AN ITERATIVE NONLINEAR PREDICTIVE CONTROL ALGORITHM BASED ON LINEARISATION AND NEURAL MODELS

Maciej Lawrynczuk, Piotr Tatjewski

Warsaw University of Technology,
Institute of Control and Computation Engineering
ul. Nowowiejska 15/19, 00-665 Warszawa, Poland
tel. +48 22 660-73-97, fax. +48 22 825-37-19
lawrynczuk@ia.pw.edu.pl, tatjewski@ia.pw.edu.pl

Keywords: Nonlinear model predictive control, neural-network models, linearisation, quadratic programming.

Abstract

This paper is concerned with a computationally efficient suboptimal nonlinear predictive control algorithm. The nonlinear model of the plant is used to obtain a local linearisation and to calculate, by means of an iterative procedure, the nonlinear response and future control moves. In comparison with fully-fledged nonlinear algorithms, which hinge on non-convex optimisation, the presented approach is more reliable and less computationally demanding because it results in a series of convex, constrained or unconstrained, quadratic programming problems whereas its closed-loop performance is similar. The algorithm implementation for feedforward neural-network models is also discussed in the paper.

1 Introduction

Model predictive control (MPC) is recognised as the only advanced control technique which has made a substantial impact on industrial applications, largely due to its unique ability to handle hard constraints [6,9,13]. Especially, the algorithms based on linear models, mainly DMC and GPC, are usually applied to on-line control, because they result in quadratic programming problems to which there exist reliable (convexity implies foreseeable computational time) and efficient software packages. However, in the case of severe plant nonlinearity such an approach is likely to lead to poor performance. As far as nonlinear MPC algorithms are concerned the real-time implementation is an issue of crucial importance. A nonlinear model used for both prediction and optimisation purposes leads to a non-quadratic and, in general, non-convex optimisation problem which should be solved at each time instant on-line. Because gradient-based techniques may terminate in local minima different global optimisation techniques have been used, for example the modified simplex method of Nelder-Mead [7], genetic algorithms [1,7], or even branch-and-bound methods [1]. Although such approaches inevitably increase the computational burden they still give no guarantee that the global solution is found.

To circumvent the difficulties typical of nonlinear MPC a few alternatives have been suggested. For example, the computational burden can be significantly reduced when only the first control move is optimised, the remaining ones are obtained using a linear MPC [14], yet the problem is still non-convex. In the case of some models an appropriate structure exploitation [2] or change of coordinates [12] leads to convexity. First and foremost, as it is emphasised in [11], the linearisation-based MPC algorithms [1,3,4,5,10,13] are the ones which have found wide use in industry.

In this paper a control algorithm with Iteratively updated Nonlinear Prediction and Linearisation (MPC-INPL) is presented. The nonlinear model of the plant is used to obtain a local linearisation and to calculate the nonlinear free response, as it is done in the algorithm originally suggested in [4]. Moreover, to improve the overall performance, the nonlinear response and future control moves are calculated in an iterative way, therefore a quadratic optimisation is repeated a few times. Furthermore, the paper presents the algorithm implementation for feedforward neural networks because such models have the following advantages: (*i*) are able to approximate strong nonlinearities, (*ii*) have simple, regular structure, do not suffer from "the curse of dimensionality" phenomenon, which is troublesome in multivariable cases, (*iii*) can be easily incorporated into the algorithm and effectively used on-line.

The paper is organised as follows. In Section 2 the general idea of the MPC-INPL algorithm is presented. Next, in Section 3, the neural-network implementation is detailed. In Section 4 simulation results of a highly nonlinear polymerisation reactor are discussed, especially, the MPC-INPL algorithm is compared to linear MPC and MPC-NO with Nonlinear Optimisation. Finally, the paper is summarised in Section 5.

2 The MPC-INPL algorithm

Let the process under consideration be described by the following nonlinear discrete-time equation

$$y(k) = g(u(k-\tau),\ldots,u(k-n_b),y(k-1),\ldots,y(k-n_a)) \quad (1)$$

where $g:\Re^{n_a+n_b-\tau+1}\to\Re\in C^1$, $\tau\le n_b$. The linear approximation of the model (1), obtained at time instant k, is

$$\boldsymbol{A}(k,z^{-1})y(k) = \boldsymbol{B}(k,z^{-1})u(k) \quad (2)$$

where

$$\begin{aligned}\boldsymbol{A}(k,z^{-1}) &= (1+a_1(k)z^{-1}+\ldots+a_{n_a}(k)z^{-n_a})\\ \boldsymbol{B}(k,z^{-1}) &= (b_1(k)z^{-1}+\ldots+b_{n_b}(k)z^{-n_b})\end{aligned} \quad (3)$$

Defining the vector $\boldsymbol{x}(k)$, which determines the linearisation point

$$\boldsymbol{x}(k) = [u(k-\tau)\ldots u(k-n_b)\ y(k-1)\ldots y(k-n_a)]^T \quad (4)$$

the coefficients of the linear model (2) are calculated from

$$\begin{aligned}a_l(\boldsymbol{x}(k)) &= -\frac{\partial g(\boldsymbol{x}(k))}{\partial y(k-l)} \quad \text{for} \quad l=1,\ldots,n_a\\ b_l(\boldsymbol{x}(k)) &= \frac{\partial g(\boldsymbol{x}(k))}{\partial u(k-l)} \quad \text{for} \quad l=\tau,\ldots,n_b\end{aligned} \quad (5)$$

Remark 1: The coefficient $a_l(k)$, $b_l(k)$ are not influenced by the most recent output value $y(k)$, which is available. It may be crucial in the case of fast processes. Therefore it is recommended to use

$$\boldsymbol{x}(k) = [u(k-\tau+1)\ldots u(k-n_b+1)\ y(k)\ldots y(k-n_a+1)]^T \quad (6)$$

If τ=1 for linearisation purposes one may set $u(k)$=$u(k-1)$ or $u(k)$=$u(k|k-1)$.

The performance index, the minimisation of which at each time instant k yields the optimal input profile, is defined

$$\begin{aligned}J(k) &= \sum_{p=H_1}^{H}\mu_p\left(y^{ref}(k+p|k)-\hat{y}(k+p|k)\right)^2 +\\ &+ \sum_{p=0}^{H_u}\lambda_p\left(\Delta u(k+p|k)\right)^2\end{aligned} \quad (7)$$

where $y^{ref}(k+p|k)$ – known or presumed output reference trajectory, $\hat{y}(k+p|k)$ – predicted outputs, $\Delta u(k+p|k)$ – future control moves (decision variables), H, H_u – prediction and control horizon, H_1=τ, $\mu_p\ge 0$, $\lambda_p>0$. The optimisation of the cost-function (7) is carried out subject the following hard constraints, although in practice it is advisable to use soft output ones [6,13]

$$\begin{aligned}u_{\min} &\le u(k+p|k) \le u_{\max} \quad \text{for} \quad p=0,\ldots,H_u-1\\ -\Delta u_{\max} &\le \Delta u(k+p|k) \le \Delta u_{\max} \quad \text{for} \quad p=0,\ldots,H_u-1\\ y_{\min} &\le \hat{y}(k+p|k) \le y_{\max} \quad \text{for} \quad p=H_1,\ldots,H\end{aligned} \quad (8)$$

The following vectors are defined

$$\begin{aligned}\boldsymbol{y}^{ref}(k) &= [y^{ref}(k+H_1|k)\ldots y^{ref}(k+H|k)]^T\\ \hat{\boldsymbol{y}}(k) &= [\hat{y}(k+H_1|k)\ldots \hat{y}(k+H|k)]^T\\ \boldsymbol{y}^0(k) &= [y^0(k+H_1|k)\ldots y^0(k+H|k)]^T\\ \Delta\boldsymbol{u}(k) &= [\Delta u(k_1|k)\ldots \Delta u(k+H_u-1|k)]^T\end{aligned} \quad (9)$$

It is also assumed that the superposition principle holds true, as it is done in linear MPC, i.e.

$$\hat{\boldsymbol{y}}(k) = \boldsymbol{y}^0(k) + \boldsymbol{G}(k)\Delta\boldsymbol{u}(k) \quad (10)$$

where the vector $\boldsymbol{y}^0(k)$ is the free response and the matrix $\boldsymbol{G}(k)$ contains step-response coefficients of the linearised model (2)

$$\boldsymbol{G}(k) = \begin{bmatrix} g_{H_1}(k) & 0 & \ldots & 0\\ g_{H_1+1}(k) & g_{H_1}(k) & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ g_H(k) & g_{H-1}(k) & \ldots & g_{H-H_u+1}(k)\end{bmatrix} \quad (11)$$

which are obtained from $a_l(k)$ and $b_l(k)$ [13].

Although in the nonlinear case the suboptimal prediction $\hat{\boldsymbol{y}}(k)$ calculated from (10) is different from that obtained when a nonlinear model (1) is used, it leads to convexity of the performance index (7)

$$J(k) = \left\|\boldsymbol{y}^{ref}(k)-\boldsymbol{y}^0(k)-\boldsymbol{G}(k)\Delta\boldsymbol{u}(k)\right\|^2_{\boldsymbol{M}} + \left\|\Delta\boldsymbol{u}(k)\right\|^2_{\boldsymbol{\Lambda}} \quad (12)$$

where the diagonal matrices $\boldsymbol{M}$, $\boldsymbol{\Lambda}$ are comprised of coefficients μ_p, λ_p, respectively.

In the simplest kind of a linearisation-based MPC algorithm, i.e. successive linearisation approach, the coefficients $a_l(k)$, $b_l(k)$ of the linear model (2), which depend on the current state of the plant (4) or (6), are obtained from the formulae (5) and then used to calculate both free response $\boldsymbol{y}^0(k)$ and the matrix $\boldsymbol{G}(k)$, as it is done in linear MPC [6,13]. Much more effective are the algorithms which use the nonlinear model (1) to calculate the nonlinear free response [4,13]. Provided that the model is reliable enough the accuracy of the prediction $\hat{\boldsymbol{y}}(k)$ obtained from (10) is increased and thus the overall performance can be improved.

The idea behind the MPC-INPL algorithm is to replace the nonlinear free response $\boldsymbol{y}^0(k)$ in (10) by an iteratively calculated nonlinear response $\boldsymbol{y}^n(k)$ which corresponds to future input trajectory

$$\boldsymbol{u}^n(k) = \begin{bmatrix} u^n(k|k)\\ u^n(k+1|k)\\ \vdots\\ u^n(k+H_u-1|k)\end{bmatrix} \quad (13)$$

where $u^n(k+p\,|\,k)=u^n(k+H_u-1\,|\,k)$ for $p\geq H_u$, n indicates the internal iteration. The initial input trajectory assumes no changes in the manipulated variable from time instant k, i.e

$$\boldsymbol{u}^0(k)=\begin{bmatrix} u^0(k\,|\,k) \\ u^0(k+1\,|\,k) \\ \vdots \\ u^0(k+H_u-1\,|\,k) \end{bmatrix}=\begin{bmatrix} u(k-1) \\ u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix} \tag{14}$$

The prediction equation used in the INPL algorithm, assuming the superposition principle (10), becomes

$$\hat{\boldsymbol{y}}(k)=\boldsymbol{y}^n(k)+\boldsymbol{G}(k)\Delta\boldsymbol{u}^{n+1}(k) \tag{15}$$

where the vector $\Delta\boldsymbol{u}^{n+1}(k)$ is calculated iteratively (in the internal loop). It represents the increments from the trajectory $\boldsymbol{u}^n(k)$. The performance index (12) has to take into account the total increments from the initial trajectory $\boldsymbol{u}^0(k)$, hence

$$\begin{aligned} J(k) &= \left\|\boldsymbol{y}^{ref}(k)-\boldsymbol{G}(k)\Delta\boldsymbol{u}^{n+1}(k)-\boldsymbol{y}^n(k)\right\|^2_{\boldsymbol{M}}+ \\ &+ \left\|\Delta\boldsymbol{u}^n(k)+\Delta\boldsymbol{u}^{n+1}(k)\right\|^2_{\boldsymbol{\Lambda}} \end{aligned} \tag{16}$$

The control algorithm can be summarised as follows

1. Linearisation: obtain coefficients $a_l(k)$, $b_l(k)$ and the matrix $\boldsymbol{G}(k)$.
2. Initialise the internal loop: calculate the nonlinear free response $\boldsymbol{y}^0(k)$ using the initial input trajectory $\boldsymbol{u}^0(k)$, set $n=0$, $\Delta\boldsymbol{u}^0(k)=0$.
3. Solve the quadratic programming problem

$$\begin{gathered} \min_{\Delta\boldsymbol{u}^{n+1}(k)}\left\{J(k)\right\} \\ \boldsymbol{u}_{\min}\leq\boldsymbol{u}^n(k)+\boldsymbol{J}\Delta\boldsymbol{u}^{n+1}(k)\leq\boldsymbol{u}_{\max} \\ -\Delta\boldsymbol{u}_{\max}\leq\Delta\boldsymbol{u}^n(k)+\Delta\boldsymbol{u}^{n+1}(k)\leq\Delta\boldsymbol{u}_{\max} \\ \boldsymbol{y}_{\min}\leq\boldsymbol{G}(k)\Delta\boldsymbol{u}^{n+1}(k)+\boldsymbol{y}^n(k)\leq\boldsymbol{y}_{\max} \end{gathered} \tag{17}$$

which yields the control moves $\Delta\boldsymbol{u}^{n+1}(k)$, i.e. $\Delta u^{n+1}(k+p|k)$ for $p=0,...,H_u-1$. The square matrix $\boldsymbol{J}$, of dimension H_u, is

$$\boldsymbol{J}=\begin{bmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix} \tag{18}$$

whereas the constant vectors $\boldsymbol{u}_{\min}$, $\boldsymbol{u}_{\max}$, $\Delta\boldsymbol{u}_{\max}$, $\boldsymbol{y}_{\min}$, $\boldsymbol{y}_{\max}$ result from the constraints (8).

4. If

$$\left\|\Delta u^{n+1}(k\,|\,k)-\Delta u^n(k\,|\,k)\right\|<\varepsilon\left\|\Delta u^n(k\,|\,k)\right\| \tag{19}$$

where $\varepsilon>0$, or $n=n_{\max}$, terminate the internal loop, apply

$$u(k)=u(k\,|\,k)=u^n(k\,|\,k)+\Delta u^{n+1}(k\,|\,k) \tag{20}$$

set $k=k+1$, go to step 1.

5. Modification of the input trajectory $u^{n+1}(k)$ taking into account the first control move $\Delta u^{n+1}(k|k)$ calculated in step 3

$$\boldsymbol{u}^{n+1}(k)=\begin{bmatrix} u^{n+1}(k\,|\,k) \\ u^{n+1}(k+1\,|\,k) \\ \vdots \\ u^{n+1}(k+H_u-1\,|\,k) \end{bmatrix}=\begin{bmatrix} u^n(k\,|\,k) \\ u^n(k\,|\,k) \\ \vdots \\ u^n(k\,|\,k) \end{bmatrix}+\alpha\begin{bmatrix} \Delta u^{n+1}(k\,|\,k) \\ \Delta u^{n+1}(k\,|\,k) \\ \vdots \\ \Delta u^{n+1}(k\,|\,k) \end{bmatrix} \tag{21}$$

where $u^{n+1}(k+p\,|\,k)=u^{n+1}(k+H_u-1\,|\,k)$ for $p\geq H_u$, $\alpha>0$. Set

$$\Delta\boldsymbol{u}^{n+1}(k)=\begin{bmatrix} u^{n+1}(k\,|\,k)-u(k-1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{22}$$

6. Calculate the nonlinear trajectory $\boldsymbol{y}^{n+1}(k)$, to be used in the next internal iteration, which results from the input trajectory $\boldsymbol{u}^{n+1}(k)$. Set $n=n+1$, go to step 3 (continuation of the internal loop).

Remark 2: The linearisation (step 1) is usually performed at each time instant k. However, it can be repeated less frequently, especially when the process is close to its steady-state.

Remark 3: In the presented algorithm the single-step linearisation technique is used, i.e. one set of coefficients $a_l(k)$, $b_l(k)$, is used to calculate the matrix $\boldsymbol{G}(k)$ through the entire prediction horizon. It is also possible to apply a multi-step linearisation, in which different linear approximations of the nonlinear model will be used for each instant of the prediction horizon.

Remark 4: If the MPC-INPL algorithm is implemented in its unconstrained version, the input moves $\Delta\boldsymbol{u}^{n+1}(k)$ in step 3 are calculated analytically from

$$\begin{aligned} \Delta\boldsymbol{u}^{n+1}(k) &= \left[\boldsymbol{G}^T(k)\boldsymbol{M}\boldsymbol{G}(k)+\boldsymbol{\Lambda}\right]^{-1}\times \\ &\times \left[\boldsymbol{G}^T(k)\boldsymbol{M}\left(\boldsymbol{y}^{ref}(k)-\boldsymbol{y}^n(k)\right)-\boldsymbol{\Lambda}\Delta\boldsymbol{u}^n(k)\right] \end{aligned} \tag{23}$$

Alternatively, more computationally reliable least-squares method can be used

$$\Delta\boldsymbol{u}^{n+1}(k)=\begin{bmatrix} \boldsymbol{S}_{\boldsymbol{M}}\boldsymbol{G}(k) \\ -\boldsymbol{S}_{\boldsymbol{\Lambda}} \end{bmatrix}^+\begin{bmatrix} \boldsymbol{S}_{\boldsymbol{M}}\left(\boldsymbol{y}^{ref}(k)-\boldsymbol{y}^n(k)\right) \\ \boldsymbol{S}_{\boldsymbol{\Lambda}}\Delta\boldsymbol{u}^n(k) \end{bmatrix} \tag{24}$$

where $\boldsymbol{S}_{\boldsymbol{M}}^T\boldsymbol{S}_{\boldsymbol{M}}=\boldsymbol{M}$, $\boldsymbol{S}_{\boldsymbol{\Lambda}}^T\boldsymbol{S}_{\boldsymbol{\Lambda}}=\boldsymbol{\Lambda}$ and "+" denotes Moore-Penrose pseudo-inverse.

Remark 5: In the simplest case the coefficient α (step 5) is constant, e.g. $\alpha=1$. To prevent the algorithm from the lack of convergence the following approach is recommended: having obtained the solution to the optimisation problem (17) in step 3 corresponding value of the performance index $J^{n+1}(k)$ is evaluated, using the nonlinear model. The internal loop is continued provided that the additional condition $J^{n+1}(k)<J^n(k)$

is fulfilled. Yet another technique is to calculate the value of α by means of a line-search procedure.

Remark 6: The nonlinear trajectory $\boldsymbol{y}^{n+1}(k)$ is calculated (step 6) using input trajectory $\boldsymbol{u}^{n+1}(k)$ (obtained in step 5) which is updated taking into account only the first input move $\Delta u^{n+1}(k|k)$. Instead, in step 6 the whole sequence $\Delta u^{n+1}(k+p|k)$, p=0,...,H_u–1 can be used, it may improve the overall performance. In such a case

$$\boldsymbol{u}^{n+1}(k) = \boldsymbol{u}^{n}(k) + \alpha \begin{bmatrix} \Delta u^{n+1}(k \mid k) \\ \Delta u^{n+1}(k \mid k) + \Delta u^{n+1}(k+1 \mid k) \\ \vdots \\ \Delta u^{n+1}(k \mid k) + \ldots + \Delta u^{n+1}(k+H_u-1 \mid k) \end{bmatrix} \quad (25)$$

and (22) becomes

$$\Delta \boldsymbol{u}^{n+1}(k) = \begin{bmatrix} u^{n+1}(k \mid k) - u(k-1) \\ u^{n+1}(k+1 \mid k) - u^{n+1}(k \mid k) \\ \vdots \\ u^{n+1}(k+H_u-1 \mid k) - u^{n+1}(k+H_u-2 \mid k) \end{bmatrix} \quad (26)$$

Remark 7: The initial input trajectory (14) may be chosen differently. It is advisable to use the control sequence found at previous time instance, i.e.

$$\boldsymbol{u}^{0}(k) = \begin{bmatrix} u^{0}(k \mid k) \\ \vdots \\ u^{0}(k+H_u-2 \mid k) \\ u^{0}(k+H_u-1 \mid k) \end{bmatrix} = \begin{bmatrix} u(k \mid k-1) \\ \vdots \\ u(k+H_u-2 \mid k-1) \\ u(k+H_u-2 \mid k-1) \end{bmatrix} \quad (27)$$

3 Neural MPC-INPL algorithm

The MPC-INPL algorithm presented in previous section can be used with different structures of the model: first-principles (fundamental) or empirical. In the sequel it is assumed that the feedforward neural network with one hidden layer containing K nonlinear nodes and linear output is used as the function g in (1). Output of the model is given by

$$y(k) = w_0^2 + \sum_{i=1}^{K} w_i^2 v_i(k) = w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(z_i(k)) \quad (28)$$

where $z_i(k)$ and $v_i(k)$ denote the sum of inputs and the output of the i-th hidden node, respectively, φ is the nonlinear transfer function, w_i^2 are the weights of the second (output) layer. Recalling the input arguments of the general nonlinear model (1) one has

$$z_i(k) = w_{i,0}^1 + \sum_{j=1}^{N_u} w_{i,j}^1 u(k-\tau+1-j) + \sum_{j=1}^{n_a} w_{i,N_u+j}^1 y(k-j) \quad (29)$$

where $w_{i,j}^1$ are the weights of the first (hidden) layer and N_u=n_b–τ+1.

The coefficients of the linearised model (2) are obtained on-line from equations (5), which lead to

$$\begin{aligned} a_l(\boldsymbol{x}(k)) &= -\sum_{i=1}^{K} w_i^2 \frac{d\varphi(z_i(\boldsymbol{x}(k)))}{dz_i(\boldsymbol{x}(k))} w_{i,N_u+l}^1 \quad \text{for} \quad l = 1,\ldots,n_a \\ b_l(\boldsymbol{x}(k)) &= \sum_{i=1}^{K} w_i^2 \frac{d\varphi(z_i(\boldsymbol{x}(k)))}{dz_i(\boldsymbol{x}(k))} w_{i,l-\tau+1}^1 \quad \text{for} \quad l = \tau,\ldots,n_b \end{aligned} \quad (30)$$

From (28) the output prediction for time instant $k+p|k$ is

$$\hat{y}(k+p \mid k) = w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(z_i(k+p \mid k)) + d(k) \quad (31)$$

where (32)

$$\begin{aligned} z_i(k+p \mid k) &= w_{i,0}^1 + \\ &+ \sum_{j=1}^{N_{un}(p)} w_{i,j}^1 u(k-\tau+1-j+p \mid k) + \sum_{j=N_{un}(p)+1}^{N_u} w_{i,j}^1 u(k-\tau+1-j+p) + \\ &+ \sum_{j=1}^{N_{\hat{y}}(p)} w_{i,N_u+j}^1 \hat{y}(k-j+p \mid k) + \sum_{j=N_{\hat{y}}(p)+1}^{n_a} w_{i,N_u+j}^1 y(k-j+p) \end{aligned}$$

and

$$\begin{aligned} N_{un}(p) &= \max(\min(p-\tau+1, N_u), 0) \\ N_{\hat{y}}(p) &= \min(p-1, n_a) \end{aligned} \quad (33)$$

In (31) the "DMC type" disturbance model is used, in which the unmeasured disturbance $d(k)$ is assumed to be constant over the prediction horizon. Its value is estimated from the equation

$$d(k) = y(k) - y(k \mid k-1) \quad (34)$$

where $y(k)$ is the actual output of the plant, $y(k \mid k-1)$ is calculated from the nonlinear model (1) using the measured input and output values up to time instant k–1.

The nonlinear response is calculated recursively from the prediction equation (31) and (32) using the trajectories $\boldsymbol{u}^{n+1}(k)$ and $\boldsymbol{y}^{n+1}(k)$

$$y^{n+1}(k+p \mid k) = w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(z_i^{n+1}(k+p \mid k)) + d(k) \quad (35)$$

where (36)

$$\begin{aligned} z_i^{n+1}(k+p \mid k) &= w_{i,0}^1 + \\ &+ \sum_{j=1}^{N_{un}(p)} w_{i,j}^1 u^{n+1}(k-\tau+1-j+p \mid k) + \sum_{j=N_{un}(p)+1}^{N_u} w_{i,j}^1 u(k-\tau+1-j+p) + \\ &+ \sum_{j=1}^{N_{\hat{y}}(p)} w_{i,N_u+j}^1 y^{n+1}(k-j+p \mid k) + \sum_{j=N_{\hat{y}}(p)+1}^{n_a} w_{i,N_u+j}^1 y(k-j+p) \end{aligned}$$

4 Simulation results

In this section the MPC-INPL algorithm is applied to a polymerisation reaction taking place in a jacketed continuous stirred tank reactor [8]. The reaction under consideration is

the free-radical polymerisation of methyl methacrylate with azo-bis-isobutyronitrile as initiator and toluene as solvent. The output m, which is the number average molecular weight (kg/kmol), is controlled by manipulating the inlet initiator flow rate F (m^3/h).

Introducing the system parameters into the first-principles equations the following model is obtained

$$\begin{aligned}\dot{x}_1 &= 10(6-x_1) - 2.4568x_1\sqrt{x_2}\\ \dot{x}_2 &= 80F - 10.1022x_2\\ \dot{x}_3 &= 0.0024121x_1\sqrt{x_2} + 0.112191x_2 - 10x_3\\ \dot{x}_4 &= 245.978x_1\sqrt{x_2} - 10x_4\end{aligned} \qquad m = \frac{x_4}{x_3} \quad (37)$$

The nominal operating conditions are: x_{10}=5.506774, x_{20}=0.132906, x_{30}=0.0019752, x_{40}=49.38182, F_0=0.016783, m_0=25000.5. The following bound constraints are put on manipulated variable: $m_{\min}$=0,0035, $m_{\max}$=0,033566.

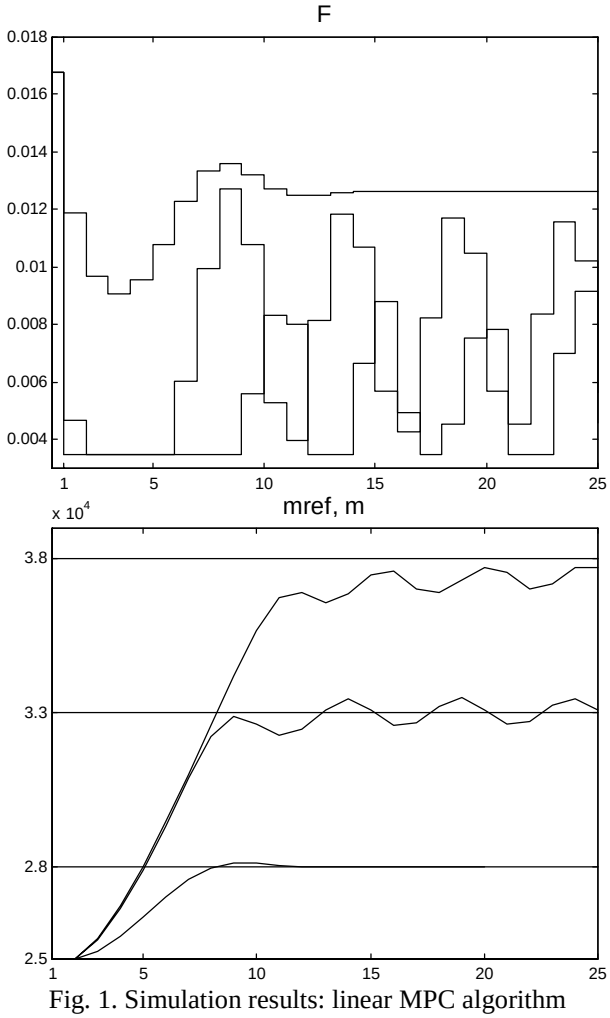

Fig. 1. Simulation results: linear MPC algorithm

For comparison purposes three different algorithms were simulated: linear MPC, MPC-INPL and MPC-NO with Nonlinear Optimisation. During the experiments carried out the plant was simulated using the fundamental model (37) while the control algorithms based on black-box models. The linear model used in linear MPC was obtained taking into account the nominal operating point. Nonlinear algorithms, i.e. MPC-INPL and MPC-NO, based on the same neural model with 6 hidden nodes. Both empirical models, i.e. linear and nonlinear, had the same inputs: $u(k–2)$, $y(k–1)$, $y(k–2)$, where $u(k)=100(F(k)–F_0)$, $y(k)=0.0001(m(k)–m_0)$. The horizons were set to: H_1=2, H=10 and H_u=3, the weighting matrices were $\boldsymbol{M}=\boldsymbol{I}$, $\boldsymbol{\Lambda}=\lambda\boldsymbol{I}$, where λ=0,2, the sampling time was set to 1.8 min. The linearisation point in the MPC-INPL algorithm was defined by (6).

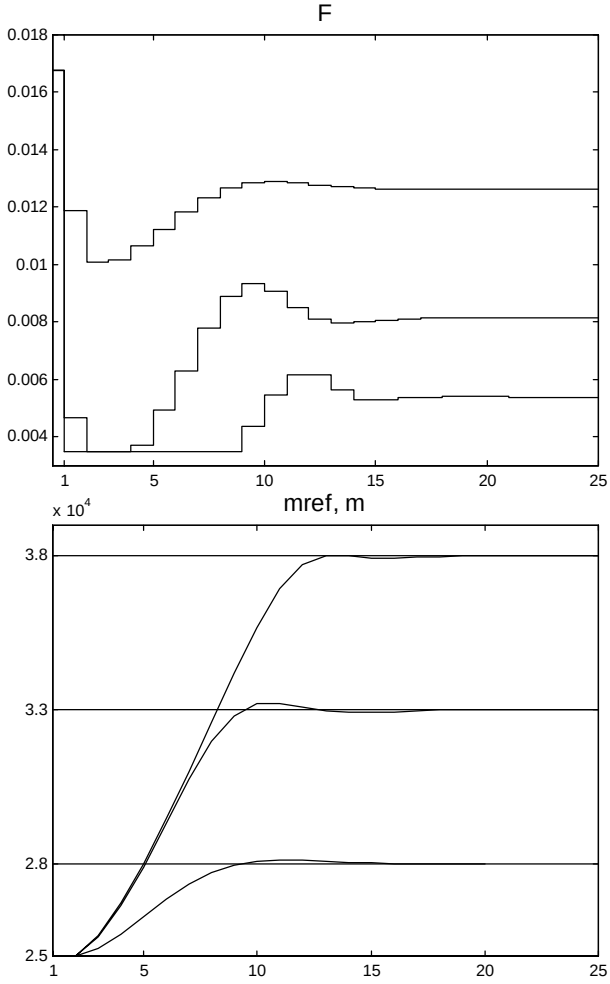

Fig. 2. Simulation results: nonlinear MPC-INPL algorithm

Selected simulation results are depicted in Fig. 1, Fig. 2 and Fig. 3. The reference trajectory is changed at time instant k=1 from its nominal value to 28000 kg/kmol, 33000 kg/kmol and 38000 kg/kmol, respectively. The linear MPC algorithm is stable only in the case of the smallest change. Both MPC-INPL and MPC-NO algorithms works well, their closed-loop performance is fairly similar, yet one can notice that the control increments in the MPC-NO algorithm are bigger, which leads to faster output responses and slightly bigger overshoot. On the other hand, the MPC-NO algorithm, in which the nonlinear model is used both for prediction and optimisation purposes, is far more computationally demanding and, in general, vulnerable to local minima. In the case of the MPC–INPL algorithm such a problem does not exist.

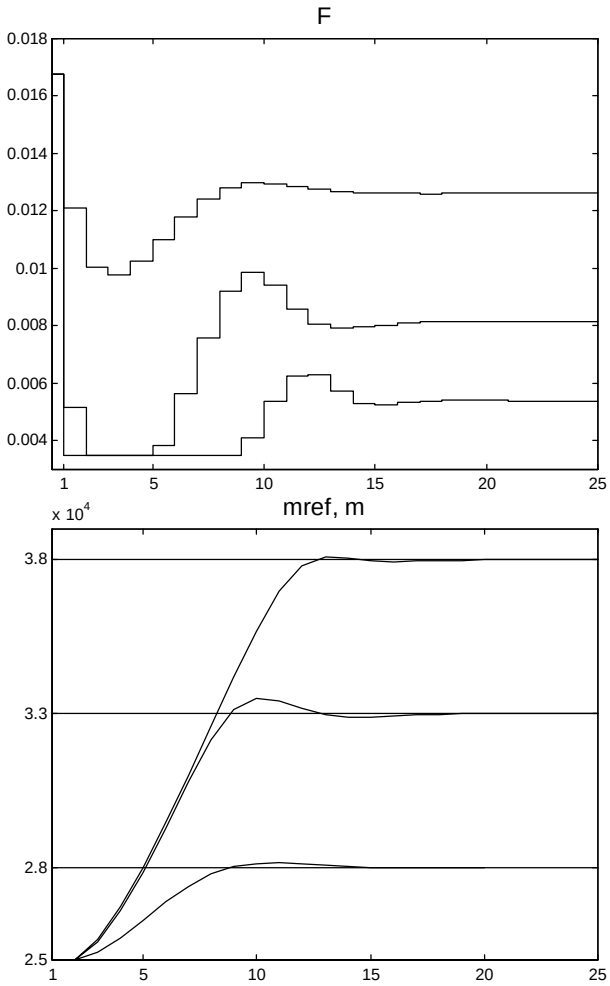

Fig. 3. Simulation results: nonlinear MPC-NO algorithm

5 Summary

Computational efficiency and ability to handle nonlinear processes are the advantages of the suboptimal MPC-INPL algorithm. It results in closed-loop behaviour comparable to that obtained when the MPC-NO algorithm, with nonlinear, in general non-convex, optimisation is applied. The proposed approach leads to a series of convex quadratic programming problems, the solution to which can be found within foreseeable time frame, it does not suffer from the false local minima problem. In comparison with existing linearisation-based iterative algorithms [3,5] the structure and implementation of the presented approach is much simpler.

Although the general MPC-INPL algorithm can be used with wide class of nonlinear models, for example fundamental ones which rely on first-principles equations, in this paper the emphasis is put on feedforward neural-network implementation. As far as practical applications are concerned it is important that such models have simple, regular structure and do not suffer from "the curse of dimensionality" phenomenon (crucial in multivariable cases). They can be easily incorporated into the described algorithm and used to on-line control.

Acknowledgement

The work presented in this paper was supported by the research grant no. 503G/0004/002 from the Dean of the Faculty of Electronics and Information Technology, WUT.

References

[1] R. Babuska, J. M. Sousa, H. B. Verbruggen. Predictive control of nonlinear systems based on fuzzy and neural models. *European Control Conference*, Karlsruhe, paper F1032-5, (1999).

[2] H. H. J. Bloemen, T. J. J. van den Boom, H. B. Verbruggen. Model-based predictive control for Hammerstein-Wiener systems. *International Journal of Control*, vol. 74, no. 5, pp. 482-495, (2001).

[3] F. Declercq, R. de Keyser. Suboptimal nonlinear predictive controllers. *International Journal of Applied Mathematics and Computer Science*, vol. 9, no. 1, pp. 129-148, (1999).

[4] C. E. Garcia. Quadratic dynamic matrix control of nonlinear processes: an application to a batch reactor process. *AIChE National Meeting*, San Francisco, (1984).

[5] W. C. Li, T. Biegler. Multistep, Newton-type control strategies for constrained, nonlinear systems. *Chem. Eng. Res. Des.*, vol. 67, pp. 562-577, (1989).

[6] J. M. Maciejowski. Predictive control with constraints. Prentice Hall, (2002).

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

[8] 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, pp. 1285-1301, (1996).

[9] D. Q. Mayne. Control of constrained dynamic systems. European Journal of Control, vol. 7, no. 2/3, pp. 87-99, (2001).

[10] D. Megias, J. Serrano, M. Y. El Ghoumari. Extended linearised predictive control: practical control algorithms for non-linear systems. *European Control Conference*, Karlsruhe, paper F0883, (1999).

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

[12] G. R. Sriniwas, Y. Arkun. A global solution to the nonlinear model predictive control algorithms using polynomial ARX models. *Computers and Chemical Engineering*, vol. 21, pp. 431-439, (1997).

[13] P. Tatjewski. Advanced control of industrial processes, Structures and algorithms (in Polish). Akademicka Oficyna Wydawnicza Exit, Warszawa, (2002).

[14] A. Zheng. A computationally efficient nonlinear MPC algorithm. *American Control Conference*, Albuquerque, pp. 1623-1627, (1997).