# Suboptimal Non-linear Predictive Control Based on MLP and RBF Neural Models with Measured Disturbance Compensation

Received 23th November 2007; accepted 28th January 2008.

*Maciej Ławryńczuk*

**Abstract:**

*This paper is concerned with a computationally efficient (suboptimal) non-linear Model Predictive Control (MPC) algorithm based on two types of neural models: Multilayer Perceptron (MLP) and Radial Basis Function (RBF) structures. The model takes into account not only controlled but also the uncontrolled input of the process, i.e. the measured disturbance. The algorithm is computationally efficient, because it results in a quadratic programming problem, which can be effectively solved on-line by means of a numerically reliable software subroutine. Moreover, the algorithm gives good closed-loop control performance, comparable to that obtained in the fully-fledged non-linear MPC technique, which hinges on non-linear, usually non-convex optimisation.*

***Keywords:*** *predictive control, neural networks, linearisation, quadratic programming*

## 1. Introduction

Model Predictive Control is recognised as the advanced control technique, which has been very successful in large-scale industrial applications, for example distillation columns or polymerisation reactors. MPC has influenced not only the directions of development of industrial control systems, but also research in this area [8, 24, 27, 33, 34, 36]. MPC algorithms have many advantages. First of all, they have the unique ability to take into account constraints imposed on process inputs (manipulated variables) and outputs (controlled variables) or state variables, which usually decide on quality, economic efficiency and safety of production. Secondly, MPC is very efficient in multivariable process control. Moreover, MPC can be successfully applied to processes whose numbers of input and output variables are different, and to processes whose control is difficult (e.g. processes with the inverse step response).

MPC closely co-operates with economic optimisation in order to maximise production profit. Different control structures can be used:

a) The classical multilayer control structure, in which the economic optimisation layer is activated less frequently than the MPC layer [4, 6, 13, 14, 36].
b) The multilayer structure with an auxiliary steady-state target optimisation, which recalculates the operating point as frequently as the MPC layer executes [3, 11, 13, 14, 21, 36, 37].
c) The integrated structure, in which economic optimisation and MPC optimisation tasks are integrated into one optimisation problem [13, 14, 15, 21, 36, 40, 42].
d) The integrated structure with an internal, unconstrained controller [16, 35].

In many practical cases disturbances (i.e. uncontrolled but measured inputs), for example flow rates, properties of feed and energy streams etc., vary significantly and not much slower than the dynamics of the controlled process. Changes of disturbances entail the necessity of solving on-line the economic optimisation problem in order to find the optimal set point for the MPC layer. Naturally, it is best when the MPC algorithm can compensate the influence of disturbances, if only dynamics of disturbances and control channels and available process models enable an efficient realisation of the compensation.

This paper describes a computationally efficient MPC algorithm with Non-linear Prediction and Linearisation (MPC-NPL) [17, 19, 22, 36, 38] with measured disturbance compensation [18]. The algorithm uses a neural model of the process on-line twice at each sampling instant, namely for determining a local linearisation and a non-linear free trajectory. Two most popular types of neural models are used: Multilayer Perceptron (MLP) and Radial Basis Function (RBF) structures. The MPC-NPL algorithm is computationally efficient because it needs solving on-line a quadratic programming problem. Although sub optimal, it gives good closed-loop performance, comparable to that obtained in the fully-fledged non-linear MPC algorithm with non-linear optimisation, in which at each sampling instant a non-linear, usually non-convex and even multimodal optimisation problem has to be solved on-line. For such problems there are no sufficiently fast and reliable optimisation algorithms, i.e. those that would be able to determine the global optimal solution at each sampling instant and within a predefined time limit, as it is required in on-line control. Gradient-based optimisation techniques may terminate in local minima while global ones substantially increase the computational burden, yet they still give no guarantee that the global solution is found [25].

Structure of the model determines accuracy, computational burden and reliability of the MPC algorithm. Fundamental (first-principles) models, although potentially very precise, are usually not suitable for on-line control since they are very complicated and may frequently lead to numerical problems resulting, for example, from ill-conditioning. Among many structures of empirical models, neural networks deserve consideration because they have the following advantages:

a) Are universal approximators [9], hence are able to approximate precisely non-linear behaviour of technological dynamic processes [10, 28].
b) Efficient identification algorithms and structure optimisation techniques have been developed [7, 29].
c) Have simple structure and relatively small number of parameters (unlike fuzzy models do not suffer from "the curse of dimensionality").
d) Numerical problems typical of MPC algorithms with comprehensive fundamental models are not encountered because neural models directly describe input-output relations of process variables, complicated systems of differential and algebraic equations do not have to be solved on-line.
e) Can be easily incorporated into different MPC algorithms and efficiently used on-line [1, 2, 5, 10, 12, 17, 18, 19, 20, 22, 28, 30, 31, 32, 36, 38, 41].

## 2. Model predictive control

Although many versions of MPC algorithms have been developed over the years, the main idea (i.e. the explicit application of a process model, the receding horizon and optimisation of a cost function) is always the same [24, 34, 36]. At each consecutive sampling instant $k$ a set of future control increments is found

$$\Delta \boldsymbol{u}(k) = \left[ \Delta u(k \mid k) \quad \ldots \quad \Delta u(k + N_u - 1 \mid k) \right]^T \tag{1}$$

It is assumed that $\Delta u(k+p \mid k)=0$ for $p \geq N_u$, where $N_u$ is the control horizon. The objective of the algorithm is to minimise the differences between the reference trajectory $y^{ref}(k+p \mid k)$ and the predicted values of the output $\hat{y}(k+p \mid k)$ over the prediction horizon $N$, i.e. $p=1,\ldots,N$. These predictions are calculated on-line using a dynamic model of the process. The following quadratic cost function is usually used

$$J(k) = \sum_{p=1}^{N} \mu_p \left( y^{ref}(k+p \mid k) - \hat{y}(k+p \mid k) \right)^2 + \sum_{p=0}^{N_u-1} \lambda_p \left( \Delta u(k+p \mid k) \right)^2 \tag{2}$$

where $\mu_p \geq 0$, $\lambda_p > 0$ are weighting factors. Typically, $N_u<N$, which decreases the dimensionality of the optimisation problem and leads to smaller computational load. Only the first element of the determined sequence (1) is applied to the process, the control law is then

$$u(k \mid k) = \Delta u(k \mid k) + u(k-1) \tag{3}$$

At the next sampling instant, $k+1$, the measurement of the process output is updated, the whole procedure is repeated.

### 2.1. Model predictive control optimisation problem

As emphasised in the Introduction, constraints handling is one the most important advantages of MPC algorithms and it determined their great success. In constrained MPC algorithms, future control increments are found as the solution to the following optimisation problem

$$\min_{\Delta \boldsymbol{u}(k)} \{ J(k) = \sum_{p=1}^{N} \mu_p \left( y^{ref}(k+p \mid k) - \hat{y}(k+p \mid k) \right)^2 + \sum_{p=0}^{N_u-1} \lambda_p \left( \Delta u(k+p \mid k) \right)^2 \} \tag{4}$$

subject to :

$$\begin{aligned} & u_{\min} \leq u(k+p \mid k) \leq u_{\max} && p = 0, \ldots, N_u - 1 \\ & -\Delta u_{\max} \leq \Delta u(k+p \mid k) \leq \Delta u_{\max} && p = 0, \ldots, N_u - 1 \\ & y_{\min} \leq \hat{y}(k+p \mid k) \leq y_{\max} && p = 1, \ldots, N \end{aligned}$$

Defining vectors of length $N$

$$\begin{aligned} \boldsymbol{y}^{ref}(k) &= \left[ y^{ref}(k+1 \mid k) \quad \ldots \quad y^{ref}(k+N \mid k) \right]^T \\ \hat{\boldsymbol{y}}(k) &= \left[ \hat{y}(k+1 \mid k) \quad \ldots \quad \hat{y}(k+N \mid k) \right]^T \\ \boldsymbol{y}_{\min} &= \left[ y_{\min} \quad \ldots \quad y_{\min} \right]^T \\ \boldsymbol{y}_{\max} &= \left[ y_{\max} \quad \ldots \quad y_{\max} \right]^T \end{aligned} \tag{5}$$

and vectors of length $N_u$

$$\begin{aligned} \boldsymbol{y}^{ref}(k) &= \left[ y^{ref}(k+1 \mid k) \quad \ldots \quad y^{ref}(k+N \mid k) \right]^T \\ \hat{\boldsymbol{y}}(k) &= \left[ \hat{y}(k+1 \mid k) \quad \ldots \quad \hat{y}(k+N \mid k) \right]^T \\ \boldsymbol{y}_{\min} &= \left[ y_{\min} \quad \ldots \quad y_{\min} \right]^T \\ \boldsymbol{y}_{\max} &= \left[ y_{\max} \quad \ldots \quad y_{\max} \right]^T \end{aligned} \tag{6}$$

the MPC optimisation problem (4) can be rewritten in a compact vector-matrix form as

$$\min_{\Delta \boldsymbol{u}(k)} \{ J(k) = \left\| \boldsymbol{y}^{ref}(k) - \hat{\boldsymbol{y}}(k) \right\|_{\boldsymbol{M}}^2 + \left\| \Delta \boldsymbol{u}(k) \right\|_{\boldsymbol{\Lambda}}^2 \}$$

subject to :

$$\begin{aligned} & \boldsymbol{u}_{\min} \leq \boldsymbol{J} \Delta \boldsymbol{u}(k) + \boldsymbol{u}^{k-1} \leq \boldsymbol{u}_{\max} \\ & -\Delta \boldsymbol{u}_{\max} \leq \Delta \boldsymbol{u}(k) \leq \Delta \boldsymbol{u}_{\max} \\ & \boldsymbol{y}_{\min} \leq \hat{\boldsymbol{y}}(k) \leq \boldsymbol{y}_{\max} \end{aligned} \tag{7}$$

where

$$\boldsymbol{J} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 1 & 1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 1 \end{bmatrix} \tag{8}$$

is the matrix of dimensionality $N_u \times N_u$, $M$ and $\Lambda$ are diagonal matrices of dimensionality $N \times N$ and $N_u \times N_u$ comprised of coefficients $\mu_p$, $\lambda_p$, respectively.

If output constraints have to be taken into account, the MPC optimisation task (7) may be affected by the infeasibility problem. So as to cope with such a situation, the original output constraints have to be softened by using slack variables [24, 36]. Using a quadratic penalty for constraint violations the MPC optimisation problem (7) is then

$$\min_{\Delta \boldsymbol{u}(k),\, \boldsymbol{\varepsilon}_{\min}(k),\, \boldsymbol{\varepsilon}_{\max}(k)} \{ J(k) = \left\| \boldsymbol{y}^{ref}(k) - \hat{\boldsymbol{y}}(k) \right\|_{\boldsymbol{M}}^2 + \left\| \Delta \boldsymbol{u}(k) \right\|_{\boldsymbol{\Lambda}}^2 + \rho_{\min} \left\| \boldsymbol{\varepsilon}_{\min}(k) \right\|^2 + \rho_{\max} \left\| \boldsymbol{\varepsilon}_{\max}(k) \right\|^2 \} \tag{9}$$

subject to :

$$\begin{aligned}
&\boldsymbol{u}_{\min} \le \boldsymbol{J}\Delta\boldsymbol{u}(k) + \boldsymbol{u}^{k-1} \le \boldsymbol{u}_{\max} \\
&-\Delta\boldsymbol{u}_{\max} \le \Delta\boldsymbol{u}(k) \le \Delta\boldsymbol{u}_{\max} \\
&\boldsymbol{y}_{\min} - \boldsymbol{\varepsilon}_{\min}(k) \le \hat{\boldsymbol{y}}(k) \le \boldsymbol{y}_{\max} + \boldsymbol{\varepsilon}_{\max}(k) \\
&\boldsymbol{\varepsilon}_{\min}(k) \ge 0,\ \boldsymbol{\varepsilon}_{\max}(k) \ge 0
\end{aligned}$$

where $\varepsilon_{min}(k)$, $\varepsilon_{max}(k)$ are vectors of length $N$ comprising the slack variables and $\rho_{min}$, $\rho_{max}$ are positive weights.

## 3. MPC-NPL algorithm with neural models

### 3.1. Structures of neural models

Let the Single-Input Single-Output (SISO) processes be described by the following discrete-time equation

$$\begin{aligned}
y(k) = f(\boldsymbol{x}(k)) = f(&u(k-\tau),\ldots,u(k-n_B), \\
&y(k-1),\ldots,y(k-n_A), \\
&h(k-\tau_h),\ldots,h(k-n_C))
\end{aligned} \tag{10}$$

where $u$ is the input of the process, $y$ is the output, $h$ is the uncontrolled but measured input (the disturbance), $f : \Re^{n_A+n_B+n_C-\tau-\tau_h+2} \to \Re \in C^1$, $\tau \le n_B$, $\tau \le n_C$. When the Multilayer Perceptron (MLP) feedforward neural network with one hidden layer and a linear output [7, 29] is used as the function $f$ in (10), output of the model is

$$y(k) = w_0^2 + \sum_{i=1}^{K} w_i^2 v_i(k) = w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(z_i(k)) \tag{11}$$

where $z_i(k)$ and $v_i(k)$ are the sum of inputs and the output of the $i$-th hidden node, respectively, $\varphi$: $\Re \to \Re$ is the non-linear transfer function (e.g. hyperbolic tangent), $K$ is the number of hidden nodes. Recalling the input arguments of the general non-linear model (10) one has

$$\begin{aligned}
z_i(k) = w_{i,0}^1 &+ \sum_{j=1}^{I_u} w_{i,j}^1 u(k-\tau+1-j) + \sum_{j=1}^{n_A} w_{i,I_u+j}^1 y(k-j) \\
&+ \sum_{j=1}^{I_h} w_{i,I_u+n_A+j}^1 h(k-\tau_h+1-j)
\end{aligned} \tag{12}$$

Weights of the network are denoted by $w_{i,j}^1$, $i=1,\ldots,K$, $j=0,\ldots,I$, and $w_i^2$, $i=0,\ldots,K$, for the first and the second layer, respectively. Total number of input nodes is $I = n_A+n_B+n_C-\tau-\tau_h+2$. The number of input nodes, which depends on the input signal $u$, is $I_u = n_B-\tau+1$, the number of input nodes, which depends on the disturbance signal $h$, is $I_h = n_C-\tau_h+1$. Structure of the MLP neural network model is shown in Fig. 1.

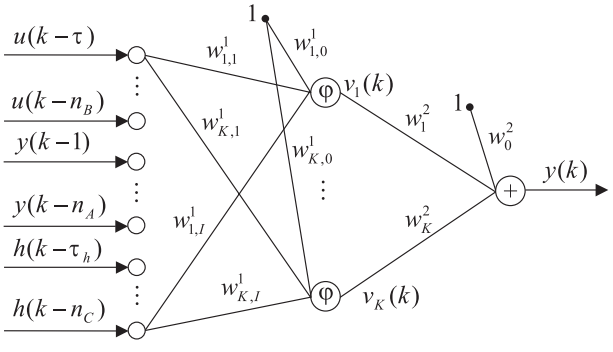


*Fig. 1. Structure of the MLP neural network model.*

When the Radial Basis Function (RBF) feedforward neural network containing one hidden layer with Gaussian functions and a linear output is used as the function $f$ in (10), output of the model is

$$\begin{aligned}
y(k) = f(\boldsymbol{x}(k)) &= w_0 + \sum_{i=1}^{K} w_i \exp(-\|\boldsymbol{x}(k) - \boldsymbol{c}_i\|_{\boldsymbol{Q}_i}) \\
&= w_0 + \sum_{i=1}^{K} w_i \exp(-z_i(k))
\end{aligned} \tag{13}$$

where $K$ is the number of hidden nodes, $\boldsymbol{c}_i$ and the diagonal weighting matrices $\boldsymbol{Q}_i = diag(q_{i,1},\ldots,q_{i,I})$ describe centres and widths of the nodes, respectively, $i=1,\ldots,K$. The model (13) is sometimes named the Hyper Radial Basis Function (HRBF) neural network in contrast to the ordinary RBF neural networks in which widths of the nodes are constant [29]. Let $z_i(k)$ be the sum of inputs of the $i$-th hidden node. Recalling arguments of the model (10) one has

$$\begin{aligned}
z_i(k) = &\sum_{j=1}^{I_u} q_{i,j} (u(k-\tau+1-j) - c_{i,j})^2 \\
&+ \sum_{j=1}^{n_A} q_{i,I_u+j} (y(k-j) - c_{i,I_u+j})^2 \\
&+ \sum_{j=1}^{I_h} q_{i,I_u+n_A+j} (h(k-\tau_h+1-j) - c_{i,I_u+n_A+j})^2
\end{aligned} \tag{14}$$

Structure of the RBF neural network model is shown in Fig. 2.

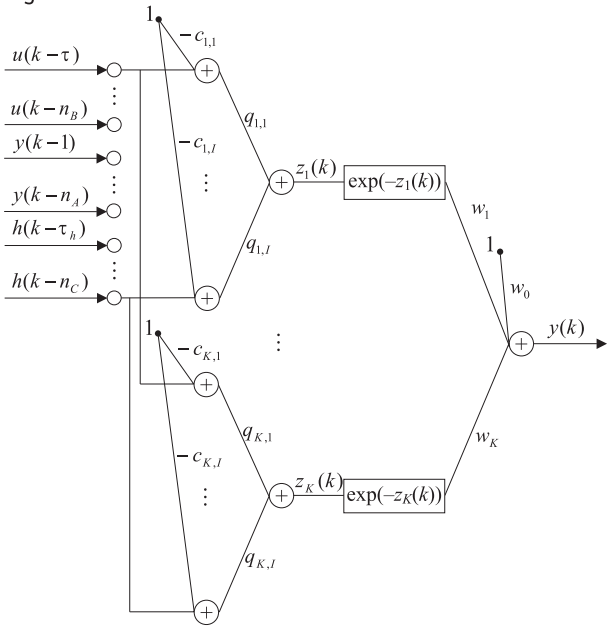


*Fig. 2. Structure of the RBF neural network model.*

### 3.2. MPC-NPL optimisation problem

When for prediction a non-linear neural model of the process is used without any simplifications, at each sampling instant a non-linear optimisation problem (9) has to be solved on-line. In order to reduce the computational burden and increase reliability of the control algorithm, the sub optimal MPC-NPL algorithm is adopted [17, 19, 20, 22, 36, 38]. At each sampling instant $k$ the neural model is used on-line twice: to determine a local linearisation and a non-linear free trajectory. It is assumed that the output prediction can be expressed as the sum of the

forced trajectory (response), which depends only on the future (i.e. on future input moves $\Delta\boldsymbol{u}(k)$) and the free trajectory $\boldsymbol{y}^0(k)$, which depends only on the past

$$\hat{\boldsymbol{y}}(k) = \boldsymbol{G}(k)\Delta\boldsymbol{u}(k) + \boldsymbol{y}^0(k) \tag{15}$$

where the dynamic matrix of dimensionality $N\times N_u$ is composed of step response coefficients of the linearised model

$$\boldsymbol{G}(k) = \begin{bmatrix} s_1(k) & 0 & \dots & 0 \\ s_2(k) & s_1(k) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s_N(k) & s_{N-1}(k) & \dots & s_{N-N_u+1}(k) \end{bmatrix} \tag{16}$$

and the free trajectory is a vector of length $N$

$$\boldsymbol{y}^0(k) = \begin{bmatrix} y^0(k+1\,|\,k) & \dots & y^0(k+N\,|\,k) \end{bmatrix}^T \tag{17}$$

Let $s_j(k)$ denote the $j$-th step response coefficient, $j=1,...,N$, calculated at the current time instant $k$ using the linearised model

$$s_j(k) = \sum_{i=1}^{\min(j,n_B)} b_i(k) - \sum_{i=1}^{\min(j-1,n_A)} a_i(k)\, s_{j-i}(k) \tag{18}$$

where $a_i(k)$, $b_i(k)$ are coefficients of the linearised model.

Of course, unlike MPC algorithms with linear models, the plant and its model are non-linear; hence the superposition equation (15) cannot be exactly satisfied. On the one hand, the sub optimal prediction calculated from (15) is different from the optimal one determined from the non-linear neural model as it is done in MPC algorithms with non-linear optimisation [17, 36, 38]. On the other hand, thanks to using (15), the optimisation problem (9) becomes the following quadratic programming task

$$\begin{aligned} \min_{\Delta\boldsymbol{u}(k),\,\varepsilon_{\min}(k),\,\varepsilon_{\max}(k)} \{ J(k) = & \left\| \boldsymbol{y}^{ref}(k) - \boldsymbol{G}\Delta\boldsymbol{u}(k) - \boldsymbol{y}^0(k) \right\|_{\boldsymbol{M}}^2 \\ & + \left\| \Delta\boldsymbol{u}(k) \right\|_{\Lambda}^2 + \rho_{\min} \left\| \varepsilon_{\min}(k) \right\|^2 \\ & + \rho_{\max} \left\| \varepsilon_{\max}(k) \right\|^2 \} \end{aligned} \tag{19}$$

subject to :

$$\begin{aligned} & \boldsymbol{u}_{\min} \le \boldsymbol{J}\Delta\boldsymbol{u}(k) + \boldsymbol{u}^{k-1} \le \boldsymbol{u}_{\max} \\ & -\Delta\boldsymbol{u}_{\max} \le \Delta\boldsymbol{u}(k) \le \Delta\boldsymbol{u}_{\max} \\ & \boldsymbol{y}_{\min} - \varepsilon_{\min}(k) \le \boldsymbol{G}(k)\Delta\boldsymbol{u}(k) + \boldsymbol{y}^0(k) \le \boldsymbol{y}_{\max} + \varepsilon_{\max}(k) \\ & \varepsilon_{\min}(k) \ge 0,\ \varepsilon_{\max}(k) \ge 0 \end{aligned}$$

Structure of the MPC-NPL algorithm is depicted in Fig. 3. At each sampling instant $k$ the following steps are repeated:

1. Linearisation: obtain the dynamic matrix $\boldsymbol{G}(k)$ .
2. Calculate the non-linear free trajectory $\boldsymbol{y}^0(k)$.
3. Solve the quadratic programming problem (19) to determine $\Delta\boldsymbol{u}(k)$.
4. Apply $u(k)=u(k\,|\,k)+u(k-1)$.
5. Set $k:=k+1$, go to step 1.

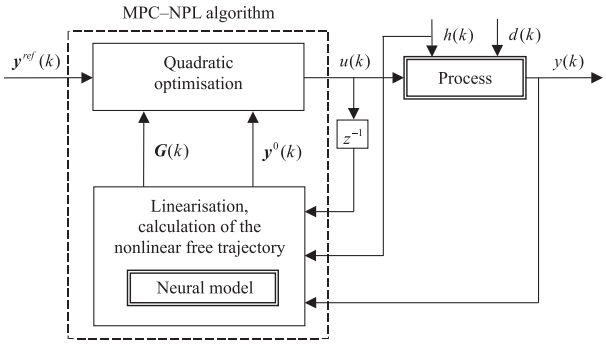


*Fig. 3. Structure of the MPC algorithm with Non-linear Prediction and Linearisation (MPC-NPL).*

Both the dynamic matrix and the non-linear free trajectory are calculated on-line using the neural model of the process.

Defining the vector of decision variables

$$\boldsymbol{x}(k) = \begin{bmatrix} \Delta\boldsymbol{u}^T(k) & \boldsymbol{\varepsilon}_{\min}^T(k) & \boldsymbol{\varepsilon}_{\max}^T(k) \end{bmatrix}^T,$$

the MPC-NPL optimisation problem (19) can easily be written in a standard quadratic programming form

$$\min_{\boldsymbol{x}(k)} \left\{ J(k) = \frac{1}{2}\boldsymbol{x}^T(k)\boldsymbol{H}_{QP}(k)\boldsymbol{x}(k) + \boldsymbol{f}_{QP}^T(k)\boldsymbol{x}(k) \right\} \tag{20}$$
$$\boldsymbol{A}_{QP}(k)\boldsymbol{x}(k) \le \boldsymbol{b}_{QP}(k)$$

where

$$\begin{aligned} & \Delta\boldsymbol{u}(k) = \boldsymbol{M}_1\boldsymbol{x}(k), \quad \boldsymbol{M}_1 = \begin{bmatrix} \boldsymbol{I}_{N_u\times N_u} & \boldsymbol{0}_{N_u\times 2N} \end{bmatrix} \\ & \boldsymbol{\varepsilon}_{\min}(k) = \boldsymbol{M}_2\boldsymbol{x}(k), \quad \boldsymbol{M}_2 = \begin{bmatrix} \boldsymbol{0}_{N\times N_u} & \boldsymbol{I}_{N\times N} & \boldsymbol{0}_{N\times N} \end{bmatrix} \\ & \boldsymbol{\varepsilon}_{\max}(k) = \boldsymbol{M}_3\boldsymbol{x}(k), \quad \boldsymbol{M}_3 = \begin{bmatrix} \boldsymbol{0}_{N\times N_u} & \boldsymbol{0}_{N\times N} & \boldsymbol{I}_{N\times N} \end{bmatrix} \end{aligned} \tag{21}$$

The cost function is defined by

$$\begin{aligned} \boldsymbol{H}_{QP}(k) = & \ 2(\boldsymbol{M}_1^T\boldsymbol{G}^T(k)\boldsymbol{M}\boldsymbol{G}(k)\boldsymbol{M}_1 + \boldsymbol{M}_1^T\boldsymbol{\Lambda}\boldsymbol{M}_1) + \\ & + 2\rho_{\min}\boldsymbol{M}_2^T\boldsymbol{M}_2 + 2\rho_{\max}\boldsymbol{M}_3^T\boldsymbol{M}_3 \end{aligned} \tag{22}$$

$$\boldsymbol{f}_{QP}(k) = -2\boldsymbol{M}_1^T\boldsymbol{G}^T(k)\boldsymbol{M}(\boldsymbol{y}^{ref}(k) - \boldsymbol{y}^0(k)) \tag{23}$$

whereas the constraints are defined by

$$\boldsymbol{A}_{QP}(k) = \begin{bmatrix} -\boldsymbol{J}\boldsymbol{M}_1 \\ \boldsymbol{J}\boldsymbol{M}_1 \\ -\boldsymbol{G}(k)\boldsymbol{M}_1 - \boldsymbol{M}_2 \\ \boldsymbol{G}(k)\boldsymbol{M}_1 - \boldsymbol{M}_3 \\ -\boldsymbol{M}_1 \\ \boldsymbol{M}_1 \\ -\boldsymbol{M}_2 \\ -\boldsymbol{M}_3 \end{bmatrix}, \quad \boldsymbol{b}_{QP}(k) = \begin{bmatrix} -\boldsymbol{u}_{\min} + \boldsymbol{u}^{k-1} \\ \boldsymbol{u}_{\max} - \boldsymbol{u}^{k-1} \\ -\boldsymbol{y}_{\min} + \boldsymbol{y}^0(k) \\ \boldsymbol{y}_{\max} - \boldsymbol{y}^0(k) \\ \Delta\boldsymbol{u}_{\max} \\ \Delta\boldsymbol{u}_{\max} \\ \boldsymbol{0}_{N\times 1} \\ \boldsymbol{0}_{N\times 1} \end{bmatrix} \tag{24}$$

### 3.3. Linearisation of neural models

The linearisation point is defined as a vector comprised of past input, output and disturbance signal values corresponding to the arguments of the non-linear model (10)

$$\bar{\boldsymbol{x}}(k)=\begin{bmatrix} x_1(k) \\ \vdots \\ x_{n_A+n_B+n_C-\tau-\tau_h+2}(k) \end{bmatrix}=\begin{bmatrix} u(k-\tau) \\ \vdots \\ u(k-n_B) \\ y(k-1) \\ \vdots \\ y(k-n_A) \\ h(k-\tau_h) \\ \vdots \\ h(k-n_C) \end{bmatrix} \tag{25}$$

Using Taylor series expansion, the linear approximation of the non-linear model (10) derived at the current sampling instant $k$ is

$$y(k)=f(\bar{\boldsymbol{x}}(k))+\sum_{l=1}^{n_B}b_l(\bar{\boldsymbol{x}}(k))(u(k-l)-x_l(k)) -\sum_{l=1}^{n_A}a_l(\bar{\boldsymbol{x}}(k))(y(k-l)-x_{n_A+\tau-1+l}(k)) \tag{26}$$

Coefficients of the linearised model are calculated from

$$a_l(\bar{\boldsymbol{x}}(k))=-\frac{\partial f(\bar{\boldsymbol{x}}(k))}{\partial y(k-l)} \quad l=1,\ldots,n_A \tag{27}$$

and

$$b_l(\bar{\boldsymbol{x}}(k))=\begin{cases} 0 & l=1,\ldots,\tau-1 \\ \dfrac{\partial f(\bar{\boldsymbol{x}}(k))}{\partial u(k-l)} & l=\tau,\ldots,n_B \end{cases} \tag{28}$$

Considering the MLP neural model given by (11) and (12)

$$a_l(k)=-\sum_{i=1}^{K}w_i^2\frac{d\varphi(z_i(\bar{\boldsymbol{x}}(k)))}{dz_i(\bar{\boldsymbol{x}}(k))}w_{i,I_u+l}^1 \quad l=1,\ldots,n_A \tag{29}$$

and

$$b_l(k)=\begin{cases} 0 & l=1,\ldots,\tau-1 \\ \displaystyle\sum_{i=1}^{K}w_i^2\frac{d\varphi(z_i(\bar{\boldsymbol{x}}(k)))}{dz_i(\bar{\boldsymbol{x}}(k))}w_{i,l-\tau+1}^1 & l=\tau,\ldots,n_B \end{cases} \tag{30}$$

If hyperbolic tangent is used as the non-linear transfer function φ in the hidden layer of the neural model, one has

$$\frac{d\varphi(z_i(\bar{\boldsymbol{x}}(k)))}{dz_i(\bar{\boldsymbol{x}}(k))}=1-\tanh^2(z_i(\bar{\boldsymbol{x}}(k))) \tag{31}$$

The linearisation point given by (25) is not influenced by the most recent output value $y(k)$, which is available for measurement. Therefore, it is recommended to use

$$\bar{\boldsymbol{x}}(k)=\begin{bmatrix} x_1(k) \\ \vdots \\ x_{n_A+n_B+n_C-\tau-\tau_h+2}(k) \end{bmatrix}=\begin{bmatrix} u(k-\tau+1) \\ \vdots \\ u(k-n_B+1) \\ y(k) \\ \vdots \\ y(k-n_A+1) \\ h(k-\tau_h+1) \\ \vdots \\ h(k-n_C+1) \end{bmatrix} \tag{32}$$

If $\tau=1$, for linearisation purposes one may set $u(k|k):=u(k-1)$ or $u(k|k):=u(k|k-1)$.

Taking into account the structure of the RBF neural model described by (13) and (14), coefficients of the linearised dynamic model are calculated on-line from

$$a_l(k)=2\sum_{i=1}^{K}w_i\exp(-z_i(\bar{\boldsymbol{x}}(k)))q_{i,I_u+l}(y(k-l)-c_{i,I_u+l}) \quad l=1,\ldots,n_A \tag{33}$$

and

$$b_l(k)=\begin{cases} 0 & l=1,\ldots,\tau-1 \\ -2\displaystyle\sum_{i=1}^{K}w_i\exp(-z_i(\bar{\boldsymbol{x}}(k)))q_{i,l-\tau+1}(u(k-l)-c_{i,l-\tau+1}) & l=\tau,\ldots,n_B \end{cases} \tag{34}$$

### 3.4. Calculation of the non-linear free trajectory

The general prediction equation for $p=1,\ldots,N$ is

$$\hat{y}(k+p|k)=y(k+p|k)+d(k) \tag{35}$$

where the quantities $y(k+p|k)$ are calculated from the non-linear neural model used for the sampling instant $k+p$ at the current sampling instant $k$. In case of the MLP neural model from (11)

$$y(k+p|k)=w_0^2+\sum_{i=1}^{K}w_i^2\varphi(z_i(k+p|k)) \tag{36}$$

The "DMC type" disturbance model is used in (35) [24, 36]. The unmeasured disturbance $d(k)$ affecting the process at the sampling instant $k$ is assumed to be constant over the prediction horizon. Its value is calculated from

$$d(k)=y(k)-y(k|k-1) \tag{37}$$

where $y(k)$ is measured whereas the quantity $y(k|k-1)$ is calculated from the non-linear neural model. From (11) one has

$$d(k)=y(k)-\left(w_0^2+\sum_{i=1}^{K}w_i^2\varphi(z_i(k))\right) \tag{38}$$

Considering the prediction over the horizon $N$, the quantities $z_i(k+p|k)$ and consequently $\hat{y}(k+p|k)$ depend on future values of the control signal (i.e. decision variables of the algorithm $u(k|k), u(k+1|k),\ldots$), values of the control signal applied to the plant at previous sampling instants ($u(k-1)$, $u(k-2),\ldots$), future output predictions ( $\hat{y}(k+1|k)$, $\hat{y}(k+2|k),\ldots$), measured values of the plant output signal ($y(k-1), y(k-2),\ldots$), future values of the disturbance $h$ ($h(k+1|k)$, $h(k+2|k),\ldots$) and values of the disturbance measured up to the current sampling instant $k$ ($h(k), h(k-1),\ldots$). From (12) one has

$$z_i(k+p|k)=w_{i,0}^1+\sum_{j=1}^{I_{uf}(p)}w_{i,j}^1u(k-\tau+1-j+p|k) +\sum_{j=I_{uf}(p)+1}^{I_u}w_{i,j}^1u(k-\tau+1-j+p) +\sum_{j=1}^{I_{yp}(p)}w_{i,I_u+j}^1\hat{y}(k-j+p|k) \tag{39}$$

$$+\sum_{j=I_{yp}(p)+1}^{n_A} w^1_{i,I_u+j}\, y(k-j+p) \qquad \text{(cd. 39)}$$
$$+\sum_{j=1}^{I_{hf}(p)} w^1_{i,I_u+n_A+j} h(k-\tau_h+1-j+p\,|\,k)$$
$$+\sum_{j=I_{hf}(p)+1}^{I_h} w^1_{i,I_u+n_A+j} h(k-\tau_h+1-j+p)$$

where $I_{uf}(p)=\max(\min(p-\tau+1, I_u),0)$ is the number of the network input nodes, which depends on future control signal, $I_{yp}(p)=\min(p-1, n_A)$ is the number of input nodes depending on output predictions, $I_{hf}(p)=\max(\min(p-\tau_h, I_h),0)$ is the number of input nodes, which depends on future disturbance signal. Since future values of the disturbance are usually not known at the sampling instant $k$, it is assumed that $h(k+p\,|\,k)=h(k)$ for $p\geq 1$.

The non-linear free trajectory $y^0(k+p\,|\,k)$ is calculated recursively on-line from the general prediction equation (35) considering only influence of the past. From (36)

$$y^0(k+p\,|\,k) = w^2_0 + \sum_{i=1}^{K} w^2_i \varphi(z^0_i(k+p\,|\,k)) + d(k) \qquad (40)$$

The quantities $z^0_i(k+p\,|\,k)$ are determined from (39) assuming no changes in the control signal from the sampling instant $k$ onwards and replacing predicted output signals by corresponding values of the free trajectory

$$z^0_i(k+p\,|\,k) = w^1_{i,0} + \sum_{j=1}^{I_{uf}(p)} w^1_{i,j} u(k-1) \qquad (41)$$
$$+\sum_{j=I_{uf}(p)+1}^{I_u} w^1_{i,j} u(k-\tau+1-j+p)$$
$$+\sum_{j=1}^{I_{yp}(p)} w^1_{i,I_u+j}\, y^0(k-j+p\,|\,k)$$
$$+\sum_{j=I_{yp}(p)+1}^{n_A} w^1_{i,I_u+j}\, y(k-j+p)$$
$$+\sum_{j=1}^{I_{hf}(p)} w^1_{i,I_u+n_A+j} h(k)$$
$$+\sum_{j=I_{hf}(p)+1}^{I_h} w^1_{i,I_u+n_A+j} h(k-\tau_h+1-j+p)$$

In case of the RBF neural model, form (13) and (37), the unmeasured disturbance is

$$d(k) = y(k) - \left( w_0 + \sum_{i=1}^{K} w_i \exp(-z_i(k)) \right) \qquad (42)$$

Analogously to (41), the quantities are determined from

$$z^0_i(k+p\,|\,k) = w^1_{i,0} + \sum_{j=1}^{uf\ p} q_{i,j} (u(k-1)-c_{i,j})^2 \qquad (43)$$
$$+\sum_{j=I_{uf}(p)+1}^{I_u} q_{i,j} (u(k-\tau+1-j+p)-c_{i,j})^2$$
$$+\sum_{j=1}^{I_{yp}(p)} q_{i,I_u+j} (y^0(k-j+p\,|\,k)-c_{i,I_u+j})^2$$
$$+\sum_{j=I_{yp}(p)+1}^{n_A} q_{i,I_u+j} (y(k-j+p)-c_{i,I_u+j})^2$$
$$+\sum_{j=1}^{I_{hf}(p)} q_{i,I_u+n_A+j} (h(k)-c_{i,I_u+n_A+j})^2$$
$$+\sum_{j=I_{hf}(p)+1}^{I_h} q_{i,I_u+n_A+j} (h(k-\tau_h+1-j+p)-c_{i,I_u+n_A+j})^2$$

## 4. Experiments

### 4.1. The polymerisation reactor

The control process under consideration is a polymerisation reaction taking place in a jacketed continuous stirred tank reactor depicted in Fig. 4 [26]. The reaction is the free-radical polymerisation of methyl methacrylate with azo-bis-isobutyronitrile as initiator and toluene as solvent. The output *NAMW* (Number Average Molecular Weight) is controlled by manipulating the inlet initiator flow rate $F_I$. The monomer flow rate $F$ is the measured disturbance. The reactor is frequently used as a benchmark process for comparing non-linear control strategies. MPC with a constant linear model is unable to control the plant effectively (i.e. it is unacceptably oscillatory or slow) [17, 18, 19, 20, 36]. Although the process is open-loop stable (it is of inertia type), it is difficult to control because its dynamic and steady-state properties are significantly non-linear. Hence, in this case it is justified to use non-linear models of the process and non-linear MPC algorithms. Different suboptimal MPC techniques with on-line quadratic programming were applied to the reactor yielding good control quality, comparable to that obtained in MPC with on-line non-linear optimisation [17, 18, 19, 20, 36].

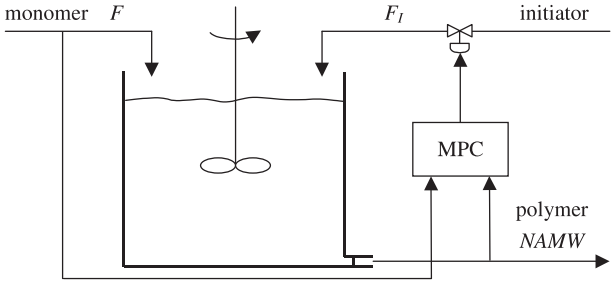


*Fig. 4. Polymerisation reactor control system structure.*

### 4.2. Models and compared MPC algorithms

The fundamental model [26] is used as the real process during simulations. An identification procedure is carried out, as a result of which two MLP empirical models are obtained:

a) a neural model without the measured disturbance

$$y(k) = f(u(k-2), y(k-1), y(k-2)) \qquad (44)$$

b) a neural model with the measured disturbance

$$y(k) = f(u(k-2), y(k-1), y(k-2), h(k-1), h(k-2)) \qquad (45)$$

where $u=F_I$, $y=NAMW$, $h=F$. Neural model have $K=5$ and $K=6$ hidden nodes, respectively. Empirical models have input arguments determined by $\tau=\tau_h=n_A=n_B=$

$n_C=2$. To demonstrate efficiency of the proposed sub optimal MPC-NPL algorithm with measured disturbance compensation three algorithms are compared:

1. MPC-NPLa: the sub optimal MPC-NPL algorithm without disturbance compensation (the model (44) is used).
2. MPC-NPLb: the sub optimal MPC-NPL algorithm with disturbance compensation (the model (45) is used).
3. MPC-NO: the "optimal" but computationally prohibitive MPC algorithm with Non-linear Optimisation with disturbance compensation (the model (45) is used).

Parameters of all three MPC algorithms are the same. The horizons are $N=10$, $N_u=3$, the weighting matrices $M_p=I$ and $\Lambda_p=\lambda I$, $\lambda=0.2$. The manipulated variable is constrained, $F_{I\min}=0.003$, $F_{I\max}=0.06$, the sampling time is 1.8 min.

### 4.3. Control accuracy

The reference trajectory is constant $NAMW^{ref}=20000$. Four experiments are carried out:

- Experiment 1: $F$ changes from 1 to 0.5 m$^3$/h at the sampling instant $k=5$.
- Experiment 2: $F$ changes from 1 to 2 m$^3$/h at the sampling instant $k=5$.
- Experiment 3: a series of stochastic step changes in $F$ is used as shown in Fig. 7.
- Experiment 4: sinusoidal changes in F are used

$$F(k)=\begin{cases}1\ \text{m}^3/\text{h} & k=1,\ldots,4\\ 1-0.7(\sin(0.07k)-\\ -\sin(0.35))\ \text{m}^3/\text{h} & k=5,\ldots,100\end{cases} \tag{46}$$

Simulation results of compared MPC algorithms are depicted in Fig. 5, 6, 8, 9. When the MPC-NPLa algorithm is implemented without disturbance compensation, responses of the system are relatively slow, amplitudes are quite big. On the contrary, when disturbance compensation is used in the MPC-NPLb algorithm, responses of the process are reasonably faster; amplitudes of the changes are significantly smaller. It is because the MPC-NPLb algorithm takes the disturbance into account during control value calculation. As a result, the manipulated variable $F_I$ changes faster in comparison to the MPC-NPLa scheme. Moreover, the closed-loop performance obtained in the sub optimal MPC-NPLb algorithm with quadratic programming is very close to that obtained in computationally prohibitive MPC-NO approach with disturbance compensation, in which a non-linear optimisation problem has to be solved on-line at each sampling instant. Table 1 compares Integral Squared Error (ISE) in studied MPC algorithms.

Finally, RBF neural network models (44) and (45) are obtained and three considered MPC algorithms simulated. Because both MLP and RBF neural structures are universal approximators, simulation results are practically the same.

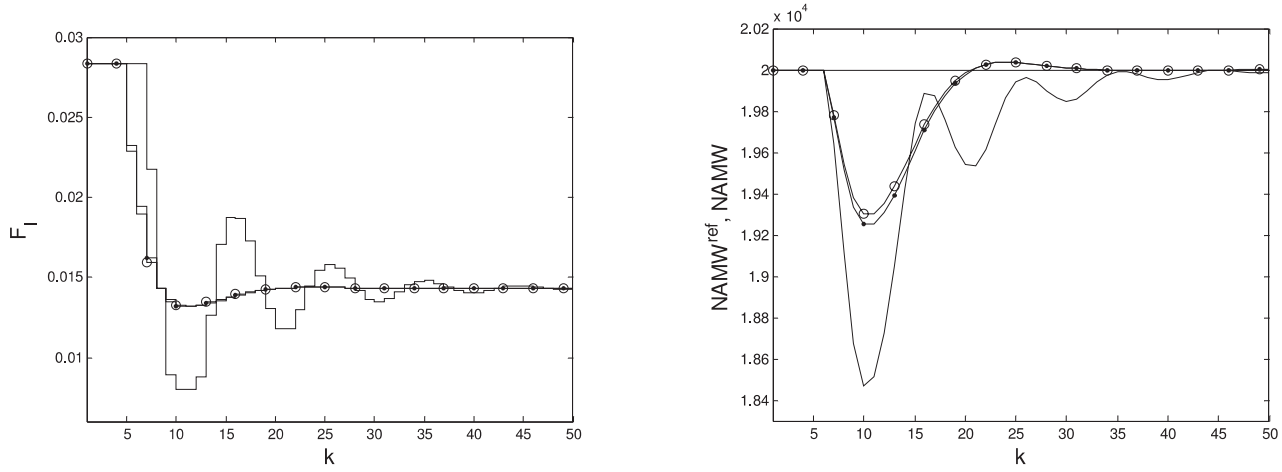


*Fig. 5. Experiment 1: Simulation results of the MPC-NPLa (solid), MPC-NPLb (solid-points) and MPC-NO algorithm (solid-circles).*

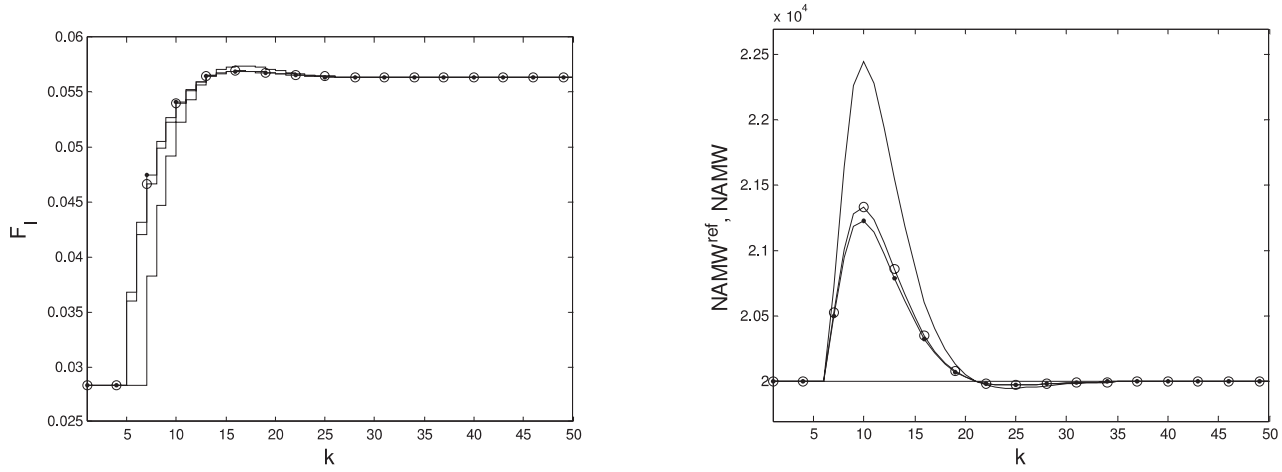


*Fig. 6. Experiment 2: Simulation results of the MPC-NPLa (solid), MPC-NPLb (solid-points) and MPC-NO algorithm (solid-circles).*

*Table 1. Comparison of Integral Squared Error (ISE) in different MPC algorithms.*

| | Experiment 1 | Experiment 2 | Experiment 3 | Experiment 4 |
|---|---|---|---|---|
| MPC-NPLa | $1.11 \cdot 10^7$ | $2.85 \cdot 10^7$ | $1.51 \cdot 10^8$ | $3.00 \cdot 10^7$ |
| MPC-NPLb | $3.25 \cdot 10^6$ | $7.75 \cdot 10^6$ | $4.19 \cdot 10^7$ | $9.21 \cdot 10^6$ |
| MPC-NO | $2.81 \cdot 10^6$ | $9.02 \cdot 10^6$ | $3.61 \cdot 10^7$ | $9.17 \cdot 10^6$ |

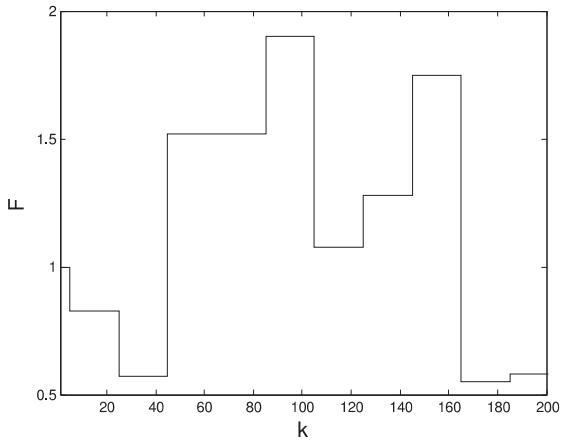


*Fig. 7. Experiment 3: Disturbance F.*

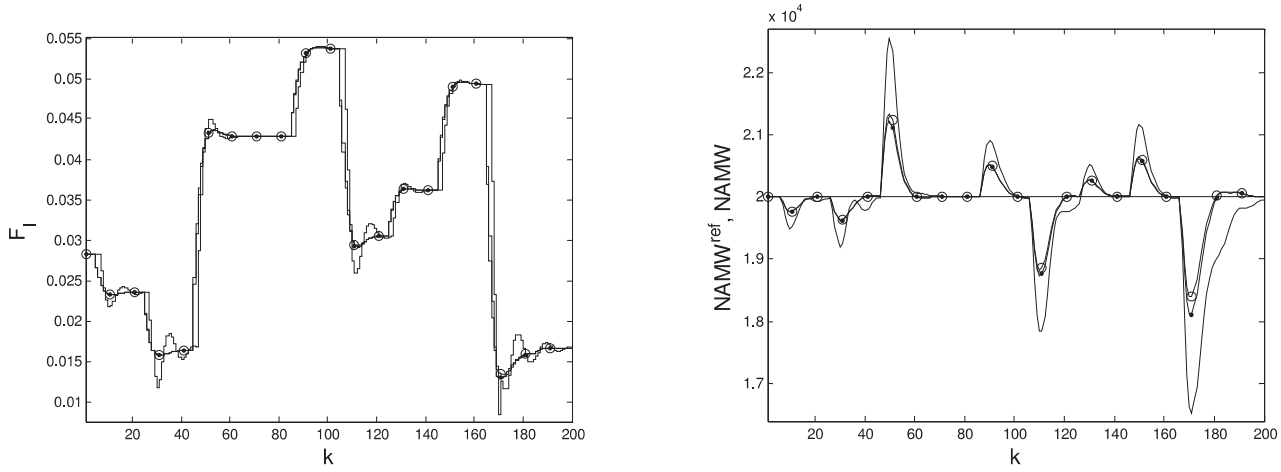


*Fig. 8. Experiment 3: Simulation results of the MPC-NPLa (solid), MPC-NPLb (solid-points) and MPC-NO algorithm (solid-circles).*

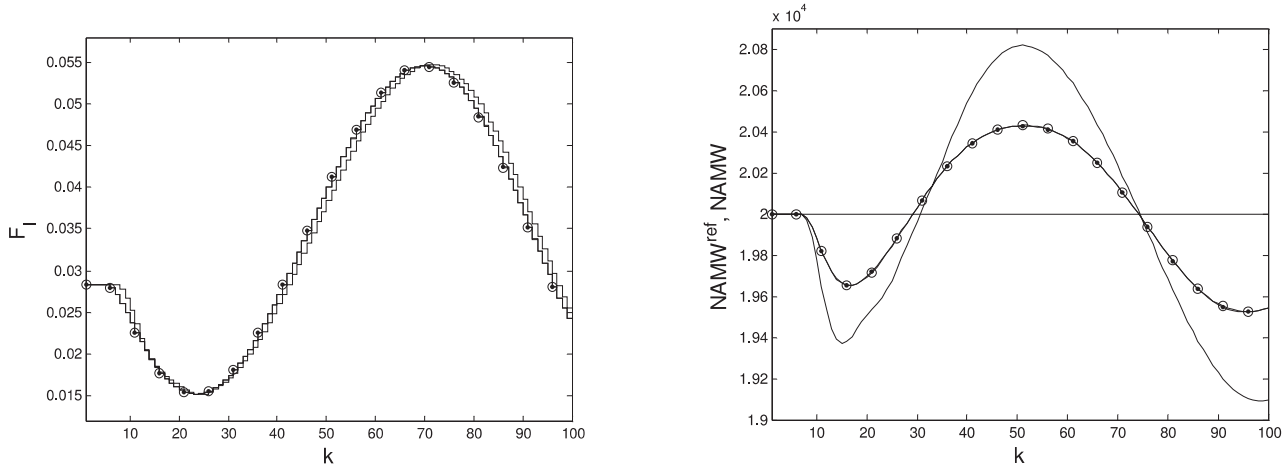


*Fig. 9. Experiment 4: Simulation results of the MPC-NPLa (solid), MPC-NPLb (solid-points) and MPC-NO algorithm (solid-circles).*

### 4.4. Computational efficiency

Considering the simulation results presented above, one can see that the sub optimal MPC-NPLb algorithm gives the closed-loop control profile very close (practically identical) to that obtained in the MPC-NO algorithm. It is then interesting to compare computational complexity of these studied non-linear algorithms with measured disturbance compensation. In the MPC-NO algorithm the SQP (Sequential Quadratic Programming) non-linear optimisation routine is used. As the initial point of optimisation, $N_u$–1 control values calculated at the previous sampling instant and not applied to the process is used

$$\boldsymbol{u}^0(k) = \begin{bmatrix} u^0(k \mid k) \\ \vdots \\ u^0(k+N_u-3 \mid k) \\ u^0(k+N_u-2 \mid k) \\ u^0(k+N_u-1 \mid k) \end{bmatrix} = \begin{bmatrix} u(k \mid k-1) \\ \vdots \\ u(k+N_u-3 \mid k-1) \\ u(k+N_u-2 \mid k-1) \\ u(k+N_u-2 \mid k-1) \end{bmatrix} \tag{47}$$

If $N_u=1$, manipulated variable calculated and applied to the plant at the previous sampling instant is used, i.e. $\boldsymbol{u}^0(k)=[u(k-1) \ \ldots \ u(k-1)]^T$. Although the initial point can be chosen in different ways, the method used is natural and the most efficient in terms of computational complexity [20].

Table 2 shows the influence of control and prediction horizons on floating point operation (MFLOPS) in Experiment 3. Six control horizon ($N_u=1, 2, 3, 4, 5, 10$) and two prediction horizon ($N=5,10$) lengths are considered. In general, the control horizon has far bigger impact on computational burden than the prediction horizon has. It is obvious, since $N_u$ is the number of the decision variables of the MPC optimisation problem (19). The MPC-NO algorithm is many times more computationally demanding than the sub optimal MPC-NPLb algorithm.

*Table 2. Influence of control and prediction horizons on floating point operation (MFLOPS) in the MPC-NPLb algorithm and the MPC-NO algorithm*

| Algorithm | $N$ | $N_u=1$ | $N_u=2$ | $N_u=3$ | $N_u=4$ | $N_u=5$ | $N_u=6$ |
|---|---|---|---|---|---|---|---|
| MPC-NPLb | 5 | 0.32 | 0.38 | 0.50 | 0.67 | 0.92 | – |
| MPC-NO | 5 | 1.85 | 3.65 | 7.89 | 11.70 | 17.00 | – |
| MPC-NPLb | 10 | 0.47 | 0.53 | 0.67 | 0.86 | 1.13 | 3.68 |
| MPC-NO | 10 | 2.85 | 4.13 | 7.63 | 12.85 | 19.24 | 70.57 |

## 5. Conclusions

The paper details the computationally efficient MPC algorithm with Non-linear Prediction and Linearisation (MPC-NPL) with measured disturbance compensation and presents its application to a polymerisation reactor. The algorithm can be used when the process is significantly affected by the uncontrolled but measured inputs (i.e. disturbances). In particular, the algorithm can be used at the advanced MPC layer in control systems with economic optimisation. When MPC co-operates with economic optimisation, compensation of measured disturbances in MPC is particularly important. Disturbance compensation significantly improves control quality. The MPC-NPL algorithm can take into account constraints imposed both on process inputs and outputs, very important from economic and technological reasons.

In comparison with the MPC algorithm with Non-linear Optimisation (MPC-NO), good closed-loop performance and computational efficiency are the advantages of the presented approach. Thanks to linearisation and non-linear free trajectory calculation which are determined on-line from the neural model, the performance of the sub optimal MPC-NPL and potentially "optimal" MPC-NO algorithms is very similar. Computational efficiency of the MPC-NPL algorithm is twofold. Computational burden is significantly smaller than that of the MPC-NO scheme. Furthermore, the sub optimal algorithm needs solving on-line only a quadratic programming problem, which can be done within foreseeable time limit. The necessity of full, non-linear optimisation is avoided.

Neural networks of MLP and RBF types are used as process models. Having excellent approximation abilities, in comparison with popular fuzzy models they do not suffer from "the curse of dimensionality", which is troublesome in multivariable cases. Furthermore, unlike fundamental models, neural models have simple, regular structure. Hence, as detailed in the paper, they can be easily incorporated into the described MPC algorithms and efficiently used on-line. Neural models directly describe input-output relations of process variables; numerical problems typical of MPC algorithms with fundamental models are not encountered.

MPC algorithms whose simulation results are presented in this paper use MLP neural models. From a control engineer's perspective, as detailed in the previous paragraph, both structures can be used in MPC, compare [17] and [19]. On the other hand, MLP networks are global approximators whereas RBF networks are local approximators. It is because in the first case all hidden nodes are used to calculate the output for a given input, in the second case only selected hidden nodes are employed, other is inactive. It affects training, i.e. for MLP networks it is impossible to establish a link between available data and parameters of hidden nodes. As a result, such networks are usually initialised randomly, training (a non-linear optimisation task) is difficult, time consuming and

likely to stop in shallow local minima. Conversely, training of RBF models is much more efficient because parameters of basis functions are directly found from available data. Moreover, selection of the optimal structure of a RBF model which leads to desired approximation accuracy can be included in a training procedure, whereas selection of the structure of a MLP model is tedious, it usually needs training many networks [29].

The presented algorithm can be relatively easy extended to deal with Multi-Input Multi-Output (MIMO) processes, similarly as shown in [17]. Moreover, although in practice stability of the algorithm is usually achieved by proper tuning of the weighting factors $\mu_p$ , $\lambda_p$, it can be combined with the stabilising dual-mode approach in which stability is guaranteed [23].

**ACKNOWLEDGMENTS**

The work presented in this paper was supported by the Polish national budget funds for science in 2005-2007 as a research project.

**AUTHOR**

**Maciej Ławryńczuk** - Institute of Control and Computation Engineering, Faculty of Electronics and Information Technology, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warszawa, Poland. Tel. +48 22 234-76-73, fax. +48 22 825-37-19.
E-mail: M.Lawrynczuk@ia.pw.edu.pl.

**References**

[1] B. M. Åkesson, H. T. Toivonen, "A neural network model predictive controller", *Journal of Process Control*, vol. 16, no. 3, 2006, pp. 937-946.

[2] R. Babuška, J. M. Sousa, H. B. Verbruggen, "Predictive control of nonlinear systems based on fuzzy and neural models", *The European Control Conference*, Karlsruhe, Germany, 1999, CD-ROM, paper F1032-5.

[3] T. L. Blevins, G. K. Mcmillan, M. W. Wojsznis, *Advanced control unleashed*. ISA, 2003.

[4] M. Brdyś, P. Tatjewski, *Iterative algorithms for multilayer optimizing control*, Imperial College Press, London, 2005.

[5] L. Cavagnari, L. Magni, R. Scattolini, "Neural network implementation of nonlinear receding-horizon control", *Neural Computing and Applications*, vol. 8, no. 1, 1999, pp. 86-92.

[6] W. Findeisen, F. N. Bailey, M. Brdyś, K. Malinowski, P. Tatjewski, A. Woźniak, *Control and coordination in hierarchical systems*, J. Wiley & Sons, Chichester - New York - Brisbane - Toronto, 1980.

[7] S. Haykin, *Neural networks - a comprehensive foundation*, Prentice Hall, Upper Saddle River, 1999.

[8] M. A. Henson, "Nonlinear model predictive control: current status and future directions", *Computers and Chemical Engineering*, vol. 23, no. 2, 1998, pp. 187-202.

[9] K. Hornik, M. Stinchcombe, H. White, "Multilayer feedforward networks are universal approximators", *Neural networks*, vol. 2, no. 5, 1989, pp. 359-366.

[10] M. A. Hussain, "Review of the applications of neural networks in chemical process control simulation and online implementation", *Artificial Intelligence in Engineering*, vol. 13, no. 1, 1999, pp. 55-68.

[11] D. E. Kassmann, T. A. Badgwell, R. B. Hawkins, "Robust Steady-State Target Calculation for Model Predictive Control", *AIChE Journal*, vol. 46, 2000, no. 5, pp. 1007-1024.

[12] G. P. Liu, V. Kadirkamanathan, S. A. Billings, "Predictive control of nonlinear systems using neural networks", *International Journal of Control*, vol. 71, 1998, no. 6, pp. 1119-1132.

[13] M. Ławryńczuk, P. Marusak, P. Tatjewski, "Structures and algorithms of co-operation of predictive control and on-line economic optimisation", *Pomiary, Automatyka, Kontrola*, vol. 53, no. 10, 2007, pp. 55-61. (In Polish)

[14] M. Ławryńczuk P. Marusak, P. Tatjewski, "Multilayer and integrated structures for predictive control and economic optimisation". In: *The 11th IFAC/IFORS/IMACS/IFIP Symposium on Large Scale Systems: Theory and Applications*, LSS 2007, Gdańsk, Poland, 2007, CD-ROM, paper 60.

[15] M. Ławryńczuk, P. Marusak, P. Tatjewski, "An efficient MPC algorithm integrated with economic optimisation for MIMO systems". In: *The 13th IEEE/IFAC International Conference on Methods and Models in Automation and Robotics*, MMAR 2007, Szczecin, Poland, 2007, pp. 295-302.

[16] M. Ławryńczuk, P. Marusak, P. Tatjewski, "Set-point optimisation and predictive constrained control for fast feedback controlled processes". In: *The 13th IEEE/IFAC International Conference on Methods and Models in Automation and Robotics*, MMAR 2007, Szczecin, Poland, 2007, pp. 357-362.

[17] M. Ławryńczuk, "A family of model predictive control algorithms with artificial neural networks", *International Journal of Applied Mathematics and Computer Science*, vol. 17, no 2, 2007, pp. 217-232.

[18] M. Ławryńczuk, "A suboptimal nonlinear predictive control algorithm based on neural models with measured disturbance compensation". In: *The 13th IEEE/IFAC International Conference on Methods and Models in Automation and Robotics*, MMAR 2007, Szczecin, Poland, 2007, pp. 509-516.

[19] M. Ławryńczuk, P. Tatjewski, "A computationally efficient nonlinear predictive control algorithm with RBF neural models and its application". In: *Lecture Notes in Artificial Intelligence, vol. 4585, Springer: The International Conference Rough Sets and Emerging Intelligent Systems Paradigms, RSEISP 2007*, Warszawa, Poland, 2007, pp. 603-612.

[20] M. Ławryńczuk:, "Computational efficiency of suboptimal nonlinear predictive control with neural models". In: *The International Conference on Advances in Artificial Intelligence and Applications*, AAIA 2007, Wisła 2007, Poland, pp. 257-266.

[21] M. Ławryńczuk, P. Marusak, P. Tatjewski, "Integrating predictive control with steady-state optimisation", The *12th IEEE International Conference on Methods and Models in Automation and Robotics*, MMAR 2006, Międzyzdroje, Poland, 2006, pp. 445-452.

[22] M. Ławryńczuk, P. Tatjewski, "An efficient nonlinear predictive control algorithm with neural models and its application to a high-purity distillation process". In: *Lecture notes in Artificial Intelligence, vol. 4029, Springer: The Eighth International Conference on Artificial Intelligence and Soft Computing ICAISC 2006*, Zakopane, Poland, 2006, pp. 76-85.

[23] M. Ławryńczuk, P. Tatjewski, "A stable dual-mode type nonlinear predictive control algorithm based on on-line linearisation and quadratic programming". In: *The 10th IEEE International Conference on Methods and Models in Automation and Robotics*, MMAR 2004, Międzyzdroje, Poland, 2004, pp. 503-510.

[24] J. M. Maciejowski, *Predictive control with constraints*, Prentice Hall, Harlow, 2002.

[25] M. Mahfouf, D. A. Linkens, "Non-linear generalized predictive control (NLGPC) applied to muscle relaxant anaesthesia". *International Journal of Control*, vol. 71, no. 2, 1998, pp. 239-257.

[26] B. R. Maner, F. J. Doyle, B. A. Ogunnaike, R. K. Pearson, "Nonlinear model predictive control of a simulated multivariable polymerization reactor using second-order Volterra models", *Automatica*, vol. 32, no. 9, 1996, pp. 1285-1301.

[27] M. Morari, J. Lee, "Model predictive control: past, present and future". *Computers and Chemical Engineering*, vol. 23, no. 4/5, 1999, pp. 667-682.

[28] M. Nørgaard, O. Ravn, N. K. Poulsen, L. K. Hansen, *Neural networks for modelling and control of dynamic systems*, Springer, London, 2000.

[29] S. Osowski, *Neural networks - an algorithmic approach* (in Polish), Publ. House: WNT, Warszawa 1996.

[30] S. Piche, B. Sayyar-Rodsari, D. Johnson, M. Gerules, "Nonlinear model predictive control using neural networks", *IEEE Control Systems Magazine*, vol. 20, no. 3, 2000, pp. 56-62.

[31] T. Parisini, M. Sanguineti, R. Zoppoli, "Nonlinear stabilization by receding-horizon neural regulators", *International Journal of Control*, vol. 70, no. 3, 1998, pp. 341-362.

[32] M. Pottmann, D. E. Seborg, "A nonlinear predictive control strategy based on radial basis function networks", *Computers and Chemical Engineering*, vol. 21, no. 9, 1997, pp. 965-980.

[33] S. J. Qin, T. Badgwell, "A survey of industrial model predictive control technology", *Control Engineering Practice*, vol. 11, no. 7, 2003, pp. 733-764.

[34] J. A. Rossiter, *Model-based predictive control*, CRC Press, Boca Raton, 2003.

[35] D. Saez, A. Cipriano, A. W. Ordys, *Optimisation of industrial processes at supervisory level: application to control of thermal power plants*, Springer, London, 2002.

[36] P. Tatjewski, *Advanced control of industrial processes, structures and algorithms*, Springer, London, 2007.

[37] P. Tatjewski, M. Ławryńczuk, P. Marusak, "Linking nonlinear steady-state and target set-point optimisation for model predictive control". *The IEE Control Conference*, Glasgow, United Kingdom, 2006, CD-ROM.

[38] P. Tatjewski, M. Ławryńczuk, "Soft computing in model-based predictive control", *International Journal of Applied Mathematics and Computer Science*, vol. 16, no. 1, 2006, pp. 101-120.

[39] Z. Trajanoski, P. Wach, "Neural predictive control for insulin delivery using the subcutaneous route", *IEEE Transactions on Biomedical Engineering*, vol. 45, no. 9, 1998, pp. 1122-1134.

[40] M. Tvrzska de Gouvea, D. Odloak, "One-layer real time optimization of LPG production in the FCC unit: procedure, advantages and disadvantages", *Computers & Chemical Engineering*, vol. 22, Supplement, 1998, pp. S191-S198.

[41] D. L. Yu, J. B. Gomm, "Implementation of neural network predictive control to a multivariable chemical reactor", *Control Engineering Practice*, vol. 11, no. 11, 2003, pp. 1315-1323.

[42] A. Zanin, M. Tvrzska de Gouvea, D. Odloak, "Integrating real-time optimization into model predictive controller of the FCC system", *Control Engineering Practice*, vol. 10, no. 8, 2002, pp. 819-831.