Information Sciences 616 (2022) 229–254

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Advanced predictive control for GRU and LSTM networks

Krzysztof Zarzycki [a], Maciej Ławryńczuk [a,*]

[a] Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland

ARTICLE INFO

Article history:
Received 20 May 2022
Received in revised form 18 August 2022
Accepted 15 October 2022
Available online 22 October 2022

Keywords:
LSTM network
GRU network
Model Predictive Control

ABSTRACT

This article is concerned with Model Predictive Control (MPC) algorithms that use Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural networks for prediction. For two benchmark processes, it is shown that the typical approach to MPC that hinges on successively linearized LSTM or GRU models do not give precise predictions and satisfactory control quality. The presented MPC control schemes utilize online advanced trajectory linearization, which yields simple quadratic optimization programs. It is shown that the discussed approaches give excellent prediction accuracy and control quality, very similar to that possible in MPC with full nonlinear prediction and nonlinear optimization done online. It is also demonstrated that the described MPC algorithms are a few times faster than the MPC method with nonlinear optimization. Moreover, the performance of MPC based on LSTM and GRU networks is compared, and simpler GRU networks are recommended.

1. Introduction

Model Predictive Control (MPC) is an advanced control technique that has been widely successful in practice [34]. This has become possible due to two advantages of MPC over the classical control techniques. Firstly, MPC provides very good quality of control, especially for multidimensional processes with multiple input and output variables. Secondly, MPC has the unique ability to take into account the limitations of both the process control signals and the predicted output signals. Many examples of the practical use of MPC can be found. Applications include complex processes, such as iron sintering process [4], waste-water treatment systems [16], oil fractionator process [26], robotic manipulators [31], power plants [38] and extraction processes [44]. In recent years, MPC algorithms are used not only in industry but also in embedded systems, e.g., to regulate glucose level [10], in autonomous vehicles [27], in robots [43], in hypersonic vehicles [33] and in networked systems [45].

Neural network-based MPC algorithms deserve special attention due to the essential advantages of such models, i.e., great accuracy and straightforward structure. Simple Multilayer Perceptrons (MLP) [23] or Radial Basis Function [1] (RFB) are often used. Because in MPC the scenario of future predictions of the process output variable for a chosen time horizon is calculated at each sampling instant online, recurrent neural networks are particularly suited for MPC. The Short Term Memory (LSTM) networks [17] are more and more popular. They combine the advantages of recurrent neural networks [2,15,28,30] while reducing the occurrence of the phenomenon of vanishing gradient. LSTM networks may be utilized to perform numerous tasks, e.g. speech synthesis [5], detection of deepfake videos [6], industrial product quality estimation [9], semantic relation extraction [11], handwriting recognition [13], speech recognition [14], forecasting stock prices [20], signals' classification

* Corresponding author.
E-mail addresses: Krzysztof.Zarzycki@pw.edu.pl (K. Zarzycki), Maciej.Lawrynczuk@pw.edu.pl (M. Ławryńczuk).

https://doi.org/10.1016/j.ins.2022.10.078
0020-0255/

[37] and recognition of personality [49]. Furthermore, the LSTM networks have been utilized to approximate dynamical systems, e.g. chemical neutralization (pH) reactors [32,35], reverse osmosis plants [21], cooling systems [36], temperature control changes [19], pharmaceutical manufacturing systems [39] and self-driving vehicles [18]. Another noteworthy model is the Gated Recurrent Unit (GRU) [7], a modification of the standard LSTM architecture. Compared to LSTM, it has fewer parameters, enabling faster computation time. Interestingly, the use of the GRU network to model dynamical processes is less common in the literature; the number of publications concerned with applications of GRU networks is significantly lower compared with LSTM. Examples of considered processes are: neutralization reactors [3,47], polymerization reactors [47], tandem-wing quad-plane drones [29] and paper machines [22]. In addition to LSTM and GRU models, ensembles of recurrent neural networks can also be considered in Lyapunov-based MPC as discussed in [41,42]. As a result of using a Lyapunov function in MPC formulation, closed-loop state boundedness and convergence to the origin are guaranteed. An interesting application of the Lyapunov-based MPC combined with recurrent neural networks is discussed in [40]. The use of recurrent neural networks to modeling and MPC of a batch crystallization process is presented in [48].

1.1. Motivations

Literature provides reports of direct utilization of LSTM neural networks for prediction in MPC [19,21,35,47,39]. Similarly, GRU models may also be used in this manner [3,22,47]. Such a methodology is hereafter called MPC with Nonlinear Optimization (MPC-NO). That MPC policy uses an accurate nonlinear dynamical process model without any truncation. It has an undoubted advantage because the model accurately represents the actual behavior of the process, resulting in accurate predictions generated by the algorithm and good quality control. However, in this approach, one must use nonlinear optimization algorithms to solve the MPC optimization task online. The main disadvantage of this method is the significant computational effort required to get the solution to the nonlinear optimization task.

To simplify and speed up the calculations, the MPC scheme with online linearization of the LSTM model can be used [32,36]. In this approach, the cyclic linearization of the model at the current point of process operation is performed. The obtained time-varying linear counterpart of the nonlinear LSTM network is then applied to determine predictions. This approach transforms the nonlinear optimization task into a simple quadratic optimization task, the solution of which is much simpler in comparison with the nonlinear one. However, simple LSTM (or GRU) model linearization at the process operating point turns out to have an essential weak point. Calculation of the linearized time-varying model proves to be cumbersome in the case of LSTM and GRU networks. Due to their recurrent nature, the network's output depends on an array of the past values of the process input signal, as shown in [46]. Hence, the linearized model and its predictions are relatively inaccurate, which is likely to lead to not satisfying control quality. Therefore, our motivation for conducting the research reported in this work is to propose an MPC controller relying on the LSTM and GRU networks, which combines the advantages of both approaches described above: accuracy of prediction, good control quality and high computation speed.

1.2. Objective and Contribution

This work presents a computationally efficient approach to MPC using LSTM and GRU models, relying only on quadratic optimization but providing very good control quality. To this end, an advanced online trajectory linearization is utilized. The usefulness and efficacy of the discussed algorithms are shown for two simulated benchmark processes; comparisons with the classical MPC algorithms based on nonlinear optimization and simple online model linearization are given. The calculation times of the described MPC algorithms are evaluated compared to that of the classical MPC scheme relying on nonlinear optimization. In addition, we discuss the issues related to the choice of the model order of dynamics and possible cases when deterioration of control quality is possible. It is also essential to determine how MPC based on the less popular GRU network performs compared to the LSTM case. In theory, GRUs networks have a similar structure and operating principle as the LSTMs. However, they need fewer internal weights, so they should give a similar quality of control while having a lower computational cost.

1.3. Organization

The layout of this work is the following. Section 2 defines the objective MPC. Section 3 describes the structure of LSTM and GRU models. Section 4 briefly recalls the classical MPC controllers based on nonlinear optimization and the MPC employing the successive model linearization. Section 5 presents the operating principle of the MPC algorithms with advanced trajectory linearizations in detail. Section 6 presents and thoroughly discusses the simulation results. The controllers are tested using two benchmarks: a pH reactor and a polymerization reactor. Section 7 summarizes the most important findings of this work.

2. MPC Objective

Let u be the input of the process and y be the output. Alternatively, the terms the manipulated variable and the controlled variable are used. At each discrete-time instant $k, k = 0, 1, 2, \ldots$, the MPC controller computes the vector of decision variables online. This vector is defined as the following increments of the input variable [34]

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

where N_{u} is the control horizon. The basic MPC optimization problem is given by

$$\begin{aligned} &\underset{\triangle \boldsymbol{u}(k)}{\text{minimize}} \left\{ \sum_{i=1}^{N} (y^{\mathrm{sp}}(k+i|k) - \hat{y}(k+i|k))^2 + \lambda \sum_{i=0}^{N_{\mathrm{u}}-1} (\triangle u(k+i|k))^2 \right\} \\ &\text{subject to} \\ &\underline{u} \leqslant u(k+i|k) \leqslant \overline{u},\ i = 0, \ldots, N_{\mathrm{u}} - 1 \\ &\triangle \underline{u} \leqslant \triangle u(k+i|k) \leqslant \triangle \overline{u},\ i = 0, \ldots, N_{\mathrm{u}} - 1 \\ &\underline{y} \leqslant \hat{y}(k+i|k) \leqslant \overline{y},\ i = 1, \ldots, N. \end{aligned} \tag{2}$$

The cost function has two components. The first computes the future (predicted) control error of the controlled process output for the prediction horizon N. Usually, the squared sum of the differences between the set point $y^{\mathrm{sp}}(k+i|k)$ and the predictions $\hat{y}(k+i|k)$ are utilized. The second part of the cost function computes the squared sum of the increments of the input signal. The coefficient λ can be used to penalize rapid changes in the control signal.

Solving the optimization task (2) allows to determine the MPC decision vector (1). However, only its first element is sent to the process at the time step k. The same procedure is then repeated at the subsequent discrete sampling steps.

Unlike simple controllers used in practice, e.g., the PID controllers, in MPC, the dynamical model of the controlled process is not used during controller development carried out offline but is directly used online. Specifically, the controlled process output predictions over the prediction horizon are computed from the process model at each sampling time k.

3. LSTM and GRU Models

In this work, we consider the general discrete-time dynamical model

$$y(k) = F(\boldsymbol{x}(k)) \tag{3}$$

The model arguments are established by the vector

$$\boldsymbol{x}(k) = [u(k-1)\ u(k-2) \ldots u(k-n_{\mathrm{B}})\ y(k-1)\ y(k-2) \ldots y(k-n_{\mathrm{A}})]^{\mathrm{T}} \tag{4}$$

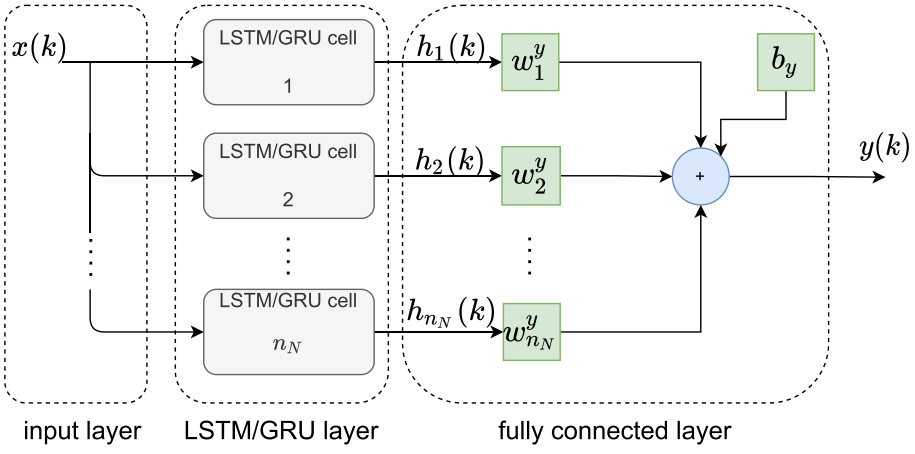

Fig. 1. The LSTM/GRU network topology.

The general structure of LSTM and GRU neural networks is presented in Fig. 1. The networks consist of three layers. The first one, i.e., the input layer supplies the model input vector $\boldsymbol{x}(k)$ to the next layer. Input signals enter the second layer, which contains n_{N} neurons of LSTM or GRU type, often referred to as cells. Ultimately, the output signals from the cells, the hidden state h enter the fully connected layer of the network. The output signal of the entire network is calculated as the sum of the hidden states weighted by some coefficients w_i^{y}. The network's output can be computed as

$$y(k) = \sum_{i=1}^{n_{\mathrm{N}}} w_i^{\mathrm{y}} h_i(k) + b_{\mathrm{y}} \tag{5}$$

where n_{N} is the number of cells and b_{y} is a bias.

3.1. LSTM Model

Four gates comprise the LSTM cell displayed in Fig. 2 [17]. The forget gate f selects values from the cell state that are discarded. The input gate i selects which new information from the input vector should be added to the cell state. Together with the cell candidate gate g, the gate i is used to compute the current cell state $c(k)$. The output gate o generates the current value of the hidden state h. The symbol $\sigma(x) = \frac{1}{1+e^{-x}}$ denotes the sigmoid function and the symbol $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ denotes the hyperbolic tangent function.

The $n_{\mathrm{N}} \times 1$ vectors $\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{i}, \boldsymbol{o}$ are utilized to represent the LSTM cells. Some additional weights are also necessary. Let $\boldsymbol{W}$ be the weights associated with the input signals $\boldsymbol{x}$; $\boldsymbol{R}$ denotes the recursive weights, associated with the hidden state of the cell from the previous moment $\boldsymbol{h}(k-1)$; $\boldsymbol{b}$ is the constant (bias) components. All weights can be gathered as the following matrices

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{W}_{\mathrm{i}} \\ \boldsymbol{W}_{\mathrm{f}} \\ \boldsymbol{W}_{\mathrm{g}} \\ \boldsymbol{W}_{\mathrm{o}} \end{bmatrix}, \quad \boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_{\mathrm{i}} \\ \boldsymbol{R}_{\mathrm{f}} \\ \boldsymbol{R}_{\mathrm{g}} \\ \boldsymbol{R}_{\mathrm{o}} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_{\mathrm{i}} \\ \boldsymbol{b}_{\mathrm{f}} \\ \boldsymbol{b}_{\mathrm{g}} \\ \boldsymbol{b}_{\mathrm{o}} \end{bmatrix} \tag{6}$$

The $n_{\mathrm{N}} \times (n_{\mathrm{A}} + n_{\mathrm{B}})$ matrices are denoted by $\boldsymbol{W}_{\mathrm{i}}, \boldsymbol{W}_{\mathrm{f}}, \boldsymbol{W}_{\mathrm{g}}$ and $\boldsymbol{W}_{\mathrm{o}}$; the $n_{\mathrm{N}} \times n_{\mathrm{N}}$ matrices are denoted by $\boldsymbol{R}_{\mathrm{i}}, \boldsymbol{R}_{\mathrm{f}}, \boldsymbol{R}_{\mathrm{g}}$ and $\boldsymbol{R}_{\mathrm{o}}$; the $n_{\mathrm{N}} \times 1$ vectors are named $\boldsymbol{b}_{\mathrm{i}}, \boldsymbol{b}_{\mathrm{f}}, \boldsymbol{b}_{\mathrm{g}}$ and $\boldsymbol{b}_{\mathrm{o}}$, respectively.

At the discrete time instant k, the network first calculates the outputs of each gate

$$\boldsymbol{i}(k) = \sigma(\boldsymbol{W}_{\mathrm{i}}\boldsymbol{x}(k) + \boldsymbol{R}_{\mathrm{i}}\boldsymbol{h}(k-1) + \boldsymbol{b}_{\mathrm{i}}) \tag{7}$$

$$\boldsymbol{f}(k) = \sigma(\boldsymbol{W}_{\mathrm{f}}\boldsymbol{x}(k) + \boldsymbol{R}_{\mathrm{f}}\boldsymbol{h}(k-1) + \boldsymbol{b}_{\mathrm{f}}) \tag{8}$$

$$\boldsymbol{g}(k) = \tanh\left(\boldsymbol{W}_{\mathrm{g}}\boldsymbol{x}(k) + \boldsymbol{R}_{\mathrm{g}}\boldsymbol{h}(k-1) + \boldsymbol{b}_{\mathrm{g}}\right) \tag{9}$$

$$\boldsymbol{o}(k) = \sigma(\boldsymbol{W}_{\mathrm{o}}\boldsymbol{x}(k) + \boldsymbol{R}_{\mathrm{o}}\boldsymbol{h}(k-1) + \boldsymbol{b}_{\mathrm{o}}) \tag{10}$$

Then, the cell state of the network can be computed

$$\boldsymbol{c}(k) = \boldsymbol{f}(k) \circ \boldsymbol{c}(k-1) + \boldsymbol{i}(k) \circ \boldsymbol{g}(k) \tag{11}$$

where the symbol $\circ$ denotes the Hadamard product of vectors. Finally, the hidden state can be calculated

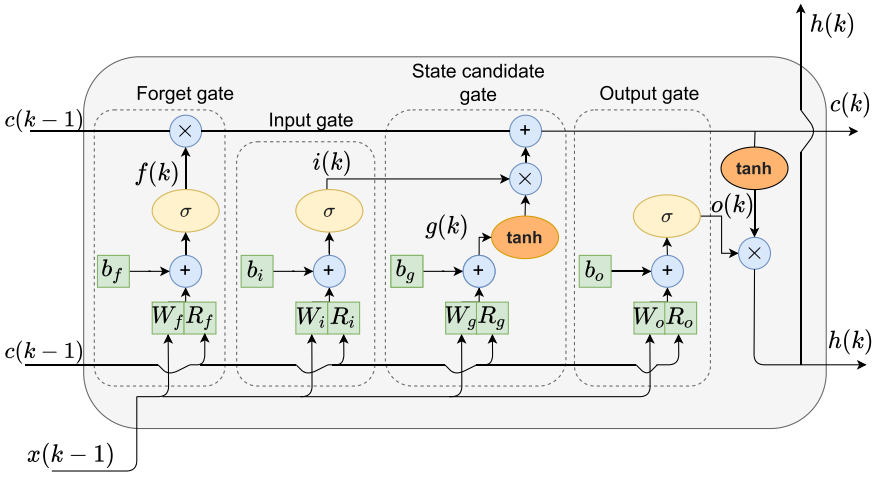

Fig. 2. The LSTM cell structure.

$$\boldsymbol{h}(k) = \boldsymbol{o}(k) \circ \tanh(\boldsymbol{c}(k)) \tag{12}$$

The output of the whole network (Fig. 1) at the time instant k can be computed from Eq. (5) which in vector–matrix notation becomes

$$y(k) = \boldsymbol{W}_{\mathrm{y}}\boldsymbol{h}(k) + b_{\mathrm{y}} \tag{13}$$

where $\boldsymbol{W}_{\mathrm{y}}$ is a $1 \times n_{\mathrm{N}}$ vector and b_{y} is a bias scalar coefficient. One can write Eqs. (7)–(12) in a scalar form. They will be useful for the derivation of MPC algorithms. The n-th elements of the gate and state vectors in the scalar form are

$$i_n(k) = \sigma\left(\sum_{m=1}^{n_{\mathrm{A}}+n_{\mathrm{B}}}\left(w^{\mathrm{i}}_{n,m}x_m(k)\right) + \sum_{m=1}^{n_{\mathrm{N}}}\left(r^{\mathrm{i}}_{n,m}h_m(k-1)\right) + b^{\mathrm{i}}_n\right) \tag{14}$$

$$f_n(k) = \sigma\left(\sum_{m=1}^{n_{\mathrm{A}}+n_{\mathrm{B}}}\left(w^{\mathrm{f}}_{n,m}x_m(k)\right) + \sum_{m=1}^{n_{\mathrm{N}}}\left(r^{\mathrm{f}}_{n,m}h_m(k-1)\right) + b^{\mathrm{f}}_n\right) \tag{15}$$

$$g_n(k) = \tanh\left(\sum_{m=1}^{n_{\mathrm{A}}+n_{\mathrm{B}}}\left(w^{\mathrm{g}}_{n,m}x_m(k)\right) + \sum_{m=1}^{n_{\mathrm{N}}}\left(r^{\mathrm{g}}_{n,m}h_m(k-1)\right) + b^{\mathrm{g}}_n\right) \tag{16}$$

$$o_n(k) = \sigma\left(\sum_{m=1}^{n_{\mathrm{A}}+n_{\mathrm{B}}}\left(w^{\mathrm{o}}_{n,m}x_m(k)\right) + \sum_{m=1}^{n_{\mathrm{N}}}\left(r^{\mathrm{o}}_{n,m}h_m(k-1)\right) + b^{\mathrm{o}}_n\right) \tag{17}$$

$$c_n(k) = f_n(k)c_n(k-1) + i_n(k)g_n(k) \tag{18}$$

$$h_n(k) = o_n(k)\tanh\left(c_n(k)\right) \tag{19}$$

$$y(k) = \sum_{n=1}^{n_{\mathrm{N}}} w^n_{\mathrm{y}}h_n(k) + b_{\mathrm{y}} \tag{20}$$

From Eqs. (14)–(20) it is possible to derive the output of the network

$$y(k) = \sum_{n=1}^{n_{\mathrm{N}}} w^n_{\mathrm{y}}(o_n(k)\tanh\left(f_n(k)c_n(k-1) + i_n(k)g_n(k)\right)) + b_{\mathrm{y}} \tag{21}$$

3.2. GRU Model

In the GRU network, the standard LSTM cell layout is modified in the following ways: 3 gates instead of 4 are only used and the cell state c is discarded [7]. The single GRU cell schematic is depicted in Fig. 3. The reset gate r selects information from the previous state h to be discarded. The update gate z selects new information from the input vector and the previous state h to be added to a new state. The candidate for the future hidden state is found by the candidate state gate g.

The GRU network has n_{N} cells. The weighting matrices are

$$\boldsymbol{W} = \begin{bmatrix}\boldsymbol{W}_{\mathrm{q}}\\ \boldsymbol{W}_{\mathrm{z}}\\ \boldsymbol{W}_{\mathrm{g}}\end{bmatrix},\ \boldsymbol{R} = \begin{bmatrix}\boldsymbol{R}_{\mathrm{q}}\\ \boldsymbol{R}_{\mathrm{z}}\\ \boldsymbol{b}_{\mathrm{g}}\end{bmatrix},\ \boldsymbol{b} = \begin{bmatrix}\boldsymbol{b}_{\mathrm{q}}\\ \boldsymbol{b}_{\mathrm{z}}\\ \boldsymbol{b}_{\mathrm{g}}\end{bmatrix} \tag{22}$$

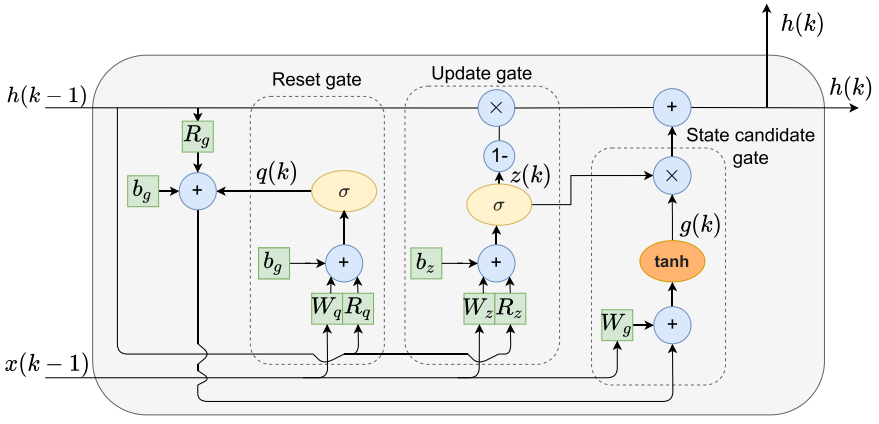

Fig. 3. The GRU cell structure.

The $n_N \times (n_A + n_B)$ matrices are named $\boldsymbol{W}_q, \boldsymbol{W}_z, \boldsymbol{W}_g$, the $n_N \times n_N$ matrices are denoted by $\boldsymbol{R}_q, \boldsymbol{R}_z, \boldsymbol{R}_g$, and $n_N \times 1$ vectors are represented by $\boldsymbol{b}_q, \boldsymbol{b}_z, \boldsymbol{b}_g$.

At the sampling time k, the calculations are as follows. Firstly the output of each gate is determined. Then, the current value of hidden state h is calculated. We obtain

$$\boldsymbol{q}(k) = \sigma(\boldsymbol{W}_q \boldsymbol{x}(k) + \boldsymbol{q}_q \boldsymbol{h}(k-1) + \boldsymbol{b}_q) \tag{23}$$

$$\boldsymbol{z}(k) = \sigma(\boldsymbol{W}_z \boldsymbol{x}(k) + \boldsymbol{R}_z \boldsymbol{h}(k-1) + \boldsymbol{b}_z) \tag{24}$$

$$\boldsymbol{g}(k) = \tanh(\boldsymbol{W}_g \boldsymbol{x}(k) + \boldsymbol{q}(k) \circ (\boldsymbol{R}_g \boldsymbol{h}(k-1)) + \boldsymbol{b}_g) \tag{25}$$

$$\boldsymbol{h}(k) = (\boldsymbol{1}_{n_N \times 1} - \boldsymbol{z}(k)) \circ \boldsymbol{g}(k) + \boldsymbol{z}(k) \circ \boldsymbol{h}(k-1) \tag{26}$$

The output of the whole network at the time k can be computed from Eqs. (5) or (13). The n-th elements of the gate and state vectors in scalar form and the output of the network are

$$q_n(k) = \sigma\left(\sum_{m=1}^{(n_A+n_B)} \left(w^q_{n,m} x_m(k)\right) + \sum_{m=1}^{n_N} \left(r^q_{n,m} h_m(k-1)\right) + b^q_n\right) \tag{27}$$

$$z_n(k) = \sigma\left(\sum_{m=1}^{(n_A+n_B)} \left(w^z_{n,m} x_m(k)\right) + \sum_{m=1}^{n_N} \left(r^z_{n,m} h_m(k-1)\right) + b^z_n\right) \tag{28}$$

$$g_n(k) = \tanh\left(\sum_{m=1}^{(n_A+n_B)} \left(w^g_{n,m} x_m(k)\right) + q_n \sum_{m=1}^{n_N} \left(r^g_{n,m} h_m(k-1)\right) + b^g_n\right) \tag{29}$$

$$h_n(k) = (1 - z_n(k)) g_n(k) + z_n(k) h_n(k-1) \tag{30}$$

$$y(k) = \sum_{n=1}^{n_N} w^n_y h_n(k) + b_y \tag{31}$$

Combining Eqs. (27)–(31), one can determine the output of the network in a form of a single equation

$$y(k) = \sum_{n=1}^{n_N} w^n_y ((1 - z_n(k)) o_n(k) + z_n(k) h_n(k-1)) + b_y \tag{32}$$

4. Two Classical Approaches to MPC Using LSTM and GRU Models

4.1. MPC Using Nonlinear Optimization

At each sampling instant of the MPC algorithm with Nonlinear Optimization (MPC-NO), the LSTM or GRU network is directly used for prediction calculation and optimization. The predictions are

$$\hat{y}(k+i|k) = y(k+i|k) + d(k) \tag{33}$$

for $i = 1, \ldots, N$. The model output for the future discrete time $k+i$ found at the current time k is $y(k+i|k)$. To guarantee the offset-free control, i.e., no steady-state error of the controlled process output variable, the unmeasured disturbance $d(k)$ acting on the process must be taken into account during prediction calculation in Eq. (33). It reflects actual process disturbances and model inaccuracies. It is approximated as the difference between the measurement of the process output and the LSTM or GRU model's output. During the calculation of the model's output, the signals up to the sampling instant $k-1$ are used, i.e.

$$d(k) = y(k) - y_{\text{model}}(k|k-1) \tag{34}$$

For the LSTM structure, using Eqs. (21) and (33), the prediction for the $k+i$ time instant is

$$\hat{y}(k+i|k) = \sum_{n=1}^{n_N} w^n_y (o_n(k+i|k) \tanh(f_n(k+i|k) c_n(k+i-1|k) + i_n(k+i|k) g_n(k+i|k))) + b_y + d(k) \tag{35}$$

where from Eqs. (21) and (34), we have

$$d(k) = y(k) - \sum_{n=1}^{n_N} w^n_y (o_n(k|k-1) \tanh(f_n(k|k-1) c_n(k|k-1) + i_n(k|k-1) g_n(k|k-1))) - b_y \tag{36}$$

The signals o_n, f_n, c_n, i_n and g_n are calculated recurrently from Eqs. (14)–(19).

For the GRU model, using Eqs. (32) and (33), the prediction for the $k+i$ time instant are

$$\hat{y}(k+i|k) = \sum_{n=1}^{n_N} w^n_y ((1 - z_n(k+i|k)) o_n(k+i|k) + z_n(k+i|k) h_n(k+i-1|k)) + b_y + d(k) \tag{37}$$

where from Eqs. (32) and (34), we have

$$d(k) = y(k) - \sum_{n=1}^{n_N} w_y^n((1 - z_n(k|k-1))o_n(k|k-1) + z_n(k|k-1)h_n((k-1|k-1))) - b_y \tag{38}$$

The signals z_n, o_n and h_n are calculated recurrently from Eqs. (28)–(30).

In the case of two considered model structures, since the predictions $\hat{y}(k+i|k)$ depend nonlinearly on the future control signals $u(k+i|k)$, we have to solve the nonlinear MPC-NO optimization task at each sampling time online. Examples of that approach are discussed in [19,21,35,47,39,3,22,47] for LSTM and GRU networks, respectively.

4.2. MPC Using Online Model Linearization and Quadratic Optimization

To reduce the high computational difficulty of the MPC-NO algorithm, the classical solution in MPC is to get successively online a linear time-varying equivalent of the nonlinear model; in our case, the LSTM or GRU models are considered. In the MPC algorithm with Nonlinear Prediction and Linearization (MPC-NPL), the following linearized model is used [23,24]. The time-varying linearized model is

$$y(k) = \sum_{m=1}^{n_B} b_m(k)u(k-m) - \sum_{m=1}^{n_A} a_m(k)y(k-m) \tag{39}$$

Note, that the linear coefficients b_m and a_m are not constant. They are calculated in each time iteration k as

$$b_m(k) = \frac{\partial y(k)}{\partial u(k-m)}, \; a_m(k) = -\frac{\partial y(k)}{\partial y(k-m)} \tag{40}$$

where $y(k)$ is obtained from Eqs. (21) or (32) for LSTM and GRU models, respectively. The linearized process description (39) is used to capture the impact of the current and future manipulated variables, which comprise the MPC decision vector (1). The influence of the past signals is performed using the full LSTM or GRU nonlinear model. As a result, the forecast signals $\hat{y}(k+i|k)$ depend linearly on the decision vector and we obtain a quadratic optimization task (2). Derivation and discussion of the MPC-NPL control procedure are presented in [23,24], implementation for the LSTM network is presented in [32,36]. Such a technique is much simpler than the nonlinear MPC-NO control scheme that requires nonlinear programming. Unfortunately, as shown later in Section 6, the MPC-NPL algorithm relying on simple successive model linearization is likely to give imprecise predictions and not satisfying control quality. Note that for both LSTM and GRU networks, the output signal for the current time step, i.e. $y(k)$, is actually a general function of past process signals as specified in Eq. (4). In fact, having considered specific equations for LSTM and GRU networks, i.e., Eqs. (14)–(19) and Eqs. (27)–(31), respectively, we may conclude that the model output is in fact a function of the model input vector (4) and the signal $h(k-1)$. Hence, the model performs the mapping

$$y(k) = F_1(u(k-1), \ldots, u(k-n_B), y(k-1), \ldots, y(k-n_A), h(k-1)) \tag{41}$$

where the signal $h(k-1)$ depends recurrently on the signal $h(k-2)$

$$h(k-1) = F_2(u(k-2), \ldots, u(k-n_B-1), y(k-2), \ldots, y(k-n_A-1), h(k-2)) \tag{42}$$

the signal $h(k-2)$ depends recurrently on the signal $h(k-3)$

$$h(k-2) = F_3(u(k-3), \ldots, u(k-n_B-2), y(k-3), \ldots, y(k-n_A-2), h(k-3)) \tag{43}$$

etc. The simple successive model linearization is described by Eqs. (39) and (40) disregards all the information contained in the hidden state $h(k-1)$. In other words, this type of linearization does not consider signal history, leading to poor linear model quality. Hence, we will consider a more complex approach to MPC, which also requires quadratic optimization without solving nonlinear tasks.

5. Computationally Efficient MPC Algorithms Using LSTM and GRU Models

Having pointed out weak points of MPC with full nonlinear optimization and MPC with successive model linearization, we will concentrate on MPC algorithms with online trajectory linearization. The general idea of such MPC methods is discussed thoroughly in [24,25]; here, we describe the algorithm's details for LSTM and GRU networks. In general, in the MPC algorithm with Nonlinear Prediction and Linearization around the Trajectory (MPC-NPLT), the model is not successively linearized which is next used for finding the output predicted trajectory, but the predicted trajectory of the controlled variable within the entire prediction horizon, i.e., the vector

$$\hat{\boldsymbol{y}}(k) = [\hat{y}(k+1|k) \; \hat{y}(k+2|k) \ldots \hat{y}(k+N|k)]^T \tag{44}$$

is actually directly linearized. The linearization is performed for an assumed trajectory of future values of the control signal over the control horizon

$$\boldsymbol{u}^{\text{traj}}(k) = \left[u^{\text{traj}}(k|k)\ u^{\text{traj}}(k+1|k)\ldots u^{\text{traj}}(k+N_{\text{u}}-1|k)\right]^{\text{T}} \tag{45}$$

From the definition of the control horizon, it follows that $u^{\text{traj}}(k+i|k) = u^{\text{traj}}(k+N_{\text{u}}-1|k)$ for $i = N_{\text{u}},\ldots,N$. The trajectory about which linearization is performed has to be assumed due to the fact that at the current time k the current and future values of the controlled process input are not known. When the vector $\boldsymbol{u}^{\text{traj}}(k)$ is defined by the last calculated and applied to the process value of the control signal, i.e. the quantity $u(k-1)$, we use the term MPC-NPLT1. If the vector $\boldsymbol{u}^{\text{traj}}(k)$ is defined by the tail of the optimal control sequence found at the previous time step, i.e., $k-1$, (without the signal $u(k-1)$ actually applied to the process), we use the term MPC-NPLT2. Using the nonlinear process model recurrently, in our case represented by the LSTM or GRU neural network, the output trajectory that corresponds to the assumed input trajectory can be calculated

$$\hat{\boldsymbol{y}}^{\text{traj}}(k) = \left[\hat{y}^{\text{traj}}(k+1|k)\ \hat{y}^{\text{traj}}(k+2|k)\ldots\hat{y}^{\text{traj}}(k+N|k)\right]^{\text{T}} \tag{46}$$

For this purpose, the equations presented in Section 4.1 are used. The linear approximation of the future trajectory of the process output signal (Eq. (44)) is

$$\hat{\boldsymbol{y}}(k) = \hat{\boldsymbol{y}}^{\text{traj}}(k) + \boldsymbol{H}(k)\left(\boldsymbol{u}(k) - \boldsymbol{u}^{\text{traj}}(k)\right) \tag{47}$$

where

$$\boldsymbol{H}(k) = \frac{\text{d}\hat{\boldsymbol{y}}^{\text{traj}}(k)}{\text{d}\boldsymbol{u}^{\text{traj}}(k)} \tag{48}$$

is the $N \times N_{\text{u}}$ matrix comprised of the partial derivatives of the predicted trajectory $\hat{\boldsymbol{y}}(k)$ with respect to the input trajectory $\boldsymbol{u}^{\text{traj}}(k)$. The elements of that matrix are

$$H_{j+1,i}(k) = \frac{\partial\hat{y}^{\text{traj}}(k+i|k)}{\partial u^{\text{traj}}(k+j|k)} \tag{49}$$

The vector $\boldsymbol{u}(k)$ in Eq. (47) corresponds to the decision variable vector $\triangle\boldsymbol{u}(k)$ (Eq. 1)

$$\boldsymbol{u}(k) = \boldsymbol{J}\triangle\boldsymbol{u}(k) + \boldsymbol{u}(k-1) \tag{50}$$

where the entries of the $N_{\text{u}} \times N_{\text{u}}$ matrix $\boldsymbol{J}$ are established as

$$J_{i,j} = \begin{cases} 0 & \text{if } i < j \\ 1 & \text{if } i \geqslant j \end{cases} \tag{51}$$

and

$$\boldsymbol{u}(k-1) = u(k-1)[1\ \ldots\ 1]^{\text{T}} \tag{52}$$

is the vector of length N_{u}.

In the discussed MPC algorithms with online trajectory linearization, taking into account Eqs. (47) and (50), the basic MPC optimization task (2) is converted to the quadratic optimization task

$$\begin{aligned} &\underset{\triangle\boldsymbol{u}(k)}{\text{minimize}} \left\{\|\boldsymbol{y}^{\text{sp}}(k) - \boldsymbol{H}(k)\boldsymbol{J}\triangle\boldsymbol{u}(k) - \hat{\boldsymbol{y}}(k) - \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}(k))\|^2 + \|\triangle\boldsymbol{u}(k)\|_{\boldsymbol{\Lambda}}^2\right\} \\ &\text{subject to} \\ &\underline{\boldsymbol{u}} \leqslant \boldsymbol{J}\triangle\boldsymbol{u}(k) + \boldsymbol{u}(k-1) \leqslant \overline{\boldsymbol{u}} \\ &\triangle\underline{\boldsymbol{u}} \leqslant \triangle\boldsymbol{u}(k) \leqslant \triangle\overline{\boldsymbol{u}} \\ &\underline{\boldsymbol{y}} \leqslant \boldsymbol{H}(k)\boldsymbol{J}\triangle\boldsymbol{u}(k) + \hat{\boldsymbol{y}}(k) + \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}(k)) \leqslant \overline{\boldsymbol{y}} \end{aligned} \tag{53}$$

Definition of all symbols used in the above problem is the following: $\boldsymbol{\Lambda}$ is a diagonal $N_{\text{u}} \times N_{\text{u}}$ matrix whose diagonal elements are all equal to the weighting coefficient λ; the vectors of length N_{u} are $\underline{\boldsymbol{u}} = \underline{u}[1\ \ldots\ 1]^{\text{T}}, \overline{\boldsymbol{u}} = \overline{u}[1\ \ldots\ 1]^{\text{T}}$, $\triangle\underline{\boldsymbol{u}} = \triangle\underline{u}[1\ \ldots\ 1]^{\text{T}}, \triangle\overline{\boldsymbol{u}} = \triangle\overline{u}[1\ \ldots\ 1]^{\text{T}}$; the vectors of length N are $\underline{\boldsymbol{y}} = \underline{y}[1\ \ldots\ 1]^{\text{T}}, \overline{\boldsymbol{y}} = \overline{y}[1\ \ldots\ 1]^{\text{T}}$.

In the control strategy described above, linearization is accomplished once at each algorithm's execution and it is followed by the solution of a quadratic programming task (53). A straightforward extension of that approach, named MPC algorithm with Nonlinear Prediction and Linearization around the Predicted Trajectory (MPC-NPLPT) [24] is possible. At each execution of that algorithm, at first, linearization and optimization are carried out for the selected trajectory $\boldsymbol{u}^{\text{traj}}(k)$. Next, the obtained solution is used for the next linearization, followed by optimization. Such repetitions may be repeated 2–5 times, particularly when the operating conditions change abruptly.

5.1. Implementation for LSTM Model

For the LSTM model, it is obligatory to derive the predicted vector $\hat{\boldsymbol{y}}^{\text{traj}}(k)$ defined by Eq. (46) and the elements of the derivatives' matrix $\boldsymbol{H}(k)$ defined by Eq. (49).

The consecutive elements of the trajectory $\hat{\boldsymbol{y}}^{\text{traj}}(k)$ within the prediction horizon, i.e. for $i = 1, \ldots, N$, are found for the set trajectory $\boldsymbol{u}^{\text{traj}}(k)$ (Eq. 45) from Eq. (35) and Eqs. (14)–(19). Firstly, let us note that two nonlinear functions appear in the formulas describing the LSTM and GRU networks: the sigmoid function and the hyperbolic tangent. The general formulas for their derivatives are

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x)) \tag{54}$$

$$\frac{\partial \tanh(x)}{\partial x} = (1 - \tanh(x))^2 \tag{55}$$

Having that in mind let as focus on the partial derivatives of the predicted trajectory $\hat{y}^{\text{traj}}(k)$, defined by Eq. (49). They are calculated by differentiating Eq. (21)

$$\frac{\partial \hat{y}(k+i|k)}{\partial u(k+j|k)} = \sum_{n=1}^{n_{\text{N}}} w_{\text{n}}^{y} \frac{\partial h_n(k+i|k)}{\partial u(k+j|k)} \tag{56}$$

for all $i = 1, \ldots, N$ and for $j = 0, \ldots, N_{\text{u}} - 1$. For simplicity of derivation and presentation, $u = u^{\text{traj}}$ in all subsequent formulas. Taking into account Eq. (4), the vector of predicted future model arguments at the moment $k + i$, where $i = 1, 2, 3, \ldots$, is

$$\begin{aligned}
x(k+1|k) &= [u(k|k)\ u(k-1)\ u(k-2) \ldots u(k-n_{\text{B}}+1)\ y(k)\ y(k-1)\ y(k-2) \ldots y(k-n_{\text{A}}+1)]^{\text{T}} \\
x(k+2|k) &= [u(k+1|k)\ u(k|k)\ u(k-1) \ldots u(k-n_{\text{B}}+2)\ \hat{y}(k+1|k)\ y(k)\ y(k-1) \ldots y(k-n_{\text{A}}+2)]^{\text{T}} \\
x(k+3|k) &= [u(k+2|k)\ u(k+1|k)\ u(k|k) \ldots u(k-n_{\text{B}}+3)\ \hat{y}(k+2|k)\ \hat{y}(k+1|k)\ y(k) \ldots y(k-n_{\text{A}}+3)]^{\text{T}} \\
&\vdots
\end{aligned} \tag{57}$$

Let $I_{\text{uf}}(i) = \max(\min(i, n_{\text{B}}), 0)$ and $I_{\text{yf}}(i) = \min(i-1, n_{\text{A}})$ be a number of future control signals and predicted output values for the future moment $k + i$, respectively. Then, using Eqs. (14)–(17), we can express the predicted gate outputs as

$$\begin{aligned}
i_n(k+i|k) = \sigma\Bigg(&\sum_{m=1}^{I_{\text{uf}}(i)} w_{n,m}^{\text{i}} u(k-m+i|k) + \sum_{m=I_{\text{uf}}(i)+1}^{n_{\text{B}}} w_{n,I_{\text{uf}}(i)+m}^{\text{i}} u(k-m+i) + \sum_{m=1}^{I_{\text{yf}}(i)} w_{n,n_{\text{B}}+m}^{\text{i}} \hat{y}(k-m+i|k) \\
&+ \sum_{m=I_{\text{yf}}(i)+1}^{n_{\text{A}}} w_{n,I_{\text{yf}}(i)+m}^{\text{i}} y(k-m+i) + \sum_{m=1}^{n_{\text{N}}} r_{n,m}^{\text{i}} h_m(k+i-1|k) + b_n^{\text{i}} \Bigg)
\end{aligned} \tag{58}$$

$$\begin{aligned}
f_n(k+i|k) = \sigma\Bigg(&w_{n,m}^{\text{f}} \sum_{m=1}^{I_{\text{uf}}(i)} u(k-m+i|k) + w_{n,I_{\text{uf}}(i)+m}^{\text{f}} \sum_{m=I_{\text{uf}}(i)+1}^{n_{\text{B}}} u(k-m+i) + w_{n,n_{\text{B}}+m}^{\text{f}} \sum_{m=1}^{I_{\text{yf}}(i)} \hat{y}(k-m+i|k) \\
&+ w_{n,I_{\text{yf}}(i)+m}^{\text{f}} \sum_{m=I_{\text{yf}}(i)+1}^{n_{\text{A}}} y(k-m+i) + \sum_{m=1}^{n_{\text{N}}} \left(r_{n,m}^{\text{f}} h_m(k) \right) + b_n^{\text{f}} \Bigg)
\end{aligned} \tag{59}$$

$$\begin{aligned}
g_n(k+i|k) = \tanh\Bigg(&w_{n,m}^{\text{g}} \sum_{m=1}^{I_{\text{uf}}(i)} u(k-m+i|k) + w_{n,I_{\text{uf}}(i)+m}^{\text{g}} \sum_{m=I_{\text{uf}}(i)+1}^{n_{\text{B}}} u(k-m+i) + w_{n,n_{\text{B}}+m}^{\text{g}} \sum_{m=1}^{I_{\text{yf}}(i)} \hat{y}(k-m+i|k) \\
&+ w_{n,I_{\text{yf}}(i)+m}^{\text{g}} \sum_{m=I_{\text{yf}}(i)+1}^{n_{\text{A}}} y(k-m+i) + \sum_{m=1}^{n_{\text{N}}} \left(r_{n,m}^{\text{g}} h_m(k) \right) + b_n^{\text{g}} \Bigg)
\end{aligned} \tag{60}$$

$$\begin{aligned}
o_n(k+i|k) = \sigma\Bigg(&w_{n,m}^{\text{o}} \sum_{m=1}^{I_{\text{uf}}(i)} u(k-m+i|k) + w_{n,I_{\text{uf}}(i)+m}^{\text{o}} \sum_{m=I_{\text{uf}}(i)+1}^{n_{\text{B}}} u(k-m+i) + w_{n,n_{\text{B}}+m}^{\text{o}} \sum_{m=1}^{I_{\text{yf}}(i)} \hat{y}(k-m+i|k) \\
&+ w_{n,I_{\text{yf}}(i)+m}^{\text{o}} \sum_{m=I_{\text{yf}}(i)+1}^{n_{\text{A}}} y(k-m+i) + \sum_{m=1}^{n_{\text{N}}} \left(r_{n,m}^{\text{o}} h_m(k) \right) + b_n^{\text{o}} \Bigg)
\end{aligned} \tag{61}$$

In each of these equations, there are five weighted sums. The first takes into account all future control signals, while the second is concerned with past control signals. The third and fourth sums specify predicted and past output signals, respectively. Finally, the last sums are required to consider all hidden state vector elements. Of note, elements of the second and the

fourth sums group only the past control signals. Therefore, differentiating them with respect to $u(k+j|k)$ results in zero. The predicted cell and hidden state vectors elements are

$$c_n(k+i|k) = f_n(k+1|k)c_n(k) + i_n(k+i|k)g_n(k+i|k) \tag{62}$$

$$h_n(k+i|k) = o_n(k+i|k)\tanh\left(c_n(k+i|k)\right) \tag{63}$$

Now, we can calculate the derivatives of future gates i, f, g and o outputs. All of those are composite functions $f(g)$, where $f = \sigma(x)$ for i, f and o gates and $f = \tanh(x)$ for g gate. Therefore, differentiating Eq. (58), we have

$$\frac{\partial i_n(k+i|k)}{\partial u(k+j|k)} = i_n(k+i|k) \times \left(1 - i_n(k+i|k)\right)\left(\sum_{m=1}^{I_{\mathrm{uf}}(i)} w^{\mathrm{i}}_{n,m}\frac{\partial u(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{I_{\mathrm{yf}}(i)} w^{\mathrm{i}}_{n,n_{\mathrm{B}}+m}\frac{\partial \hat{y}(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{n_{\mathrm{N}}} r^{\mathrm{i}}_{n,m}\frac{\partial h_m(k+i-1|k)}{\partial u(k+j|k)}\right) \tag{64}$$

from Eq. (59), we obtain

$$\frac{\partial f_n(k+i|k)}{\partial u(k+j|k)} = f_n(k+i|k) \times \left(1 - f_n(k+i|k)\right)\left(\sum_{m=1}^{I_{\mathrm{uf}}(i)} w^{\mathrm{f}}_{n,m}\frac{\partial u(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{I_{\mathrm{yf}}(i)} w^{\mathrm{f}}_{n,n_{\mathrm{B}}+m}\frac{\partial \hat{y}(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{n_{\mathrm{N}}} r^{\mathrm{f}}_{n,m}\frac{\partial h_m(k+i-1|k)}{\partial u(k+j|k)}\right) \tag{65}$$

Eq. (60) gives

$$\frac{\partial g_n(k+i|k)}{\partial u(k+j|k)} = \left(1 - g_n^2(k+1|k)\right)\left(\sum_{m=1}^{I_{\mathrm{uf}}(i)} w^{\mathrm{g}}_{n,m}\frac{\partial u(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{I_{\mathrm{yf}}(i)} w^{\mathrm{g}}_{n,n_{\mathrm{B}}+m}\frac{\partial \hat{y}(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{n_{\mathrm{N}}} r^{\mathrm{g}}_{n,m}\frac{\partial h_m(k+i-1|k)}{\partial u(k+j|k)}\right) \tag{66}$$

and using Eq. (61), we derive

$$\frac{\partial o_n(k+i|k)}{\partial u(k+j|k)} = o_n(k+i|k) \times \left(1 - o_n(k+i|k)\right)\left(\sum_{m=1}^{I_{\mathrm{uf}}(i)} w^{\mathrm{o}}_{n,m}\frac{\partial u(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{I_{\mathrm{yf}}(i)} w^{\mathrm{o}}_{n,n_{\mathrm{B}}+m}\frac{\partial \hat{y}(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{n_{\mathrm{N}}} r^{\mathrm{o}}_{n,m}\frac{\partial h_m(k+i-1|k)}{\partial u(k+j|k)}\right) \tag{67}$$

Next, the derivative of the cell state c can be computed from Eq. (18). This is a sum of two function products $f_1 g_1 + f_2 g_2$. Therefore, we use the Leibniz formula

$$\frac{\partial c_n(k+i|k)}{\partial u(k+j|k)} = \frac{\partial f_n(k+i|k)}{\partial u(k+j|k)}c_n(k+i-1|k) + f_n(k+i|k)\frac{\partial c_n(k+i-1|k)}{\partial u(k+j|k)} + \frac{\partial i_n(k+i|k)}{\partial u(k+j|k)}g_n(k+i|k) + i_n(k+i|k) \times \frac{\partial g_n(k+i|k)}{\partial u(k+j|k)} \tag{68}$$

Next, we can use Eq. (19) to find derivatives of the hidden state h. This is a product of two functions $f_1 f_2(g)$. Therefore, we have

$$\frac{\partial h_n(k+i|k)}{\partial u(k+j|k)} = \frac{\partial o_n(k+i|k)}{\partial u(k+j|k)}\tanh\left(c_n(k+i|k)\right) + o_n(k+i|k)\left(1 - \tanh^2\left(c_n(k+i|k)\right)\right)\frac{\partial c_n(k+i|k)}{\partial u(k+j|k)} \tag{69}$$

Let us emphasize that the derivatives $\frac{\partial u(k+i|k)}{\partial u(k+i|k)}$ can only take the values 1 or 0. Once the control horizon is over, the value of the control signal remains constant, i.e. $u(k+i|k) = u(k+N_{\mathrm{u}}-1|k)$ for $i \geqslant N_{\mathrm{u}}$. The derivatives can, therefore, be written as

$$\frac{\partial u(k+i|k)}{\partial u(k+i|k)} = \begin{cases} 1 & \text{when } i = j \text{ or } (i > j \text{ and } j = N_{\mathrm{u}} - 1) \\ 0 & \text{otherwise} \end{cases} \tag{70}$$

On the other hand, the derivatives $\frac{\partial \hat{y}(k-m+i|k)}{\partial u(k+j|k)}$ and $\frac{\partial h_m(k+i-1|k)}{\partial u(k+j|k)}$ are calculated recursively from Eqs. (56)–(64), (57)–(70). The calculations during the online control are performed in the following order:

1. The partial derivatives of cell output signals are calculated from Eqs. (64)–(66).
2. The cell state partial derivatives are computed from Eq. (68).

3. The hidden state partial derivatives are calculated from Eq. (69).
4. Finally, the network output derivatives can be found from Eq. (56).

All derivatives are determined for $i = 1, \ldots, N$ and $j = 0, \ldots, N_u$.

5.2. Implementation for GRU Model

The calculations are performed similarly for the GRU model as they are carried out for the LSTM model.

The following elements of the predicted trajectory $\hat{y}^{traj}(k)$, i.e. for $i = 1, \ldots, N$, are calculated for the assumed trajectory $u^{traj}(k)$ (Eq. 45) from Eq. (37) and Eqs. (27)–(32).

The partial derivatives of the output trajectory $\hat{y}^{traj}(k)$, defined by Eq. (49), are calculated by differentiating Eq. (32)

$$\frac{\partial \hat{y}(k+i|k)}{\partial u(k+j|k)} = \sum_{n=1}^{n_N} w_n^y \frac{\partial h_n(k+i|k)}{\partial u(k+j|k)} \tag{71}$$

Next, we one must found the derivatives of the q, z and g gates outputs. Considering the discussion presented for the LSTM network in Section 5.1, from Eq. (27), we derive for the GRU model

$$\begin{aligned}\frac{\partial q_n(k+i|k)}{\partial u(k+j|k)} &= q_n(k+i|k) \\ &\times (1 - q_n(k+i|k)) \left(\sum_{m=1}^{I_{uf}(i)} w_{n,m}^q \frac{\partial u(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{I_{yf}(i)} w_{n,n_B+m}^q \frac{\partial \hat{y}(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{n_N} r_{n,m}^q \frac{\partial h_m(k+i-1|k)}{\partial u(k+j|k)} \right)\end{aligned} \tag{72}$$

from Eq. (28), we obtain

$$\begin{aligned}\frac{\partial z_n(k+i|k)}{\partial u(k+j|k)} &= z_n(k+i|k) \\ &\times (1 - z_n(k+i|k)) \left(\sum_{m=1}^{I_{uf}(i)} w_{n,m}^z \frac{\partial u(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{I_{yf}(i)} w_{n,n_B+m}^z \frac{\partial \hat{y}(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{n_N} r_{n,m}^z \frac{\partial h_m(k+i-1|k)}{\partial u(k+j|k)} \right)\end{aligned} \tag{73}$$

and from Eq. (29), we have

$$\begin{aligned}\frac{\partial g_n(k+i|k)}{\partial u(k+j|k)} &= \left(1 - \tanh^2(g_n(k+i|k)) \left(\sum_{m=1}^{I_{uf}(i)} w_{n,m}^g \frac{\partial u(k-m+i|k)}{\partial u(k+j|k)} + \sum_{m=1}^{I_{yf}(i)} w_{n,n_B+m}^g \frac{\partial \hat{y}(k-m+i|k)}{\partial u(k+j|k)} \right. \right. \\ &\left. \left. + q_n(k+i|k) \sum_{m=1}^{n_N} r_{n,m}^g \frac{\partial h_m(k+i-1|k)}{\partial u(k+j|k)} + \frac{\partial q_n(k+i|k)}{\partial u(k+j|k)} \sum_{m=1}^{n_N} r_{n,m}^g h_m(k+i-1|k) \right) \right.\end{aligned} \tag{74}$$

Next, the derivatives of the hidden state h must be found. From Eq. (30), we have

$$\begin{aligned}\frac{\partial h_n(k+i|k)}{\partial u(k+j|k)} &= -\frac{\partial z_n(k+i|k)}{\partial u(k+j|k)} g_n(k+i|k) + (1 - z_n(k+i|k)) \frac{\partial g_n(k+i|k)}{\partial u(k+j|k)} + \frac{\partial z_n(k+i|k)}{\partial u(k+j|k)} h_n(k+i-1|k) + z_n(k+i|k) \\ &\times \frac{\partial h_n(k+i-1|k)}{\partial u(k+j|k)}\end{aligned} \tag{75}$$

The calculations during the online control are performed in the following order:

1. The partial derivatives of cell output signals are calculated from Eqs. (72)–(74).
2. The hidden state partial derivatives are calculated from Eq. (75).
3. Finally, the network output derivatives can be found from Eq. (71).

6. Simulation Results

This section describes two benchmark processes, followed by a brief specification of LSTM and GRU networks employed. Then, the performance of MPC controllers with LSTM and GRU networks is compared. We also analyze the reasons for the different performances of the considered MPC algorithms by investigating the correctness of the predictions generated in different MPC schemes. Finally, the resulting control quality is compared when GRU and LSTM networks with different orders of dynamics are presented.

6.1. Benchmark Processes

The first process under study is a polymerization chemical reactor [8]. The process has one input: the initiator's flow rate. Let it be denoted by F_{I} ($\mathrm{m}^3\,\mathrm{h}^{-1}$). The process has one output: the Number Average Molecular Weight, denoted as (NAMW) ($\mathrm{kg\,kmol}^{-1}$). The input and output signals have been normalized to allow for easy training of neural networks. The scaling is: $u = 100(F_{\mathrm{I}} - 0.016783), y = 0.0001(\mathrm{NAMW} - 20000)$. The polymerization process has a sampling time equal to 1.8 s.

The next process is a neutralization (pH) reactor [12]. It has one input: the (NaOH) stream flow rate, denoted as q_1. It has one output: the pH value of the product. The values of input and output of the process have been scaled: $u = q_1 - 15.5, y = \mathrm{pH} - 7$. The neutralization process has a sampling time equal to 10 s.

Both considered processes are strongly nonlinear; their dynamic and static behavior depends on the operating point. Therefore, these processes are often used as benchmarks to evaluate many nonlinear control methods, e.g., [24].

6.2. Modelling of Benchmark Processes for MPC

All MPC controllers utilize neural models with seven hidden neurons. The results of LSTM and GRU model training for two considered benchmark processes, thoroughly discussed in [47], suggest that models with seven hidden neurons provide a reasonable compromise between modeling quality and model complexity. Models with a lower number of nodes tend to represent the dynamic properties of the process poorly. In contrast, models with a larger number of neurons only marginally improve the control quality while the model's computational cost increases rapidly with each additional neuron. In this work, for two benchmark plants, we compare the results when in MPC, the following networks are used:

- LSTM models with the order of dynamics defined by $n_{\mathrm{A}} = 0$ and $n_{\mathrm{B}} = 2$,
- GRU models with $n_{\mathrm{A}} = 0$ and $n_{\mathrm{B}} = 2$,
- LSTM models with $n_{\mathrm{A}} = n_{\mathrm{B}} = 2$,
- GRU models with $n_{\mathrm{A}} = n_{\mathrm{B}} = 2$.

This choice is also motivated by the results presented in [47]. Models with $n_{\mathrm{A}} = 0$ and $n_{\mathrm{B}} = 1$ are too simple to accurately represent the processes, particularly when the GRU structure is used. On the other hand, choosing a larger order of dynamics produces a model with many internal parameters. The models with $n_{\mathrm{A}} = 0$ and $n_{\mathrm{B}} = 2$ provide a good trade-off between accuracy and complexity consistently for both network types. Therefore, they are treated as the default ones. Their training is relatively fast; they have relatively low errors and, at the same time, their number of internal parameters (weights) is relatively low. The models with $n_{\mathrm{A}} = n_{\mathrm{B}} = 2$ are only used when it is mentioned. The adverse effect of the order of dynamics when $n_{\mathrm{A}} \neq 0$ on possible control quality is discussed when we present the results of simulations.

6.3. Simulation Results and Discussion

In the following subsection, a discussion of the results of the simulation experiments is presented. We compare the possible control quality of the following control algorithms:

- MPC-NO (with nonlinear optimization),
- MPC-NPL (with simple online model linearization and quadratic optimization),
- MPC-NPLT1 (with online trajectory linearization about the manipulated variable utilized for control at the previous algorithm's execution and quadratic optimization),
- MPC-NPLT2 (with online trajectory linearization about the tail of the optimal control sequence calculated at the previous algorithm's execution and quadratic optimization),
- MCP-NPLPT (with successive online trajectory linearizations about the predicted trajectory and quadratic optimization).

The procedure for tuning the controllers is as follows:

1. A constant weighting coefficient $\lambda = 1$ is assumed.
2. The lengths of the prediction horizon, N, and the control horizon, N_{u}, are set as equal to 1 and then gradually increased (both at the same time); the shortest length that provides satisfactory quality of control is chosen.
3. The control horizon is gradually shortened; the shortest length that does not negatively affect the quality of control is selected.
4. The weighting coefficient, λ, is fine-tuned to improve the control quality.

The above procedure results in finding the following parameters: the prediction horizon, N, equals to 10, the control horizon, N_{u}, equals to 5 and the weighting factor, λ, is 0.5. These settings have been used for all MPC types. The fundamental models of the considered processes, comprised of ordinary differential equations [8,12], are used for process simulation. The following control quality indicators are considered: the sum of square errors between the process output and the required set-point, the settling time, and the overshoot.

Simulation results are presented in the following way. At first, in Section 6.3.1, all MPC algorithms relying on LSTM models are compared separately (for two considered benchmark plants). Next, in Section 6.3.2, all MPC algorithms relying on GRU models are analyzed separately. Section 6.3.3 compares the consecutive MPC algorithms for LSTM and GRU models and indicates similarities and differences. Section 6.3.5 analyzes the reasons for different control quality performances of the discussed MPC algorithms by comparing example predicted trajectories. Finally, Section 6.3.6 studies the impact of model order on quality of control.

6.3.1. Comparison of MPC Algorithms Relying on LSTM Networks

Simulation results for the first process, i.e., the polymerization plant, when the LSTM model is used, are presented in Figs. 4 and 5. The MPC-NO and MPC-NPLPT controllers provide the best control quality; the obtained settling time is relatively short, and the overshoot is small. The differences in the performance of both algorithms are negligibly small. Of course, we prefer the second of these algorithms since it needs quadratic programming. On the contrary, MPC-NPL performs correctly for most of the simulation time but fails to provide good control for the step-point step occurring at $k = 120$. It is because a time-varying linearized model is used. As discussed in Section 4.2, during successive model linearization, simplifications are inevitable. The inaccurate model and, therefore, inaccurate prediction in the controller causes the settling time to increase significantly. MPC-NPLT1 and MPC-NPLT2 also work very correctly. The differences between the control quality provided by them are not significantly different from that of MPC-NO.

Simulation results for the second process, i.e., the neutralization plant, when the LSTM model is used, are presented in Figs. 6 and 7. Also, for this process, the MPC-NO and MPC-NPLPT controllers provide the best control quality; the obtained trajectories are very close. The MPC-NPL, on the contrary, is characterized by a much higher overshoot. MPC-NPLT1 and MPC-NPLT2 perform very similarly to MPC-NO for most simulation time. Apparent differences are observed around the time step $k = 160$. In the case of the neutralization process, multiple linearizations utilized in the MPC-NPLPT control scheme are necessary.

6.3.2. Comparison of MPC Algorithms Relying on GRU Networks

Simulation results for the first process, i.e., the polymerization plant, when the GRU model is used, are presented in Figs. 8 and 9. The simulation results confirm that also, for GRU models, the MPC-NO provides the best control quality. MPC-NPLPT performs in a very similar way to MPC-NO. The performance of the MPC-NPL is significantly worse; after the set-point step at the time step $k = 150$, the control time increases, and overshoots occur. What is worse, the last set-point is not reached during the simulation horizon. The MPC-NPLT1 and MPC-NPLT2 controllers provide results close to those obtained for MPC-NO.

Simulation results for the second process, i.e., the neutralization plant, when the GRU model is used, are presented in Figs. 10 and 11. The conclusions are the same as those formulated when the LSTM model is employed, i.e., MPC-NO and

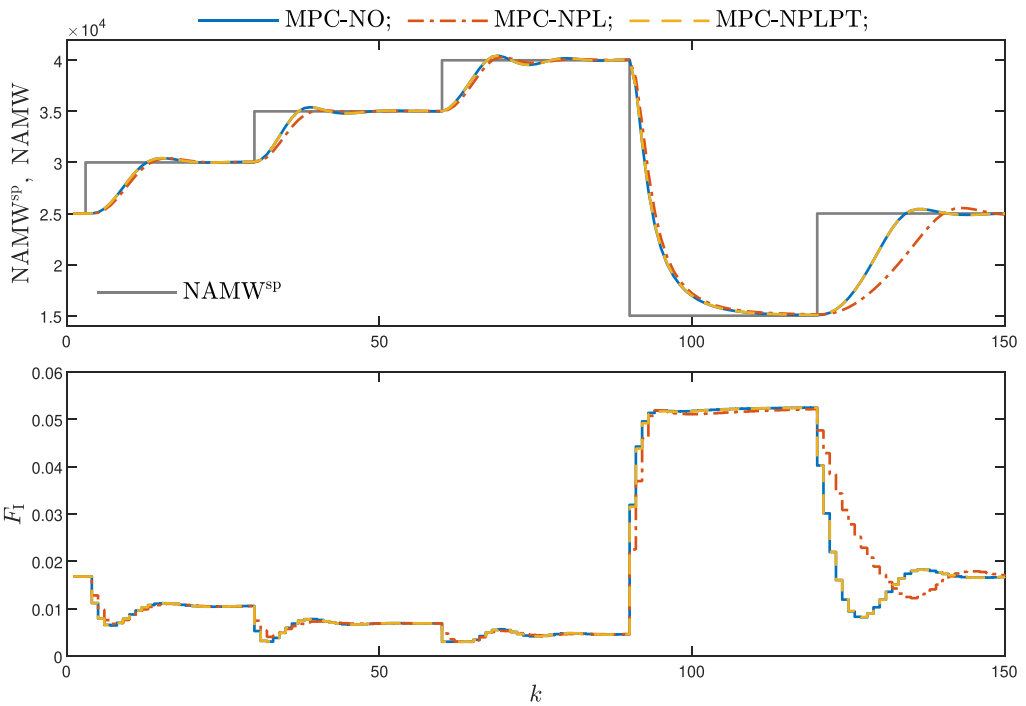

Fig. 4. The process 1: the process output and input signals for the MPC-NO, MPC-NPL and MPC-NPLPT algorithms relying on the LSTM network.

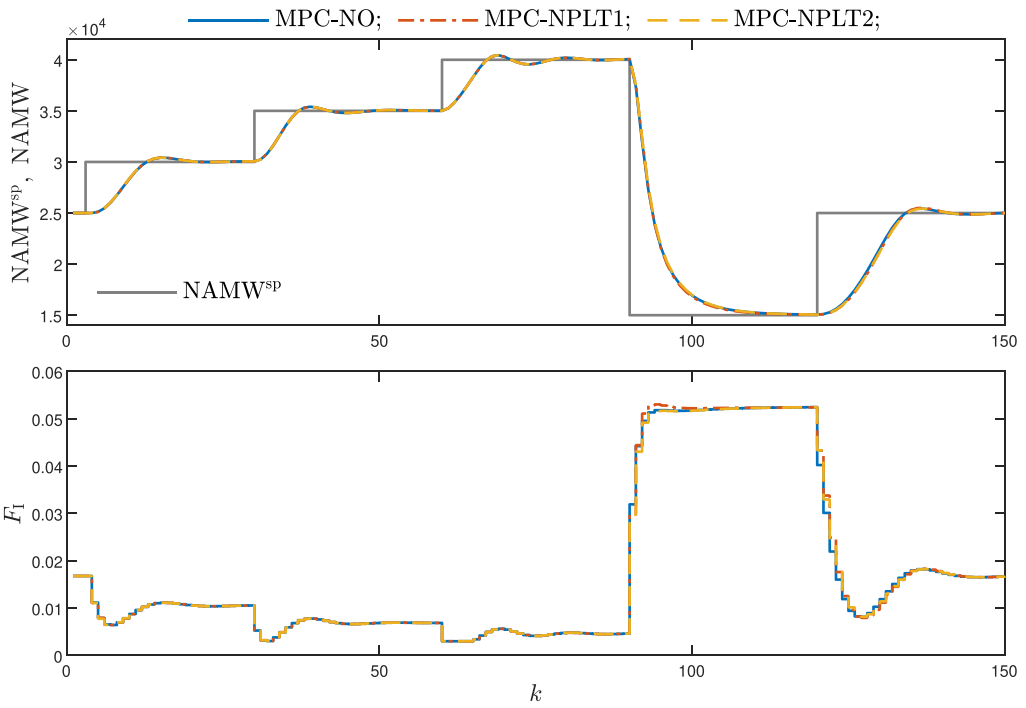

Fig. 5. The process 1: the process output and input signals for the MPC-NO, MPC-NPLT1 and MPC-NPLT2 algorithms relying on the LSTM network.

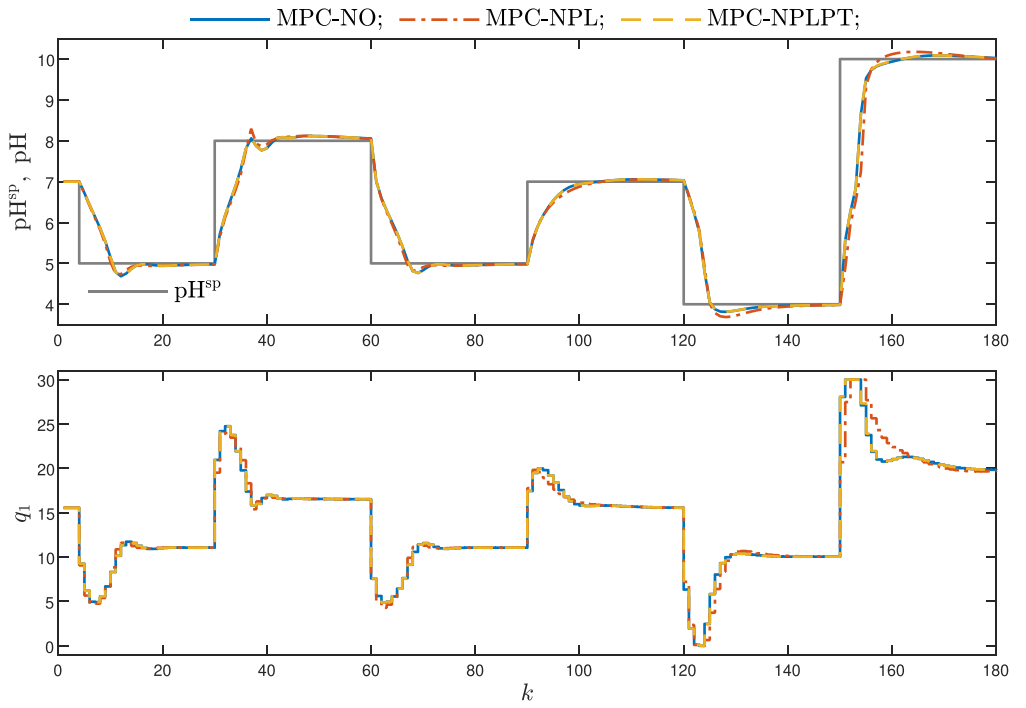

Fig. 6. The process 2: the process output and input signals for the MPC-NO, MPC-NPL and MPC-NPLPT algorithms relying on the LSTM network.

MPC-NPLT provide the best control quality. The MPC-NPL performs worse; the setting time increases significantly, it is observed for the second, fourth, and sixth set-point changes. We can observe that the control quality of MPC-NPLT1 and MPC-NPLT2, in general, is near to that possible in MPC-NO.

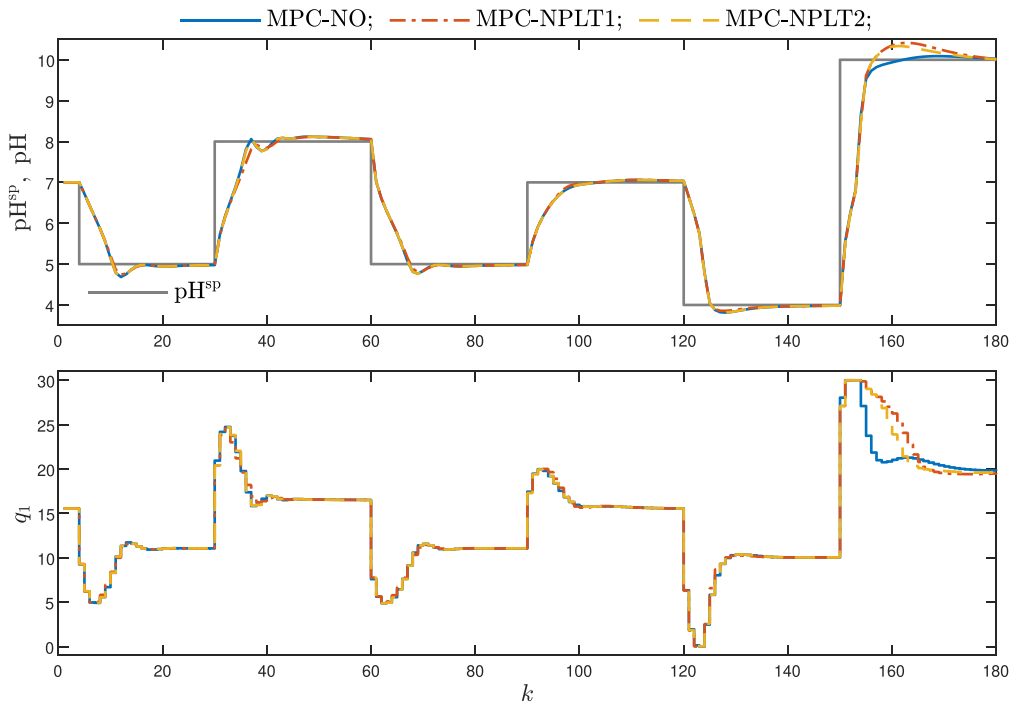

Fig. 7. The process 2: the process output and input signals for the MPC-NO, MPC-NPLT1 and MPC-NPLT2 algorithms relying on the LSTM network.

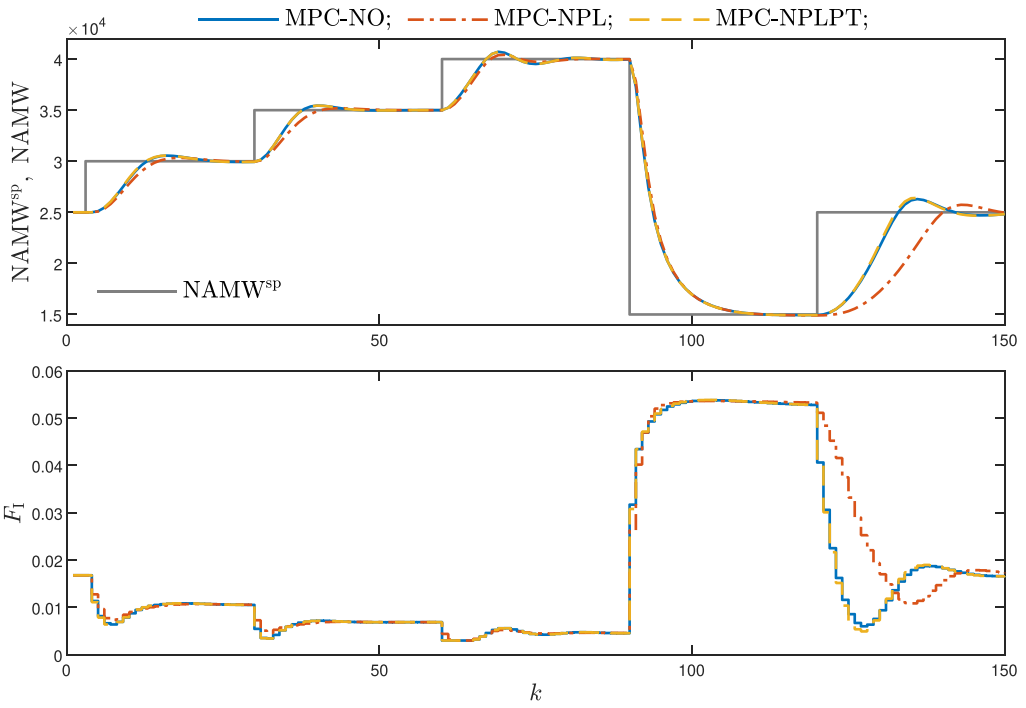

Fig. 8. The process 1: the process output and input signals for the MPC-NO, MPC-NPL, and MPC-NPLPT algorithms relying on the GRU network.

6.3.3. Comparison of MPC Algorithms Relying on LSTM and GRU Networks

For two considered model classes, i.e., LSTM and GRU, our observations and conclusions concerned with the control quality of the examined controllers are the same. Namely, the trajectories obtained in the case of the MPC-NPLPT method are the

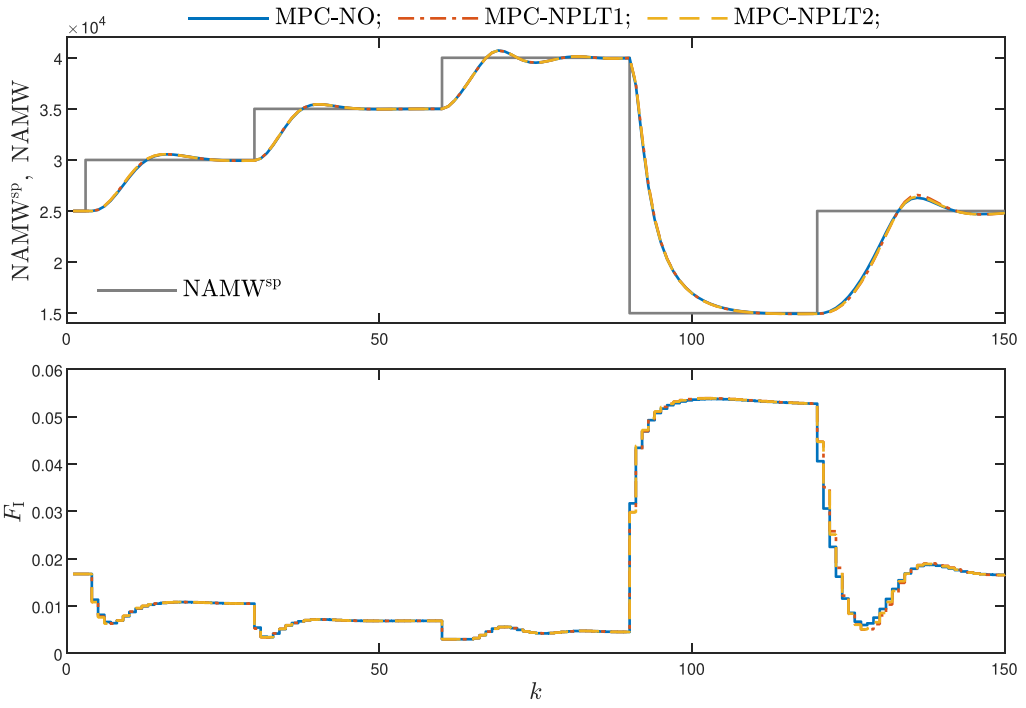

Fig. 9. The process 1: the process output and input signals for the MPC-NO, MPC-NPLT1, and MPC-NPLT2 algorithms relying on the GRU network.

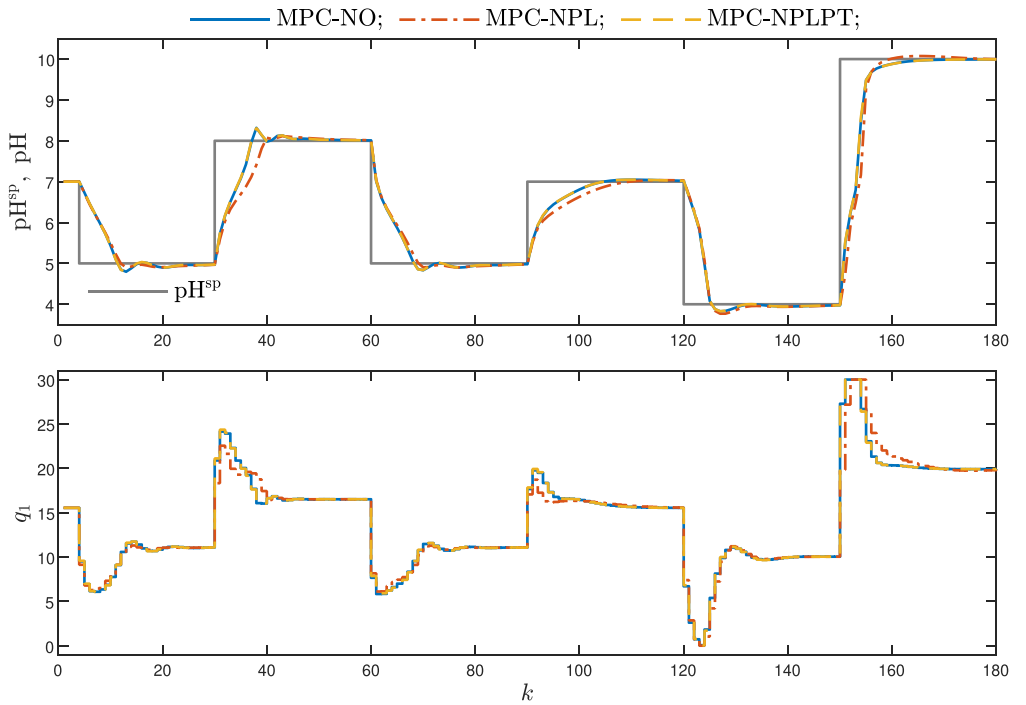

Fig. 10. The process 2: the process output and input signals for the MPC-NO, MPC-NPL, and MPC-NPLPT algorithms relying on the GRU network.

same as those in the MPC-NO scheme but require only quadratic optimization; MPC-NPLT1 and MPC-NPLT2 are good while MPC-NPL is much worse. Now, it is worth comparing the MPC-NPLPT algorithm relying on two models. Fig. 12 presents such a comparison for the polymerization reactor. The use of the LSTM model results in a slightly larger overshoot, for example, at the sampling time $k = 40$, and a slower response for $k = 80$, but the differences are not significant.

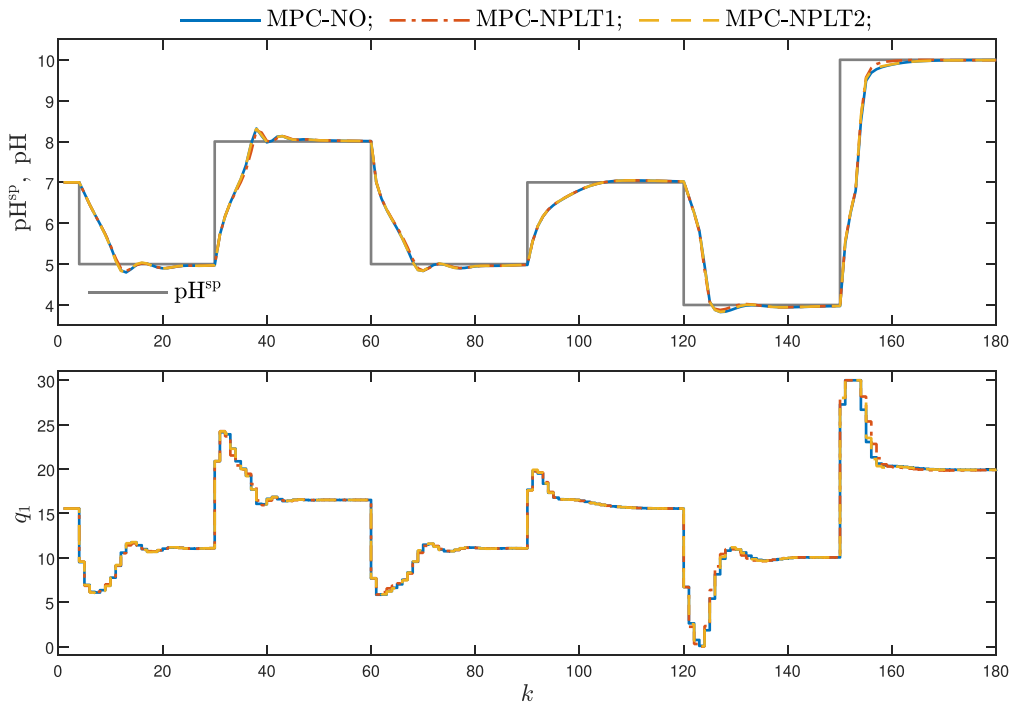

Fig. 11. The process 2: the process output and input signals for the MPC-NO, MPC-NPLT1 and MPC-NPLT2 algorithms relying on the GRU network.

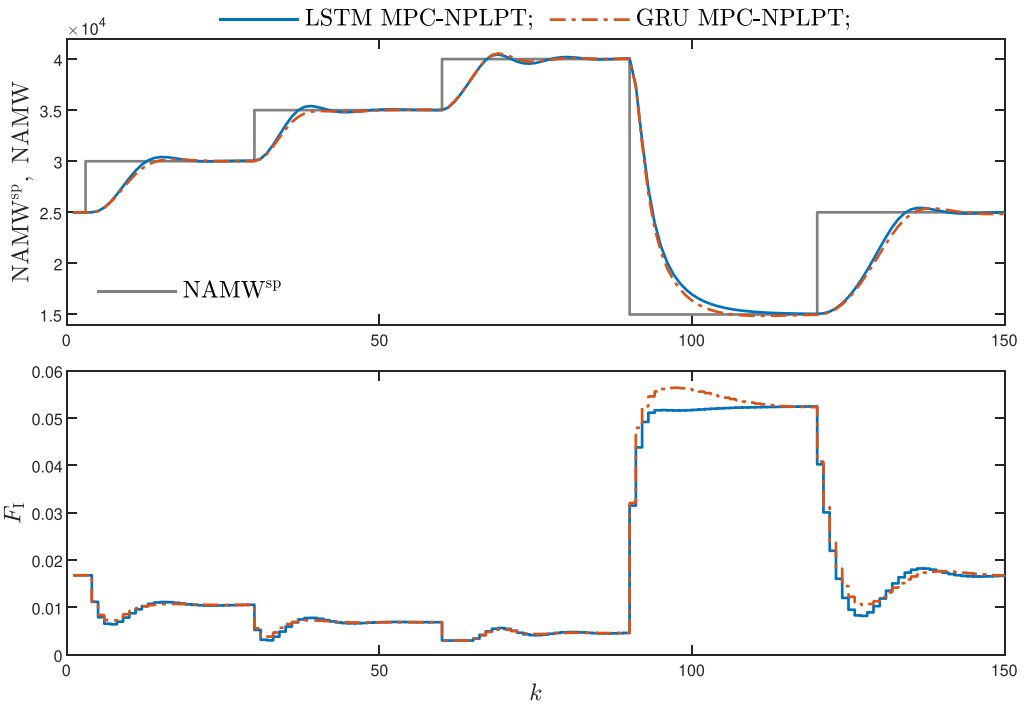

Fig. 12. The process 1: the process output and input signals for the MPC-NPLPT algorithm relying on the LSTM and GRU networks.

Fig. 13 depicts an analogous comparison for the neutralization reactor. The controller with the LSTM model has a larger overshoot at the sampling time $k = 10$, while at time $k = 40$, a larger overshoot occurs for the controller with the GRU model. The regulation time of both controllers is similar, but there are moments when the MPC with the LSTM model works faster (e.g., around the sampling instants $k = 90, \ldots, 100$).

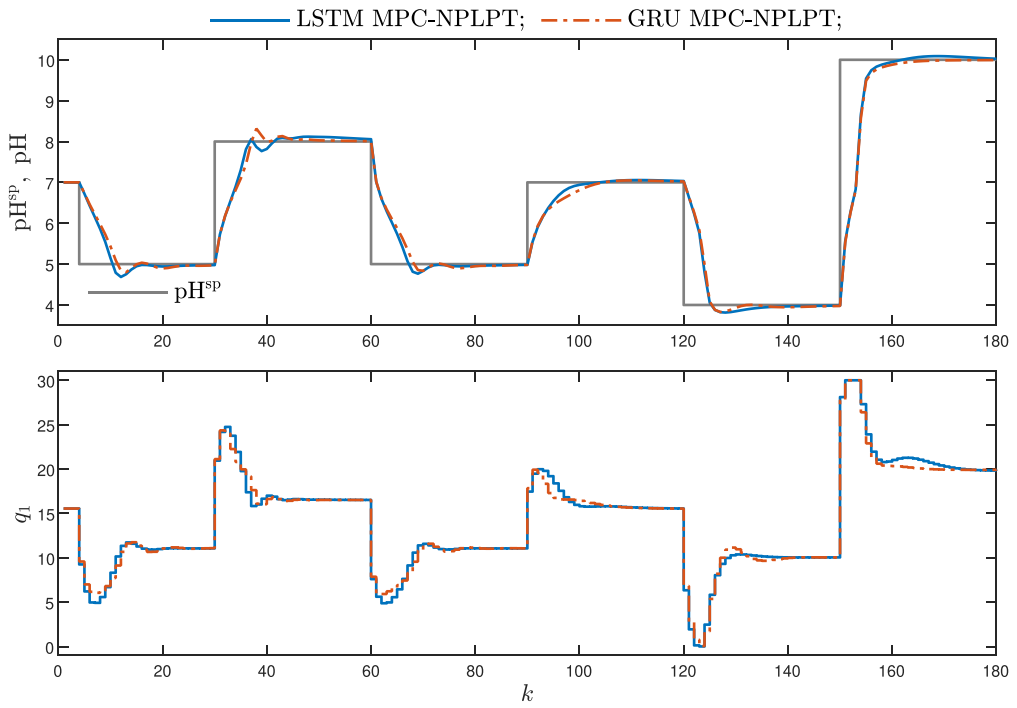

Fig. 13. The process 2: the process output and input signals for the MPC-NPLPT algorithm relying on the LSTM and GRU networks.

It is difficult to find a winner of this comparison; in some moments of the simulation, the controller with the LSTM model provides better regulation quality; in others, the one with the GRU model. In general, however, their quality is very similar and very good. To precisely compare the algorithms, let us analyze the Sum of Squared control Errors defined as $\text{SSE} = \sum_{k=1}^{k^{\max}} (y^{\text{sp}}(k) - y(k))^2$, where $k^{\max}$ is the number of iterations of the simulation. The obtained values are specified in Tables 1 and 2. Here, one can observe that the GRU model performs better for the polymerization process. Conversely, the LSTM model outperforms GRU for the pH reactor. Of note, the observed differences in control quality are not significant. One more observation can also be made. The MPC-NO and MPC-NPLPT controllers perform best out of all of the presented algorithms in all cases. MPC-NPLT1 and MPC-NPLT2 provide minimally worse quality but still perform very well. MPC-NPL has the largest errors out of all algorithms under study. Of course, the MPC-NO needs nonlinear optimization. Hence, the MPC-NPLPT algorithm is the best regarding the quality of control and calculation efficiency. Tables 1 and 2 also report the results for $n_A = 0$ and $n_A = 2$, the differences will be discussed in Section 6.3.6.

6.3.4. Comparison of Computation Time for Different MPC Algorithms

Online linearization aims to transform the nonlinear MPC optimization task into a quadratic one and, therefore, lessens the computational burden of calculating the optimal control signals. A series of experiments has been performed to check if this objective has been achieved. Tables 3 and 4 present the results. The experiments have been performed in MATLAB as follows: we simulated the process and a controller in a closed loop for each MPC type. The time necessary to find the optimal control signal value has been measured during simulations. This measurement includes the time required to complete trajectory linearization and execute the MPC optimization task. As those times may differ when the simulation is run on different hardware platforms, all results are presented as a percentage of execution times needed by MPC-NO, which is, of course, the most computationally demanding.

Table 1
The process 1: comparison of the control performance index SSE for different MPC algorithms relying on LSTM and GRU networks.

Algorithm	LSTM, $n_A = 0$	GRU, $n_A = 0$	LSTM, $n_A = 2$	GRU, $n_A = 2$
MPC-NPL	3.3202×10^9	4.4359×10^9	3.3371×10^9	3.4827×10^9
MPC-NPLT1	2.8043×10^9	2.7924×10^9	2.8857×10^9	2.8596×10^9
MPC-NPLT2	2.8271×10^9	2.7932×10^9	2.8857×10^9	2.9395×10^9
MPC-NPLPT	2.7712×10^9	2.7474×10^9	2.8116×10^9	2.7501×10^9
MPC-NO	2.7606×10^9	2.7205×10^9	2.8026×10^9	2.8138×10^9

Table 2
The process 2: comparison of the control performance index SSE for different MPC algorithms relying on LSTM and GRU networks.

Algorithm	LSTM, $n_A = 0$	GRU, $n_A = 0$	LSTM, $n_A = 2$	GRU, $n_A = 2$
MPC-NPL	1.8758×10^2	2.0088×10^2	1.7198×10^3	3.2882×10^3
MPC-NPLT1	1.7257×10^2	1.7476×10^2	1.7693×10^2	2.1061×10^2
MPC-NPLT2	1.7157×10^2	1.7447×10^2	1.7642×10^2	2.1708×10^2
MPC-NPLPT	1.6889×10^2	1.7312×10^2	1.7549×10^2	1.8254×10^2
MPC-NO	1.6901×10^2	1.7394×10^2	1.7551×10^2	1.9274×10^2

Table 3
The process 1: comparison of the computation time of different MPC algorithms relying on LSTM and GRU networks.

Algorithm	LSTM, $n_A = 0$	GRU, $n_A = 0$
MPC-NPL	8.56%	10.87%
MPC-NPLT1	14.83%	14.09%
MPC-NPLT2	15.00%	14.35%
MPC-NPLPT	15.82%	14.85%
MPC-NO	100%	85%

Table 4
The process 2: comparison of the computation time of different MPC algorithms relying on LSTM and GRU networks.

Algorithm	LSTM, $n_A = 0$	GRU, $n_A = 0$
MPC-NPL	7.25%	9.42%
MPC-NPLT1	8.27%	7.96%
MPC-NPLT2	8.42%	8.05%
MPC-NPLPT	14.51%	13.74%
MPC-NO	100%	94.42%

Table 3 presents the results for the polymerization reactor process. Firstly, one can observe that using GRU networks as a model helps slightly reduce the controller's computation burden for all controller types, except MPC-NPL. It can also be noticed that the execution time of MPC-NPL is the shortest. However, as mentioned earlier, the control quality achieved with this controller type is quite poor. Of the three well-performing controller types, the MPC-NPLT1 one is the fastest, having an execution time about 6.75 times shorter than the time required by MPC-NO. MPC-NPLT2 and MPC-NPL execute slightly slower, although their execution times are below 16% of those required by MPC-NO.

Results for the pH reactor are shown in Table 4. Also, in this case, using the GRU network as a model still helps to achieve better results. However, the differences in performance of both neural network types are smaller than in the previous example. On the other hand, larger differences can be observed when we consider the MPC type. MPC-NPLT1 and MPC-NPLT2 are very fast and require only about 8% of the time needed by MPC-NO. MPC-NPLPT is slower but still around seven times faster than MPC-NO.

Let us summarize the comparison calculation time for two considered processes shortly. Firstly, all three MPC algorithms with online linearization and quadratic optimization are a few times faster than the MPC-NO method. Secondly, the algorithms with one repetition of linearization and optimization at each sampling instant are clearly more efficient than the MPC-NPLPT method, which allows multiple linearization and optimization executions. Thirdly, using GRU models in trajectory linearization algorithms leads to slightly shorter calculation times than in the case of the LSTM models.

6.3.5. Comparison of Predicted Trajectories in Different MPC Algorithms

So far, we have compared process output and input signals for different MPC algorithms and when two classes of models, i.e., LSTM and GRU, are used. Now, let us present the reasons for the observed different control performance of the algorithms. To investigate this issue, this Section presents the predicted process output trajectories used by control algorithms to determine the control signal. The general rule is that the more precise the prediction generated by a model, the better control quality.

For the first benchmark process, the Fig. 14 presents predicted trajectories of the controlled variable within the prediction horizon, i.e. the values of $\hat{y}(k+1|k), \ldots, \hat{y}(k+10|k)$. The trajectories are shown for four example sampling instants: $k = 8, 30, 90, 120$. In all cases, the GRU model is used in MPC. In addition to the trajectories acquired in the compared controllers, the predicted trajectories of the simulated process are also given (they are computed from the same fundamental

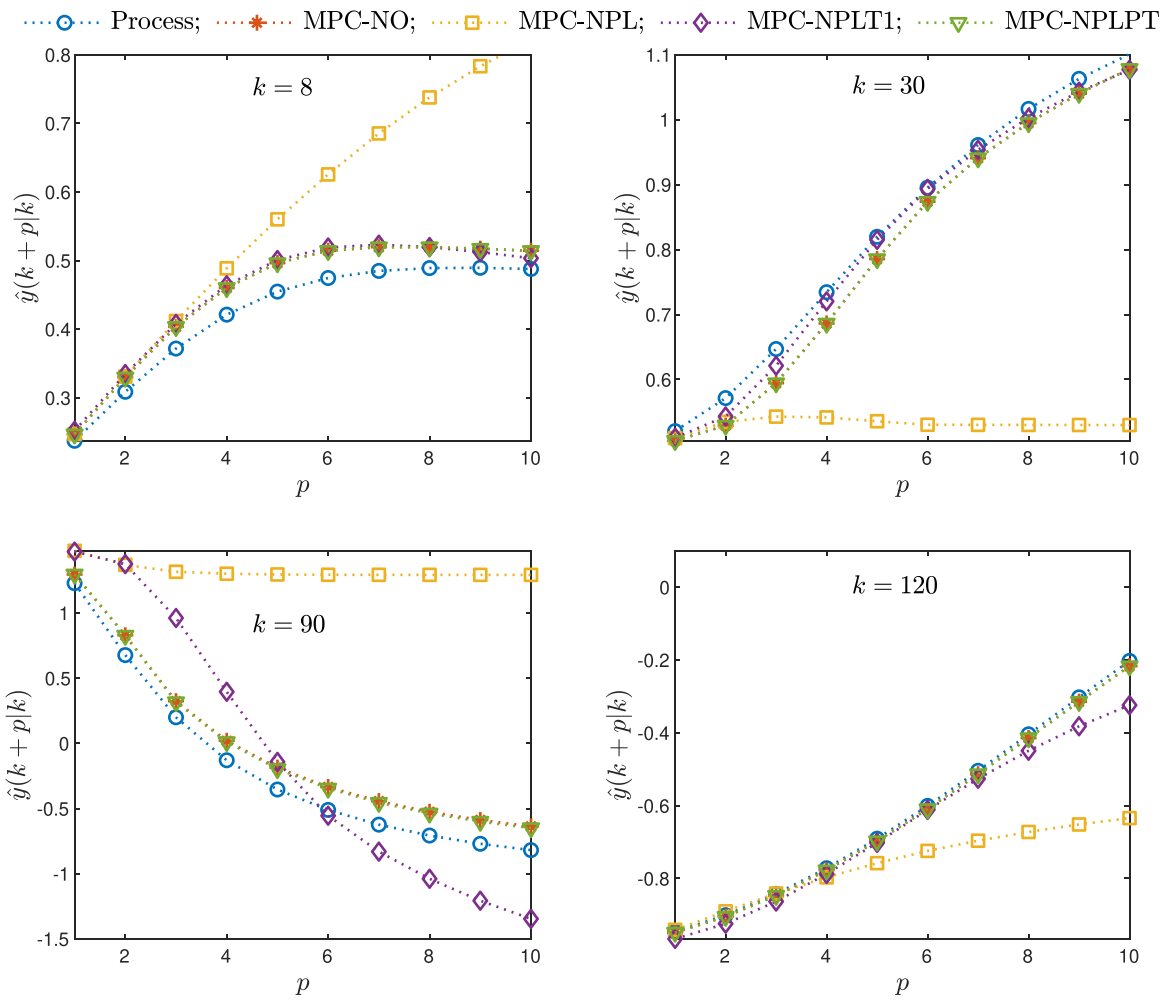

Fig. 14. The process 1: the process trajectory vs. the predicted trajectories in different MPC algorithms relying on the GRU network.

models that are used for process simulations [8,12]). The predictions produced by the MPC-NO algorithm, in which the nonlinear model is used without linearizations, are used as a proxy for prediction quality. The predictions in the MPC-NO algorithm are not the same as the process predictions since GRU models are not perfect; they all are characterized by some errors [47]. An important point to mention is that even the best predictions possible in MPC-NO sometimes deviate from the behavior of the actual process. It is particularly evident at $k = 8$ and $k = 90$. However, this does not prevent the MPC-NO, MPC-NPLT, and MPC-NPLPT controllers working correctly. The imperfections of predicted trajectories are compensated by the negative feedback and integral action contained in MPC.

We can observe that the predictions generated by MPC-NPL deviate from the actual process behavior and the prediction possible in the MPC-NO. This phenomenon is due to oversimplifications in the linearized model used as described in Section 4.2. Better prediction accuracy is produced by MPC-NPLT1 algorithm. At sampling time $k = 8$ and $k = 30$, these predictions practically do not differ from those possible in MPC-NO. A slight discrepancy occurs at $k = 120$, while more pronounced differences appear at $k = 90$. Most of the time, the linearized model used in MPC-NPLT1 is sufficient, but there are some moments where model errors become apparent. The MPC-NPLPT controller generates identical predictions to the MPC-NO controller for all simulations compared in Fig. 14.

For the second benchmark, Fig. 15 compares predicted output trajectories calculated using the GRU model in all compared MPC algorithms. The results confirm that MPC-NPL produces predictions significantly different from the process. The best performer is MPC-NPLPT, whose predictions are the same as those determined by MPC-NO. It can be observed that there is more difference between the process and MPC-NO predictions than in the case of the polymerization reactor. This issue relates to the nature of the process itself, which is more difficult to model. However, these dissimilarities are not so remarkable and the MPC-NO, as well as MPC-NPLT1 controllers, work correctly. Because predictions observed in the MPC-NPLT1 algorithm are quite good, good control quality is observed, as shown in Figs. 5, 7, 9 and 11. Interestingly, the

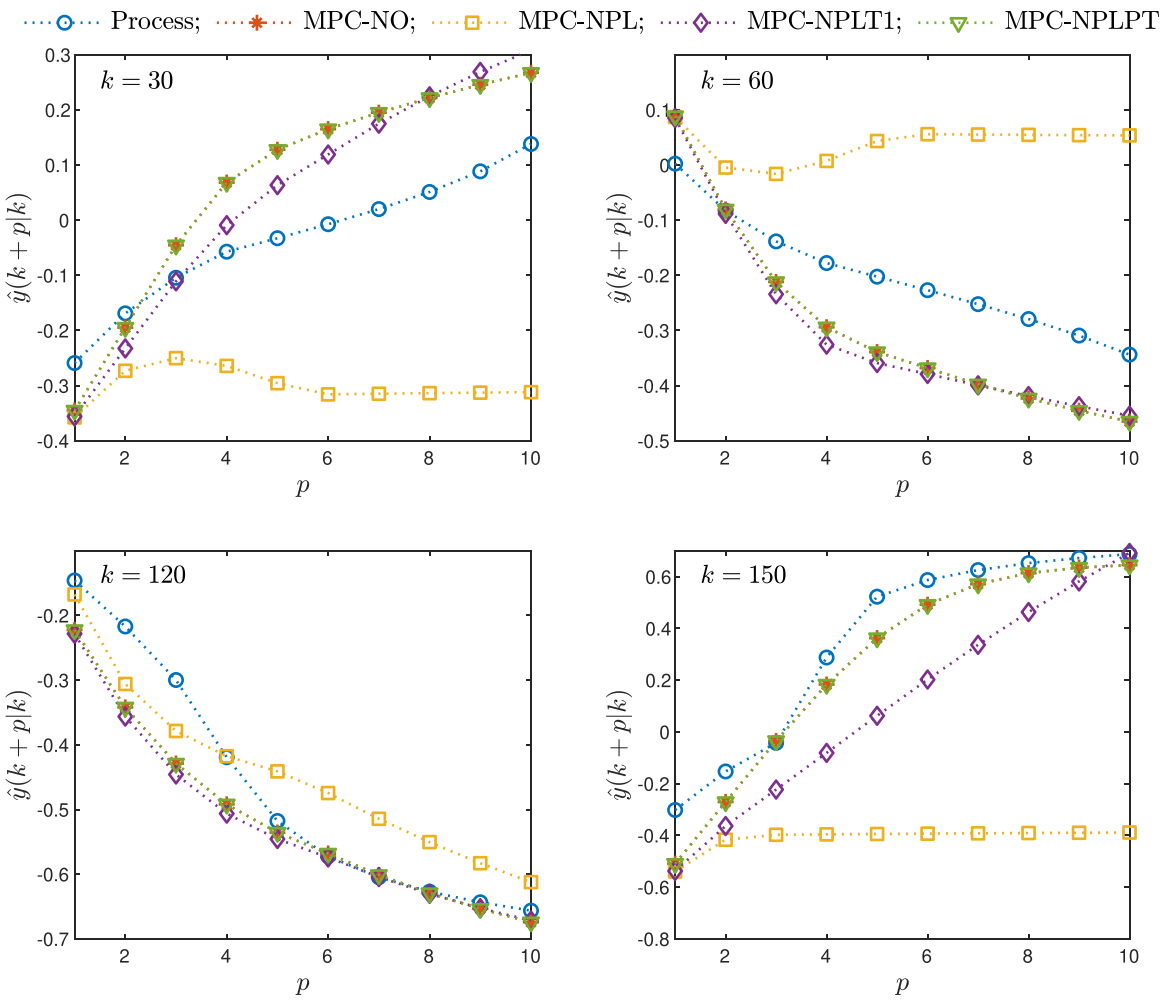

Fig. 15. The process 2: the process trajectory vs. the predicted trajectories in different MPC algorithms relying on the GRU network.

significant imperfections of predicted trajectories are very efficiently compensated by the negative feedback and integral action present in MPC. We do not intentionally present similar trajectories for the MPC-NPLT2 algorithm as they are also characterized by significant errors, such as those observed in the MPC-NPLT1. Finally, since predictions observed in the MPC-NPLPT algorithm are excellent, practically the same as in the MPC-NO algorithm, excellent control quality is observed, as shown in Figs. 4, 6, 8, 10.

6.3.6. The Impact of Model Order on Control Quality

One interesting issue remained to be discussed. All simulations discussed so far use LSTM and GRU models of the order of dynamics defined by $n_A = 0, n_B = 2$. On the contrary, in the literature, models with $n_A = n_B = 2$ are typically used for benchmarks under study. Therefore, several models of the order of dynamics defined by $n_A > 0$ have also been trained. However, after employing them in the MPC algorithm, it is found that they provide significantly worse control quality. For the first benchmark, the controllers with such models work correctly for most of the simulation time; unfortunately, they lead to strong oscillations around the set-point in some operation points. This phenomenon can be observed after the third set-point step for the first benchmark process. It occurs for both the LSTM model as shown in Fig. 16 and GRU model as depicted in Fig. 17. Additionally, for the GRU model, oscillations appear after the second set-point step and a large overshoot is observed for the last set-point step. Of note, performance of MPC-NPLPT controller when $n_A = 0$ and $n_B = 2$ is better.

For the second benchmark, on the other hand, unsatisfactory control performance when models of the order of dynamics defined by $n_A = n_B = 2$ are used, is observed after the second set-point change. For the LSTM model, the oscillations fade out after a moment which is shown in Fig. 18, while for the controller with the GRU model strong undying oscillations occur as depicted in Fig. 19. Controller quality when $n_A = 0$ and $n_B = 2$ is better.

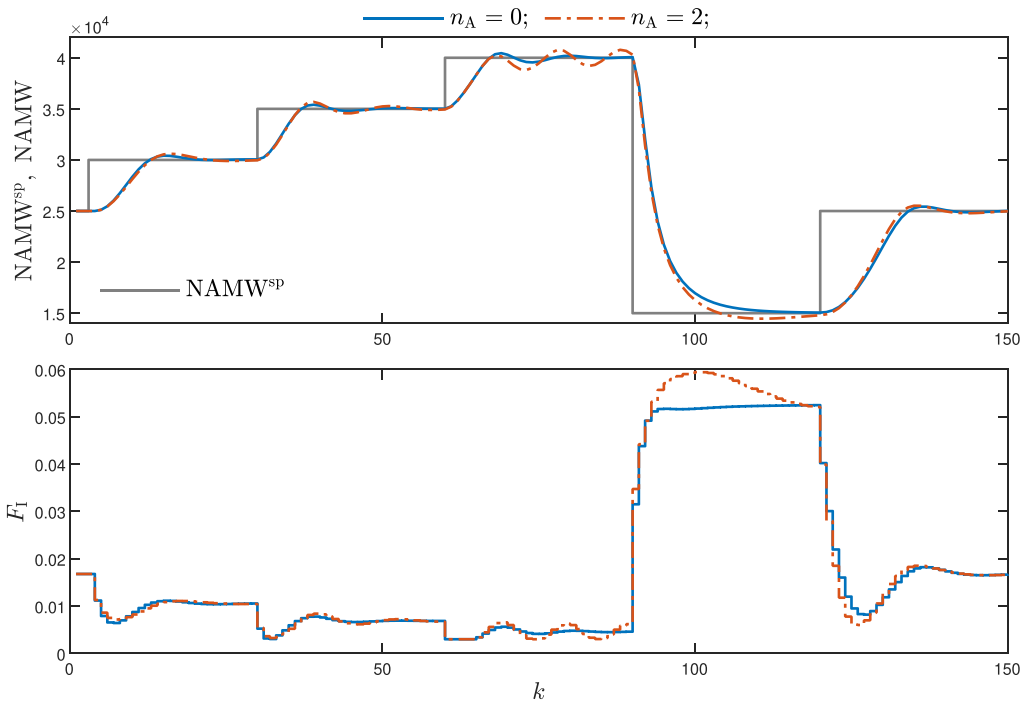

Fig. 16. The process 1: the process output and input signals for the MPC-NPLPT algorithm relying on the LSTM network with $n_A = 0$ and $n_A = 2$.

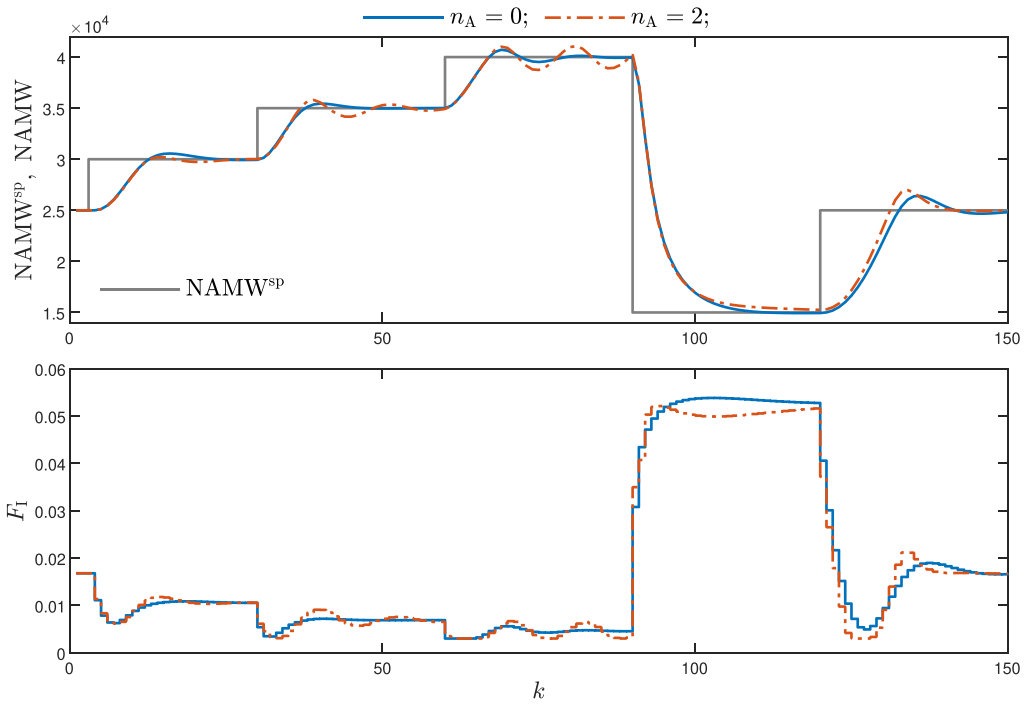

Fig. 17. The process 1: the process output and input signals for the MPC-NPLPT algorithm relying on the GRU network with $n_A = 0$ and $n_A = 2$.

This phenomenon occurs because the LSTM and GRU models are recurrent networks. The network has two inputs: input vector $x(k)$ and hidden state from previous time step $h(k-1)$

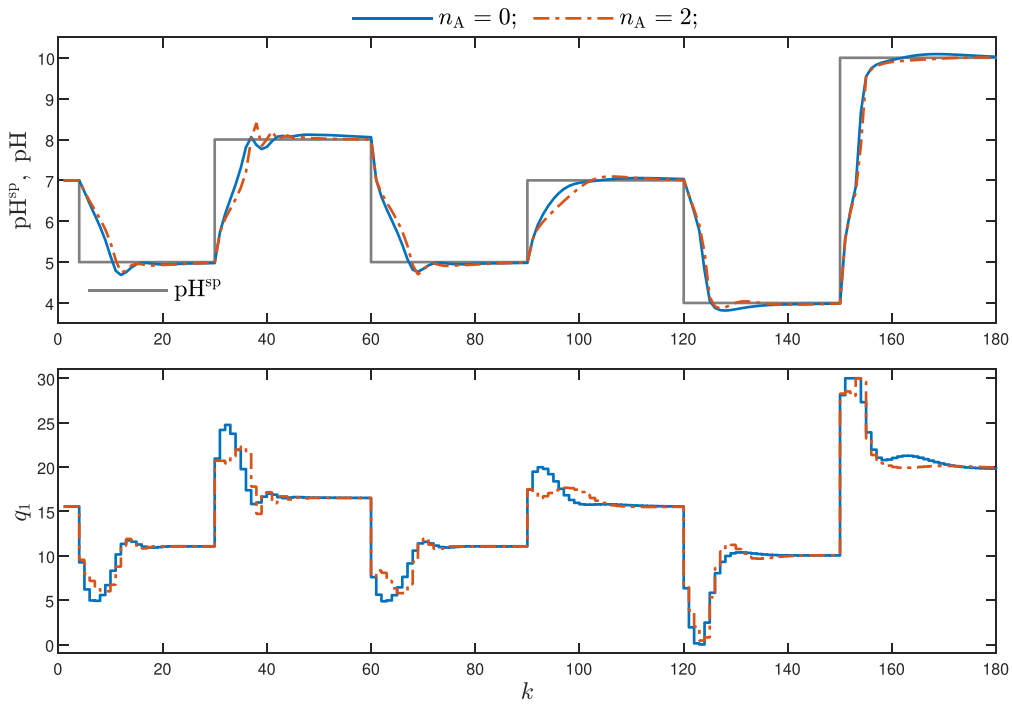

Fig. 18. The process 2: the process output and input signals for the MPC-NPLPT algorithm relying on the LSTM network with $n_A = 0$ and $n_A = 2$.

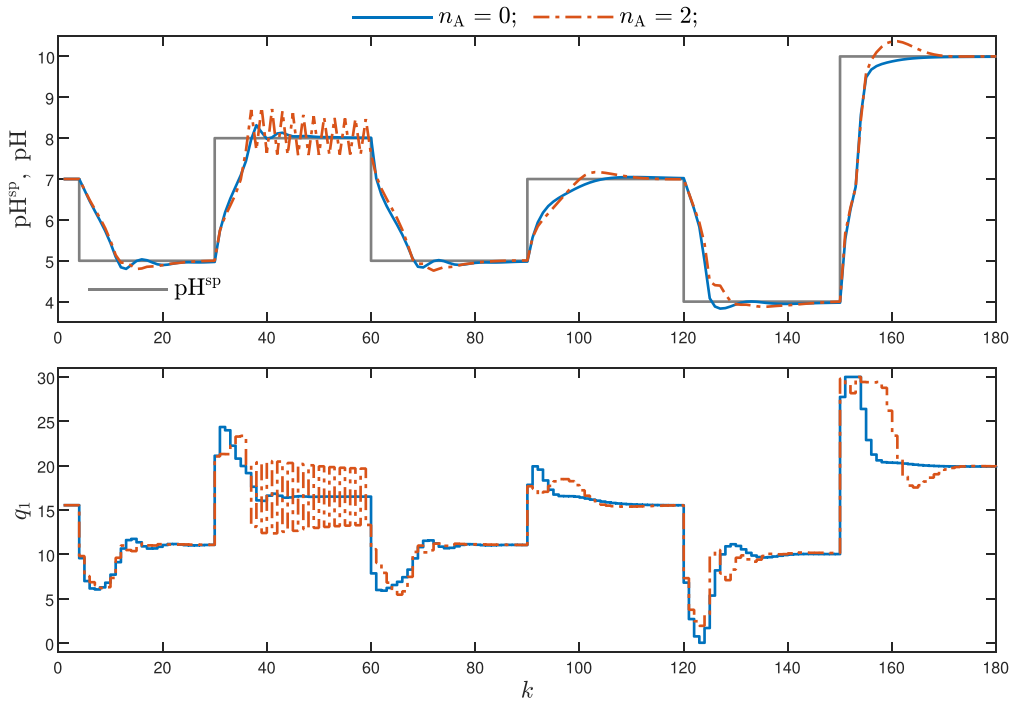

Fig. 19. The process 2: the process output and input signals for the MPC-NPLPT algorithm relying on the GRU network with $n_A = 0$ and $n_A = 2$.

$$y(k) = G(\boldsymbol{x}(k), h(k-1)) \tag{76}$$

Note, that the latter is in fact a function of control signal u and the output signal y as the arguments. Let consider the model of the order of dynamics specified by $n_A = n_B$. The input vector and hidden state are

$$x(k) = [u(k-1)\dots u(k-n_B)\ y(k-1)\dots y(k-n_A)]^T \tag{77}$$
$$h(k-1) = F_1(u(k-2),\dots,u(k-n_B-1), y(k-2),\dots,y(k-n_A-1), h(k-2)) \tag{78}$$

From Eq. (13), the network output can be also expressed

$$\begin{aligned} y(k-1) &= \boldsymbol{W}_y h(k-1) + b_y \\ y(k-2) &= \boldsymbol{W}_y h(k-2) + b_y \\ &\vdots \\ y(k-n_A) &= \boldsymbol{W}_y h(k-n_A) + b_y \end{aligned} \tag{79}$$

Combining Eqs. (77)–(79), we have

$$x(k) = \left[u(k-1)\dots u(k-n_B)\ \boldsymbol{W}_y h(k-1) + b_y \dots \boldsymbol{W}_y h(k-n_A) + b_y\right]^T \tag{80}$$
$$h(k-1) = F_1\left(u\left(k-2,\dots,u(k-n_B), \boldsymbol{W}_y h(k-2) + b_y, \dots, \boldsymbol{W}_y h(k-n_A) + b_y, h(k-2)\right)\right) \tag{81}$$

As seen above, the information contained in the past hidden state enters the network multiple times. This multiple recursion leads to more difficult network training process. The network tends to over-fit the training data. When the network is subsequently implemented in the MPC controller, the adverse effects of this over-fitting can be seen in some operation areas of the controller. On the contrary, when we consider a model with $n_A = 0$, we obtain

$$x(k) = [u(k-1)\dots u(k-n_B-1)]^T \tag{82}$$
$$h(k-1) = F_1(u(k-2,\dots,u(k-n_B-1), y(k-2),\dots,y(k-n_A), h(k-2)) \tag{83}$$

Here, the multiple recurrence problem does not exist. To sum up, using a hidden state ensures that the network is inherently recursive. Inserting y output signals into the x vector is hence discouraged.

7. Conclusions

Typically, the MPC algorithms for LSTM neural models use online nonlinear optimization executed at each sampling instant [19,21,35,47,39]. The same approach may be utilized for GRU models [3,22,47]. To reduce computational effort, MPC algorithms using successive model linearizations may be used; implementations for the LSTM model are considered in [32,36]. The findings of this work may be summarized in the following way:

1. Unfortunately, as clearly shown for two benchmarks, the MPC method with online model linearization gives: a) predicted trajectories far from those possible in the MPC based on nonlinear optimization, b) unsatisfactory control quality.
2. This work presents alternative formulations of MPC for dynamical processes modeled by LSTM and GRU neural networks. Complex linear estimation of the predicted trajectory is repeatedly found, which leads to simple to solve quadratic programs.
3. The presented MPC formulations yield predicted trajectories, and the resulting control quality is very close to those achievable in the MPC with nonlinear optimization. These results are the same for two considered benchmarks.
4. All MPC algorithms with online linearization and quadratic optimization are a few times faster than the MPC method with nonlinear optimization. The algorithms with one repetition of linearization and optimization at each sampling instant are more efficient than the method that allows multiple executions of linearization and optimization.
5. We detail implementation details for both LSTM and GRU models. Although in model training, the LSTM networks usually achieve better accuracy [47] when used for prediction in MPC, the differences between MPC based on these two classes of models are insignificant, as demonstrated in this work. Hence, for MPC, we recommend using the less complex GRU models in place of LSTM ones. The use of GRU models in trajectory linearization MPC algorithms leads to slightly shorter calculation times than in the case of the LSTM models.
6. For two considered benchmark processes, better control quality is obtained when the input vectors of LSTM and GRU models do not include past values of the output, i.e., the signals $y(k-1),\dots,y(k-n_A)$. We recommend to use the models which only use the signals $u(k-1),\dots,u(k-n_B)$ as model inputs. It is because LSTM and GRU models themselves are recurrent and the introduction of additional recurrence is discouraged.

The objective of this work is to present new sophisticated MPC algorithms for LSTM and GRU networks. The accuracy of prediction calculation and the resulting control quality of compared algorithms are thoroughly discussed for two simulated benchmark processes. In the future, it is planned to apply some of the algorithms to a real laboratory process.

CRediT authorship contribution statement

Krzysztof Zarzycki: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Visualization. **Maciej Ławryńczuk:** Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing, Visualization, Supervision, Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] K.M. Balla, J.T. Nørgaard, J.D. Bendtsen, C.S. Kallesøe, Model predictive control using linearized radial basis function neural models for water distribution networks, in: 2019 IEEE Conference on Control Technology and Applications (CCTA), Hong Kong, 2019, pp. 368–373.
[2] F.M. Bianchi, E. Maiorino, M.C. Kampffmeyer, A. Rizzi, R. Jenssen, Recurrent neural networks for short-term load forecasting: An overview and comparative analysis, in: Springer Briefs in Computer Science, Springer, Berlin, 2017.
[3] F. Bonassi, C.F. Oliveira da Silva, R. Scattolini, Nonlinear MPC for offset-free tracking of systems learned by GRU neural networks, IFAC-PapersOnLine 54 (2021) 54–59.
[4] W. Cao, Y. Zhang, J. She, M. Wu, Y. Cao, A dynamic subspace model for predicting burn-through point in iron sintering process, Information Sciences 466 (2018) 1–12.
[5] T. Capes, P. Coles, A. Conkie, L. Golipour, A. Hadjitarkhani, Q. Hu, N. Huddleston, M. Hunt, J. Li, M. Neeracher, K. Prahallad, T. Raitio, R. Rasipuram, G. Townsend, B. Williamson, D. Winarsky, Z. Wu, and H. Zhang. Siri on-device deep learning-guided unit selection text-to-speech system. In Proc. Interspeech 2017, Stockholm, Sweden, pages 4011–4015, 2017.
[6] B. Chen, T. Li, W. Ding, Detecting deepfake videos based on spatiotemporal attention and convolutional LSTM, Information Sciences 601 (2022) 58–70.
[7] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, in: NIPS 2014 Workshop on Deep Learning, 2014.
[8] F.J. Doyle, B.A. Ogunnaike, R.K. Pearson, Nonlinear model-based control using second-order Volterra models, Automatica 31 (1995) 697–714.
[9] Z. Feng, Y. Li, B. Sun, C. Yang, T. Huang, A multimode mechanism-guided product quality estimation approach for multi-rate industrial processes, Information Sciences 596 (2022) 489–500.
[10] J. Garcia-Tirado, J.P. Corbett, D. Boiroux, J.B. Jørgensen, M.D. Breton, Closed-loop control with unannounced exercise for adults with type 1 diabetes using the ensemble model predictive control, Journal of Process Control 80 (2019) 202–210.
[11] Z. Geng, G. Chen, Y. Han, G. Lu, F. Li, Semantic relation extraction using sequential and tree-structured LSTM with attention, Information Sciences 509 (2020) 183–192.
[12] J.C. Gómez, A. Jutan, E. Baeyens, Wiener model identification and predictive control of a pH neutralisation process, Proceedings of IEE, Part D, Control Theory and Applications 151 (2004) 329–338.
[13] A. Graves, M. Abdel-rahman, H. Geoffrey, Speech recognition with deep recurrent neural networks, in: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 2013, pp. 6645–6649.
[14] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, Advances in Neural Information Processing Systems, volume 21, pages 1–8, 2009.
[15] B. Hammer, Learning with Recurrent Neural Networks, Lecture Notes in Control and Information Sciences, volume 254, Springer, Berlin, 2000.
[16] H. Han, C. Chen, H. Sun, S. Du, J. Qiao, Multi-objective model predictive control with gradient eigenvector algorithm, Information Sciences 601 (2022) 114–128.
[17] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (1997) 1735–1780.
[18] R. Iglesias, F. Rossi, K. Wang, D. Hallac, J. Leskovec, M. Pavone, Data-driven model predictive control of autonomous mobility-on-demand systems, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 2018, pp. 6019–6025.
[19] B.K. Jeon, E.J. Kim, LSTM-based model predictive control for optimal temperature set-point planning, Sustainability 13 (2021) 894.
[20] A.F. Kamara, E. Chen, Z. Pan, An ensemble of a boosted hybrid of deep learning models and technical analysis for forecasting stock prices, Information Sciences 594 (2022) 1–19.
[21] D. Karimanzira, T. Rauschenbach, Deep learning based model predictive control for a reverse osmosis desalination plant, Journal of Applied Mathematics and Physics 8 (2020) 2713–2731.
[22] N. Lanzetti, Y.Z. Lian, A. Cortinovis, L. Domiguez, M. Mercangöz, C. Jones, Recurrent neural network based MPC for process industries, in: 2019 European Control Conference (ECC), Napoli, Italy, 2019, pp. 1005–1010.
[23] M. Ławryńczuk. Computationally Efficient Model Predictive Control Algorithms: a Neural Network Approach, volume 3 of Studies in Systems, Decision and Control. Springer, Cham, 2014.
[24] M. Ławryńczuk. Nonlinear Predictive Control Using Wiener Models: Computationally Efficient Approaches for Polynomial and Neural Structures, volume 389 of Studies in Systems, Decision and Control. Springer, Cham, 2022.
[25] M. Ławryńczuk, P. Tatjewski, Offset-free state-space nonlinear predictive control for Wiener systems, Information Sciences 511 (2020) 127–151.
[26] S. Li, Y. Zhang, Q. Zhu, Nash-optimization enhanced distributed model predictive control applied to the shell benchmark problem, Information Sciences 170 (2005) 329–349.
[27] P.F. Lima, G.C. Pereira, J. Mårtensson, B. Wahlberg, Experimental validation of model predictive control stability for autonomous driving, Control Engineering Practice 81 (2018) 244–255.
[28] D.P. Mandic, J.A. Chambers, Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability, Wiley, Chichester, 2001.
[29] M. Okulski, M. Ławryńczuk, A novel neural network model applied to modeling of a tandem-wing quadplane drone, IEEE Access 9 (2021) 14159–14178.
[30] G.A. Rovithakis, M.A. Christodoulou, Adaptive Control with Recurrent High-order Neural Networks, Springer, Berlin, 2000.
[31] T. Rybus, K. Seweryn, J.Z. Sasiadek, Application of predictive control for manipulator mounted on a satellite, Archives of Control Sciences 28 (2018) 105–118.
[32] B.B. Schwedersky, R.C.C. Flesch, H.A.S. Dangui, Practical nonlinear model predictive control algorithm for long short-term memory networks, IFAC-PapersOnLine 52 (2019) 468–473.
[33] X. Tao, N. Li, S. Li, Multiple model predictive control for large envelope flight of hypersonic vehicle systems, Information Sciences 328 (2016) 115–126.
[34] P. Tatjewski, Advanced control of industrial processes, structures and algorithms, Springer, London, 2007.
[35] E. Terzi, T. Bonetti, M. Farina, R. Scattolini, Learning-based model predictive control with long short-term memory networks, International Journal of Robust and Nonlinear Control 31 (2021) 8877–8896.

[36] E. Terzi, T. Bonetti, D. Saccani, M. Farina, L. Fagiano, R. Scattolini, Learning-based predictive control of the cooling system of a large business centre, Control Engineering Practice 97 (2020) 104348.
[37] J. Wang, An intelligent computer-aided approach for atrial fibrillation and atrial flutter signals classification using modified bidirectional LSTM network, Information Sciences 574 (2021) 320–332.
[38] Y. Wang, M. Wang, D. Wang, Y. Chang, Stochastic configuration network based cascade generalized predictive control of main steam temperature in power plants, Information Sciences 587 (2022) 123–141.
[39] W.C. Wong, E. Chee, J. Li, Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing, Mathematics 6 (2018) 242.
[40] Z. Wu, A. Alnajdi, Q. Gu, P.D. Christofides, Statistical machine-learning-based predictive control of uncertain nonlinear processes, AIChE Journal 68 (2022) e17642.
[41] Z. Wu, A. Tran, D. Rincon, P.D. Christofides, Machine learning-based predictive control of nonlinear processes, Part I: Theory. AIChE Journal 65 (2019) e16729.
[42] Z. Wu, A. Tran, D. Rincon, P.D. Christofides, Machine learning-based predictive control of nonlinear processes. Part II: Computational implementation, AIChE Journal 65 (2019) e16734.
[43] H. Xiao, D. Yu, C.L.P. Chen, Self-triggered-organized mecanum-wheeled robots consensus system using model predictive based protocol, Information Sciences 590 (2022) 45–59.
[44] H. Yang, L. He, Z. Zhang, R. Lu, C. Tan, Multiple-model predictive control for component content of CePr/Nd countercurrent extraction process, Information Sciences 360 (2016) 244–255.
[45] H. Yang, S. Ju, J. Zhang, H. Yuan, Model predictive control for cloud-integrated networked multiagent systems under bandwidth allocation, Information Sciences 500 (2019) 156–172.
[46] K. Zarzycki and Ławryńczuk. fast nonlinear model predictive control using LSTM networks: A model linearisation approach. In Proceedings of the 30th Mediterranean Conference on Control and Automation (MED), pages 1–6, Vouliagmeni, Greece, 2022.
[47] K. Zarzycki, M. Ławryńczuk, LSTM and GRU neural networks as models of dynamical processes used in predictive control: A comparison for two chemical reactors, Sensors 21 (2021) 5625.
[48] Y. Zheng, X. Wang, W. Zhe, Machine learning modeling and predictive control of the batch crystallization processblack, Industrial & Engineering Chemistry Research 61 (2022) 5578–5592.
[49] L. Zhou, Z. Zhang, L. Zhao, P. Yang, Attention-based BiLSTM models for personality recognition from user-generated content, Information Sciences 596 (2022) 460–471.

Krzysztof Zarzycki was born in Pruszków, Poland, in 1993. He received his B.Sc. degree in 2017 at the Faculty of Mechatronics, Warsaw University of Technology, and his M.Sc. degree in 2020 at the Faculty of Electronics and Information Technology, the same university, both in automatic control and robotics. He has been with the Institute of Control and Computation Engineering, Warsaw University of Technology, since 2020, where he is currently employed as an assistant pursuing his Ph.D. His scientific interests include algorithms for industrial process control, mainly advanced Model Predictive Control (MPC), and applications of artificial intelligence as a tool in process control and modeling.

Maciej Ławryńczuk was born in Warsaw, Poland, in 1972. He obtained his M.Sc. in 1998, Ph.D. in 2003, D.Sc. in 2013, in automatic control, from Warsaw University of Technology, Faculty of Electronics and Information Technology. Currently, he is employed at the same university at the Institute of Control and Computation Engineering as a professor (full). He is the author or a co-author of 7 books and more than 190 other publications, including more than 70 journal articles. His research interests include advanced control algorithms, particularly Model Predictive Control (MPC) algorithms, set-point optimization algorithms, soft computing methods, neural networks, modeling and simulation. Currently, Maciej Ławryńczuk is an associate editor of ISA Transactions (Elsevier), a member of the editorial board of International Journal of Applied Mathematics and Computer Science (De Gruyter), an academic editor of Electronics (MDPI) and Computational Intelligence and Neuroscience (Hindawi) journals.