Control of pH Neutralization Process using Simulation Based Dynamic Programming (ICCAS 2003)

Dong Kyu Kim 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)

Abstract: 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. For general nonlinear processes, it is difficult to control with a linear model-based control method so nonlinear controls must be considered. Among the numerous approaches suggested, the most rigorous approach is the dynamic optimization. However, as the size of the problem grows, the dynamic programming approach is suffered from the curse of dimensionality. In order to avoid this problem, the Neuro-Dynamic Programming (NDP) approach was proposed by Bertsekas and Tsitsiklis (1996). The NDP approach is to utilize all the data collected to generate an approximation of optimal cost-to-go function which was used to find the optimal input movement in real time control. The approximation could be any type of function such as polynomials, neural networks and etc. In this study, an algorithm using NDP approach was applied to a pH neutralization process to investigate the feasibility of the NDP algorithm and to deepen the understanding of the basic characteristics of this algorithm. As the global approximator, the neural network which requires training and k-nearest neighbor method which requires querying instead of training are investigated. The global approximator requires optimal control strategy. If the optimal control strategy is not available, suboptimal control strategy can be used even though the laborious Bellman iterations are necessary. For pH neutralization process it is rather easy to devise an optimal control strategy. Thus, we used an optimal control strategy and did not perform the Bellman iteration. Also, the effects of constraints on control moves are studied. From the simulations, the NDP method outperforms the conventional PID control.

Keywords: pH neutralization process, the NDP (Neuro-Dynamic Programming), the SAE (simulation-approximation-evolution) algorithm , k-nearest neighbor method

1. INTRODUCTION

Generally, 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 [1]. 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 and to calculate the optimal strategy using the approximator online. 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 method.

In this study, Simulation-Approximation-Evolution (SAE) algorithm suggested by Kaisare et al. [1] is investigated against a pH neutralization process. Through the simulations, the neural network and KNN method are compared. An optimal control of pH neutralization process to avoid the Bellman iteration is suggested and the effects of constraints on input moves are investigated.

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 [2, 3],

$$x(k+1) = F_h\left(x(k), u(k)\right) \quad (1)$$

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, …, N–1 to minimize a performance index which is related with cost-to-go function.

Before defining the cost-to-go function, the one-stage-cost, ϕ should be defined. Among many ways, the most popular one-stage-cost can be chosen as follows, with the weighting factors Q and R.

$$\phi(x(k), u(k)) = \{Q \times [x(k+1) - x_{sp}]^2\} + \{R \times \Delta u(k)^2\} . \quad (2)$$

where k=0, …, N-1, $u(0)=u_0$, and x_{sp} denotes the set point.

Then the optimal cost-to-go can be expressed as follow.

$$J_k(x(k)) = \min_u \mathrm{E}\left[\sum_{i=k}^{N-1} \phi(x(i),u(i)) + \phi_N\right] \quad (3)$$

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

$$J_k(x(k)) = \min_u \mathrm{E}\left[\phi(x(k),u(k)) + J_{k+1}(F_h(x(k),u(k))\right] \quad (4)$$

It can be shown to satisfy the Bellman equation [2].

$$J^*(k) = \min_u \mathrm{E}\left[\phi(x(k),u(k)) + J^*(k+1)\right] \quad (5)$$

where E[•] denotes expected value and superscript * implies the optimal value. For simplicity, $J^*(x(k))$ will be shortened as $J^*(k)$. The final goal of DP is to find the input strategy $u(k)$, k=1, …, 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-Approximation-Evolution (SAE) algorithm

SAE algorithm [1] 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.

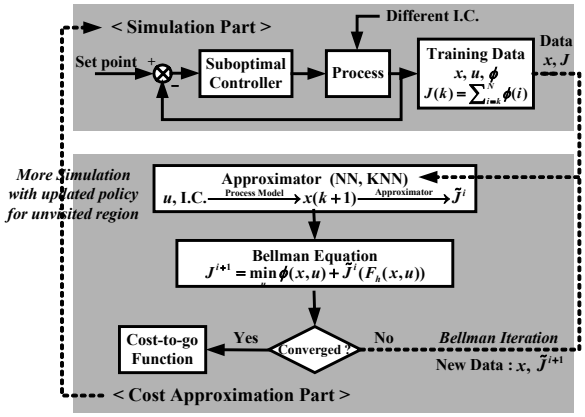


Figure 1. Architecture for offline computation of cost-to-go approximation.

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. (5)) for each state visited during the simulation, and the suboptimal cost-to-go function is calculated by

$$J(k) = \sum_{i=k}^{N} \phi(i)\,. \quad (6)$$

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 [1].

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

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 [4] as in Figure 2. 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.

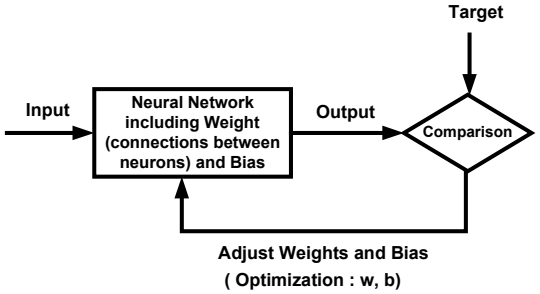


Figure 2. The schematic diagram for neural networks.

2.3.2 K-Nearest Neighbor method

The k-nearest neighbor (KNN) 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 KNN only requires an integer k, a set of labeled examples (training data), and a metric to measure closeness [6]. The KNN 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 [5].

2.4 Bellman Iteration

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 [1].

$$\tilde{J}^{i+1} = \min_{u} \phi(x,u) + \tilde{J}^{i}\left(F_h(x,u)\right) \quad (7)$$

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

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 (Henson and Seborg, 1994, 1997). The process is illustrated in Fig. 3 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 [7]. Table 1 shows the typical operating conditions of the process of concern.

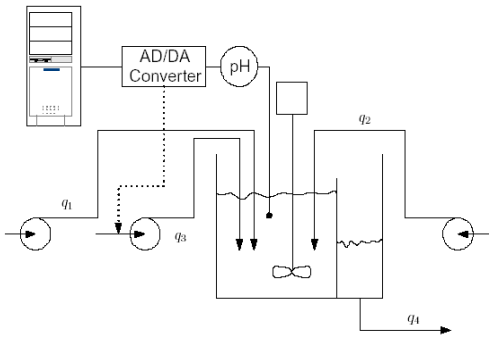


Figure 3. The pH neutralization process.

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) [7].

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 [7].

$$\begin{aligned} &H_2CO_3 \leftrightarrow HCO_3^- + H^+, \\ &HCO_3^- \leftrightarrow CO_3^{2-} + H^+, \\ &H_2O \leftrightarrow OH^- + H^+. \end{aligned} \quad (8)$$

The equilibrium constants for the reactions are defined as [7]:

$$K_{a1} = \frac{[HCO_3^-][H^+]}{[H_2CO_3]},\ K_{a2} = \frac{[CO_3^{2-}][H^+]}{[HCO_3^-]},\ K_w = [H^+][OH^-] \quad (9)$$

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) [7].

Reaction invariants for this process are defined as:

$$\begin{aligned} &W_{ai} = [H^+]_i - [OH^-]_i - [HCO_3^-]_i - 2[CO_3^{2-}]_i\ , \\ &W_{bi} = [H_2CO_3]_i + [HCO_3^-]_i + [CO_3^{2-}]_i\ . \end{aligned} \quad (10)$$

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. 3.

The output equation is derived from Eqs. (9)~(10) which represent the relation between a hydrogen ion concentration and reaction invariants [7].

$$W_b \frac{K_{a1}/[H^+] + 2K_{a1}K_{a2}/[H^+]^2}{1 + K_{a1}/[H^+] + K_{a1}K_{a2}/[H^+]^2} + W_a + \frac{K_w}{[H^+]} - [H^+] = 0 \quad (11)$$

The pH value is the negative logarithm of the hydrogen ion concentration ($\mathrm{pH} = -\log\left[H^+\right]$), 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 [7]:

$$\begin{aligned} V\frac{dW_{a4}}{dt} &= q_1(W_{a1} - W_{a4}) + q_2(W_{a2} - W_{a4}) + u(W_{a3} - W_{a4}) \\ V\frac{dW_{b4}}{dt} &= q_1(W_{b1} - W_{b4}) + q_2(W_{b2} - W_{b4}) + u(W_{b3} - W_{b4}) \end{aligned} \quad (12)$$

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. (13) can be rearranged to the following state space model [7]:

$$\begin{aligned} &\dot{x} = f(x,t) + g(x,t)u + F_\theta(t)\theta \\ &c(x,y) = 0 \end{aligned} \quad (13)$$

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$_4$, $pK_1=-\log K_{a1}$, $pK_2=-\log K_{a2}$,

$$c(x,y)=x_1+10^{y-14}-10^{-y}+x_2\frac{1+2\times10^{y-pK_2}}{1+10^{pK_1-y}+10^{y-pK_2}}=0\ .$$

3.3 Optimal Control Strategy

The suboptimal control law is used in the "Simulation Part" of SAE algorithm. 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 can be calculated from the information of the inlet streams and the additional amount of titrating stream to make the contents of the CSTR with the desired pH value has to be injected in a shortest-possible time. 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.

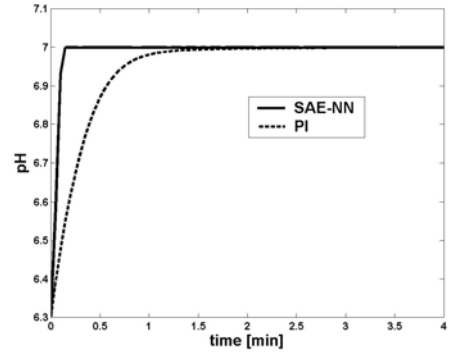
4. RESULTS AND DISCUSSIONS

The SAE algorithm is applied to the pH neutralization process with each a global approximator (NN) and a nonparametric local optimization (KNN).

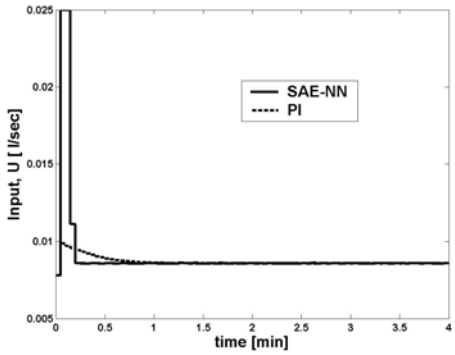
4.1 Results by Neural Network Approximator

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 $R=1$, $Q=1$. For training of NN, the optimal control law mentioned in the previous section is used to generate the training data. The comparisons of the results between well-tuned PI control and SAE by NDP for a step change in set point (pH 6.3→7) are shown in Fig. 4. The SAE by NDP is much better than PI control as shown Fig. 4.

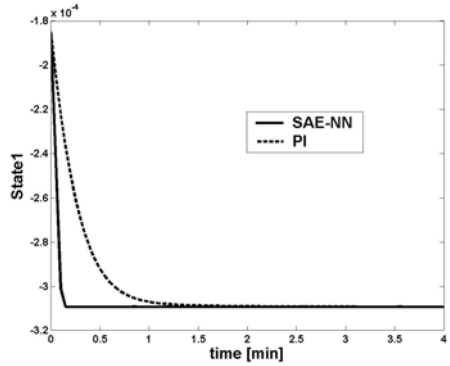
The case of multiple step changes in set point and the disturbance case in feed concentration (W_{a1} change at 10 min) are depicted in Figures 5 and 6, respectively. From these results, the SAE by NDP outperforms the well-tuned PI control as expected. However, the weighting factor of the one-stage-cost function on error has to be increased to get rid of small steady-state offset. 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.



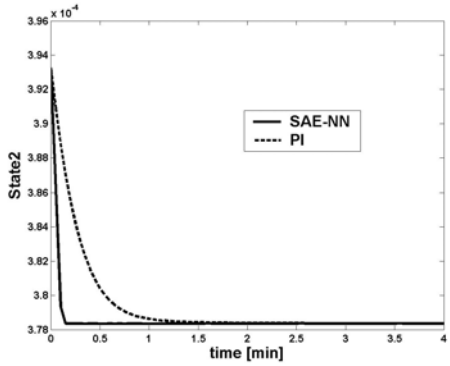
(a) pH change



(b) Input change



(c) State 1 (W_{a4}) change



(d) State 2 (W_{b4}) change

Figure 4. Comparison of results between PI control and SAE by NDP with respect to a step change in set point (pH 6.3→7).

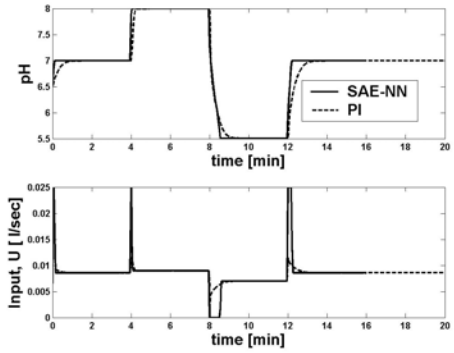
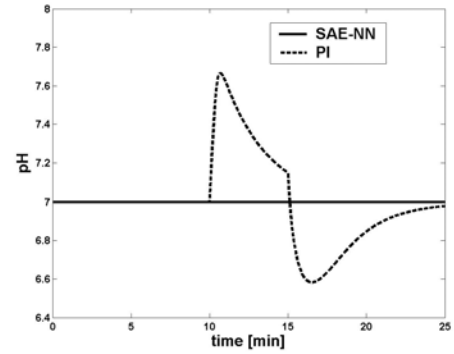
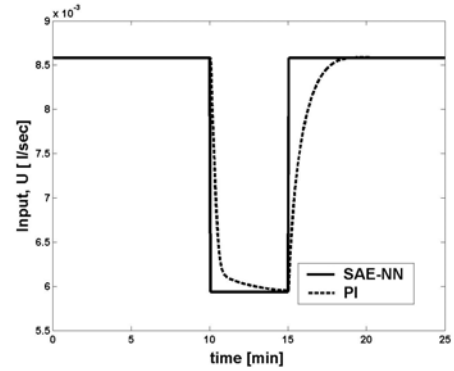


Figure 5. Comparison of results between PI control and SAE by NDP with respect to multi-step set point change.



(a) pH change



(b) Input change

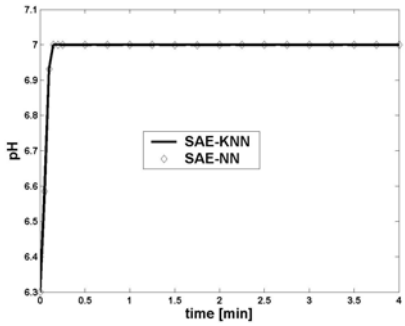
Figure 6. Comparison of results between PI control and SAE by NDP with respect to disturbance.

4.2 Results by K-Nearest Approximator

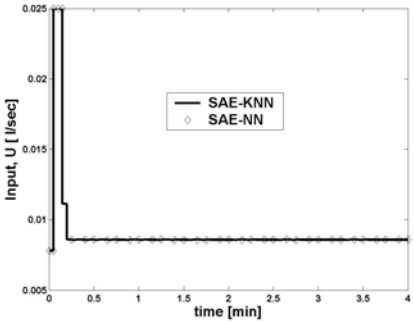
As a local approximator, KNN method is also applied. The KNN method does not require tedious training as in NN approach and it was very simple to apply. Since out process is relatively simple, only two points nearest neighbor could result a satisfactory performance. The performance using KNN method shown in Figure 7 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.

4.3 Results with restriction in $\Delta u_{\max}$

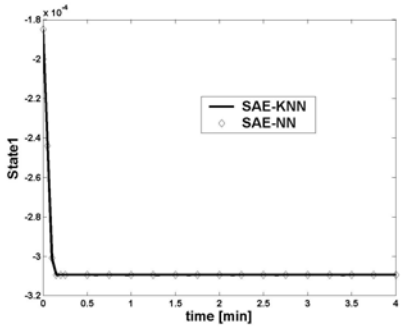
In the cases of previous study (**4.1** and **4.2**), the results is acquired with no restriction in $\Delta u_{\max}$ ($\Delta u_{\max} = u_{\max} = 0.025$). But if there is the restriction in $\Delta u_{\max}$, the SAE control strategy cannot handle the situation correctly. To make a smooth landing on the new set point, for example, the amount of accumulation of input change should not exceed the required value (additional amount of titrating stream injection).



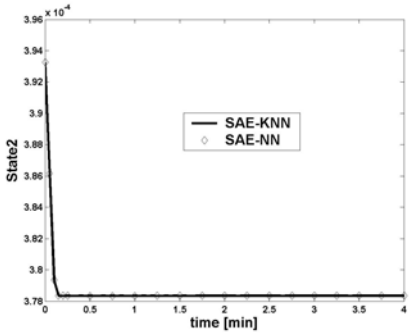
(a) pH change



(b) Input change



(c) State 1 (W_{a4}) change



(d) State 2 (W_{b4}) change

Figure 7. Comparison of results between SAE by KNN and SAE by NN with respect to set point change (pH 6.3→7).

If it exceeds, the system will overshoot and oscillate to compensate the over-injection of the titrating stream. However, the standard SAE 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 and results overshoot and oscillation. In order to prevent this short coming, the standard SAE 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.

$$J_k^* = \sum_{j=k}^{p+k} \phi_j + J_{k+p}^* \qquad (14)$$

If p=1, Eq. (14) is same as Eq. (5) 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. Figure 8 shows the performance of the new approach ($\Delta u_{\max}$=0.002) and the overshoot can be reduced significantly in the new approach. Also, the decrease in overshoot for the increase in p is observed.

5. CONCLUSIONS

From the simulation of a pH neutralization process, the SAE algorithm using the global approximator (NN) or the local approximator (KNN) outperforms the well-tuned PI control. These results are not surprising because SAE uses much more information and computation. However, if the process is quite complex, this approach will achieve precise optimal control performance without excessive online computational burden. In this study, the new approach, SAE is applied to a chemical process and the feasibility is investigated. Also, a modification of neuro-dynamic programming approach for the case of constraints in input change is suggested and the performance is verified through the simulation.

REFERENCES

[1] 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

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

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

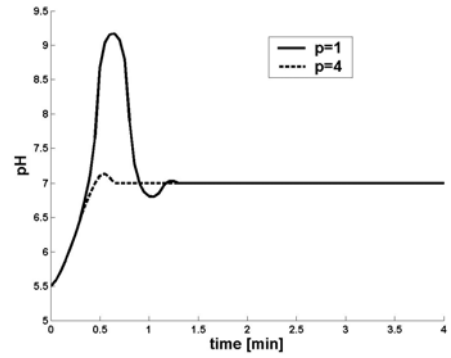
[4] Howard Demuth and Mark Beale, *MATLAB neural network toolbox use's guide ver 3.0*, The Math Works, 1998

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

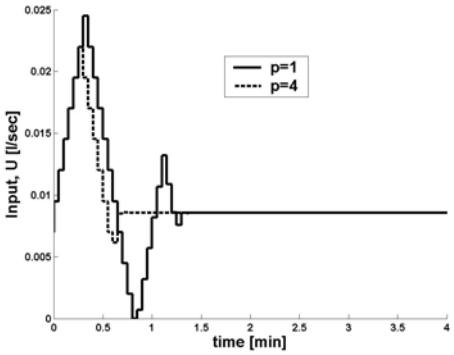
[6] Ricardo Gutierrez-Osuna, *Lecture Note: Introduction to Pattern Recognition*, Wright state University

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

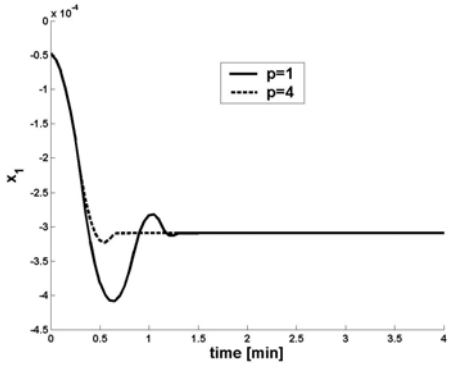
[8] J. Choi, J.H. Lee, and M.J. Realff, "An algorithmic framework for improving heuristic solutions part II: A new version of stochastic traveling salseman problem", *Computers and Chemical Engineering*, Submitted To, 2002.



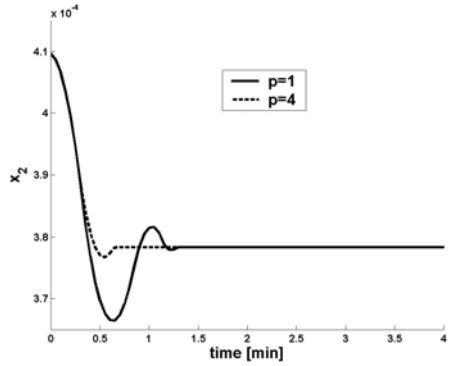
(a) pH change



(b) Input change



(c) State 1 (W_{a4}) change



(d) State 2 (W_{b4}) change

Figure 8. Comparison of results between multi-step forms (p=1 and p=4) in SAE by KNN with $\Delta u_{\max}$ restriction.