On-Line Trajectory-Based Linearisation of Neural Models for a Computationally Efficient Predictive Control Algorithm

Maciej Ławryńczuk

Institute of Control and Computation Engineering,
Warsaw University of Technology
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
M.Lawrynczuk@ia.pw.edu.pl

Abstract. The direct application of a neural model in Model Predictive Control (MPC) algorithms results in a nonlinear, in general non-convex, optimisation problem which must be solved on-line. A linear approximation of the model for the current operating point can be used for prediction in MPC, but for significantly nonlinear processes control accuracy may be not sufficient. MPC algorithm in which the neural model is linearised on-line along a trajectory is discussed. The control policy is calculated from a quadratic programming problem, nonlinear optimisation is not necessary. Accuracy and computational burden of the algorithm are demonstrated for a high-purity high-pressure distillation column.

1 Introduction

A unique feature of Model Predictive Control (MPC) algorithms is the fact that a dynamic model of the process is directly used on-line to predict its behavior over some time horizon and to optimise the future control policy [8,13]. When compared with other control techniques, their advantages are: constraints can be easily imposed on process inputs (manipulated variables) and outputs (controlled variables), they are able to control multivariable processes very efficiently, they can be applied for processes with difficult dynamic properties (e.g. with significant time-delays or the inverse response). In consequence, MPC algorithms have been successfully used for years in thousands of advanced industrial applications, e.g. in refineries, in chemical engineering, in the paper industry, in mining and metallurgy, in food processing, in the automobile industry and even in aerospace [12].

In the simplest case linear models are used for prediction in MPC. Because the majority of technological processes have nonlinear properties, linear MPC techniques may give insufficient control accuracy. Nonlinear MPC algorithms in which nonlinear models are used have been researched over the last years [4,10,13,14]. Different nonlinear models can be used in MPC, e.g. fuzzy structures, polynomials, Volterra series, wavelets. Neural models are particularly interesting, because they offer excellent approximation accuracy, as practical experience clearly indicates, they have a moderate number of parameters and their

L. Rutkowski et al. (Eds.): ICAISC 2012, Part I, LNCS 7267, pp. 126–134, 2012.

structure is not complicated. As a result, neural models can be efficiently used in different nonlinear MPC algorithms [6,11,13].

When a neural model is directly used for prediction in MPC, a nonlinear, in general non-convex, optimisation problem must be solved on-line at each sampling instant. Despite significant progress in optimisation algorithms [1,2,9,15], practical application of on-line nonlinear optimisation is always an issue. Since the solution must be obtained in real-time, low computational complexity is very desirable. A straightforward solution is to calculate successively on-line a linear approximation of the neural model and use the linearised model for prediction [6]. Unfortunately, for significantly nonlinear processes obtained control accuracy may be not sufficient. This paper discusses an MPC algorithm in which the neural model is linearised on-line along a trajectory. The control policy is calculated on-line from a quadratic programming problem, nonlinear optimisation is not necessary. Control accuracy and computational burden of the described algorithm are demonstrated for a high-purity high-pressure distillation column.

2 Model Predictive Control (MPC) Algorithms

In MPC algorithms at each consecutive sampling instant k, $k = 0, 1, 2, \ldots$, a set of future control increments

$$\triangle \boldsymbol{u}(k) = [\triangle u(k|k)\ \triangle u(k+1|k) \ldots \triangle u(k+N_{\mathrm{u}}-1|k)]^{\mathrm{T}} \tag{1}$$

is calculated, where $\triangle u(k+p|k) = u(k+p|k) - u(k+p-1|k)$. It is assumed that $\triangle u(k+p|k) = 0$ for $p \geq N_{\mathrm{u}}$, where N_{u} is the control horizon. The objective is to minimise differences between the reference trajectory $y^{\mathrm{ref}}(k+p|k)$ and predicted values of the output $\hat{y}(k+p|k)$ over the prediction horizon $N \geq N_{\mathrm{u}}$. Constraints are usually imposed on input and output variables. Future control increments (1) are determined from the following MPC optimisation task (hard output constraints are used for simplicity of presentation)

$$\begin{aligned} &\min_{\triangle \boldsymbol{u}(k)} \left\{ \sum_{p=1}^{N} (y^{\mathrm{ref}}(k+p|k) - \hat{y}(k+p|k))^2 + \lambda \sum_{p=0}^{N_{\mathrm{u}}-1} (\triangle u(k+p|k))^2 \right\} \\ &\text{subject to} \\ &\quad u^{\min} \leq u(k+p|k) \leq u^{\max}, \quad p = 0, \ldots, N_{\mathrm{u}} - 1 \\ &\quad -\triangle u^{\max} \leq \triangle u(k+p|k) \leq \triangle u^{\max}, \quad p = 0, \ldots, N_{\mathrm{u}} - 1 \\ &\quad y^{\min} \leq \hat{y}(k+p|k) \leq y^{\max}, \quad p = 1, \ldots, N \end{aligned} \tag{2}$$

Only the first element of the determined sequence (1) is applied to the process, i.e. $u(k) = \triangle u(k|k) + u(k-1)$. At the next sampling instant, $k+1$, the output measurement is updated, and the whole procedure is repeated.

Let the dynamic process under consideration be described by the following discrete-time Nonlinear Auto Regressive with eXternal input (NARX) model

$$y(k) = f(\boldsymbol{x}(k)) = f(u(k-\tau), \ldots, u(k-n_{\mathrm{B}}), y(k-1), \ldots, y(k-n_{\mathrm{A}})) \tag{3}$$

As the model a neural network of Multi Layer Perceptron (MLP) or Radial Radial Basis Function (RBF) type [3] can be used. Consecutive output predictions over the prediction horizon ($p = 1, \ldots, N$) are calculated recurrently

$$\hat{y}(k+p|k) = f(\underbrace{u(k-\tau+p|k), \ldots, u(k|k)}_{I_{\mathrm{uf}}(p)}, \underbrace{u(k-1), \ldots, u(k-n_{\mathrm{B}}+p)}_{I_{\mathrm{u}}-I_{\mathrm{uf}}(p)},$$
$$\underbrace{\hat{y}(k-1+p|k), \ldots, \hat{y}(k+1|k)}_{I_{\mathrm{yf}}(p)}, \underbrace{y(k), \ldots, y(k-n_{\mathrm{A}}+p)}_{n_{\mathrm{A}}-I_{\mathrm{yf}}(p)}) + d(k)$$

where $I_{\mathrm{uf}}(p) = \max(\min(p-\tau+1, I_{\mathrm{u}}), 0)$, $I_{\mathrm{yf}}(p) = \min(p-1, n_{\mathrm{A}})$ and $d(k)$ is an estimation of the unmeasured disturbance [13]. Since the model is nonlinear, future predictions are nonlinear functions of the calculated control policy (1). As a result, the MPC optimisation problem (2) is in fact a nonlinear, in general non-convex, task which must be solved in real time on-line. Computational complexity of such an approach may be high and the whole algorithm may be unable to find the solution within the required time.

The general idea of reducing computational burden of nonlinear MPC is quite intuitive: at each sampling instant a linear approximation

$$y(k) = \sum_{l=1}^{n_{\mathrm{B}}} b_l(k) u(k-l) - \sum_{l=1}^{n_{\mathrm{A}}} a_l(k) y(k-l)$$

of the nonlinear neural model (3) is obtained on-line for the current operating point, where

$$a_l(k) = -\frac{\partial f(\boldsymbol{x}(k))}{\partial y(k-l)}, \quad b_l(k) = \frac{\partial f(\boldsymbol{x}(k))}{\partial u(k-l)}$$

are coefficients of the linearised model. The linearised model is used for prediction over the whole prediction horizon. Thanks to linearisation, predictions $\hat{y}(k+1|k), \ldots, \hat{y}(k+N|k)$ are linear functions of future control increments (1), i.e. the decision variables of the algorithm. In consequence, the MPC optimisation problem (2) becomes a quadratic programming task. The described linearisation method is used in the MPC algorithm with Nonlinear Prediction and Linearisation (MPC-NPL) [6,7,13]. During calculations the structure of the neural model is exploited.

3 MPC Algorithm with Nonlinear Prediction and Linearisation along the Trajectory (MPC-NPLT)

In the MPC-NPL algorithm linearisation is carried out for the current operating point of the process and the same linearised model is used for prediction over the whole prediction horizon. Intuitively, prediction accuracy of such a model may be insufficient, in particular when the process is significantly nonlinear and changes of the reference trajectory are fast and big. A potentially better method is to linearise the model for an assumed future input trajectory

$$\boldsymbol{u}^{\mathrm{traj}}(k) = \left[u^{\mathrm{traj}}(k|k) \ldots u^{\mathrm{traj}}(k+N_{\mathrm{u}}-1|k)\right]^{\mathrm{T}}$$

of course remembering that $u^{\text{traj}}(k+p|k) = u^{\text{traj}}(k+N_{\text{u}}-1|k)$ for $p = N_{\text{u}}, \ldots, N$. The input trajectory $\boldsymbol{u}^{\text{traj}}(k)$ corresponds to the future output trajectory

$$\hat{\boldsymbol{y}}^{\text{traj}}(k) = \left[\hat{y}^{\text{traj}}(k+1|k) \ldots \hat{y}^{\text{traj}}(k+N|k)\right]^{\text{T}}$$

Recalling the Taylor series formula for a scalar function $y(x)\colon \mathbb{R} \to \mathbb{R}$

$$y(x) = y(\bar{x}) + \left.\frac{\text{d}y(x)}{\text{d}x}\right|_{x=\bar{x}} (x - \bar{x}) + \ldots$$

a linear approximation of the nonlinear trajectory $\hat{\boldsymbol{y}}(\boldsymbol{u}(k))\colon \mathbb{R}^{N_{\text{u}}} \to \mathbb{R}^{N}$ where

$$\begin{aligned} \hat{\boldsymbol{y}}(k) &= [\hat{y}(k+1|k) \ldots \hat{y}(k+N|k)]^{\text{T}} \\ \boldsymbol{u}(k) &= [u(k|k) \ldots u(k+N_{\text{u}}-1|k)]^{\text{T}} \end{aligned}$$

along the trajectory $\hat{\boldsymbol{y}}^{\text{traj}}(k)$ is

$$\hat{\boldsymbol{y}}(k) = \hat{\boldsymbol{y}}^{\text{traj}}(k) + \boldsymbol{H}(k)(\boldsymbol{u}(k) - \boldsymbol{u}^{\text{traj}}(k)) \tag{4}$$

where

$$\boldsymbol{H}(k) = \left.\frac{\text{d}\hat{\boldsymbol{y}}(k)}{\text{d}\boldsymbol{u}(k)}\right|_{\substack{\hat{\boldsymbol{y}}(k)=\hat{\boldsymbol{y}}^{\text{traj}}(k) \\ \boldsymbol{u}(k)=\boldsymbol{u}^{\text{traj}}(k)}} = \begin{bmatrix} \dfrac{\partial \hat{y}^{\text{traj}}(k+1|k)}{\partial u^{\text{traj}}(k|k)} & \cdots & \dfrac{\partial \hat{y}^{\text{traj}}(k+1|k)}{\partial u^{\text{traj}}(k+N_{\text{u}}-1|k)} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \hat{y}^{\text{traj}}(k+N|k)}{\partial u^{\text{traj}}(k|k)} & \cdots & \dfrac{\partial \hat{y}^{\text{traj}}(k+N|k)}{\partial u^{\text{traj}}(k+N_{\text{u}}-1|k)} \end{bmatrix}$$

is a matrix of dimensionality $N \times N_{\text{u}}$. Thanks to using the prediction equation (4), the optimisation problem (2) becomes the quadratic programming task

$$\begin{aligned} &\min_{\triangle \boldsymbol{u}(k)} \Big\{ J(k) = \big\| \boldsymbol{y}^{\text{ref}}(k) - \boldsymbol{H}(k)\boldsymbol{J}\triangle\boldsymbol{u}(k) - \hat{\boldsymbol{y}}^{\text{traj}}(k) \\ &\qquad\qquad - \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\text{traj}}(k)) \big\|^2 + \|\triangle\boldsymbol{u}(k)\|^2_{\boldsymbol{\Lambda}} \Big\} \\ &\text{subject to} \\ &\quad \boldsymbol{u}^{\min} \le \boldsymbol{J}\triangle\boldsymbol{u}(k) + \boldsymbol{u}(k-1) \le \boldsymbol{u}^{\max} \\ &\quad -\triangle\boldsymbol{u}^{\max} \le \triangle\boldsymbol{u}(k) \le \triangle\boldsymbol{u}^{\max} \\ &\quad \boldsymbol{y}^{\min} \le \boldsymbol{H}(k)\boldsymbol{J}\triangle\boldsymbol{u}(k) + \hat{\boldsymbol{y}}^{\text{traj}}(k) + \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\text{traj}}(k)) \le \boldsymbol{y}^{\max} \end{aligned} \tag{5}$$

where

$$\begin{aligned} \boldsymbol{y}^{\text{ref}}(k) &= \left[y^{\text{ref}}(k+1|k) \ldots y^{\text{ref}}(k+N|k)\right]^{\text{T}} \\ \boldsymbol{y}^{\min} &= \left[y^{\min} \ldots y^{\min}\right]^{\text{T}} \\ \boldsymbol{y}^{\max} &= \left[y^{\max} \ldots y^{\max}\right]^{\text{T}} \end{aligned}$$

are vectors of length N,

$$\begin{aligned}
\boldsymbol{u}^{\min} &= \left[u^{\min} \ldots u^{\min}\right]^{\mathrm{T}} \\
\boldsymbol{u}^{\max} &= \left[u^{\max} \ldots u^{\max}\right]^{\mathrm{T}} \\
\boldsymbol{u}(k-1) &= \left[u(k-1) \ldots u(k-1)\right]^{\mathrm{T}} \\
\triangle\boldsymbol{u}^{\max} &= \left[\triangle u^{\max} \ldots \triangle u^{\max}\right]^{\mathrm{T}}
\end{aligned}$$

are vectors of length N_{u}, $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda, \ldots, \lambda)$, $\boldsymbol{J}$ is the all ones lower triangular matrix of dimensionality $N_{\mathrm{u}} \times N_{\mathrm{u}}$.

Steps repeated at each sampling instant k of the MPC-NPLT algorithm are:

1. The neural model is used to find the future output trajectory $\hat{\boldsymbol{y}}^{\mathrm{traj}}(k)$ which corresponds to the assumed input trajectory $\boldsymbol{u}^{\mathrm{traj}}(k)$.
2. The neural model is linearised along the trajectory $\hat{\boldsymbol{y}}^{\mathrm{traj}}(k)$: the matrix $\boldsymbol{H}(k)$ is obtained.
3. The quadratic programming task (5) is solved to find $\triangle\boldsymbol{u}(k)$.
4. The first element of the obtained future control policy is applied to the process: $u(k) = \triangle u(k|k) + u(k-1)$.
5. Set $k := k+1$, go to step 1.

During calculation of the output trajectory $\hat{\boldsymbol{y}}^{\mathrm{traj}}(k)$ and linearisation along this trajectory (calculation of the matrix $\boldsymbol{H}(k)$) the structure of the neural model is exploited.

Selection of the future input trajectory $\boldsymbol{u}^{\mathrm{traj}}(k)$ affects the linearisation accuracy and, in consequence, quality of control. A straightforward choice is to use the control signal calculated at the previous sampling instant, i.e.

$$\boldsymbol{u}^{\mathrm{traj}}(k) = \left[u(k-1) \ldots u(k-1)\right]^{\mathrm{T}}$$

As a result, the neural model is linearised along the free trajectory, the algorithm is denoted by MPC-NPLT$_{\mathbf{y}^0(\mathrm{k})}$. The alternative is to use $N_{\mathrm{u}}-1$ elements of the future control sequence calculated at the previous sampling instant (the quantity $u(k-1|k-1)$ is actually used for control at the sampling instant $k-1$), i.e.

$$\boldsymbol{u}^{\mathrm{traj}}(k) = \left[u(k|k-1) \ldots u(k+N_{\mathrm{u}}-3|k-1)\; u(k+N_{\mathrm{u}}-2|k-1)\; u(k+N_{\mathrm{u}}-2|k-1)\right]^{\mathrm{T}}$$

The algorithm is denoted by MPC-NPLT$_{\hat{\mathbf{y}}(\mathrm{k}-1)}$. It is possible to combine MPC-NPL and MPC-NPLT techniques (the MPC-NPL-NPLT algorithm). In the first phase of the hybrid approach the neural model is linearised for the current operating point, the MPC-NPL quadratic programming task is solved. In the second phase the neural model is linearised once again along the predicted trajectory which corresponds to the obtained input trajectory and the MPC-NPLT quadratic programming task (5) is solved. Furthermore, linearisation along the predicted trajectory can be repeated in an iterative manner [5]: nonlinear prediction, linearisation and quadrating programming are repeated a few times at each sampling instant.

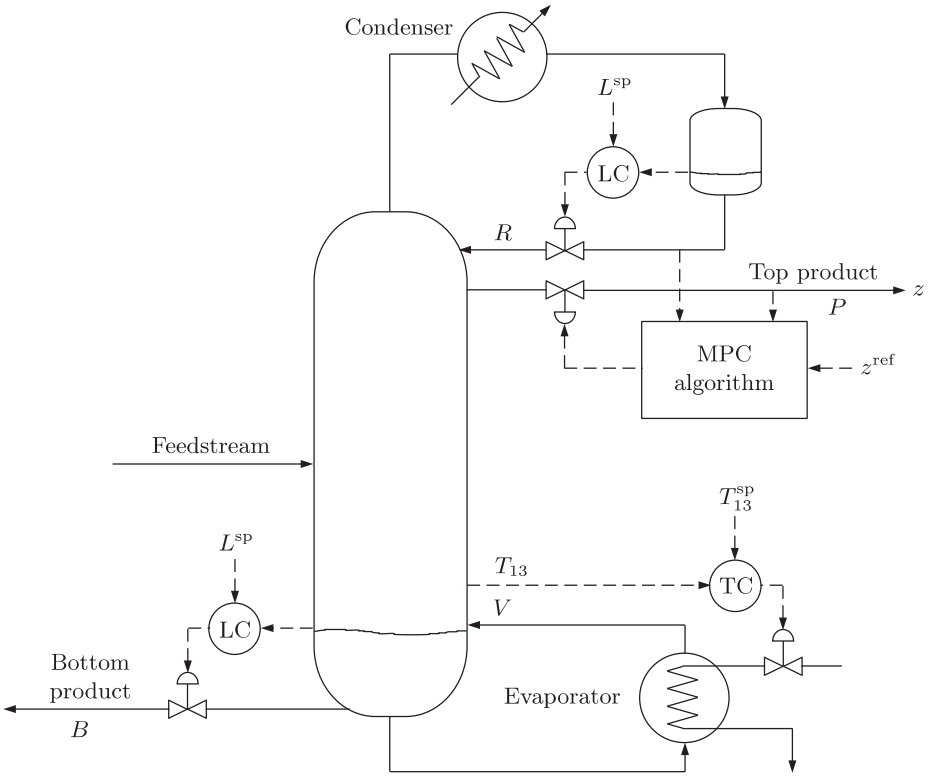

Fig. 1. High-purity ethylene-ethane distillation column control system structure

4 Simulations

The considered process is a high-purity, high-pressure (1.93 MPa) ethylene-ethane distillation column shown in Fig. 1. The feed stream consists of ethylene (approx. 80%), ethane (approx. 20%) and traces of hydrogen, methane and propylene. The distillation product (the top product) is ethylene which can contain up to 1000 ppm (parts per million) of ethane. The MPC algorithm must be able to increase relatively fast the impurity level of the product. Reducing the purity of the product results in decreasing energy consumption. Production scale is very big, nominal value of the product stream flow rate is 43 ton/h.

The supervisory control loop has one manipulated variable r, which is the reflux ratio $r = R/P$, where R and P are reflux and product stream flow rates, respectively, and one controlled variable z, which is the impurity of the product. The column has 121 trays, the feed stream is delivered to the tray number 37. The reflux is delivered to the column by the top tray and the product is taken from the tray number 110.

As shown in [5,7,13,14] the process is significantly nonlinear and difficult to control. A simple linear model is inadequate, hence, the MPC algorithm which

uses such a model does not work properly. In contrast to the linear model, a neural model (of the MLP structure) is very accurate as discussed in [5]. Because the linearisation method affects control accuracy, the following nonlinear MPC algorithms based on the same neural model are compared:

a) The rudimentary MPC-NPL algorithm with on-line linearisation for the current operating point and quadratic programming [6,7,13].
b) Two versions of the discussed MPC-NPLT algorithm (MPC-NPLT$_{\mathbf{y}^0(\mathrm{k})}$ and MPC-NPLT$_{\hat{\mathbf{y}}(\mathrm{k}-1)}$) with linearisation along the trajectory and quadratic programming.
c) The "ideal" algorithm with Nonlinear Optimisation (MPC-NO) [6,13].

Parameters of all algorithms are the same: $N = 10$, $N_{\mathrm{u}} = 3$, $\lambda = 2$, constraints are $r^{\min} = 4.051$, $r^{\max} = 4.4571$. Three reference trajectories are considered: at the sampling instant $k = 1$ the trajectory changes from 100 ppm to 350 ppm, 600 ppm and 850 ppm, respectively.

Fig. 2 compares the MPC-NPL algorithm with on-line linearisation for the current operating point and the MPC-NO approach. Unfortunately, due to the nonlinear nature of the distillation process, when the linearised model obtained for the current operating point is used for the whole prediction horizon, its inaccuracy is important, the algorithm gives significantly slower trajectories than the MPC-NO approach. Slow behaviour of the MPC-NPL algorithm is disadvantageous in light of a very big production scale.

As shown in Fig. 3, both versions of the MPC-NPLT algorithm give much faster trajectories than the MPC-NPL approach. The algorithm with linearisation along the optimal trajectory calculated at the previous sampling instant (MPC-NPLT$_{\hat{\mathbf{y}}(\mathrm{k}-1)}$) is faster than the algorithm with linearisation along the free trajectory (MPC-NPLT$_{\mathbf{y}^0(\mathrm{k})}$). It is not surprising, because in the first approach for linearisation predicted behaviour of the process is taken into account whereas in the second one the influence of the past is only considered.

Table 1 shows accuracy (in terms of Sum of Squared Errors, SSE) and computational load (in terms of floating point operations, MFLOPS) of compared nonlinear algorithms, summarised results for all three reference trajectories are given. Computational burden of the MPC-NPLT algorithm is approximately 6.43 times smaller when compared with that of the MPC-NO approach.

Table 1. Accuracy (SSE) and computational load (MFLOPS) of compared nonlinear MPC algorithms based on the same neural model

Algorithm	Optimisation	SSE	MFLOPS
MPC-NPL	Quadratic	5.3717×10^6	0.1545
MPC-NPLT$_{\mathbf{y}^0(\mathrm{k})}$	Quadratic	5.1085×10^6	0.3313
MPC-NPLT$_{\hat{\mathbf{y}}(\mathrm{k}-1)}$	Quadratic	4.8599×10^6	0.3408
MPC-NO	Nonlinear	4.3869×10^6	2.1299

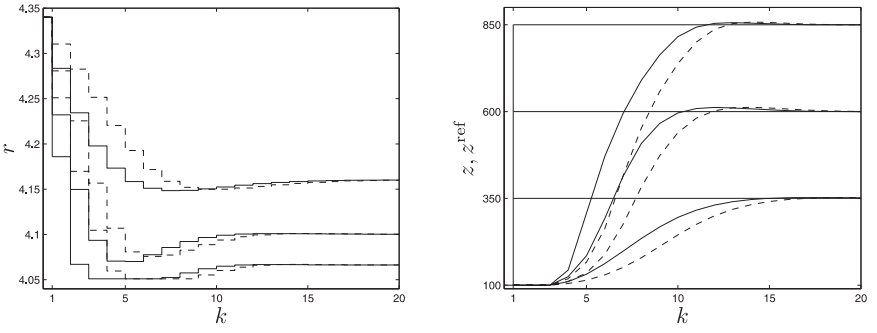

Fig. 2. The MPC-NPL algorithm with linearisation for the current operating point and quadratic programming (*dashed line*) vs. the MPC-NO algorithm with nonlinear optimisation (*solid line*)

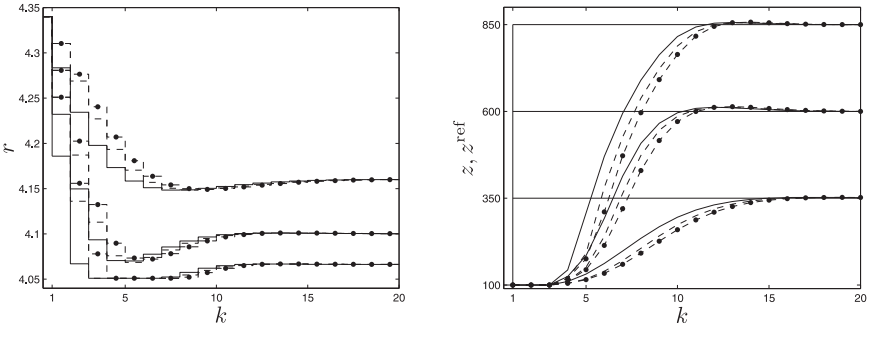

Fig. 3. The MPC-NPLT$_{\mathbf{y}^0(\mathrm{k})}$ algorithm (*dash-dotted line*) and the MPC-NPLT$_{\hat{\mathbf{y}}(\mathrm{k}-1)}$ algorithm (*dashed line*) with linearisation along the trajectory and quadratic programming vs. the MPC-NO algorithm with nonlinear optimisation (*solid line*)

5 Conclusions

For the considered distillation column the MPC-NPLT algorithm in which the neural model is linearised on-line along a trajectory is much faster than the rudimentary MPC-NPL algorithm with linearisation for the current operating point. At each sampling instant of the MPC-NPLT algorithm only one quadratic programming problem is solved, nonlinear optimisation is not necessary. Of course, linearisation along the predicted trajectory and quadratic programming can be repeated a few times at each sampling instant [5], but it increases the computational burden. Although in simulations presented in the paper the MLP neural model is used, the described algorithm is very general, different types of models can be used. The chosen model structure must be taken into account during calculation of the output trajectory $\hat{\boldsymbol{y}}^{\mathrm{traj}}(k)$ and linearisation along this trajectory (calculation of the matrix $\boldsymbol{H}(k)$).

Acknowledgement. The work presented in this paper was supported by Polish national budget funds for science.

References

1. Biegler, L.T., Grossmann, I.E.: Retrospective on optimization. Computers and Chemical Engineering 28, 1169–1192 (2004)
2. Grossmann, I.E., Biegler, L.T.: Part II. Future perspective on optimization. Computers and Chemical Engineering 28, 1193–1218 (2004)
3. Haykin, S.: Neural networks–a comprehensive foundation. Prentice Hall, Englewood Cliffs (1999)
4. Henson, M.A.: Nonlinear model predictive control: current status and future directions. Computers and Chemical Engineering 23, 187–202 (1998)
5. Ławryńczuk, M.: On improving accuracy of computationally efficient nonlinear predictive control based on neural models. Chemical Engineering Science 66, 5253–5267 (2011)
6. Ławryńczuk, M.: A family of model predictive control algorithms with artificial neural networks. International Journal of Applied Mathematics and Computer Science 17, 217–232 (2007)
7. Ławryńczuk, M., Tatjewski, P.: An Efficient Nonlinear Predictive Control Algorithm with Neural Models and Its Application to a High-Purity Distillation Process. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 76–85. Springer, Heidelberg (2006)
8. Maciejowski, J.M.: Predictive control with constraints. Prentice Hall, Harlow (2002)
9. Martinsen, F., Biegler, L.T., Foss, B.A.: A new optimization algorithm with application to nonlinear MPC. Journal of Process Control 14, 853–865 (2004)
10. Morari, M., Lee, J.H.: Model predictive control: past, present and future. Computers and Chemical Engineering 23, 667–682 (1999)
11. Nørgaard, M., Ravn, O., Poulsen, N.K., Hansen, L.K.: Neural networks for modelling and control of dynamic systems. Springer, London (2000)
12. Qin, S.J., Badgwell, T.A.: A survey of industrial model predictive control technology. Control Engineering Practice 11, 733–764 (2003)
13. Tatjewski, P.: Advanced control of industrial processes. Structures and Algorithms. Springer, London (2007)
14. Tatjewski, P., Ławryńczuk, M.: Soft computing in model-based predictive control. International Journal of Applied Mathematics and Computer Science 16, 101–120 (2006)
15. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming 106, 25–57 (2006)