.com
**Director, *School of Engineering*
Shiv Nadar University, Noida (Uttar Pradesh), INDIA
E-mail: mgopal@snu.edu.in

Abstract: Model predictive control (MPC) is a model-based control philosophy in which the current control action is obtained by on-line optimization of objective function. MPC is, by now, considered to be a mature technology owing to the plethora of research and industrial process control applications. The model under consideration is either linear or piece-wise linear. However, turning to the nonlinear processes, the difficulties are in obtaining a good nonlinear model, and the excessive computational burden associated with the control optimization. Proposed framework, named as model-free predictive control (MFPC), takes care of both the issues of conventional MPC. Model-free reinforcement learning formulates predictive control problem with a control horizon of only length one, but takes a decision based on infinite horizon information. In order to facilitate generalization in continuous state and action spaces, fuzzy inference system is used as a function approximator in conjunction with *Q*-learning. Empirical study on a continuous stirred tank reactor shows that the MFPC reinforcement learning framework is efficient, and strongly robust.

Keywords: Model predictive control, Reinforcement learning, *Q*-learning.

1. INTRODUCTION

Model predictive control (MPC) is the most popular advanced control technique in the process industry [1]. The essence of MPC is to optimize, over the manipulable inputs, forecasts of process behavior. The forecasting is accomplished with process model, and therefore, the model is the essential element of the MPC controller.

Whereas the dynamic behavior of most chemical processes is nonlinear, linear models have predominantly been used for process control in practice because of the difficulty associated with building an accurate nonlinear model, either by first principles or by system identification techniques [1]. MPC approach determines a sequence of actions based on predictions using the system model that guarantee stability and certain optimal properties of the system in terms of the desired behavior. A model is, however, always an approximation of the system under consideration. Predictions about the behavior of the system become more and more inaccurate when considered further into the future. To deal with this, MPC techniques use a *rolling horizon* to increase robustness. The rolling horizon principle consists of synchronizing the state of the model with the state of the true system at every decision step. At every decision step, the MPC agent observes the state of the true system and synchronizes the estimate that it has of the state of the system with this, and tries to find the best sequence of actions, given the updated state. Typically, the agent only executes the first action of this sequence. It then observes the system state again and finds a new sequence of actions. However, the rolling horizon increases computational costs if the horizon over which the agent has to determine actions is infinite, since at each decision step the MPC agent has to find a sequence of actions. A finite horizon is therefore assumed. However, because of the limited horizon over which actions are considered, the resulting policy may be suboptimal. The smaller the control horizon used to reduce on-line computation, the more suboptimal the resulting solution may become. The MPC algorithm therefore might suffer from the dilemma of very high computational requirements *vs* suboptimality.

However, both the difficulties — obtaining a good model of the nonlinear process and the excessive computational burden associated with the control optimization, have been serious obstacles to wide spread use of MPC in industrial implementations. To deal with the issue of modeling difficulties, and the dilemma of very high computational cost versus suboptimality, a control approach based on learning is more adequate. Different model-free learning control approaches, useful for process control problems, have been proposed in the literature. Anders Stenman [2] presented the concept of model-free predictive control, which combines the idea of model-on-demand with MPC techniques. They estimate the process dynamics locally and on-line, using process data stored in the database. However, a drawback of their approach is that the performance of the controller depends on the quality of the data base, and the controller normally requires large computational resources due to the nature of the underlying estimation procedure. Lee and Lee [3] & Lee and Wong [4] have suggested an MPC approach based on approximate dynamic programming (ADP). This approach has its roots in the artificial intelligence (AI) literature and closely follows the ideas of reinforcement learning (RL) [5] and neuro-dynamic programming [6]. This

2405-8963 © 2016, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.03.034

approach attempts to solve the stochastic optimal control problem through dynamic programming (DP) but only approximately and within limited regions of state space. The usual barrier of *curse-of-dimensionality* is alleviated by employing closed-loop simulations and function approximation.

An ADP-based data driven control algorithm presented in [4], iteratively learns cost-to-go function and maps the state to the cost-to-go value. If we map all relevant state and action pairs to cost-to-go values, no model would be necessary at all. Such a model-less scheme is called *Q*-learning in the field of AI [5]. A model-free learning control (MFLC) approach, proposed by S. Syafiie, et. al. [7, 8], is based on reinforcement learning algorithms. The MFLC controller algorithm, based on the one-step ahead *Q*-learning look-up table, performs reasonably well. However, for continuous state space systems, such as in most process control problems, the conventional *Q*-learning framework designed for finite Markov Decision Problems (MDP) is not appropriate since it is unlikely that exactly same states would be visited multiple times. An alternative methodology is based on replacing the table with a function approximation [9]. This methodology has been quite successful in many practical cases, and does not need to use explicitly the table of *Q*-values.

We propose a novel intelligent control method that is based on reinforcement learning, where in the design and on-line learning is not based on a model; rather can be implemented only by evaluative feedback during interaction with the plant. Explicit and exact modeling of system dynamics is not required; and the machine learning algorithm realizes adaptivity to uncertainties, without requiring any prior knowledge. RL framework is, in fact, a means to deal with issues arising in MPC—system-model requirement, computational complexity, and suboptimality of actions due to limited horizon over which actions are considered. A model-free predictive control (MFPC) based on RL takes decisions on infinite-horizon information, with computational cost of a one-step MPC [10].

To obtain a MFPC controller, we run RL experiment on the plant under nominal conditions (no disturbances) with no prior knowledge on the ranges of the PID parameters (trial values are used). The controller is model-free, and it learns from interactions with the environment using the values of the state variables and the cost signal at each time step. Subsequently, controller finds the parameters corresponding to steady-state behavior under nominal conditions. Simulation studies demonstrate the setpoint tracking and disturbance rejection capability of the MFPC controller based on RL. Further, to add adaptivity to uncertainties, we redesign the RL control scheme with the prior knowledge of the expected ranges of parameters for good control performance. The redesigned RL provides on-line tuning of parameters.

The proposed MFPC based on RL scheme provides a flexible approach to the design of intelligent agents (here, PID Controllers) in situations for which both conventional and supervised learning methods are impractical or difficult to be employed. Unlike conventional methods, the RL scheme is not based on approximate model of the plant; it gives parameters by on-line interaction with the plant, thereby producing results which are expected to be close enough to the design if exact model of the plant were available. Unlike supervised learning methods, RL can be applied to problems where significant domain knowledge is either unavailable or costly to obtain.

We base our analysis and simulations on fuzzy inference system (FIS) as a function approximation in RL. The fuzzy *Q*-learning [11] requires some prior knowledge. In case the required knowledge is not available or available knowledge is not reliable, one can acquire the knowledge using dynamic fuzzy-*Q* [12] learning algorithm. The proposed dynamic fuzzy-*Q* adaptive MFPC controller is very effective for complex nonlinear systems, and it doesn't need any prior knowledge to find optimal PID parameters.

The paper is structured as follows. Section 2 describes basics behind model-free predictive control. Section 3 presents the proposed reinforcement learning controller framework, its architecture and the design steps of adaptive MFPC controller. Section 4 discusses basics of model predictive control. Section 5 gives details of a highly nonlinear process—the continuous stirred tank reactor (CSTR): its dynamics and controller learning. Section 6 compares and discusses the empirical performance of set point tracking and disturbance rejection for MPC and MFPC controllers on the basis of simulation results. Finally in Section 7, the conclusions are presented.

2. MODEL FREE- PREDICTIVE CONTROL

The basic idea behind the model-free predictive control (MFPC) philosophy is to learn action values directly, by trial and error, without building an explicit model of the environment, and thus it retains no explicit estimate of the probabilities that govern state transitions. Reinforcement learning (RL) provides a way to build learning agents (intelligent controllers) that optimize their behavior in unknown environments [13, 14]. In RL, experience is built up over time through interaction with the system the agent has to control, rather than assumed available a priori (system model). The experience is based on the performance indicators that give information about how well a certain action was in a certain state of the system. The experience is also based on the state transitions of the system under actions taken. The performance function (value function) is approximated by keeping track of the performance obtained at each decision step considering the system state, performed action, and the resulting system state. At each decision step, the value function of the previous decision step is updated with experience built up over that previous decision step. By accumulating sufficient experience, the agent may accurately estimate the true value function—an infinite horizon estimation.

Thus, once the value functions are known well, an RL problem reduces to an MPC problem with a control horizon of only length one. At the same time, decisions are based on infinite horizon information. This takes care of both the issues associated with conventional MPC.

2.1 Learning framework

RL is a promising technique for adaptive control problems, where learning and control are performed simultaneously. Value iteration algorithms directly search for an optimal value function, which they use to compute an optimal policy. An important and most widely used RL algorithm from the value iteration class is *Q*-iteration (widely known as *Q*-learning), which updates the *Q*-function (state-action value function) online, using observed state transitions and rewards. Conventional *Q*-learning guarantees exactness to store distinct *Q*-values of every state or state-action pair. Unfortunately, this is impossible for continuous and very large or infinite state-action space. Instead, function approximations must be used to represent value functions.

3. REINFORCEMENT LEARNING CONTROLLER

We consider an adaptive learning system (controller) that interacts with its environment (discrete-time dynamical system). For each state $s^t = \{s_1^t, s_2^t, \cdots, s_n^t\} \in \mathrm{S}$ (a finite state space) of the dynamical system at discrete-time *t*, there is a finite set of possible decisions (control actions), $u(s^t) \in \mathrm{U}$ that may be taken by the learning system (controller). The index t represents a stage variable to describe how many decisions have thus far been made, where $t = 0, 1, 2, \cdots$

3.1 Controller framework

FIS have been widely adopted as function approximators for reinforcement learning problems, in particular in conjunction with *Q*-learning, known as fuzzy *Q*-Learning (FQL). FQL is a RL method to tune fuzzy inference system conclusions. FQL is an adaptation of Watkin's *Q*-learning [15] for FIS, where both the actions and *Q*-functions are inferred from fuzzy rules. In FQL, the fuzzy rules are based on the initial knowledge of the designer, and the conclusion part of the inference system is adjusted online. However, structure identification, such as partitioning the input and output spaces and determination of the number of fuzzy rules, are still carried out offline and it is time consuming. To cope with this difficulty, many researchers have been working to find automatic methods for fuzzy system design [11, 16, 17]. Dynamic fuzzy *Q*-Learning (DFQL) proposed in [12] is an automatic method, capable of self-tuning FIS based on reinforcement signals. The DFQL is an efficient learning method whereby not only the conclusion part of a FIS can be constructed online, but also its structure can be generated automatically. This self-organizing feature makes system performance better than that of a conventional fuzzy *Q*-learning. DFQL fulfills our objective of model-free and on-line fuzzy-*Q* learning control of nonlinear systems. We have reformulated this approach to suit our tuning-based PID-control structure.

In FIS structure, precondition part (FIS(A)) depends on the number of variables in the state vector, and the number of fuzzy sets generated dynamically to sense each variable. There can be various types of membership functions, e.g., triangular, trapezoidal, Gaussian, etc. The *N* number of rules in the FIS rule base are constructed online using aligned clustering approach [18]; $N = \prod_{s=1}^{n} N_L(s)$, where $N_L(s)$ = number of fuzzy labels for variable s. The consequent part (FIS(B)) of fuzzy inference system adjusts/tunes only the conclusion part of the fuzzy rules in an incremental manner, as per the concept of FQL specified in [11]. We describe a fuzzy *Q*-learning method based on the TS fuzzy model [19].

3.2 Controller architecture

The architecture of an adaptive controller based on dynamic fuzzy *Q*-learning (DFQL) is shown in Fig. 1.

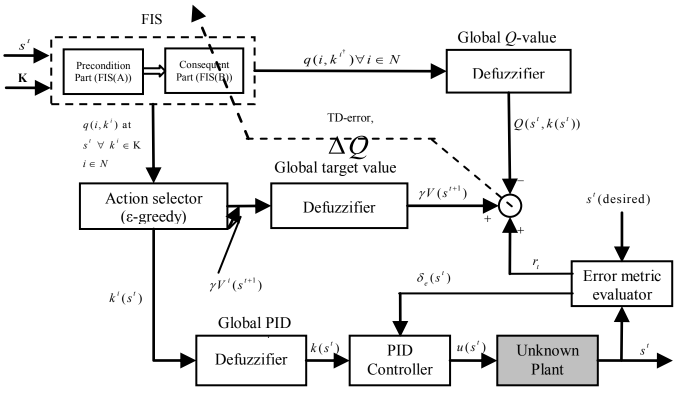

Fig. 1 Dynamic fuzzy-*Q* controller architecture

Each fuzzy rule $R_i; \, i = 1, 2, \cdots, N$, is a local representation over a region defined in the input space, S, constructed automatically using online aligned clustering approach [18]. It memorizes the quality vector *q* associated with each set of PID parameters, $k^c = [k_p \quad k_v \quad k_i]; \; c = 1, 2, \ldots, m$. $k^c \in \mathrm{K} = \{k^1, k^2, \cdots, k^m\}$ are m subsets of possible parameter values which PID controller can assume, constructed on the basis of prior experience and knowledge. The parameter q associates to each action in R_i, a quality with respect to the task. The fuzzy reasoning applied by FIS is an extension of modus ponens with competing actions [19], and in DFQL one builds an FIS with competing actions for each rule R_i as follows:

$$\begin{aligned} R_i : \text{If } s_1^t \text{ is } L_1^i \text{ and } \cdots \text{ and } s_n^t \text{ is } L_n^i \text{ then } & k^i = k^1 \text{ with } q(i,1) \\ \text{or } & k^i = k^2 \text{ with } q(i,2) \\ & \vdots \\ \text{or } & k^i = k^m \text{ with } q(i,m) \end{aligned} \tag{1}$$

where L_j^i is the linguistic term (fuzzy label) of input variable $s_j^t; \, j = 1, 2, ..., n$, in rule $R_i; \, i = 1, 2, ...N$; and its membership function denoted by $\mu_{L_j^i}$. The membership function is chosen as a Gaussian function. Each input variable $s_j^t \; (j = 1, 2, \ldots, n)$ has l membership functions given by

$$\mu_{L_{jp}^i}(s_j^t) = \exp\left[-\frac{\left(s_j^t - c_{jp}\right)^2}{\sigma_{jp}^2}\right] \quad ; j = 1, 2, \cdots, n; \;\; p = 1, 2, \cdots, l. \tag{2}$$

where $\mu_{L_{jp}^i}$ is the pth membership function of s_j^t, while c_{jp} and σ_{jp} are, respectively, the center and width of this membership function. The output of the ith rule $R_i \, (i = 1, 2, \cdots, N)$ is given by

$$\phi_i\left(s_1^t, s_2^t, \cdots, s_n^t\right) = \prod_{j=1}^{n} \mu_{L_{jp}^i}(s_j^t) = \exp\left[-\sum_{j=1}^{n} \frac{\left(s_j^t - c_{jp}\right)^2}{\sigma_{jp}^2}\right] \; ; \; p = 1, 2, \cdots, l. \tag{3}$$

Normalized output of the ith rule is

$$\alpha_i(s^t) = \frac{\phi_i(s^t)}{\sum_{r=1}^{N} \phi_r(s^t)} \quad ; s^t \square \left\{s_1^t, s_2^t, \cdots, s_n^t\right\} \in \mathrm{S} \;; i = 1, 2, \cdots, N. \tag{4}$$

The quality vector q is used to select PID parameters to maximize discounted sum of reinforcements received by the controller. Learning system (controller) can choose single set of PID parameters k^i from the controller PID parameters set K in each rule R_i (calculated using Eq. (7)). The global continuous PID parameter set $k(s^t)$ for an input vector s^t, inferred by the center-of-gravity (COG) method for defuzzification, is:

$$k(s^t) = \sum_{i=1}^{N} \alpha_i(s^t) k^i(s^t) \tag{5}$$

where $k^i \in \mathrm{K}$ is the PID parameter set selected in rule R_i.

The continuous control action $u(s^t)$ for an input vector s^t is the output of PID controller, and this action is given as:

$$u(s^t) = k_p e(s^t) + k_v \frac{de(s^t)}{dt} + k_i \int_0^t e(s^t) d\tau = k(s^t)\delta_e(s^t) \tag{6}$$

where, $\delta_e(s^t)$ is the system control error vector obtained from the error metric evaluator. In order to explore the set of possible PID parameters and acquire experience through the RL signals, PID parameters are selected using an exploration/exploitation policy (EEP) [5]. We use a pseudo stochastic exploration (ε-greedy) as in [20]. In ε-greedy exploration, we gradually reduce the exploration (determined by the ε parameter) according to some schedule; we have reduced ε to its 90 percent value after every 10 iterations. The lower limit of parameter ε has been kept fixed at 0.002 (to maintain exploration).

We choose a random parameter action set, $k^{i^\dagger}$ in rule R_i with ε-greedy exploration, and a quality maximizing action, k^{i^*} derived as per $q(i, k^{i^*}) = \max_{b \le m} q(i, b)$, otherwise.

$$k^i(s^t) = \begin{cases} k^{i^\dagger}(s^t), & \text{with } \varepsilon\text{-greedy} \\ k^{i^*}(s^t), & \text{otherwise} \end{cases} \tag{7}$$

Then the Q-value for the inferred action $k(s^t)$ is

$$Q(s^t, k(s^t)) = \sum_{i=1}^{N} \alpha_i(s^t) q(i, k^{i^\dagger}) \tag{8}$$

and value of state s^t is:

$$V(s^t) = \sum_{i=1}^{N} \alpha_i(s^t) q(i, k^{i^*}) \tag{9}$$

Under the action $u(s^t)$, the system undergoes transition $s^t \xrightarrow{r_t} s^{t+1}$, where r_t is the reinforcement received by the controller. This information is used to calculate *temporal difference* (TD) [5] approximation error as: $\Delta Q = r_t + \gamma V(s^{t+1}) - Q(s^t, k(s^t))$.

The q parameter values and Q-values are updated as

$$q(i, k^{i^\dagger}) \leftarrow q(i, k^{i^\dagger}) + \eta \; \Delta Q \alpha_i(s^t) \tag{10}$$

$$Q(s^t, k(s^t)) \leftarrow Q(s^t, k(s^t)) + \eta\left[r_t + \gamma V(s^{t+1}) - Q(s^t, k(s^t))\right] \tag{11}$$

where $0 \le \gamma \le 1$ is the discount factor that controls how much effect future costs have on current optimal decisions, η is the learning-rate parameter.

4. CONTROLLER REALIZATION

In order to test the validity of the proposed model-free predictive controller (MFPC) based on reinforcement learning, we use continuous stirred tank reactor (CSTR), a highly nonlinear process control problem, as the standard benchmark. The system dynamics and controller learning details are as follows:

4.1 CSTR dynamics and control

The first principles model of the CSTR, the operating point data and process parameters are as specified in [22], have been used in this simulation study.

It has been shown that an optimized PID controller is able to control the CSTR for certain operating region [22]. It is desired to design and simulate an optimized PID controller to manipulate the coolant flow rate in CSTR. The equivalent discrete-time form of the PID controller is given as:

$$u(t) = k_p e(t) + k_i T_s \sum_{i=1}^{t} \frac{e(i-1) + e(i)}{2} + k_d \frac{[e(t) - e(t-1)]}{T_s} \tag{12}$$

where T_s is the sampling time.

4.2 Controller learning details

MFPC controller: In CSTR process control problem, the objective is to control the measured concentration of the product, C, by manipulating the coolant flow rate, q_c. The CSTR system has two state variables, namely, the reactor product concentration and the reactor temperature. We define error as the difference between desired and actual values of product concentration, *i.e.*, $e(t) = C_d - C$, where C_d is the desired value of the product concentration. The reinforcement signal is given as

$$r_t = \begin{cases} -1, & otherwise \\ 10, & |e(t)| < 0.001 \text{ mol}/l \\ 0, & 0.001 < |e(t)| < 0.005 \text{ mol}/l \end{cases} \tag{13}$$

In MFPC controller realization, the two state variables—the product concentration and the reactor temperature, $s^t = [C; T]$, were normalized and fed to the controller input. Initially, for each state variable, a single fuzzy set with Gaussian membership function with zero mean and 0.2 standard deviation is used, which results in a single rule. Then fuzzy rules are dynamically generated as per the method discussed above.

The parameters for DFQ-learning are: the discount factor γ is set to 0.8; learning-rate parameter η is set to 0.2, TD error factor (K_D) 50, TD error criterion (k_e) 1, similarity of membership function (k_{mf}) 0.25, similarity criterion (ρ) 0.7, ε-completeness threshold (k_d) $\sqrt{\log 2}$. Exploration level ε decays from $0.5 \to 0.002$ over the iterations. We consider a set of PID parameters: $k_p = 180:10:200$; $k_i = 330:10:350$; $k_d = 150:5:160$. We deliberately introduce deterministic noise of ±1% of PID parameters with a probability of (1/3), for stochastic simulation.

MPC controller: For CSTR process control problem, the MPC controller is implemented using MATLAB® MPC toolbox [21]. The linear state-space model was obtained at each sampling instant by linearizing the nonlinear CSTR model at the present operating condition. The MPC controller controls the deviations in CSTR output concentration from set-point with a weight of 3.5 for the corresponding objective function term; CSTR output temperature is left un-controlled. The MPC controller manipulates the coolant flow rate by minimizing the change in manipulated variable with a weight of 0.1 for the corresponding objective function term subject to hard constraints with $u_{\min} = 90$; $u_{\max} = 110$; $\Delta u_{\min} = -10$; $\Delta u_{\max} = 10$. State estimation is done with the default state estimator of MATLAB's MPC toolbox. Other experimental settings are as follows: prediction horizon, $N_p = 10$; control horizon $N_c = 2$; sampling time = 0.1 mins.

5. SIMULATION EXPRIMENTS

To demonstrate the learning performance via setpoint changes and unmeasured disturbances, and robustness against uncertainties of adaptive MFPC controller based on RL, simulation experiments have been performed on CSTR process control problem. MATLAB R2014a has been used as simulation tool.

5.1 Learning performance study

The CSTR system model has been simulated for single episode of 30 min using forth-order Runge-Kutta solver, with fixed-step size of 10 msec. The dynamic behavior of the CSTR process is not the same at different operating points, and the process is, indeed, nonlinear. In order to assess the setpoint tracking capability of MFPC, we start the simulation at nominal operating point: $C = 0.0836\ mol/l$, $T = 440.2$ K , $q_c = 103.41\ l/\min$,and setpoint variation in product concentration as given in Fig. 2

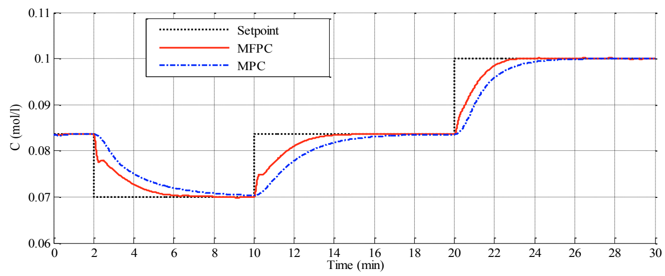

Fig. 2(a) CSTR process control problem: process output

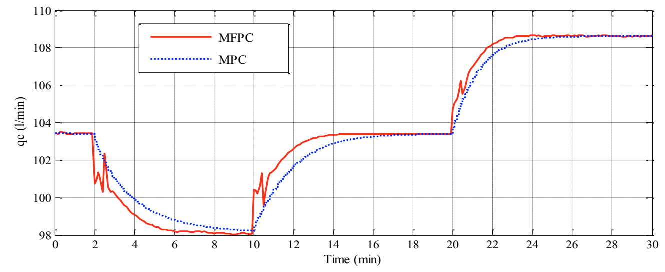

Fig. 2(b) CSTR process control problem: controller input

Both the proposed controller MFPC and MPC are able to maintain the setpoint at desired value. However, the performance of MFPC controller at all the setpoint changes is found to be better than MPC controller, as there is no overshoot and it settles to setpoint faster. From the simulation experiment of a single episode setpoint tracking, we obtain PID parameters as $k = [k_p \quad k_i \quad k_d] = [190.2 \quad 339.8 \quad 152.35]$.

Simulation studies have been carried out to demonstrate the disturbance rejection capability of the proposed MFPC controller, by introducing unmeasured disturbance of +20% in feed flow rate, at the 20th sampling instant under nominal operating point (Fig. 3)

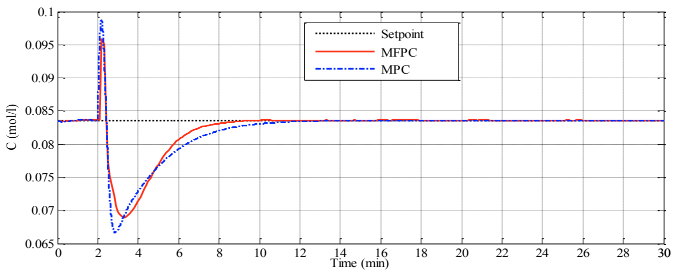

Fig. 3 CSTR process: disturbance rejection performance

From the disturbance rejection simulation study, it can be inferred that the controller MFPC is able to reject the disturbance quickly and bring the reactor concentration back to the nominal value of the setpoint

5.2 Robustness study

In the following, we compare the performance of MFPC and MPC controllers under uncertainties. For this study, we trained the controller for 20 episodes at steady-state operating data under nominal operating point.

5.2.1 Disturbance in product flow rate (q_{if})

We consider a variation in input product flow rate from 30 *l*/min to 180 *l*/min (nominal value 100 *l*/min), and make it to occur at time instant 15 min. Fig. 4 shows the mean square error (MSE) *vs* change in product flow rate, for MFPC and MPC controllers. Table 1 tabulates values of MSE for ± 40% variation with respect to nominal value of product flow rate.

Table 1 MSE comparison of controllers (mol/*l*)

(MSE: values shown $\times 10^{-5}$)

Controller	q_{if} 40 (*l*/min)	q_{if} 100 (*l*/min)	q_{if} 160 (*l*/min)
MPC	4.135	2.8220	3.712
MFPC	2.809	0.9718	3.121

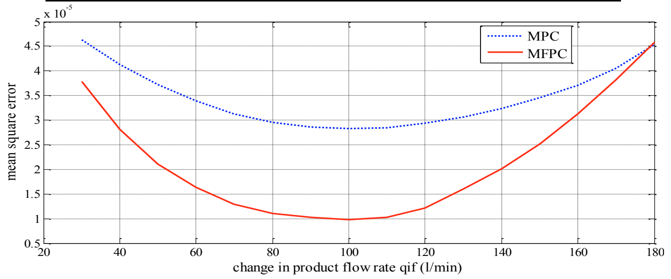

Fig. 4 CSTR process: MSE *vs* product flow rate (q_{if})

5.2.2 Disturbance in input product concentration (C_{if}):

We consider a variation in input product concentration from 0.5 mol/*l* to 1.6 mol/*l* (nominal value 1 mol/*l*), and make it to occur at time instant 15 min. Fig. 5 shows the MSE *vs* change in product concentration, for MFPC and MPC controllers. Table 2 tabulates values of MSE for ± 40% variation with respect to nominal value of product concentration.

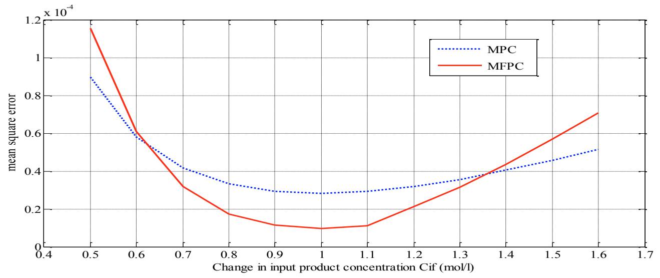

Fig. 5 CSTR process: MSE *vs* product concentration (C_{if})

Table 2 MSE comparison of controllers (mol/*l*)
(MSE: values shown $\times 10^{-4}$)

Controller	C_{if} 0.6 (mol/*l*)	C_{if} 1.0 (mol/*l*)	C_{if} 1.4 (mol/*l*)
MPC	0.5804	0.28220	0.4037
MFPC	0.6096	0.09773	0.4356

5.2.3 Disturbance in input temperature (T_{if})

We consider a variation in input temperature from 300 K to 400 K (nominal value 350 K), and make it to occur at time instant 15 min. Fig. 6 shows MSE *vs* change in input temperature, for MFPC & MPC controllers. Table 3 tabulates MSE for temperature variation with respect to nominal value.

Table 3 MSE comparison of controllers (mol/*l*)
(MSE: values shown $\times 10^{-5}$)

Controller	T_{if} 310 K	T_{if} 350 K	T_{if} 390 K
MPC	3.601	2.8220	3.304
MFPC	1.753	0.9773	1.402

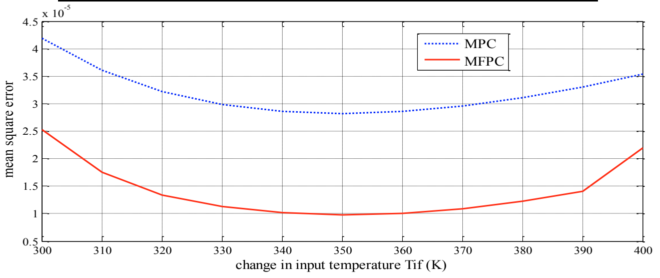

Fig. 6 CSTR process: MSE *vs* input temperature (T_{if})

To summarize, the simulation results indicate that MFPC controller, tuned using dynamic fuzzy-*Q* learning, performs satisfactorily with no prior knowledge on the ranges of PID parameters, and model-free environment. In presence of disturbances around the nominal operating conditions, the adaptive-structure MFPC controller controls the concentration of the product with low value of steady-state error in comparison with MPC controller.

6. CONCLUIONS

This paper has presented the promising concept of model-free predictive control (MFPC), which combines the potential of RL with established and well-known MPC techniques. The MFPC controller algorithm, based on dynamic fuzzy *Q*-learning for tuning of PID controller is presented.

This approach gives a general solution for tuning-based PID controllers to nonlinear time-varying systems, simple to implement in existing control hardware and easy to design without a detailed understanding of plant dynamics. The proposed MFPC based on RL scheme provides a flexible approach to design of intelligent agents (PID Controllers) in situations for which both conventional and supervised learning methods are impractical or difficult to be employed. Unlike conventional methods, the RL scheme is not based on approximate model of the plant; it gives parameters by on-line interaction with the plant, thereby producing results which are expected to be close enough to the design if exact model of the plant were available. Unlike supervised learning methods, RL can be applied to problems where significant domain knowledge is either unavailable or costly to obtain.

Simulation studies on CSTR show that MFPC controller is efficient in learning and performs well under uncertainties.

REFERENCES

[1] M. Morari, J. H. Lee, "Model predictive control: past, present and future," *Computers and Chemical Engineering*, Vol. 23, pp.667–682, 1999.

[2] A. Stenman, "Model-free predictive control," *In: Proceedings of the 38th IEEE Conference on Decision & Control*, Phoenix, AZ, USA, pp.3712–3717, 1999.

[3] J. M. Lee, J. H. Lee, "Simulation-based learning of cost-to-go for control of nonlinear processes," *Korean J Chem. Eng.*, Vol. 21(2), pp.338–344, 2004.

[4] J. H. Lee, W. Wong, "Approximate dynamic programming approach for process control," *Journal of Process Control*, Vol. 20, pp.1038–1048, 2010.

[5] R. S Sutton, A. G. Barto, Reinforcement learning: An introduction, Cambridge, MA: MIT Press, 1998.

[6] D. P. Bertsekas, J. N. Tsitsiklis, Neurodynamic Programming, Belmont: Athena Scientific, 1996.

[7] S. Syafiie, F. Tadeo, E. Martinez, "Learning to control pH processes at multiple time scales: performance assessment in a laboratory plant," *Chem. Prod. and Process Modeling*, Vol. 2(1), pp.1-20, 2007.

[8] S. Syafiie, F. Tadeo, E. Martinez, "Model-free learning control of neutralization process using reinforcement learning," *Engineering Applications of Artificial Intelligence*, Vol 20(6), pp.767–782, 2007.

[9] H. Shah, M. Gopal, "Reinforcement Learning Control of Robot Manipulators in Uncertain Environments," *IEEE International Conference on Industrial Technology*, pp.1-6, 2009.

[10] H. Shah, M. Gopal, "Reinforcement Learning Framework for Adaptive Control of Nonlinear Chemical Processes," *Asia-Pacific Journal of Chemical Engineering*, Vol. 6, pp.138-146, 2011.

[11] L. Jouffe, "Fuzzy inference system learning by reinforcement methods," *IEEE Transaction Systems Man and Cybernetics* C, Vol. 28(3), pp.338-355, 1998.

[12] M. J Er., C. Deng, "Online Tuning of Fuzzy Inference Systems using Dynamic Fuzzy Q-learning," *IEEE Transaction on Systems Man and Cybernetics* B, Vol. 34(3), pp.1478-1489, 2004.

[13] Sigaud, O. Buffet, Eds., Markov Decision Processes in Artificial Intelligence. Wiley, 2010.

[14] Cs. Szepesv´ari, Algorithms for Reinforcement Learning, Morgan & Claypool Publishers, 2010.

[15] C. H. Watkins, Learning from delayed rewards, Thesis, University of Cambridge, England, 1989.

[16] T. H. Stephan, K. Ben, *Q*-learning for Systems with Continuous State and Action Spaces, 10th Belgian-Dutch Conference on Machine Learning, BENELEARN, 2000.

[17] J. D. R. Millan, D. Posenato, E. Dedieu, "Continuous-action Q-learning," *Machine Learning*, Vol. 49(2-3), pp. 247-265, 2002.

[18] C. F. Juang, "Combination of Online Clustering and Q-value Based GA for Reinforcement Fuzzy System Design," *IEEE Transaction on Fuzzy Systems*, Vol. 13(3), pp. 289-302, 2005.

[19] J. S. R. Jang, C. T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing, Englewood Cliffs, NJ, Prentice Hall, 1997.

[20] H. Shah, M. Gopal, "A fuzzy decision tree-based robust Markov game controller for robot manipulators," *International Journal Automation and Control*, Vol. 4(4), pp.417- 439, 2010.

[21] L. Wang, Model predictive control system design and implementation using MATLAB®, London: Springer Verlag, 2009.

[22] M. Nikravesh, A.E. Farell, T.G. Stanford, "Control of nonisothermal CSTR with time varying parameters via dynamic neural network control," *Chemical Engineering Journal*, Vol. 76, pp.1-16, 2000.