Control of pH Neutralization Process using Simulation Based Dynamic Programming in Simulation and Experiment (ICCAS 2004)

Dong Kyu Kim*, Kwang Soon Lee**, and Dae Ryook Yang*

* Department of Chemical and Biological Engineering, Korea University, Seoul, Korea
(Tel : +82-2-3290-3298; E-mail: dryang@korea.ac.kr)
** Department of Chemical Engineering, Sogang University, Seoul, Korea
(Tel : +82-2-705-8477; E-mail: kslee@sogang.ac.kr)

Abstract: For general nonlinear processes, it is difficult to control with a linear model-based control method and nonlinear controls are considered. Among the numerous approaches suggested, the most rigorous approach is to use dynamic optimization. Many general engineering problems like control, scheduling, planning etc. are expressed by functional optimization problem and most of them can be changed into dynamic programming (DP) problems. However the DP problems are used in just few cases because as the size of the problem grows, the dynamic programming approach is suffered from the burden of calculation which is called as 'curse of dimensionality'. In order to avoid this problem, the Neuro-Dynamic Programming (NDP) approach is proposed by Bertsekas and Tsitsiklis (1996). To get the solution of seriously nonlinear process control, the interest in NDP approach is enlarged and NDP algorithm is applied to diverse areas such as retailing, finance, inventory management, communication networks, etc. and it has been extended to chemical engineering parts.

In the NDP approach, we select the optimal control input policy to minimize the value of cost which is calculated by the sum of current stage cost and future stages cost starting from the next state. The cost value is related with a weight square sum of error and input movement. During the calculation of optimal input policy, if the approximate cost function by using simulation data is utilized with Bellman iteration, the burden of calculation can be relieved and the curse of dimensionality problem of DP can be overcome. It is very important issue how to construct the cost-to-go function which has a good approximate performance. The neural network is one of the eager learning methods and it works as a global approximator to cost-to-go function. In this algorithm, the training of neural network is important and difficult part, and it gives significant effect on the performance of control. To avoid the difficulty in neural network training, the lazy learning method like k-nearest neighbor method can be exploited. The training is unnecessary for this method but requires more computation time and greater data storage.

The pH neutralization process has long been taken as a representative benchmark problem of nonlinear chemical process control due to its nonlinearity and time-varying nature. In this study, the NDP algorithm was applied to pH neutralization process. At first, the pH neutralization process control to use NDP algorithm was performed through simulations with various approximators. The global and local approximators are used for NDP calculation. After that, the verification of NDP in real system was made by pH neutralization experiment. The control results by NDP algorithm was compared with those by the PI controller which is traditionally used, in both simulations and experiments. From the comparison of results, the control by NDP algorithm showed faster and better control performance than PI controller. In addition to that, the control by NDP algorithm showed the good results when it applied to the cases with disturbances and multiple set point changes.

Keywords: pH neutralization process, the NDP (Neuro-Dynamic Programming), Simulation-Approximation-Evolution (SAE) algorithm, *k*-nearest neighbor method (*k*NN), neural network (NN)

1. INTRODUCTION

Generally, the traditional target processes for chemical process were continuous petrochemical processes. Most of them are very hard to control manually because they are big scaled and continuous. So, the automatic control had been introduced early from the first commercial pneumatic PID controller of Foxboro in 1934. Because of conservative spirit of engineers in chemical plants, the control method had not been out of PID controller till Model-based Predictive Control (MPC) were successfully introduced in FCC (Fluidized Catalyst Cracking) process of Canada Shell, 1979 [1]. After that, though MPC has been a key control method in chemical process until the middle of 1990s [2, 3], there was no optimal control method in the case of seriously nonlinear and integer programming problems.

As time goes on, the target processes of chemical process have been extended to Biotechnology (BT) and Nano-technology (NT) and many nonlinear problems are faced. In that cases, it is often inefficient to control the nonlinear processes with linear control methods. In order to achieve more accurate and precise control performance, the most rigorous solution for the control of nonlinear system is to use the optimal control strategy obtained by dynamic optimization considering the nonlinearity of the process.

The optimal control strategy can be obtained using standard Dynamic Programming (DP). The aim of Dynamic Programming is to find the optimal time-varying input policies by minimizing the objective function which is defined according to the specific control purposes and in most cases, the optimal strategy is calculated rather numerically than analytically. If the size of problem is large, the calculation load can be enormous and the solution cannot be obtained within the given sampling time even with quite fast computer. This problem is called as 'Curse of Dimensionality' and this makes the on-line control using DP virtually impossible [4]. However, as the Neuro-Dynamic Programming (NDP) approach is introduced, the application of DP to nonlinear processes becomes possible and the field of application for NDP is growing. This approach is to perform the vast amount of calculation offline, to learn the optimal strategy in a simple form of approximation by using simulations and experiments data and to calculate the optimal strategy using the approximator online, which is trained offline. Cost-to-go or profit-to-go function as a performance objective function can be approximated by a nonlinear function or neural network (NN) and this can reduce the calculation burden so that the

dynamic programming approach can be applied online. But the NN requires appropriate training before use and the training of NN is not trivial for many cases. To avoid the difficulty in NN training, local approximation method could be used such as *k*-nearest neighbor (*k*NN) method.

In this study, Simulation-Approximation-Evolution (SAE) algorithm suggested by Kaisare *et al*. [4] is investigated against a pH neutralization process. The SAE algorithm is based on Neuro-Dynamic Programming (NDP). Through the simulations, the NN and *k*NN method are compared and the results shows they have same performance. An optimal control of pH neutralization process to avoid the Bellman iteration is suggested and this is a help for training of NN and *k*NN. The effects of constraints on input moves are investigated and the new NDP algorithm is suggested in the case of that.

2. NEURO DYNAMIC PROGRAMMNG (NDP)

2.1 Dynamic Programming

A discrete-time dynamic system can be described by an *n*-dimensional state vector $x(k)$ and an *m*-dimensional input vector $u(k)$ at time step *k*. Choice of an *m*-dimensional control vector $u(k)$ determines the transition of the system from $x(k)$ state to $x(k+1)$ through the following relations [5, 6],

$$x(k+1) = F_h\left(x(k), u(k)\right) \tag{1}$$

where F_h denotes the process model equation and *h* represents the sampling time. A general dynamic optimization problem for such system is to find the optimal sequence of control vectors $u(k)$ for $k=0, \ldots, N-1$ to minimize a performance index which is related with cost-to-go function (J).

$$J = \sum_{i=0}^{N-1} \phi(x(i), u(i)) + \phi_N \tag{2}$$

where ϕ is the one-stage-cost and ϕ_N represents the final cost. Among many ways, the most popular one-stage-cost can be chosen as follows, with the weighting factors Q and R.

$$\begin{aligned} \phi(x(k), u(k)) = &\left(x(k+1) - x_{sp}\right)^T Q \left(x(k+1) - x_{sp}\right) \\ &+ \Delta u(k)^T R \Delta u(k) \end{aligned} \tag{3}$$

$$\phi_N = \left(x(N+1) - x_{sp}\right)^T M \left(x(N+1) - x_{sp}\right) \tag{4}$$

where $k=0, \ldots, N-1$, $u(0)=u_0$, and x_{sp} denotes the set point. In addition to that, $\Delta u(k)=u(k)-u(k-1)$.

Then the optimal cost-to-go can be expressed as follow at time-step *k* and superscript * implies the optimal value.

$$J_k^*(x(k)) = \min_u \left[\sum_{i=k}^{N-1} \phi(x(i), u(i)) + \phi_N \right] \tag{5}$$

where If *N* is infinite, then it becomes the infinite horizon cost-to-go function. It can be expressed as a recursive form.

$$\begin{aligned} J_k^*(x(k)) &= \min_u \left[\phi(x(k), u(k)) + \sum_{i=k+1}^{N-1} \phi(x(i), u(i)) + \phi_N \right] \\ &= \min_u \left[\phi(x(k), u(k)) + J_{k+1}^*(x(k+1)) \right] \\ &= \min_u \left[\phi(x(k), u(k)) + J_{k+1}^*(F(x(k), u(k))) \right] \end{aligned} \tag{6}$$

$$J_N^*(x(N)) = \phi_N \tag{7}$$

It can be shown to satisfy the Bellman equation [5].

For simplicity, $J_k^*(x(k))$ will be shortened as $J^*(k)$. The final goal of DP is to find the input strategy $u(k)$, $k=1, \ldots, N-1$ so that the optimal cost-to-go function $J(k)$ satisfies the Bellman equation for all time-step *k*. The solution can usually be obtained numerically and it suffers from the curse of dimensionality when it involves the gridding of large state space dimension. In order to circumvent the problem, one approach suggested by Kaisare *et. al*. described in the next section can be applied.

2.2 Simulation-based dynamic programming

Simulation-Approximation-Evolution (SAE) algorithm [4] is one of the reinforcement learning methods and it involves computation of the converged cost-to-go approximation offline, which is described in Fig. 1. SAE algorithm is based on Neuro-Dynamic Programming (NDP).

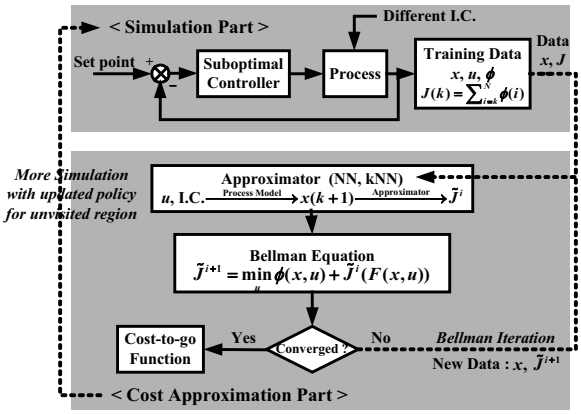


Fig. 1 Architecture for offline computation of cost-to-go approximation.

SAE algorithm is roughly composed of two parts. The first part is "Simulation Part". Simulation is performed with sub-optimal control law to make training data set which is used for the calculation of the infinite horizon cost-to-go function (Eq. (6)) for each state visited during the simulation, and the suboptimal cost-to-go function is calculated by

$$J(k) = \sum_{i=k}^{N} \phi(i)\,. \tag{8}$$

where *N* is sufficiently large for the system to reach new steady state. The second part is "Cost Approximation Part". In this part, the cost-to-go function approximation is performed by fitting a neural network or other function approximator to the data from "Simulation Part". In addition to that, Bellman iteration and policy update procedure is performed to improve the approximation of the cost-to-go function [4].

Since the optimal control law is not available to begin with, a suboptimal control policy is used for the cost-to-go approximation and the resulting control law has to be suboptimal. To improve the approximation, the cost or value iteration can be performed until convergence based on the Bellman equation [4].

$$\tilde{J}^{i+1} = \min_u \phi(x,u) + \tilde{J}^i\left(F_h(x,u)\right) \tag{9}$$

This step may impose an enormous computational burden, but it is performed offline.

2.3 Approximator

In the algorithms related to neuro-dynamic programming, the performance of the approximator for the cost-to-go approximation is crucial. As approximators, the global approximator and the local approximator can be considered. Global

approximators like neural network, polynomial, and etc. are the parametric approximators which require extensive offline training, and the local approximators like *k*-nearest neighbor, kernel-based aproximator, and etc. are nonparametric approximators which require extensive querying instead of offline training.

2.3.1 Neural Network

As a representative global approximator, Neural networks are composed of simple computing elements in parallel. These elements are inspired by biological nervous systems. A neural network (NN) to perform a particular function can be trained by adjusting the weights of the connections between elements [7]. Because the NN is one of global approximator, it is difficult to confirm the safeguard against over estimation and the ability of extrapolation but the rate for evaluation is fast once trained. Furthermore, the convergence for Bellman iteration using NN is not guaranteed. Thus, the training of NN is quite critical to the performance of the neuro-dynamic programming approaches.

2.3.2 *k*-Nearest Neighbor method

As a representative local approximator, The k-nearest neighbor (*k*NN) method is a very intuitive method that classifies unlabeled examples based on their similarity with examples in the training set. For a given unlabeled example $x_u \in \Re^D$ ($\Re^D$ is a workspace), the k "closest" labeled examples in the training data set are found and assigned as x_u to the class that appears most frequently within the *k*-subset. The *k*NN only requires an integer *k*, a set of labeled examples (training data), and a metric to measure closeness [8]. The *k*NN can conveniently handle the quite complex nonlinearity with sufficient data set and training effort is not needed. However, finding the neighboring data set may require extensive data querying procedure. The convergence for Bellman iteration can be guaranteed. But the query time for nearest neighbor is increased in proportion to the number of training data [9].

3. pH NEUTRALIZATION PROCESS

The pH neutralization process has long been taken as a representative benchmark problem of nonlinear chemical process control due to its nonlinearity and time-varying nature. In this study, the pH neutralization process is selected as the control target system with neuro-dynamic programming approach.

3.1 pH neutralization process

The neutralization is a chemical reaction. The control objectives are to drive the system to a different pH conditions (tracking control) or to regulate the effluent pH value despite the disturbance by manipulating the flow rate of titrating stream [10, 11]. The process is illustrated in Fig. 2 and the operating conditions are shown in Table 1. The reactor type of the neutralization process is a continuous stirred tank reactor (CSTR) with baffles, which has a volume of 2.5L. The inlet stream consists of a strong acid stream (q_1 : feed solution), a weak acid stream (q_2 : buffer solution) and a strong base stream (q_3 : titrating solution), which are pumped to the reactor and well-mixed in CSTR. It is assumed that the perfect mixing in tank and the complete dissociation in solution at 25°C are reached [12]. Table 1 shows the typical operating conditions of the process of concern.

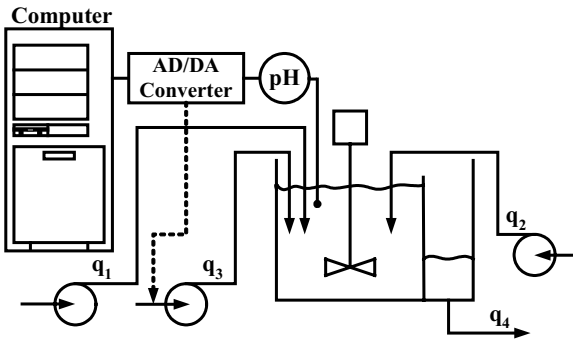


Fig. 2 The pH neutralization process.

Table 1. Operating conditions of pH neutralization process.

Symbols	Values	Symbols	Values
V	2500 [ml]	$[q_1]$	0.003 M HNO_3
q_1	9.0 [ml/s]		5.0×10^{-5} M H_2CO_3
q_2	0.6 [ml/s]	$[q_2]$	0.01 M $NaHCO_3$
q_3	8.5 [ml/s]	$[q_3]$	0.003 M NaOH
			5.0×10^{-5} M $NaHCO_3$

3.2 pH neutralization process model

Generally, the strong acid-base reaction is always assumed to reach equilibrium in water solution almost instantly. This implies the reaction rates approach infinity. So, the reaction rate terms can be ignored in process model which can be simplified. From this approach, Gustafsson and Waller proposed a model using reaction invariants (1983) [12].

As the strong acid and base solutions are completely dissociated into ions, the chemical reactions with a weak acid solution reach equilibrium state. The chemical reactions in the system are as follows [12].

$$\begin{aligned} &H_2CO_3 \leftrightarrow HCO_3^- + H^+, \\ &HCO_3^- \leftrightarrow CO_3^{2-} + H^+, \\ &H_2O \leftrightarrow OH^- + H^+. \end{aligned} \tag{10}$$

The equilibrium constants for the reactions are defined as [12]:

$$K_{a1} = \frac{[HCO_3^-][H^+]}{[H_2CO_3]},\ K_{a2} = \frac{[CO_3^{2-}][H^+]}{[HCO_3^-]},\ K_w = [H^+][OH^-] \tag{11}$$

The total amount of the reaction invariant is not affected by the degree of chemical. According to this fact, the reaction invariants can be derived from the stoichiometry. As Gustafsson and Waller proposed, two kinds of reaction invariant variables are defined in this process. The first reaction invariant (state variable) is the concentration of charge related ions. The other reaction invariant (another state variable) is the total concentrations related to carbonate ions. The relationship between pH and the reaction invariants is given by an nonlinear equation (Gustafsson, 1992) [12].

Reaction invariants for this process are defined as:

$$W_{ai} = [\mathrm{H^+}]_i - [\mathrm{OH^-}]_i - [\mathrm{HCO_3^-}]_i - 2[\mathrm{CO_3^{2-}}]_i \ , \quad (12)$$
$$W_{bi} = [\mathrm{H_2CO_3}]_i + [\mathrm{HCO_3^-}]_i + [\mathrm{CO_3^{2-}}]_i \ .$$

where W_a denotes the charge related reaction invariant, W_b denotes the carbonate ion related reaction invariant, and $i = 1,2,3,4$ for each stream in Fig. 2.

The output equation is derived from Eqs. (11)~(12) which represent the relation between a hydrogen ion concentration and reaction invariants [12].

$$W_b \frac{K_{a1}/[\mathrm{H^+}] + 2K_{a1}K_{a2}/[\mathrm{H^+}]^2}{1 + K_{a1}/[\mathrm{H^+}] + K_{a1}K_{a2}/[\mathrm{H^+}]^2} + W_a + \frac{K_w}{[\mathrm{H^+}]} - [\mathrm{H^+}] = 0 \quad (13)$$

The pH value is the negative logarithm of the hydrogen ion concentration ($\mathrm{pH} = -\log\left[\mathrm{H^+}\right]$), so the pH value can be determined if W_a and W_b are known.

The dynamic process model for the pH neutralization process can be derived from the component balance for the reaction invariants [12]:

$$V\frac{dW_{a4}}{dt} = q_1(W_{a1} - W_{a4}) + q_2(W_{a2} - W_{a4}) + u(W_{a3} - W_{a4})$$
$$V\frac{dW_{b4}}{dt} = q_1(W_{b1} - W_{b4}) + q_2(W_{b2} - W_{b4}) + u(W_{b3} - W_{b4}) \quad (14)$$

In the above dynamic process model, it is assumed that the flow rates and the concentrations of the feed and buffer streams are known except for two properties, W_{a1} and W_{b2} and they consists of unknown parameters (θ). From this assumption, the dynamic process model, Eq. (14) can be rearranged to the following state space model [12]:

$$\dot{x} = f(x,t) + g(x,t)u + F_\theta(t)\theta, \quad (15)$$
$$c(x,y) = 0$$

where

$$f(x,t) = \frac{1}{V}\begin{bmatrix} q_2(W_{a2} - x_1) - q_1x_1 \\ q_1(W_{b1} - x_2) - q_2x_2 \end{bmatrix}, \ g(x,t) = \frac{1}{V}\begin{bmatrix} W_{a3} - x_1 \\ W_{b3} - x_2 \end{bmatrix},$$

$$F_\theta(t) = \frac{1}{V}\begin{bmatrix} q_1 & 0 \\ 0 & q_2 \end{bmatrix}, \theta = \begin{bmatrix} W_{a1} & W_{b2} \end{bmatrix}^T, x = \begin{bmatrix} W_{a4} & W_{b4} \end{bmatrix}^T,$$

$u = q_3$, y=pH$_4$, $pK_1 = -\log K_{a1}$, $pK_2 = -\log K_{a2}$,

$$c(x,y) = x_1 + 10^{y-14} - 10^{-y} + x_2\frac{1 + 2\times 10^{y-pK_2}}{1 + 10^{pK_1 - y} + 10^{y-pK_2}} = 0 \ .$$

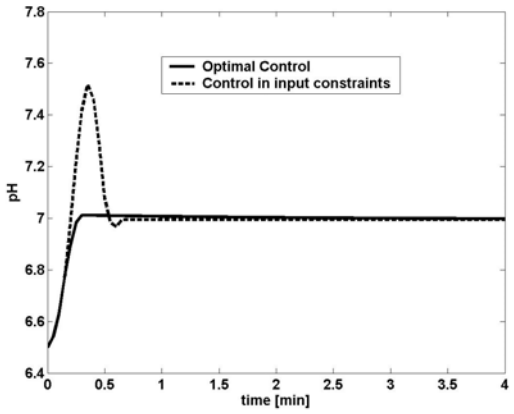
3.3 Optimal Control Strategy

In the "Simulation Part" of SAE algorithm (Fig. 1), the suboptimal control law is necessary to get good training data sets. If we want to improve the performance of SAE control, originally, the Bellman equation and Bellman iteration with new data have to be used. However, If the suboptimal control is close to optimal control, the improvement of cost-to-go function by Bellman iteration is not necessary. Fortunately, in this process, an optimal control can be devised from a simple principle. The required flow rate of titrating stream to make the mixture of inlet streams with the desired pH value (S+S$_1$ in Fig. 3 (b)) can be calculated from the information of the inlet streams (u_0 is the former steady state input and u_f is new steady state input from new set point in Fig. 3 (b)) and the additional amount of titrating stream (S in Fig. 3 (b)) to make the contents of the CSTR with the desired pH value has to be injected in a shortest-possible time.

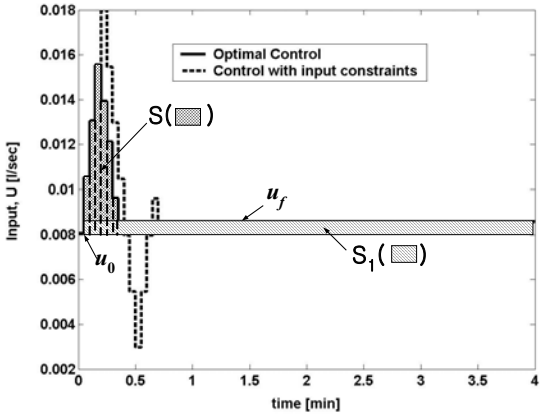
$$\sum_1^n \Delta u_i = u_f - u_0 = \Delta u \ , \quad (16)$$

$$\begin{bmatrix} n & n-1 & \dots & 2 & 1 \end{bmatrix}\begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \vdots \\ \Delta u_n \end{bmatrix} = \left[S/\Delta t + \Delta u \times n \right] \quad (17)$$

where $-\Delta u_{\max} \le \Delta u_i \le \Delta u_{\max}$.



(a) pH change



(b) Input change

Fig. 3 Comparison of results between PI control and SAE by NDP with respect to disturbance.

In this manner, the effluent pH value can be reached to the desired value in shortest time without overshoot or undershoot. This control law is not exactly optimal due to the residence time of the effluent stream considering the constraints of the flow rates but it is close enough to the optimal control law. Moreover, the amount of additional injection of the titrating stream can be adjusted to make the performance better. By using this optimal strategy, the laborious Bellman iteration is omitted in this study.

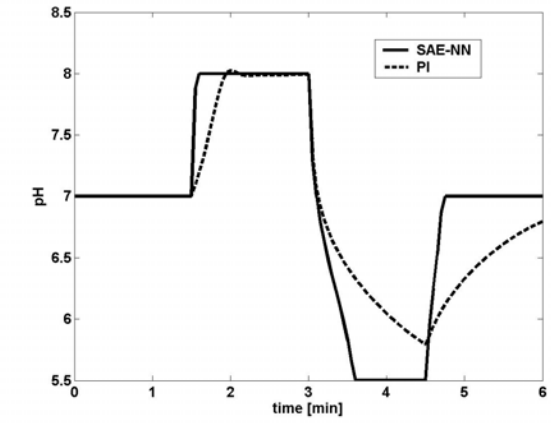
4. RESULTS AND DISCUSSIONS

The SAE algorithm is applied to the pH neutralization process with NN as a global approximator and *k*NN as a nonparametric local optimization . Furthermore, when there are input constraints, the SAE algorithm does not show the good performance and some solutions are suggested for that.

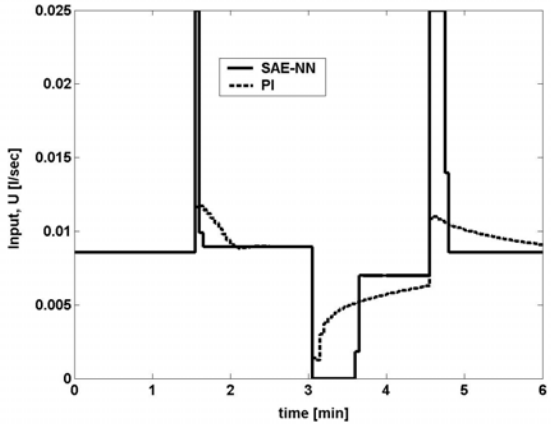
4.1 Results by Neural Network Approximator

As the approximator, the multilayer feedforward NN is used, which consists of two input state, 5 neuron hidden layer,

and 1 neuron output layer. The used weighting factors of one-stage-cost function are $R = 1$, $Q = 1e5$. By the optimal control law previously mentioned, the training data sets are generated for training of NN. The case of multiple step changes in set point (Fig. 4) shows that the SAE by NDP outperforms the well-tuned PI control as expected.

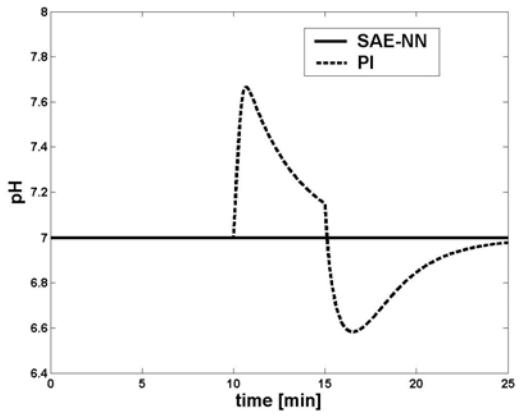


(a) pH change

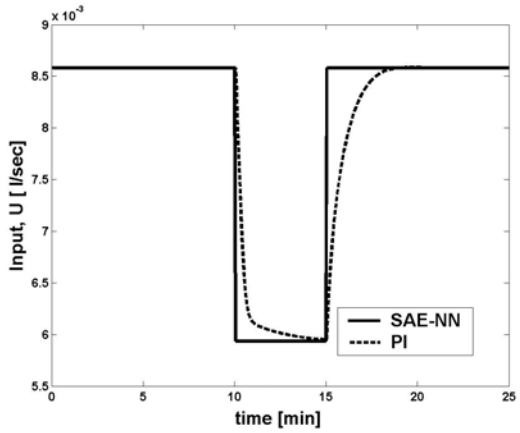


(b) Input change

Fig. 4 Comparison of results between PI control and SAE by NDP with respect to multi-step set point change.



(a) pH change



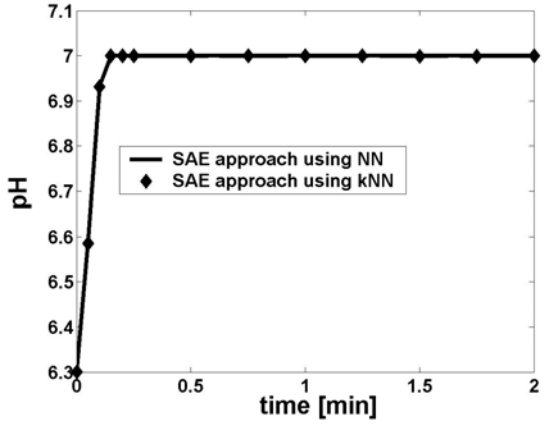
(b) Input change

Fig. 5 Comparison of results between PI control and SAE by NDP with respect to disturbance.

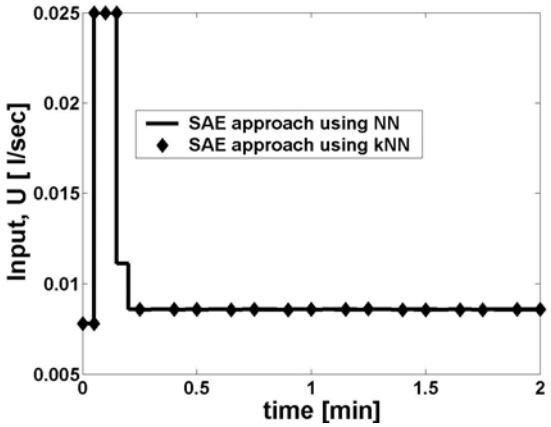
In addition to that, the disturbance case in feed concentration (W_{a1} change at 10 min) are depicted in Fig. 5 and the results shows that the disturbances are well overcome. However, to get rid of small steady-state offset, the weighting factor of the one-stage-cost function on error has to be increased. The small steady-state offset can be observed in SAE case because the NN is an approximation and the precise value cannot be calculated from the NN. To enhance this difficulty, either more data around the steady state should be used for training or a weighting factor adjustment strategy has to be employed.

4.2 Results by *k*-Nearest Neighbor (*k*NN) Approximator

As a local approximator, *k*NN method is also applied. The *k*NN method does not require tedious training as in NN approach and it was very simple to apply. Since our process is relatively simple, only two points nearest neighbor could result a satisfactory performance. The performance using *k*NN method shown in Fig. 6 is almost same as the case using NN. This is because the model we used has only two states and the nonlinearity is not very high. If the process has very complicated nonlinear behavior with many states, the training of neural network is not trivial and many computational issues regarding training and Bellman iteration can be brought out.



(a) pH change



(b) Input change

Fig. 6 Comparison of results between SAE approaches using *k*NN and NN with respect to set point change (pH 6.3→7).

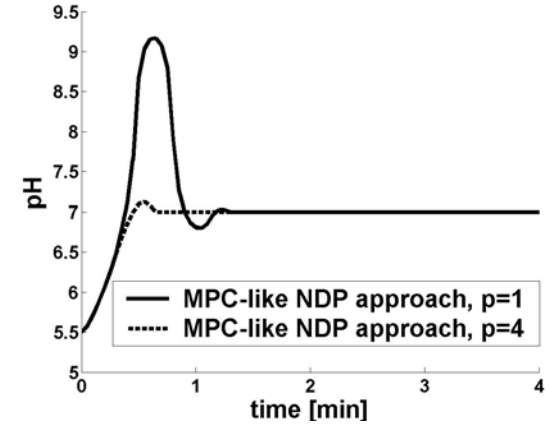
4.3 Results with restriction in Δu_{max}

In the cases of previous section, the simulation results are obtained with no restriction in the size of control movement in one sampling time (Δu_{max}) even though there were the lower and upper limits of the control input size ($0 \le u \le 0.025$ *l*/sec). But if there is a restriction in Δu_{max}, the SAE control strategy which is based on NDP algorithm cannot handle the situation correctly. For example, for a step change in set point, SAE method will calculate the additional amount of titrating stream injection without considering the limitation of the control moves in each sampling time and the response may oscillate and the performance will be deteriorated. The SAE controller will increases the control input to inject the needed amount of titrating stream to compensate the difference in the holdups in CSTR assuming that it can stop injecting immediately when needed, However, due to the limitation of control input movement, it cannot decrease the titrating stream to desired level in a sampling time. Thus, it results over-injection and the process will overshoot and oscillate to compensate the over-injection of the titrating stream. The standard NDP only tries to push the system to the new state as fast as possible under given condition and not to moderate the amount of additional injection considering the limitations and results overshoot and oscillation. In order to prevent this short coming, the standard NDP has to be modified to accommodate the situation. Thus, we suggest that the recursive cost-to-go function calculation should be modified in the following way.

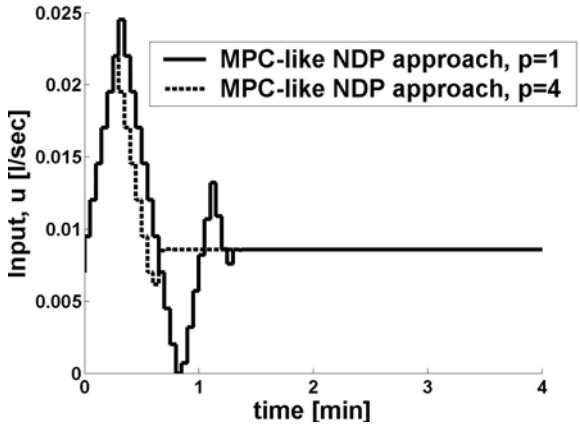
$$J_k^* = \sum_{j=k}^{p+k} \phi_j + J_{k+p}^* \qquad (16)$$

If p=1, Eq. (16) is same as Eq. (6) of original neuro-dynamic approach and if p=∞, it becomes original dynamic programming (DP). This modification increases computational burden to find the optimal input at time step k, but this can prevent the performance degradation due to the constraints on the input change. Fig. 7 shows the performance of the new approach (Δu_{max}=0.0025 *l*/sec) and the overshoot can be reduced significantly in the new approach. Also, the decrease in overshoot for the increase in p is observed. This approach is to incorporate the prediction capability as in Model Predictive Control (MPC) to prevent the performance deterioration due to incorrect information of the process.

Another, yet better approach to resolve this sort of problem is to incorporate the input as a state in the cost-to-go approximation. If we add the information on input to the approximator in NDP algorithm, then the NDP method with augmented states will consider the value of input variable and calculate the required amount of the injection based on the correct information on the process. In Fig. 8, the performance of this approach is shown and the response did not overshoot and provided an excellent performance. In MPC-like approach, the NDP method resulted a small overshoot even with p=4. However, this approach provided better performance with slight increase in computational burden which is almost negligible. Furthermore, even for the disturbances, the NDP controller with augmented states shows good results as in Fig. 9.

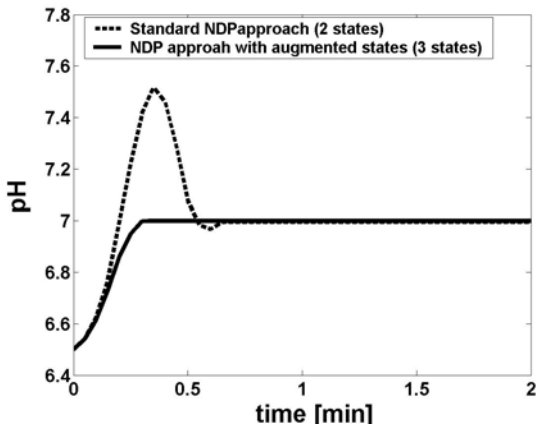


(a) pH change

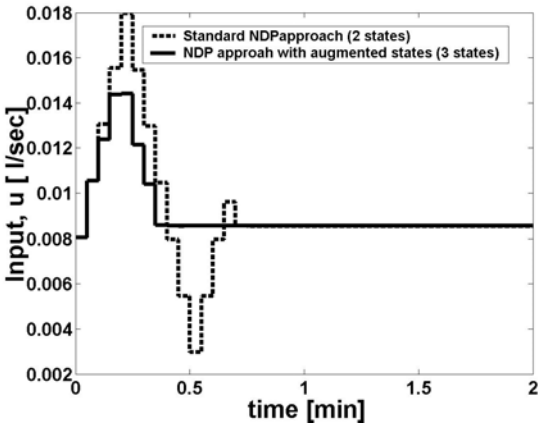


(b) Input change

Fig. 7 Comparison of results between multi-step forms (p=1 and p=4) in SAE by *k*NN with Δu_{max} restriction.

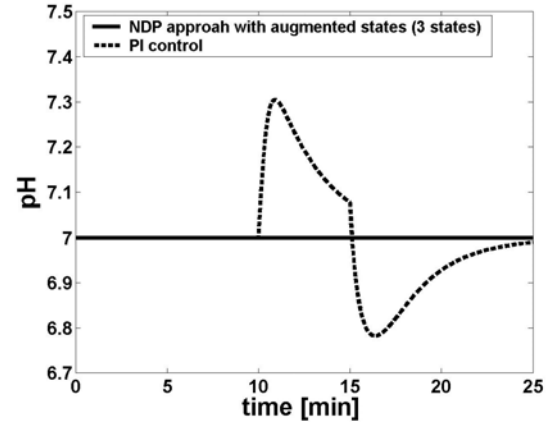


(a) pH change

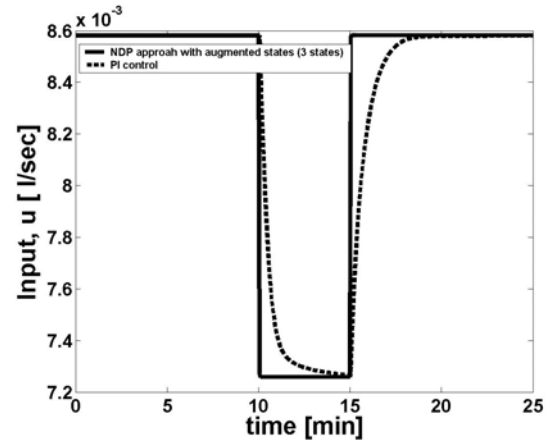


(b) Input change

Fig. 8 Comparison of results between standard NDP approach and NDP approach with augmented states when there is Δu_{max} restriction (In the case of NDP approach using *k*NN).



(a) pH change



(b) Input change

Fig. 9 Comparison of results between PI control and NDP approach using *k*NN with augmented states with respect to disturbance (15% decrease in Wa1 at time=10 min.).

5. CONCLUSIONS

From the simulation of a pH neutralization process, the SAE method which is based on NDP algorithm, using either the global approximator (NN) or the local approximator (*k*NN) outperforms the well-tuned PI control. These results are not surprising because NDP method uses much more information and computation. However, if the process is quite complex, this approach can achieve precise optimal control performance without excessive online computational burden. In this study, the NDP approach is applied to a chemical process of pH neutralization which is a representative benchmark nonlinear process and the possibility of applying DP concept even with short sampling period to complex nonlinear chemical processes is verified. In terms of offline preparation of NDP approach, the local approximators such as *k*NN are preferred over global approximators in the light of cost-to-go approximation. The local approximator can avoid the hard problem of training of approximator. Also, the remedies for the cases of limitation in input movement are suggested.

ACKNOWLEDGEMENTS

This work was supported by the Korea Science and Engineering Foundation (R01-2002-000-00574-0).

REFERENCES

[1] Cutler, C. R., & Ramaker, B. L., "Dynamic matrix control-a computer control algorithm", *In Proceedings of the joint automatic control conference*, 1980.

[2] Garcia, C.E., Prett, D. M., & Morari,M., "Model preditive control: Theory and practice-a survey", *Automatica*, **25**(3), pp. 335-348, 1989.

[3] S. Joe Qin, Thomas A.Badgwell, "A survey of industrial model predictive control technology", *Control Engineering Practice*, **11**, pp. 733-764, 2003.

[4] Niket S. Kaisare, Jong Min Lee and Jay H. Lee, "Simulation based strategy for nonlinear optimal control : Application to a microbial cell reactor," *Int. J. Robust Nonlinear Control*, pp.347-363, 2003

[5] Dimitri P. Bertsekas and John N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Massachusetts, 1996

[6] Arthur E. Bryson, Jr., *Dynamic Optimization*, Addison-Wesley Longman, Inc., California, 1999

[7] Howard Demuth and Mark Beale, *MATLAB neural network toolbox use's guide ver 3.0*, The Math Works, 1998

[8] Ricardo Gutierrez-Osuna, *Lecture Note: Introduction to Pattern Recognition*, Wright state University

[9] Jay H. Lee, *Lecture Note: From Model Predictive Control to Simulation Based Dynamic Programming: A Paradigm Shift*, Georgia Institute of Technology

[10] Henson, M. A. and Seborg, D. E., "Adaptive Nonlinear Control of a pH Neutralization Process", *IEEE Trans. on Control Systems Technology*, **2**, pp.169, 1994

[11] Henson, M. A. and Seborg, D. E. "Adaptive Input-Output Linearization of a pH Neutralization Process", *Int. J. Adapt. Control Signal Process*, **11**, pp.171, 1997

[12] Ahrim Yoo, "Experimental Parameter Identification and Control of pH Neutralization Process Based on an Extended Kalman Filter", *Master Thesis*, Korea University, 2002