Factorized Approach to Nonlinear MPC Using a Radial Basis Function Model

Sharad Bhartiya and James R. Whiteley
School of Chemical Engineering, Oklahoma State University, Stillwater, OK 74078

A new computationally efficient approach for nonlinear model predictive control (NMPC) presented here uses the factorability of radial basis function (RBF) process models in a traditional model predictive control (MPC) framework. The key to the approach is to formulate the RBF process model that can make nonlinear predictions across a p-step horizon without using future unknown process measurements. The RBF model avoids error propagation from use of model predictions as input in a recursive or iterative manner. The resulting NMPC formulation using the RBF model provides analytic expressions for the gradient and Hessian of the controller's objective function in terms of RBF network parameters. Solution of the NMPC optimization problem is simplified significantly by factorization of the RBF model output into terms containing only known and unknown parts of the process.

Introduction

Many chemical processes exhibit nonlinear behavior. Application of commercial linear model predictive control (MPC) technology is only partially successful in such cases. As process conditions deviate from the nominal operating point, model mismatch increases with a corresponding degradation in control performance. The problem is particularly severe in the process industries where the areas of a plant with the greatest economic incentives to apply MPC (such as a reactor system) typically exhibit strong nonlinearities. Plant operators frequently disable an MPC system when model mismatch compromises overall control performance. Recommissioning cannot be performed until the MPC models are updated or the operating conditions return to original design point.

The notion of using a nonlinear model to control a significantly nonlinear process within the model predictive control paradigm has led to an active interest in the development and application of nonlinear MPC (NMPC). NMPC adheres to the general MPC philosophy, that is, use of an explicit model to predict the process behavior over a future horizon, and implementation of control action that steers the process toward predetermined objectives in an optimal sense. NMPC uses a nonlinear model to provide a better approximation of the underlying nonlinear system. However, use of a nonlinear model presents additional challenges relative to linear MPC: (1) the complexity of nonlinear systems makes systematic development of nonlinear system identification techniques difficult (Pearson and Ogunnaike, 1997), and (2) nonlinear MPC requires solution of a nonlinear program at each sampling instant, making implementation more involved (Henson, 1998).

Use of artificial neural networks as nonlinear dynamic models has been studied in recent years. When applied for predictive control, most utilize a feedforward network architecture, while a few use the recurrent type. Hussain (1999) provides a summary of a number of applications reported in the literature.

In this article, we propose a radial basis function (RBF) network-based NMPC approach. The proposed RBF model is capable of providing noniterative sequential predictions over a prediction horizon of length p. The factorability of Gaussian functions, employed by the RBF network nodes is leveraged to formulate a compact representation of the model predictions over the prediction horizon. The traditional MPC controller objective function and the associated gradient and Hessian are then directly parameterized in terms of the network parameters. The resulting NMPC system is computationally efficient and provides the enhanced control expected

from use of a nonlinear model. Simulations examples are provided to demonstrate identification and control with the proposed technique.

Nonlinear System Identification Using Neural Networks

A feedforward network can be regarded as a nonlinear autoregressive model with external inputs (NARX),

$$\hat{y}_k = F(y_{k-1},\ldots,y_{k-N_y},u_{k-1},\ldots,u_{k-N_u}), \tag{1}$$

where a time delay of unity is assumed between the model output, $\hat{y}_k$, and the previous process inputs, u_{k-1}; N_y and N_u are integers defined by the order of the model; scalars $u_{k-1},\ldots,u_{k-N_u}$ represent the sequence of inputs used by the model; and function F depends on the network architecture and the type of activation function employed by the nodes.

A feedforward network is typically trained to minimize the 1-step-ahead prediction error:

$$E_{FFN} = \sum_{k=1}^{N} |\hat{y}_{k/k-1} - y_k|^2. \tag{2}$$

The subscript of $\hat{y}_{k/k-1}$ emphasizes the fact that the model prediction, $\hat{y}_k$, at sample k is based on measurements up to and including $k-1$. However, one of the primary purposes of MPC is to deal with complex dynamics over an extended horizon. Thus, an MPC model must predict the process dynamics over a prediction horizon, p, usually greater than one. Equation 1 cannot be directly employed to provide the desired long-term predictions, since future measurements needed in the computation are not available. However, a feedforward network can be cascaded to itself so that the model outputs are used as inputs for future predictions. Thus, the p predictions can be obtained as follows,

$$\begin{aligned}
\hat{y}_{k+1/k} &= F(y_k, \ldots, y_{k+1-N_y}, u_k, \ldots, u_{k+1-N_u})\\
\hat{y}_{k+2/k} &= F(\hat{y}_{k+1/k}, y_k, \ldots, y_{k+2-N_y}, u_{k+1}, \ldots, u_{k+2-N_u})\\
&\vdots\\
\hat{y}_{k+p/k} &= F(\hat{y}_{k+p-1/k}, \ldots, \hat{y}_{k+p-N_y/k}, u_{k+p-1}, \ldots, u_{k+p-N_u}).
\end{aligned} \tag{3}$$

In writing the preceding predictions, it has been assumed that $p > N_y$. We will refer to Eq. 3 as "cascaded 1-step" predictions.

Su and McAvoy (1997) tested the long-range predictive capability of a cascaded 1-step feedforward network on a biological wastewater-treatment system. The results showed that a feedforward neural network makes poor long-term predictions when compared to a recurrent neural network. The poor performance was attributed to the fact that the feedforward network training based on Eq. 2 does not take multistep prediction into account. Thus, accumulation of prediction errors leads to deterioration of model performance as the prediction horizon increases.

In contrast, a recurrent neural network is trained based on minimization of the following criterion:

$$E_{RNN} = \sum_{k=1}^{p} |\hat{y}_{k/0} - y_k|^2. \tag{4}$$

The training criterion simultaneously minimizes the prediction errors for 1-step, 2-step, and so forth up to p-steps in the future. Su and McAvoy reported good results using the recurrent network on the wastewater treatment plant. While appealing for use in an MPC system, recurrent networks are extremely difficult to train (Narendra and Parthasarathy, 1990). Until this problem is overcome, recurrent networks cannot be considered for general-purpose use in an MPC system.

In the remainder of this section, we discuss an alternative approach where predictions up to p-steps in the future can be made without requiring future (and yet unknown) process outputs. Thus, no cascading is necessary to provide the future p predictions, and problems with accumulation and propagation of modeling errors are avoided. The proposed approach retains the simplicity of 1-step-ahead training.

The cascaded 1-step model in Eq. 3 uses model predictions between $k+1$ and $k+j$ to predict future process outputs $k+j+i$. The dummy indices, j and i, assume values in the range, $[2,p-1]$ and $[1,p-j]$, respectively. However, we want to avoid dependency of model predictions later in the control horizon, p, on previous model predictions. In a real-time control setting, measurements are available only up to the current instant, k. Thus, we require that measurements input to our process model be limited to instant k or earlier. This can be accomplished by starting with the input-output model of Eq. 1 and applying successive iterations of this map until the measurements needed in the model input refer to available measurements.

To illustrate this idea, consider a model with a prediction horizon, $p=3$, output order, $N_y=2$, and input order, $N_u=2$. Then Eq. 1 can be rewritten as

$$\hat{y}_k = F(y_{k-1},y_{k-2},u_{k-1},u_{k-2}). \tag{5}$$

Applying successive iterations of function F yields the following expressions,

$$\begin{aligned}
\hat{y}_k &= F(F(y_{k-2},y_{k-3},u_{k-2},u_{k-3}),\\
&\qquad F(y_{k-3},y_{k-4},u_{k-3},u_{k-4}),u_{k-1},u_{k-2})\\
\hat{y}_k &= F(F(F(y_{k-3},y_{k-4},u_{k-3},u_{k-4}),y_{k-3},u_{k-2},u_{k-3}),\\
&\qquad F(y_{k-3},y_{k-4},u_{k-3},u_{k-4}),u_{k-1},u_{k-2}).
\end{aligned} \tag{6}$$

Equation 6 can be written as

$$\hat{y}_{k/k-3} = \bar{F}(y_{k-3},y_{k-4},u_{k-1},u_{k-2},u_{k-3},u_{k-4}). \tag{7}$$

The argument of the resulting composite function, $\bar{F}$, contains delayed process outputs and process inputs ranging from $k-p$ to $k-p+1-N_y$ (that is, $k-3$ to $k-4$) and $k-1$ to $k-p+1-N_u$ (that is, $k-1$ to $k-4$), respectively. Note that

the model defined by Eq. 7 can be used to make $p = 3$ future predictions without needing future plant outputs,

$$\hat{y}_{k+1/k-2} = \bar{F}(y_{k-2}, y_{k-3}, u_k, u_{k-1}, u_{k-2}, u_{k-3})$$
$$\hat{y}_{k+2/k-1} = \bar{F}(y_{k-1}, y_{k-2}, u_{k+1}, u_k, u_{k-1}, u_{k-2})$$
$$\hat{y}_{k+3/k} = \bar{F}(y_k, y_{k-1}, u_{k+2}, u_{k+1}, u_k, u_{k-1}). \quad (8)$$

Thus, we have eliminated the need to use predicted model outputs to obtain future predictions. During prediction of $\hat{y}_{k+j}$ in Eq. 8, factors that affect the process between the time $k+j-p$ and $k+j$ are accounted for by process inputs calculated for the interval $[k+j-1, k+j-p-N_u+1]$. Thus, the model in Eq. 7 maintains the causal relationship between the inputs and output. For the general case of a prediction horizon of p samples, and input and output orders of N_u and N_y, respectively, Eq. 7 takes the form:

$$\hat{y}_{k/k-p} = \bar{F}(y_{k-p}, \ldots, y_{k-p+1-N_y}, u_{k-1}, \ldots, u_{k-p+1-N_u}). \quad (9)$$

We modify the preceding model to make it better suited for control by replacing the past p process inputs, $u_{k-1}, \ldots, u_{k-p}$ by the corresponding control moves, $\Delta u_{k-1}, \ldots, \Delta u_{k-p}$, where,

$$\Delta u_k = u_k - u_{k-1}. \quad (10)$$

Revisiting the model in Eq. 7, the inputs are then modified as follows,

$$\hat{y}_{k/k-3} = \bar{F}(y_{k-3}, y_{k-4}, \Delta u_{k-1} + u_{k-2}\Delta u_{k-2} + \Delta u_{k-3} + u_{k-4}, \Delta u_{k-3} + u_{k-4}, u_{k-4})$$
$$\hat{y}_{k/k-3} = \bar{G}(y_{k-3}, y_{k-4}, \Delta u_{k-1}, \Delta u_{k-2}, \Delta u_{k-3}, u_{k-4}). \quad (11)$$

Thus for the general model in Eq. 9,

$$\hat{y}_{k/k-p} = \bar{G}(y_{k-p}, \ldots, y_{k-p+1-N_y}, u_{k-p-1}, \ldots, u_{k-p+1-N_u}, \Delta u_{k-p}, \ldots, \Delta u_{k-1}). \quad (12)$$

Replacement of recent p control inputs by the corresponding control moves simplifies expressions presented later in the article. From a process response approximation point of view, the delayed input/output measurements, namely, $y_{k-p}, \ldots, y_{k-p+1-N_y}$ and $u_{k-p-1}, \ldots, u_{k-p-N_u}$, respectively, provide a reference to the state of the system p samples in the past. Causality is provided by the most recent p input moves $\Delta u_{k-p}, \ldots, \Delta u_{k-1}$. To ensure that delayed process inputs appear in the model, in addition to input moves and process outputs, $N_u \geq 2$.

Since p future predictions can be made without use of model outputs, we will refer to Eq. 12 as the "p-step control model." The p future predictions with the p-step control model can be expressed as

$$\hat{y}_{k+1/k-p+1} = \bar{G}(y_{k+1-p}, \ldots, y_{k+2-p-N_y}, u_{k-p}, \ldots, u_{k+2-p-N_u}, \Delta u_{k+1-p}, \ldots, \Delta u_k)$$
$$\vdots$$
$$\hat{y}_{k+p/k} = \bar{G}(y_k, \ldots, y_{k+1-N_y}, u_{k-1}, \ldots, u_{k+1-N_u}, \Delta u_k, \ldots, \Delta u_{k+p-1}). \quad (13)$$

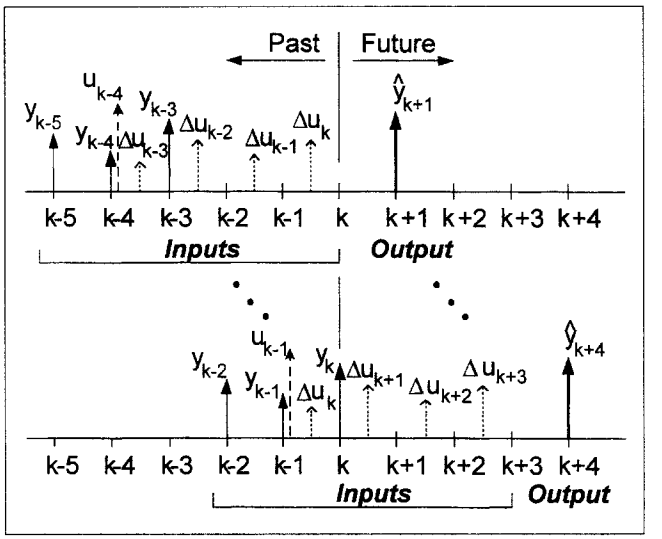

Figure 1. Timelines showing inputs to the p-step control model.

For this example, $p = 4$, $N_y = 3$, and $N_u = 2$. Predicted outputs are generated from known information only; previous model predictions for y are not used in model input.

To graphically illustrate the arrangement of the model inputs, consider a model with a prediction horizon, $p = 4$. Let us also assume that $N_y = 3$ and $N_u = 2$. Then the inputs to the networks will be as shown in Figure 1. Note that the future four predictions require only current and past output measurements. As discussed later in the fourth section, the unknown future input moves are calculated by the MPC controller. In the next section, we use the p-step control model formulated as a radial basis function network to predict $\hat{y}_{k+i}$. An example is provided that compares predictive performance of the p-step control model with the cascaded 1-step model in Eq. 3.

Dynamic Modeling Using RBF Network

Feedforward RBF networks have been widely used to model dynamic processes (Chen et al., 1990; Pottmann and Seborg, 1997). The dynamics are approximated by using past process input/output information in the network input. The RBF network is trained based on minimizing prediction error over the training trajectory (Moody and Darken, 1989).

Based on the p-step model for a single-input, single-output system, the input vector for an RBF network would be

$$x_k = [y_{k-p} \cdots y_{k-p+1-N_y} \quad u_{k-p-1} \cdots u_{k-p+1-N_u} \quad \Delta u_{k-p} \cdots \Delta u_{k-1}]. \quad (14)$$

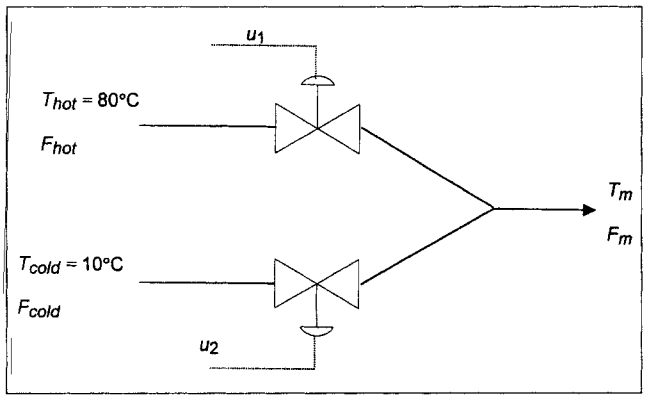

Figure 2. Hot/cold mixing process.

Measurable disturbances can be accounted by augmenting the input vector $\boldsymbol{x}_k$ to reflect the disturbance variables. The RBF prediction of the process output at instant k is

$$\hat{y}_k = \sum_{j=1}^{m_1} w_j \exp\left(-\frac{\| \boldsymbol{x}_k - \boldsymbol{t}_j \|^2}{\sigma^2}\right), \tag{15}$$

where m_1 represents the number of nodes, and $\boldsymbol{t}_j$ and w_j are the center and weight associated with the jth node, respectively. It is assumed that each node is of fixed width σ. Multiple inputs and multiple outputs can be handled by including these in the RBF input vector, $\boldsymbol{x}_k$.

The RBF network parameters are determined through a two-step training procedure. First, the center locations, $\boldsymbol{t}_j$, and width, σ, are fixed by an unsupervised training algorithm, that is, a clustering algorithm (Hush and Horne, 1993) (such as Kohonen feature map, k-means). The weights are obtained by regression with some form of regularization incorporated (German et al., 1992; Poggio and Girosi, 1990).

To compare future predictions by the p-step control model with the cascaded 1-step model of Eq. 1, we consider a simulation example of a hot/cold water mixing process. The process is shown in Figure 2. Water at 80°C and 10°C enters through the hot and cold legs, respectively. The mixed stream temperature, T_m, and flow rate, F_m, are controlled by the hot- and cold-leg control valves. The valves are regulated by control signals u_1 and u_2 whose range is 0–100%. The process is simulated by a first-principles model that describes the flow dynamics in response to valve-stem positions. The flow rate through each valve is a nonlinear function of the stem position. The temperature sensor is assumed to be located at the mixing point and is modeled as a third-order response to the true mixing-point temperature. The true mixing-point temperature is calculated as a flow-rate weighted average of the hot- and cold-leg temperatures. To emphasize nonlinearity, we choose to predict T_m and F_m rather than F_{hot} and F_{cold}, using signals u_1 and u_2, where

$$T_m = \frac{F_{\text{hot}}(u_1,u_2)T_{\text{hot}} - F_{\text{cold}}(u_1,u_2)T_{\text{cold}}}{F_{\text{hot}}(u_1,u_2) + F_{\text{cold}}(u_1,u_2)} \tag{16}$$

and

$$F_m = F_{hot}(u_1,u_2) + F_{cold}(u_1,u_2). \tag{17}$$

A summary of the network configuration for the cascaded 1-step and the p-step models is provided in Table 1. The training and test data were generated by exciting the process with random values of inputs u_1 and u_2 between 10% and 100% and holding the inputs for a random period between 5 and 60 s. The sample period was 5 s. In formulating the RBF input vector for the p-step control model, the prediction horizon p was set to eight samples, the time taken by the process to reach the new steady-state value. As noted in Table 1, the test set statistics for the 1-step model were better than those for the p-step model. This result was expected, since the 1-step model had access to the previous actual output measurement, while the most recent output measurement employed by the p-step model was eight measurements in the past. As discussed below, the test-set statistics are deceiving for situations where more than a single output prediction is needed.

A comparison of 20 consecutive future predictions of the cascaded 1-step and the p-step RBF models to a step change in the input u_2 from 60% to 40% is shown in Figure 3. Input u_1 was held constant at 25%. The 20 future predictions by the cascaded 1-step and the p-step models were generated using Eq. 3 and 13, respectively. Note that the p-step control model uses only measurements available up to the current instant to make eight future predictions, while the cascaded model uses future predictions as inputs to the RBF network. It is evident from Figure 3 that the 1-step cascaded model performs well until $k = 6$. However, beyond $k = 6$, error accumulation becomes significant and the 1-step model predic-

Table 1. RBF Model for Hot/Cold Mixing Example

	Cascaded 1-Step Model ($p = 1$)	p-Step Model ($p = 8$)
Training-set/test-set size	3,000/2,500	3,000/2,500
Number of hidden nodes	15	75
Number of input units	9	25
Model	N_y: 2 for T_m; 1 for F_m N_u: 1 for u_1; 1 for u_2 N_d 1 for T_{hot}; 1 for T_{cold}	N_y: 2 for T_m; 1 for F_m N_u: 1 for u_1; 1 for u_2 N_d: 1 for T_{hot}; 1 for T_{cold}
Model inputs	$T_{m,k-1}$, $T_{m,k-2}$, $F_{m,k-1}$, $u_{1,k-1}$, $u_{2,k-1}$, $T_{\text{hot},k-1}$, $T_{\text{hot},k-2}$, $T_{\text{cold},k-1}$, $T_{\text{cold},k-2}$	$T_{m,k-8}$, $T_{m,k-9}$, $F_{m,k-8}$ $u_{1,k-9}$, $u_{2,k-9}$ $\Delta u_{1,k-8}$, …, $\Delta u_{1,k-1}$, $\Delta u_{2,k-8}$, …, $\Delta u_{2,k-1}$ $T_{\text{hot},k-1}$, $T_{\text{hot},k-2}$, $T_{\text{cold},k-1}$, $T_{\text{cold},k-2}$
Test-set performance (root mean square error)	0.8°C for T_m 0.3 kg/min for F_m	2.1°C for T_m 1.1 kg/min for F_m

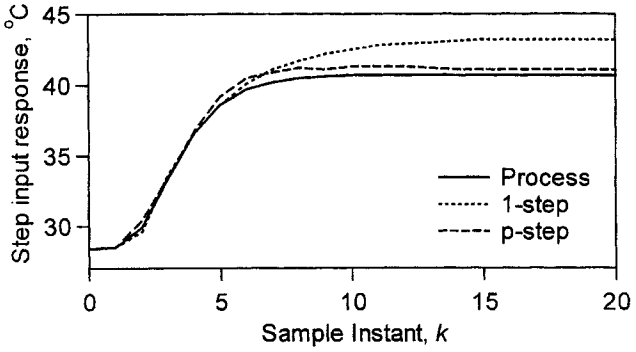

Figure 3. Cascaded one-step predictions vs. p-step model ($p = 8$).

tions degrade. This trend is consistent with the observation made by Su and McAvoy. On the other hand, the p-step control model (with $p = 8$) performs more uniformly over the entire horizon of 20 samples. Thus, although the cascaded 1-step model provides excellent one-step predictions, long-range predictions are problematic and better addressed using a p-step model. This advantage becomes more pronounced as the prediction horizon increases and would be beneficial in industrial applications of MPC, which typically use a large prediction horizon in the range of 20 to 50 (Marlin, 1995).

An RBF model can be easily manipulated to predict the steady-state gain at any point whenever the prediction horizon p exceeds the settling time for the process. In the absence of control moves between $k - p$ and $k - 1$, the model prediction at k, $\hat{y}_k$, will correspond to the steady-state measurement in response to input u_{k-p-1}. Thus, the RBF model can predict steady-state process output in a single computation step. Figure 4 illustrates the steady-state prediction of mixed stream temperature by the RBF p-step model. The input to the hot-leg valve was maintained at 25%, while the input to cold-leg valve, u_2, was varied from 20% to 100% in steps of 5% to generate the steady-state RBF model predictions and process response. The control input moves, $\Delta u_{1,k-i}$, and $\Delta u_{2,k-i}$, $i = 1, \ldots, p$, were set to zero. The similarity between the model prediction and process steady-state values

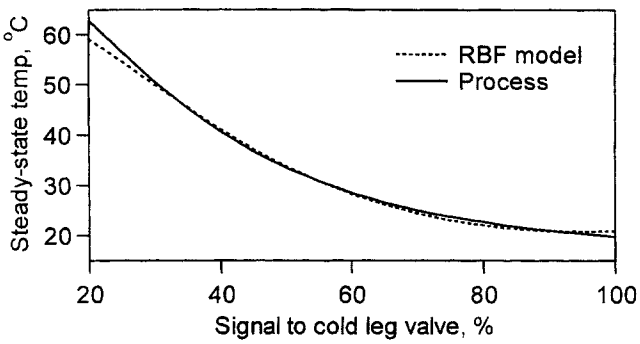

Figure 4. Steady-state process output vs. RBF model predictions.

Summary of RBF network is provided in Table 1. The signal to the hot-leg valve was maintained at a fixed value of 25%. The cold-leg valve input was varied in steps of 5%. Each RBF prediction was obtained in one computational step.

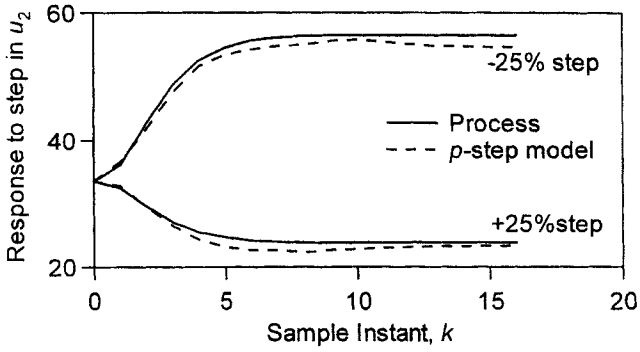

Figure 5. Performance of RBF model for approximation of dynamic response.

The two cases shown depict response of temperature to steps in cold-leg valve from 50% and 75% and from 50% to 25%. The hot-leg valve was maintained at 25%.

confirms that information on nonlinear sensitivity of the process is embedded within the RBF model.

Finally, to illustrate approximation of the process dynamics by the RBF model, we present results of a simulation exercise in Figure 5. The figure shows the results of a step test run on the process equations and the RBF model. The cold-water valve was stepped up from 50% to 75% in one simulation run and stepped down from 50% to 25% in the other. In both runs, the hot-leg valve was maintained at 25%. Excellent performance is indicated in both cases. In the following section, we parameterize the MPC control problem in terms of the p-step control model.

MPC Using p-Step Control Model

Model predictive control algorithms compute a manipulated variable profile over a control horizon by optimizing an objective function defined over the prediction horizon, subject to constraints. Only the first move is implemented and the procedure is repeated at every sampling instant. The optimization function reflects the process objectives that must be achieved, including minimization of overall cost. However, economic optimization is often performed by a higher level system, which determines the optimal setpoints. We utilize the traditional MPC optimization function that penalizes deviation of future model predictions, $\hat{y}_{k+j}$, from setpoints, r_{k+j}, while minimizing future control moves, Δu_{k+i}:

$$\phi = \sum_{i=1}^{p} \Gamma_i (r_{k+i} - \hat{y}_{k+i})^2 + \sum_{i=0}^{c-1} \Lambda_i (\Delta u_{k+i})^2. \qquad (18)$$

Variables p and c represent the prediction and control horizons, respectively, and Γ_i and Λ_i denote the error penalty and move suppression factors at the ith instant. Then, the MPC control law can be stated as

$$\underset{\Delta u_k, \Delta u_{k+1}, \ldots, \Delta u_{k+c-1}}{\arg(\min \phi)}$$

such that

$$y_{\min} \le \hat{y}_{k+i} \le y_{\max}$$
$$\Delta u_{\min} \le \Delta u_{k+i} \le \Delta u_{\max}$$
$$u_{\min} \le u_{k+i} \le u_{\max}. \qquad (19)$$

Based on the p-step control model, the future model predictions, $\hat{y}_{k+i}$, are seen to depend on past control moves and the future control-move variables, Δu_{k+i} (see Eqs. 14 and 15). The future control moves, $\Delta u_k, \ldots, \Delta u_{k+c-1}$, represent the decision variables for the optimization problem in Eq. 19. For calculation purposes, it is desirable to express $\hat{y}_{k+i}$ such that the unknown decision variables appear explicitly in the objective function. Since Gaussian functions are factorable, it is possible to express the model prediction, $\hat{y}_{k+i}$, as an inner product of two vectors. The unknown decision variables are contained in one vector and all other known past quantities, including the network weights, in the other. Thus, the RBF output can be rearranged as follows:

$$\hat{y}_{k+i} = \sum_{j=1}^{m_1} w_j \exp(\text{past} + \text{future})$$

$$= \begin{bmatrix} w_1 \exp(\text{past}) \\ \vdots \\ w_{m_1} \exp(\text{past}) \end{bmatrix}^T \begin{bmatrix} \exp(\text{future}) \\ \vdots \\ \exp(\text{future}) \end{bmatrix}, \quad (20)$$

$$\hat{y}_{k+1} = \hat{y}_{p,k+i}^T \hat{y}_{f,k+i}. \quad (21)$$

Subscripts p and f refer to the fact that the corresponding factors contain all known ($p$ast) and unknown ($f$uture) terms, respectively. Thus, only $\hat{y}_{f,k+i}$ needs to be computed during every function call by the optimization algorithm.

As an example of RBF output factorization, consider the p-step control model with a prediction horizon, $p = 2$, output order, $N_y = 1$, and input order, $N_u = 2$. Then, the output of a 3-node RBF network, $\hat{y}_{k+1}$, in response to input vector (see Eq. 14),

$$\boldsymbol{x}_{k+1} = [y_{k-1} \quad u_{k-2} \quad \Delta u_{k-1} \quad \Delta u_k] \quad (22)$$

can be written as

$$\hat{y}_{k+1} = \sum_{j=1}^{3} w_j \exp\left(- \frac{\| \boldsymbol{x}_{k+1} - \boldsymbol{t}_j \|^2}{\sigma^2} \right). \quad (23)$$

Let $t_{j,l}$ represent the lth element of the node-center vector, $\boldsymbol{t}_j$. Then Eq. 23 can be rewritten as the product of two vectors as follows,

$$\hat{y}_{k+1} =$$

$$\begin{bmatrix} w_1 \exp\left(\dfrac{(y_{k-1} - t_{1,1})^2 + (u_{k-2} - t_{1,2})^2 + (\Delta u_{k-1} - t_{1,3})^2}{\sigma^2} \right) \\ w_2 \exp\left(\dfrac{(y_{k-1} - t_{2,1})^2 + (u_{k-2} - t_{2,2})^2 + (\Delta u_{k-1} - t_{2,3})^2}{\sigma^2} \right) \\ w_3 \exp\left(\dfrac{(y_{k-1} - t_{3,1})^2 + (u_{k-2} - t_{3,2})^2 + (\Delta u_{k-1} - t_{3,3})^2}{\sigma^2} \right) \end{bmatrix}^T$$

$$\times \begin{bmatrix} \exp\left(\dfrac{(\Delta u_k - t_{1,4})^2}{\sigma^2} \right) \\ \exp\left(\dfrac{(\Delta u_k - t_{2,4})^2}{\sigma^2} \right) \\ \exp\left(\dfrac{(\Delta u_k - t_{3,4})^2}{\sigma^2} \right) \end{bmatrix}. \quad (24)$$

As discussed previously, the first factor on the righthand side represents $\hat{y}_{p,k+1}$ and contains all known measurements (including the current measurement) and the known past inputs applied to the process. The second vector represents $\hat{y}_{f,k+1}$ consists of the unknown control move, Δu_k, which is a decision variable of the optimization program presented in Eq. 19. All future p predictions can be expressed in a similar way. As shown later, this factorized form facilitates analytic expressions for the gradient and Hessian of the objective function.

To illustrate the factorization of the RBF model prediction based on the general p-step control-model structure, consider the model prediction at $k+1$. Let the center, $\boldsymbol{t}$, associated with a given node be partitioned as follows:

$$\boldsymbol{t} = \left[\boldsymbol{t}^{y^T} | \boldsymbol{t}^{u^T} | \boldsymbol{t}^{\Delta u^T} \right]^T. \quad (25)$$

Vectors, $\boldsymbol{t}^y$ and $\boldsymbol{t}^u$ contain elements corresponding to the delayed process outputs and inputs, while $\boldsymbol{t}^{\Delta u}$ corresponds to the elements of the input moves. Then the factors $\hat{y}_{p,k+1}$ and $\hat{y}_{f,k+1}$ can be written as follows:

$$\hat{y}_{p,k+1} = \begin{bmatrix} w_1 \exp\left\{ -\dfrac{1}{\sigma^2} \left[\begin{array}{l} (y_{k-p+1} - t_{1,1}^y)^2 + \cdots + (y_{k-p-N_y+1} - t_{N_y,1}^y)^2 + (u_{k-p} - t_{1,1}^u)^2 + \cdots + \\ (u_{k-p+2-N_u} - t_{N_u-1,1}^u) + (\Delta u_{k-p+1} - t_{1,1}^{\Delta u})^2 + \cdots + (\Delta u_{k-1} - t_{p-1,1}^{\Delta u})^2 \end{array} \right] \right\} \\ \vdots \\ w_{m_1} \exp\left\{ -\dfrac{1}{\sigma^2} \left[\begin{array}{l} (y_{k-p+1} - t_{1,m_1}^y)^2 + \cdots + (y_{k-p+1-N_y} - t_{N_y,m_1}^y)^2 + (u_{k-p} - t_{1,m_1}^u)^2 + \cdots + \\ (u_{k-p+2-N_u} - t_{N_u-1,m_1}^u) + (\Delta u_{k-p+1} - t_{1,m_1}^{\Delta u})^2 + \cdots + (\Delta u_{k-1} - t_{p-1,m_1}^{\Delta u})^2 \end{array} \right] \right\} \end{bmatrix} \quad (26)$$

and

$$\hat{y}_{f,k+1} = \begin{bmatrix} \exp\left[-\frac{1}{\sigma^2}\left\{\Delta u_k - t_{p,1}^{\Delta u}\right\}^2\right] \\ \exp\left[-\frac{1}{\sigma^2}\left\{\Delta u_k - t_{p,m_1}^{\Delta u}\right\}^2\right] \end{bmatrix}. \quad (27)$$

The center element, $t_{l,m}$ corresponds to the mth component of the lth node center. Thus, $\hat{y}_{f,k+i}$ contains all the input moves the optimizer must calculate, while $\hat{y}_{p,k+i}$ is formed by completely known quantities, including past measurements, prior control moves, and network weights.

As in linear MPC, we assume zero input moves after a control horizon of length c. Then, performing a similar exercise as before, the factors for the predictions, $\hat{y}_{k+1}, \ldots, \hat{y}_{k+p}$, based on c future moves, $\Delta u_k, \ldots, \Delta u_{k+c-1}$, take the following form:

$$\hat{y}_{p,k+i} = \boldsymbol{w} .* \exp\left\{-\frac{1}{\sigma^2}\left[\sum_{j=0}^{N_y-1}\left(y_{k-p-j+1}\boldsymbol{1} - t_{j+1}^{y}\right)^2 + \sum_{j=1}^{N_u-1}\left(u_{k-p-j+1}\boldsymbol{1} - t_j^{u}\right)^2 + \sum_{j=0}^{p-1}\left(\Delta u_{k-p+j+i}\boldsymbol{1} - t_{j+1}\right)^2\right]\right\} \quad (28)$$

where i ranges from 1 to p; the symbol $\boldsymbol{1}$ represents a column vector of size m_1 with unity elements, and the operator ".*" is used to denote element-by-element multiplication. Note that when the index, i, equals p, the final summation drops out, since j varies from 0 to -1. The factor $\hat{y}_{f,k+i}$ that contains the future moves the optimization algorithm must calculate is

$$\hat{y}_{f,k+i} = \exp\left[-\frac{1}{\sigma^2}\sum_{j=0}^{i-1}\left(\Delta u_{k+i-j-1}\boldsymbol{1} - t_{p-j}^{\Delta u}\right)^2\right], \quad i = 1, \ldots, c. \quad (29a)$$

$$\hat{y}_{f,k+i} = \exp\left[-\frac{1}{\sigma^2}\sum_{j=i-c}^{i-1}\left(\Delta u_{k+i-j-1}\boldsymbol{1} - t_{p-j}^{\Delta u}\right)^2\right], \quad i = c+1, \ldots, p. \quad (29b)$$

The exponential function in Eqs. 28 and 29 implies a term-by-term application to each element of the column vector in the square brackets. Column vector t_j is constructed by using the jth element of all m_1 centers. Thus, any future prediction, $\hat{y}_{k+1}$, can be computed using Eqs. 21, 28 and 29.

The objective function in Eq. 18 can be parameterized in terms of the network weights and the decision variables as follows:

$$\phi = \sum_{i=1}^{p}\Gamma_i\left[r_{k+i} - \hat{y}_{p,k+i}^T\hat{y}_{f,k+i}(\Delta \boldsymbol{u})\right]^2 + \sum_{i=0}^{c-1}\Lambda_i(\Delta u_{k+i})^2. \quad (30)$$

The gradient of the objective function can be computed analytically as follows:

$$(\Lambda\phi)_m = -2\sum_{i=1}^{p}\Gamma_i\left(r_{k+i} - \hat{y}_{p,k+i}^T\hat{y}_{f,k+i}\right)\hat{y}_{p,k+i}^T\frac{\partial \hat{y}_{f,k+i}}{\partial \Delta u_m} + 2\Lambda_m\Delta u_{k+m}, \quad (31)$$

where $(\nabla\phi)_m$ denotes the mth component of the gradient and $m = 0, \ldots, c-1$. The partial derivative of $\hat{y}_{f,k+i}$ is evaluated from Eq. 29 as

$$\frac{\partial \hat{y}_{f,k+i}}{\partial \Delta u_m} = \begin{cases} -\left(\frac{2}{\sigma^2}\right)\hat{y}_{f,k+i} .* \left(\Delta u_{k+m}\boldsymbol{1} - t_{p-i+m+1}^{\Delta u}\right), & \text{for } m+1 < i \\ 0, & \text{for } m+1 \geq i. \end{cases} \quad (32)$$

Similarly, the (m,n) component of the Hessian matrix can be computed as follows:

$$(\nabla^2\phi)_{m,n} = -2\sum_{i=1}^{p}\left\{\Gamma_i\left(r_{k+i} - \hat{y}_{p,k+i}^T\hat{y}_{f,k+i}\right)\left(\hat{y}_{p,k+i}^T\frac{\partial^2 \hat{y}_{f,k+i}}{\partial \Delta u_n \partial \Delta u_m}\right) - \left(\hat{y}_{p,k+i}^T\frac{\partial \hat{y}_{f,k+i}}{\partial \Delta u_m}\right) \times \left(\hat{y}_{p,k+i}^T\frac{\partial \hat{y}_{f,k+i}}{\partial \Delta u_n}\right)\right\}. \quad (33)$$

The second-order partial derivative in the preceding expression is calculated by

$$\frac{\partial^2 \hat{y}_{f,k+i}}{\partial \Delta u_n \partial \Delta u_m} = -\left(\frac{2}{\sigma^2}\right)\left[\hat{y}_{f,k+i} .* \delta_{mn}\boldsymbol{1} + \frac{\partial \hat{y}_{f,k+i}}{\partial \Delta u_n} \times \left(\Delta u_{k+m}\boldsymbol{1} - t_{p-j}\right)\right] \quad (34)$$

when $(m+1), (n+1) \leq i$, and zero otherwise.

Using the preceding expressions for the gradient and Hessian, the optimization of the objective function in Eq. 30 can be performed using sequential quadratic programming (SQP). The nonlinear output constraint in Eq. 19 can be written in terms of the factors of the model prediction and linearized using Eq. 32. The input constraints can be converted to input move constraints as in quadratic dynamic matrix control (Garcia and Morshedi, 1986). Analytical expressions for the gradient and Hessian greatly reduce the number of function calls, and hence the computational burden during optimiza-

tion. In addition, the separation of the decision variables in the model prediction ensures that only the unknown parts of the objective function and the gradient and Hessian required by the SQP algorithm are recalculated during optimization. Although the preceding expressions are derived for a single-input–single-output system, similar expressions can be written for a multiple-input–multiple-output (MIMO) system by augmenting the objective function $\hat{y}_p$, and $\hat{y}_f$ to include the multiple input/output variables.

Simulation Examples

To illustrate the performance of our proposed NMPC approach for a MIMO problem, consider control of the 2×2 hot/cold water mixing process discussed previously. In implementing the factorized RBF-based NMPC for a MIMO process with N_o outputs, the network weights are stored in a $m \times N_o$ matrix, W:

$$W = [w^1 \quad w^2 \quad \cdots \quad w^{N_o}]. \tag{35}$$

Equation 28 is then modified to provide $\hat{y}_p$ for the lth process outputs as follows:

$$\hat{y}^l_{p,k+i} = w^l . * \exp\left\{ -\frac{1}{\sigma^2}\left[\sum_{j=0}^{N_y-1} \left(y_{k-p-j+i} I - t^y_{j+1} \right)^2 + \sum_{j=1}^{N_u-1} \left(u_{k-p-j+i} I - t^u_j \right)^2 + \sum_{j=0}^{p-1} \left(\Delta u_{k-p+j+i} I - t_{j+1} \right)^2 \right] \right\}, \tag{36}$$

where l ranges from 1 to the number of process outputs, N_o. Vector $\hat{y}_f$, which contains the future input moves, remains unchanged. The summary for the p-step control model is shown in Table 1. The error penalties Γ_i, $i = 1, \ldots, p$ were set to 0.5 and 1.1 for the temperature and flow, respectively, with Λ_j, $j = 1, \ldots, c$ set to 0.7 for both the inputs, u_1 and u_2. The prediction and control horizons were assumed to be 8 and 3 sample intervals, respectively. At each control step, $\hat{y}_p$ was calculated only once. During optimization, each objective function call involved calculation of $\hat{y}_f$ and the computation of the model predictions based on Eq. 21. Optimization was performed using MATLAB's SQP function, *constr*, with analytical values of the gradient generated by Eqs. 31 and 32. MATLAB's SQP function calculated the Hessian by finite difference. Inputs u_1 and u_2 were constrained to lie between 10% and 100%. No constraints were imposed on the outputs, the mixed stream temperature, and flow rate. As in traditional linear MPC, the future p prediction are biased by the current value of the mismatch at each control execution step.

Control of the mixed-stream temperature and flow rate for multiple setpoint changes is presented in Figure 6. For the purpose of comparison, control of the mixing process by QDMC is also shown. The linear model used by QDMC is identified by step tests in the 40% to 60% region of the hot- and cold-valve signals. Disturbances enter the process at k = 50, 125, 200, and 375 by step changes in the hot- and cold-leg temperatures, as shown in the figure. The RBF model accounts for hot and cold fluid temperature changes by incorporating these measurements in the RBF input vector, as shown below,

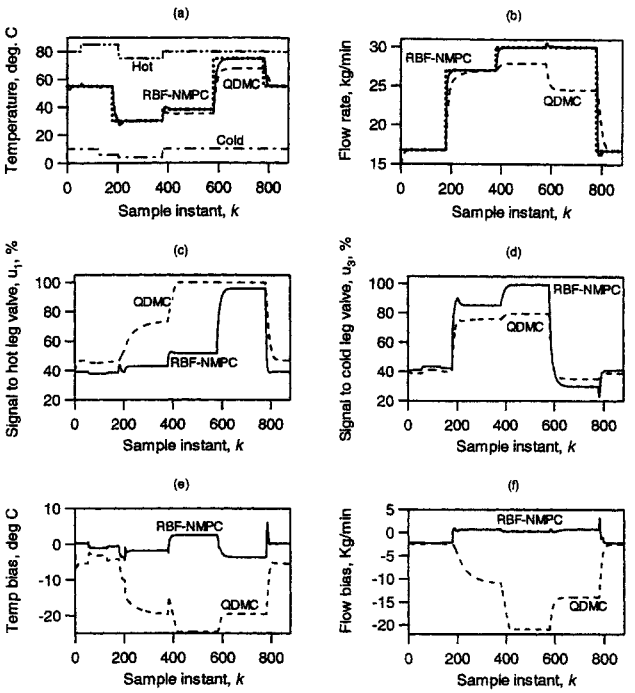

Figure 6. (a),(b) control of 2×2 mixing process in presence of measured disturbances in hot-leg (at k = 50, 200, and 375) and cold-leg (at k = 125, 200, and 375) temperatures by RBF-based NMPC and linear QDMC; (c),(d) control action by NMPC and QDMC controllers; (e),(f) process model mismatch for temperature and flow rate.

Setpoint changes were made at k = 175, 375, 575, and 775 for temperature and at k = 175, 375, and 775 for flow rate.

$$x^{\text{augmented}}_{k+i} = \left[x_{k+1} \;\vdots\; T_{\text{hot},k} \quad T_{\text{hot},k-1} \quad T_{\text{cold},k} \quad T_{\text{cold},k-1} \right]. \tag{37}$$

Similarly, the center vector is also augmented to include center elements corresponding to the disturbance measurements. During the future prediction phase, it is assumed that the current values of the hot- and cold-leg temperatures remain constant over the prediction horizon. In this example, the QDMC also uses a model to account for the temperature disturbances. Both the RBF-based NMPC and QDMC successfully reject these measured disturbances by use of measurement bias.

At k = 175, the temperature setpoint is changed from the initial value of 55°C to 30°C, while the flow setpoint is changed from 17 kg/min to 27 kg/min. The QDMC controller responds by making large positive changes in the cold leg valve to decrease the temperature of the mixed stream. The hot-leg valve has a higher throughput (nearly double) than the cold-leg valve at a given stem position. Thus, the QDMC controller also opens the hot-leg valve to allow for increased flow

rate. However, at lower temperatures the mixed stream temperature becomes increasingly sensitive to hot-water flow. When the flow-rate setpoint is further increased at $k = 375$, the hot-leg valve further opens until it saturates at 100% and the process is no longer maintained at the respective setpoints. At $k = 775$, the setpoints are brought to the region of linear model development and the linear QDMC controller is once again able to control the plant. On the other hand, the RBF-based NMPC controller exhibits excellent control over the entire operating region. Also shown in Figure 6 is the process-model mismatch for the two outputs. Tight control by the RBF-based NMPC controller is a consequence of good predictions by the RBF model over the entire range of operation.

It is of practical interest to investigate the computational requirements for the factorized RBF-model-based NMPC algorithm. To evaluate computational benefits, the computation time needed by the factorized RBF-model-based NMPC for the problem given earlier is compared with a nonfactorized RBF-based NMPC algorithm, which also uses the p-step control model. Unlike the factorized approach where $\hat{y}_p$ is calculated only once during each control execution, the nonfactorized algorithm computes the entire expression for RBF model predictions (similar to Eq. 2) during each iteration of the nonlinear program at every control execution. Also, the gradient information is calculated numerically with the nonfactorized approach. Table 2 documents the results. The results were generated by using the *tic-toc* command in MATLAB and represent the actual time needed by the computer to complete the controller-related calculations. As evident from Table 2, the factorized RBF-based NMPC is an order of magnitude more efficient than the nonfactorized approach. This is a significant reduction, considering the nonfactorized NMPC approach is two orders of magnitude more demanding than linear QDMC.

To test the RBF-model-based NMPC algorithm in the presence of unmeasured disturbances, we again consider control of the hot/cold mixing process. Unlike the previous example, however, the RBF model does not utilize the hot- and cold-leg temperatures. Additionally, various process nonidealities, including valve stiction, drifts in process parameters, drifts in temperature, and drifts and spikes in upstream pressure drops of the hot and cold legs, are built into the governing equations for the process. The control simulation results are shown in Figure 7. The mixed-stream flow-rate measurement was filtered using a CUSUM filter (Rhinehart, 1992) prior to input to the MPC scheme. No filtering was employed for the temperature measurement. The RBF network was trained on the unfiltered noisy data. Use of regularization during training ensured that the network does not overfit the noisy data. The number of nodes and the regularization parameter were selected so that similar statistics are obtained

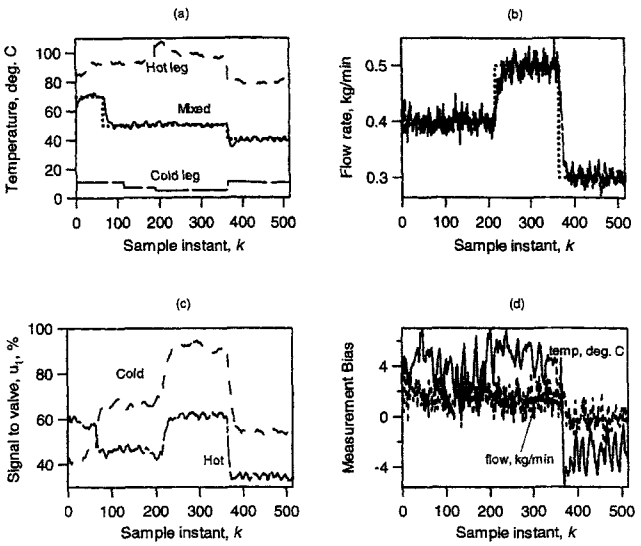

Figure 7. (a),(b) control of the 2×2 mixing process in presence of unmeasured disturbances in hot- and cold-leg temperatures (steps, drifts, and spikes); (c) control action by NMPC controller; (d) process model mismatch for temperature and flow rate.

Setpoint changes were made at $k = 0$, 65, and 365 for temperature and at $k = 0$, 200, and 365 for flow rate.

for network performances on the training and test sets. Based on the simulation results, it is observed that RBF-based MPC exhibits tight control of the mixing processes in face of the nonidealities described earlier.

As a final example, we consider control of a nonadiabatic, continuous, stirred-tank reactor with a first-order irreversible reaction. The heat of reaction is removed by circulation of cooling water in the reactor jacket. Uppal et al. (1974) described the following system of equations that govern the process dynamics:

$$\dot{x}_1 = -x_1 + Da(1-x_1)\exp\left(\frac{x_2}{1+x_2/\gamma}\right) \tag{38a}$$

$$\dot{x}_2 = -x_2 + BDa(1-x_1)\exp\left(\frac{x_2}{1+x_2/\gamma}\right) + \beta(u-x_2). \tag{38b}$$

States x_1 and x_2 represent reactant conversion and a dimensionless reactor temperature. The manipulated variable, u, is a dimensionless reactor jacket temperature. The steady-state

Table 2. Comparison of Computation Time Needed in Control of Hot/Cold Mixing Example*

	Factorized RBF-Based NMPC	Nonfactorized RBF-Based NMPC**	QDMC
Real time needed for Computation (s)	268	2,984	17

*Simulation for 900 samples or 4,450 s.
**Gradients evaluated numerically.

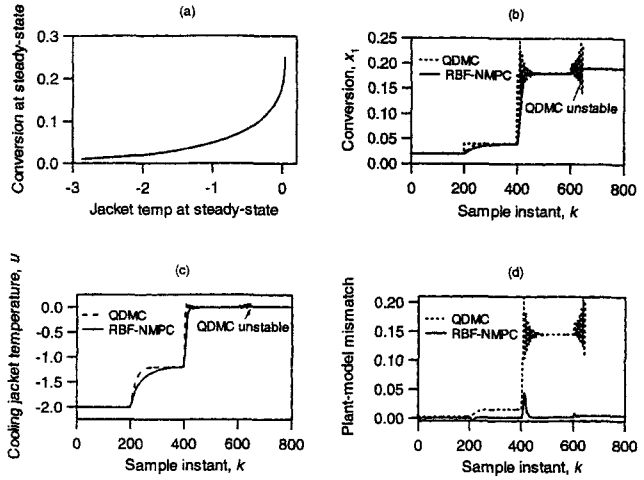

Figure 8. (a) steady-state characteristics of the CSTR process; (b) control of the CSTR process using a linear QDMC controller and RBF-based NMPC algorithm; (c) control action; (d) process model mismatch.

Setpoint changes were made at $k = 200$, 400, and 600. The linear QDMC control system becomes unstable in high-gain region ($k > 646$). The RBF-NMPC controller provides tight control in both the low- and high-gain regions.

characteristic of the reactor for parameters, $\beta = 3.0$, $\gamma = 40$, $B = 22$, and $Da = 0.082$, is shown in Figure 8. The process exhibits low gain at small values of conversion (1% to 4%) and considerably higher gain (in excess of 80 times the low gain) at higher conversions (> 18%). The performance of the RBF-based NMPC and linear QDMC is shown in Figure 8. A 125-node RBF network was trained to emulate process behavior over the 1% to 20% range of conversion. For the QDMC controller, a linear model was developed by conducting a step test at low conversions (1% to 4%). A prediction and control horizon of 9 and 3 samples, respectively, was used for both algorithms. At low conversion, the QDMC controller shows adequate performance. When a new setpoint ($x_1 =$ 0.18) is implemented, however, the QDMC controller does not recognize the high plant gain in this new region, resulting in aggressive control action. The system becomes unbounded when the next setpoint change (from 0.18 to 0.19) is implemented. On the other hand, the RBF-based NMPC controller provides tight control over the entire range of operation.

Conclusions

The most significant contribution of the proposed methodology is the ability to provide nonlinear control. The proposed NMPC technique integrates two well-accepted concepts in the modeling and control communities, RBF networks, and model-predictive control (NMPC). RBF-based NMPC controller design offers the potential of a generic methodology for a large number of industrial processes, as many process nonlinearities can be expressed in terms of a set of radial basis functions (Hartman et al., 1990).

The methodology can be applied to multivariable systems as illustrated by application to the 2×2 hot/cold-water mixing simulation. Conceptually, the methodology can be applied to any $m \times n$ system. However, the use of RBF networks for the process model introduces practical questions of scale. As evident from the development presented in the fourth section, the number of nodes used to model a multivariable system directly impacts the computational resources required to implement the proposed methodology. There is an obvious premium on efficient modeling to minimize the total number of RBF nodes. Additional work is required to determine the point at which scale becomes a problem. The potential to compensate for reduced control model fidelity with additional computational effort is clearly an issue of interest in this situation.

The potential problems of scale mentioned previously have been mitigated in part by leveraging the factorability of the Gaussian functions used in RBF networks. This property was exploited to express model predictions as an inner product of two vectors, one containing the decision variables of the MPC optimization program with the other made up entirely of known past quantities. This minimizes computational effort, since only the unknown parts of the objective function need to be reevaluated during each optimization call. Additional computational benefits are realized due to the compact representations for the MPC controller objective function and the availability of analytic expressions for the gradient and Hessian. The objective function takes the form of a sum of weighted Gaussian functions. Opportunities may exist to further exploit the radial symmetry of the Gaussian functions to tailor more efficient MPC optimization algorithms.

In the proposed NMPC scheme, the choice of the prediction horizon, p, influences both the controller performance as well as the control model. The p-step model specifically eliminates iterative dependency on model predictions. However, there is a cost. The most recent output measurement available for use by the model is always p steps prior in time. The p-step model was proposed due to problems with cascaded 1-step models using industrial magnitude prediction horizons. Performance of p-step models using these relatively long prediction horizons needs to be demonstrated.

Literature Cited

Chen, S., S. Billings, C. Cowen, and P. Gren, "Practical Identification of NARMAX Models Using Radial Basis Functions," *Int. J. Control*, **52**, 1327 (1990).

Garcia, C. E., and A. M. Morshedi, "Quadratic Dynamic Matrix Control," *Chem. Eng. Commun.*, **46**, 73 (1986).

German, S., E. Bienenstock, and R. Doursat, "Neural Networks and the Bias/Variance Dilemma," *Neural Comput.*, **4**, 1 (1992).

Hartman, E. J., J. D. Keeler, and J. M. Kowalski, "Layered Neural Networks with Gaussian Hidden Units as Universal Approximations," *Neural Comput.*, **2**, 210 (1990).

Henson, M. A., "Nonlinear Model Predictive Control: Current Status and Future Directions," *Comput. Chem. Eng.*, **23**, 187 (1998).

Hush, D. R., and B. G. Horne, "Progress in Supervised Neural Networks: What's New Since Lippmann," *IEEE Signal Process. Mag.*, **10**, 8 (1993).

Hussain, M. A., "Review of the Applications of Neural Networks in Chemical Process Control-Simulation and Online Implementation," *Artif. Intell. in Eng.*, **13**, 55 (1999).

Marlin, T. E., *Process Control — Designing Processes and Control Systems for Dynamic Performance*, McGraw-Hill, New York (1995).

Moody, J., and C. J. Darken, "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Comput.*, **1**, 281 (1989).

Narendra, K. S., and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Trans. Neural Networks*, **1**, (1990).
Pearson, R. K., and B. A. Qgunnaike, "Nonlinear Process Identification," *Nonlinear Process Control*, M. A. Henson and D. E. Seborg, eds., Prentice Hall, Englewood Cliffs, NJ (1997).
Poggio, T., and F. Girosi, "Networks for Approximation and Learning," *Proc. IEEE*, **78**, 1481 (1990).
Pottmann, M., and D. E. Seborg, "A Nonlinear Predictive Control Strategy Based on Radial Basis Function Models," *Comput. Chem. Eng.*, **21**, 965 (1997).
Rhinehart, R. R., "A CUSUM Type On-Line Filter," *Process Control Qual.*, **2**, 169 (1992).
Su, T. H., and T. J. McAvoy, "Artificial Neural Networks for Nonlinear Process Identification and Control," *Nonlinear Process Control*, M. A. Henson and D. E. Seborg, eds., Prentice Hall, Englewood Cliffs, NJ (1997).
Uppal, A., W. H. Ray, and A. B. Poore, "On the Dynamic Behavior of Continuous Stirred Tank Reactor," *Chem. Eng. Sci.*, **29**, 967 (1974).

Manuscript received Dec. 9, 1999, and revision received May 22, 2000.