Proceeding of the 2004 American Control Conference
Boston, Massachusetts June 30 - July 2, 2004

Empirical Model Based Control of Nonlinear Processes using Approximate Dynamic Programming

Jong Min Lee and Jay H. Lee*
School of Chemical and Biomolecular Engineering
Georgia Institute of Technology
311 Ferst Dr., Atlanta, GA 30332-0100, USA
{jongmin.lee, jay.lee}@chbe.gatech.edu

***Abstract*— A major difficulty associated with using an *empirical nonlinear* model for model-based control is that the model can be unduly extrapolated into regions of the state space where identification data were scarce or even nonexistent. Optimal control solutions obtained with such over-extrapolations can result in performances far worse than predicted by the model. In the multi-step predictive control setting, it is not straightforward to prevent such overuse of the model by forcing the optimizer to find a solution within the "trusted" regions of state space. Given the difficulty, we propose an *Approximate Dynamic Programming* based approach for designing a model-based controller that avoids such abusage of an empirical model with respect to the distribution of the identification data. The approach starts with closed-loop test data obtained with some suboptimal controllers, e.g., PI controllers, and attempts to derive a new control policy that improves upon their performances. Iterative improvement based on successive closed-loop testing is possible. A diabatic CSTR example is provided to illustrate the proposed approach.**

I. INTRODUCTION

Model predictive control (MPC) is the most popular advanced control technique in the process industry owing to its ability to handle complex multivariable control problems with constraints. In a typical MPC scheme, a dynamic model is used to build a prediction of future output behavior, based on which an online optimization finds a sequence of input moves that minimize the output deviation from a set-point trajectory. Thus, an accurate model is essential for a successful outcome.

While dynamic behavior of most chemical processes is nonlinear, linear models are used predominantly in practice because of the difficulty associated with building an accurate nonlinear model, either by first principles or system identification. For a widespread use of nonlinear model-based control, a model-based control method tightly integrated with a nonlinear system identification methodology needs to emerge. Currently, popular nonlinear model structures studied include Volterra series models, block-oriented models such as Hammerstein and Wiener models, bilinear models, and NARX (Nonlinear AutoRegressive with eXogeneous input) models [1].

To perform a nonlinear system identification, one must first decide on the regressor structure. For example, for the NARX model of the form

$$\hat{y}(k+1) = f\left(y(k), \cdots, y(k-n_y), u(k), \cdots, u(k-n_u)\right) \tag{1}$$

one must choose the order parameters n_y and n_u. It has been suggested that this be done by using methods such as the False Nearest Neighborhood and Akaike Information Criterion. This is followed by choosing a functional structure of f, which can be a basis function series expansion or a neural network. Finally, the unknown parameters of f are determined by least squares estimation. All three steps can contribute significantly to model error. Given little prior knowledge one typically has in the beginning, the model order may have to be chosen high and the number of parameters very large to avoid bias, leading to an ill-conditioned estimation problem. This is particularly serious given the difficulty associated with designing and implementing a persistently exciting signal for a general nonlinear model structure. For example, most industrial processes can only be mildly perturbed around a few operating points and this may result in insufficient information about many parts of transient nonlinear dynamics [2]. Control actions and performance predictions calculated from a resulting model may not be reliable given the large variances of the model and the MPC's inability to account for them systematically.

Most of the studies on nonlinear-empirical-model-based control have not addressed the accounting of uncertainty. One exception is [3] who derived the input weighting function for a 2nd order Volterra model with stochastic parameters based on minimization of an expected cost value. It is worth noting that most formulations for robust MPC, even those for linear systems, are based on *open-loop* performance objective, which can lead to very conservative control actions [4].

In an attempt to address several shortcomings of the current MPC formulation, including its inability to account for model uncertainty properly in feedback control's context, Lee and Lee [5] suggested an Approximate Dynamic Programming (ADP) based strategy, which was inspired by the approaches of Reinforcement Learning (RL) [6] and Neuro-Dynamic Programming (NDP) [7] developed by the Artificial Intelligence (AI) community. This approach attempts to solve the DP approximately only within limited regions

*To whom all correspondence should be addressed.

0-7803-8335-4/04/$17.00 ©2004 AACC

of state space by utilizing simulation and function approximation so as to combat the 'curse-of-dimensionality'. From the method, one gets an approximation of the cost-to-go function, which maps the state to the cost-to-go value. The cost-to-go map then defines an online control policy, which involves solving a one-stage optimization problem rather than a multi-stage one. In addition, uncertainties can be accounted for by including them in the simulation (e.g., via Monte Carlo simulation). It showed promise in several nonlinear control problems [8], [9] and a stochastic optimal control problem [10].

In this paper, building upon our previous work on ADP, we present a cost-to-go based control scheme that utilizes an empirical nonlinear model and is designed to use the empirical model only in the regions of the state space with sufficient amounts of identification data. To this end, a localized approximator is employed for building the cost-to-go function approximation. A quadratic penalty term based on the local data density is added in order to discourage the optimizer from finding a solution in the regions without adequate amounts of identification data. The approach is applied on a diabatic CSTR example to highlight the performance differences with the conventional MPC approach.

II. MODEL PREDICTIVE CONTROL USING A NARX MODEL STRUCTURE

In this section, we present a model predictive control formulation based on a NARX model structure, which is given as

$$\begin{aligned} y(k+1) &= f\left(y(k),\ldots,y(k-n_y),u(k),\ldots,u(k-n_u)\right) \\ &+ e(k+1) \end{aligned} \tag{2}$$

where e is a white noise. f can be parameterized in many different forms such as neural network, polynomial, etc.

A state space realization of Eq. (2) is constructed as

$$x^T(k) = [y(k),\cdots,y(k-n_y),u(k-1),\cdots,u(k-n_u)] \tag{3}$$

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 0 & \cdots & 0 & 0 & & & & \\ I & & & 0 & & & & \\ & \ddots & & \vdots & & & & \\ & & I & 0 & & & & \\ & & & & 0 & \cdots & 0 & 0 \\ & & & & I & & & 0 \\ & & & & & \ddots & & \vdots \\ & & & & & & I & 0 \end{bmatrix} x(k) \\ &+ \begin{bmatrix} f(x(k),u(k)) \\ 0 \\ \vdots \\ 0 \\ u(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} e(k+1) \end{aligned} \tag{4}$$

$$y(k) = \begin{bmatrix} I & 0 & \cdots & 0 \end{bmatrix} x(k) \tag{5}$$

We denote Eq. (4) as

$$x(k+1) = F(x(k),u(k)) \tag{6}$$

In using the model for control, in order to compensate for a possible plant/model mismatch and assure integral action, we add to the output of the NARX model an integrated white noise $\zeta(k)$. Hence, the overall model becomes

$$\begin{aligned} x^{aug}(k) &= \begin{bmatrix} x(k) \\ \zeta(k) \end{bmatrix} = \begin{bmatrix} F(x(k-1),u(k-1)) \\ \zeta(k-1) \end{bmatrix} \\ &+ \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} e_1(k) \\ e_2(k) \end{bmatrix} \end{aligned} \tag{7}$$

where $e_1(k)$ and $e_2(k)$ are independent white noises with covariance matrices R_1 and R_2, respectively, and

$$G = \begin{bmatrix} I & 0 & \cdots & 0 \end{bmatrix}^T \tag{8}$$

Eq. (7) can be recast as

$$\begin{cases} x^{aug}(k) &= \mathcal{F}(x^{aug}(k-1),u(k-1)) + \Gamma^e e(k) \\ y(k) &= Hx^{aug}(k) + \nu(k) \end{cases} \tag{9}$$

where

$$H = \begin{bmatrix} I & 0 & \cdots & 0 & I \end{bmatrix} \tag{10}$$

and

$$\Gamma^e = \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix} \tag{11}$$

Here artificial white noise ν is added in the model output for tuning purposes. The augmented state can be estimated using the following conventional EKF approach:

Prediction

$$\hat{x}^{aug}(k|k-1) = \mathcal{F}\left(\hat{x}^{aug}(k-1|k-1),u(k-1)\right) \tag{12}$$

$$\hat{y}(k|k-1) = H\hat{x}^{aug}(k|k-1) \tag{13}$$

Measurement correction

$$\hat{x}^{aug}(k|k) = \hat{x}^{aug}(k|k-1) + L(k)\left(\tilde{y}(k) - \hat{y}(k|k-1)\right) \tag{14}$$

$$\hat{y}(k|k) = H\hat{x}^{aug}(k|k) \tag{15}$$

EKF gain calculation

$$L(k) = \Sigma(k|k-1)H^T\left(H\Sigma(k|k-1)H^T + R^\nu\right)^{-1} \tag{16}$$

$$\Sigma(k|k-1) = \Phi(k-1)\Sigma(k-1|k-1)\Phi(k-1)^T + \Gamma^e R^e (\Gamma^e)^T \tag{17}$$

$$\Sigma(k|k) = \left(I - L(k)H\right)\Sigma(k|k-1) \tag{18}$$

where $\Phi(k-1) = \begin{bmatrix} A(k-1) & 0 \\ 0 & I \end{bmatrix}$, $R^e = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}$, and $A(k-1) = \frac{\partial F}{\partial x}\big|_{\hat{x}(k-1|k-1),u(k-1)}$.

Given the state estimate at each time, multi-step predictions of y can be computed recursively for the purpose of optimal control move calculation. A successive linearization based scheme suggested by Lee and Ricker [11] computes

the output predictions using the nonlinear model under the constant input assumption and then adds the effect of additional input moves based on the linearized model in order to obtain an affine prediction model with respect to the input moves. This leads to a quadratic program (QP) instead of a nonlinear program (NLP) for the optimal control calculation. The detailed algorithm can be found in the paper. We compare the performances of this algorithm and full-fledged nonlinear MPC with the proposed ADP based method in this paper.

III. APPROXIMATE DYNAMIC PROGRAMMING

In DP, one attempts to solve the following 'Bellman equation' for an entire state space:

$$J^*_\infty(x) = \min_u \{\phi(x,u) + \alpha J^*_\infty(F_{t_s}(x,u))\} \tag{19}$$

where $\phi(x,u)$ is the single-stage cost, J^*_∞ is the α-optimal cost-to-go function for the infinite horizon problem, $F_{t_s}(x,u)$ is the successor state of x by applying control action u and α is a discount factor between 0 and 1. Stochastic version of the above equation is

$$J^*_\infty(\mathcal{I}) = \min_u E\{\phi(x,u) + \alpha J^*_\infty(F_{t_s}(\mathcal{I},u))|\mathcal{I}\} \tag{20}$$

where $\mathcal{I}$ is the information vector available at each time. For Gaussian systems, it would consist of the state estimate and the error covariance matrix.

Traditionally, one attempted to solve the equation numerically through discretization of the state space but this caused the computation to become unwieldy, even for a very small size problem. The approximate approach attempts to solve the equation in a limited and approximate sense, by performing simulations with a number of suboptimal policies, building an approximate cost-to-go function using some function approximator, and then improving the cost-to-go function approximation through the iteration of the Bellman equation (as in 'value iteration') or the iteration between the Bellman equation and the policy evaluation (as in 'policy iteration'). The cost-to-go approximation is made only in the regions of state space visited during the closed-loop simulation, thereby significantly reducing the computational load. The rationale behind this is that, even though the state space may be huge, only a very small portion of it will actually be relevant under optimal control, given a small number of disturbances and setpoint changes occurring in real operations. In this sense, the dimension of state space should not be the main determining factor for the DP computation; rather it should be the number of operating points and potential disturbances. The obstacle though is that since one does not have the optimal controller, the working state space regions for the optimal controller under given disturbances and setpoint changes cannot be identified. Hence, one attempts to construct a superset that includes the relevant regions by closed-loop simulations with judiciously chosen suboptimal policies. If one is not successful, the resulting control policy will be only suboptimal, even though it may still represent a great improvement from the starting suboptimal policies. Since the state space for most chemical process control problems is continuous, some function approximation scheme based on sampled data should be employed for the cost-to-go approximation.

In summary, the proposed scheme is characterized by simulation, function approximation, and evolutionary improvement of a control policy within limited regions of state space. The resulting converged cost-to-go function should yield an approximately optimal (or at least improved) control policy. The following is the basic algorithm of the suggested approach.

1) Perform closed-loop simulations (or identification experiments) with judiciously chosen suboptimal control policies (μ^0) under all representative operating conditions. μ^0 can be the PI controller, MPC, etc.
2) Using the simulation (identification) data, calculate the infinite (or finite) horizon cost-to-go for each state (J^{μ^0}) visited during the simulation.

$$J^{\mu^0}(x(k)) = \sum_{i=0}^{\infty} \alpha^i \phi(x(k+i), u(k+i)) \tag{21}$$

3) Construct a function approximator for the data to approximate the cost-to-go as a function of the continuous state variables, denoted hereafter as $\tilde{J}^{\mu^0}$.
4) To improve the cost-to-go approximation, perform the value or policy iteration until convergence. In this work, we use the value iteration algorithm. The difference with the policy iteration algorithm can be found in [12]. The value iteration is one step approach where a 'greedy' policy is taken based on the current cost-to-go approximation. The greedy policy chooses an action for each state that will result in the next state with the lowest cost-to-go value, i.e., in each iteration loop, we calculate J^{i+1} for the given sample points of x by solving

$$J^{i+1}(x) = \min_u \left\{\phi(x,u) + \alpha \tilde{J}^i\left(F_{t_s}(x,u)\right)\right\} \tag{22}$$

where i denotes ith iteration step. Once the cost-to-go values are updated for all the states, then we fit another function approximator to the x vs. $J^{i+1}(x)$ data. Note that F_{t_s} is the identified NARX model in our case.
5) Once the cost-to-go converges, one can implement the control policy defined by the solution to the following optimization problem:

$$u(k) = \arg\min_{u(k)} \left[\phi(x(k),u(k)) + \alpha \tilde{J}^*\left(x(k+1)\right)\right] \tag{23}$$

where $\tilde{J}^*$ is the converged cost-to-go and $x(k+1) = F_{t_s}(x(k),u(k))$.

IV. PROPOSED APPROACH

One potential problem of using the cost-to-go approximation based on the simulation or identification data is that it is only accurate within the regions of the state space where adequate amounts of data existed. During online optimization, however, the optimizer can easily push the solution to other regions where little data existed thus rendering the cost-to-go value from the approximator not trustworthy. Thus, quantifying the accuracy of the cost-to-go estimate and incorporating this information into the optimization is essential for success of the suggested approach. We found that global approximators such as neural networks can amplify an approximation error, during the offline value iteration, whereas a certain class of localized function approximators including the k-nearest neighbors and kernel-based local averager show nice convergence behavior [13]. Local averaging has another advantage that quantifying the confidence in the cost-to-go approximation can be done very naturally based on the local data distribution. The measure of confidence can be used to define a 'risk' term, which can be included in the objective function to discourage the controller from venturing into the regions of state space for which the confidence is low.

In this paper, we employ a distance-weighted k-nearest neighbor approximation scheme defined as

$$\tilde{J}(x) = \sum_{x_i \in N_k(x)} w_i J_i \tag{24}$$

where

$$w_i = \frac{1/d_i}{\sum_i 1/d_i} \tag{25}$$

,

$$d_i = \|x - x_i\|_2 \tag{26}$$

, and $N_k(x)$ is the k-nearest neighboring points of x.

We can show that this local averager scheme *guarantees* the convergence of the offline value iteration, but it can still introduce significant bias where the data amount is not adequate. Thus, it is important to use the approximated cost-to-go values cautiously by considering the data distribution associated with them. To estimate the local data density, a multi-dimensional Parzen probability density function [14] is employed. Suppose that we have a training data set Ω and a new query point x. Estimate of Parzen density, denoted here by $f_\Omega(x)$ is obtained as a sum of kernel functions placed at each sample in Ω.

$$f_\Omega(x) = \frac{1}{N\sigma^{m_0}} \sum_{i=1}^{N} \frac{1}{(2\pi\sigma^2)^{\frac{m_0}{2}}} \exp\left(-\frac{\|x - x_i\|_2^2}{2\sigma^2}\right) \tag{27}$$

where $x, x_i \in \mathbb{R}^{m_0}$, m_0 is the state dimension, and σ is a user-given bandwidth parameter. Note that N is the number of neighboring data points for estimating cost-to-go.

Then, we incorporate into the cost-to-go a quadratic penalty term based on $f_\Omega(x)$:

$$\tilde{J}(x) \Leftarrow \tilde{J}(x) + J_{bias}(x) \tag{28}$$

$$J_{bias}(x) = A \cdot H\left(\frac{1}{f_\Omega(x)} - \rho\right) \cdot \left[\frac{\frac{1}{f_\Omega(x)} - \rho}{\rho}\right]^2 \tag{29}$$

where H is a heavy-side step function, A is a scaling parameter, and ρ is a threshold value. In this work, ρ is the data density corresponding to the user-given bandwidth parameter σ, and A is calculated so as to assign some large cost-to-go value J_{max} to J_{bias} at $\|x - x_i\|_2^2 = (3\sigma)^2$.

With the above, we can attempt to learn an improved control policy while utilizing the identified model cautiously. The procedure is described as follows:

1) Perform closed-loop identification experiments in all possible operating regions by injecting dither signals into the control actions.
2) Identify a NARX model by fitting a parameterized structure (e.g., neural network or polynomial, etc) to the data.
3) Perform the value iteration with the identified model and the initial cost-to-go data until the cost-to-go's converge. To bound the cost-to-go in the offline iteration steps, the additive penalty term is set as J_{max} whenever $\tilde{J}(x) \geq J_{max}$.
4) Online control action is calculated from (23).

Note that, in both the offline iteration and the online control calculation, the modified $\tilde{J}$ of (28) is used. The penalty term is intended to prevent the optimizer from exploring into sparse data regions where both the cost-to-go and the identified state transition model are highly uncertain.

Since the approximated cost-to-go based on the local averager is not smooth in general, manipulated variable u is discretized into a set of values for global optimization in order to avoid local minima in solving (22) and (23).

V. SIMULATION EXAMPLE: CSTR

An example of CSTR with a first-order exothermic reaction is considered to highlight the key aspects of the suggested approach.

$$\begin{aligned} \dot{x}_1 &= -x_1 + D_a(1 - x_1)\exp\left(\frac{x_2}{1 + x_2/\varphi}\right) \\ \dot{x}_2 &= -x_2 + BD_a(1 - x_1)\exp\left(\frac{x_2}{1 + x_2/\varphi}\right) \\ &+ \beta(u - x_2) \\ y &= x_1 \end{aligned} \tag{30}$$

where x_1 and x_2 are the dimensionless reactant concentration and reactor temperature, respectively. The input u is the cooling jacket temperature. D_a, φ, B and β are Damköhler number, dimensionless activation energy, heat of reaction and heat-transfer coefficients, respectively. With the following choice of the parameters:

$$D_a = 0.072, \quad \varphi = 20.0, \quad B = 8.0, \quad \beta = 0.3 \tag{31}$$

the system shows three steady states. The control objective is to take the system from a stable equilibrium point ($x_1 = 0.144, x_2 = 0.886, u = 0.0$) to an unstable one ($x_1 = 0.445, x_2 = 2.7404, u = 0.0$).

A. Identification using Closed-Loop Data

A NARX structure with a feed-forward neural network was used to identify f using data from closed-loop operations under a PI controller. One can decide the particular structure of the model by using a step-wise model building algorithm discussed in [15]. With the dimensionless sample time of 0.5, the output of next time step $y(k+1)$ was found to be a function of $[y(k), \cdots, y(k-3), u(k), \cdots, u(k-3)]$ [16]. Thus, the state vector x is defined as in (3) with $n_y = 3$ and $n_u = 3$.

To cover pertinent operating ranges, different controller gains were used under the setpoint of 0.4450. The selected gain values were 9, 6.75, and 4.5 with the same integration time of 83.3 (sample time). For each closed-loop experiment, the input was dithered at each sample time with a random noise generated from [-0.03 0.03] under the uniform distribution. 12 set of experiments were executed and we collected 2820 input-output data points.

The neural network has seven hidden nodes with eight inputs, $x(k), u(k)$, and one output, $y(k+1)$. The parameters were identified with the MATLAB Neural Network Toolbox [17] with the fitting tolerance (MSE) set as 1e-5.

B. Model Predictive Control

We tested both successive linearization based MPC (slMPC) method described in [11] and nonlinear program based MPC (NMPC) with the EKF estimator. A prediction horizon of 7, a control horizon of 1, an output weight of 50 and an input weight of 1 were used. The regulation performances are shown in Fig. 2. Larger control horizon choices were found to yield worse performances, probably due to the optimizer settling at local minima.

Fig. 1 shows clearly that the poor regulation performance during the transient period is due to the extrapolation to unexplored regions. It is noteworthy that the output and

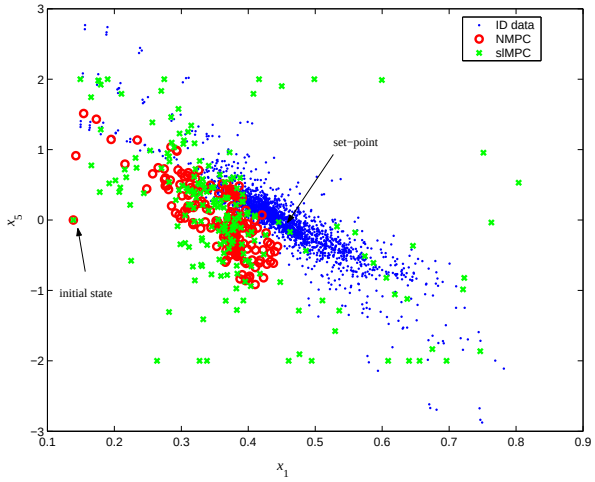

Fig. 1. State trajectories of MPC: plot of x_1 vs. x_5.

input weights had to be detuned significantly in order to achieve closed-loop stability. The ratio between the output weight and the input weight had to be decreased to 5.

C. Value Iteration Scheme: J-Learning

For the 2820 data, value iteration was performed with the k-nearest neighborhood approximator, which averages four neighboring points ($k = 4$). Using a discount factor of 0.98, initial cost-to-go values were calculated by (21) with the following definition of one-stage cost:

$$\phi(x(k), u(k)) = 50(0.4450 - y(k+1))^2 + (u(k) - u(k-1))^2 \tag{32}$$

Hence, with the ADP based method, we are attempting to derive a much less detuned controller that still maintains closed-loop stability. The convergence criterion used for the value iteration was

$$e_{rel} = \left\| \frac{J^i(x_k) - J^{i-1}(x_k)}{J^{i-1}(x_k)} \right\|_\infty < 0.01 \tag{33}$$

where $k = 1, \cdots, 2820$ and i is the iteration index.

The offline value iteration converged after 31 steps and the e_{rel} was decreased monotonically. The parameters of the penalty function were set as $\sigma = 0.1587$ (1% of normalized distance range), $\rho = 6.6 \times 10^{-9}$, $A = 0.0696$, and $J_{max} = 200$. Fig. 2 shows the improved performance and Fig. 3 illustrates that the suggested strategy uses the model in the vicinity of the data and avoids undue extrapolations.

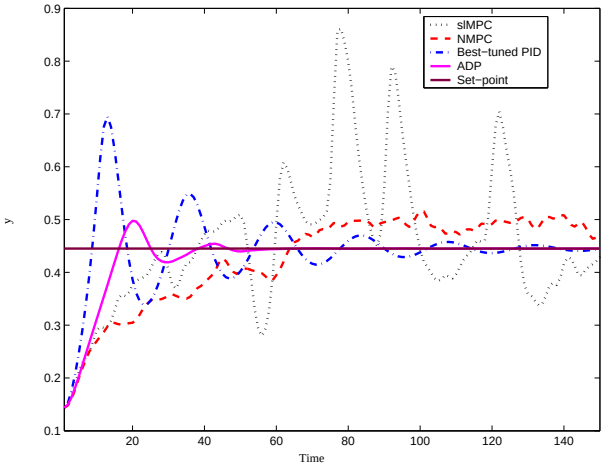

Fig. 2. Comparison of the closed-loop performances.

We compare the online regulation performance by calculating the infinite horizon cost, defined as

$$\sum_{k=0}^{\infty} \phi(x(k), u(k)) \tag{34}$$

Table I shows that the ADP-based scheme improved the starting control policies (PI controllers), while avoiding the extrapolation problem seen in the nonlinear empirical model based MPCs.

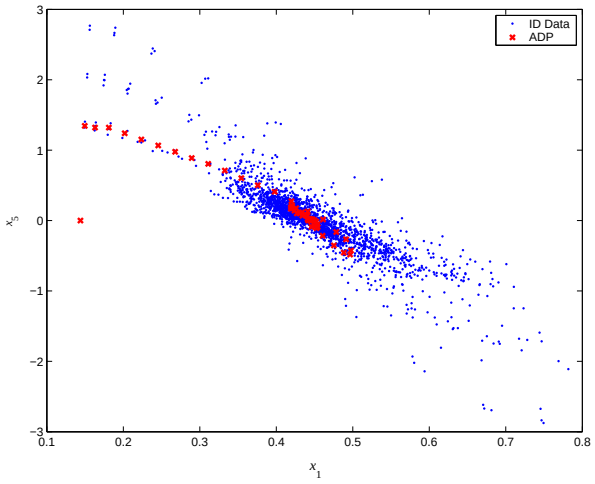

Fig. 3. State trajectory of ADP-based approach: plot of x_1 vs. x_5.

TABLE I
COMPARISON OF ONLINE PERFORMANCES: INFINITE HORIZON COST

Detuned		PI			ADP
slMPC	NMPC	Kc = 4.5	6.75	9.0	
165	42.0	60.9	43.8	58.0	27.2

VI. CONCLUSIONS AND FUTURE WORKS

In this paper, we presented an ADP-based data-driven control algorithm, which iteratively learns a cost-to-go function from the data collected through closed-loop identification experiments. The learned cost-to-go function was adjusted by a penalty term based on the distribution of local data used for averaging. The penalty term was intended to discourage the optimizer from finding a solution in the regions of the state space with insufficient amounts of data.

If we map all relevant state and action pairs into cost-to-go values, no model would be necessary at all. This would let us avoid the problem of choosing the model's functional structure. Such a model-less scheme is called Q-learning in the AI field [6], [7]. However, for continuous state space systems such as in most process control problems, the conventional Q-learning framework designed for finite Markov Decision Problems is not appropriate since it is unlikely that exactly same states will be visited multiple times.

Motivated by this, we also developed a policy-iteration type Q-learning method, in which the policy evaluation step requires the real implementation of the improved policy. The policy implemented based on the current estimate of Q^i is in the form of

$$\mu^i(x(k)) = \arg\min_u \tilde{Q}^i(x(k), u) \tag{35}$$

After collecting closed-loop data with the above policy in place, the policy evaluation for μ^i can be done iteratively using the equation

$$Q^{i+1,\ell+1}(x(k), u(k)) = Q^{i+1,\ell}(x(k), u(k)) + \gamma \left[\phi(k) + \alpha \tilde{Q}^{i+1,\ell}(x(k+1), \mu^i(x(k+1))) - Q^{i+1,\ell}(x(k), u(k))\right] \tag{36}$$

where i is the iteration index for policy improvement, ℓ is the index for the policy evaluation and γ is a learning rate parameter. The new Q function yields yet another policy and this can continue on until no more improvement is seen. This leads to an evolutionary improvement scheme, which is not elaborated here due to the space limitation.

VII. ACKNOWLEDGMENTS

The authors gratefully acknowledge the financial support of National Science Foundation (Grant CTS - 0301993).

REFERENCES

[1] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Delyon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky, "Nonlinear black-box modeling in system identification: A unified overview," *Automatica*, vol. 31, pp. 1691–1724, 1995.

[2] H. T. Su and T. J. McAvoy, "Integration of multilayer perceptron networks and linear dynamic models: A hammerstein modelling approach," *Ind. Eng. Chem. Res.*, vol. 32, pp. 1927–1936, 1993.

[3] Y. Chikkula and J. H. Lee, "Robust adaptive predictive control of nonlinear processes using nonlinear moving averager system models," *Ind. Eng. Chem. Res.*, vol. 39, pp. 2010–2023, 2000.

[4] J. H. Lee and Z. Yu, "Worst-case formulation of model predictive control for systems with bounded parameters," *Automatica*, vol. 29, pp. 911–928, 1993.

[5] J. M. Lee and J. H. Lee, "Simulation-based learning of cost-to-go for control of nonlinear processes," *Korean J. Chem. Eng.*, vol. 21, no. 2, 2004, in press.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[7] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[8] N. S. Kaisare, J. M. Lee, and J. H. Lee, "Simulation-based strategy for nonlinear optimal control: Application to a microbial cell reactor," *International Journal of Robust and Nonlinear Control*, vol. 13, pp. 347–363, 2002.

[9] J. M. Lee and J. H. Lee, "Simulation-based dual mode controller for nonlinear processes," in *Preprints of 7th International Symposium on Advanced Control of Chemical Processes*, vol. 1, Hongkong, China, jan 2004, pp. 225–230.

[10] ——, "Neuro-dynamic programming approach to dual control problem," presented at the AIChE Annual Meeting, Reno, NV, 2001.

[11] J. H. Lee and N. L. Ricker, "Extended Kalman filter based nonlinear model predictive control," *Ind. Eng. Chem. Res.*, vol. 33, pp. 1530–1541, 1994.

[12] N. S. Kaisare, J. M. Lee, and J. H. Lee, "Comparison of policy iteration, value iteration and temporal difference learning," presented at the AIChE Annual Meeting, Indianapolis, IN, 2002.

[13] J. M. Lee, N. S. Kaisare, and J. H. Lee, "Simulation-based dynamic programming strategy for improvement of control policies," presented at the AIChE Annual Meeting, San Francisco, CA, 2003.

[14] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, pp. 1065–1076, 1962.

[15] M. Kortmann and H. Unbehauen, "Structure detection in the identification of nonlinear systems," *Autom. Prod. Infor. Ind.*, vol. 22, pp. 5–25, 1988.

[16] E. Hernández and Y. Arkun, "Control of nonlinear systems using polynomial ARMA models," *AIChE Journal*, vol. 3, pp. 446–460, 1993.

[17] H. Demuth and M. Beale, *Neural Network Toolbox User's Guide (MATLAB)*. Natick, MA: The MathWorks, Inc., 2002.