Int. J. Appl. Math. Comput. Sci., 2020, Vol. 30, No. 1, 35–46
DOI: 10.34768/amcs-2020-0003

NONLINEAR MODEL PREDICTIVE CONTROL FOR PROCESSES WITH COMPLEX DYNAMICS: A PARAMETERISATION APPROACH USING LAGUERRE FUNCTIONS

MACIEJ ŁAWRYŃCZUK [a]

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

Classical model predictive control (MPC) algorithms need very long horizons when the controlled process has complex dynamics. In particular, the control horizon, which determines the number of decision variables optimised on-line at each sampling instant, is crucial since it significantly affects computational complexity. This work discusses a nonlinear MPC algorithm with on-line trajectory linearisation, which makes it possible to formulate a quadratic optimisation problem, as well as parameterisation using Laguerre functions, which reduces the number of decision variables. Simulation results of classical (not parameterised) MPC algorithms and some strategies with parameterisation are thoroughly compared. It is shown that for a benchmark system the MPC algorithm with on-line linearisation and parameterisation gives very good quality of control, comparable with that possible in classical MPC with long horizons and nonlinear optimisation.

Keywords: process control, nonlinear model predictive control, Laguerre functions, linearisation.

1. Introduction

Various control approaches have been developed for nonlinear processes, e.g., fuzzy (Harrabi *et al.*, 2018), neural (Li *et al.*, 2019), model predictive control (MPC) (Maciejowski, 2002; Tatjewski, 2007), robust (Pazera *et al.*, 2018), adaptive (Witkowska and Śmierzchalski, 2018) and hybrid (Falconí *et al.*, 2018). In particular, MPC algorithms are characterised by very good quality of control, particularly for multivariable processes. This is because MPC algorithms calculate repeatedly on-line the best possible sequence of manipulated variable(s) taking into account future predictions found from a dynamical model of the controlled process, and solve an optimisation problem whose objective is to minimise the predicted control errors. Typically, MPC algorithms have been used in industrial applications (e.g., Karimi Pour *et al.*, 2018), frequently in supervisory control and multilayer control system structures with set-point optimisation (Tatjewski, 2007). Currently, thanks to availability of cheap but powerful and fast microcontrollers, MPC algorithms are used in embedded systems (Chaber and Ławryńczuk, 2019; Takács *et al.*, 2016). In contrast to industrial applications, embedded systems are characterised by very short sampling times, of the order of milliseconds.

Computational complexity has always been an issue in MPC. The time necessary to calculate the values of the manipulated variable(s) must not exceed that determined by the sampling time. In practical solutions the control horizon is typically shorter than the prediction one, but such an approach may be only used for processes with relatively simple dynamics for which shortening the control horizon does not lead to deterioration of control quality. In predictive functional control (PFC), which uses linear models for prediction (Richalet and O'Donovan, 2009), the sequence of future manipulated variable(s) is parameterised using a set of basis functions. The optimisation routine does not directly calculate the future manipulated variable(s), but the coefficients of the basis functions. A different idea is utilised in the explicit linear quadratic regulator for linear constrained systems (Bemporad *et al.*, 2002). In this approach the quadratic optimisation problem is not solved on-line, but a number of simple explicit local controllers are activated/deactivated depending on the operating point of the process. For nonlinear MPC algorithms fast nonlinear

optimisation methods are possible (e.g., Wang and Boyd, 2010). Alternatively, computationally efficient nonlinear MPC algorithms with on-line linearisation may be used (Ławrynczuk, 2014). Successive model or trajectory linearisation leads to simple quadratic optimisation tasks—nonlinear optimisation is not necessary.

Laguerre, Kautz and other orthonormal functions may be successfully used for modelling dynamical systems in linear (Oliveira *et al*., 2011) and nonlinear (Oliveira *et al*., 2012) cases, respectively. Furthermore, application of orthonormal Laguerre functions to parameterise the calculated future sequence of the manipulatedation of variable(s) was used in MPC algorithms based on linear models: in continuous-time (Wang, 2001) and discrete-time (Wang, 2004) versions. A systematic tuning methodology to find parameters of Laguerre functions in parameterised MPC is discussed by Gutiérrez-Urquídez *et al*. (2015) or Khan and Rossiter (2013). MPC algorithms with parameterisation using Laguerre functions have been developed for different technological processes. Example applications include buildings (Bosschaerts *et al*., 2017), wave energy converters (Jama *et al*., 2018), magnetically actuated satellites (Kim *et al*., 2018), wind turbines (Lasheen *et al*., 2017), hexacopters (Ligthart *et al*., 2017) and power systems (Zheng *et al*., 2017).

This work shortly presents a nonlinear MPC algorithm for processes with complex dynamics which combines two concepts: on-line trajectory linearisation and parameterisation using Laguerre functions. Firstly, trajectory linearisation makes it possible to control nonlinear processes using easy-to-solve quadratic optimisation tasks. Secondly, since in the case of complicated dynamics reduction of the control horizon leads to low control quality, parameterisation of the calculated control policy makes it possible to reduce the number of actually optimised on-line decision variables but keeps the control horizon as long as necessary. In simulations, different configurations of classical, i.e., not parameterised, MPC algorithms and some strategies with parameterisation are thoroughly compared. The influence of parameters of Laguerre filters is investigated.

2. MPC problem formulation

Let the manipulated variable, i.e., the input of the process, be denoted by u and the controlled variable, i.e., the output of the process, be denoted by y.

In classical MPC algorithms as many as N_{u} (which is called the control horizon) future increments of the manipulated variable are calculated on-line at each sampling instant $k = 0, 1, \ldots,$

$$\triangle \boldsymbol{u}(k) = \begin{bmatrix} \triangle u(k|k) \\ \vdots \\ \triangle u(k+N_{\mathrm{u}}-1|k) \end{bmatrix}, \tag{1}$$

where $\triangle u(k+p|k)$ denotes a backward difference of the manipulated variable for the future sampling instant $k+p$ calculated at the current instant k. The decision variables of MPC are calculated from the following rudimentary MPC constrained optimisation problem:

$$\begin{aligned} \min_{\triangle \boldsymbol{u}(k)} \Big\{ J(k) = \sum_{p=1}^{N} (y^{\mathrm{sp}}(k+p|k) - \hat{y}(k+p|k))^2 \\ + \lambda \sum_{p=0}^{N_{\mathrm{u}}-1} (\triangle u(k+p|k))^2 \Big\}, \end{aligned}$$

subject to (2)

$$\begin{aligned} u^{\min} \le u(k+p|k) \le u^{\max}, \quad p = 0, \ldots, N_{\mathrm{u}}-1, \\ -\triangle u^{\max} \le \triangle u(k+p|k) \le \triangle u^{\max}, \\ p = 0, \ldots, N_{\mathrm{u}}-1, \\ y^{\min} \le \hat{y}(k+p|k) \le y^{\max}, \quad p = 1, \ldots, N. \end{aligned}$$

The first part of the minimised cost function $J(k)$ takes into account the sum of predicted control errors measured over the prediction horizon N, while the second one is a penalty term; $y^{\mathrm{sp}}(k+p|k)$ and $\hat{y}(k+p|k)$ denote the set-points and the predicted values for the future sampling instant $k+p$ known at the instant k, $\lambda > 0$ is a weighting coefficient, $u^{\min}$, $u^{\max}$ and $\triangle u^{\max}$ denote minimal and maximal values as well as the maximal change in the manipulated variable, respectively, $y^{\min}$ and $y^{\max}$ denote minimal and maximal values of the predicted controlled variable, respectively. Having calculated the optimal vector $\triangle \boldsymbol{u}^{\mathrm{opt}}(k)$, its first element is applied to the process, i.e., $u(k) = \triangle u^{\mathrm{opt}}(k|k) + u(k-1)$.

3. Parameterisation using Laguerre functions

Let $l_1(k),\ldots,l_{n_{\mathrm{L}}}(k)$ denote n_{L} Laguerre functions. The transfer function of the Laguerre function of order n is (Wahlberg, 1991)

$$G_n(z) = \frac{\sqrt{1-a^2}}{z-a} \left(\frac{1-az}{z-a} \right)^{n-1}, \tag{3}$$

where a is a scaling factor, often named a Laguerre pole. For stability, the condition $0 \le a < 1$ must be satisfied. The Laguerre functions are defined as inverse $\mathcal{Z}$-transforms of the transfer function $G(z)$, i.e.,

$$l_n(k) = \mathcal{Z}^{-1}(G_n(z)). \tag{4}$$

Taking into account the structure of the obtained Laguerre functions, it can be found that (Wang, 2004)

$$L(k+1) = \Omega L(k), \tag{5}$$

where a vector of length n_{L} is

$$L(k) = [l_1(k) \ \ldots \ l_{n_{\mathrm{L}}}(k)]^{\mathrm{T}}, \tag{6}$$

and an $n_{\mathrm{L}} \times n_{\mathrm{L}}$ matrix is

$$\Omega = \begin{bmatrix} a & 0 & 0 & \ldots & 0 \\ \beta & a & 0 & \ldots & 0 \\ -a\beta & \beta & a & \ldots & 0 \\ a^2\beta & -a\beta & \beta & \ldots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ (-a)^{n_{\mathrm{L}}-2}\beta & (-a)^{n_{\mathrm{L}}-3}\beta & \ldots & \beta & a \end{bmatrix}; \tag{7}$$

$\beta = 1 - a^2$ and the initial condition is

$$L(0) = \sqrt{1-a^2} \begin{bmatrix} 1 \\ -a \\ a^2 \\ -a^3 \\ \vdots \\ (-a)^{n_{\mathrm{L}}-1} \end{bmatrix}. \tag{8}$$

The future control increments are parameterised using the Laguerre functions (Wang, 2004) in the following way:

$$\triangle u(k+p|k) = \sum_{n=1}^{n_{\mathrm{L}}} l_n(p) c_n(k). \tag{9}$$

Using the vector notation, one has

$$\triangle u(k+p|k) = L^{\mathrm{T}}(p) c(k), \tag{10}$$

where a vector of coefficients is

$$c(k) = [c_1(k) \ \ldots \ c_{n_{\mathrm{L}}}(k)]^{\mathrm{T}}. \tag{11}$$

For the whole vector of future increments over the control horizon one has

$$\triangle \boldsymbol{u}(k) = \boldsymbol{L} c(k), \tag{12}$$

where an $N_{\mathrm{u}} \times n_{\mathrm{L}}$ matrix is

$$\boldsymbol{L} = \begin{bmatrix} l_1(0) & l_2(0) & \ldots & l_{n_{\mathrm{L}}}(0) \\ l_1(1) & l_2(1) & \ldots & l_{n_{\mathrm{L}}}(1) \\ \vdots & \vdots & \ddots & \vdots \\ l_1(N_{\mathrm{u}}-1) & l_2(N_{\mathrm{u}}-1) & \ldots & l_{n_{\mathrm{L}}}(N_{\mathrm{u}}-1) \end{bmatrix}. \tag{13}$$

In parameterised MPC the vector of decision variables is $c(k)$, not $\triangle \boldsymbol{u}(k)$. Since $n_{\mathrm{L}} < N_{\mathrm{u}}$, the number of decision variables used in the MPC optimisation problem solved on-line is reduced.

4. Nonlinear MPC with on-line trajectory linearisation and parameterisation using Laguerre functions

If a model used for prediction is linear, e.g., a step response, a discrete-time difference equation, a state-space model, future predictions of the controlled variable, i.e., $\hat{y}(k+1|k), \ldots, \hat{y}(k+N|k)$, are linear functions of the decision variables (1). Consequently, the rudimentary MPC optimisation problem (2) is of a quadratic optimisation type. On the other hand, for a nonlinear model the predictions are nonlinear and the resulting MPC optimisation task is nonlinear.

In order to reduce the computational effort, a number of computationally efficient nonlinear MPC algorithms with on-line linearisation have been formulated (Ławrynczuk, 2014). In the simplest case a linear approximation of the nonlinear model is successively calculated on-line for the current operating point of the process and used for prediction. In more advanced solutions a linear approximation of the predicted trajectory of the controlled variable (over the prediction horizon) is directly calculated. In all cases quite simple to solve quadratic optimisation MPC problems are obtained. However, such classical MPC algorithms with on-line linearisation are characterised by as many as N_{u} decision variables (1), which still may be a problem.

In this work only MPC with nonlinear prediction and linearisation around the trajectory (MPC-NPLT) is considered (Ławrynczuk, 2014). The predicted vector of the controlled variable is defined as

$$\hat{\boldsymbol{y}}(k) = \begin{bmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+N|k) \end{bmatrix}. \tag{14}$$

Its linear approximation along some assumed future trajectory of the manipulated variable,

$$\boldsymbol{u}^{\mathrm{traj}}(k) = \begin{bmatrix} u^{\mathrm{traj}}(k|k) \\ \vdots \\ u^{\mathrm{traj}}(k+N_{\mathrm{u}}-1|k) \end{bmatrix}, \tag{15}$$

is found using the Taylor series expansion. Provided that the process model is differentiable, as proved by Ławryńczuk (2014), the linearised trajectory is

$$\begin{aligned} \hat{\boldsymbol{y}}(k) = {} & \boldsymbol{H}(k)\boldsymbol{J}\triangle\boldsymbol{u}(k) \\ & + \hat{\boldsymbol{y}}^{\mathrm{traj}}(k) + \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\mathrm{traj}}(k)). \end{aligned} \tag{16}$$

The $N \times N_{\mathrm{u}}$ matrix of derivatives of the predicted trajectory of the controlled variable with respect to the

trajectory of the manipulated one has the structure

$$\boldsymbol{H}(k) = \left.\frac{\partial \hat{\boldsymbol{y}}(k)}{\partial \boldsymbol{u}(k)}\right|_{\substack{\hat{\boldsymbol{y}}(k)=\hat{\boldsymbol{y}}^{\mathrm{traj}}(k) \\ \boldsymbol{u}(k)=\boldsymbol{u}^{\mathrm{traj}}(k)}} \tag{17}$$

$$= \begin{bmatrix} \dfrac{\partial \hat{y}^{\mathrm{traj}}(k+1|k)}{\partial u^{\mathrm{traj}}(k|k)} & \cdots & \dfrac{\partial \hat{y}^{\mathrm{traj}}(k+1|k)}{\partial u^{\mathrm{traj}}(k+N_{\mathrm{u}}-1|k)} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \hat{y}^{\mathrm{traj}}(k+N|k)}{\partial u^{\mathrm{traj}}(k|k)} & \cdots & \dfrac{\partial \hat{y}^{\mathrm{traj}}(k+N|k)}{\partial u^{\mathrm{traj}}(k+N_{\mathrm{u}}-1|k)} \end{bmatrix}.$$

The $N_{\mathrm{u}} \times N_{\mathrm{u}}$ matrix $\boldsymbol{J}$ has zero upper-diagonal entries; the others are equal to 1. For the assumed trajectory $\boldsymbol{u}^{\mathrm{traj}}(k)$ of the manipulated variable, the resulting nonlinear trajectory

$$\hat{\boldsymbol{y}}^{\mathrm{traj}}(k) = \begin{bmatrix} \hat{y}^{\mathrm{traj}}(k+1|k) \\ \vdots \\ \hat{y}^{\mathrm{traj}}(k+N|k) \end{bmatrix}, \tag{18}$$

of the controlled variable is calculated successively using the nonlinear model. The vector $\boldsymbol{u}(k-1) = u(k-1)\boldsymbol{I}_{N_{\mathrm{u}},1}$, where $\boldsymbol{I}_{N_{\mathrm{u}},1}$ is an all-ones vector of length N_{u}.

In the classical nonlinear MPC algorithms with on-line linearisation the linear approximation (16) of the predicted trajectory is a function of as many as N_{u} independent future increments $\triangle\boldsymbol{u}(k)$. In order to obtain a computationally efficient MPC algorithm with a reduced number of decision variables, the parameterisation defined by Eqn. (12) is used. Hence, the linearised trajectory of the controlled variable becomes

$$\begin{aligned} \hat{\boldsymbol{y}}(k) = {} & \boldsymbol{H}(k)\boldsymbol{J}\boldsymbol{L}c(k) \\ & + \hat{\boldsymbol{y}}^{\mathrm{traj}}(k) + \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\mathrm{traj}}(k)). \end{aligned} \tag{19}$$

After some simple transformations, from the basic MPC optimisation task (2) and Eqn. (12) one obtains the following quadratic optimisation problem solved at each sampling instant of the MPC-NPLT algorithm with parameterisation (MPC-NPLTp):

$$\begin{aligned} \min_{c(k)} \Big\{ J(k) = {} & \left\| \boldsymbol{y}^{\mathrm{sp}}(k) - \boldsymbol{H}(k)\boldsymbol{J}\boldsymbol{L}c(k) - \hat{\boldsymbol{y}}^{\mathrm{traj}}(k) \right. \\ & \left. - \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\mathrm{traj}}(k)) \right\|^2 \\ & + \|\boldsymbol{L}c(k)\|^2_{\boldsymbol{\Lambda}} \Big\}, \end{aligned}$$

subject to (20)

$$\begin{aligned} \boldsymbol{u}^{\min} \le {} & \boldsymbol{J}\boldsymbol{L}c(k) + \boldsymbol{u}(k-1) \le \boldsymbol{u}^{\max}, \\ -\triangle\boldsymbol{u}^{\max} \le {} & \boldsymbol{L}c(k) \le \triangle\boldsymbol{u}^{\max}, \\ \boldsymbol{y}^{\min} \le {} & \boldsymbol{H}(k)\boldsymbol{J}\boldsymbol{L}c(k) + \hat{\boldsymbol{y}}^{\mathrm{traj}}(k) \\ & + \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\mathrm{traj}}(k)) \le \boldsymbol{y}^{\max}. \end{aligned}$$

The matrix $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda, \ldots, \lambda)$ is of dimensionality $N_{\mathrm{u}} \times N_{\mathrm{u}}$,

$$\boldsymbol{u}^{\min} = u^{\min}\boldsymbol{I}_{N_{\mathrm{u}},1}, \qquad \boldsymbol{y}^{\min} = y^{\min}\boldsymbol{I}_{N,1}, \tag{21}$$

$$\boldsymbol{u}^{\max} = u^{\max}\boldsymbol{I}_{N_{\mathrm{u}},1}, \qquad \boldsymbol{y}^{\max} = y^{\max}\boldsymbol{I}_{N,1}, \tag{22}$$

$$\triangle\boldsymbol{u}^{\max} = u^{\max}\boldsymbol{I}_{N_{\mathrm{u}},1}. \tag{23}$$

In the MPC-NPLTp optimisation problem (20) hard constraints imposed on the predicted controlled variable are considered for simplicity of presentation. In real applications, in order to guarantee feasibility, soft constraints must be implemented (Maciejowski, 2002; Tatjewski, 2007).

Having calculated the optimal vector $c^{\mathrm{opt}}(k)$ from the optimisation problem (20), the current optimal value of the manipulated variable is calculated from

$$\begin{aligned} u(k) = {} & \begin{bmatrix} l_1(0) & l_2(0) & \ldots & l_{n_{\mathrm{L}}}(0) \end{bmatrix} c^{\mathrm{opt}}(k) \\ & + u(k-1) \end{aligned} \tag{24}$$

and applied to the process.

The MPC-NPLT and MPC-NPLTp algorithms solve one quadratic optimisation problem at each sampling instant. The fundamental difference is the fact that in the first case as many as N_{u} decision variables are necessary while in the second case only $n_{\mathrm{L}} < N_{\mathrm{u}}$.

The MPC-NPLTp algorithm can be used for different types of input-output models. Minor modifications are necessary when state-space models must be used (Ławrynczuk, 2014).

5. Simulation results

In order to demonstrate advantages of parameterisation using Laguerre functions in computationally efficient nonlinear MPC, control of a Wiener system is considered since such systems have great practical significance (Greblicki, 2010; Janczak and Korbicz, 2019; Mzyk, 2014). Its dynamical part is described by the following discrete-time linear equation (van Donkelaar *et al*., 1999):

$$\begin{aligned} v(k) = {} & b_1u(k-1) + b_2u(k-2) + b_3u(k-3) \\ & + b_4u(k-4) - a_1v(k-1) - a_2v(k-2) \\ & - a_3v(k-3) - a_4v(k-4). \end{aligned} \tag{25}$$

The parameters are $a_1 = -3.0228$, $a_2 = 3.8630$, $a_3 = -2.6426$, $a_4 = 0.8084$, $b_1 = -1.4316$, $b_2 = 4.8180$, $b_3 = -5.3445$, $b_4 = 1.9641$. The dynamical part of the process is followed by a steady-state block

$$y(k) = -\exp\left(-v(k)\right) + 1. \tag{26}$$

The steady-state characteristic $y(u)$ of the whole Wiener process is depicted in Fig. 1.

The process considered has very complex dynamics. Figure 2 shows the step-response of its linear part (25). It

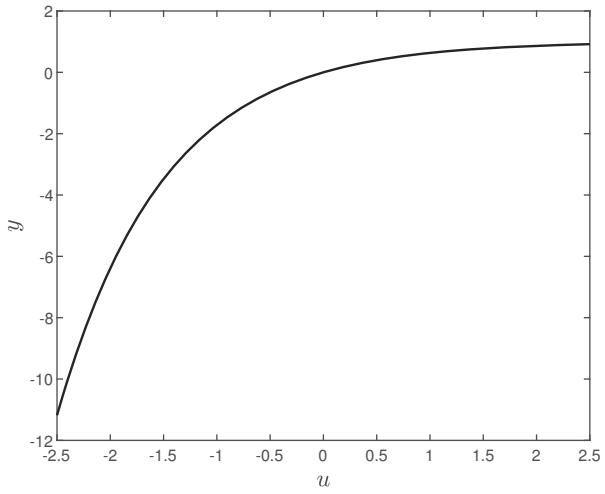

Fig. 1. Steady-state characteristic of the Wiener system.

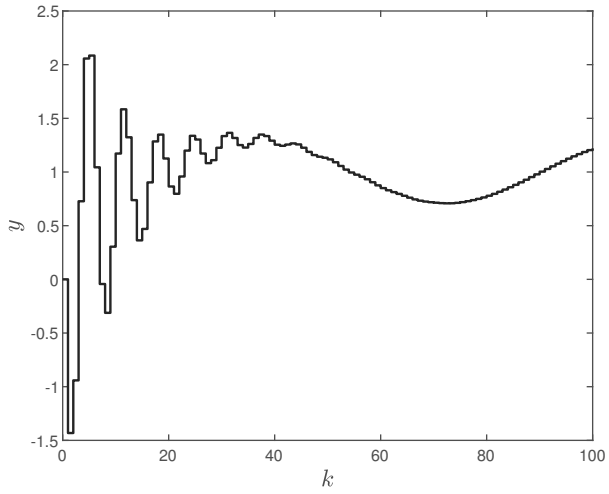

Fig. 2. Step-response of the linear part (25) of the Wiener system.

suggests that very long horizons should be used in MPC. In the work of Wang (2004), where only control of the linear process (25) (with a changed steady-state gain) is considered, the horizons are $N = N_{\mathrm{u}} = 100$. The default value of the parameter $\lambda = 0.25$.

The following MPC algorithms without parameterisation are compared:

1. The classical linear generalized predictive control (GPC) algorithm (Clarke *et al.*, 1987). For prediction, the linear part of the process (25) is used. The gain of the nonlinear steady-state block (26) is 1 for the operating point $u = y = v = 0$.

2. The MPC algorithm with nonlinear prediction and simplified linearisation (MPC-NPSL) (Ławryńczuk, 2014). Simplified model linearisation is carried out in the following way: for the current operating point of the process, the current gain of the nonlinear block (26) is calculated. Next, the linear dynamical block (25) is multiplied by such a time-varying gain. The obtained linearised model is used for prediction of the influence of the future (calculated) control policy. The full nonlinear model is used to find the influence of the past.

3. The MPC algorithm with nonlinear prediction and linearisation around the predicted trajectory (MPC-NPLPT) (Ławryńczuk, 2014). In this approach, if the process is close to the required set-point, the MPC-NPLT algorithm is executed once at each sampling instant. If the process is far from the set-point, a few (in this work maximally 5) executions of the MPC-NPLT strategy are performed in such a way that the calculated future trajectory of manipulated variables is used for linearisation at the next internal iteration.

4. The MPC algorithm with nonlinear optimisation (MPC-NO), in which the full nonlinear model is used for prediction without any simplifications.

The following MPC algorithms with parameterisation using Laguerre functions are compared:

1. The MPC-NPLT1p algorithm, in which the future trajectory of the manipulated variable used for linearisation, $\boldsymbol{u}^{\mathrm{traj}}(k)$, Eqn. (15), is defined by the value of the manipulated variable calculated and applied to the process at the previous sampling instant (Ławryńczuk, 2014).

2. The MPC-NPLT2p algorithm, in which the trajectory $\boldsymbol{u}^{\mathrm{traj}}(k)$ is defined by $N_{\mathrm{u}} - 1$ last elements of the optimal trajectory found at the previous sampling instant and not applied to the process (Ławryńczuk, 2014).

3. The MPC-NPLPTp algorithm with maximally 5 executions of the MPC-NPLTp strategy at each sampling instant.

4. The MPC-NOp algorithm with nonlinear optimisation.

The MPC-NO and MPC-NOp algorithms need solving on-line a nonlinear optimisation problem, while all other algorithms use quadratic optimisation tasks.

In all MPC algorithms a nonlinear model defined by Eqns. (25)–(26) is used, although in a different way.

Figure 3 presents simulation results for the classical linear GPC algorithm with different values of the parameter λ. Long prediction and control horizons are used, i.e., $N = N_{\mathrm{u}} = 100$. The GPC algorithm practically does not work since the process is nonlinear while its model used for prediction in GPC is linear.

Simulation results for the MPC-NPSL algorithm with the default value of the parameter $\lambda = 0.25$ are shown in Fig. 4; $N = N_{\mathrm{u}} = 100$. For the first, second and fourth step changes in the set-point quite good control is obtained, but for the third one the MPC-NPSL algorithm gives a very bad trajectory. The problem can be reduced by increasing the value of the weight to $\lambda = 0.5$; the obtained simulation results are depicted in Fig. 5. Unfortunately, the results are still unsatisfactory.

Because the classical GPC algorithm and the simple MPC-NPSL approach with model linearisation, despite using long horizons, give poor results, the MPC-NPLPT algorithm with advanced trajectory linearisation is considered. Figure 6 shows the obtained trajectories for three different values of the control horizon N_{u}, in all cases $N = 100$. It is evident that the MPC-NPLPT algorithm makes it possible to obtain good control quality, but very long horizons must be used ($N = N_{\mathrm{u}} = 100$).

Next, the classical MPC-NPLPT algorithm and the MPC-NPLPTp strategy with parameterisation are compared. In both cases the horizons are $N = N_{\mathrm{u}} = 100$. It is interesting to study the influence of the parameters a and n_{L} on control quality of the MPC-NPLPTp algorithm. To this end, the following performance index is used:

$$E_1 = \sum_{k=1}^{120} (y^{\mathrm{MPC\text{-}NPLPT}}(k) - y^{\mathrm{MPC\text{-}NPLPTp}}(k))^2. \quad (27)$$

It measures the sum of squared differences between the process output controlled by these two algorithms; the first one of those is treated as the reference. The obtained numerical values of the performance index E_1 are given in Table 1. The emphasised results refer to very good performance of the MPC-NPLPTp approach, i.e., when it gives practically the same trajectories as the MPC-NPLPT one. Since the objective of parameterisation is to reduce the number of decision variables in MPC, the value $a = 0.7$ is chosen because it makes it possible to obtain very good control quality for the lowest value of the parameter n_{L}. Table 1 suggests that for $a = 0.7$ it is sufficient to use only $n_{\mathrm{L}} = 20$ or $n_{\mathrm{L}} = 25$ decision variables. Figure 7 shows the process trajectories for $n_{\mathrm{L}} = 5, 10, 20$, $a = 0.7$. Of course, for $n_{\mathrm{L}} = 5, 10$ control quality is not satisfactory. The influence of the parameter a is depicted in Fig. 8. In this case $n_{\mathrm{L}} = 20$ and $a = 0.1, 0.7, 0.9$. Too small and too big values of the parameter a lead to bad control quality.

Table 2 compares the classical MPC algorithms against the MPC strategies with parameterisation in terms of two performance indices. The first one is the sum of squared differences between the required set point and the actual process output for the whole simulation horizon,

$$E_2 = \sum_{k=1}^{120} (y^{\mathrm{sp}}(k) - y(k))^2. \quad (28)$$

Table 1. Classical MPC-NPLPT algorithm vs. the MPC-NPLPTp strategy with parameterisation: the values of the performance index E_1 (Eqn. (27)); the highlighted results refer to very good performance of the MPC-NPLPTp approach.

n_{L} \ a	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
5	2.24×10^{1}	2.21×10^{1}	2.22×10^{1}	1.37×10^{1}	7.29	9.46	1.44×10^{1}	1.50×10^{1}	1.04×10^{1}
10	1.92×10^{1}	9.15	5.43	4.70	5.03	3.13	2.68	7.52	8.88
15	5.92	5.22	4.85	4.75	1.99	2.33×10^{-1}	3.08×10^{-1}	2.30	5.63
20	4.71	4.73	3.41	1.52	1.44×10^{-1}	8.08×10^{-2}	5.08×10^{-3}	7.25×10^{-1}	3.13
25	4.49	3.31	1.52	1.84×10^{-1}	1.04×10^{-1}	2.05×10^{-4}	7.63×10^{-6}	2.85×10^{-2}	1.90
30	3.42	1.76	3.10×10^{-1}	1.14×10^{-1}	6.22×10^{-3}	4.43×10^{-7}	1.54×10^{-8}	5.16×10^{-4}	1.16
35	2.18	5.79×10^{-1}	9.74×10^{-2}	4.39×10^{-2}	2.30×10^{-6}	8.46×10^{-9}	6.72×10^{-10}	6.79×10^{-6}	6.68×10^{-1}
40	1.03	1.28×10^{-1}	1.01×10^{-1}	1.07×10^{-3}	1.98×10^{-7}	6.03×10^{-10}	4.38×10^{-10}	1.44×10^{-7}	3.52×10^{-1}
45	3.41×10^{-1}	1.05×10^{-1}	2.78×10^{-2}	6.57×10^{-7}	1.20×10^{-9}	2.15×10^{-10}	1.27×10^{-10}	7.28×10^{-9}	1.02×10^{-1}
50	1.09×10^{-1}	9.48×10^{-2}	9.58×10^{-4}	1.16×10^{-7}	5.09×10^{-11}	1.94×10^{-11}	3.34×10^{-11}	5.64×10^{-9}	2.13×10^{-2}

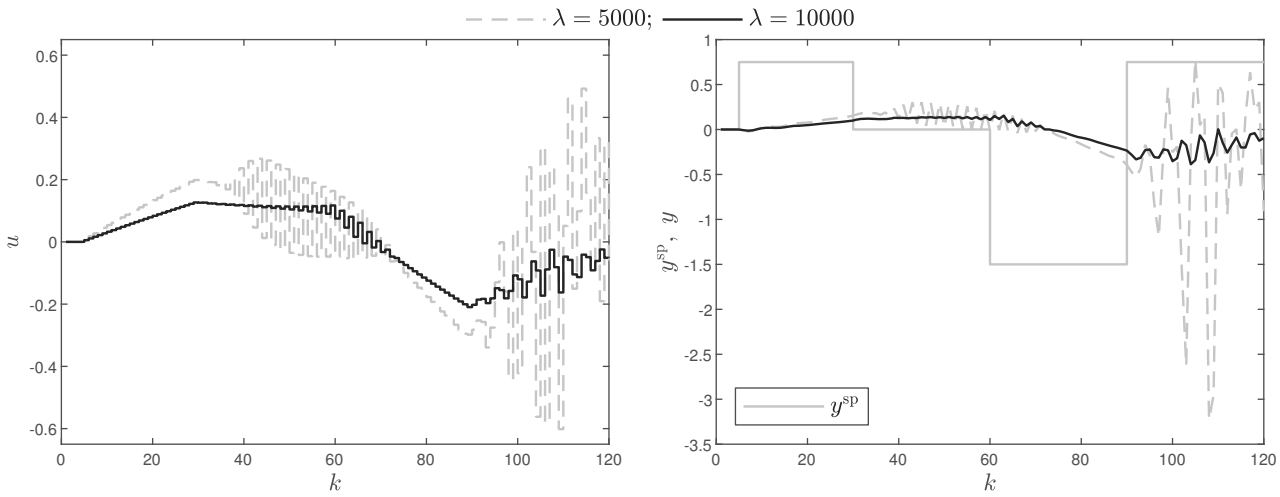

Fig. 3. Simulation results for the GPC algorithm with different values of the parameter λ, $N = N_{\mathrm{u}} = 100$.

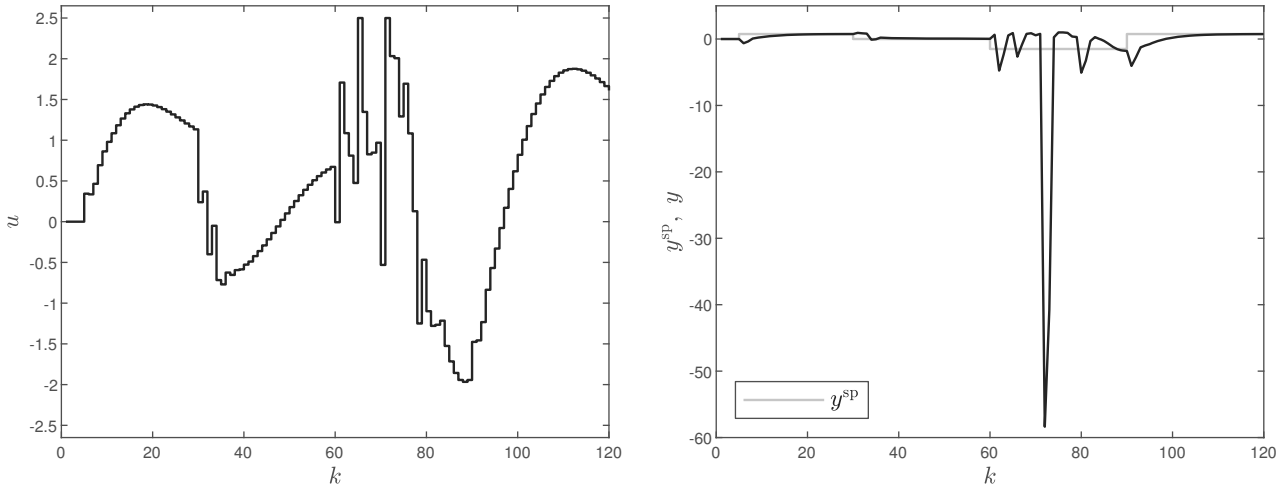

Fig. 4. Simulation results for the MPC-NPSL algorithm with $\lambda = 0.25$, $N = N_{\mathrm{u}} = 100$.

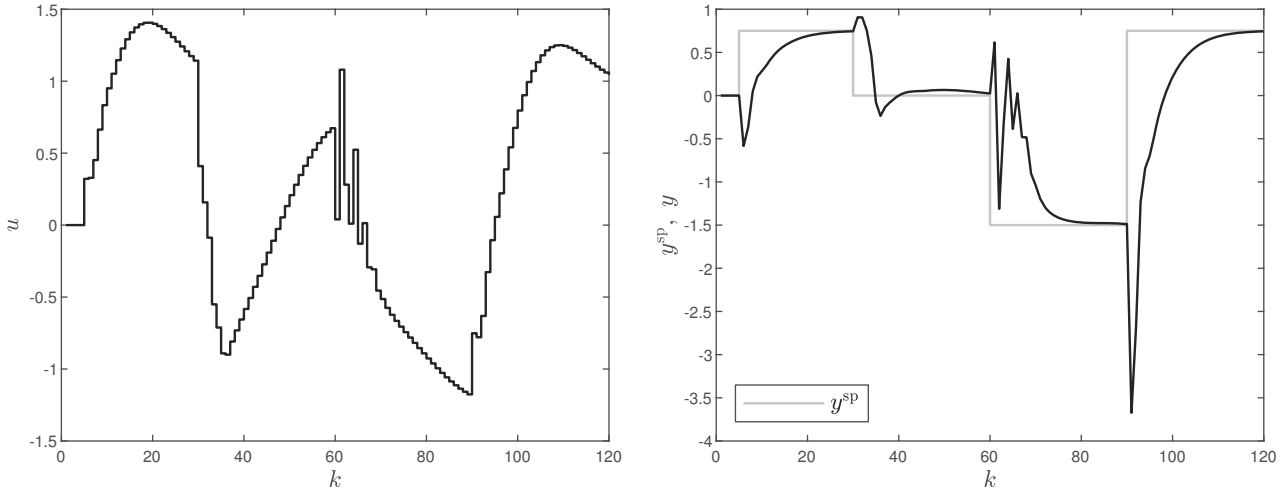

Fig. 5. Simulation results for the MPC-NPSL algorithm with $\lambda = 0.5$, $N = N_{\mathrm{u}} = 100$.

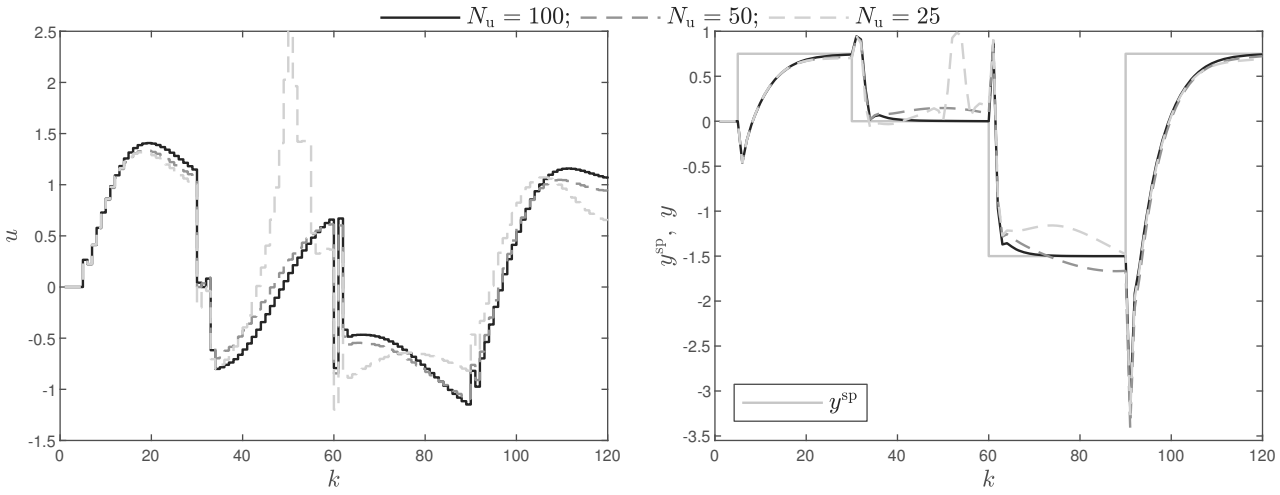

Fig. 6. Simulation results for the MPC-NPLPT algorithm with different values of the control horizons N_u, $N = 100$.

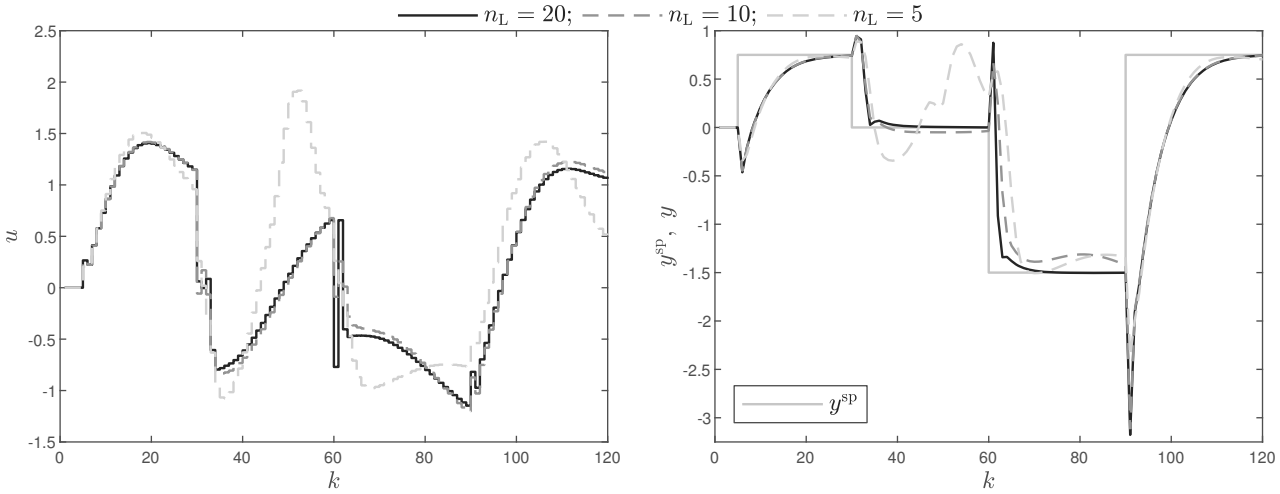

Fig. 7. Simulation results for the MPC-NPLPTp algorithm with parameterisation with different values of the parameter n_L, $a = 0.7$, $N = N_u = 100$.

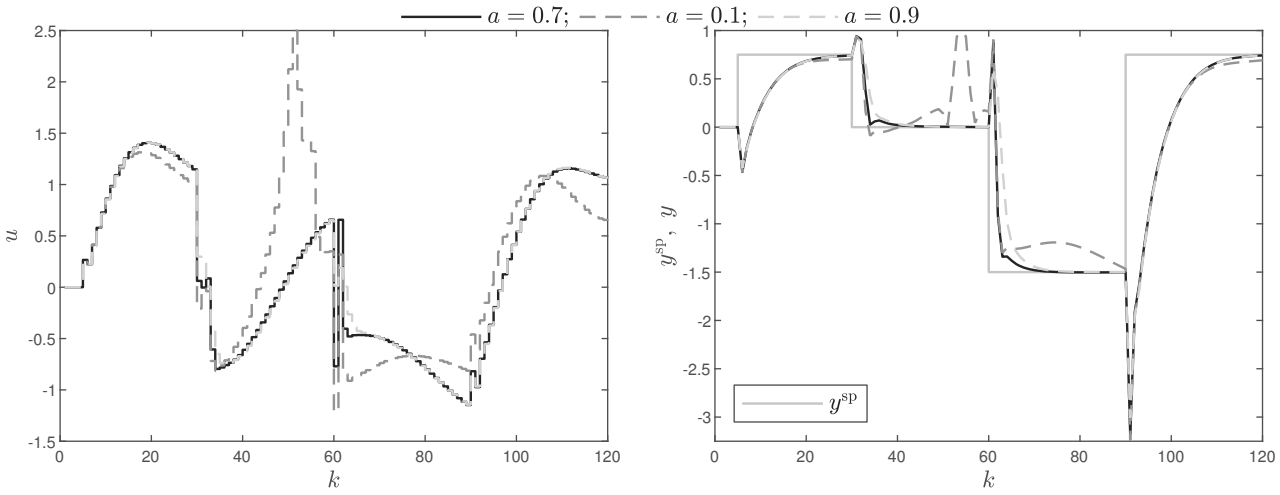

Fig. 8. Simulation results for the MPC-NPLPTp algorithm with parameterisation with different values of the parameter a, $n_L = 20$, $N = N_u = 100$.

Table 2. MPC strategies with parameterisation vs. the classical MPC algorithms: the values of the performance indices E_2 (Eqn. (28)) and E_3 (Eqn. (29)) and optimisation time; n_{var} is the number of decision variables.

Algorithm	Parameterisation	E_2	E_3	n_{var}	Optimisation type	Optimisation time
MPC-NPLT1p	Yes	7.9098×10^1	1.2012×10^1	$n_{\mathrm{L}} = 20$	Quadratic	50.18%
MPC-NPLT2p		6.7355×10^1	4.2176		Quadratic	49.28%
MPC-NPLPTp		6.1999×10^1	3.4170×10^{-1}		Quadratic	100.00%
MPC-NOp		6.1366×10^1	8.1167×10^{-3}		Nonlinear	1754.84%
MPC-NPLT1p		7.9945×10^1	1.3052×10^1	$n_{\mathrm{L}} = 25$	Quadratic	54.12%
MPC-NPLT2p		6.7340×10^1	4.2085		Quadratic	51.45%
MPC-NPLPTp		6.1905×10^1	3.2881×10^{-1}		Quadratic	102.18%
MPC-NOp		6.1283×10^1	1.1372×10^{-3}		Nonlinear	2762.48%
MPC-NPLPT	No	6.1900×10^1	3.2856×10^{-1}	$N_{\mathrm{u}} = 100$	Quadratic	132.26%
MPC-NO		6.1269×10^1	–		Nonlinear	15734.30%

The second one measures the sum of squared differences between the process output controlled by the "ideal" MPC-NO algorithm without parameterisation and the consecutive compared algorithms,

$$E_3 = \sum_{k=1}^{120} (y^{\mathrm{MPC\text{-}NO}}(k) - y(k))^2. \tag{29}$$

The performance index (29) is similar to the one given by Eqn. (27), but now the MPC-NO algorithm is treated as the reference. Additionally, Table 2 specifies the number of optimised variables n_{var}, the optimisation type (nonlinear or quadratic) and relative optimisation time (MATLAB). The following observations may be made:

1. The MPC-NPLT1p and MPC-NPLT2p algorithms with single trajectory linearisation and optimisation at every sampling instant are the worst ones (the performance indices E_2 and E_3 are the biggest). Multiple linearisation and optimisation possible in the MPC-NPLPTp and MPC-NPLPT algorithms give much better results. Figure 9 compares trajectories obtained when the MPC-NPLPTp, MPC-NPLT1p and MPC-NPLT2p algorithms with parameterisation are used, in all cases $a = 0.7$, $n_{\mathrm{L}} = 20$, $N = N_{\mathrm{u}} = 100$. It is clear that for the benchmark process considered only one trajectory linearisation and quadratic optimisation at every sampling instant is not satisfactory, i.e., the resulting trajectories are different from those obtained when the MPC-NPLPTp algorithm is used, in which maximally 5 repetitions of linearisation and optimisation at every sampling instant are allowed.

2. The MPC-NPLPTp algorithm with quadratic optimisation gives very similar results as MPC-NOp with nonlinear optimisation. Figure 10 compares trajectories of these algorithms with parameterisation, $a = 0.7$, $n_{\mathrm{L}} = 20$.

3. The MPC-NPLPTp algorithm with parameterisation (the optimisation problem has only $n_{\mathrm{L}} = 20$ or $n_{\mathrm{L}} = 25$ decision variables) gives practically the same results as the classical MPC-NPLPT algorithm without parameterisation (the optimisation problem has as many as $N_{\mathrm{u}} = 100$ decision variables). Figure 10 compares trajectories of the MPC-NPLPTp and MPC-NPLPT algorithms. The results for $n_{\mathrm{L}} = 25$ are even better.

4. Taking into account the MPC-NPLPTp algorithm (and also the MPC-NOp one), increasing the number of decision variables (n_{L}) leads to better results. This is also clear from Table 1.

5. The MPC-NPLPTp algorithm with parameterisation and quadratic optimisation gives very similar results as the computationally demanding, best possible MPC-NO algorithm with nonlinear optimisation and $N_{\mathrm{u}} = 100$ decision variables.

6. As far as the optimisation time is concerned, the comparison between the MPC-NPLPTp algorithm with parameterisation and the classical MPC-NPLPT algorithm, with as many as $N_{\mathrm{u}} = 100$ decision variables, is the most important. When $n_{\mathrm{L}} = 20$, the classical MPC-NPLPT algorithm is characterised by over 30% longer optimisation time. Secondly, the MPC-NPLT1p and MPC-NPLT2p algorithms with only one on-line linearisation and optimisation at every sampling instant are characterised by an approximately 50% shorter optimisation time than the MPC-NPLPTp strategy with multiple linearisation and optimisation. Thirdly, the MPC-NOp and MPC-NO algorithms are very computationally demanding, but parametrisation makes it possible to significantly reduce the optimisation time.

Finally, the performance of all the discussed nonlinear MPC algorithms is compared when the process

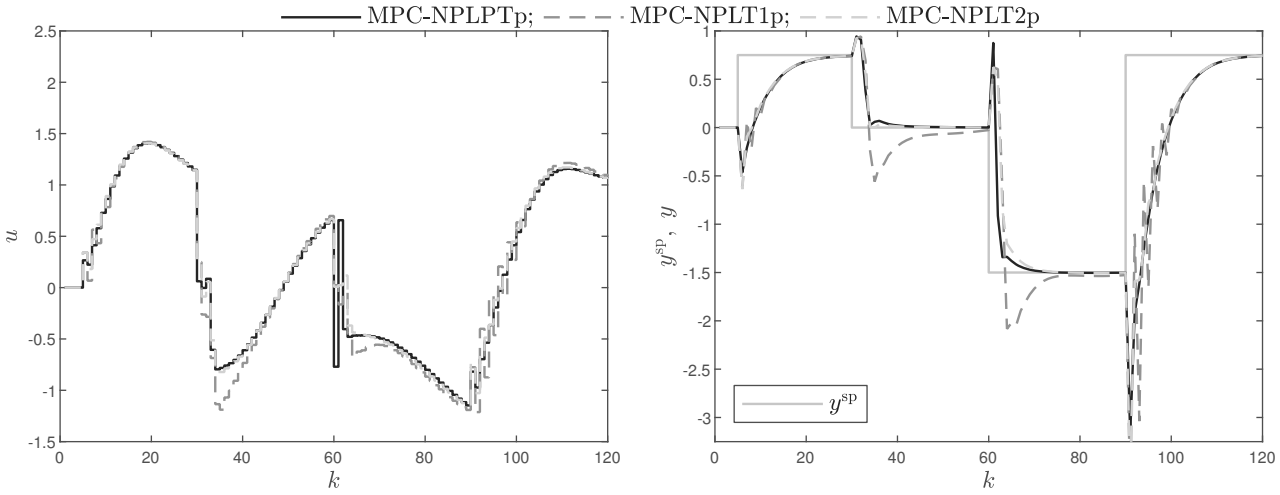

Fig. 9. Simulation results for the MPC-NPLPTp, MPC-NPLT1p and MPC-NPLT2p algorithms with parameterisation, $a = 0.7$, $n_{\mathrm{L}} = 20$, $N = N_{\mathrm{u}} = 100$.

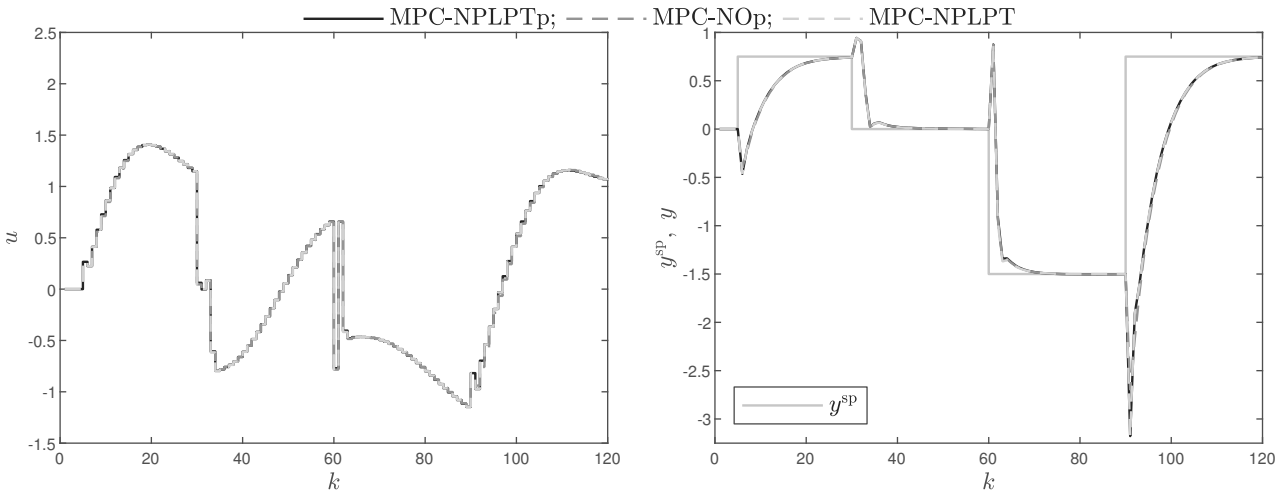

Fig. 10. Simulation results for the MPC-NPLPTp and MPC-NOp algorithm with parameterisation, $a = 0.7$, $n_{\mathrm{L}} = 20$, vs. the MPC-NPLPT algorithm, $N = N_{\mathrm{u}} = 100$.

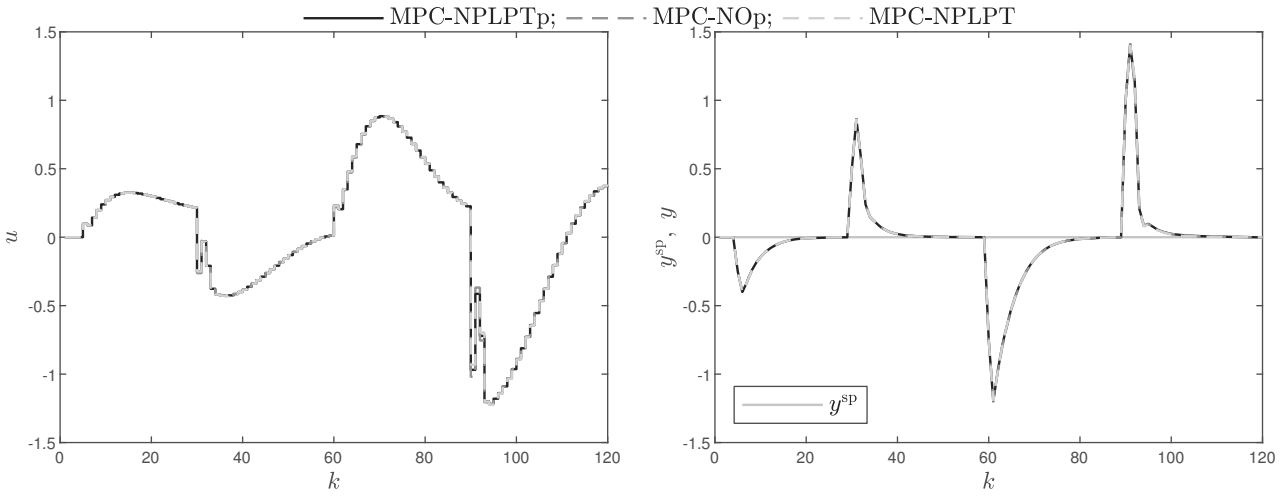

Fig. 11. Simulation results for the MPC-NPLPTp and MPC-NOp algorithms with parameterisation when the process is affected by disturbances, $a = 0.7$, $n_{\mathrm{L}} = 20$, vs. the MPC-NPLPT algorithm, $N = N_{\mathrm{u}} = 100$.

Table 3. MPC strategies with parameterisation vs. the classical MPC algorithms when the process is affected by disturbances: the values of the performance indices E_2 (Eqn. (28)) and E_3 (Eqn. (29)) and optimisation time; n_{var} is the number of decision variables.

Algorithm	Parameterisation	E_2	E_3	n_{var}	Optimisation type	Optimisation time
MPC-NPLT1p	Yes	1.3052×10^1	1.8343	$n_{\mathrm{L}} = 20$	Quadratic	55.99%
MPC-NPLT2p		1.1536×10^1	2.3643×10^{-1}		Quadratic	54.23%
MPC-NPLPTp		1.0951×10^1	2.7907×10^{-3}		Quadratic	100.00%
MPC-NOp		1.0877×10^1	1.6367×10^{-3}		Nonlinear	2368.32%
MPC-NPLT1p		1.3791×10^1	2.6299	$n_{\mathrm{L}} = 25$	Quadratic	59.03%
MPC-NPLT2p		1.1533×10^1	2.3665×10^{-1}		Quadratic	58.03%
MPC-NPLPTp		1.0990×10^1	4.0057×10^{-3}		Quadratic	103.53%
MPC-NOp		1.0910×10^1	8.4355×10^{-4}		Nonlinear	3488.62%
MPC-NPLPT	No	1.0994×10^1	4.2286×10^{-3}	$N_{\mathrm{u}} = 100$	Quadratic	132.16%
MPC-NO		1.0917×10^1	–		Nonlinear	21471.16%

is affected by unmeasured additive output disturbances. Four disturbance steps are considered: the first step, -0.25, starts at the sampling instant $k = 5$, the second step, 0.25, starts at $k = 30$, the third step, -0.5, starts at $k = 60$, the fourth step, 0.5, starts at $k = 90$, the set-point is zero for the whole simulation horizon. The algorithms are compared in terms of the indices E_2 and E_3 as well as the optimisation time in Table 3. The observations made for the set-point tracking case hold true. In particular, the MPC-NPLPTp algorithm with parameterisation gives practically the same results as the classical MPC-NPLPT algorithm without parameterisation. Moreover, the trajectories of the MPC-NPLPTp algorithm are very similar to those of the MPC-NOp and MPC-NO schemes with nonlinear optimisation. In terms of the optimisation time, the classical MPC-NPLPT algorithm is slower by more than 30% than the MPC-NPLPTp algorithm. Figure 10 compares trajectories of the MPC-NPLPTp and MPC-NOp algorithms with parameterisation, $a = 0.7$, $n_{\mathrm{L}} = 20$, vs. the classical MPC-NPLPT scheme.

6. Conclusions

This work discussed a nonlinear MPC approach to processes with complex dynamics which need long horizons and numerous optimised decision variables calculated at every sampling instant. The presented solution uses two concepts: advanced on-line linearisation of the predicted trajectory of the controlled variable and parameterisation using Laguerre functions. As a result, the MPC-NPLPTp algorithm needs only on-line quadratic optimisation and the number of decision variables is significantly lower than the control horizon. For a benchmark system the MPC-NPLPTp algorithm with only 20 decision variables gives very good quality of control. In particular, the obtained trajectories are practically the same as those possible in the MPC-NPLPT algorithm with as many as 100 decision variables. Furthermore, it is very important that the MPC-NPLPTp algorithm gives results comparable with those possible in the "ideal" MPC-NO scheme with nonlinear optimisation, but is very computationally efficient.

References

Bemporad, A., Morari, M., Dua, V. and Pistikopoulos, E. (2002). The explicit linear quadratic regulator for constrained systems, *Automatica* **38**(1): 3–20.

Bosschaerts, W., Van Renterghem, T., Hasan, O.A. and Limam, K. (2017). Development of a model based predictive control system for heating buildings, *Energy Procedia* **122**(1): 519–528.

Chaber, P. and Ławryńczuk, M. (2019). Fast analytical model predictive controllers and their implementation for STM32 ARM microcontroller, *IEEE Transactions on Industrial Informatics* **15**(8): 4580–45900.

Clarke, D.W., Mohtadi, C. and Tuffs, P.S. (1987). Generalized predictive control. Part I: The basic algorithm, *Automatica* **23**(2): 137–148.

Falconí, G.P., Angelov, J. and Holzapfel, F. (2018). Adaptive fault-tolerant position control of a hexacopter subject to an unknown motor failure, *International Journal of Applied Mathematics and Computer Science* **28**(2): 309–321, DOI: 10.2478/amcs-2018-0022.

Greblicki, W. (2010). Nonparametric input density-free estimation of the nonlinearity in Wiener systems, *IEEE Transactions on Information Theory* **56**(7): 3575–3580.

Gutiérrez-Urquídez, R.C., Valencia-Palomo, G., Rodríguez-Elias, O.M. and Trujillo, L. (2015). Systematic selection of tuning parameters for efficient predictive controllers using a multiobjective evolutionary algorithm, *Applied Soft Computing* **31**(6): 326–338.

Harrabi, N., Kharrat, M., Aitouche, A. and Souissi, M. (2018). Control strategies for the grid side converter in a wind generation system based on a fuzzy approach, *International Journal of Applied Mathematics and Computer Science* **28**(2): 323–333, DOI: 10.2478/amcs-2018-0023.

Jama, M., Wahyudie, A. and Noura, H. (2018). Robust predictive control for heaving wave energy converters, *Control Engineering Practice* **77**(1): 138–149.

Janczak, A. and Korbicz, J. (2019). Two-stage instrumental variables identification of polynomial Wiener systems with invertible nonlinearities, *International Journal of Applied Mathematics and Computer Science* **29**(3): 571–580, DOI: 10.2478/amcs-2019-0042.

Karimi Pour, F., Puig, V. and Ocampo-Martinez, C. (2018). Multi-layer health-aware economic predictive control of a pasteurization pilot plant, *International Journal of Applied Mathematics and Computer Science* **28**(1): 97–110, DOI: 10.2478/amcs-2018-0007.

Khan, B. and Rossiter, J. A. (2013). Alternative parameterisation within predictive control: A systematic selection, *International Journal of Control* **86**(8): 1397–1409.

Kim, J., Jung, Y. and Bang, H. (2018). Linear time-varying model predictive control of magnetically actuated satellites in elliptic orbits, *Acta Astronautica* **151**(1): 791 – 804.

Lasheen, A., Saad, M.S., Emara, H.M. and Elshafei, A.L. (2017). Continuous-time tube-based explicit model predictive control for collective pitching of wind turbine, *Energy* **118**(1): 1222–1233.

Li, Y., Wang, H. and Meng, X. (2019). Almost periodic synchronization of fuzzy celluar neural networks with time-varying delays via state-feedback and impulsive control, *International Journal of Applied Mathematics and Computer Science* **29**(2): 337–349, DOI: 10.2478/amcs-2019-0025.

Ligthart, J.A.J., Poksawat, P., Wang, L. and Nijmeijer, H. (2017). Experimentally validated model predictive controller for a hexacopter, *IFAC-PapersOnLine* **50**(1): 4076–4081.

Ławryńczuk, M. (2014). *Computationally Efficient Model Predictive Control Algorithms: A Neural Network Approach*, Springer, Cham.

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

Mzyk, G. (2014). *Combined Parametric-Nonparametric Identification of Block-Oriented Systems,* Lecture Notes in Control and Information Sciences, Vol. 454, Springer Verlag, Berlin.

Oliveira, G.H.C., da Rosa, A., Campello, R.J.G.B., Machado, J.B. and Amaral, W.C. (2011). An introduction to models based on Laguerre, Kautz and other related orthonormal functions. Part I: Linear and uncertain models, *International Journal of Modelling, Identification and Control* **14**(1/2): 121–132.

Oliveira, G.H.C., da Rosa, A., Campello, R.J.G.B., Machado, J.B. and Amaral, W.C. (2012). An introduction to models based on Laguerre, Kautz and other related orthonormal functions. Part II: Non-linear models, *International Journal of Modelling, Identification and Control* **16**(1): 1–14.

Pazera, M., Buciakowski, M. and Witczak, M. (2018). Robust multiple sensor fault-tolerant control for dynamic non-linear systems: Application to the aerodynamical twin-rotor system, *International Journal of Applied Mathematics and Computer Science* **28**(2): 297–308, DOI: 10.2478/amcs-2018-0021.

Richalet, J. and O'Donovan, D. (2009). *Predictive Functional Control: Principles and Industrial Applications*, Springer, London.

Takács, G., Batista, G., Gulan, M. and Rohal'-Ilkiv, B. (2016). Embedded explicit model predictive vibration control, *Mechatronics* **36**(1): 54–62.

Tatjewski, P. (2007). *Advanced Control of Industrial Processes, Structures and Algorithms*, Springer, London.

van Donkelaar, E.T., Bosgra, O.H. and Van den Hof, P.M.J. (1999). Model predictive control with generalized input parametrization, *Proceedings of the European Control Conference, ECC 1999, Karlsruhe, Germany*, pp. 443–454, paper F0599.

Wahlberg, B. (1991). System identification using Laguerre models, *IEEE Transactions on Automatic Control* **36**(5): 551–562.

Wang, L. (2001). Continuous time model predictive control design using orthonormal functions, *International Journal of Control* **74**(16): 1588–1600.

Wang, L. (2004). Discrete model predictive controller design using Laguerre functions, *Journal of Process Control* **14**(2): 131–142.

Wang, Y. and Boyd, S. (2010). Fast model predictive control using online optimization, *IEEE Transactions on Control Systems Technology* **18**(2): 267–278.

Witkowska, A. and Śmierzchalski, R. (2018). Adaptive backstepping tracking control for an over-actuated DP marine vessel with inertia uncertainties, *International Journal of Applied Mathematics and Computer Science* **28**(4): 679–693, DOI: 10.2478/amcs-2018-0052.

Zheng, Y., Zhou, J., Xu, Y., Zhang, Y. and Qian, Z. (2017). A distributed model predictive control based load frequency control scheme for multi-area interconnected power system using discrete-time Laguerre functions, *ISA Transactions* **68**(1): 127–140.

Maciej Ławryńczuk was born in Warsaw, Poland, in 1972. He obtained his MSc in 1998, PhD in 2003, and DSc in 2013, all in automatic control from the Warsaw University of Technology, Faculty of Electronics and Information Technology. Currently he is employed as an associate professor at the same university at the Institute of Control and Computation Engineering. He is the author or a co-author of 6 books and more than 100 other publications, including over 30 journal articles. His research interests include advanced control algorithms, in particular model predictive control (MPC) algorithms, set-point optimization algorithms, soft computing methods, especially neural networks, modelling and simulation.

Received: 14 June 2019
Revised: 27 September 2019
Accepted: 18 October 2019