Int. J. Appl. Math. Comput. Sci., 2006, Vol. 16, No. 1, 7–26

SOFT COMPUTING IN MODEL–BASED PREDICTIVE CONTROL[†]

PIOTR TATJEWSKI, MACIEJ ŁAWRYŃCZUK

Institute of Control and Computation Engineering
Faculty of Electronics and Information Technology
Warsaw University of Technology
ul. Nowowiejska 15/19, 00–665 Warsaw, Poland
e-mail: {P.Tatjewski,M.Lawrynczuk}@ia.pw.edu.pl

The application of fuzzy reasoning techniques and neural network structures to model-based predictive control (MPC) is studied. First, basic structures of MPC algorithms are reviewed. Then, applications of fuzzy systems of the Takagi-Sugeno type in explicit and numerical nonlinear MPC algorithms are presented. Next, many techniques using neural network modeling to improve structural or computational properties of MPC algorithms are presented and discussed, from a neural network model of a process in standard MPC structures to modeling parts or entire MPC controllers with neural networks. Finally, a simulation example and conclusions are given.

Keywords: process control, model predictive control, nonlinear systems, fuzzy systems, neural networks

1. Introduction

The Model-Based Predictive Control (MPC) technique is the only one among *advanced control techniques* – usually defined as techniques more advanced than the well-known PID control – which was tremendously successful in practical applications, exerting great influence on the directions of the development of industrial control systems as well as research conducted in this area (Allgöwer *et al.*, 1999; Brdyś and Tatjewski, 2005; Eder, 1999; Maciejowski, 2002; Mayne *et al.*, 2000; Morari and Lee, 1999; Qin and Badgwell, 2003; Rossiter, 2003; Tatjewski, 2002). There are several reasons for this success. First, predictive control algorithms can take into account in a natural way constraints on both process inputs (control signals) and process output values (controlled variables), which often decide on the quality, effectiveness and safety of production. They generate control signals taking on-line into account these constraints and, owing to a direct use of a model, also internal interactions within the process. Thus, they can naturally be applied to multivariable process control, also when the numbers of the control inputs and the controlled variables differ. Third, the principle of operation of these algorithms is comprehensible and relatively easy to explain to engineering and operator staff, which is a very important aspect when introducing new techniques into industrial practice.

[†] The work was supported by a statutory research project.

2. Model-Based Predictive Control

2.1. Predictive Control Principle

The *general principle of predictive control* is the following: At each consecutive sampling instant k, the control inputs $u(k) = u(k|k), u(k+1|k), \ldots, u(k+N_u-1|k)$ are calculated, assuming $u(k+p|k) = u(k+N_u-1|k)$ for $p \geq N_u$, where N_u is the *control horizon*. The applied notation '$u(k+p|k)$' means the prediction of the control input value for the future time $k+p$, performed at the time k. The control inputs are calculated in such a way as to minimize differences between the predicted controlled outputs $y(k+p|k)$ and the foreseen set points for these outputs $y^{sp}(k+p|k)$ over the *prediction horizon* N ($p = 1, 2, \ldots, N$). Then, only the first element $u(k|k)$ of the calculated sequence is applied to the process, i.e., $u(k) = u(k|k)$. At the next sample $(k+1)$, there occurs a new measurement of the process outputs and the whole procedure is repeated, with the prediction horizon of the same length N, but shifted by one step forward (the principle of a *receding horizon*, also called the *repetitive control* principle (e.g., Findeisen, 1997; Findeisen *et al.*, 1980)). This principle is presented in Fig. 1, for a SISO (single-input single-output) process.

A model of the process used in the MPC algorithm is usually only an approximation of reality. Moreover, there is some uncertainty in the uncontrolled inputs, which can be inaccurately measured or not measured at all. Therefore, the output predictions usually differ from the (later) measured values. This fact is depicted in Fig. 1 as a dis-

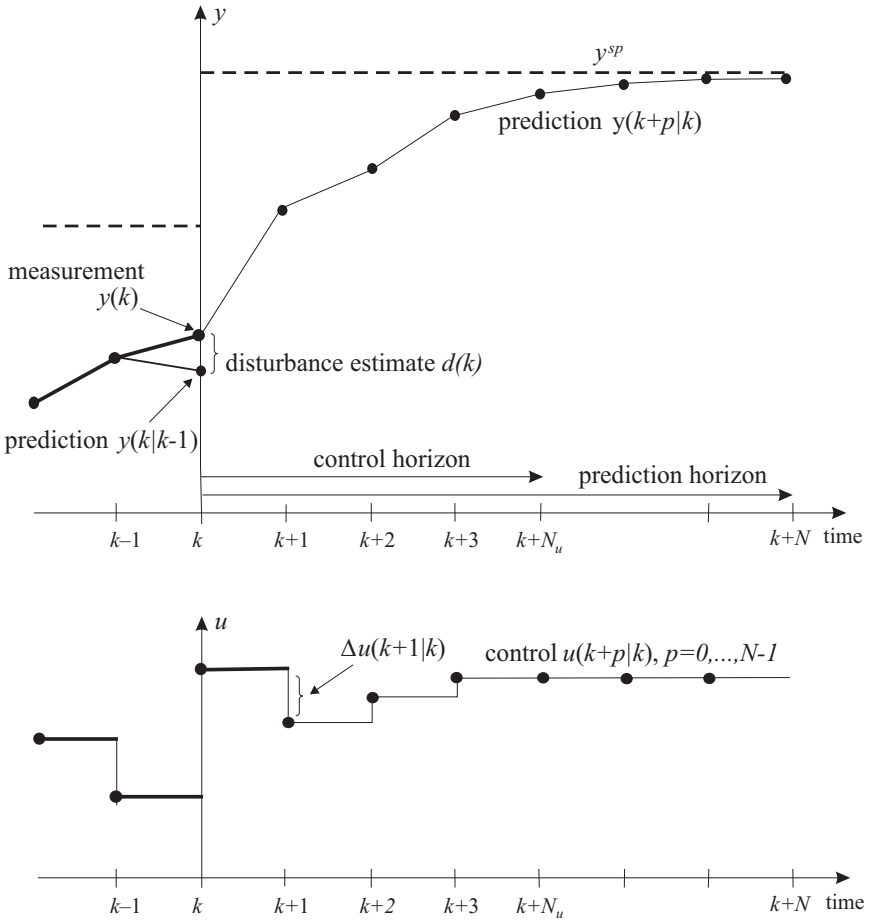

Fig. 1. Principle of predictive control.

turbance $d(k)$, $d(k) = y(k) - y(k|k-1)$, occurring at the process output at sampling instant k, where $y(k|k-1)$ is the output value predicted for the sample k at the previous sample $(k-1)$. The dependence of the trajectory of the process output predicted at sampling instant k on the currently measured value $y(k)$, and not on the value $y(k|k-1)$, signifies the application of a *discrete feedback* in the control system.

Determination of the control input trajectory over the control horizon is realized in the predictive algorithms on the basis of a model, by minimizing a cost function describing the control quality over the prediction horizon. A prime component of this function is the *cost of deviations of the predicted outputs from the set points*, i.e., the cost of predicted control errors. Moreover, it is also typical to include into the cost function penalties on control changes. As regards the two components, the following most commonly used *quadratic cost function* can be formulated (see, e.g., (Maciejowski, 2002; Tatjewski, 2002) for more general or alternative formulations):

$$J(k) = \sum_{p=N_1}^{N} \left\| y^{sp}(k+p|k) - y(k+p|k) \right\|^2 + \lambda \sum_{p=0}^{N_u-1} \left\| \triangle u(k+p|k) \right\|^2 , \quad (1)$$

where the vectors $y^{sp}(k+p|k)$ and $y(k+p|k)$ are of dimensionality $n_y = \dim y$ equal to the number of the controlled outputs, while the vector of control increments $\triangle u(k+p|k)$ is of dimensionality $n_u = \dim u$ and $1 \leq N_1 \leq N$ ($N_1 > 1$ is reasonable if there is a delay causing no reaction of the outputs at the sampling instants $k+1, \ldots, k+N_1-1$ to the change in the control input at the sampling instant k). The length of the control horizon N_u must satisfy the constraints $0 < N_u \leq N$. It is usually assumed that $N_u < N$, as this results in a decreased dimensionality of the optimization problem, and

thus leads to a smaller computational load. The role of the coefficient λ is to scale the second sum of squared control increments against the first sum representing squared predicted control errors.

The optimal trajectory of control inputs is evaluated, at every sampling instant, by the minimization of the cost function (1) subject to several constraints:

$$\min_{\triangle u(k|k),\ldots,\triangle u(k+N_u-1|k)} \left\{ \sum_{p=N_1}^{N} \left\| \left[y^{sp}(k+p|k) - y(k+p|k) \right] \right\|^2 + \lambda \sum_{p=0}^{N_u-1} \left\| \triangle u(k+p|k) \right\|^2 \right\} \quad (2)$$

subject to

$$\begin{aligned} &y(k+p|k) \in Y, && p = 1,\ldots,N,\\ &u(k+p|k) \in U, && p = 0,\ldots,N_u-1,\\ &u(k+p|k) = u(k+N_u-1|k), && p = N_u,\ldots,N-1,\\ &\triangle u(k+p|k) = u(k+p|k) - u(k+p-1|k), \end{aligned}$$

where the values of predicted outputs are calculated using a *dynamic process model*, in general, a *nonlinear* one. Assuming that it is in the input-output form, the following formulation is fairly general:

$$\begin{aligned} y(k+p|k) = f_p\big(&u(k+p-1|k),\ldots,\\ &u(k|k),u(k-1),\ldots,u(k-n_B),y(k),y(k-1),\\ &\ldots,y(k-n_A),d(k)\big), \quad p=1,\ldots,N. \end{aligned} \quad (3)$$

The class that the process model belongs to is critical, as model equations are equality constraints in the MPC optimization problem with the quadratic cost function. Therefore, nonlinear models result in nonquadratic, generally nonconvex optimization problems – thus creating serious problems for on-line applications, as required in MPC algorithms. A classification of basic classes of MPC algorithms is presented in Fig. 2. It should be treated as a rather simplified one, i.e., many classes of nonlinear optimal MPC algorithms (with optimization using nonlinear process models) can be further distinguished, but this is beyond the scope of this paper.

2.2. MPC with Linear Process Models

So far, MPC algorithms with *linear process models* have been of the greatest importance. First of all, the range of direct applications of these algorithms is fairly wide. Secondly, they constitute a basis for the construction of relatively simple and efficient nonlinear algorithms using linearizations of nonlinear models.

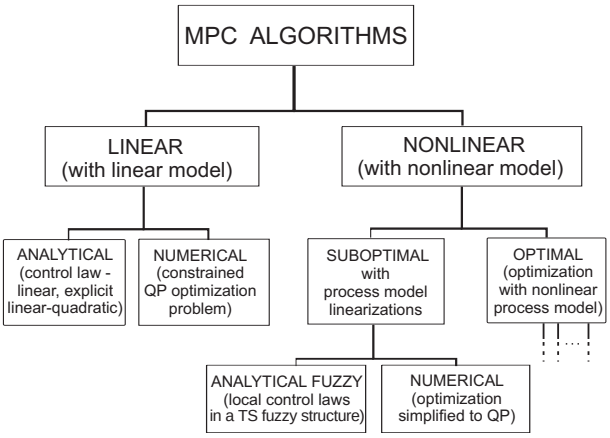

Fig. 2. General classification of basic MPC algorithms.

In a linear case, by applying the principle of superposition it is possible to present the trajectory of the predicted outputs $y(k+p|k)$ in the form of a sum of a *free trajectory* $y^0(k+p|k)$ dependent only on the realized (past) process inputs, and a trajectory $\triangle y(k+p|k)$ dependent only on the decision variables (current and future control inputs $u(k+p|k)$), called the *forced output trajectory*. Thus we have

$$y(k+p|k) = y^0(k+p|k) + \triangle y(k+p|k), \quad p = N_1,\ldots,N.$$

The above partition is convenient, though not necessary for the realization of a predictive control algorithm, because the values $y^0(k+p|k)$, as dependent only on the past of the process, are calculated at a given sampling instant k *only once*, and remain then as *fixed parameters* in the following optimization of the control moves. On the other hand, the increments $\triangle y(k+p|k)$, as dependent on the current and future control moves $\triangle u(k+p|k)$, are calculated *many times* in the process of numerical optimization. As regards the presented decomposition, the cost function (1) can be written in the form

$$J(k) = \sum_{p=N_1}^{N} \left\| \left[y^{sp}(k+p|k) - y^0(k+p|k) \right] - \triangle y(k+p|k) \right\|^2 + \lambda \sum_{p=0}^{N_u-1} \left\| \triangle u(k+p|k) \right\|^2, \quad (4)$$

where free and forced output trajectories are given by the linear models

$$y^0(k+p|k) = y(k) + \mathbf{g_p}\left[u(k-1)\ \cdots\ u(k-n_B)\right]^T + \mathbf{h_p}\left[y(k-1)\ \cdots\ y(k-n_A)\right]^T, \quad (5)$$

$$\triangle y(k+p|k) = \mathbf{m_p}\left[\triangle u(k|k)\ \triangle u(k+1|k)\ \cdots \triangle u(k+p-1|k)\right]^T,\ p = N_1,\ldots,N, \quad (6)$$

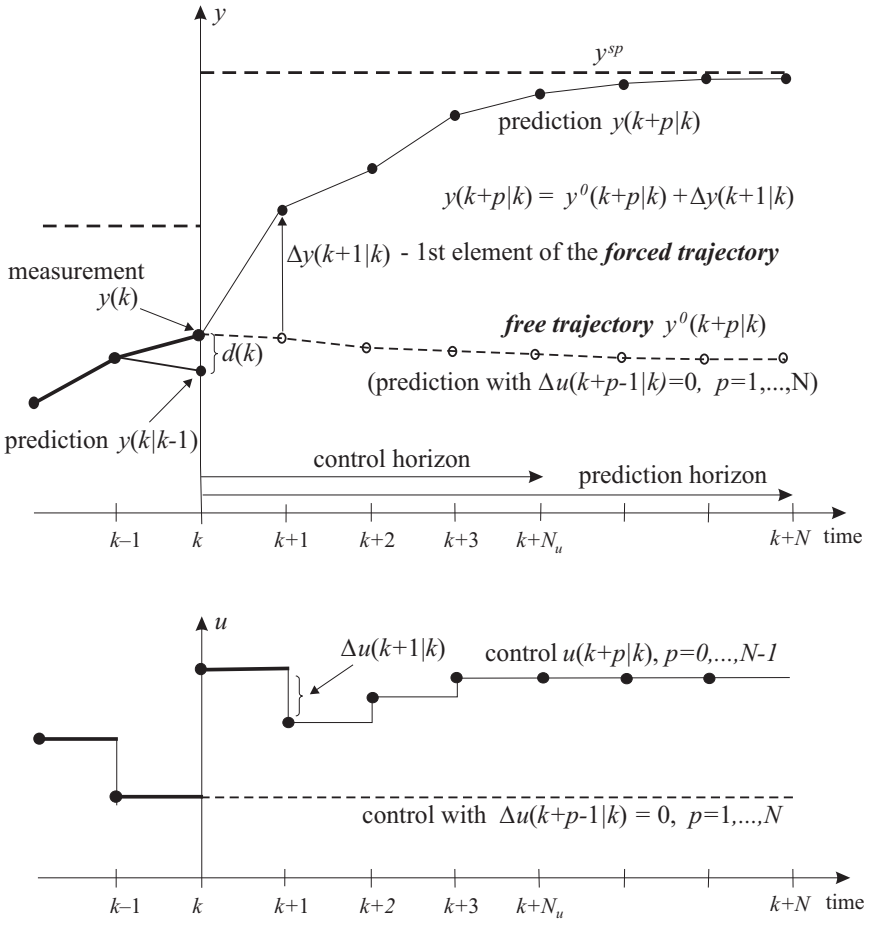

Fig. 3. Predictive control with a linear process model.

(this is a SISO case, in MIMO cases row vectors $\mathbf{g_p}$, $\mathbf{h_p}$ and $\mathbf{m_p}$ become matrices). The principle of predictive control for a linear process model is shown in Fig. 3 (compare it with Fig. 1).

Owing to the linearity of the model, the optimization problem (2) is a *strictly convex quadratic programming problem*. Therefore, it has a unique, global minimum which can be fast and reliably computed numerically in a constrained case. In a constraint-free case the solution can be computed analytically as a *linear feedback control law*, i.e., in the following general form (for the set point y^{sp} constant over the prediction trajectory):

$$\begin{aligned}\triangle u(k) &= \triangle u(k|k) \\ &= k^e\big(y^{sp}(k) - y(k)\big) \\ &\quad + (\mathbf{k}^u)^T\big[u(k-1)\ \cdots\ u(k-n_B)\big]^T \\ &\quad + (\mathbf{k}^y)^T\big[y(k)\ \cdots\ y(k-n_A)\big]^T, \qquad (7)\end{aligned}$$

where $\mathbf{k}^u$ and $\mathbf{k}^y$ are the (column) vectors of feedback coefficients (SISO case).

We shall describe MPC algorithms formulated as control laws given by an *explicit analytical formula* (or a set of formulas) as *explicit MPC algorithms*. This class of algorithms consists of two basic groups:

- simple, unconstrained algorithms given by constraint-free control laws (7),
- more sophisticated, constrained algorithms given by a number of constrained linear control laws, each valid for a different set of active constraints, and a switching mechanism (originally defined as *explicit linear quadratic algorithms for constrained systems* (Bemporad *et al.*, 2002)) – they are in fact nonlinear algorithms.

On the other hand, the MPC algorithms solving numerically, at each sampling instant, constrained QP problems will be described as *numerical MPC algorithms* (see also

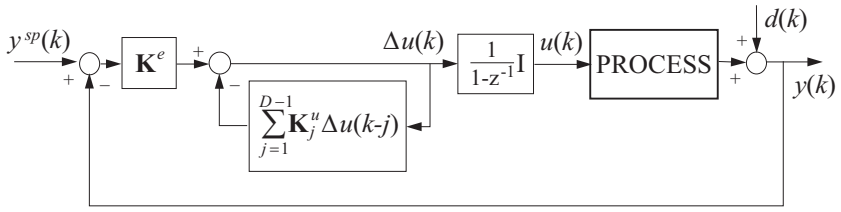

Fig. 4. Structure of the DMC control law.

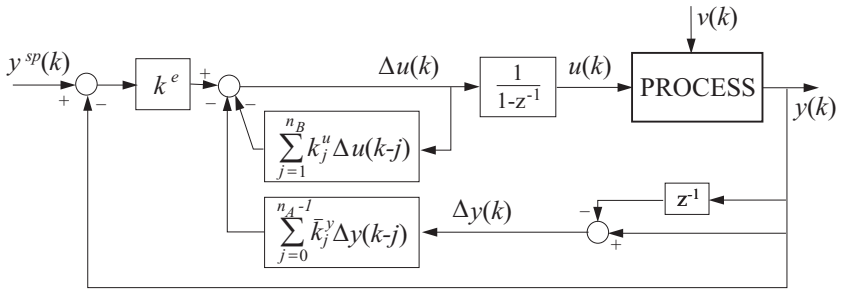

Fig. 5. Structure of the GPC control law.

Fig. 2). Unconstrained linear feedback control laws for two most popular MPC algorithms using input-output type process models will be briefly shown as examples.

When the process model is in the form of a discrete step response, then the linear MPC algorithm is the well-known, most popular in process control applications DMC (*Dynamic Matrix Control*) algorithm (Cutler and Ramaker, 1979; Maciejowski, 2002; Rossiter, 2003; Tatjewski, 2002). The feedback control law (7) takes then the following form of an *explicit DMC algorithm*:

$$\begin{aligned}\triangle u(k) &= k^e\big(y^{sp}(k) - y(k)\big)\\ &\quad +(\mathbf{k}^u)^T\big[\triangle u(k-1)\cdots\triangle u(k-n_B+1)\big]^T,\end{aligned} \quad (8)$$

where $n_B = D$ denotes the process dynamics horizon, i.e., the number of step response coefficients s_i present until their values become constant, ($s_i = \text{const}$ for $i \geq D$). The structure of the control law (8) is shown in Fig. 4.

When the process model is in the form of a discrete transfer function or, equivalently, the difference equation (ARX-type model):

$$y(k) = -\sum_{j=1}^{n_A} a_j y(k-j) + \sum_{j=0}^{n_B} b_j u(k-j-1) + d(k), \quad (9)$$

then we obtain the popular GPC (*Generalized Predictive Control*) algorithm (Camacho and Bordons, 1999; Clarke *et al.*, 1987; Maciejowski, 2002; Rossiter, 2003; Tatjewski, 2002). The feedback control law (7) can then be formulated in the following form (Tatjewski, 2002):

$$\begin{aligned}\triangle u(k) &= k^e\big(y^{sp}(k) - y(k)\big)\\ &\quad +(\mathbf{k}^u)^T\big[\triangle u(k-1)\ \cdots\ \triangle u(k-n_B)\big]^T\\ &\quad +(\mathbf{k}^y)^T\big[\triangle y(k)\cdots\triangle y(k-n_A+1)\big]^T,\end{aligned} \quad (10)$$

illustrated in Fig. 5. By comparison with Fig. 4, it can easily be seen that n_A measured past outputs are added as additional feedback information, but there are fairly fewer feedback instances from past inputs (usually $n_B \ll D$).

The explicit MPC algorithm (control law; DMC, GPC or other) can be easily implemented, as defined by the appropriate formula, if there is no risk that the generated control signal will exceed limits defined by physical possibilities of the process actuator, i.e., if constraints on the process input are always inactive, as they have been ignored when constructing the laws. However, this is often not the case and the *constraints must be taken into account* in order to get practically applicable controllers. This can be done effectively only if the following two principles hold:

- the past control values $\triangle u(k-j)$ which are fed back into the controller structure must be those corresponding to the *process past constrained inputs*, not the past values generated by the constraint-free control law (7),
- an *anti-windup scheme* must be added.

This is illustrated for a case of the DMC algorithm in Fig. 6, where models of input constraints (rate of change

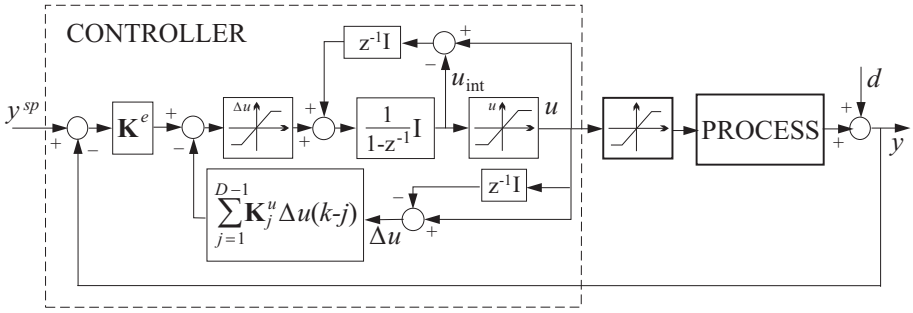

Fig. 6. Structure of the explicit DMC controller taking into account constraints on the rate of change and amplitude of the controller output, with an additional anti-windup scheme (upper feedback loop).

and amplitude saturations) are elements of the controller structure *within* its inner feedback loop, and an additional anti-wind up feedback loop has been added (compare this with Fig. 4). In (Tatjewski, 2002), simulation examples are presented comparing the results of MPC control with an explicit controller in structures where input constraints are and are not taken into account. These results show that implementation as in Fig. 4 leads to drastically decreased control quality (or even instability) when input constraints are active, while implementation in the structure as in Fig. 6 works usually very well, often comparably to numerical MPC implementation (with constrained optimization at each sampling instant). A further discussion of this point is beyond the scope of the paper, and the reader is referred to the cited reference for more details.

Obviously, explicit MPC algorithms can be recommended in constraint-free cases, also when input constraints can become active, but a simple and fast (short sampling periods) controller is needed – in this case controller implementation must be augmented taking into account the constraints, as has just been discussed. When implementing numerical MPC algorithms with linear models, significantly more powerful controllers are required, especially for shorter sampling periods, but the control is optimal (in the sense of the applied cost function).

There is no need for soft computing methods in MPC algorithms with linear models. But these algorithms and their philosophy are basic for more involved MPC algorithms with nonlinear plant models, when soft computing methods play an important role.

3. Soft Computing in Nonlinear MPC Algorithms

The general formulation of the MPC optimization problem has been previously defined as (2), where predictions of controlled outputs are calculated using an input-output process model (3). When this model is nonlinear, then the optimization problem (2) is certainly not linear-quadratic; it is generally a nonconvex and even multimodal one. For such problems there are no sufficiently fast and reliable numerical optimization procedures, i.e., procedures yielding always an optimal point and within predefined time limit – as is required in on-line control applications. Therefore, many attempts have been made to construct simplified (and generally suboptimal) nonlinear MPC algorithms avoiding full on-line nonlinear optimization, first of all using model linearizations or multiple linear models in fuzzy structures, see Fig. 2. The resulting predictive control algorithms apply soft computing techniques to a great extent. On the other hand, there are also many designs of predictive algorithms based on nonlinear optimization and also using soft computing, mainly those applying artificial neural network techniques.

Therefore, the presentation of predictive control algorithms using soft computing techniques will be done within the following groups:

- MPC algorithms using fuzzy reasoning:
  - Multi-model explicit algorithms in the fuzzy Takagi-Sugeno (TS) structure (Babuška *et al.*, 1999; Marusak and Tatjewski, 2002; Tatjewski, 2002),
  - Algorithms with on-line linearization of a fuzzy TS model (usually with the same structure of all fuzzy rule consequents) and QP optimization (Babuška *et al.*, 1999; Tatjewski, 2002).
- MPC algorithms using artificial neural networks:
  - Algorithms with nonlinear optimization and a neural network process model or a neural network prediction model (Ławryńczuk, 2003; Ławryńczuk and Tatjewski, 2001a; 2001b; 2004; Najim *et al.*, 1997; Nørgaard *et al.*, 2000; Ortega and Camacho, 1996; Temeng *et al.*, 1995; Trajanoski and Wach, 1998; Yu and Gomm, 2003),

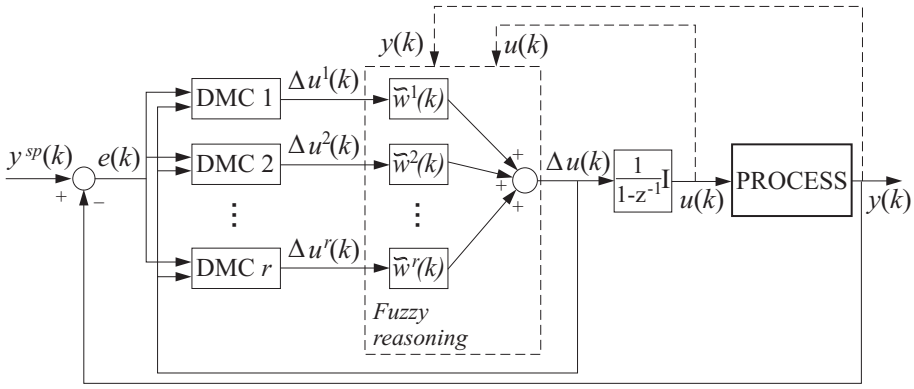

Fig. 7. Structure of the explicit (unconstrained) fuzzy DMC controller.

– Algorithms with on-line linearization of a neural network model and QP optimization (Ławryńczuk, 2003; Ławryńczuk and Tatjewski, 2002; 2004; Nørgaard *et al.*, 2000),

– Neural network modeling applied to reduce computational complexity and to approximate the controller (optimizer) (Liu and Daley, 1999; Liu *et al.*, 1998; Parisini and Sacone, 2001; Parisini *et al.*, 1998; Parisini and Zoppoli, 1995; Piche *et al.*, 2000; Vila and Wagner, 2003; Wang and Wan, 2001).

4. MPC Algorithms Using Fuzzy Reasoning Techniques

4.1. Multimodel Explicit Algorithms in the Fuzzy TS Structure

A simple and effective way to generalize a linear explicit MPC algorithm (a linear control law) to the case of a nonlinear process model is to apply fuzzy reasoning techniques. In particular, fuzzy systems with the Takagi-Sugeno (TS) structure are most suitable for the design of fuzzy controllers (Tatjewski, 2002). This design can be divided into the following stages:

1. Design a TS fuzzy model of the nonlinear process considered, i.e., with fuzzy rules with functional consequents which are linear dynamic models of the process in particular local subdomains. The range of variability of the input and output signals should cover the entire operation domain of the designed controller:

   (a) Define variables occurring in rule antecedents. The values of these variables will describe the membership of the current state (operating point) of the process to *local subdomains*, in which the process can be approximated by linear models, for control purposes.

   (b) Divide the range of variability of each variable of rule antecedents into overlapping fuzzy sets (in relation to process non-linearity), defining the number of these sets and assigning the shape and values of the parameters of the membership function of each set. This creates a division of the whole domain into a number of overlapping subdomains (multidimensional fuzzy subsets).

   (c) Formulate a linear process model for each subdomain constructed so far (each subdomain is defined by one fuzzy rule).

2. For each linear model (submodel) of the process, *design an unconstrained explicit MPC controller* operating correctly in the assigned subdomain. Create a TS fuzzy MPC controller – a TS fuzzy system with rule antecedents created in the previous point (division into fuzzy sets as in the TS fuzzy model of the process) and functional consequents which are the designed explicit MPC control algorithms.

3. Perform the *analysis* of the obtained TS fuzzy algorithm. If the results are not satisfactory, return to the previous points and correct the design of the TS fuzzy process model and/or local linear MPC controllers.

The resulting fuzzy MPC controller structure, in the case of r fuzzy rules (r subdomains) with local DMC control laws as rule consequents, is depicted in Fig. 7, where $\tilde{w}^i(k)$ denotes the normalized activation level of the i-th fuzzy rule at the k-th sampling instant, $i = 1, \ldots, r$.

The presented approach leads to a very simple controller implementation if each local process model and

thus each local MPC controller are precisely of the same structure (have the same number of parameters). For example, if each of r explicit DMC controllers is of the form (8)

$$\begin{aligned}\triangle u^i(k) &= k_i^e\big(y^{sp}(k)-y(k)\big)\\ &\quad+(\mathbf{k}_i^u)^T\big[\triangle u(k-1)\ \cdots\ \triangle u(k-n_B+1)\big]^T,\\ &\quad i=1,\ldots,r,\end{aligned}\tag{11}$$

then fuzzy reasoning results in the following fuzzy controller output:

$$\begin{aligned}\triangle u(k) &= \sum_{i=1}^{r}\tilde{w}^i(k)\triangle u^i(k)\\ &= \Big[\sum_{i=1}^{r}\tilde{w}^i(k)k_i^e\Big]\big(y^{sp}(k)-y(k)\big)\\ &\quad+\Big[\sum_{i=1}^{r}\tilde{w}^i(k)(\mathbf{k}_i^u)^T\Big]\\ &\quad\times\big[\triangle u(k-1)\ \cdots\ \triangle u(k-n_B+1)\big]^T,\end{aligned}\tag{12}$$

that is, the nonlinear fuzzy controller equation is structurally identical to the linear DMC equation (8), and only the values of gain coefficients (given in square brackets in (12)) are time-varying, being time-varying weighted sums of local controller gains.

When constraints on process inputs are to be taken into account, then the structure of the presented fuzzy TS predictive controller, as shown in Fig. 7 or in a simplified version following from (12), should be modified exactly in the same way as the structure from Fig. 4 was modified to obtain that of Fig. 6.

4.2. Algorithms with On-line Linearization of Fuzzy TS Models

The explicit fuzzy TS MPC controller presented in the previous section was designed as an unconstrained controller, and process input constraints can be taken into account only in a suboptimal way as discussed at the end of that section. Moreover, constraints on process outputs could not be taken into account. As has been mentioned earlier, numerical MPC algorithms numerically solving at each sampling instant an optimization problem yield optimal constrained solutions, but, in general, there are no fast and reliable numerical procedures for solving optimization problems with nonlinear process models. One of the most natural suboptimal solutions to this problem is to linearize the nonlinear process model at each sampling instant, and then to use a linear MPC algorithm to evaluate the controller output signal – we get in this way MPC-NSL (MPC-Nonlinear with Successive Linearization) algorithms (Maciejowski, 2002; Tatjewski, 2002).

However, this is not the most clever use of a linearized process model in the MPC algorithm. To see the reason, let us recall the general principle of predictive control with a linear process model, depicted in Fig. 3. The predicted process output trajectory is divided here into two parts: the free output trajectory dependent on past process inputs and outputs, and the forced output trajectory dependent on controller decision variables – actual and future process input moves. The difference between the two trajectories is fundamental from the computational point of view: at each sampling instant the former is calculated *only once* (and serves as a set of constant parameters in the optimization problem) whereas the latter must be evaluated every time the future control moves are changed, i.e., *several times* during the optimization process, thus its properties define the properties of optimization. Therefore, if the free trajectory is calculated using the nonlinear process model but the forced trajectory is calculated using the linear (linearized) process model, then the optimization problem is still a QP (Quadratic Programming) problem that can be solved fast and reliably. This is the basic idea of probably the most clever use of linearization in nonlinear MPC structures – the idea of the MPC-NPL (MPC with Nonlinear Prediction and Linearization) algorithm (Garcia, 1984; Tatjewski, 2002). It is explained in Fig. 8, where the free output trajectory calculated at the k-th sampling instant is denoted by $\mathcal{Y}^0(k)$,

$$\mathcal{Y}^0(k)=\big[y^0(k+1|k)\ y^0(k+2|k)\ \cdots\ y^0(k+N|k)\big]^T,\tag{13}$$

whereas the nonlinear model linearization (over the free trajectory) is used only to calculate the dynamic matrix $\mathbf{M}(k)$ describing the relation between the future control moves and the corresponding elements of the forced tra-

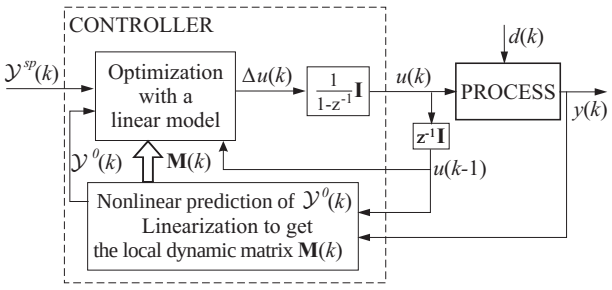

Fig. 8. Structure of the MPC-NPL controller.

jectory, c.f. (6),

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_N \end{bmatrix}. \tag{14}$$

A computationally simple linearization can be obtained when the nonlinear process model is a *fuzzy TS model* with all rule consequents of the same structure, i.e., when all local linear process models are of the same kind and structure. Denoting the vector of variables of a general process model by x, $x = [x_1, x_2, \ldots, x_n]^T$, where the elements of x can consist of process inputs and outputs (current and delayed), the rules R^i of a TS fuzzy model can be written is the following form:

$$\mathrm{R}^i: \ \text{IF} \ x_1(k) \text{ is } A_1^i \ \text{and} \ \cdots \ \text{and} \ x_n(k) \text{ is } A_n^i$$
$$\text{THEN} \ y^i(k+1) = a_0^i + a_1^i x_1(k) + \cdots + a_n^i x_n(k).$$

The output of the fuzzy model is known to be given by the formula

$$y(k+1) = \sum_{i=1}^{r} \tilde{w}^i(k) \Big[a_0^i + \sum_{j=1}^{n} a_j^i x_j(k) \Big], \tag{15}$$

where r denotes the number of rules and $\tilde{w}^i(k)$ are normalized activation levels of the fuzzy model at the sampling instant k (at a point $x(k)$). Due to the linearity and identical structure of all consequents of the rules, we can present (15) as follows:

$$y(k+1) = a_0(k) + \sum_{j=1}^{n} a_j(k) x_j(k), \tag{16}$$

where

$$a_j(k) = \sum_{i=1}^{r} \tilde{w}^i(k) a_j^i, \quad j = 0, \ldots, n. \tag{17}$$

In this way we obtain a fuzzy model in the form of a *linear model with variable coefficients*, with values depending on the current process state. If at the current step the process state is $x(k)$, then the simplest and most natural way of generating a local linear model to be used in the optimization problem of a predictive algorithm is to take the model in the form (16).

For example, let us assume that local models of process dynamics (identified at a number of process equilibrium points) are in the form of the difference equation

$$y^i(k) = -\sum_{j=1}^{n_A} a_j^i y(k-j) + \sum_{j=0}^{n_B} b_j^i u(k-j-1) + d(k). \tag{18}$$

Each equation (18) will be then a consequent of a corresponding fuzzy rule of the nonlinear fuzzy TS process model. Then, the linearization of this model at the k-th sampling instant will simply be given by

$$y(k) = -\sum_{j=1}^{n_A} a_j(k) y(k-j) + \sum_{j=0}^{n_B} b_j(k) u(k-j-1) + d(k), \tag{19}$$

where

$$a_j(k) = \sum_{i=1}^{r} \tilde{w}^i(k) a_j^i, \quad j = 1, \ldots, n_A, \tag{20}$$

$$b_j(k) = \sum_{i=1}^{r} \tilde{w}^i(k) b_j^i, \quad j = 0, \ldots, n_B, \tag{21}$$

and the matrix $\mathbf{M}(k)$ will be calculated as in the standard GPC algorithm using the coefficients $a_j(k)$ and $b_j(k)$, instead of a_j and b_j.

If local linear models are given by the local step response coefficients s_j^i, $j = 1, \ldots, D$, $i = 1, \ldots, r$ (r stands for the number of fuzzy rules), then at each sampling instant the matrix $\mathbf{M}(k)$ will be composed of the coefficients of a "fuzzy" step response given by the formula

$$s_j(k) = \sum_{i=1}^{r} \tilde{w}^i(k) s_j^i, \quad j = 1, \ldots, D. \tag{22}$$

5. MPC Algorithms Using Artificial Neural Networks

5.1. Neural-Network Model of the Plant

Let the single-input single-output (SISO) process under consideration be described by the following nonlinear discrete-time equation:

$$y(k) = g\big(u(k-\tau), \ldots, u(k-n_B), y(k-1), \ldots, y(k-n_A)\big), \tag{23}$$

where $g : \mathbb{R}^{n_A+n_B-\tau+1} \longrightarrow \mathbb{R}$, $g \in C^1$, $\tau \leq n_B$. In the sequel, it is assumed that the feedforward neural network with one hidden layer and linear output (Haykin, 1999) is used as the function g in (23). The structure of the neural network is depicted in Fig. 9. The output of the model can be expressed as

$$y(k) = w_2(0) + \sum_{i=1}^{K} w_2(i) v_i(k) = w_2(0) + \sum_{i=1}^{K} w_2(i) \varphi\big(z_i(k)\big), \tag{24}$$

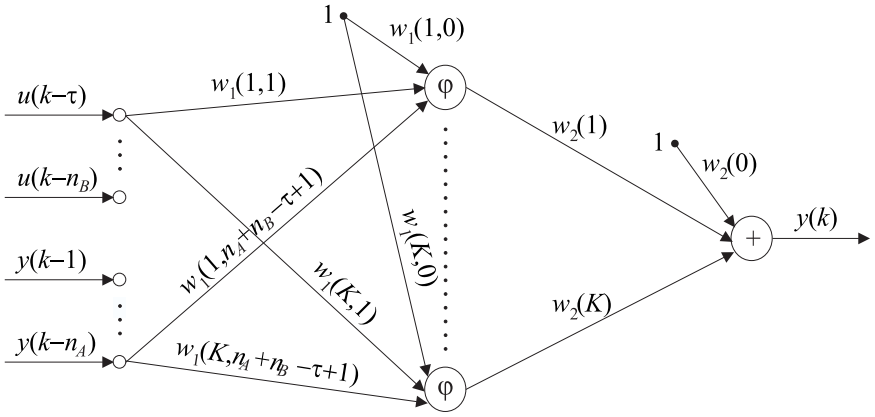

Fig. 9. Structure of the neural network.

where $z_i(k)$ is the sum of inputs and and $v_i(k)$ is the output of the i-th hidden node, respectively, $\varphi : \mathbb{R} \longrightarrow \mathbb{R}$ is a nonlinear transfer function, K is the number of hidden nodes. Recalling the input arguments of the general nonlinear model (23), one has

$$z_i(k) = w_1(i,0) + \sum_{j=1}^{I_u} w_1(i,j)u(k-\tau+1-j) + \sum_{j=1}^{n_A} w_1(i,I_u+j)y(k-j). \quad (25)$$

The weights of the network are denoted by $w_1(i,j)$, $i = 1,\ldots,K$, $j = 0,\ldots,n_A+n_B-\tau+1$, and $w_2(i)$, $i = 0,\ldots,K$, for the first and second layers, respectively. The number of the network's input nodes depending on the input signal u is $I_u = n_B - \tau + 1$. The total number of weights is $(n_A+I_u+1)K+K+1 = (n_A+n_B-\tau+2)K+K+1$.

5.2. MPC Algorithms with Nonlinear Optimization (MPC-NO) and Neural Network Models

In general, there are two methods of using neural models in MPC schemes with nonlinear optimization. In the first approach, the gradients of the cost function $J(k)$ are approximated numerically and the nonlinear optimization problem (2) is solved on-line. In the second approach, the structure of the neural model is exploited (Ławryńczuk, 2003; Ławryńczuk and Tatjewski, 2001a; 2001b; Nørgaard *et al.*, 2000). Defining the vectors

$$Y^{sp}(k) = \begin{bmatrix} y^{sp}(k+N_1|k) \\ \vdots \\ y^{sp}(k+N|k) \end{bmatrix}, \quad Y(k) = \begin{bmatrix} y(k+N_1|k) \\ \vdots \\ y(k+N|k) \end{bmatrix}, \quad U(k) = \begin{bmatrix} u(k|k) \\ \vdots \\ u(k+N_u-1|k) \end{bmatrix}, \quad (26)$$

it is convenient to express the cost function (1) in the following form:

$$J(k) = \|Y^{sp}(k) - Y(k)\|^2 + \lambda \left\|(I+J^{NO})U(k) + U^{NO}\right\|^2, \quad (27)$$

where the matrices I (unit matrix) and J^{NO} are of dimension $N_u \times N_u$, and the vector U^{NO} is of length N_u,

$$J^{NO} = \begin{bmatrix} 0 & \cdots & 0 & 0 \\ -1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & -1 & 0 \end{bmatrix}, \quad U^{NO} = \begin{bmatrix} u(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (28)$$

Differentiating (27) with respect to a future control sequence, $U(k)$, results in

$$\frac{\partial J(k)}{\partial U(k)} = 2\left(\frac{\partial Y(k)}{\partial U(k)}\right)^T \left(Y(k) - Y^{sp}(k)\right) + 2\lambda\left(I+J^{NO}\right)^T U(k). \quad (29)$$

The matrix of dimension $(N-N_u+1)\times N_u$, containing partial derivatives of the predicted output with respect to future control is

$$\frac{\partial Y(k)}{\partial U(k)}=\begin{bmatrix}\dfrac{\partial y(k+N_1|k)}{\partial u(k|k)} & \cdots & \dfrac{\partial y(k+N_1|k)}{\partial u(k+N_u-1|k)}\\ \vdots & \ddots & \vdots\\ \dfrac{\partial y(k+N|k)}{\partial u(k|k)} & \cdots & \dfrac{\partial y(k+N|k)}{\partial u(k+N_u-1|k)}\end{bmatrix}. \tag{30}$$

The predictions $y(k+p|k)$ for $p=N_1,\ldots,N$ are calculated from the general prediction equation

$$y(k+p|k)=y(k+p)+d(k), \tag{31}$$

where the quantities $y(k+p)$ are calculated from the model. The above formulation uses the "DMC type" disturbance model (Maciejowski, 2002; Tatjewski, 2002), in which the unmeasured disturbance $d(k)$ is assumed to be constant over the prediction horizon. Its value is estimated from the equation

$$\begin{aligned}d(k)&=y(k)-y(k|k-1)\\&=y(k)-\left(w_2(0)+\sum_{i=1}^{K}w_2(i)v_i(k)\right).\end{aligned} \tag{32}$$

From the neural model (24) one has

$$y(k+p)=w_2(0)+\sum_{i=1}^{K}w_2(i)\varphi\big(z_i(k+p)\big). \tag{33}$$

As regards prediction over the horizon N, the quantities $z_i(k+p)$ and, consequently, $y(k+p)$ will depend on some control signal values applied to the plant at previous sampling instances, future control signals, i.e., decision variables of the control algorithm, measured values of the plant output signal and future output predictions. From (25) one has

$$\begin{aligned}z_i(k+p)&=w_1(i,0)\\&\quad+\sum_{j=1}^{I_{uf}(p)}w_1(i,j)u(k-\tau+1-j+p|k)\\&\quad+\sum_{j=I_{uf}(p)+1}^{I_u}w_1(i,j)u(k-\tau+1-j+p)\\&\quad+\sum_{j=1}^{I_{yp}(p)}w_1(i,I_u+j)y(k-j+p|k)\\&\quad+\sum_{j=I_{yp}(p)+1}^{n_A}w_1(I_u+j)y(k-j+p),\end{aligned} \tag{34}$$

where $I_{uf}(p)=\max\{\min\{p-\tau+1,I_u\},0\}$ is the number of the network's input nodes depending on future control signals and $I_{yp}(p)=\min\{p-1,n_A\}$ is the number of the network's input nodes depending on output predictions. Because typically $N_u<N$ (hence $u(k+p|k)=u(k+N_u-1|k)$ for $p\geq N_u$), it can be noticed that

$$\begin{aligned}&\sum_{j=1}^{I_{uf}(p)}u(k-\tau+1-j+p|k)\\&\quad=I_{N_u}(p)\,u(k+N_u-1|k)\\&\qquad+\sum_{j=I_{N_u}(p)+1}^{I_{uf}(p)}u(k-\tau+1-j+p|k),\end{aligned} \tag{35}$$

where $I_{N_u}(p)=\min\{\max\{p-\tau-N_u+1,0\},I_u\}$. Taking into account (35), Eqn. (34) can be written as

$$\begin{aligned}z_i(k+p)&=w_1(i,0)\\&\quad+\sum_{j=1}^{I_{N_u}(p)}w_1(i,j)u(k+N_u-1|k)\\&\quad+\sum_{j=I_{N_u}(p)+1}^{I_{uf}(p)}w_1(i,j)u(k-\tau+1-j+p|k)\\&\quad+\sum_{j=I_{uf}(p)+1}^{I_u}w_1(i,j)u(k-\tau+1-j+p)\\&\quad+\sum_{j=1}^{I_{yp}(p)}w_1(i,I_u+j)y(k-j+p|k)\\&\quad+\sum_{j=I_{yp}(p)+1}^{n_A}w_1(I_u+j)y(k-j+p).\end{aligned} \tag{36}$$

Taking into account (31) and (33), the entries of the matrix $\partial Y(k)/\partial U(k)$ given by (30), i.e., the partial derivatives of the predicted output signal with respect to future controls are calculated from

$$\frac{\partial y(k+p|k)}{\partial u(k+r|k)}=\sum_{i=1}^{K}w_2(i)\frac{\mathrm{d}\varphi(z_i(k+p))}{\mathrm{d}z_i(k+p)}\frac{\partial z_i(k+p)}{\partial u(k+r|k)}. \tag{37}$$

Obviously,

$$\frac{\partial z_i(k+p)}{\partial u(k+r|k)}=\frac{\partial y(k+p|k)}{\partial u(k+r|k)}=0,\quad r\geq p-\tau+1. \tag{38}$$

It can be noted that decision variables of the algorithm affect only the first, second and fourth sums in (36). It can

be also noted that only some of the output predictions are influenced by future controls. Hence

$$\frac{\partial z_i(k+p)}{\partial u(k+r|k)} = \sum_{j=1}^{I_{N_u}(p)} w_1(i,j)\frac{\partial u(k+N_u-1|k)}{\partial u(k+r|k)} + \sum_{j=I_{N_u}(p)+1}^{I_{uf}(p)} w_1(i,j)\frac{\partial u(k-\tau+1-j+p|k)}{\partial u(k+r|k)} + \sum_{j=1}^{I_{ypf}(p)} w_1(i,I_u+j)\frac{\partial y(k-j+p|k)}{\partial u(k+r|k)}, \quad (39)$$

where $I_{ypf}(p) = \max\{\min\{p-\tau, n_A\}, 0\}$ is the number of network input nodes depending on output predictions which are affected by future controls. Obviously,

$$\frac{\partial u(k+p|k)}{\partial u(k+r|k)} = \begin{cases} 0 & \text{if} \quad p \neq r, \\ 1 & \text{if} \quad p = r, \end{cases} \quad (40)$$

whereas the derivatives of the predicted output signals with respect to future controls have to be calculated recursively.

The discussed method of calculating the gradients of the predicted output trajectory with respect to future controls is used not only for obtaining the gradients of the cost function $J(k)$, but also for finding the gradients of output constraints if they have to be taken into account. In some nonlinear optimization algorithms, for example, SQP (Bazaraa *et al.*, 1993), the analytical Hessian matrix can be used. Unfortunately, it requires much more computational effort than the calculation of gradients. That is why in the presented solution the optimization routine is provided with analytical gradients while the Hessian is approximated, as is done in most SQP practical implementations.

The extension of the presented MPC-NO algorithm with neural networks to systems with many inputs and many outputs (MIMO) is discussed in (Ławryńczuk, 2003) and (Ławryńczuk and Tatjewski, 2001a). As the model of the process n_y ($= \dim y$) MISO nonlinear models (neural networks) are used.

5.3. Linearization-Based MPC Algorithm with a Neural Network Model

As was mentioned in Section 4.2, one of the most precise and numerically effective combinations is a nonlinear free response prediction together with the use of a linearized model for forced response calculations, i.e., for on-line optimization in the form of a QP problem.

Defining a linearization point as a vector composed of past input and output values:

$$\overline{x}(k) = \left[\overline{u}(k-\tau)\ \ldots\ \overline{u}(k-n_B)\ \overline{y}(k-1)\ \ldots\ \overline{y}(k-n_A)\right]^T \quad (41)$$

and using the Taylor series expansion at this point, the linearized model has the form

$$y(k) = g\big(\overline{x}(k)\big) + \sum_{i=1}^{n_B} b_l\big(\overline{x}(k)\big)\big(u(k-l)-\overline{u}(k-l)\big) - \sum_{i=1}^{n_A} a_l\big(\overline{x}(k)\big)\big(y(k-l)-\overline{y}(k-l)\big). \quad (42)$$

Taking into account the structure of the neural model and the corresponding equations (24) and (25), the coefficients of the linearized model are calculated from

$$a_l\big(\overline{x}(k)\big) = -\frac{\partial g\big(\overline{x}(k)\big)}{\partial y(k-l)} = -\sum_{i=1}^{K} w_2(i)\frac{\mathrm{d}\varphi(z_i(\overline{x}(k)))}{\mathrm{d}z_i(\overline{x}(k))}(i, I_u+l), \quad l = 1,\ldots,n_A \quad (43)$$

and

$$b_l\big(\overline{x}(k)\big) = \begin{cases} 0 & \text{if} \quad l = 1,\ldots,\tau-1, \\ \dfrac{\partial g(\overline{x}(k))}{\partial u(k-l)} & \\ \quad = \displaystyle\sum_{i=1}^{K} w_2(i)\frac{\mathrm{d}\varphi(z_i(\overline{x}(k)))}{\mathrm{d}z_i(\overline{x}(k))}(i, l-\tau+1) & \\ & \text{if} \quad l = \tau,\ldots,n_B. \end{cases} \quad (44)$$

Let

$$a_l(k) = a_l\big(\overline{x}(k)\big), \quad b_l(k) = b_l\big(\overline{x}(k)\big) \quad (45)$$

and redefine the variables

$$\begin{aligned} y(k) &:= y(k) - g\big(\overline{x}(k)\big), \\ y(k-i) &:= y(k-i) - \overline{y}(k-i), \quad l = 1,\ldots,n_A, \\ u(k-i) &:= u(k-i) - \overline{u}(k-i), \quad l = 1,\ldots,n_B. \end{aligned} \quad (46)$$

The linear approximation of the model (23), obtained at the sampling instant k, can be expressed as

$$\mathbf{A}(k, z^{-1})y(k) = \mathbf{B}(k, z^{-1})u(k), \quad (47)$$

where

$$\mathbf{A}(k,z^{-1}) = (1 + a_1(k)z^{-1} + \cdots + a_{n_A}(k)z^{-n_A}),$$
$$\mathbf{B}(k,z^{-1}) = (b_1(k)z^{-1} + \cdots + b_{n_B}(k)z^{-n_B}). \quad (48)$$

It can be noted that the linearization point given by (41) and hence the coefficients $a_l(k),\ b_l(k)$ are not influenced by the most recent output value $y(k)$, which is available to measurements. It may be crucial in the case of fast processes. Therefore, it is then recommended to use

$$\overline{x}(k) = \big[\overline{u}(k-\tau+1)\ \ldots\ \overline{u}(k-n_B+1)\ \overline{y}(k) \ldots\ \overline{y}(k-n_A+1)\big]^T. \quad (49)$$

If $\tau = 1$, for linearization purposes one may set $u(k) = u(k-1)$ or $u(k) = u(k|k-1)$. The MPC-NPL algorithm using the linearization point (41) will be named NPL1 whereas the one which uses (49) – NPL2.

The nonlinear free response $y^0(k+p|k),\ p = 1,\ldots,N$, is calculated recursively from the general prediction equation (31), taking into account the output of the neural model given by (33) and the DMC disturbance model (32):

$$y^0(k+p|k) = w_2(0) + \sum_{i=1}^{K} w_2(i)\varphi\big(z_i^0(k+p)\big) + d(k). \quad (50)$$

The quantities $z_i^0(k+p)$ are determined from (34) assuming no changes in control signals from the sampling instant k and replacing the predicted output signal from $k+1$ by the corresponding values of the free response:

$$\begin{aligned} u(k+p|k) &:= u(k-1), \qquad p \geq 0, \\ y(k+p|k) &:= y^0(k+p|k), \quad p \geq 1. \end{aligned} \quad (51)$$

Hence

$$\begin{aligned} z_i^0(k+p) = w_1(i,0) &+ \sum_{j=1}^{I_{uf}(p)} w_1(i,j)u(k-1) \\ &+ \sum_{j=I_{uf}(p)+1}^{I_u} w_1(i,j)u(k-\tau+1-j+p) \\ &+ \sum_{j=1}^{I_{yp}(p)} w_1(i,I_u+j)y^0(k-j+p|k) \\ &+ \sum_{j=I_{yp}(p)+1}^{n_A} w_1(I_u+j)y(k-j+p). \end{aligned} \quad (52)$$

The extension of the presented MPC-NPL algorithm with neural networks to systems with many inputs and many outputs (MIMO) is discussed in (Ławryńczuk, 2003).

The stability of the presented MPC-NPL and NO algorithms with neural models is achieved by properly tuning the weighting coefficient λ in the cost function $J(k)$. Both of these MPC-NPL and MPC-NO algorithms can be combined with the stabilizing dual-mode approach (Ławryńczuk and Tatjewski, 2004), (Ławryńczuk, 2003) developed by Michalska and Mayne (1993). The control law is calculated as the solution to the standard MPC optimization problem if the current state of the system lies outside a neighborhood of the origin while a local, usually linear, controller is used otherwise. An additional inequality constraint is imposed on the predicted terminal state. In this approach merely feasibility, rather than optimality, is sufficient to guarantee stability.

5.4. Reducing Computational Complexity in MPC with Neural Networks

The MPC-NO algorithm is computationally demanding and the computation time is much longer than that of linearization-based algorithms. Moreover, the main difficulty comes from the fact that the optimization problem that has to be solved on-line is usually nonconvex and it may terminate in local minima. That is why linearization-based MPC approaches attract so much attention and are widely used in industry (Henson, 1998; Morari and Lee, 1999). On the other hand, to reduce the computational complexity, a few neural-network based alternatives have been suggested. In general, these approaches can be divided into two groups: in the first one, special structures of neural models are used to make the optimization problem simpler (convex), while in the second, explicit approximate MPC algorithms (without on-line optimization) combined with neural networks are used.

5.4.1. Neural Network Based MPC with On-Line Optimization

Even though linearization-based MPC algorithms result in the quadratic optimization problem which poses no numerical difficulty, in some specific applications, namely, in the case of large-scale or fast-sampling systems, it is necessary to reduce the computational burden. An interesting neural approach is presented in (Wang and Wan, 2001). A structured neural network that implements the gradient projection algorithm is developed to solve the constrained QP problem in a massively parallel fashion. Specifically, the structured network consists of a projection network and a network which implements the gradient projection algorithm. The projection network consists of specially structured linear neurons. A training algo-

rithm is formulated for which the convergence is guaranteed. The networks are trained off-line, whereas the controls are calculated on-line from the networks without any optimization.

In addition to linearization-based MPC schemes which use the QP approach, it is also possible to develop a specially structured neural model to avoid the necessity of nonlinear optimization. In (Liu *et al.*, 1998), affine nonlinear models of the following form are considered:

$$y(k) = F\big(\underline{y}(k)\big) + G\big(\underline{y}(k)\big)u(k-\tau), \tag{53}$$

where $F(\cdot)$ and $G(\cdot)$ are nonlinear functions and

$$\underline{y}(k) = \big[y(k-1)\ \ldots\ y(k-n_A)\big]^T. \tag{54}$$

A set of nonlinear affine predictors of the following structure is used in the MPC algorithm:

$$y(k+\tau+p|k) = F_p\big(\underline{x}(k)\big) + \sum_{j=0}^{p} G_{pj}\big(\underline{x}(k)\big)u(k+j|k), \tag{55}$$

where $p = 0,\ldots,N$. The quantities $F_p(\underline{x}(k))$ and $G_{pj}(\underline{x}(k))$, which depend on the current state of the plant

$$\underline{x}(k) = \big[y(k)\ \ldots\ y(k-n_A)\ u(k-1)\ \ldots\ u(k-\tau)\big]^T, \tag{56}$$

are calculated by neural networks. The key idea is that the present and future controls, i.e., the decision variables of the optimization problem occur linearly in the predictor's equation (55). The predictor depends in a nonlinear way only on the past values of input and output signals. Hence, the resulting MPC optimization problem is convex. In some cases, however, such an approach to modeling may turn out to be insufficient to capture the nonlinearity precisely enough. In (Liu *et al.*, 1998), the corresponding on-line training algorithm and the MPC scheme based on such predictors are discussed.

Neural networks are also used in commercially available software packages, e.g., Process Perfecter from Pavilion Technologies Inc. (Piche *et al.*, 2000). In this interesting approach, to simplify the identification task and solve the MPC optimization problem on-line, a neural network is used to capture the steady-state properties of the process. A second-order quadratic-in-the-input dynamic model is used,

$$\delta y(k) = \sum_{i=1}^{2}\Big(v_i\delta u(k-i) + w_i\left(\delta u(k-i)\right)^2 - a_i\delta y(k-i)\Big). \tag{57}$$

The coefficients of the dynamic model, v_i and w_i, which depend on the current state of the plant are calculated from the neural static model. The resulting MPC optimization task is not convex. Nevertheless, the model is relatively simple and the approach is reported to be successful in many industrial applications.

5.4.2. Neural Network Based MPC without On-Line Optimization

In recent years approximate MPC algorithms have attracted much attention (Bemporad *et al.*, 2002; Johansen, 2004). The key idea is to calculate control signals on-line without any optimization. The main advantage of this approach is its speed. On the other hand, the control law must be precomputed off-line and stored somehow in the controller's memory. Because, in general, a very large amount of computer memory may be necessary to store the control law, it is justified to find its approximation to efficiently compute control on-line.

Bemporad *et al.* (2002) show that for linear systems with linear constraints and a quadratic cost function the optimal control action is a continuous and piecewise affine function of the state. The state space is partitioned into polyhedral sets and a control law is calculated for every set. This results in a search tree which determines the set to which a given state belongs. The tree is computed off-line, but the classification and calculation of the control signals is performed on-line. The main advantage is the fact that the necessity of on-line optimization is avoided. Unfortunately, the number of polyhedral sets can be huge, which affects the overall computational efficiency. Neural networks can be effectively used to improve the situation (Haimovich *et al.*, 2003). The solution to the constrained linear MPC optimization problem is pre-computed off-line in an explicit form as a piecewise affine state feedback law defined on polyhedral regions of the state space (Bemporad *et al.*, 2002). The problem of determining on-line in which polyhedral region the state lies is solved by using a neural network, i.e., for a given state of the system the network determines the corresponding region. An apparent advantage of this approach is that regions that have the same control law are joined. The structure of the closed-loop system is depicted in Fig. 10. Unlike the approxi-

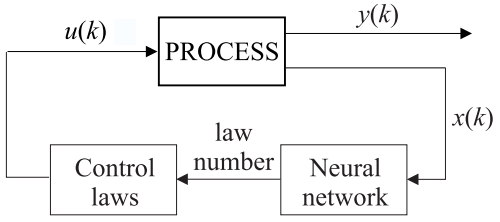

Fig. 10. Approximate neural MPC scheme with neural selection of the control law.

mate neural controller which replaces the whole MPC algorithm as in (Parisini *et al.*, 1998; Parisini and Zoppoli, 1995), the neural network is not used for finding the actual control directly. Comparing to other solutions, the advantage of the neural approach is the fact that the complexity of the neural network grows linearly with the number of control laws. The multilayer feedforward neural network with two hidden layers is used.

Another version of the neural approach for the "explicit" constrained MPC scheme for linear systems is described in (Hoekstra *et al.*, 2001). It is known that the solution to the standard MPC optimization problem is a continuous function of the state, the reference signal, the noise and the disturbances. Hence, it can be approximated arbitrarily well by a feedforward neural network. In order to develop a analytic predictive controller, at first the standard one, an optimization-based one, has to be developed. Closed-loop simulations have to be performed to gather examples to cover the operation domain. Because the obtained controller is only an approximation of the real one, precise satisfaction of constraints cannot be guaranteed. A trivial solution is to saturate the output of the neural network. The constraints cannot be modified for a given neural network.

In (Parisini *et al.*, 1998; Parisini and Zoppoli, 1995), an interesting nonlinear stabilizing approximate neural predictive controller is suggested. The structure of the closed-loop system is depicted in Fig. 11. The overall procedure consists of two steps. At first, a stabilizing MPC algorithm is developed. The formulation used allows us to take into account constraints imposed on system inputs and states. Stability is enforced by a proper terminal penalty term in the cost function, and no stabilizing terminal equality constraints are necessary:

$$J(k) = \sum_{p=0}^{N-1} h\big(x(k+p|k), u(k+p|k)\big) + a\,\|x(k+N|k)\|_P^2, \tag{58}$$

where $h(x(k+p|k), u(k+p|k))$ is the cost function (in general, a nonquadratic one), a is a positive scalar and P is a positive definite symmetric matrix. The algorithm is

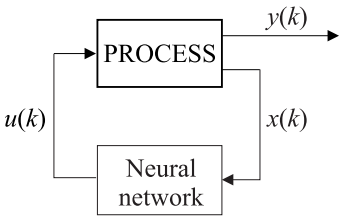

Fig. 11. Approximate neural MPC scheme.

then approximated by $n_u = \dim u$ single-output feedforward neural networks with one hidden layer with nonlinear transfer functions and linear output units. Networks are trained off-line. Bounds on the approximation are established. The main parameters of the algorithm are determined partly analytically (the scalar a and the penalty matrix P) and partly by solving specific global optimization problems (the number of hidden units in the neural network). Because neural networks work as approximators, nonoptimal control values may be obtained. It can be proved, however, that, being suboptimal, the algorithm remains stable. Training neural networks is a serious issue, because a sufficiently large set of patterns is necessary to cover the whole operation domain.

An interesting hybrid approximate neural predictive controller is presented in (Parisini and Sacone, 2001). A hybrid scheme consists of two control layers: a continuous one, in which a set of neural approximate MPC regulators are used, and a supervisory discrete-event one, aimed at choosing the best control action (from those proposed by the regulators) which should be applied to the plant. The choice depends on the current condition of the system and on external events. Such an approach may be efficient for complex systems with many decision variables. The stability of the overall hybrid system is guaranteed.

A predictive neural-network controller of uncertain systems is investigated in (Vila and Wagner, 2003). In this solution neural networks are used both for modeling and minimization. More specifically, to increase the accuracy of the prediction, the model consists of N separate networks. Both the system and controller neural networks are first selected off-line by a statistical Bayesian procedure in order to make the predictive controller work effectively. An off-line selection procedure of the neural predictor's and neurocontroller's architecture are discussed, and the issue of the stochastic stability of the closed loop is considered as well.

5.5. Applications and Exemplary Simulation Results

MPC algorithms with neural network models of different structures have been applied to a wide class of processes, for example, a combustion system (Liu and Daley, 1999), a pneumatic servo system (Nørgaard *et al.*, 2000), a mobile robot (Ortega and Camacho, 1996), an industrial packed bed reactor (Temeng *et al.*, 1995), an insulin delivery problem (Trajanoski and Wach, 1998), a multivariable chemical reactor (Yu and Gomm, 2003), traffic control on freeways (Parisini and Sacone, 2001), and a biological depolluting treatment of wastewater (Vila and Wagner, 2003). In the following part of the article, simulation examples of a control system with MPC using neural networks will be given.

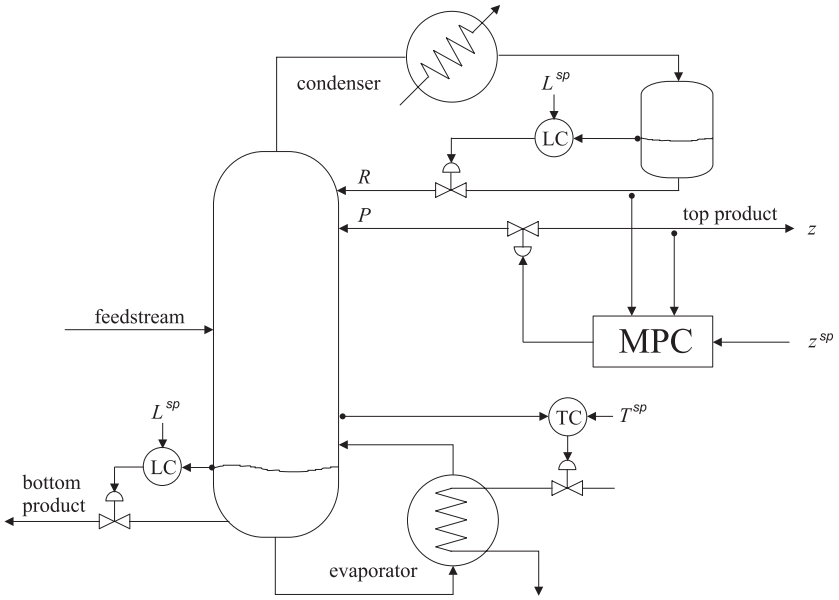

Fig. 12. High-purity high-pressure ethylene-ethane distillation column control system.

5.5.1. High-Purity High-Pressure Ethylene-Ethane Distillation Column

The plant under consideration is a high purity, high pressure (1,93 MPa) ethylene-ethane distillation column shown in Fig. 12 (Ławryńczuk, 2003). The feedstream consists of ethylene (approx. 80%), ethane (approx. 20%), and traces of hydrogen, methane and propylene. The product of the distillation is ethylene which can contain up to 1000 ppm (parts per million) of ethane. The main problem is to develop a supervisory controller which would be able to increase relatively fast the impurity level when composition changes in the feedstream are relatively small. Reducing the purity of the product, of course taking into account technological limits, results in a decreased energy consumption. The production scale is very large, the nominal value of the product stream being equal to 43 tons/h. The column has 121 trays and the feedstream is delivered to the tray number 37.

Two fast single-loop PID controllers (denoted by LC) are used to maintain the levels in the reflux tank and the bottom product tank. Yet another PID controller (denoted by TC) is also used to control the temperature on the tray number 13. The PID controllers comprise the basic control layer. As far as the supervisory MPC algorithm is concerned, the control loop has one input variable r, which is the reflux ratio $r = R/P$, where R is the reflux stream delivered to the column by the top tray and P is the product stream taken from the tray number 110, and one output variable z, which represents the impurity level in the product (ethylene). The sampling time of the MPC algorithm is relatively long (a slow composition analyzer), equal to $Tp = 40$ min.

Four models of the plant were used. The first one was used as the real process during the simulations. It was based on technological considerations (Ławryńczuk, 2003). An identification procedure was carried out, and as a result two linear models for different operating points and a neural one were obtained. For the empirical models we have $n_A = 1$, $\tau = n_B = 3$. The horizons were set to $N = 10$, $N_u = 3$, the weighting coefficient $\lambda = 2$. In all the simulations it is assumed that at the sampling instant $k = 1$ the set-point value is changed from 100 ppm to 350 ppm, 600 ppm and 850 ppm. Because of some technological reasons, the following constraints were imposed on the reflux ratio: $r^{\min} = 4.051$, $r^{\max} = 4.4571$.

At first, MPC algorithms based on two linear models were developed. The first linear model is valid for a "low" impurity level and the resulting control algorithm works well in this region, but exhibits unacceptable oscillatory behavior for medium and big setpoint changes, as is shown in Fig. 13. On the contrary, the second linear model captures the process properties for a "high" impurity level and the closed-loop response is fast enough for the biggest setpoint change, but very slow for smaller ones, as is shown in Fig. 14.

Simulation results of MPC-NPL algorithms with a neural network are depicted in Fig. 15. Both algorithms work well for all three setpoint changes. The NPL1 algorithm is slightly slower than NPL2. Simulation results of the MPC-NO algorithm with a neural network are shown in Fig. 16. Compared with suboptimal linearization-based algorithms, nonlinear optimization leads to faster closed-loop responses.

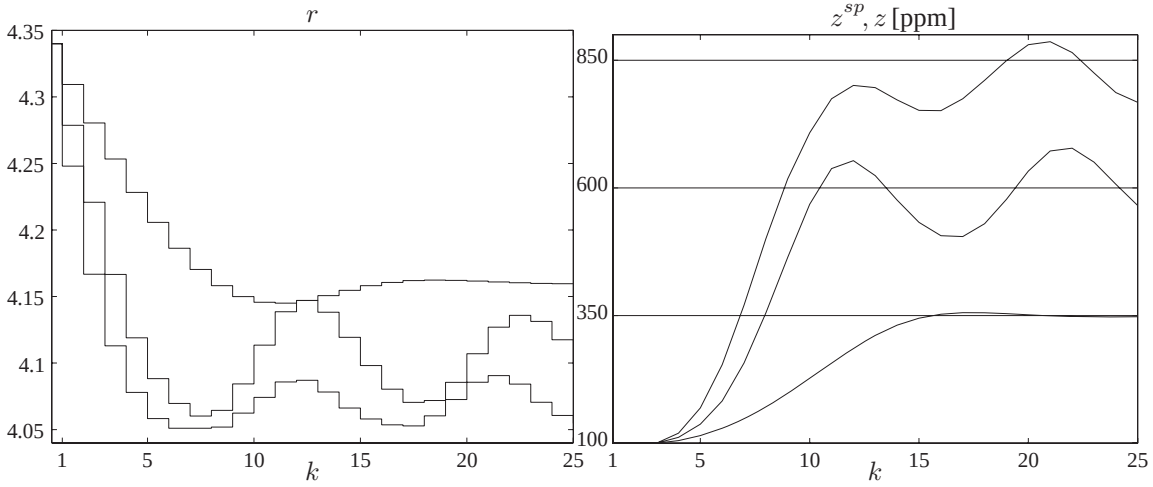

Fig. 13. Simulation results of the ethylene-ethane distillation column with the MPC algorithm based on a linear model for a “low” impurity level.

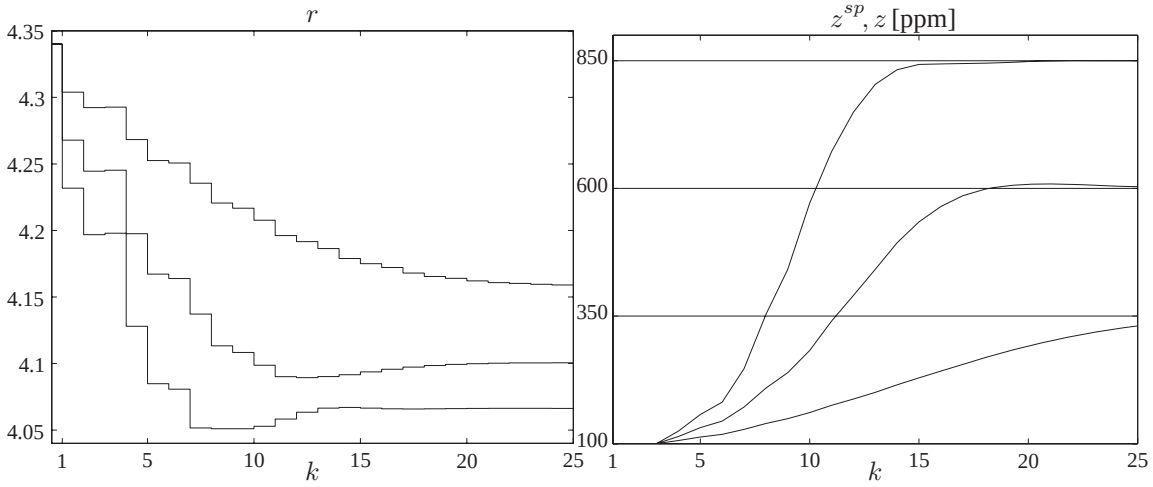

Fig. 14. Simulation results of the ethylene-ethane distillation column with the MPC algorithm based on a linear model for a “high” impurity level.

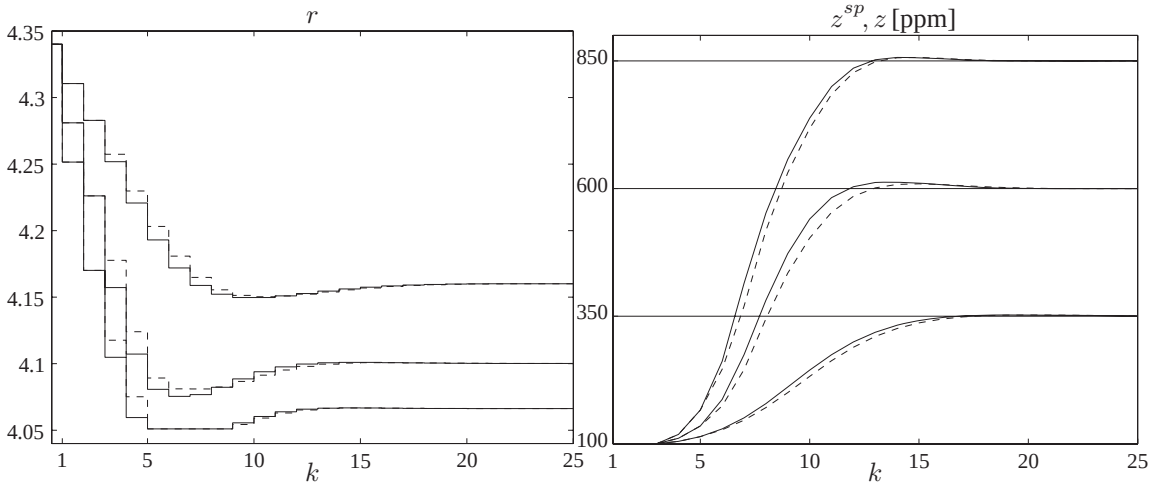

Fig. 15. Simulation results of the ethylene-ethane distillation column with MPC-NPL1 (dashed line) and MPC-NPL2 (solid line) algorithms with neural networks.

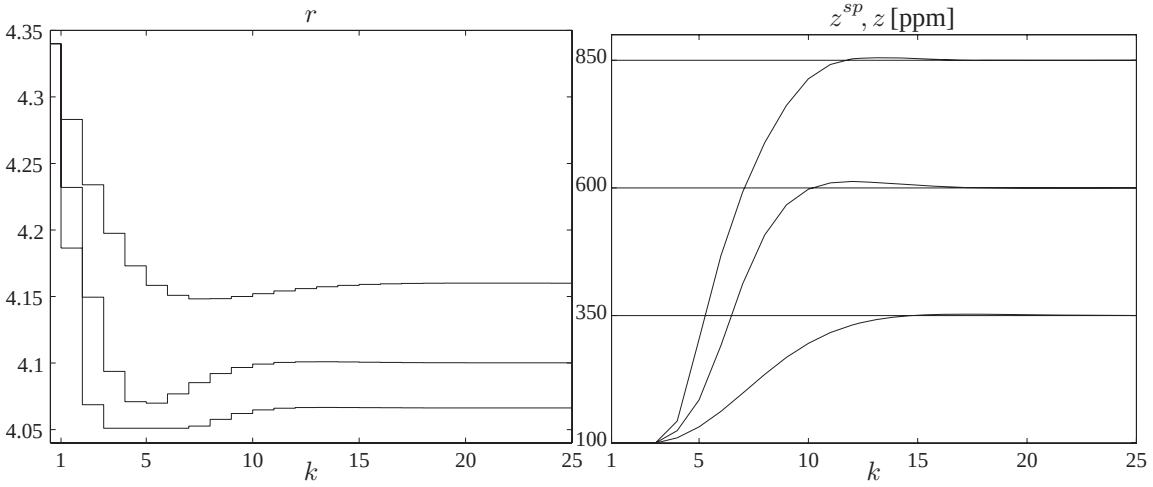

Fig. 16. Simulation results of the ethylene-ethane distillation column with the MPC-NO algorithm with a neural network.

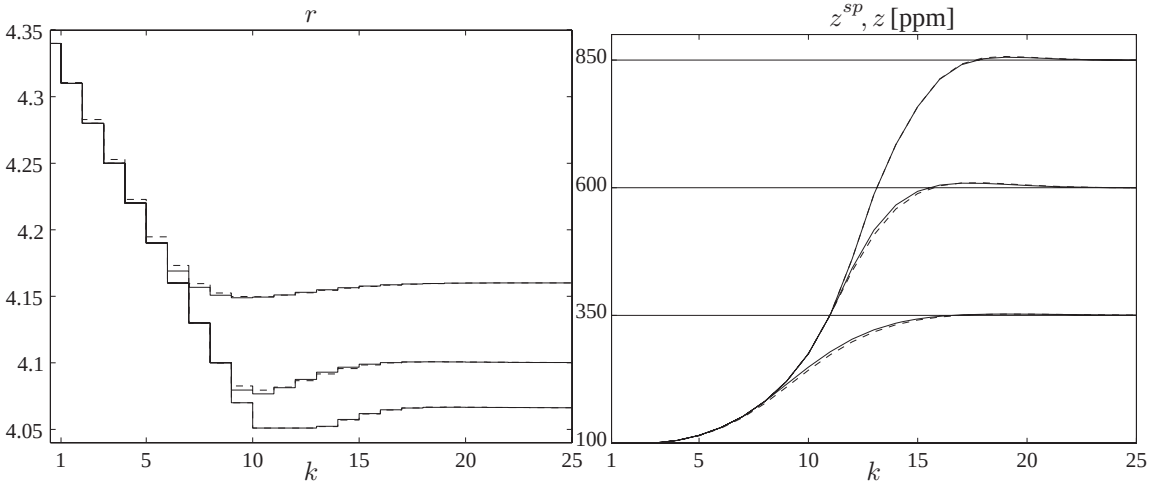

Fig. 17. Simulation results of the ethylene-ethane distillation column with the MPC-NPL2 (dashed line) and MPC-NO (solid line) algorithms with neural networks and the constraint $\Delta r^{\max} = 0.03$.

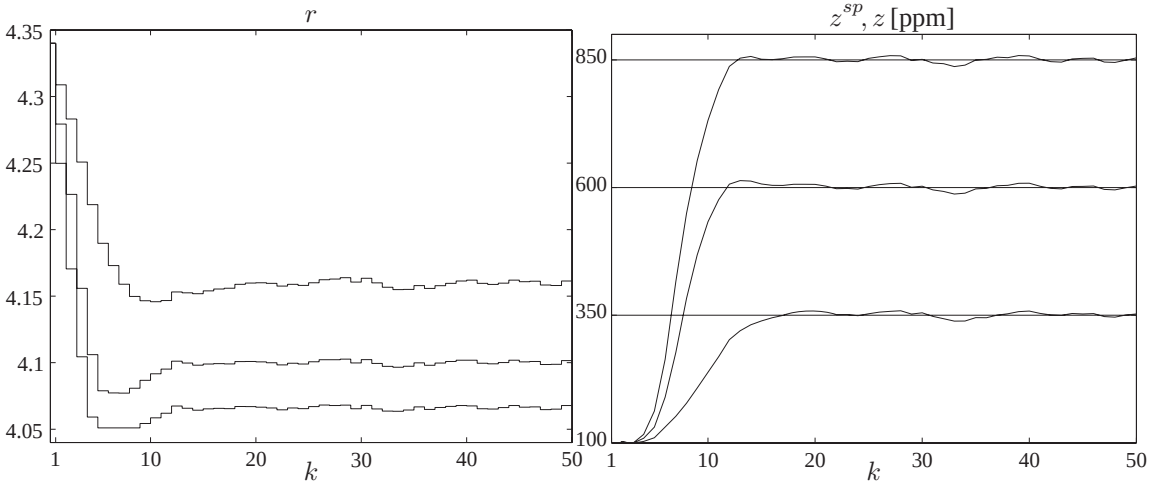

Fig. 18. Simulation results of the ethylene-ethane distillation column with the MPC-NPL2 algorithm with a neural network and unmeasured disturbances.

In practice, big changes in the manipulated variable r are not allowed because of technological and safety reasons (high pressure, big production scale). That is why an additional constraint $\Delta r^{\max} = 0.03$ was used. Figure 17 compares simulation results of the MPC-NPL2 and MPC-NO algorithms with a neural network. Although the constraint significantly slows closed-loop responses, it can still be noticed that the MPC-NO algorithm is somewhat faster.

Simulation results of the MPC-NPL2 algorithm and unmeasured stochastic disturbances are presented in Fig. 18. Such disturbances are unavoidable in industry.

6. Conclusions

The subject of the paper was applications of soft computing methods to model-based predictive control techniques. The resulting algorithms make it possible to effectively control highly nonlinear, multidimensional processes, usually subject to constraints which result from technological and safety reasons. The algorithms considered can be easily implemented and used on-line.

As far as MPC algorithms using fuzzy techniques are concerned, explicit (unconstrained) multimodel algorithms exploiting the fuzzy Takagi-Sugeno (TS) structure and algorithms with on-line linearization of a fuzzy TS model were described. In the first case the values of control variables can be computed by a set of explicit formulas, which makes the application very easy. In the second case (MPC-NPL algorithms), decision variables are obtained by solving a QP optimization problem, which allows taking constraints explicitly into account.

A family of MPC algorithms using artificial neural networks (i.e., the most popular multilayer perceptron) were also described. In comparison with fuzzy models, neural structures do not suffer from the "curse of dimensionality", which is troublesome in multivariable cases. Three classes of solutions were considered. Algorithms with nonlinear optimization are potentially very precise, but they hinge on the effectiveness of the optimization routine used. Because, in practice, convergence to a global optimum cannot be guaranteed, MPC-NPL algorithms with on-line linearization of a neural network model were also presented. A number of approaches aimed at reducing the computational complexity of the optimization problem or to approximate the whole control algorithm were also reviewed.

In the paper, emphasis was put on computational efficiency. It was shown that the application of fuzzy reasoning and neural networks leads to efficient realizations of nonlinear predictive control algorithms, suitable for effective on-line applications. It should be also mentioned that the research area concerned with using soft computing techniques in predictive control is still open and many new results appear every year.

References

Allgöwer F., Badgwell T.A., Qin J.S., Rawlings J.B. and Wright S.J. (1999): *Nonlinear predictive control and moving horizon estimation – An introductory overview*, In: Advances in Control – Highlights of ECC'99 (P.M. Frank, ed.).— London: Springer, pp. 391–449.

Babuška R., Sousa J.M. and Verbruggen H.B. (1999): *Predictive control of nonlinear systems based on fuzzy and neural models.* — Proc. *European Control Conf.*, Karlsruhe, Germany, (on CD-ROM).

Bazaraa M.S., Sherali J. and Shetty K. (1993): *Nonlinear Programming: Theory and Algorithms.* — New York: Wiley.

Bemporad A., Morari M, Dua V. and Pistikopoulos E.N. (2002): *The explicit linear quadratic regulator for constrained systems.* — Automatica, Vol. 38, No. 1, pp. 3–20.

Brdyś M.A. and Tatjewski P. (2005): *Iterative Algorithms for Multilayer Optimizing Control.* — London: Imperial College Press/World Scientific.

Camacho E.F. and Bordons C. (1999): *Model Predictive Control.* — London: Springer.

Clarke D.W., Mohtadi C. and Tuffs P.S. (1987): *Generalized predictive control—I. The basic algorithm.* — Automatica, Vol. 23, No. 2, pp. 137–148.

Cutler R. and Ramaker B.L. (1979): *Dynamic matrix control – A computer control algorithm.* — Proc. *AIChE National Meeting*, Houston.

Eder H.H. (1999): *MBPC benefits and key success factors.* — Proc. 5-th *European Control Conf., ECC'99*, Karlsruhe, Germany, (on CD-ROM).

Findeisen W., Bailey F.N., Brdys M., Malinowski K., Tatjewski P. and Wozniak A. (1980): *Control and Coordination in Hierarchical Systems.* — Chichester: Wiley.

Findeisen W. (1997): *Control Structures for Complex Processes*. — Warsaw: Warsaw University of Technology Press, (in Polish).

Garcia C.E. (1984): *Quadratic/dynamic matrix control of nonlinear processes: An application to a batch reaction process.* — Proc. *AIChE Annual Meeting*, San Francisco.

Haimovich H., Seron. M.M., Goodwin G.C. and Agüero J.C. (2003): *A neural approximation to the explicit solution of constrained linear MPC.* — Proc. *European Control Conf.*, Cambridge, UK, (on CD-ROM).

Haykin S. (1999): *Neural Networks – A Comprehensive Foundation.* — Englewood Cliffs, NY: Prentice Hall.

Henson M.A. (1998): *Nonlinear model predictive control: Current status and future directions.* — Comput. Chem. Eng., Vol. 23, No. 2, pp. 187–202.

Hoekstra P., van den Boom T.J.J. and Botto M.A. (2001): *Design of an analytic constrained predictive controller using neural networks.* — Proc. *European Control Conf.*, Porto, Portugal, (on CD-ROM).

Johansen T.A. (2004): *Approximate explicit receding horizon control of constrained nonlinear systems.* — Automatica, Vol. 40, No. 2, pp. 293–300.

Liu G.P. and Daley S. (1999): *Output-model-based predictive control of unstable combustion systems using neural networks.* — Contr. Eng. Pract., Vol. 7, No. 5, pp. 591–600.

Liu G.P., Kadirkamanathan V. and Billings S.A. (1998): *Predictive control for non-linear systems using neural networks.* — Int. J. Contr., Vol. 71, No. 6, pp. 1119–1132.

Ławryńczuk M. (2003): *Nonlinear model predictive control algorithms with neural models*. — Ph.D. thesis, Warsaw University of Technology, Warsaw, Poland.

Ławryńczuk M. and Tatjewski P. (2001a): *A multivariable neural predictive control algorithm.* — Proc. IFAC *Advanced Fuzzy-Neural Control Workshop*, Valencia, Spain, pp. 191–196.

Ławryńczuk M. and Tatjewski P. (2001b): *A nonlinear predictive control algorithm for processes modelled by means of neural networks.* — Proc. 7-th Int. Conf. *Methods and Models in Automation and Robotics*, Międzyzdroje, Poland, pp. 489–494.

Ławryńczuk M. and Tatjewski P. (2002): *A computationally efficient nonlinear predictive control algorithm based on neural models.* — Proc. 8-th Int. Conf. *Methods and Models in Automation and Robotics*, Szczecin, Poland, pp. 781–786.

Ławryńczuk M. and Tatjewski P. (2004): *A stable dual-mode type nonlinear predictive control algorithm based on online linearisation and quadratic programming.* — Proc. 10-th Int. Conf. *Methods and Models in Automation and Robotics*, Międzyzdroje, Poland, pp. 503–510.

Maciejowski J.M. (2002): *Predictive Control.* — Harlow: Prentice Hall.

Marusak P. and Tatjewski P. (2002): *Stability analysis of nonlinear control systems with unconstrained fuzzy predictive controllers.* — Arch. Contr. Sci., Vol. 12, pp. 267–288.

Mayne D.Q., Rawlings J.B., Rao C.V. and Scokaert P.O.M. (2000): *Constrained model predictive control: Stability and optimality.* — Automatica, Vol. 36, No. 6, pp. 789–814.

Michalska H. and Mayne D.Q. (1993) (2000): *Robust receding horizon control of constrained nonlinear systems.* — IEEE Trans. Automat. Contr., Vol. 38, No. 11, pp. 1623–1633.

Morari M. and Lee J.H. (1999): *Model predictive control: Past, present and future.* — Comput. Chem. Eng., Vol. 23, No. 4/5, pp. 667–682.

Najim K., Rusnak A., Meszaros A. and Fikar M. (1997): *Constrained long-range predictive control based on artificial neural networks.* — Int. J. Syst. Sci., Vol. 28, No. 12, pp. 1211–1226.

Nørgaard M., Ravn O., Poulsen N.K. and Hansen L.K. (2000): *Neural Networks for Modelling and Control of Dynamic Systems.* — London: Springer.

Ortega J.G. and Camacho E.F. (1996): *Mobile robot navigation in a partially structured static environment, using neural predictive control.* — Contr. Eng. Pract., Vol. 4, No. 12, pp. 1669–1679.

Parisini T. and Sacone S. (2001): *Stable hybrid control based on discrete-event automata and receding-horizon neural regulators.* — Automatica, Vol. 37, No. 8, pp. 1279–1292.

Parisini T., Sanguineti M. and Zoppoli R. (1998): *Nonlinear stabilization by receding-horizon neural regulators.* — Int. J. Contr., Vol. 70, No. 3, pp. 341–362.

Parisini T. and Zoppoli R. (1995): *A receding-horizon regulator for nonlinear systems and a neural approximation.* — Automatica, Vol. 31, No. 10, pp. 1443–1451.

Piche S., Sayyar-Rodsari B., Johnson D. and Gerules M. (2000): *Nonlinear model predictive control using neural networks.* — IEEE Contr. Syst. Mag., Vol. 20, No. 3, pp. 56–62.

Qin S.J. and Badgwell T.A. (2003): *A survey of industrial model predictive control technology.* — Contr. Eng. Pract., Vol. 11, No. 7, pp. 733–764.

Rossiter J.A. (2003): *Model-Based Predictive Control*. — Boca Raton, FL: CRC Press.

Tatjewski P. (2002): *Advanced Control of Industrial Processes. Structures and Algorithms.* — Warsaw: Akademicka Oficyna Wydawnicza EXIT, (in Polish, revised and extended English edition in preparation).

Temeng K.O., Schnelle P.D. and McAvoy T.J. (1995): *Model predictive control of an industrial packed bed reactor using neural networks.* — J. Process Contr., Vol. 5, No. 1, pp. 19–27.

Trajanoski Z. and Wach P. (1998): *Neural predictive control for insulin delivery using the subcutaneous route.* — IEEE Trans. Biomed. Eng., Vol. 45, No. 9, pp. 1122–1134.

Vila J.P. and Wagner V. (2003): *Predictive neuro-control of uncertain systems: Design and use of a neuro-optimizer.* — Automatica, Vol. 39, No. 5, pp. 767–777.

Wang L.X. and Wan F. (2001): *Structured neural networks for constrained model predictive control.* — Automatica, Vol. 37, No. 8, pp. 1235–1243.

Yu D.L. and Gomm J.B. (2003): *Implementation of neural network predictive control to a multivariable chemical reactor.* — Contr. Eng. Pract., Vol. 11, No. 11, pp. 1315–1323.

Received: 18 October 2005
Revised: 21 December 2005