Int. J. Appl. Math. Comput. Sci., 2009, Vol. 19, No. 2, 219–232
DOI: 10.2478/v10006-009-0018-2

INPUT CONSTRAINTS HANDLING IN AN MPC/FEEDBACK LINEARIZATION SCHEME

JIAMEI DENG [*], VICTOR M. BECERRA [**], RICHARD STOBART [*]

[*] Department of Aeronautical and Automotive Engineering
Loughborough University, Leicestershire LE11 3TU, UK
e-mail: {j.deng,r.k.stobart}@lboro.ac.uk

[**] School of Systems Engineering
University of Reading, Reading, RG6 6AY, UK
e-mail: v.m.becerra@reading.ac.uk

The combination of model predictive control based on linear models (MPC) with feedback linearization (FL) has attracted interest for a number of years, giving rise to MPC+FL control schemes. An important advantage of such schemes is that feedback linearizable plants can be controlled with a linear predictive controller with a fixed model. Handling input constraints within such schemes is difficult since simple bound contraints on the input become state dependent because of the nonlinear transformation introduced by feedback linearization. This paper introduces a technique for handling input constraints within a real time MPC/FL scheme, where the plant model employed is a class of dynamic neural networks. The technique is based on a simple affine transformation of the feasible area. A simulated case study is presented to illustrate the use and benefits of the technique.

Keywords: predictive control, feedback linearization, neural networks, nonlinear systems, constraints.

1. Introduction

The great success of predictive control is mainly due to its handling of constraints in an optimal way. There is early work on the integration of a feedback linearizing controller in an unconstrained MPC scheme (Henson and Seborg, 1993). However, in practical control schemes constraints have to be dealt with. Del-Re *et al.* (1993) dealt with the integration of output constraints by mapping the problem into a linear MPC problem. The basic idea of this method is to linearize the process using feedback linearization so that linear MPC solutions can be employed. However, in most of the cases, this mapping transforms the original input constraints of the process into nonlinear and state-dependent constraints, which cannot be handled by means of quadratic programming (Nevistic, 1994; Oliveiria *et al.*, 1995). Some methods have been presented in order to deal with such nonlinear constraints mapping while using QP routines to solve an approximate linear MPC problem (Henson and Kurtz, 1994; Oliveiria *et al.*, 1995). However, these approaches suffer from an important limitation as convergence to a feasible solution over the optimization horizon within the available time becomes a problem (Nevistic and Morari, 1995).

Ayala-Botto *et al.* (1999) proposed a new optimization procedure that guarantees a feasible control solution without input constraints violation over the complete optimization horizon, in a finite number of steps, while allowing only a small overall closed-loop performance degradation. Ayala-Botto *et al.* (1996) integrated in two complementary iterative procedures the solution of the QP optimization converting the nonlinear state-dependent constraints into linear ones. Van den Boom (1997) derived a robust MPC algorithm using feedback linearization at the cost of possibly conservative constraint handling. Nevistic and Primbs (1996) considered the problem of model/plant mismatch of MPC with feedback linearization. Guemghar *et al.* (2005) proposed a cascade structure of predictive control and feedback linearization for unstable systems. Scattolini and Colaneri (2007) presented a multi-layer cascaded predictive controller to guaranteed stability and feasibility. Ayala-Botto *et al.* (1999) presented a new solution for the problem of incorporating an input-output linearization scheme based on static neural networks in a predictive controller, considering the presence of level and rate inequality constraints applied to the plant input. However, this scheme can only handle single-input single-output systems. Moreover, Kurtz and Henson (1997) presented an input constraint mapping technology, the transformed constraints at each sampling time were decided by solving a constrained optimization problem. Also Casavola and Mosca (1996) tried to find box constraints by employing an optimization algorithm.

In this paper, focus will be positioned on input constraint handling on an integrated MPC+FL scheme based on dynamic neural networks. The aim is to find a constraint handling method that is suitable for real time application and avoids the shortcomings of the method by Ayala-Botto *et al.* (1999). Solving an additional optimization problem to find the transformed constraints is avoided in the proposed method.

The paper is organised as follows: Section 2 briefly describes the model predictive control formulation employed in this work. Section 3 describes the neural network structure used and discusses the training problem. Section 4 discusses the feedback linearization technique employed. Section 5 presents the proposed method for handling input constraints within an MPC/FL scheme. Section 6 discusses how the dynamic neural network is employed as a closed-loop observer. Section 7 presents a simulated case study. Concluding remarks are given in Section 8.

2. Model-based predictive control

The predictive control formulation employed (Maciejowski, 2002) is based on a linear, discrete-time state-space model of the plant. This makes sense within the framework of this paper as the plant is assumed to be feedback linearized prior to the application of predictive control. The model has the form

$$\begin{aligned} x_m(k+1) &= Ax_m(k) + Bu(k),\\ y_m(k) &= C_y x_m(k),\\ z(k) &= C_z x(k), \end{aligned} \tag{1}$$

where $x_m(k) \in \mathbb{R}^N$ is the state vector at time k, $u(k) \in \mathbb{R}^m$ is the vector of inputs, $y_m(k) \in \mathbb{R}^n$ is the vector of measured outputs, and $z(k) \in \mathbb{R}^\gamma$ is the vector of outputs which are to be controlled to satisfy some constraints, or to particular set-points, or both. In this work, a Kalman filter was used that can be described as follows:

$$\begin{aligned} \hat{x}_m(k+1|k) &= A\hat{x}_m(k|k-1) + Bu(k) + L\hat{e}(k|k),\\ \hat{y}_m(k|k-1) &= C_y\hat{x}_m(k|k-1),\\ \hat{z}(k|k-1) &= C_z\hat{x}_m(k|k-1), \end{aligned} \tag{2}$$

where $\hat{x}_m(k+1|k)$ is the estimate of the state at future time $k+1$ based on the information available at time k, $\hat{y}_m(k|k-1)$ is the estimate of the plant output at time k based on information at time $k-1$, L is the Kalman filter gain matrix and $\hat{e}(k|k)$ is the estimated error: $\hat{e}(k|k) = y_m(k) - \hat{y}_m(k|k-1)$.

The prediction model of MPC is similiar to Eqn. (2), without $\hat{e}(k|k)$. This formulation is inspired in the constrained algorithm presented in (Maciejowski, 2002). The cost function V minimised by the predictive controller penalises deviations of the predicted controlled outputs $\hat{z}(k+i|k)$ from a reference trajectory $r(k+i|k)$, and it also penalises changes in the future manipulated inputs $\Delta u(k+i|k)$. Define the cost function as follows:

$$\begin{aligned} V(k) = &\sum_{i=1}^{p} ||\hat{z}(k+i|k) - r(k+i|k)||^2_{Q_w(i)}\\ &+ \sum_{i=0}^{m-1} ||\Delta u(k+i|k)||^2_{R_w(i)}, \end{aligned} \tag{3}$$

where the prediction and control horizons are p and m, respectively, $Q_w(i)$ and $R_w(i)$ are output weight and input weight matrices, respectively. The cost function is subject to the inequality constraints:

$$\begin{aligned} u_{\max}(k) &\geq u(k+i-1|k) \geq u_{\min}(k),\\ \Delta u_{\max}(k) &\geq \Delta u(k+i-1|k) \geq \Delta u_{\min}(k),\\ z_{\max}(k) &\geq z(k+j|k) \geq z_{\min}(k), \end{aligned} \tag{4}$$

where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, p$. The constrained predictive control algorithm has been implemented in the C programming language using quadratic programming (Deng and Becerra, 2004), and it has been interfaced to SIMULINK through an S-Function as it is difficult to handle constraints using the MPC toolbox available in Matlab and in a real time case.

3. Dynamic neural networks

A dynamic neural network is used as a model of the plant for control synthesis in this paper as it can easily approximate any nonlinear systems. It can be expressed as a vector differential equation:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), \theta),\\ y(t) &= h(x(t), \theta), \end{aligned} \tag{5}$$

where $x \in \mathbb{R}^N$ represents the state vector, $y \in \mathbb{R}^n$ is the output vector, $u \in \mathbb{R}^m$ is the external input vector, $\theta \in \mathbb{R}^l$ is a parameter vector. Here f is a vector-valued function that represents the structure of the network, and h is a vector-valued function that represents the relationships between the state vector and the output vector.

The structure of the dynamic neural network used in this paper is a particular case of Eqn. (5). Any finite time trajectory of a given dynamic system can be approximated

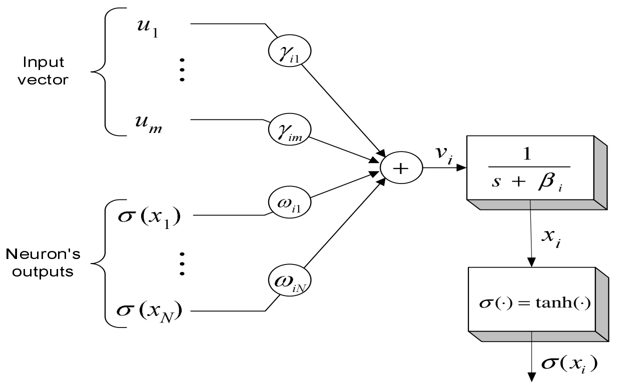


Fig. 1. Illustration of a dynamic neuron.

by the internal state of the output units of a continuous time dynamic neural network with N dynamic units, m inputs and n outputs (Garces *et al.*, 2003). These networks are neural unit arrays which have connections both ways between a pair of units, and from a unit to itself. The model is defined by a one-dimensional array of N neurons or units, in which each neuron of the dynamic neural network can be described as follows:

$$\dot{x}_i = -\beta_i x_i + \sum_{j=1}^{N} \omega_{ij}\sigma(x_j) + \sum_{j=1}^{m} \gamma_{ij} u_j, \tag{6}$$

where β_i, ω_{ij} and γ_{ij} are adjustable weights, with $1/\beta_i$ as a positive time constant and $n \le N$, x_i the activation state of Unit i, and $u_1, \dots, u_m$ the input signals (see Fig. 1).

The states of the first n neurons are taken as the output of the network, leaving $N-n$ units as hidden neurons. The network is defined by the vectorised expression of (6) as follows:

$$\dot{x} = -\beta x + \omega\sigma(x) + \gamma u, \tag{7}$$

$$\hat{y} = Cx, \tag{8}$$

where x are the coordinates on $\mathbb{R}^N$, $\beta \in \mathbb{R}^{N\times N}$, $\gamma \in \mathbb{R}^{N\times m}$, $u \in \mathbb{R}^m$, $\sigma(x) = [\sigma(x_1), \dots, \sigma(x_N)]^T$, $\sigma(\cdot)$ is a sigmoidal function, such as the hyperbolic tangent, $I_{n\times n}$ is the $n \times n$ identity matrix, $0_{n\times(N-n)}$ is an $n \times (N-n)$ matrix of zeros. $C = [I_{n\times n}\ 0_{n\times(N-n)}]$, and $\beta = \mathrm{diag}(\beta_1, \dots, \beta_N)$ is a diagonal matrix. This paper used a dynamic neural network described by Eqns. (7)–(8). The state vector x of the dynamic neural network of Eqn. (7) can be partitioned into the output state x^o and the hidden states x^h:

$$x = \begin{bmatrix} x^o \\ x^h \end{bmatrix}. \tag{9}$$

A dynamic neural network training problem can be cast as a nonlinear unconstrained optimization problem:

$$\min_{\theta} \quad F_M(\theta, Z_M) = \frac{1}{2M}\sum_{k=1}^{M} ||y(t_k) - \hat{y}(t_k|\theta)||^2, \tag{10}$$

where $Z_M = [y(t_k), u(t_k)]_{k=1,M}$ is a training data set, $y(t_k)$ represents the measured output, $\hat{y}(t_k|\theta)$ is the dynamic neural network output, and θ is a parameter vector.

The optimization problem associated with training usually exhibits local minima. Hence, training dynamic neural networks is typically performed using unconstrained local optimization with multiple random starts, global optimization methods, or hybrid methods. Global optimization or hybrid methods are usually better choices for training DNNs when dealing with multivariable plants (Garces *et al.*, 2003).

In this work dynamic networks are trained using a hybrid method involving the DIRECT algorithm (Perttunen *et al.*, 1993), which is a deterministic global optimization method, and a gradient based local optimization method. The solutions obtained by DIRECT are improved using gradient based local optimization. The training algorithm will find the parameters of the network in Eqn. (7) for which the error function F_M is minimized. The weight vector θ used by the algorithm is the aggregate of the neurons feedback weight matrix $\omega_{N\times N}$, the input weight matrix $\gamma_{N\times m}$, the state feedback weight vector $\beta_{N\times 1} = [\beta_1, \dots, \beta_N]^T$ and the initial values of the hidden states of the dynamic neural network, $x^h(t_0)$. That is,

$$\theta = \begin{bmatrix} \beta_{N\times 1} \\ \mathrm{vec}(\gamma_{N\times m}) \\ \mathrm{vec}(\omega_{N\times N}) \\ x^h(t_0)_{(N-n)\times 1} \end{bmatrix}, \tag{11}$$

where vec(.) transfers a matrix into a vector.

4. Approximate input-output feedback linearization and MPC integration

Many nonlinear control methods are based on state space models where the time derivative of the states depends nonlinearly on the states and linearly on the control inputs (Isidori, 1995), which are known as control affine systems and are described as follows:

$$\begin{aligned} \dot{x} &= f(x) + g(x)u, \\ y &= h(x), \end{aligned} \tag{12}$$

where $x \in \mathbb{R}^N$ is a state vector, $u \in \mathbb{R}^m$ is a vector of manipulated inputs, f and g are differentiable vector fields, and $y \in \mathbb{R}^n$ is a vector of outputs variables.

The purpose of input-output linearization is to introduce a new input variable v and a nonlinear transformation that uses state feedback to compute the original input u, so that a system described by Eqn. (12) behaves linearly from the new input v to the output y.

It is of considerable importance to assess which systems can be input-output linearized. The necessary and sufficient conditions for the existence of a feedback law are presented in (Isidori, 1995).

Input-output linearization and decoupling is a particular case of input-output linearization. An appropriate selection of the design parameters leads to a feedback linearized system where the i-th output depends on the i-th external input only. The basic linearizing-decoupling technique is described in (Isidori, 1995). In this paper, we apply the linearizing-decoupling technique to the dynamic neural network model, which is a control affine system, as formulated by Garces *et al.* (2002). The control affine system mappings related to Eqn. (12) can be related to the dynamic neural network parameters as follows:

$$\begin{aligned} f(x) &= -\beta x + \omega\sigma(x), \\ g(x) &= \gamma, \\ h(x) &= Cx. \end{aligned} \tag{13}$$

Consider the dynamic neural network described by Eqns. (7) and (8), and suppose that the dynamic neural network has a vector relative degree given by $r = [r_1, r_2, \dots, r_n]$ at a point x_0. Assume that the dynamic neural network has the same number of inputs and outputs ($n = m$). For arbitrary values $\hat{\lambda}_{ik}$ ($i = 1, \dots, n$ and $k = 0, \dots, r_i$), a state feedback law with

$$u = -S(x)^{-1}E(x) + S(x)^{-1}v, \tag{14}$$

where

$$S(x) = \begin{bmatrix} \hat{\lambda}_{1r_1} L_{g_1} L_f^{r_1-1} x_1 & \cdots & \hat{\lambda}_{1r_1} L_{g_n} L_f^{r_1-1} x_1 \\ \vdots & \ddots & \vdots \\ \hat{\lambda}_{nr_n} L_{g_1} L_f^{r_n-1} x_n & \cdots & \hat{\lambda}_{nr_n} L_{g_n} L_f^{r_n-1} x_n \end{bmatrix}_{n\times n} \tag{15}$$

and

$$E(x) = \begin{bmatrix} \sum\limits_{k=0}^{r_1} \hat{\lambda}_{1k} L_f^k x_1 \\ \vdots \\ \sum\limits_{k=0}^{r_n} \hat{\lambda}_{nk} L_f^k x_n \end{bmatrix}_{n\times 1}, \tag{16}$$

where $L_f^k h$ denotes a Lie derivative of order k of a scalar function $h(x)$ along vector field f, $\hat{\lambda}_{ik}$s are scalar design parameters, and r_i is the relative degree of the i-th output $\hat{y}_i$, produced when applied to a dynamic neural network described by Eqns. (7) and (8) a linearized-decoupled system that obeys

$$\sum_{k=0}^{r_i} \hat{\lambda}_{ik} \frac{\mathrm{d}^k \hat{y}_i}{\mathrm{d}t^k} = v_i, \quad i = 1 \dots n, \tag{17}$$

where $\hat{y}_i$, not y_i, is used as it is based on a neural network.

If the relative vector is well defined, $S(x)$ is invertible and

$$\det\left[\mathrm{diag}\left(\hat{\lambda}_{1r_1}, \hat{\lambda}_{2r_2}, \dots, \hat{\lambda}_{nr_n}\right)\right] \neq 0. \tag{18}$$

It should be noted that the approximate feedback linearization and decoupling laws described above require state information. Since the model employed in this work to generate the linearizing-decoupling laws is a dynamic neural network, the same model can also be used as a closed loop observer to provide state information to the feedback linearizing laws.

Once the feedback linearization is tested and validated on the actual plant, the constrained predictive controller can be used to control the linearized system. This controller can deal with disturbances arising from modelling errors and other sources, and it can handle constraints on plant variables. While handling output constraints using the method presented in this paper is straightforward, input constraints require special treatment due to the presence of the nonlinear transformations associated with feedback linearization. This aspect will be discussed in detail in Section 5. The scheme is illustrated in Fig. 2.

5. Algorithm for handling input constraints

Output constraints are straightforward for feedback linearization. Therefore, only input constraints are discussed in this paper. The feedback linearization-decoupling law (14) can be written as follows in discrete time based on the sampling time T_s, which is chosen using the Nyquist-Shannon theorem:

$$\begin{aligned} &u(k+1|k) \\ &\quad = S(x(k+1|k))^{-1} \\ &\qquad \cdot (v(k+1|k) - E(x(k+1|k))), \end{aligned} \tag{19}$$

where $i = 1, 2, \dots, m$, $S(x(k+1|k))$ and $E(x(k+1|k))$ are given by Eqns. (15) and (16), and $x(k+1|k)$ is a sampled state estimate obtained from an observer as as described in Section 6. Consider the following constraints on the input u_k:

$$\begin{aligned} u_{\min}(k) &\le u_k \le u_{\max}(k), \\ \Delta u_{\min}(k) &\le \Delta u_k \le \Delta u_{\max}(k). \end{aligned} \tag{20}$$

From Eqns. (19) and (20), it is possible to see that the resulting constraints on $v(k+i-1|k)$ are state dependent and hence are not suitable to be enforced by means of an MPC algorithm such as the one presented in Section 2, which assumes constant bounds. Although MPC based on quadratic programming could solve linear-variant, even nonlinear-variant constraints, it is not suitable for a real-time control purpose. This paper proposes the use of an

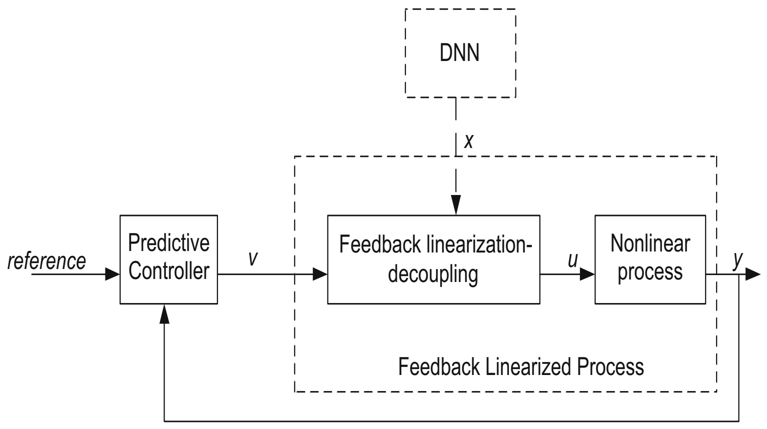

DNN
x
reference
Predictive Controller
v
Feedback linearization-decoupling
u
Nonlinear process
y
Feedback Linearized Process

Fig. 2. Hybrid MPC/FL control scheme based on a dynamic neural network.

affine transformation to solve this problem for a real-time application. In order to describe the technique in a simple way, this paper considers a two-input two-output system. However, this technique can be used in cases with more inputs and outputs, simply by increasing the dimensions.

Assume that $u = [u_1(k+i-1|k)\ u_2(k+i-1|k)]^T$. Then, at sampling time k,

$$\begin{aligned} u_{1\min} &\le u_1(k+i-1|k) \le u_{1\max}, \\ u_{2\min} &\le u_2(k+i-1|k) \le u_{2\max}, \\ \Delta u_{1\min} &\le \Delta u_1(k+i-1|k) \le \Delta u_{1\max}, \\ \Delta u_{2\min} &\le \Delta u_2(k+i-1|k) \le \Delta u_{2\max}. \end{aligned} \tag{21}$$

Figure 3 illustrates the feasible area of $u_1(k+i-1|k)$ and $u_2(k+i-1|k)$ at sampling time k. It has four corners labelled by O, R, S, T. The coordinates of the corners are $(u_{1\min}, u_{2\min})$, $(u_{1\min}, u_{2\max})$, $(u_{1\max}, u_{2\max})$, $(u_{1\max}, u_{2\min})$, respectively. Note that since these are bound constraints, the sides of the box are parallel to the co-ordinates axes x_c and y_c. Figure 4 illustrates the feasible area of v_1 and v_2 after each point in the feasible area of $u_1(k+i-1|k)$ and $u_2(k+i-1|k)$ is multiplied by the matrix $S(x(k+1|k))$ and translated by $E(x(k+1|k))$. The coordinates of these four corners O, R, S, T have changed into $(a_1, a_2), (b_1, b_2), (c_1, c_2), (d_1, d_2)$. The new coordinates have the following relationship with the old ones:

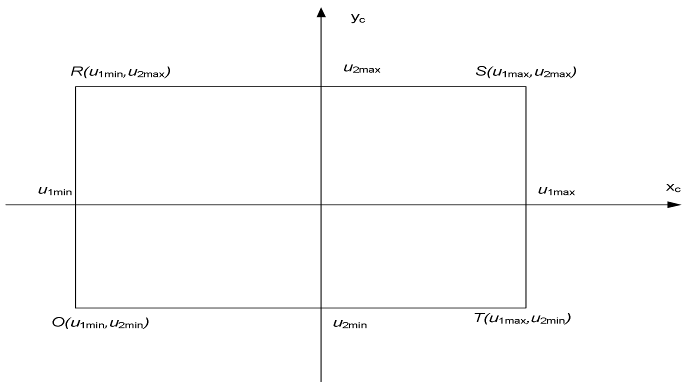
$$\begin{aligned} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} &= S(x(k+1|k)) \begin{bmatrix} u_{1\min} \\ u_{2\min} \end{bmatrix} \\ &\quad + E(x(k+1|k)), \\ \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} &= S(x(k+1|k)) \begin{bmatrix} u_{1\min} \\ u_{2\max} \end{bmatrix} \\ &\quad + E(x(k+1|k)), \\ \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} &= S(x(k+1|k)) \begin{bmatrix} u_{1\max} \\ u_{2\max} \end{bmatrix} \\ &\quad + E(x(k+1|k)), \\ \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} &= S(x(k+1|k)) \begin{bmatrix} u_{1\max} \\ u_{2\min} \end{bmatrix} \\ &\quad + E(x(k+1|k)). \end{aligned} \tag{22}$$

The feasible area in Fig. 4 of the transformed inputs is not suitable for use with the predictive control algorithm described in Section 2, as the box is oblique with respect to the co-ordinate system x_c–y_c. From Eqn. (22), it is not difficult to see that the side OT is parallel with the side RS; the side OR is parallel with the side TS. Therefore, the feasible area of the transformed inputs can be expressed in a suitable form for the predictive control algorithm described in Section 2, after an affine change of coordinates, which involves in the 2D case a rotation and a translation. Figure 5 shows the new co-ordinates. Axis x'_c is parallel with sides OT and RS. Axis y'_c is parallel with the sides OR and TS. In the new coordinates, the points O, R, S, T map into $(a'_1, a'_2), (b'_1, b'_2), (c'_1, c'_2), (d'_1, d'_2)$. It is not difficult to prove that $a'_1 = b'_1$, $a'_2 = d'_2$, $b'_2 = c'_2$ and $c'_1 = d'_1$.

Suppose the affine change of variables is done by means of a 2×2 affine matrix H and a translation. Matrix H can be obtained using the following equations:

$$\begin{aligned} \begin{bmatrix} a'_1 \\ a'_2 \end{bmatrix} &= H\left\{\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - E(x(k+1|k))\right\}, \\ \begin{bmatrix} b'_1 \\ b'_2 \end{bmatrix} &= H\left\{\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} - E(x(k+1|k))\right\}, \\ \begin{bmatrix} c'_1 \\ c'_2 \end{bmatrix} &= H\left\{\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} - E(x(k+1|k))\right\}, \\ \begin{bmatrix} d'_1 \\ d'_2 \end{bmatrix} &= H\left\{\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} - E(x(k+1|k))\right\}. \end{aligned} \tag{23}$$

Notice that by choosing to have the origin of the x'_c–y'_c co-ordinate system centered in the box, the effect of the shift $E(x(k+1|k))$ from Eqn. (22) is cancelled by Eqn. (23). Hence it is safe to ignore the effect of the shift $E(x(k+1|k))$ when computing the new bounds for



y_c
$R(u_{1\min}, u_{2\max})$
$u_{2\max}$
$S(u_{1\max}, u_{2\max})$
$u_{1\min}$
$u_{1\max}$
x_c
$O(u_{1\min}, u_{2\min})$
$u_{2\min}$
$T(u_{1\max}, u_{2\min})$

Fig. 3. Illustration of the feasible area of the manipulated inputs: Feedback linearization incorporated in a predictive control scheme after the affine transformation.

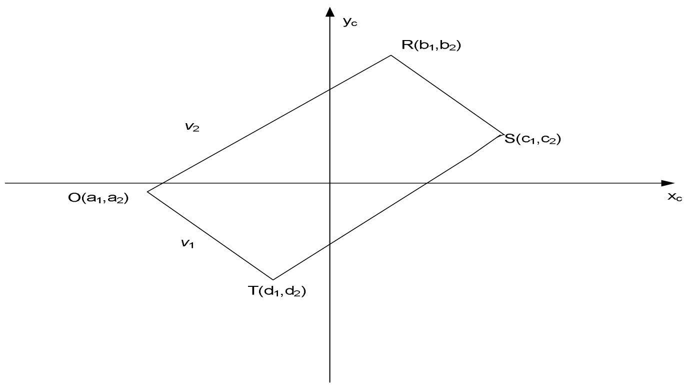
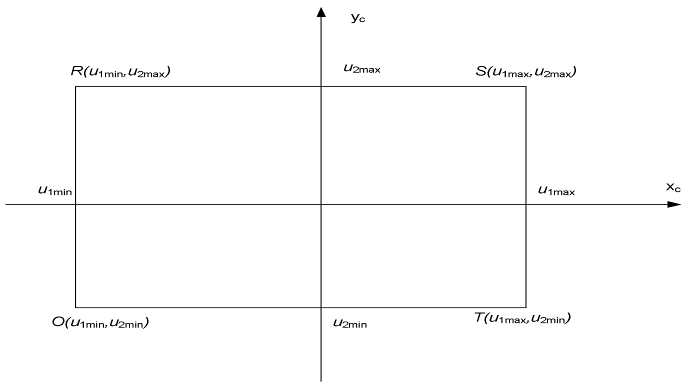


Fig. 4. Illustration of the feasible area of the transformed inputs.



y_c
$R(u_{1\min}, u_{2\max})$
$u_{2\max}$
$S(u_{1\max}, u_{2\max})$
$u_{1\min}$
$u_{1\max}$
x_c
$O(u_{1\min}, u_{2\min})$
$u_{2\min}$
$T(u_{1\max}, u_{2\min})$

Fig. 5. New coordinates after the affine transformation.

$v(k+i-1|k)'$ in the x'_c–y'_c co-ordinates. The following relationships are also true:

$$\begin{aligned} b'_2 - a'_2 &= \sqrt{(b_1-a_1)^2 + (b_2-a_2)^2}, \\ c'_1 - b'_1 &= \sqrt{(b_1-c_1)^2 + (b_2-c_2)^2}, \\ c'_2 - d'_2 &= \sqrt{(d_1-c_1)^2 + (d_2-c_2)^2}, \\ d'_1 - a'_1 &= \sqrt{(d_1-a_1)^2 + (d_2-a_2)^2}. \end{aligned} \tag{24}$$

After a'_1, a'_2, b'_1, b'_2, c'_1, c'_2, d'_1, and d'_2 are obtained from Eqn. (24), H can be easily obtained from Eqn. (23).

Thus any vector z in the x_c–y_c co-ordinates can be transformed into a vector z'_c in the x'_c–y'_c co-ordinates as follows:

$$z' = H(z - E(x(k+1|k))), \tag{25}$$

while any vector z' in the x'_c–y'_c co-ordinates can be transformed into a vector in the x_c–y_c co-ordinates as

$$z = H^{-1}z' + E(x(k+1|k)). \tag{26}$$

After the affine transformation, for a given time k, $v(k+i-1|k)'$ will be bounded as follows:

$$\begin{bmatrix} a'_1 \\ a'_2 \end{bmatrix} \le v(k+i-1|k)' \le \begin{bmatrix} c'_1 \\ c'_2 \end{bmatrix}. \tag{27}$$

Therefore, we could change the first line of Eqn. (4) into Eqn. (27) as this is the real input constraint for MPC.

For a given k,

$$\begin{aligned} v(k-1|k-2) &= S(x(k-1|k-2))u(k-1|k-2) \\ &\quad + E(x(k-1|k-2)), \\ v(k|k-1) &= S(x(k|k-1))u_k + E(x(k|k-1)). \end{aligned} \tag{28}$$

Hence,

$$\begin{aligned} &\Delta v(k+i-1|k) \\ &\quad = v(k+i-1|k) - v(k-1|k-2) \\ &\quad = S(x(k+1|k))u_k \\ &\qquad - S(u(k-1|k-2))u(k-1|k-2) \\ &\qquad + E(x(k+1|k)) \\ &\qquad - E(x(k-1|k-2)). \end{aligned} \tag{29}$$

$\Delta v(k+i-1|k)'$ is obtained by transforming $\Delta v(k+i-1|k)$ into the new coordinates x'_c and y'_c, i.e.,

$$\Delta v(k+i-1|k)' = H\Delta v(k+i-1|k). \tag{30}$$

Combining Eqns. (29) and (30), we get

$$\begin{aligned} &\Delta v(k+i-1|k)' \\ &\quad = HS(x(k+1|k))\Delta u(k+1|k) \\ &\qquad + Hu(k-1|k-2)(S(x(k+1|k)) \\ &\qquad - S(x(k-1|k-2))) \\ &\qquad + H(E(x(k+1|k)) \\ &\qquad - E(x(k-1|k-2)). \end{aligned} \tag{31}$$

Equation (31) also can be writen as

$$\begin{aligned} &\Delta v(k+i-1|k)' \\ &\quad = H\Delta v(k+i-1|k) \\ &\qquad + Hu(k-1|k-2)(S(x(k+1|k)) \\ &\qquad - S(x(k-1|k-2))) \\ &\qquad + H(E(x(k+1|k)) \\ &\qquad - E(x(k-1|k-2))). \end{aligned} \tag{32}$$

The following two equations will be obtained by using constraints on Eqn. (31):

$$\begin{aligned} &\Delta v'_{\min} \\ &\quad = H\Delta v_{\min}(k+i-1|k) \\ &\qquad + Hu(k-1|k-2)(S(x(k+1|k)) \\ &\qquad - S(x(k-1|k-2))) \\ &\qquad + H(E(x(k+1|k)) - E(x(k-1|k-2))). \end{aligned} \tag{33}$$

$$\begin{aligned} &\Delta v'_{\max} \\ &\quad = H\Delta v_{\max}(k+i-1|k) \\ &\qquad + Hu(k-1|k-2)(S(x(k+1|k)) \\ &\qquad - S(x(k-1|k-2))) \\ &\qquad + H(E(x(k+1|k)) - E(x(k-1|k-2))). \end{aligned} \tag{34}$$

Therefore, we could change the second line of Eqn. (4) into Eqns. (33) and (34) as this is the real rate constraint for MPC. Now that the constraints have been transformed into a new coordinate frame. The next step is to transform the inputs to the predictive controller. Figure 6 illustrates the control scheme, where the forward transformation and its inverse are denoted by H blocks and H^{-1} blocks, respectively.

The use of the affine transformation to define the bounds for $v(k+i-1|k)$ does not affect the transfer function of the plant seen by the predictive controller, provided the output and the reference signals undergo the same transformation. This can be proved as follows. For the scheme in Fig. 6, the matrix transfer function can be written as follows:

$$\begin{aligned} &\frac{y(s)}{v(s)} \\ &= \begin{bmatrix} \frac{1}{c_{1n}s^n+\cdots+c_{11}s+c_{10}} & 0 \\ 0 & \frac{1}{c_{2n}s^n+\cdots+c_{21}s+c_{20}} \end{bmatrix}, \end{aligned} \tag{35}$$

Here, $c_{1n}, \dots, c_{10}$ and $c_{2n}, \dots, c_{20}$ are the arbitrary values chosen when applying feedback linearization-decoupling. From Fig. 6, $y_1(s)$ can be obtained by

$$y_1(s) = H \times G(s) \times H^{-1}v_1(s), \tag{36}$$

Let

$$G_1(s) = H \times G(s) \times H^{-1}, \tag{37}$$

where G_1 is the new transfer function for the predictive controller, which can easily be transformed into a state space form. Table 1 gives a step-by-step summary of the method for handling input constraints in an MPC/FL. Consider now the multi-input multi-output case and suppose the number of inputs and outputs is m. The number of inequalities in Eqn. (21) will be $2m$. The dimension of Fig. 3 should be m and the corner number of the feasible area will be 2^m. The dimension of Fig. 4 is also m and the corner number of new coordinates is 2^m. The number of equations in Eqn. (22) will become 2^m, so will the number of equations in Eqn. (23). H can still be obtained using equations which are similar to (22) and (23). Therefore, the multi-input and multi-output case can easily be solved.

6. Using the dynamic neural network model as a closed-loop observer

Poznyak *et al.* (2001) proposed the use of dynamic neural networks as closed loop observers. In this paper, a

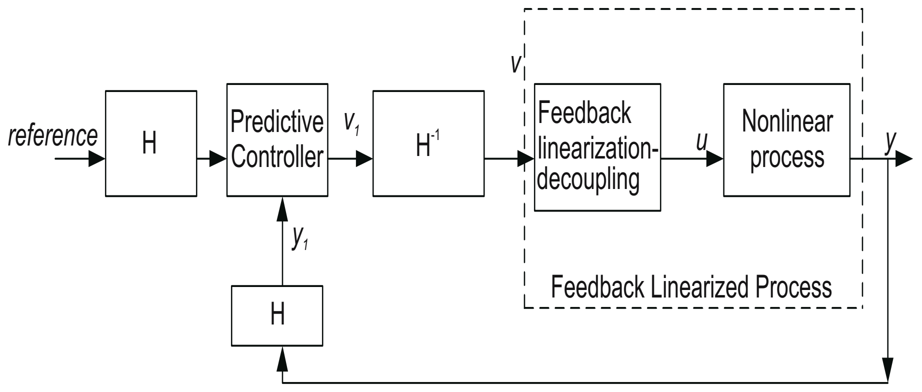


Fig. 6. Feedback linearization incorporated in a predictive control scheme after the affine transformation.

Table 1. Method for handling input constraints in an MPC/FL scheme.

Step	Description
Step 1	Set $k = 0$. Obtain a'_1, a'_2, b'_1, b'_2, c'_1, c'_2, d'_1, and d'_2 according to Eqn. (24) given that $a'_1 = b'_1$, $a'_2 = d'_2$, $b'_2 = c'_2$ and $c'_1 = d'_1$.
Step 2	Obtain H is from Eqn. (23).
Step 3	Define the bounds for $v(k+i-1\|k)'$ as given in Eqn. (27).
Step 5	Calculate $\Delta v_{\min}{}'$ and $\Delta v_{\max}{}'$ according to Eqns. (33) and (34).
Step 6	Measure the current plant output y_k and transform it using Eqn. (25) to obtain y'_k.
Step 7	Transform the current reference r_k using Eqn. (25) to obtain r'_k.
Step 8	Execute one step of the predictive control algorithm to obtain $v(k+i-1\|k)'$.
Step 8	Transform $v(k+i-1\|k)'$ to obtain $v(k+i-1\|k)$ using Eqn. (26).
Step 9	Provide $v(k+i-1\|k)$ to the feedback linearization/decoupling algorithm.
Step 10	Set $k = k+1$ and return to Step 1.

dynamic neural network is employed as a closed-loop observer to provide state information to be used by the feedback linearization laws. A dynamic neural network can be used as either an open-loop observer or a closed-loop observer. A dynamic neural network was used as an open-loop observer in (Garces *et al.*, 2003; Garces, 2000), where the state vector of the dynamic neural network was used to provide state information to the feedback linearization law, with no plant output information used to correct the state estimates. A closed-loop observer with output feedback can reduce the state estimation error and improve the robustness of the estimator. An extended Kalman filter algorithm (Maybeck, 1982) is employed in this work as a state estimator, where a dynamic neural network is employed to represent the plant dynamics.

We used a simplified extended Kalman filter which assumes the following stochastic model of the plant dynamics and discrete observations with a sampling interval T_s:

$$\begin{aligned} \dot{x}(t) &= F(x,u) + Gw(t) \\ &= -\beta x(t) + \omega\sigma(x(t)) + \gamma u(t) + Gw(t), \\ x(0) &= x_0, \\ y(t_k) &= Cx(t_k) + v(t_k), \end{aligned} \tag{38}$$

where x is the state vector, y is the measurement vector, u is the external input vector, w is assumed to be zero mean continuous white noise with covariance matrix Q, v is assumed to be zero mean discrete white noise with covariance matrix R, and x_0 is a random vector with zero mean and covariance matrix P_0.

The algorithm assumes that the Jacobian matrix remains constant between samples, so that

$$\Phi(k) = \exp\left(\left[\frac{\partial F(x,u))}{\partial x}\right]_{t_k} T_s\right). \tag{39}$$

A summary of the algorithm is as follows:

- *Algorithm initialisation*: Give values to Q, R, P_0 and G. Set $t = 0$, $k = 0$, $P(0) = P_0$, $x(0) = x_0$.

- ***Step 1:*** *Time update*. Integrate from t_{k-1} to t_k the following differential equation to obtain $\hat{x}(k)^-$:

$$\dot{x} = -\beta x(t) + \omega\sigma(x(t)) + \gamma u(t). \tag{40}$$

Also compute the *a priori* state covariance from

$$P^-(k) = \Phi(k)P(k-1)\Phi(k)^T + Q. \tag{41}$$

- ***Step 2:*** *Measurement update*. Measure the current output $y(k)$ and then compute the Kalman filter gain

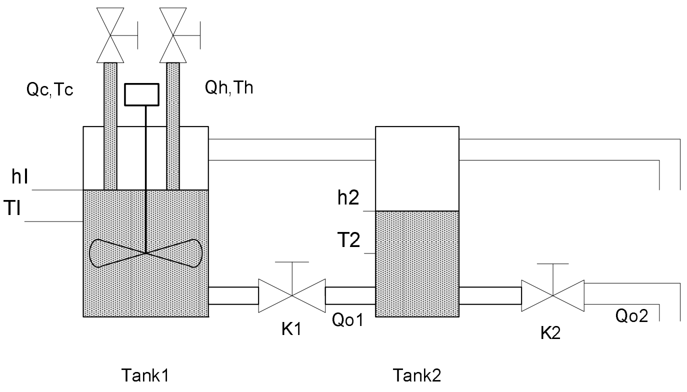


Fig. 7. Schematic diagram of the two tank mixing process.

$K(k)$, the corrected state estimate $\hat{x}(k)$, and the corrected covariance matrix $P(k)$:

$$K(k) = P^-(k)C^T(CP^-(k)C^T + R)^{-1}, \tag{42}$$

$$\hat{x}(k) = \hat{x}^-(k) + K(k)(y(k) - y^-(k)), \tag{43}$$

$$P(k) = (I - K(k)C)P^-(k). \tag{44}$$

Set $k = k + 1$ and return to Step 1.

7. Case study

This section describes a case study based on a simulated mixing process. A section of the process is shown in Fig. 7.

Two streams of cold and hot water enter the first tank, where a motorized mixer operates. These streams are controlled by means of two pneumatic valves. The two tanks are interconnected via a lower pipe with a manual valve that is normally open. The outlet valve from Tank 2 is also normally open. The two measured variables are the level of Tank 1 and the temperature of Tank 2. A simple modular model of the mixing process is described below (Becerra *et al.*, 2001). The model is obtained from mass and energy balances. The main simplifying assumptions that were made to derive this model were as follows:

- The mixing is perfect.
- Heat losses are negligible.
- The valves have linear characteristics.
- The temperatures of the cold and hot water streams are constant.

The model for the first tank is given by the following algebraic and differential equations:

$$\begin{aligned} Q_c &= C_c v_c/100, \\ Q_h &= C_h v_h/100, \\ Q_{o1} &= K_1\sqrt{h_1 - h_2}, \\ \frac{\mathrm{d}h_1}{\mathrm{d}t} &= (Q_c + Q_h - Q_{o1})/A_1, \\ \frac{\mathrm{d}T_1}{\mathrm{d}t} &= [Q_c(T_c - T_1) + Q_h(T_h - T_1)]/(A_1 h_1), \end{aligned} \tag{45}$$

where v_c is the opening of the cold water control valve (%), v_h is the opening of the hot water control valve (%), C_c is the constant of the cold water control valve ($\mathrm{cm}^3/\mathrm{s}\%$), C_h is the constant of the cold water control valve ($\mathrm{cm}^3/\mathrm{s})\%$), Q_c is the cold water flow rate (cm^3/s), Q_h is the hot water flow rate (cm^3/s), Q_{o1} is the outlet flow rate from Tank 1 (cm^3/s), h_1 is the liquid level in Tank 1 (cm), T_1 is the liquid temperature in Tank 1 (^{o}C), T_c is the temperature of the cold water stream (^{o}C), T_h is the temperature of the hot water stream (^{o}C), K_1 is the restriction of the interconnection valve and pipe ($\mathrm{cm}^{5/2}/\mathrm{s}$), A_1 is the cross-sectional area of Tank 1 (cm^2).

The model for the second tank is given by the following algebraic and differential equations:

$$\begin{aligned} Q_{o2} &= K_2\sqrt{h_2}, \\ \frac{\mathrm{d}h_2}{\mathrm{d}t} &= (Q_1 - Q_{o2})/A_2, \\ \frac{\mathrm{d}T_2}{\mathrm{d}t} &= [Q_{o1}(T_1 - T_2)]/(A_2 h_2), \end{aligned} \tag{46}$$

where Q_{o2} is the outlet flow rate from Tank 2 (cm^3/s), h_2 is the liquid level in Tank 2 (cm), T_2 is the liquid temperature in Tank 2 (^{o}C), K_2 is the restriction of the Tank 2 outlet valve and pipe ($\mathrm{cm}^{5/2}/\mathrm{s}$), A_2 is the cross-sectional area of Tank 2 (cm^2).

The following values were used for the model parameters: $A_1 = 289\ \mathrm{cm}^2$, $A_2 = 144\ \mathrm{cm}^2$, $C_c = 280.0\ \mathrm{cm}^3/\mathrm{s}\%$, $C - h = 100.0\ \mathrm{cm}^3/\mathrm{s}\%$, $T = 20^o\mathrm{C}$, $T_h = 72^o\mathrm{C}$, $K_1 = 30\ \mathrm{cm}^{5/2}/\mathrm{s}$, $K_2 = 30\ \mathrm{cm}^{5/2}/\mathrm{s}$. Q_h

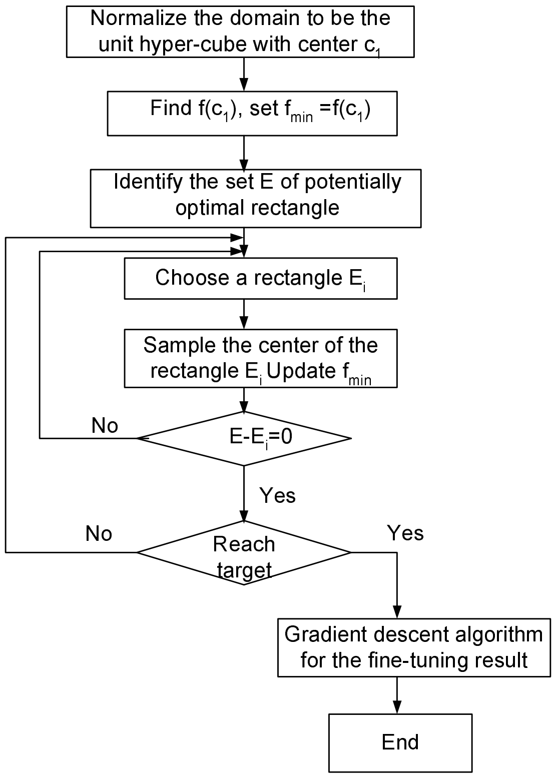

Normalize the domain to be the unit hyper-cube with center c1
Find f(c1), set fmin =f(c1)
Identify the set E of potentially optimal rectangle
Choose a rectangle Ei
Sample the center of the rectangle Ei Update fmin
No
E-Ei=0
Yes
No
Reach target
Yes
Gradient descent algorithm for the fine-tuning result
End

Fig. 8. Descriptive diagram of a combined DIRECT gradient based training algorithm.

and Q_c are the manipulated inputs. An identification experiment was carried out on this two-tank system model by using random steps in the inputs $u_1(k+i-1|k)$ and $u_2(k+i-1|k)$ over 4000 sampling points. The first 2000 sampling points data were used for training and the second 2000 sampling points data were used for validation. The sampling time is 20 s. Training was carried out by means of the DIRECT algorithm combined with the gradient descent algorithm. Fifty training runs were carried for each network size, starting from the random initial weights, and the best network in terms of mean square error for the training data was selected for each network size. The values of the parameters of the best model found were

$$\beta = \begin{bmatrix} 0.0059 & 0 \\ 0 & 0.0106 \end{bmatrix}, \tag{47}$$

$$\gamma = \begin{bmatrix} 0.0084 & 0.0030 \\ -0.0034 & 0.0038 \end{bmatrix}, \tag{48}$$

$$\omega = \begin{bmatrix} -0.0891 & -0.0783 \\ 0.1488 & 0.1850 \end{bmatrix}, \tag{49}$$

$$C = [1\ 0]. \tag{50}$$

In order to explore a reduced region of the error surface, some methods, such as Quasi-Newton ones, result in finding a local minimum from the given initial parameters. At the same time, values obtained from the DIRECT algorithm are close to different local minima. A combined algorithm starts with the DIRECT algorithm and then fine-tunes the result using a faster local search procedure. This is illustrated in Fig. 8.

Figure 9 compares the network output obtained for the training input with the training output using the DIRECT algorithm combined with the gradient descent algorithm. Figure 10 compares the network output obtained for the validation input with the validation output using the DIRECT algorithm combined with the gradient descent algorithm. After identifying the dynamic neural network model, a feedback linearization based on this DNN has being calculated. In order to improve the feedback linearization, the dynamic neural network is also used as a closed loop observer. The two-tank system yields a 2-unit network:

$$\begin{aligned} h(x) &= x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \\ g(x) &= \gamma = \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix}, \\ f(x) &= -\beta x + \omega\sigma(x). \end{aligned} \tag{51}$$

It has the relative degree of

$$r = [1\ 1]. \tag{52}$$

The control law is given by

$$\begin{aligned} u(t) = &- S(x(k+1|k))^{-1} E(x(k+1|k)) \\ &+ S(x(k+1|k))^{-1} v(t), \end{aligned} \tag{53}$$

where

$$\begin{aligned} S(x(k+1|k)) &= \begin{bmatrix} \lambda_{11} L_{g1} L_f^0 x_{1,k} & \lambda_{11} L_{g2} L_f^0 x_{1,k} \\ \lambda_{21} L_{g1} L_f^0 x_{2,k} & \lambda_{21} L_{g2} L_f^0 x_{2,k} \end{bmatrix} \\ &= \begin{bmatrix} \lambda_{11}\gamma_{11} & \lambda_{11}\gamma_{12} \\ \lambda_{21}\gamma_{21} & \lambda_{21}\gamma_{22} \end{bmatrix}, \end{aligned} \tag{54}$$

$$\begin{aligned} &E(x(k+1|k)) \\ &= \begin{bmatrix} \lambda_{10} L_f^0 x_{1,k} + \lambda_{11} L_f x_{1,k} \\ \lambda_{20} L_f^0 x_{2,k} + \lambda_{21} L_f x_{2,k} \end{bmatrix} \\ &= \left[\begin{matrix} \lambda_{10} x(k+1|k) + \lambda_{11}(-\beta_1 x(k+1|k) \\ \lambda_{20} x(k+1|k) + \lambda_{21}(-\beta_1 x(k+1|k)\sigma(x(k+1|k))) \end{matrix} \right. \\ &\qquad \left. \begin{matrix} +w_{11}\sigma(x(k+1|k)) + w_{12}\sigma(x(k+1|k))) \\ +w_{21}\sigma(x(k+1|k)) + w_{22}\sigma(x(k+1|k))) \end{matrix} \right], \end{aligned} \tag{55}$$

where $\lambda_{10} = 0.000062861$, $\lambda_{11} = 0.0165$, $\lambda_{20} = 0.000062861$, $\lambda_{21} = 0.0165$. The design parameters λ_{ij} are chosen so that $\lambda_{10} = \lambda_{20}$ and $\lambda_{11} = \lambda_{21}$, and the static gain and dominant time constant of the linearized system $v - y$ is as near to that of the Jacobian linearization of the

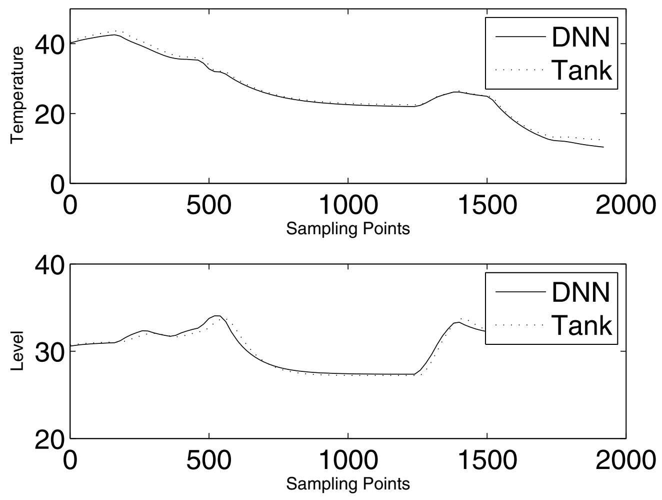


Fig. 9. Comparison of training trajectories using the DIRECT algorithm combined with the gradient descent algorithm.

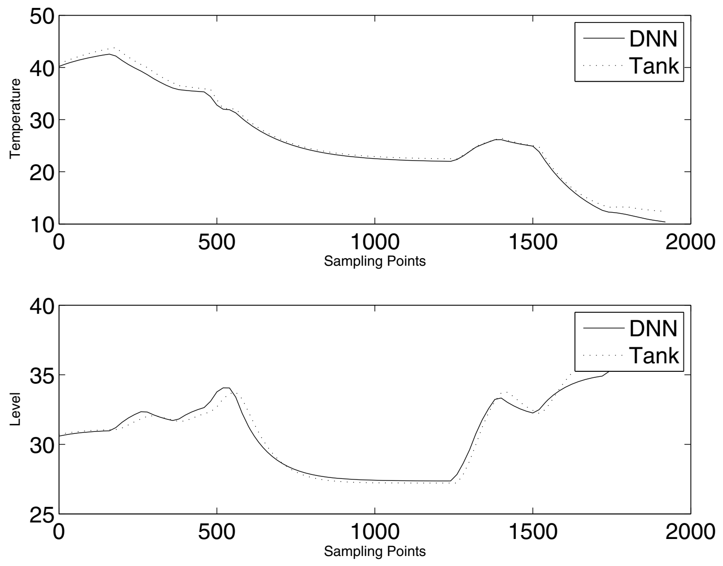


Fig. 10. Comparison of validation trajectories using the DIRECT algorithm combined with the gradient descent algorithm.

dynamic neural network model as possible. For this case, $S(x(k+1|k))$ is a constant matrix:

$$S(x(k+1|k)) = \begin{bmatrix} 0.0001394 & 0.0000485 \\ -0.0000565 & 0.0000628 \end{bmatrix}. \quad (56)$$

The input constraints are given as

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \le u \le \begin{bmatrix} 100 \\ 100 \end{bmatrix}. \quad (57)$$

After the constraint area is transformed using $S(x(k+1|k))$, the coordinates of the four corners are

$$\begin{aligned} (a_1, a_2) &= (0, 0), \\ (b_1, b_2) &= (0.004983, 0.0062799), \\ (c_1, c_2) &= (0.01889, 0.00063129), \\ (d_1, d_2) &= (0.0139387, -0.0056489). \end{aligned} \quad (58)$$

Therefore,

$$(a'_1, a'_2) = (0, 0),$$

$$\begin{aligned}(b_1', b_2') &= (0, 0.0079984),\\ (c_1', c_2') &= (0.01504, 0.0079984),\\ (d_1', d_2') &= (0.01504, 0).\end{aligned} \tag{59}$$

According to Eqn. (23),

$$H = \begin{bmatrix} 0.81764 & -0.64492 \\ 0.391138 & 0.96513 \end{bmatrix}. \tag{60}$$

In this example, $S(x(k+1|k))$ and H are constant matrices. However, if $S(x(k+1|k))$ and H are state-dependent, the proposed method for handling input constraints within an MPC/FL scheme remains applicable, only that the constraint values change at every sampling time.

The Kalman filter gain was computed using the linearized model and the covarience matrices $Q_f = \mathrm{diag}(1,1)$, $R_f = \mathrm{diag}(0.001, 0.001)$. The references used in the simulation were two square waves with different frequencies. The sampling time is $T_s = 20$ s, the control horizon is $m = 2$, the prediction horizon is $p = 320$, the input and output weights are $uwt = [1,\ 1]^T$, $ywt = [0.0001,\ 0.0001]^T$, the manipulated variables constraints are $u_{\min} = [0,\ 0]^T$, $u_{\max} = [100,\ 100]^T$, the incremental constraints on the manipulated variable are $\Delta u_{\max} = [0.01,\ 0.01]^T$. The tuning parameters of the extended Kalman filter are $P_0 = \mathrm{diag}(0.1,\ 0.1)$, $Q = \mathrm{diag}(0.1,\ 0.1)$, $R = \mathrm{diag}(100,\ 100)$. The diagonal elements of the output noise covariance matrix R were chosen to account for the covariance of the measurement noise. The diagonal elements of the process noise covariance matrix Q were chosen small and positive in order to prevent the covariance of the state error from becoming zero. The initial covariance of the state error P_0 is chosen to reflect some uncertainty in the initial state estimate.

Figure 11 shows the outputs responses under the MPC/FL scheme. Figure 12 shows the actual inputs to the process under the MPC/FL control method. In this case, the input magnitude constraints do not become active. Figures 13 and 14 show the output of the controller and input to the process under the proposed scheme in a scenario where one of the input magnitude constraints becomes active as can be seen in Fig. 13.

8. Conclusion

This paper has introduced a technique for handling input constraints within a real time MPC/FL scheme, where the plant model employed is a class of dynamic neural network. Handling input constraints within such schemes is difficult since simple bound contraints on the input become state dependent because of the nonlinear transformation introduced by feedback linearization. The technique is based on a simple affine transformation of the feasible area. A simulated case study was presented to illustrate the use of the technique. The issue of recursive feasiblity of MPC is important (Rossiter, 2003) in the context of the work presented in this paper. This is currently being investigated by the authors and will be the subject of a future paper.

Acknowledgment

This project is co-funded by the UK Technology Strategy Board's Collaborative Research and Development programme, following an open competition. The authors would like to thank the Technology Strategy Board Program under Grant Reference No. TP/3/DSM/6/I/152. Thanks also are given to the UK Engineering and Physical Science Research Council under Grant Reference No. GR/R64193 and the University of Reading Chinese Studentship Programme for their support.

References

Ayala-Botto, M., Boom, T. V. D., Krijgsman, A. and da Costa, J. S. (1999). Predictive control based on neural network models with I/O feedback linearization, *International Journal of Control* **72**(17): 1538–1554.

Ayala-Botto, M., Braake, H. T., da Costa, J. S. and Verbruggen, H. (1996). Constrained nonlinear predictive control based on input-output linearization using a neural network, *Proceedings of the 13-th IFAC World Congress, San Francisco, CA, USA*, pp. 175–180.

Becerra, V. M., Roberts, P. D. and Griffiths, G. W. (2001). Applying the extended Kalman filter to systems desribed by nonlinear differential-algebraic equations, *Control Engineering Practice* **9**(3): 267–281.

Casavola, A. and Mosca, E. (1996). Reference governor for constrained uncertain linear systems subject to bounded input disturbances, *Preceedings of the 35-th Conference on Decision and Control, Kobe, Japan*, pp. 3531–3536.

Del-Re, L., Chapuis, J. and Nevistic, V. (1993). Stability of neural net based model predivtive control, *Proceedings of the 32-nd Conference on Decision and Control, San Antonio, TX, USA*, pp. 2984–2989.

Deng, J. and Becerra, V. M. (2004). Real-time constrained predictive control of a 3d crane system, *Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics, Singapore*, pp. 583–587.

Garces, F. (2000). *Dynamic Neural Networks for Approximate Input-Output Linearisation-Decoupling of Dynamic Systems*, Ph.D. thesis, University of Reading.

Garces, F., Becerra, V., Kambhampati, C. and Warwick, K. (2003). *Strategies for Feedback Linearisation: A Dynamic Neural Network Approach*, Springer, London.

Guemghar, K., Srinivasan, B., Mullhaupt, P. and Bonvin, D. (2005). Analysis of cascade structure with predictive control and feedback linearisation, *IEE Proceedings: Control Theory and Applications* **152**(3): 317–324.

Henson, M. A. and Kurtz, M. J. (1994). Input-output linearisation of constrained nonlinear processes, *Nonlinear Control, AICHE Annual Meeting, San Franciso, CA, USA*, pp. 1–20.

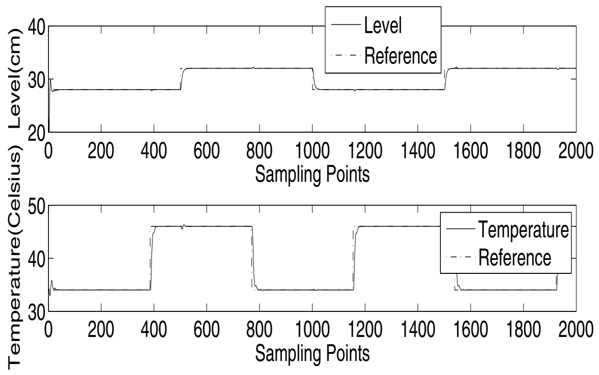


Fig. 11. Mixing process outputs under the MPC/FL method. The reference signals are square waves.

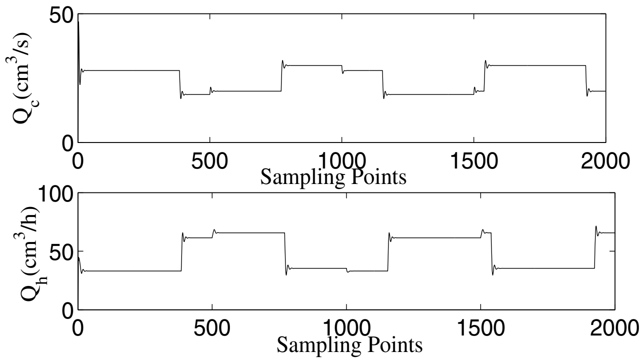


Fig. 12. Inputs applied to the mixing process the MPC/FL method.

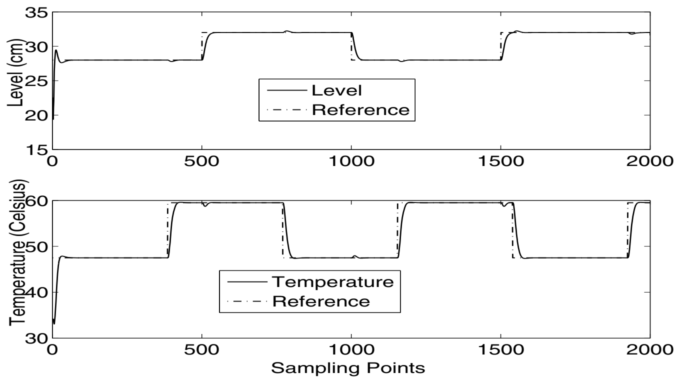


Fig. 13. Mixing process outputs under the MPC/FL control method.

Henson, M. A. and Seborg, D. E. (1993). Theoretical analysis of unconstrained nonlinear model predictive control, *International Journal of Control* **58**(5): 1053–1080.

Isidori, A. (1995). *Nonlinear Control Systems, 2nd Edition*, Springer, Berlin/New York, NY.

Kurtz, M. and Henson, M. (1997). Input-output linearizing control of constrained nonlinear processes, *Journal of Process Control* **7**(1): 3–17.

Maciejowski, J. M. (2002). *Predictive Control with Constraints*, Prentice Hall, London.

Maybeck, P. S. (1982). *Stochastic Models, Estimation and Control*, Academic Press, New York, NY.

Nevistic, V. (1994). *Feasible Suboptimal Model Predictive Control for Linear Plants with State Dependent Constraints*, Postdiploma thesis, Swiss Federal Institute of Technology, Automatica Control Laboratory, ETH, Zurich.

Nevistic, V. and Morari, M. (1995). Constrained control of feedback-linearizable systems, *Proceedings of the European Control Conference, Rome, Italy,* pp. 1726–1731.

Nevistic, V. and Primbs, J. A. (1996). Model predictive con-

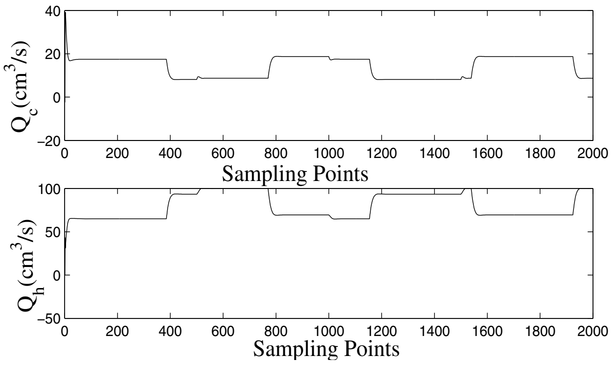


Fig. 14. Inputs applied to the mixing process under the MPC/FL control method.

trol: Breaking through constraints, *Proceedings of the 35-th Conference on Decision and Control, Kobe, Japan*, pp. 3932–3937.

Oliveiria, S. D., Nevistic, V. and Morari, M. (1995). Control of nonlinear systems subject to input constraints, *Preprints of the IFAC Symposium on Nonlinear Control Systems, NOLCOS'95, Tahoe City, CA, USA*, Vol. 1, pp. 15–20.

Perttunen, C. D., Jones, D. R. and Stuckman, B. E. (1993). Lipschitzian optimization without the Lipschitz constant, *Journal of Optimization Theory and Application* **79**(1): 157–181.

Poznyak, A. S., Sanchez, E. N. and Yu, W. (2001). *Differential Neural Networks for Robust Nonlinear Control*, World Scientific, Singapore.

Rossiter, J. A. (2003). *Model Based Predictive Control: A Practical Approach*, CRC Press, Boca Raton, FL, USA.

Scattolini, R. and Colaneri, P. (2007). Hierarchical model predictive control, *Proceedings of the 46-th IEEE Conference on Decision and Control, New Orleans, LA, USA*, pp. 4803–4808.

van den Boom, T. (1997). Robust nonlinear predictive control using feedback linearization and linear matrix inequalities, *Proceedings of the American Control Conference, Albuquerque, NM, USA*, pp. 3068–3072.

Jiamei Deng received the Ph.D. degree in cybernetics from the University of Reading in 2005, the B.Eng. degree in electrical engineering from the Hua Zhong University of Science and Technology, Wuhan, China, in 1988, and the M.Eng. degree in mechanical engineering from the Shanghai Institute of Mechanical Engineering, China, in 1991. She currently works as a research associate at Loughborough University. She was a research fellow at the University of Sussex between 2006 and 2007. Her research interests include predictive control, nonlinear control, sliding mode control, optimal control, system identification, optimization, process control, power control, black-box modeling, automotive control, and energy systems.

Victor M. Becerra obtained his Ph.D. degree in control engineering from City University, London, in 1994, for the development of novel optimal control algorithms, a B.Eng. degree in electrical engineering (Cum Laude) from Simon Bolivar University, Caracas, Venezuela, in 1990, and an M.Sc. degree in financial management from Middesex University, London, in 2001. He worked in power systems analysis for CVG Edelca, Caracas, between 1989 and 1991. He was a research fellow at City University between 1994 and 1999, before joining the University of Reading, UK, as a lecturer in 2000, where he is currently a reader in cybernetics. His main research areas include optimal control, nonlinear control, intelligent control, and system identification. Dr. Becerra has published over 100 papers in international journals and conferences, and he has co-authored a Springer book on neural network based control. Since 2003 he has been the head of the Cybernetics Intelligence Research Group at the University of Reading. He is a member of the Technical Committees on Optimal Control as well as Adaptive Control and Learning of the International Federation of Automatic Control (IFAC). He is a member of the IASTED technical committee on robotics. He is also a Senior Member of the IEEE, a member of the IET, and a Chartered Engineer.

Richard Stobart is a Ford professor of automotive engineering at Loughborough University in the Department of Aeronautical and Automotive Engineering. As a professor, he is responsible for the development of research and education in the field of vehicle engineering, with a particular focus on propulsion and power generation systems. In 2007 he completed his term as the chair of the Powertrain, Fuels and Lubricants Activity of the US based Society of Automotive Engineers (SAE) and was a programme co-chair (with Prof. Li of Tongji University) for the 2008 SAE Congress on Powertrain Fuels and Lubricants in Shanghai in 2008. He was a professor of automotive engineering at the University of Sussex (Brighton, UK) from 2001 to 2007, where he was also the head of Engineering and Design from 2003 to 2006. Professor Stobart was a member of the team who in 1997 received Arthur D Little's Ketteringham prize for their work in developing fuel cell technology. He is a graduate from the University of Cambridge with a first class honours degree in mechanical engineering. He was elected a Fellow of the Institution of Mechanical Engineers in 2000.

Received: 12 May 2008
Revised: 11 November 2008
Re-revised: 18 November 2008