Korean J. Chem. Eng., **21**(2), 338-344 (2004)

Simulation-Based Learning of Cost-To-Go for Control of Nonlinear Processes

Jong Min Lee and Jay H. Lee[†]

311 Ferst Dr. NW, School of Chemical and Biomolecular Engineering,
Georgia Institute of Technology, Atlanta, GA 30332-0100, USA

(*Received 2 October 2003 • accepted 16 December 2003*)

Abstract−In this paper, we present a simulation-based dynamic programming method that learns the 'cost-to-go' function in an iterative manner. The method is intended to combat two important drawbacks of the conventional Model Predictive Control (MPC) formulation, which are the potentially exorbitant online computational requirement and the inability to consider the future interplay between uncertainty and estimation in the optimal control calculation. We use a nonlinear Van de Vusse reactor to investigate the efficacy of the proposed approach and identify further research issues.

Key words: Nonlinear Model Predictive Control, Dynamic Programming, Stochastic Optimal Control, Reinforcement Learning, Neuro-dynamic Programming, Function Approximation

INTRODUCTION

Model predictive control (MPC) has been the most popular advanced control technique for the process industry owing to its ability to handle a large multi-variable system with constraints. In a typical MPC formulation, a dynamic model is used at each sample time to build a prediction of future output behavior, which is subsequently used in the optimization routine to find a sequence of input moves to minimize output deviation from a set-point trajectory. Since its arrival in the early 80s, the two decades of intensive research has brought sound theories and fundamental understandings into its behavior, and spawned a myriad of design methods that guarantee stability and certain optimality properties [Morari and Lee, 1999; Mayne et al., 2000].

Despite this, two important issues remain for MPC, which are both theoretical and practical in nature. The first is the potentially exorbitant online computation needed to calculate the optimal control moves at each sample time. This issue is particularly relevant when the underlying system model is large in dimension, demands the use of long prediction/control horizons, and is nonlinear or hybrid in nature [Morari and Lee, 1999; Mayne et al., 2000; Bemporad and Morari, 1999]. The resulting optimization problem to be solved online is a large-scale nonlinear program or mixed integer program, which still presents significant computational challenges despite all the advances made in computational hardware and numerical methods. The second is the MPC's inability to take into account the future interplay between uncertainty and estimation in the optimal control calculation [Lee and Cooley, 1997; Chikkula and Lee, 2000]. The problem the conventional MPC solves at each time is a *deterministic* open-loop optimal control problem, which thus ignores the uncertainty and feedback at future time points. MPC tries to address the issue of uncertainty by updating the prediction equation based on fresh measurements and re-solving the optimization on a moving window at each sample time, which is referred to as receding horizon control implementation. For problems that involve uncertainties and feedback, however, this approach is inherently suboptimal.

Both issues can be addressed in theory by the approach of (stochastic) dynamic programming (DP) [Bellman, 1957]. The 'cost-to-go' function in DP can be used to reduce a multi-stage problem into an equivalent single stage problem, thus reducing the online computational load dramatically. Also, in stochastic DP, the 'cost-to-go' function is calculated with respect to the information vector (sometimes called 'hyper-state') to reflect the effect of uncertainty on the future costs under the optimal *feedback* control [Bertsekas, 2000; Åström and Helmersson, 1986; Åström and Wittenmark, 1989; Lee and Yu, 1997]. The proper accounting of the uncertainty endows the resulting control policy with several desirable properties like cautiousness and active reduction of uncertainty (i.e., active probing) according to its importance for future control performance. However, the DP approach is considered largely impractical because analytical solution is seldom possible and numerical solution via discretization suffers from what is referred to as 'curse-of-dimensionality' [Bellman, 1957].

In this paper, to tackle the aforementioned deficiencies of the MPC formulation, we present an approach based on simulation and dynamic programming, which is inspired by Reinforcement Learning (RL) [Sutton and Barto, 1998] and Neuro-Dynamic Programming (NDP) [Bertsekas and Tsitsiklis, 1996] developed in the field of Artificial Intelligence. RL and NDP approaches were both intended to alleviate the curse-of-dimensionality. They both use simulation or real data to build an approximation of the 'cost-to-go' function, typically by fitting an artificial neural network. Compared to the classical numerical solution approach for DP, which performs exhaustive sampling of the entire state (or hyper-state) space in solving the stage-wise optimization, these approaches sample only a small fraction of the state space and thus require dramatically less computation. The sample points of the state space are decided by performing simulations of the system under various known (sub-optimal) policies and all conditions to be encountered potentially. This "judicious" sampling makes them applicable to practical-size

[†]To whom correspondence should be addressed.
E-mail: jay.lee@chbe.gatech.edu
[‡]This paper is dedicated to Professor Hyun-Ku Rhee on the occasion of his retirement from Seoul National University.

problems. The approach has received significant attention for its successes in several applications, including a program that plays Backgammon at the world champion level, an elevator dispatcher program that is more efficient than several heuristic-based algorithms used in practice, and job shop scheduling problems [Tesauro, 1992; Crites and Barto, 1996; Zhang and Dietterich, 1995]. We had previously explored the approach for chemical process control problems including nonlinear control problems [Kaisare et al., 2002; Lee and Lee, 2003] and a stochastic optimal control problem [Lee and Lee, 2001].

The rest of the paper is organized as follows. In section 2, we compare the conventional MPC formulation with the DP formulation to bring out the key advantages of the latter. In section 3, we discuss the simulation-based DP formulation and an algorithm for iterative improvement of the 'cost-to-go' function. We also show how the resulting 'cost-to-go' function can be used online for real-time control calculations. In section 4, a case study involving a nonlinear Van de Vusse reactor is presented to illustrate how the new approach can be used to lower the online computational burden and also improve upon the performance compared to the suboptimal MPC policy we started out with. In section 5, we offer some conclusions and suggestions for successful implementation of the approach.

PRELIMINARIES

1. Conventional MPC: Open-Loop Optimal Feedback Control

Conventional MPC solves the following open-loop optimal control problem at each sample time after a feedback update of the state,

$$\min_{u(0),\ldots,u(p-1)} \left\{ \left[\sum_{i=0}^{p-1} \phi(\mathrm{x}(\mathrm{i}),\mathrm{u}(\mathrm{i})) \right] + \phi_t(\mathrm{x}(\mathrm{p})) \right\} \tag{1}$$

with a dynamic model $\dot{\mathrm{x}}=\mathrm{f}(\mathrm{x}, \mathrm{u})$ for given initial state x(0), which is the current state (estimate) and a piecewise constant input $\mathrm{u}(\tau)=\mathrm{u}(\mathrm{i})$ for $\mathrm{i}\cdot\mathrm{t}_s\le\tau\le(\mathrm{i}+1)\cdot\mathrm{t}_s$. ϕ is the single-stage cost, ϕ_t is the cost of the terminal state and t_s is a sample time. Some additional constraints such as input and output limits can be added to the above. The above open-loop optimal control problem (OLOCP) implicitly defines a relationship between initial state x(0) and the optimal initial control adjustment u(0), which can be denoted as $\mathrm{u}(0)=\mu(\mathrm{x}(0))$. μ is a policy that maps the state to action. Thus $\mathrm{u}(\mathrm{t})=\mu(\mathrm{x}(\mathrm{t}))$ represents the feedback law for MPC. Since the feedback law is only implicitly defined through the OLOCP, its implementation requires solving the optimization problem online with $\mathrm{x}(0)=\mathrm{x}(\mathrm{t})$ (or $=\hat{\mathrm{x}}(\mathrm{t})$, which is an estimate of x(t)) at each sample time. Depending on the size and the nature of the underlying model, the computational burden can be unwieldy. This is particularly true when the underlying model is nonlinear leading to a nonlinear program [Henson, 1998], or hybrid leading to a mixed integer program [Bemporad and Morari, 1999]. Moreover, it was shown to be advantageous to solve an infinite horizon problem, which worsens the complexity of the optimization problem.

In addition, for a process for which uncertainties are characterized, as a stochastic process for example, the estimation problem and the control problem are interlaced as the quality of estimation is affected by control and vice versa. The formulation based on the OLOCP is fundamentally incapable of addressing this interplay. For example, since sampled data values for the input trajectories are directly optimized in the formulation, the ameliorating effects of exploratory input actions on future estimation through the generation of additional signals are not considered. This is true even when the uncertainty information is incorporated explicitly into the OLOCP - to minimize the average cost or the worst-case cost. In summary, the MPC's approach of solving the OLOCP repeatedly with feedback updates leads to only suboptimal control in the case of an uncertain system [Lee and Yu, 1997; Lee and Cooley, 1997].

2. Dynamic Programming Approach for Optimal Control

Dynamic Programming (DP) is a general mathematical framework for solving multi-stage optimization problems with or without uncertainties [Bellman, 1957]. The approach can provide a *closed-loop* optimal solution and involves stage-wise calculation of the so-called 'cost-to-go' values for *all* states. The 'cost-to-go' of a state is the sum of all costs that one can expect to incur under a given policy (usually the optimal policy) starting from the state, and hence expresses the quality of a state in terms of *future* performance. Hence, given the optimal cost-to-go function, one can easily calculate the optimal action simply by minimizing the sum of the cost of the current state and the cost-to-go of the next state.

2-1. Deterministic Systems

For a deterministic system with a fixed starting state and a deterministic policy, the entire future sequence of states and actions is determined. The 'cost-to-go' function under a policy μ is the sum of stage-wise costs up to the end of the horizon.

$$\mathrm{J}_i^{\mu}(\mathrm{x}(\mathrm{p}-\mathrm{i})) = \sum_{j=p-i}^{p-1} \phi(\mathrm{x}(\mathrm{j}),\mu(\mathrm{x}(\mathrm{j}))) + \phi_t(\mathrm{x}(\mathrm{p})) \tag{2}$$

The optimal 'cost-to-go' function, $\mathrm{J}^*=\mathrm{J}^{\mu^*}$, is the cost-to-go function under an optimal policy and is unique. For a finite horizon problem, the optimal cost-to-go function should satisfy

$$\mathrm{J}_i^*(\mathrm{x}) = \min_u \{\phi(\mathrm{x},\mathrm{u}) + \mathrm{J}_{i-1}^*(\mathrm{F}_{t_s}(\mathrm{x},\mathrm{u}))\} \tag{3}$$

where $\mathrm{F}_{t_s}(\mathrm{x}, \mathrm{u})$ denotes the state resulting from integrating the differential equation for one sample time interval with initial state x and constant input u. To solve the above optimality equation, sequential calculation of J^* for all states is performed, usually in a backward manner starting from the terminal stage where i=0 and $\mathrm{J}_0^*=\phi_p(\mathrm{x})$.

With the optimal cost-to-go function for the p−1 stage ($\mathrm{J}_{p-1}^*(\mathrm{x})$) calculated offline, one can solve the following single stage problem, which is equivalent to *p*-stage problem defined earlier:

$$\min_u \{\phi(\mathrm{x},\mathrm{u}) + \mathrm{J}_{p-1}^*(\mathrm{F}_{t_s}(\mathrm{x},\mathrm{u}))\} \tag{4}$$

It may even be possible to solve the above optimization offline for all x to characterize the optimal feedback law completely.

For infinite horizon problems, by letting $\mathrm{i}\rightarrow\infty$, we obtain the following 'Bellman Equation,' which must be solved for all x to obtain the cost-to-go function:

$$\mathrm{J}_\infty^*(\mathrm{x}) = \min_u \{\phi(\mathrm{x},\mathrm{u}) + \mathrm{J}_\infty^*(\mathrm{F}_{t_s}(\mathrm{x},\mathrm{u}))\} \tag{5}$$

While DP can in principle help reduce the online computational burden for optimal control, in order for the approach to be practical, one must be able to solve the equation for the cost-to-go function (i.e., Bellman Equation). Analytical solution is not possible except for a few special cases (e.g. unconstrained linear quadratic control problem) and numerical solution through discretization of the state space is possible only for small-size problems [Bertsekas, 2000].

2-2. Stochastic Systems

Whereas the DP (closed-loop) and the MPC (open-loop) formulations result in the same solution in the case of a deterministic system, the two approaches lead to very different results in the case of a stochastic system. Because the conventional MPC formulation treats the future inputs as deterministic, it represents only a suboptimal feedback strategy. For optimal feedback control, one should solve the following *p*-stage problem.

$$\min_{u(i)=f_i(I_i)} \mathrm{E}\left\{\left[\sum_{i=0}^{p-1}\phi(\mathrm{x(i)},\mathrm{u(i)})\right]+\phi_t(\mathrm{x(p)})\right\} \tag{6}$$

where I_i is a vector summarizing the information available at the ith stage and f_i is an arbitrary function. It is assumed that there is an underlying equation for dynamic propagation of x and I. Note that the inputs are no longer optimized as deterministic variables as I is stochastic. The consideration of feedback control gives rise to a stochastic dynamic program, which must be solved in a similarly sequential manner.

For the infinite horizon problem, the following discounted cost index can be used to bound the cost [Bertsekas, 2000]:

$$\mathrm{E}\left\{\sum_{i=0}^{\infty}\alpha^i\phi(\mathrm{x(i)},\mathrm{u(i)})\right\} \tag{7}$$

where α is a parameter between 0 and 1. Then, the exponentially weighted infinite horizon optimal cost-to-go function can be calculated through the equation

$$\mathrm{J}_\infty^*(\mathrm{I})=\min_u \mathrm{E}\{\phi(\mathrm{x},\mathrm{u})+\alpha \mathrm{J}_\infty^*(\mathrm{F}_{t_s}(\mathrm{I},\mathrm{u}))|\mathrm{I}\} \tag{8}$$

where I is the information vector. It typically consists of the parameters defining the conditional probability distribution of the state, e.g., the state estimate and the error covariance matrix for a Gaussian system. $\mathrm{F}_{t_s}(\mathrm{I}, \mathrm{u})$ is the stochastic equation that relates the information vector from one sample time to the next. It is assumed that the equation for recursive calculation of I is available.

Again, the approach is computationally infeasible, even for small-size problems, because of the large sample space (the information vector space, which is generally quite large) and the need to evaluate the expectation.

SIMULATION-BASED DYNAMIC PROGRAMMING APPROACH

The simulation-based approach attempts to overcome the 'curse-of-dimensionality' of DP by solving the DP only within limited regions of the state space. This is done by performing simulations of the system under known suboptimal policies subject to various possible disturbances/operation conditions. The premise is that although the state space may be huge, only a very small fraction of it would be relevant for optimal or near-optimal control in practice. The rationale for using closed-loop simulations is that several suboptimal policies combined together should span an envelope in the state space, within which satisfactory controls can be found. Of course, this is not always true and one should consider adaptive adjustment of the envelope through cautious exploration and additional simulations from the resulting DP-based policies. Hence, the proposed scheme is characterized by simulation and iteration between function approximation and improvement of the 'cost-to-go' approximation. Next, the offline iteration and online implementation procedures are described for both deterministic and stochastic systems.

1. Deterministic Systems

The following are the steps for constructing and improving the cost-to-go function offline:

(1) Perform simulations of the closed-loop system with some suboptimal control policy or policies (μ^0) under all representative operating conditions. The quality of suboptimal control policies matter: The closer they are to the optimal policy, the better. However, optimal or near optimal policies may be unknown or computationally too expensive to simulate. It is recommended that several policies effective in different regions of the state space and conditions be simulated.

(2) Using the simulation data, calculate the infinite (or finite) horizon cost-to-go (J^{μ^0}) for each state visited during the simulation.

(3) Construct a function approximator for the data to approximate the cost-to-go as a function of the state, denoted as $\tilde{\mathrm{J}}^{\mu^0}(\mathrm{x})$.

(4) To improve the cost-to-go, which is suboptimal because it is with respect to the simulated suboptimal policy, use a value or policy iteration. The algorithms described below are for infinite horizon problems. The formula for the finite horizon case is equally straightforward to derive based on Eq. (3).

A. Value Iteration

In value iteration, one attempts to solve the Bellman equation of (5) for each sampled state in an iterative manner. At each iteration step, we calculate J^{i+1} for the given sample points of x by solving

$$\mathrm{J}^{i+1}(\mathrm{x})=\min_u\{\phi(\mathrm{x},\mathrm{u})+\tilde{\mathrm{J}}^i(\mathrm{F}_{t_s}(\mathrm{x},\mathrm{u}))\} \tag{9}$$

where i denotes the ith iteration step. Once the cost-to-go values are updated for all the states, then we fit another cost-to-go function approximation to the x vs. $\mathrm{J}^{i+1}(\mathrm{x})$ data.

B. Policy Iteration

Policy iteration consists of two steps. The first step is *policy evaluation* where the cost-to-go function is computed and approximated for a given policy. The second step is *policy improvement* where a new greedy policy is computed.

The policy evaluation algorithm computes and approximates cost-to-go function for a given policy until convergence.

$$\tilde{\mathrm{J}}^{l+1}(\mathrm{x})=\phi(\mathrm{x},\mu^i(\mathrm{x}))+\tilde{\mathrm{J}}^l(\mathrm{F}_{t_s}(\mathrm{x},\mu^i(\mathrm{x}))) \tag{10}$$

Once the policy evaluation step is converged (say after L steps), the policy is improved by taking the greedy policy, which is given by

$$\mu^{i+1}(\mathrm{x})=\arg\min_u\{\phi(\mathrm{x},\mathrm{u})+\tilde{\mathrm{J}}^L(\mathrm{F}_{t_s}(\mathrm{x},\mathrm{u}))\} \tag{11}$$

where *l* and i are the iteration indices of policy evaluation and improvement steps, respectively. The policy iteration continues until $\mu^{i+1}(\mathrm{x})=\mu^i(\mathrm{x})$.

C. Comments

For finite Markov Decision Processes (MDPs), it was shown that value and policy iteration algorithms (with exact 'cost-to-go' calculation without function approximation error for the entire state space) converge to an optimal policy [Howard, 1960]. While policy iteration requires the policy evaluation between steps of policy improvement, this can usually be done in relatively few iteration steps because the cost-to-go function seldom changes much when the policy is improved only slightly. In addition, we can expect the policy iteration to require fewer policy improvement steps than the value iteration because the policy improvements are based on more accurate cost-to-go information. The results have been shown theoretically [Puterman, 1994]. However, since function approximations based on limited data in a continuous state space are used in our iterations, approximation errors at each step can be significant and the above results do not hold. In addition, even convergence cannot be guaranteed. Also, the question of which algorithm performs better for various practical scenarios is an open question.

The main premise behind working with only the sampled states from the closed-loop simulation, which generally covers only a very small fraction of the entire state space, is that not all parts of the state space are needed for good control, and critical regions needed for good control can be identified by performing closed-loop simulations with a judiciously chosen set of suboptimal policies. Hence, conditions for the simulation as well as the suboptimal policies must be chosen carefully. Points not sampled during the simulation are left at the mercy of interpolation and extrapolation. Compared to the classical numerical solution approach for DP, the above approach reduces the computational burden significantly for two reasons: First, even for very high-dimensional systems, the operating regions the closed-loop system occupies may represent a low-dimensional manifold. Second, for the infinite horizon cost-to-go calculation, the iteration of the Bellman equation is started with a very good estimate $\tilde{J}^{\mu^0}$, which is obtained through simulation with a suboptimal (but good) control policy.

Online implementation of the optimal control law is based on the converged cost-to-go function and involves solving at each sample time

$$\min_u \phi(x,u) + \tilde{J}(F_{t_s}(x,u)) \tag{12}$$

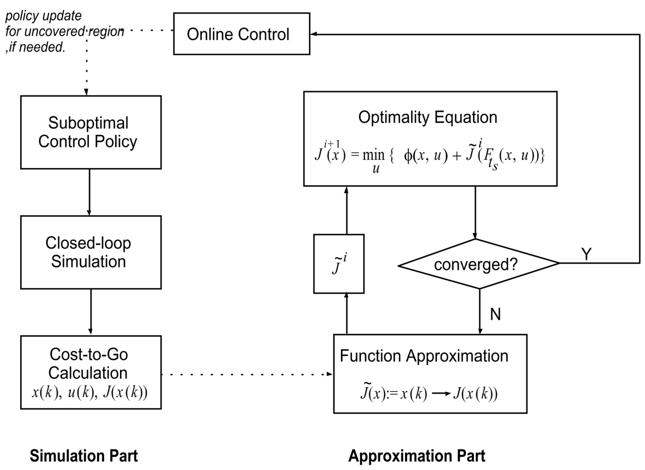
policy update for uncovered region ,if needed.
Online Control
Suboptimal Control Policy
Closed-loop Simulation
Cost-to-Go Calculation $x(k),\ u(k),\ J(x(k))$
Optimality Equation $J^{i+1}(x) = \min_u \{\phi(x,u) + \tilde{J}^i(F_{t_s}(x,u))\}$
$\tilde{J}^i$
converged?
Y
N
Function Approximation $\tilde{J}(x) := x(k) \rightarrow J(x(k))$
Simulation Part
Approximation Part

Fig. 1. A schematic diagram of value iteration algorithm with simulation and function approximation.

which is relatively simple to solve computationally. Of course, one could iterate the closed-loop simulation with the new control policy defined above in order to add more state samples and obtain more accurate cost-to-go data for them. This evolutionary scheme is outside the scope of this paper although the research is already underway. Fig. 1 shows the value iteration scheme described above.

2. Stochastic Systems

In the stochastic case, the procedure is further complicated by the need to evaluate the expectation operator. However, the general idea remains the same as the following procedure shows:

(1) Execute Monte Carlo simulations of the closed-loop system with some judiciously chosen suboptimal control law. For example, start with a control scheme that couples some state estimator like the extended Kalman Filter (EKF) and a deterministic control policy.

(2) From the simulation data, calculate the discounted cost-to-go for all the visited states and construct data for 'information vector' vs. cost-to-go.

(3) Using a function approximation method, fit the data to obtain an initial approximation for the cost-to-go function, $\tilde{J}^{\mu^0}(I)$.

(4) Perform value (or policy) iteration until convergence as follows.

A. With the current estimate $\tilde{J}^i(I)$, calculate J^{i+1} for the given sample points of I by solving

$$J^{i+1}(I) = \min_u E\{\phi(I,u) + \alpha\tilde{J}^i(F_{t_s}(I,u))\} \tag{13}$$

This step is more demanding than before owing to the presence of the expectation operator (which arises because of the fact that $F_{t_s}(I, u)$ is a stochastic equation).

B. Fit an improved cost-to-go approximation to the I vs. $J^{i+1}(I)$ data.

(5) One may also iterate the steps (1)-(4) with the updated suboptimal control policy for more improvement.

Online implementation involves solving the following one-step optimization problem:

$$\min_u E\{\phi(I,u) + \alpha\tilde{J}(F_{t_s}(I,u))\} \tag{14}$$

Note that, in calculating the expected cost-to-go from simulation data in the above iteration step, the expectation operator is not explicitly evaluated. Instead, it is thought that, by fitting an approximator to the data from various realizations of the stochastic system (i.e., Monte Carlo simulations), the fitted cost will represent the expected cost. This simulation-based approach also provides some additional flexibility such as the choice of disturbances. One does not have to limit oneself to some linear process driven by an i.i.d. Gaussian noise. One can work with non-normal distributions and (randomly occurring) deterministic disturbances mixed with stochastic noises.

Here we assumed a fixed estimator and it is not further optimized. This is not limiting as long as the model does not have any structural error. However, the information supplied to the estimator is influenced by the control decision and hence is optimized (for the particularly chosen estimator) directly for future control performance. Thus, one automatically gets some probing (i.e., active learning), designed for optimal control performance.

In the stochastic case, the exploration step (described in Step 5 of the aforementioned procedure) may be more important because the optimal control policy is known to behave very differently from the certainty equivalence control policy that ignores the uncertainty. Also, one may enhance the performance by performing closed-loop data subjected to some dithering. In this way, the benefit of active exploration can be incorporated into the cost-to-go approximation.

This line of thought also suggests a way to improve an already-implemented control policy in an evolutionary manner. One could collect the cost-to-go values from real operation and use it to improve the cost-to-go approximation and the corresponding control policy with the relatively simple framework described above.

APPLICATION TO VAN DE VUSSE REACTOR

We consider a Van de Vusse reaction [Van de Vusse, 1963] in isothermal CSTR described by

$$\frac{dx_1}{dt} = -k_1x_1 - k_3x_1^2 + (x_{1f} - x_1)u \tag{15}$$

$$\frac{dx_2}{dt} = k_1x_1 - k_2x_2 - x_2u \tag{16}$$

where $k_1=50\ h^{-1}$, $k_2=100\ h^{-1}$, $k_3=10\ l/mol\cdot h$ and $x_{1f}=10\ mol/l$. The state variables x_1 and x_2 represent the concentrations of the reactant and the intermediate, respectively. x_{1f} represents the concentration of the reactant in the feed, and u represents the dilution rate. The control objective is to regulate the output $y=x_2$, the concentration of the intermediate, at the set-point of 1.2 by manipulating the dilution rate with a sample time of 0.002 h. With the set-point, we have two steady state solutions: $\{x_1, x_2, u\}=\{5.5362, 1.2, 130.6758\}$ and $\{3.4960, 1.2, 45.6683\}$.

This reactor shows input multiplicities and process zero shifts from left-half-plane zero to right-half-plane zero at u=77.5 when the output variable is x_2 [Sistu and Bequette, 1995]. A sufficiently long horizon is needed due to the nonminimum phase behavior of this system [Meadows and Rawlings, 1997].

1. Deterministic Case

1-1. Suboptimal Nonlinear MPC

We consider the deterministic system and introduce step changes of various sizes in the parameters, k_1 and x_{1f} at t=0. Hence, the cost-to-go function we construct is a function of k_1 and x_{1f} as well as the two states.

As a starting suboptimal control policy, we use the nonlinear MPC algorithm based on successive linearization described in Lee and Ricker [1994]. The method linearizes the nonlinear model at each current state and input values to calculate a prediction equation linear in terms of the future manipulated input moves. The control is computed by solving quadratic programming (QP), of which the Hessian matrix and the gradient vector are updated at each sample time. Here we assumed that the full state variables are measured.

The one-stage cost $\phi(x, u)$ was chosen as:

$$\phi(x,u) = F_{t_s}^T(x,u)QF_{t_s}(x,u) + \Delta u^T R \Delta u \tag{17}$$

Note that $F_{t_s}^T(x(t), u(t))QF_{t_s}(x(t), u(t))=x^T(t+1)Qx(t+1)$. By formulating the single stage cost to include a penalty on the error of the state at next sample time, we ensure that, in the online control calculation of Eq. (12), the one-step-ahead error is counted exactly (rather than through the approximated cost-to-go function), thus making the formulation more robust to approximation errors. The weighting factors we used are Q=1000 and R=1 in Eq. (17).

Assuming higher dilution rate is undesirable, we place an upper limit on the control input ($70\ h^{-1}$) to prevent the input from drifting to the other side of the steady-state curve. The problem is the large inverse response, which can cause the MPC to drive the process away from the desired operating condition. To prevent this, we used a fairly large prediction horizon p=50.

1-2. Simulation-Based DP Approach

We first employed the nonlinear MPC scheme as a suboptimal control policy to generate closed-loop data for the initial cost-to-go calculation. Within ±2% of the nominal parameter values, we sampled 17 representative points from the disturbance space of k_1 and x_{1f} to cover the probable operating range. From the 17 simulation runs, 1360 data points for states vs. cost-to-go were obtained. The architecture for the cost-to-go function approximation we used in this work is a multilayer perceptron with 10 hidden nodes. The value iteration was used for offline cost-to-go improvement and it converged after two runs with the termination condition,

$$\frac{1}{N}\sum_{k=1}^{N}\left|\tilde{J}^{i+1}(x_k) - \tilde{J}^{i}(x_k)\right| < 0.3 \tag{18}$$

where N is the number of data points 1360. Table 1 compares the initial and converged cost-to-go for the 1360 data points.

In Table 2, we compare the online performance by calculating the infinite horizon costs from two different control policies, the NMPC control policy with p=50 and the simulation-based DP control policy of Eq. (12). The actual infinite horizon costs were computed by performing closed-loop simulations with 10 fresh initial states that are different from those in the training set. The CPU time shown is the averaged value over the 10 simulations performed with MATLAB® 6 on Pentium III (800 MHz). We can clearly see the superior performance of the control policy obtained from the simulation-based DP approach, both in terms of the cost and computational time.

2. Stochastic Case with Full State Feedback

Consider the case where integrated white noises are introduced in k_1 and x_{1f} of Eqs. (15) and (16).

$$k_1(t+1)=x_3(t+1)=x_3(t)+e_1(t) \tag{19}$$

Table 1. Converged cost-to-go value in value iteration (deterministic case)

Infinite horizon cost	Average	Min.	Max.
Initial policy (NMPC)	1.01	0	125.27
Converged policy	0.7810	0	107.56

Table 2. Online performance: closed-loop cost comparison of two control policies for 10 sample points (deterministic case)

Infinite horizon cost	Average	Min.	Max.	CPU time
NMPC	58.95	5.92	113.25	110.17s
Sim-based DP	24.39	2.57	49.69	52.62s

$$x_{1f}(t+1)=x_4(t+1)=x_4(t)+e_2(t) \tag{20}$$

where $e_1(t)$~$N(0, 0.2^2)$ and $e_2(t)$~$N(0, 0.04^2)$.

For reasonable sampling of the augmented state space, four representative realizations of the stochastic disturbances were selected: (1) monotonic increases x_3 and x_4, (2) monotonic increase in x_3 and monotonic decrease in x_4, (3) monotonic decrease in x_3 and monotonic increase in x_4 and (4) monotonic decreases in x_3 and x_4. It is important to note that we did not sample every possible state. The use of simulation to “judiciously” sample the space is the key idea. The total number of the state samples visited during the four stochastic simulations was 1200, and 800 data points were used for cost-to-go approximation with the discount factor of 0.9.

In the simulation, the previously used state-feedback suboptimal nonlinear MPC with p=50, gave an average discounted infinite horizon cost of 33.19 (averaged over all the states visited during the four simulations). We fitted to the ‘state vs. cost-to-go’ data from the four simulations using a multi-layer perceptron with 10 hidden nodes. This was followed by value iteration where the expectation was taken over 50 realizations. The termination condition of the iteration was

$$\frac{1}{N}\sum_{k=1}^{N}\left|\tilde{J}^{i+1}(x_k)-\tilde{J}^{i}(x_k)\right|<1.0 \tag{21}$$

After 21 iterations, the average discounted cost associated with the converged approximator was 12.46, a significant reduction from the starting value, which means more optimal control policy was leamt.

In order to compare control performances, we generated 10 fresh integrated noise disturbances sets (different from those in the training set). Table 3 shows the averaged result of the closed-loop simulation under the nonlinear MPC with p=50 and the simulation-based converged cost-to-go with the test data set. The cost in the table is calculated for 300 time steps without discount factor.

SUMMARY

In this paper, a simulation-based DP framework for nonlinear process control was proposed to combat two important deficiencies of current threads of open-loop optimal control framework (e.g. MPC). By solving DP for the important region of state space with a function approximation scheme, the new framework was shown to offer enhanced computation time for online optimization and more improved control policy from a starting one. Simulation results from a nonlinear Van de Vusse reactor indicate that the suggested approach provides a promising framework to generalize MPC to handle nonlinear and/or hybrid system models as well as stochastic system models in a computationally amenable way. Our experience shows that cautious utilization of simulation data and design of approximators are requisite for the success of the proposed scheme.

Table 3. Comparison of closed-loop performance under two control policies with 10 fresh test data sets (stochastic case with full state feedback)

Averaged performance	NMPC	Sim-based DP
CPU time	326.4 s	120.4 s
Cost for 300 horizons	1082	888

ACKNOWLEDGMENT

JHL gratefully acknowledges the financial support from the National Science Foundation (CTS-# 0301993).

NOMENCLATURE

e	: sequence of white noise
f	: differential equation describing system dynamics
i	: discrete time index, number of stages, iteration index
I	: information vector
F_{t_s}	: state transition equation from one sample time to next
j	: discrete time index
J	: cost-to-go
J^*, J^{μ^*}	: optimal cost-to-go
$\tilde{J}$	: approximated cost-to-go function
k	: index number for data points
k_1	: rate constant [h^{-1}]
k_2	: rate constant [h^{-1}]
k_3	: rate constant [$l/(mol\cdot h)$]
l	: iteration index of policy evaluation step
N	: number of data points for offline iteration, normal distribution
p	: prediction horizon
Q	: output weighting matrix for objective function
R	: input weighting matrix for objective function
t	: time (continuous, discrete)
t_s	: sample time
u	: control action, dilution rate [h^{-1}]
x	: state vector
$\hat{x}$	: estimated state vector
α	: discount factor
μ	: control policy
ϕ	: single-stage cost
ϕ_t	: terminal cost
τ	: time

REFERENCES

Åström, K. J. and Helmersson, A., “Dual Control of an Integrator with Unknown Gain,” *Comp. and Maths. with Appls.*, **12A**, 653 (1986).

Åström, K. J. and Wittenmark, B., “Adaptive Control,” Addison-Wesley (1989).

Bellman, R. E., “Dynamic Programming,” Princeton University Press, New Jersey (1957).

Bemporad, A. and Morari, M., “Control of Systems Integrating Logic, Dynamics and Constraints,” *Automatica*, **35**, 407 (1999).

Bertsekas, D. P., “Dynamic Programming and Optimal Control,” 2nd ed., Athena Scientific, Belmont, MA (2000).

Bertsekas, D. P. and Tsitsiklis, J. N., “Neuro-Dynamic Programming,” Athena Scientific, Belmont, MA (1996).

Chikkula, Y. and Lee, J. H., “Robust Adaptive Predictive Control of Nonlinear Processes Using Nonlinear Moving Average System Models,” *Ind. Eng. Chem. Res.*, **39**, 2010 (2000).

Crites, R. H. and Barto, A. G., “Improving Elevator Performance Using Reinforcement Learning,” Advances in Neural Information Processing Systems 8, Touretzky, D. S., Mozer, M. C. and Haselmo,

M. E., eds., MIT Press, Cambridge, MA, 1017 (1996).

Henson, M. A., "Nonlinear Model Predictive Control: Current Status and Future Directions," *Computers and Chemical Engineering*, **23**, 187 (1998).

Howard, R. A., "Dynamic Programming and Markov Processes," MIT Press, Cambridge, MA (1960).

Kaisare, N. S., Lee, J. M. and Lee, J. H., "Simulation Based Strategy for Nonlinear Optimal Control: Application to a Microbial Cell Reactor," *International Journal of Robust and Nonlinear Control*, **13**, 347 (2002).

Lee, J. H. and Cooley, B., "Recent Advances in Model Predictive Control," *Chemical Process Control-V*, 201 (1997).

Lee, J. H. and Ricker, N. L., "Extended Kalman Filter Based Nonlinear Model Predictive Control," *Ind. Eng. Chem. Res.*, **33**, 1530 (1994).

Lee, J. H. and Yu, Z., "Worst-Case Formulations of Model Predictive Control for Systems with Bounded Parameters," *Automatica*, **33**, 1530 (1997).

Lee, J. M. and Lee, J. H., "Simulation-Based Dual Mode Controller for Nonlinear Processes," Proceedings of IFAC ADCHEM 2003, accepted (2004).

Lee, J. M. and Lee, J. H., "Neuro-Dynamic Programming Approach to Dual Control Problems," AIChE Annual Meeting, Reno, NV (2001).

Marback, P. and Tsitsiklis, J. N., "Simulation-Based Optimization of Markov Reward Processes," *IEEE Transactions on Automatic Control*, **46**, 191 (2001).

Mayne, D. Q., Rawlings, J. B., Rao, C. V. and Scokaert, P. O. M., "Constrained Model Predictive Control: Stability and Optimality," *Automatica*, **36**, 789 (2000).

Meadows, E. S. and Rawlings, J. B., "Model Predictive Control," Nonlinear Process Control, Henson, M. A. and Seborg, D. E., eds., Prentice Hall, New Jersey, 233 (1997).

Morari, M. and Lee, J. H., "Model Predictive Control: Past, Present and Future," *Computers and Chemical Engineering*, **23**, 667 (1999).

Puterman, M. L., "Markov Decision Processes," Wiley, New York, NY (1994).

Sistu, P. B. and Bequette, B. W., "Model Predictive Control of Processes with Input Multiplicities," *Chemical Engineering Science*, **50**, 921 (1995).

Sutton, R. S. and Barto, A. G., "Reinforcement Learning: An Introduction," MIT Press, Cambridge, MA (1998).

Tesauro, G. J., "Practical Issues in Temporal Difference Learning," *Machine Learning*, **8**, 257 (1992).

Van de Vusse, J. G., "Plug-Flow Type Reactor versus Tank Reactor," *Chemical Engineering Science*, **19**, 964 (1964).

Zhang, W. and Dietterich, T. G., "A Reinforcement Learning Approach to Job Shop Scheduling," Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, 1114 (1995).