sensors

MDPI

Article

Fast Nonlinear Predictive Control Using Classical and Parallel Wiener Models: A Comparison for a Neutralization Reactor Process

Robert Nebeluk * and Maciej Ławryńczuk

Institute of Control and Computation Engineering, Faculty of Electronics and Information Technology, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warsaw, Poland; maciej.lawrynczuk@pw.edu.pl
* Correspondence: robert.nebeluk@pw.edu.pl

Abstract: The Wiener model, composed of a linear dynamical block and a nonlinear static one connected in series, is frequently used for prediction in Model Predictive Control (MPC) algorithms. The parallel structure is an extension of the classical Wiener model; it is expected to offer better modeling accuracy and increase the MPC control quality. This work discusses the benefits of using the parallel Wiener model in MPC. It has three objectives. Firstly, it describes a fast MPC algorithm in which parallel Wiener models are used for online prediction. In the presented approach, sophisticated trajectory linearization is performed online, which leads to computationally fast quadratic optimization. The second objective of this work is to study the influence of the model structure on modeling accuracy. The well-known neutralization benchmark process is considered. It is shown that the parallel Wiener models in the open-loop mode generate significantly fewer errors than the classical structure. This work's third objective is to validate the efficiency of parallel Wiener models in closed-loop MPC. For the neutralization process, it is demonstrated that parallel models demonstrate better control quality using various indicators, but the difference between the classical and parallel models is not significant.

Keywords: model predictive control; wiener models; neutralization reactor

Citation: Nebeluk, R.; Ławryńczuk, M. Fast Nonlinear Predictive Control Using Classical and Parallel Wiener Models: A Comparison for a Neutralization Reactor Process. *Sensors* **2023**, *23*, 9539. https://doi.org/10.3390/s23239539

Academic Editor: Alessandro Casavola

Received: 15 September 2023
Revised: 27 November 2023
Accepted: 29 November 2023
Published: 30 November 2023

1. Introduction

Model Predictive Control (MPC) refers to an advanced control strategy in which a dynamical model of the considered process is utilized online to predict the future process state and an optimization procedure finds the best possible control action to minimize the predefined control quality index [1]. MPC algorithms have been used for years in process control; typical applications include chemical reactors [2], olefin metathesis processes [3], distillation towers [4] and power plants [5]. Nowadays, as a result of the availability of fast and relatively cheap hardware platforms necessary to carry out all online calculations, MPC algorithms are used in smart buildings [6] and several embedded systems; example applications include autonomous ground vehicle [7], autonomous driving vehicle [8], planning vehicle-parking trajectories for vertical parking spaces [9] and quadrotors [10,11]. Finally, MPC algorithms may control distributed parameter systems [12].

Two factors are essential for good control quality: precise online measurements provided by sensors and an accurate model of the controlled process. MPC algorithms utilize measurements of the process output variables (and state variables, in some cases). Significant measurement errors combined with an imprecise model result in poor predictions and, in consequence, unsatisfactory control performance. The importance of precise measurements is stressed in [13] where a wind disturbance preview is incorporated with an MPC algorithm to improve the resistance of Unmanned Aerial Vehicles during operation to wind gusts. However, observers are designed to solve this critical problem in many scenarios if there are not enough process data available. By estimating the values in such a way, a better control quality is achievable. A state observer scheme is proposed in [14]

for uninterruptible power supply applications and it is compared to classical approaches like with Kalman filters. The study presented in [15] shows an approach to deal with ocean environment disturbances by designing a nonlinear disturbance observer for unmanned surface vehicles to obtain safe and effective motion control performance. Lastly, in work [7], a Dual-Rate Extended Kalman filter is designed to obtain fast vehicle state estimation in the problem of real-time lane-keeping control for autonomous ground vehicles.

In this work, we study the impact of model structure and accuracy on the possible control performance. Although the general idea of MPC does not limit the model structure used for online prediction, the cascade Wiener model is frequently used [16]. The Wiener model consists of a linear dynamical block connected in series with a nonlinear static block. A great advantage of the Wiener model is the fact that it can efficiently approximate the properties of different processes using a limited number of parameters. Let us name a few examples reported in the literature: distillation columns [17], chemical reactors [18–20], gasifiers [21], chromato-graphic separation processes [22], fuel cells [23,24], photovoltaic cells [25], the relaxation processes during anesthesia [26], the arterial pulse transmission phenomena [27]. Additionally, due to the specialized structure of the Wiener model, we can derive a set of computationally efficient MPC algorithms in which fast quadratic optimization is used rather than complicated nonlinear programming [16].

Typically, the classical Wiener model structure is used in MPC [16], i.e., the model consists of one linear dynamical block and one nonlinear static block. A natural extension of the rudimentary Wiener structure uses a few classical sub-models connected in parallel. Such a model structure and identification issues are described in [28,29], while identification starting from linearized models is considered in [30]. The motivation to use the parallel structure is the following: the parallel model should be capable of generating better accuracy than the classical one. As a result, the parallel model is likely to offer better control quality when used in MPC compared to the classical model structure. Of course, this may be true for some processes, while for other ones, the classical structure may be sufficient.

This work has the following three objectives:

1. The first objective is to extend previous research in computationally efficient MPC algorithms in which Wiener models are used for prediction [16]. Namely, the goal is to detail a fast MPC method in which a linear approximation of the process predicted trajectory is successively obtained online using parallel Wiener models. As a result, the derived MPC algorithm requires relatively simple and fast quadratic optimization rather than a nonlinear approach.
2. The second objective of this work is to study the influence of the model structure on modeling accuracy. We compare the accuracy of the classical Wiener structure and that of the parallel Wiener models. In the latter case, the impact of the number of sub-models and the complexity of the nonlinear block are thoroughly evaluated. To the best of the authors' knowledge, a fair comparison between the classical and parallel Wiener models has not yet been presented in the literature.
3. The third objective of this work is to compare the efficiency of classical and parallel Wiener models in MPC. The problem is really important. Although more sophisticated models are likely to produce much better modeling accuracy in an open loop, the advantages of using complex models may be insignificant in MPC. Multi-criteria control quality assessment is used to demonstrate the impact of model structure.

The well-known neutralization benchmark process [19] is considered to verify the advantages of parallel Wiener models used in the open loop and MPC. Precise modeling and control of the neutralization benchmark process is essential in different areas, i.e., in chemical engineering, biotechnology and waste-water treatment industries [31]. Moreover, it is often utilized as a benchmark to assess the efficiency of new model structures and control algorithms, e.g., [16,32–38].

This work is structured as follows. Section 2 defines the structure of the classical Wiener model and its parallel variant, Section 3 derives and discusses the implementation

of the fast MPC algorithm for the parallel Wiener model, Section 4 thoroughly discusses simulation results and Section 5 summarizes the whole work.

2. Classical and Parallel Wiener Models

Let us start with the definition of the classical Wiener model [16,39]. In this work, we study Single-Input Single-Output (SISO) systems, i.e., we consider processes with one input and one output. The process input, which is also the manipulated variable in MPC, is denoted by u. The process output, which is the controlled variable in MPC, is denoted by y. Figure 1 shows the classical Wiener model consisting of a linear dynamic block and a nonlinear static block connected in series. Let us describe the model using mathematical formulas. We use the discrete-time description; k denotes the current sampling instant ($k = 0, 1, 2, \ldots$). The output signal of the linear block is

$$v(k) = \sum_{i=1}^{n_B} b_i u(k-i) - \sum_{j=1}^{n_A} a_j v(k-j), \tag{1}$$

where integer numbers A and B define the order of model dynamics while real numbers a_j and b_j stand for model coefficients. The output signal of the second block, which is also the output of the whole Wiener model, is a nonlinear static mapping

$$y(k) = f(v(k)). \tag{2}$$

Because we use online linearization of the predicted trajectory in MPC, we limit our considerations to differentiable functions f. In order to obtain precise models, we use neural networks with two layers, known to be universal approximators. Hence, the second block of the model is defined by

$$y(k) = f(v(k)) = w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi\left[w_{i,0}^1 + w_{i,1}^1 (v(k))\right]. \tag{3}$$

The first (hidden) layer is nonlinear; it has K hidden neurons and φ stands for the activation function, e.g., $\varphi = \tanh$. The second layer of the network is linear. The weights of the first layer are denoted by $w_{i,0}^1$ and $w_{i,1}^1$, while the parameters of the second layer are w_0^2 and w_i^2.

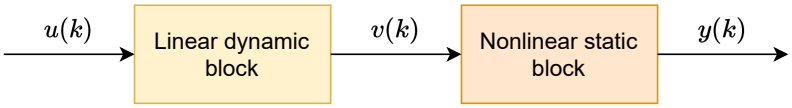

Figure 1. Classical Wiener model structure.

The general structure of the parallel Wiener model [28–30] is depicted in Figure 2. The model consists of n_g sub-models, also called branches, each of which has the classical Wiener structure. The model branches are connected in parallel; the outputs of the submodels are summarized. The outputs of the linear dynamic blocks are denoted by $v_1(k), \ldots, v_{n_g}(k)$ while the outputs of the nonlinear static blocks are denoted by $y_1(k), \ldots, y_{n_g}(k)$. Outputs of the consecutive linear blocks are calculated from the following formula

$$v_g(k) = \sum_{i=1}^{n_B} b_{i,g} u(k-i) - \sum_{j=1}^{n_A} a_{j,g} v_g(k-j), \tag{4}$$

where $g = 1, \ldots, n_g$. The nonlinear blocks use neural networks with two layers and are described as follows

$$y_g(k) = f_g(v_g(k)) = w_0^{2,g} + \sum_{i=1}^{K_g} w_i^{2,g} \varphi\left[w_{i,0}^{1,g} + w_{i,1}^{1,g} (v_g(k))\right], \tag{5}$$

where K is the number of hidden neurons. The output of the whole model is calculated from Equation (5) from

$$y(k) = \sum_{g=1}^{n_g} \left[w_0^{2,g} + \sum_{i=1}^{K_g} w_i^{2,g} \varphi \left[w_{i,0}^{1,g} + w_{i,1}^{1,g} (v_g(k)) \right] \right]. \tag{6}$$

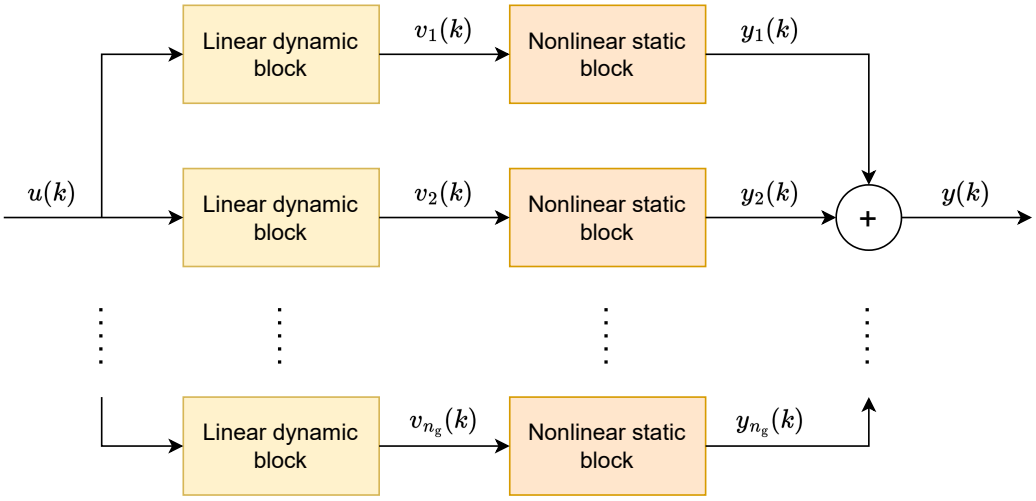

Figure 2. Parallel Wiener model structure.

3. Predictive Control Using Classical and Parallel Wiener Models

3.1. Preliminaries

At each discrete sampling instant of MPC, i.e., $k = 0, 1, 2, \ldots$, the algorithm calculates the whole decision vector, which consists of increments of the manipulated variable signal for the current and future instants,

$$\triangle \boldsymbol{u}(k) = \begin{bmatrix} \triangle u(k|k) \\ \vdots \\ \triangle u(k + N_{\mathrm{u}} - 1|k) \end{bmatrix}, \tag{7}$$

where the number of the calculated increments is defined by the control horizon denoted by N_{u}. At the current sampling instant, only the first element of the calculated vector is applied to the process, and calculations are repeated at the following instants. Let us recall the rudimentary MPC optimization task, [1,16]

$$\begin{aligned} &\min_{\triangle \boldsymbol{u}(k)} \left\{ J(k) = \sum_{p=1}^{N} \left(y^{\mathrm{sp}}(k+p|k) - \hat{y}(k+p|k) \right)^2 + \lambda \sum_{p=0}^{N_{\mathrm{u}}-1} \left(\triangle u(k+p|k) \right)^2 \right\}, \\ &\text{subject to} \\ &u^{\min} \leq u(k+p|k) \leq u^{\max}, \; p = 0, \ldots, N_{\mathrm{u}} - 1, \\ &\triangle u^{\min} \leq \triangle u(k+p|k) \leq \triangle u^{\max}, \; p = 0, \ldots, N_{\mathrm{u}} - 1, \\ &y^{\min} \leq \hat{y}(k+p|k) \leq y^{\max}, \; p = 0, \ldots, N - 1. \end{aligned} \tag{8}$$

The objective of MPC is to find online the decision variable vector, $\triangle \boldsymbol{u}(k)$, that minimizes the predefined cost function, $J(k)$, and satisfies all constraints. As far as the cost function is concerned, we consider predicted control errors, defined as differences between the setpoint trajectory, $y^{\mathrm{sp}}(k+p|k)$, and the predicted trajectory, $\hat{y}(k+p|k)$, which is found from the process model. As many as N predicted control errors are considered; N is called the prediction horizon. The second part of the cost function minimizes unwanted significant changes in the manipulated variable; λ stands for the penalty coefficient. In this

work, we consider classical MPC constraints, i.e., it is possible to consider limitations of the magnitude of the manipulated variable, the increments of that variable and the magnitude of the predicted value of the controlled variable.

3.2. Derivation of Fast MPC Algorithm

Let us note that as a result of model nonlinearity, predictions $\hat{y}(k+p|k)$ are nonlinear functions of the calculated MPC decision vector, $\triangle \boldsymbol{u}(k)$. It means that the MPC optimization task (8) is nonlinear, and a nonlinear solver is necessary at each sampling instant. This work adopts the MPC Algorithm with Nonlinear Prediction and Linearization along with the Predicted Trajectory (MPC-NPLPT) derived in [16] for the classical Wiener model. The MPC algorithm discussed next requires that the dynamical model used for prediction can be linearized online. It is true when the neural Wiener models described in Section 2 are differentiable. This assumption is fulfilled when activation function φ used in the nonlinear hidden nodes of the models' static blocks is differentiable. It is true for $\varphi = \tanh$.

Let us define the predicted output trajectory vector

$$\hat{\boldsymbol{y}}(k) = \begin{bmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+N|k) \end{bmatrix}. \tag{9}$$

The idea behind the MPC-NPLPT algorithm is to use a linear approximation of the predicted trajectory with respect to the decision vector, $\triangle \boldsymbol{u}(k)$. Trajectory linearization is performed along some predefined trajectory of the manipulated variable

$$\boldsymbol{u}^{\text{traj}}(k) = \begin{bmatrix} u^{\text{traj}}(k|k) \\ \vdots \\ u^{\text{traj}}(k+N_{\text{u}}-1|k) \end{bmatrix}. \tag{10}$$

Using the process model, we determine the predicted trajectory of the controlled variable that corresponds to the assumed trajectory $\boldsymbol{u}^{\text{traj}}(k)$

$$\hat{\boldsymbol{y}}^{\text{traj}}(k) = \begin{bmatrix} \hat{y}^{\text{traj}}(k+1|k) \\ \vdots \\ \hat{y}^{\text{traj}}(k+N|k) \end{bmatrix}. \tag{11}$$

In order to analytically derive trajectory $\hat{\boldsymbol{y}}^{\text{traj}}(k)$ over the whole prediction horizon, we have first to use Equation (4) to express the outputs of the first block of the model explicitly predicted for sampling instant $k+p$ at current instant k

$$v_g^{\text{traj}}(k+p|k) = \sum_{i=1}^{n_{\text{B}}} b_{i,g} u^{\text{traj}}(k-i+p|k) - \sum_{j=1}^{n_{\text{A}}} a_{j,g} v_g^{\text{traj}}(k-j+p|k). \tag{12}$$

Next, we use Equation (6) to express the outputs of the second block of the model, which is the model output. The predicted model output signal is

$$\hat{y}^{\text{traj}}(k+p|k) = \sum_{g=1}^{n_g} \left[w_0^{2,g} + \sum_{i=1}^{K_g} w_i^{2,g} \varphi \left[w_{i,0}^{1,g} + w_{i,1}^{1,g} (v_g^{\text{traj}}(k+p|k)) \right] \right] + d(k). \tag{13}$$

Because a model is never perfect, in prediction Rule (13), we supplement the model output by an estimated model error denoted by $d(k)$. It is determined straightforwardly as a difference between real (measured) process output denoted by $y(k)$ and model output

$$d(k) = y(k) - \sum_{g=1}^{n_g} \left[w_0^{2,g} + \sum_{i=1}^{K_g} w_i^{2,g} \varphi \left[w_{i,0}^{1,g} + w_{i,1}^{1,g} (v_g(k)) \right] \right]. \tag{14}$$

As thoroughly derived in [16], the linear approximation of the predicted trajectory of the process output is given by the following vector–matrix formula

$$\hat{\boldsymbol{y}}(k) = \boldsymbol{H}(k) \boldsymbol{J} \triangle \boldsymbol{u}(k) + \hat{\boldsymbol{y}}^{\text{traj}}(k) + \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\text{traj}}(k)). \tag{15}$$

The matrix of partial derivatives of the predicted output trajectory with respect to the input trajectory is of dimensionality $N \times N_{\text{u}}$ and has the following structure

$$\boldsymbol{H}(k) = \frac{\mathrm{d}\hat{\boldsymbol{y}}^{\text{traj}}(k)}{\mathrm{d}\boldsymbol{u}^{\text{traj}}(k)} = \begin{bmatrix} \frac{\partial \hat{y}^{\text{traj}}(k+1|k)}{\partial u^{\text{traj}}(k|k)} & \cdots & \frac{\partial \hat{y}^{\text{traj}}(k+1|k)}{\partial u^{\text{traj}}(k+N_{\text{u}}-1|k)} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}^{\text{traj}}(k+N|k)}{\partial u^{\text{traj}}(k|k)} & \cdots & \frac{\partial \hat{y}^{\text{traj}}(k+N|k)}{\partial u^{\text{traj}}(k+N_{\text{u}}-1|k)} \end{bmatrix}. \tag{16}$$

Let us now analytically derive entries of the matrix $\boldsymbol{H}(k)$ for the parallel Wiener shown in Figure 2. The partial derivatives are calculated differentiating Equation (13), which yields

$$\frac{\partial \hat{y}^{\text{traj}}(k+p|k)}{\partial u^{\text{traj}}(k+r|k)} = \sum_{g=1}^{n_g} \sum_{i=1}^{K_g} w_i^{2,g} \frac{\mathrm{d}\varphi(c_{i,g}^{\text{traj}}(k+p|k))}{\mathrm{d}c_{i,g}^{\text{traj}}(k+p|k)} \frac{\partial c_{i,g}^{\text{traj}}(k+p|k)}{\partial u^{\text{traj}}(k+r|k)}, \tag{17}$$

where predicted input signals of the first layer of neural networks used in the nonlinear static block of the Wiener model are

$$c_{i,g}^{\text{traj}}(k+p|k) = w_{i,0}^{1,g} + w_{i,1}^{1,g} (v_g^{\text{traj}}(k+p|k)). \tag{18}$$

If the hyperbolic tangent (tanh) function is used as the neural network activation function φ, we have

$$\frac{\mathrm{d}\varphi(c_{i,g}^{\text{traj}}(k+p|k))}{\mathrm{d}c_{i,g}^{\text{traj}}(k+p|k)} = 1 - (\varphi(c_{i,g}^{\text{traj}}(k+p|k)))^2. \tag{19}$$

Combining Equations (17) and (19), we obtain the general formula to determine the entries of matrix $\boldsymbol{H}(k)$

$$\frac{\partial \hat{y}^{\text{traj}}(k+p|k)}{\partial u^{\text{traj}}(k+r|k)} = \sum_{g=1}^{n_g} \sum_{i=1}^{K_g} w_{i,1}^{1,g} w_i^{2,g} \left(1 - (\varphi(c_{i,g}^{\text{traj}}(k+p|k)))^2\right) \frac{\partial v_g^{\text{traj}}(k+p|k)}{\partial u^{\text{traj}}(k+r|k)}. \tag{20}$$

Partial derivatives in the right-hand side of Equation (20) are also calculated analytically. For this purpose, we differentiate Equation (18). As far as the prediction for the first sampling instant of the prediction horizon is concerned, i.e., for sampling instant $k+1$, we obtain

$$\frac{\partial v_g^{\text{traj}}(k+1|k)}{\partial u^{\text{traj}}(k+r|k)} = \begin{cases} b_{1,g} & \text{for } r = 0 \\ 0 & \text{for } r > 0 \end{cases}. \tag{21}$$

It results in

$$\frac{\partial \hat{y}^{\text{traj}}(k+1|k)}{\partial u^{\text{traj}}(k+r|k)} = 0 \text{ for all } r > 0. \tag{22}$$

Similarly, for the prediction for the second sampling instant of the prediction horizon, i.e., for sampling instant $k+2$, we obtain

$$\frac{\partial v_g^{\text{traj}}(k+2|k)}{\partial u^{\text{traj}}(k+r|k)} = \sum_{i=1}^{n_B} b_{i,g} \frac{\partial u^{\text{traj}}(k-i+2|k)}{\partial u(k+r|k)} - \sum_{j=1}^{n_A} a_{j,g} \frac{\partial v_g^{\text{traj}}(k-j+2|k)}{\partial u(k+r|k)}. \tag{23}$$

Since the prediction horizon is typically longer than the control horizon, we have

$$\frac{\partial u^{\text{traj}}(k-i+p|k)}{\partial u^{\text{traj}}(k+r|k)} = \begin{cases} 1 & \text{for } (r=i \text{ and } r<p) \\ 0 & \text{otherwise} \end{cases}. \tag{24}$$

In general, for the prediction for sampling instant $k+p$, we obtain

$$\frac{\partial v_g^{\text{traj}}(k+p|k)}{\partial u^{\text{traj}}(k+r|k)} = \sum_{i=1}^{n_B} b_{i,g} \frac{\partial u^{\text{traj}}(k-i+p|k)}{\partial u(k+r|k)} - \sum_{j=1}^{n_A} a_{j,g} \frac{\partial v_g^{\text{traj}}(k-j+p|k)}{\partial u(k+r|k)}. \tag{25}$$

Let us stress that partial derivatives $\frac{\partial v_g^{\text{traj}}(k-j+p|k)}{\partial u(k+r|k)}$ necessary in Equation (25) are calculated recurrently. Namely, calculations are repeated for all combinations of $p = 1, \ldots, N$ and $r = 0, \ldots, N_u - 1$ to find all entries of matrix $\boldsymbol{H}(k)$.

The auxiliary matrix of dimensionality $N_u \times N_u$ used in Equation (15) has the following structure

$$\boldsymbol{J} = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \ldots & 1 \end{bmatrix}, \tag{26}$$

and the auxiliary vector of length N_u is

$$\boldsymbol{u}(k-1) = \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-1) \end{bmatrix}. \tag{27}$$

Using the linear approximation of the predicted output trajectory given by Equation (15), the general MPC optimization task (8) is transformed into the following quadratic optimization task

$$\begin{aligned} &\min_{\triangle \boldsymbol{u}(k)} \Big\{ J(k) = \big\| \boldsymbol{y}^{\text{sp}}(k) - \boldsymbol{H}(k)\boldsymbol{J}\triangle\boldsymbol{u}(k) - \hat{\boldsymbol{y}}^{\text{traj}}(k) \\ &\qquad - \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\text{traj}}(k)) \big\|^2 + \|\triangle\boldsymbol{u}(k)\|^2_{\boldsymbol{\Lambda}} \Big\}, \\ &\text{subject to} \\ &\boldsymbol{u}^{\min} \le \boldsymbol{J}\triangle\boldsymbol{u}(k) + \boldsymbol{u}(k-1) \le \boldsymbol{u}^{\max}, \\ &\triangle\boldsymbol{u}^{\min} \le \triangle\boldsymbol{u}(k) \le \triangle\boldsymbol{u}^{\max}, \\ &\boldsymbol{y}^{\min} \le \boldsymbol{H}(k)\boldsymbol{J}\triangle\boldsymbol{u}(k) + \hat{\boldsymbol{y}}^{\text{traj}}(k) + \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\text{traj}}(k)) \le \boldsymbol{y}^{\max}. \end{aligned} \tag{28}$$

The constraints are expressed using the following vectors of length N_u

$$\boldsymbol{u}^{\min} = \begin{bmatrix} u^{\min} \\ \vdots \\ u^{\min} \end{bmatrix}, \boldsymbol{u}^{\max} = \begin{bmatrix} u^{\max} \\ \vdots \\ u^{\max} \end{bmatrix}, \triangle\boldsymbol{u}^{\min} = \begin{bmatrix} \triangle u^{\min} \\ \vdots \\ \triangle u^{\min} \end{bmatrix}, \triangle\boldsymbol{u}^{\max} = \begin{bmatrix} \triangle u^{\max} \\ \vdots \\ \triangle u^{\max} \end{bmatrix} \tag{29}$$

and the vectors of length N

$$\boldsymbol{y}^{\min} = \begin{bmatrix} y^{\min} \\ \vdots \\ y^{\min} \end{bmatrix}, \ \boldsymbol{y}^{\max} = \begin{bmatrix} y^{\max} \\ \vdots \\ y^{\max} \end{bmatrix}. \tag{30}$$

The MPC-NPLPT algorithm repeats online trajectory linearization and quadratic optimization a few times at each sampling instant. Namely, the future input trajectory along which linearization is determined, i.e., $\boldsymbol{u}^{\text{traj}}(k)$ (Equation (10)), is initially set as the "tail" of the optimal control sequence found at the previous sampling instant, i.e., without its first element, $\triangle u(k|k)$. Quadratic programming task (28) is then solved. If the controlled variable of the process is close to a required setpoint, the first element of the optimized solution vector is applied to the process. If this condition is not fulfilled, the calculated decision vector is used to form trajectory $\boldsymbol{u}^{\text{traj}}(k)$; linearization is performed, followed by solving the quadratic optimization task. A few such repetitions may be used at each sampling instant. In practice, five repetitions are sufficient [16].

3.3. Classical Formulation of the MPC Quadratic Optimization Task

Let us consider the classical formulation of the quadratic optimization task

$$\begin{aligned} &\min_{\boldsymbol{x}(k)} \left\{ 0.5\boldsymbol{x}^{\mathrm{T}}(k)\boldsymbol{H}_{\mathrm{QP}}(k)\boldsymbol{x}(k) + \boldsymbol{f}_{\mathrm{QP}}^{\mathrm{T}}(k)\boldsymbol{x}(k) \right\}, \\ &\text{subject to} \\ &\boldsymbol{A}(k)\boldsymbol{x}(k) \leq \boldsymbol{B}(k), \\ &\boldsymbol{LB} \leq \boldsymbol{x}(k) \leq \boldsymbol{UB}, \end{aligned} \tag{31}$$

where $\boldsymbol{x}(k) = \triangle \boldsymbol{u}(k)$. From Equation (28), we derive the time-varying linear inequality constraints

$$\boldsymbol{A}(k) = \begin{bmatrix} -\boldsymbol{J} \\ \boldsymbol{J} \\ -\boldsymbol{H}(k)\boldsymbol{J} \\ \boldsymbol{H}(k)\boldsymbol{J} \end{bmatrix}, \ \boldsymbol{B}(k) = \begin{bmatrix} -\boldsymbol{u}^{\min} + \boldsymbol{u}(k-1) \\ \boldsymbol{u}^{\max} - \boldsymbol{u}(k-1) \\ -\boldsymbol{y}^{\min} + \hat{\boldsymbol{y}}^{\text{traj}}(k) + \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\text{traj}}(k)) \\ \boldsymbol{y}^{\max} - \hat{\boldsymbol{y}}^{\text{traj}}(k) - \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\text{traj}}(k)) \end{bmatrix} \tag{32}$$

while constant bounds are specified by

$$\boldsymbol{LB} = \triangle \boldsymbol{u}^{\min}, \ \boldsymbol{UB} = \triangle \boldsymbol{u}^{\max}. \tag{33}$$

Matrix $\boldsymbol{H}_{\mathrm{QP}}(k)$ is the second-order derivative of the cost function, $J(k)$, with respect to the decision variables, $\triangle \boldsymbol{u}(k)$. The first-order derivative is

$$\begin{aligned} \frac{\mathrm{d}J(k)}{\mathrm{d}\triangle \boldsymbol{u}(k)} &= -2\boldsymbol{J}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}(k)(\boldsymbol{y}^{\mathrm{sp}}(k) - \boldsymbol{H}(k)\boldsymbol{J}\triangle \boldsymbol{u}(k) - \hat{\boldsymbol{y}}^{\text{traj}}(k) - \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\text{traj}}(k))) \\ &\quad + 2\boldsymbol{\Lambda}\triangle \boldsymbol{u}(k) \\ &= 2(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{H}(k)\boldsymbol{J} + \boldsymbol{\Lambda})\triangle \boldsymbol{u}(k) \\ &\quad - 2\boldsymbol{J}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}(k)(\boldsymbol{y}^{\mathrm{sp}}(k) - \hat{\boldsymbol{y}}^{\text{traj}}(k) - \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\text{traj}}(k))), \end{aligned} \tag{34}$$

while the second-order derivative becomes

$$\boldsymbol{H}_{\mathrm{QP}}(k) = \frac{\mathrm{d}^2 J(k)}{\mathrm{d}(\triangle \boldsymbol{u}(k))^2} = 2(\boldsymbol{J}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{H}(k)\boldsymbol{J} + \boldsymbol{\Lambda}). \tag{35}$$

Vector $\boldsymbol{f}_{\mathrm{QP}}(k)$ is defined by the part of the first-order derivative (34) which is independent of vector $\triangle\boldsymbol{u}(k)$. We obtain

$$\boldsymbol{f}_{\mathrm{QP}}(k) = -2\boldsymbol{J}^{\mathrm{T}}\boldsymbol{H}^{\mathrm{T}}(k)(\boldsymbol{y}^{\mathrm{sp}}(k) - \hat{\boldsymbol{y}}^{\mathrm{traj}}(k) - \boldsymbol{H}(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{\mathrm{traj}}(k))). \tag{36}$$

4. Simulations

4.1. Neutralization Process Description

In this work, we consider a neutralization reactor benchmark process to validate and compare the efficiency of classical and parallel Wiener models for open-loop modeling purposes and in closed-loop MPC control. The fundamental model of this benchmark process is described in detail in [19]. It consists of two differential equations and one algebraic equation. The full model formulation is as follows

$$\frac{\mathrm{d}W_{\mathrm{a}}(t)}{\mathrm{d}t} = \frac{q_1(t)(W_{\mathrm{a}_1} - W_{\mathrm{a}}(t))}{V} + \frac{q_2(W_{\mathrm{a}_2} - W_{\mathrm{a}}(t))}{V} + \frac{q_3(W_{\mathrm{a}_3} - W_{\mathrm{a}}(t))}{V}, \tag{37}$$

$$\frac{\mathrm{d}W_{\mathrm{b}}(t)}{\mathrm{d}t} = \frac{q_1(t)(W_{\mathrm{b}_1} - W_{\mathrm{b}}(t))}{V} + \frac{q_2(W_{\mathrm{b}_2} - W_{\mathrm{b}}(t))}{V} + \frac{q_3(W_{\mathrm{b}_3} - W_{\mathrm{b}}(t))}{V}, \tag{38}$$

and

$$W_{\mathrm{a}}(t) + 10^{\mathrm{pH}(t)-14} - 10^{-\mathrm{pH}(t)} + W_{\mathrm{b}}(t)\frac{1 + 2 \times 10^{\mathrm{pH}(t)-K_2}}{1 + 10^{K_1-\mathrm{pH}(t)} + 10^{\mathrm{pH}(t)-K_2}} = 0. \tag{39}$$

State variables W_{a} and W_{b} are reaction invariants. The process manipulated variable is the base NaOH stream denoted as q_1, while the controlled variable is the pH of the product. The buffer flow rate q_2 and base flow rate q_3 remain constant. W_{a_1}, W_{a_2}, W_{a_3}, W_{b_1}, W_{b_2}, W_{b_3}, V, K_1 and K_2 are constants [19]. The fundamental model given above is utilized only for process simulation, while various Wiener models are used in MPC.

4.2. Model Identification and Validation

Two classes of Wiener models are considered: classical and parallel. Neural networks with two layers defined by Equation (5) are utilized in nonlinear static blocks in both models. We use two sets of data generated from the open-loop simulation of the fundamental model for model identification: training and validation data sets. The first set is used only to identify model parameters, while the second set is used to assess model accuracy. All models are found using the same identification procedure. It consists of the following steps:

1. Initialization of the identification procedure. The number of model branches n_{g} ($n_g = 1$ for the classical Wiener model), the number of hidden nodes in each nonlinear block $K_1, \ldots, K_{n_g}$, the order of dynamics of linear blocks (defined by integers n_{A} and n_{B}), the number of maximal optimization steps used during identification are defined. All model parameters, i.e., parameters of linear dynamical blocks and nonlinear static blocks, are initialized randomly.
2. A nonlinear optimization solver is used to calculate model parameters. The objective of optimization is to minimize the model error for the training data set defined as

$$E = \sum_{k=1}^{k_{\max}} \left(y^{\mathrm{mod}}(k) - y(k)\right)^2, \tag{40}$$

   where $y^{\mathrm{mod}}(k)$ and $y(k)$ are the model output value and the output value from the training data set, respectively, for the current sampling instant k; $k_{\max}$ is the number of available data samples. This work uses the Sequential Quadratic Programming (SQP) solver for nonlinear optimization. Model error for the training data set, denoted by E_{train}, is calculated when optimization is completed.
3. Model error for the validation data set, denoted by E_{val}, is also calculated.
4. Steps 1–4 are repeated a few times, which leads to finding a few models. Of course, initialization of model parameters may have an impact on model accuracy and it may

be necessary to repeat identification for the same structure. This is because gradient-based nonlinear optimization is used during identification. Nonlinear optimization may terminate at a shallow local minimum. The finally chosen model has the lowest validation error.

The flowchart of the model identification procedure is presented in Figure 3.

The above identification procedure is independently repeated for different model configurations. This work considers the influence of the number of branches in the parallel model and the number of hidden nodes in neural networks used in nonlinear static blocks. The second order of linear dynamic blocks is always used, i.e., $n_A = n_B = 2$. According to previous research [16,36,38], the second order of dynamics is sufficient for the considered process. Both training and validation data sets used in this work consist of 5000 data samples each.

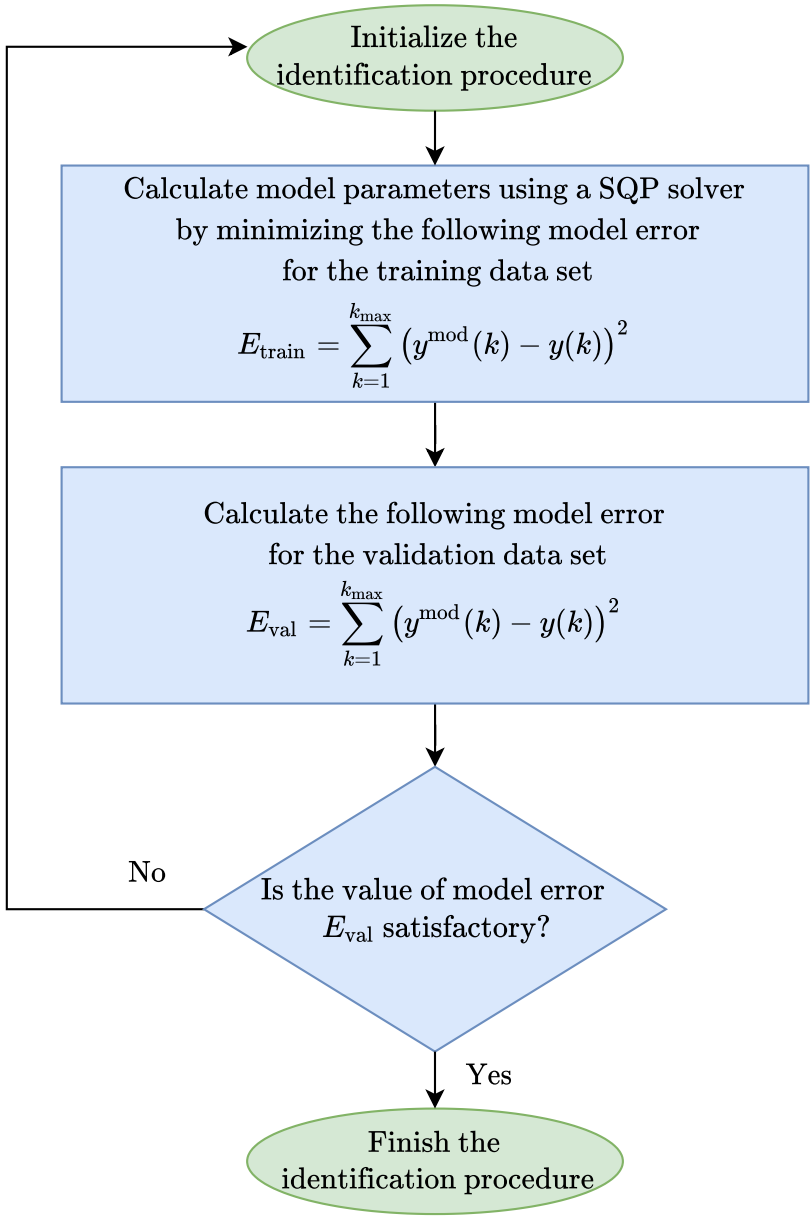

Figure 3. Graphical illustration of the Wiener model identification procedure.

Many classical and parallel Wiener models have been identified using the abovementioned procedure. We consider the classical Wiener model and parallel ones with two, three and four branches, i.e., $n_g = 1, \ldots, 4$. In each case, the number of hidden nodes in neural networks varies from one to five, i.e., $K_g = 1, \ldots, 5$. The activation function of hidden nodes is $\varphi = \tanh$. Table 1 shows the obtained numerical results of model errors. For each model structure, training and validation errors of the best model are shown, E_{train} and E_{val}, respectively. Moreover, the percentage relative validation error denoted as $E_{\text{val}}^{\text{relative}}$ is specified. It indicates how the validation error of a particular model compares to that of the best classical Wiener model, i.e., the model with five hidden nodes in the nonlinear block. Such a classical Wiener model has been considered in previous research [16]; using a greater number of nodes is discouraged as they do not lead to model improvement.

Firstly, we compare the results for parallel neural Wiener models with two branches, i.e., $n_g = 2$. We observe that the model with three hidden nodes, i.e., $K_g = 3$, results in the lowest relative validation error, equal to 40.29% of that possible for the classical Wiener model. Increasing the number of hidden nodes results in increasing the validation error. Secondly, we compare the results for parallel Wiener models with three branches. We observe that for one hidden node in both branches ($K_g = 1$), the relative validation error is greater than that for the classical model with five nodes. The best results are again obtained for three hidden nodes with the lowest relative validation error, equal to 39.56%. Increasing the number of hidden nodes increases the number of validation errors. Interestingly, the increase in the number of branches from two to three does not significantly improve model accuracy; both models with three hidden nodes practically have very similar errors. Finally, let us analyze parallel Wiener models with four branches, i.e., $K_g = 4$. Generally, all obtained models are much worse than the classical Wiener model. The best relative validation error equals 139.81% while the worst one is 3786.40%. For the considered benchmark process, four branches turn out to be unnecessary and badly influence model accuracy. Moreover, such models have multiple parameters and the nonlinear optimization procedure takes more time to find a reasonable solution than in the case of simpler model structures. We also verified parallel Wiener models with five branches and the results are even worse.

Table 1. Training and validation errors of classical and parallel Wiener models.

Model Type	n_g	K_g	E_{train}	E_{val}	$E_{\text{val}}^{\text{relative}}$
Classical neural Wiener	1	1	5.01×10^2	5.03×10^2	122.08%
	1	2	4.97×10^2	4.99×10^2	121.11%
	1	3	4.81×10^2	4.84×10^2	117.47%
	1	4	4.37×10^2	4.42×10^2	107.28%
	1	5	4.07×10^2	4.12×10^2	100.00%
Parallel neural Wiener	2	1	2.28×10^2	2.61×10^2	63.34%
	2	2	1.39×10^2	1.88×10^2	45.63%
	2	3	1.20×10^2	1.66×10^2	40.29%
	2	4	1.35×10^2	1.81×10^2	43.93%
	2	5	1.84×10^2	2.24×10^2	54.36%
	3	1	4.81×10^2	4.80×10^2	116.50%
	3	2	1.56×10^2	1.91×10^2	46.35%
	3	3	1.41×10^2	1.63×10^2	39.56%
	3	4	1.34×10^2	1.75×10^2	42.47%
	3	5	1.47×10^2	1.87×10^2	45.38%
	4	1	7.36×10^3	6.73×10^3	1633.50%
	4	2	1.64×10^3	1.65×10^3	400.49%
	4	3	6.19×10^2	6.57×10^2	159.47%
	4	4	5.78×10^2	5.76×10^2	139.81%
	4	5	1.53×10^4	1.56×10^4	3786.40%

Let us compare some of the obtained models graphically. It shows how they try to mimic the process represented by the validation data set. Figure 4 presents the results for the classical Wiener model with five hidden nodes in the nonlinear static block ($K = 5$). The top panel compares the first 1000 samples of the validation data set vs. the model output. The bottom panel shows the relationship between the whole validation data set and the model output. In general, we can see that the rudimentary Wiener model is quite precise. Hence, whether and to what extent the parallel structure can increase the model accuracy is interesting.

Figure 5 shows the efficiency of the parallel Wiener model with two branches, each of which has three hidden nodes ($n_g = 2$, $K_1 = K_2 = 3$); Figure 6 shows the efficiency of the parallel Wiener model with three branches, each of which has three hidden nodes ($n_g = 3$, $K_1 = K_2 = K_3 = 3$). We observe that these models have better accuracy than the classical Wiener model. The second one, i.e., the model with three branches, is slightly better. Figure 7 shows the efficiency of the parallel Wiener model with four branches, each of which has four hidden nodes ($n_g = 4$, $K_1 = K_2 = K_3 = K_4 = 4$). Unfortunately, although the model is the best among all models with four branches, it is noticeably worse than the classical model and parallel models with two and three branches. Finally, Figure 8 shows the efficiency of the parallel Wiener model with four branches, each of which has five hidden nodes ($n_g = 4$, $K_1 = K_2 = K_3 = K_4 = 5$). In this case, due to overparameterization, the model is very imprecise.

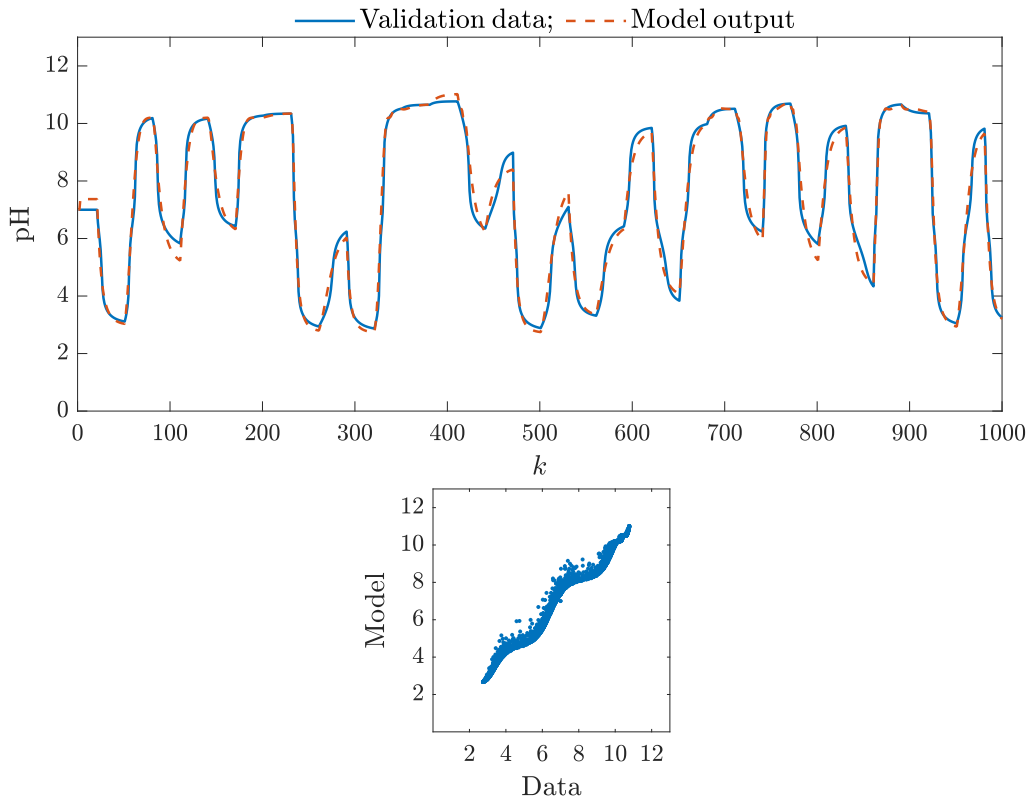

Figure 4. The classical Wiener model with five hidden nodes in the nonlinear static block ($K = 5$): the first 1000 samples of the validation data set vs. the model output (**top**), the relationship between the whole validation data set and the model output (**bottom**).

All things considered, parallel Wiener models with two branches make it possible to obtain an error as low as 40% of that observed when the classical Wiener model is used. A slight improvement is provided by models with three parallel branches, while more complex models increase the error due to overparameterization.

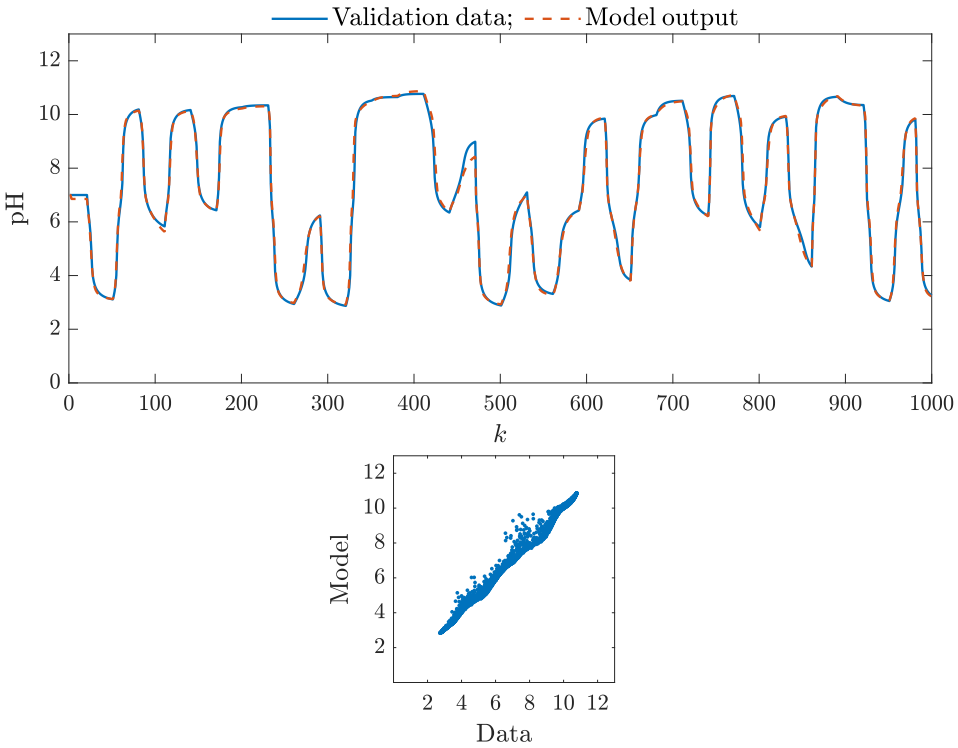

Figure 5. The parallel Wiener model with two branches, each of which has three hidden nodes ($n_g = 2$, $K_1 = K_2 = 3$): the first 1000 samples of the validation data set vs. the model output (**top**), the relationship between the whole validation data set and the model output (**bottom**).

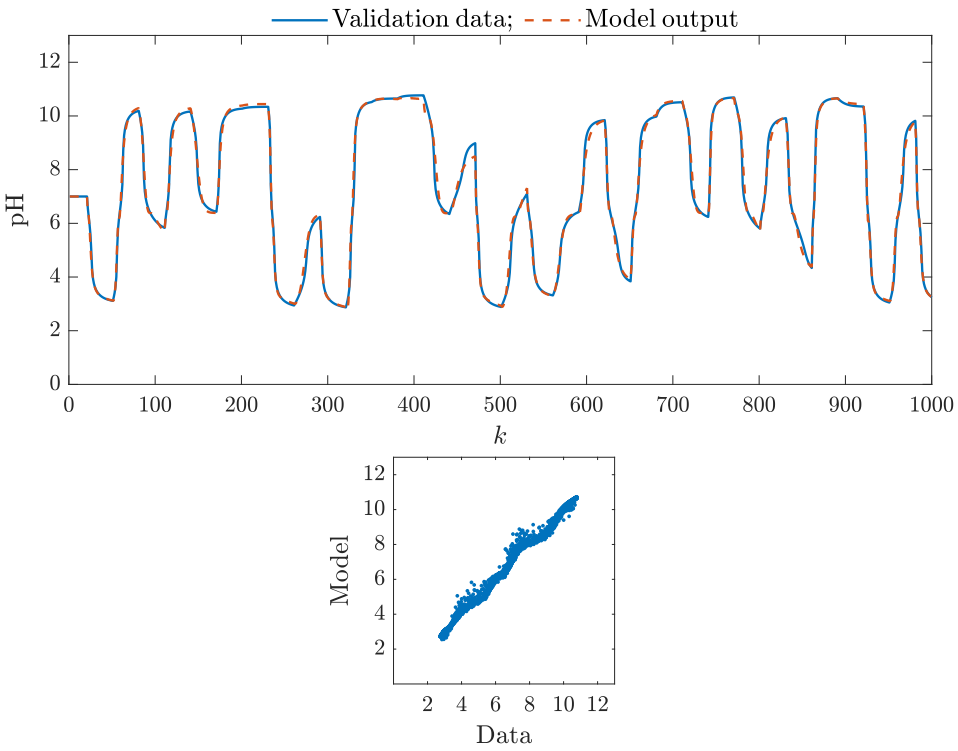

Figure 6. The parallel Wiener model with three branches, each of which has three hidden nodes ($n_g = 3$, $K_1 = K_2 = K_3 = 3$): the first 1000 samples of the validation data set vs. the model output (**top**), the relationship between the whole validation data set and the model output (**bottom**).

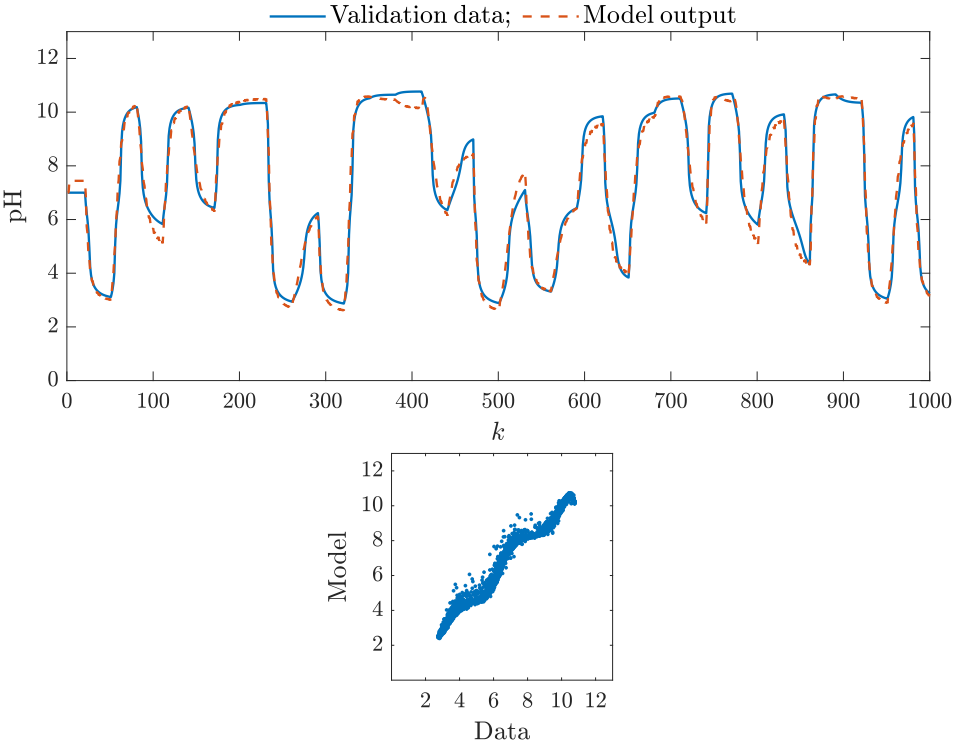

Figure 7. The parallel Wiener model with four branches, each of which has four hidden nodes ($n_g = 4$, $K_1 = K_2 = K_3 = K_4 = 4$): the first 1000 samples of the validation data set vs. the model output (**top**), the relationship between the whole validation data set and the model output (**bottom**).

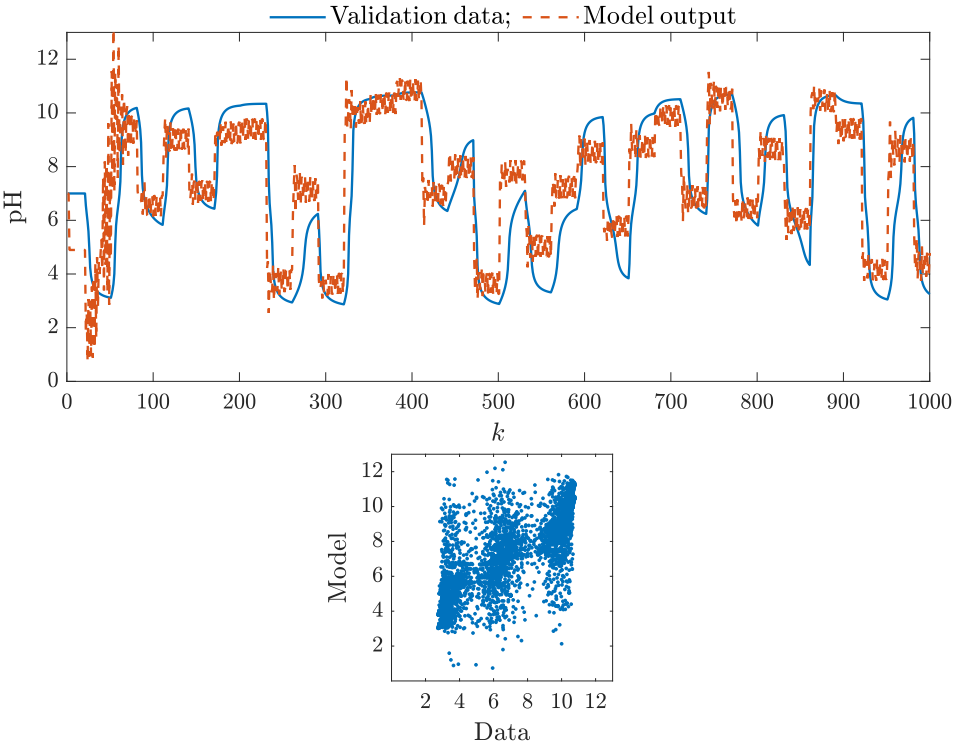

Figure 8. The parallel Wiener model with four branches, each of which has five hidden nodes ($n_g = 4$, $K_1 = K_2 = K_3 = K_4 = 5$): the first 1000 samples of the validation data set vs. the model output (**top**), the relationship between the whole validation data set and the model output (**bottom**).

4.3. Predictive Control of the Neutralization Process

Having found a set of Wiener models and compared them in an open loop, evaluating how they perform in closed-loop MPC control is interesting. In MPC algorithms, we mainly use the classical neural Wiener model with five hidden nodes and the best parallel neural Wiener model with three branches, each of which has three hidden nodes. We also use more complicated models. We consider two MPC algorithms: the discussed MPC-NPLPT algorithm with online linearization and quadratic optimization and the general MPC scheme with Nonlinear Optimization (MPC-NO). The latter uses nonlinear models for prediction, meaning a nonlinear optimization task must be solved at each sampling instant online. We want to obtain the performance of our computationally efficient MPC-NPLPT scheme as close to that of MPC-NO as possible. The following parameters are used in two considered MPC algorithms: $N = 10$, $N_u = 3$ and $\lambda = 0.25$ [16].

This work performs a multicriterial control quality assessment of MPC algorithms. For this purpose, we evaluate the control quality using the following statistical indices: the Mean Squared Error (MSE), the Mean Absolute Error (MAE), the Gauss standard deviation (σ_G), the Huber standard deviation (σ_H), the scale factor of the alpha-stable distribution (γ) and the rational entropy (H_R). The obtained numerical values of these indicators are presented in Table 2 and the calculation times necessary by MPC algorithms are given in Table 3. We consider MPC-NO and MPC-NPLPT algorithms for classical and the chosen parallel Wiener models. We can formulate the following observations:

1. The control quality indicators obtained for the MPC-NPLPT algorithm are practically the same as those for the MPC-NO control method. That means that our control algorithm is very efficient. Advanced online trajectory linearization makes it possible to use simple quadratic optimization; nonlinear programming is unnecessary. This observation can also be verified when we consider process time trajectories. Figure 9 compares simulation results of MPC-NO and MPC-NPLPT algorithms; both of them use the classical Wiener model. The controlled variable and the setpoint trajectory are displayed in the top panel. The manipulated variable is shown in the bottom panel. Although they use a completely different computational scheme, we can see that both algorithms' trajectories are very close. The same observations can be noted from Figure 10, which compares simulation results of MPC-NO and MPC-NPLPT algorithms, but now both algorithms use the parallel Wiener model with three branches.
2. From Table 2, we can find out that better control quality is achieved when MPC algorithms use the parallel Wiener model rather than the classical structure. The following indices are significantly reduced when the parallel model is used: MAE, σ_H, γ and rational entropy (H_R). The rest of the indices (MSE and σ_G) are slightly lower. Figure 11 presents the obtained trajectories possible when the same control algorithm MPC-NPLPT is used, but classical and parallel Wiener models are used for prediction. We can clearly see that the parallel model control scheme offers better control quality. Namely, the settling time is shorter and the overshoot is smaller.
3. Of course, increasing the number of model branches is likely to increase the computation time. Therefore, Wiener models with as few branches as possible should be used. Table 3 details calculation times of studied MPC algorithms for classical and parallel Wiener models. As all simulations are performed in MATLAB (not in a real industrial control system), we are interested in a relative comparison between the studied algorithms. Hence, all results are scaled so that the calculation time for the computationally demanding MPC-NO algorithm based on the classical Wiener model is assumed to be equal to 100%. It is interesting to note that increasing the number of branches significantly influences the calculation time of the MPC-NO algorithm with nonlinear optimization. On the other hand, the time required by the MPC-NPLPT algorithm developed and recommended in our work is significantly shorter and not influenced by the number of model branches. It is because the MPC-NPLPT quadratic optimization problem has a predominant influence on calculation time.

Table 2. Multi-criteria control quality indicators of MPC algorithms with classical and parallel Wiener models.

Model Type	MPC Algorithm	MSE	MAE	σ_G	σ_H	γ	H_R
Classical neural Wiener	MPC-NO	1.4375×10^{0}	5.3123×10^{-1}	1.2007×10^{0}	1.1652×10^{-1}	4.9899×10^{-2}	4.4580×10^{-1}
	MPC-NPLPT	1.4375×10^{0}	5.3123×10^{-1}	1.2007×10^{0}	1.1649×10^{-1}	4.9887×10^{-2}	4.4580×10^{-1}
Parallel neural Wiener, $n_g = 3$	MPC-NO	1.4329×10^{0}	5.0995×10^{-1}	1.1971×10^{0}	7.8351×10^{-2}	3.7433×10^{-2}	4.1628×10^{-1}
	MPC-NPLPT	1.4332×10^{0}	5.0992×10^{-1}	1.1972×10^{0}	7.8945×10^{-2}	3.7596×10^{-2}	4.1060×10^{-1}

Table 3. Calculation times for MPC algorithms with classical and parallel Wiener models.

Model Type	MPC Algorithm	Time
Classical neural Wiener	MPC-NO	100.00%
	MPC-NPLPT	42.01%
Parallel neural Wiener, $n_g = 3$	MPC-NO	146.01%
	MPC-NPLPT	56.01%

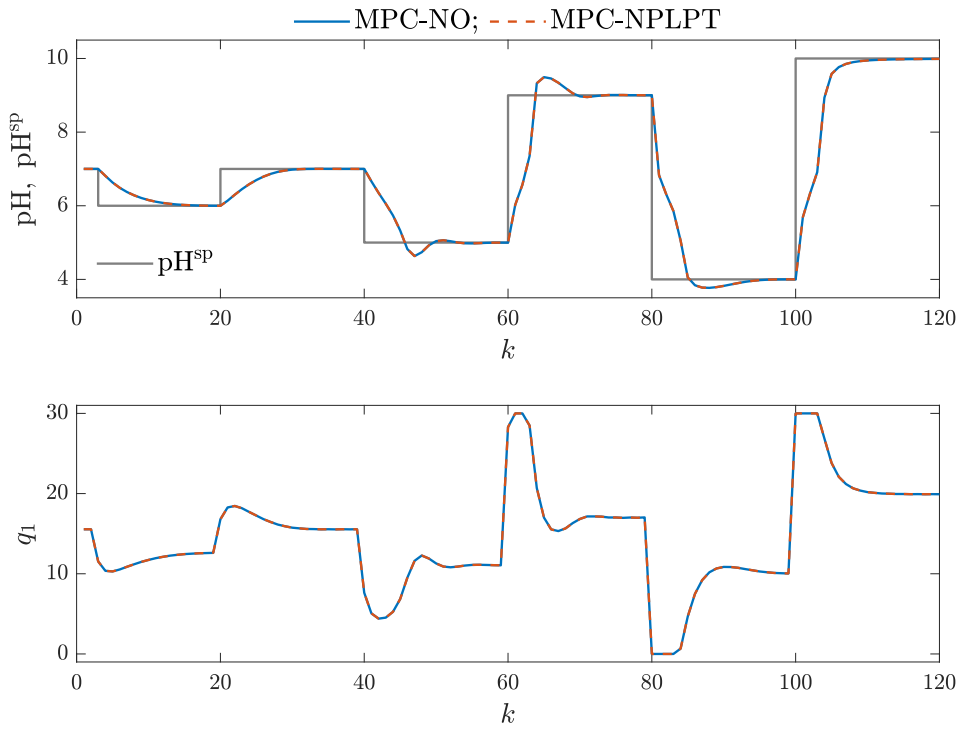

Figure 9. Simulation results: MPC-NO vs. MPC-NPLPT algorithms; both algorithms use the classical Wiener model.

It is interesting whether more complicated parallel Wiener models may be used in MPC. From Table 1 and Figures 7 and 8, we can see that increasing the number of model branches does not lead to improving open-loop model accuracy. As far as closed-loop model performance is concerned, let us consider Figure 12, which shows simulation results of MPC-NO and MPC-NPLPT algorithms that use the parallel Wiener model with $n_g = 4$ branches and neural networks with $K_g = 4$ hidden nodes. This is the best model among all models with four branches. Both algorithms produce the same trajectories, which is good because it means that our MPC-NPLPT algorithm perfectly mimics the computationally demanding MPC-NO method. Unfortunately, the control quality is generally much worse than in the case of parallel Wiener models with three branches. The manipulated variable has an oscillatory behavior, resulting in the controlled variable oscillating. Such an unwanted phenomenon occurs when the controlled variable value is close to the current set point value. Similarly, Figure 13 compares the same MPC algorithms, but now both algorithms use

the parallel Wiener model with $n_g = 4$ branches and neural networks with $K_g = 5$ hidden nodes. This is the worst model among all models with four branches. The control results are very bad. The controlled variable of the process practically does not stabilize on the required setpoint. There are frequent oscillations of manipulated and controlled variables. The amplitude of the oscillations is significant, and as a result, large overshoots are obtained.

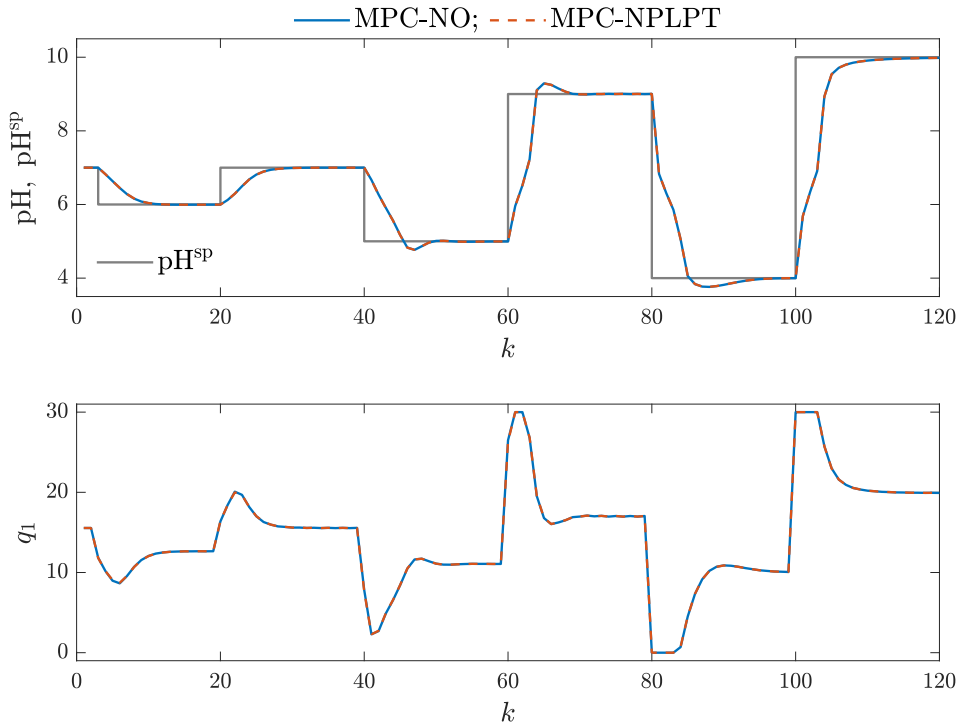

Figure 10. Simulation results: MPC-NO vs. MPC-NPLPT algorithms; both algorithms use the parallel Wiener model with three branches.

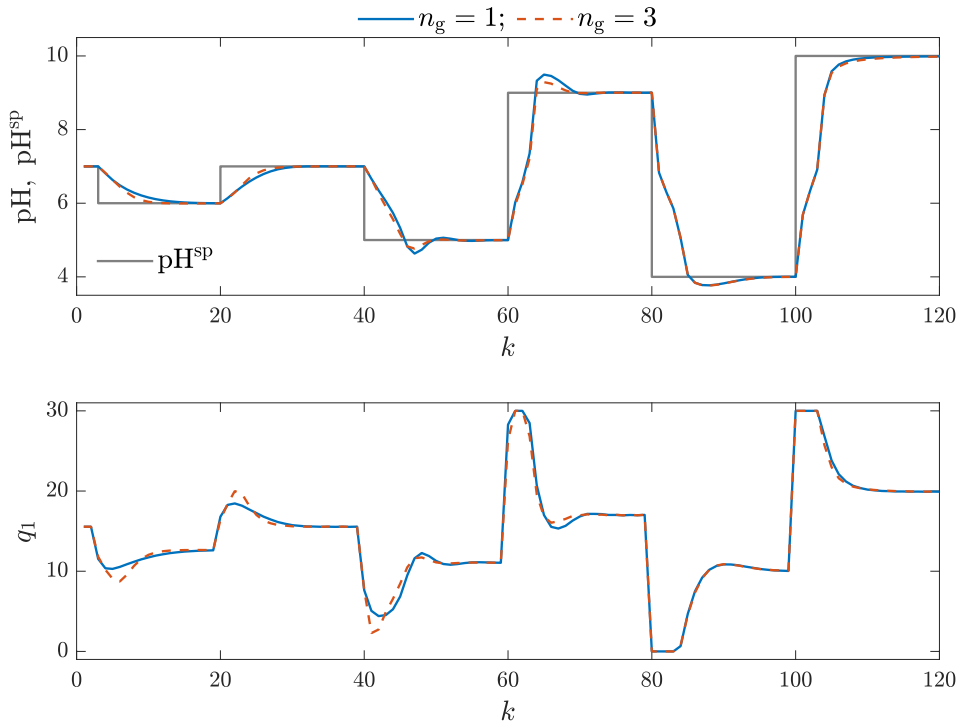

Figure 11. Simulation results: the MPC-NPLPT algorithm using the classical Wiener model vs. the MPC-NPLPT algorithms using the parallel Wiener ($n_g = 1$) model with three branches ($n_g = 3$).

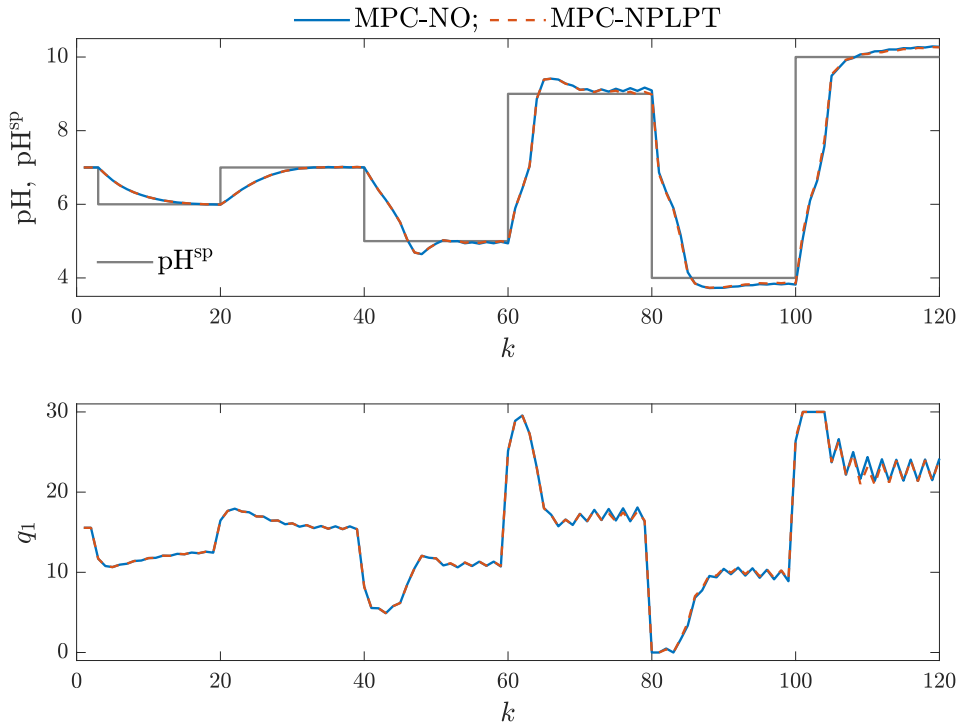

Figure 12. Simulation results: MPC-NO vs. MPC-NPLPT algorithms; both algorithms use the parallel Wiener model with $n_g = 4$ branches and neural networks with $K_g = 4$ hidden nodes.

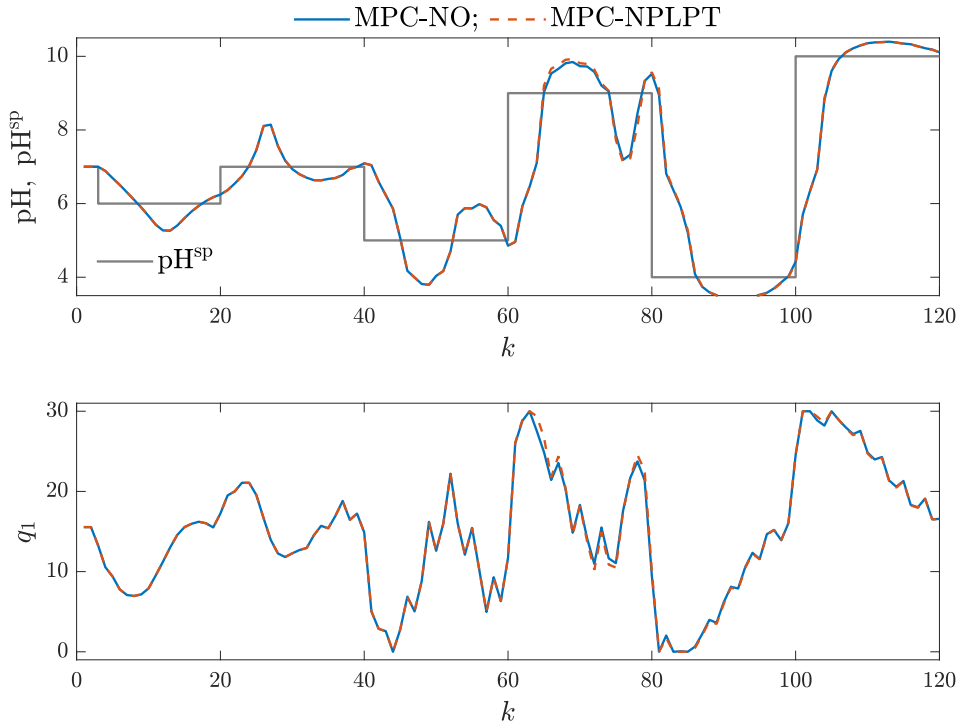

Figure 13. Simulation results: MPC-NO vs. MPC-NPLPT algorithms; both algorithms use the parallel Wiener model with $n_g = 4$ branches and neural networks with $K_g = 5$ hidden nodes.

5. Conclusions

This work is concerned with parallel Wiener models. Firstly, it details a computationally efficient MPC algorithm for the parallel Wiener model. The idea is to avoid nonlinear prediction and nonlinear online optimization. Conversely, an online linear approximation of the process predicted trajectory is successively computed, leading to a relatively simple

quadratic optimization. Secondly, parallel Wiener models are compared with classical ones for a benchmark neutralization process. Model accuracy is compared in the open-loop configuration. We find out that the parallel Wiener models really offer significantly better accuracy than the classical model. It is also necessary to stress that excessively complicated parallel models, with too many branches, suffer from overparameterization and cannot be trained fast. Hence, we suggest using parallel Wiener models with only a few branches for the considered process. Thirdly, parallel Wiener models are verified in MPC. Of note, the discussed MPC algorithm with online linearization and fast quadratic programming for the neutralization system produces practically the same results as the rudimentary MPC method with a fully nonlinear approach. Interestingly, control quality based on MPC algorithms based on parallel Wiener models is better than the classical model. However, the gain of using more complex models is not very significant due to the closed-loop negative feedback mechanism present in MPC.

Author Contributions: Conceptualisation R.N. and M.Ł.; methodology, R.N. and M.Ł.; software, R.N. and M.Ł.; validation, R.N. and M.Ł.; formal analysis, R.N. and M.Ł.; investigation, R.N.; writing—original draft preparation, R.N. and M.Ł.; writing—review and editing, R.N. and M.Ł.; visualization, R.N.; supervision, M.Ł. All authors have read and agreed to the published version of the manuscript.

Funding: This research was financed by Warsaw University of Technology in the framework of the project for the scientific discipline automatic control, electronics and electrical engineering.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Tatjewski, P. *Advanced Control of Industrial Processes, Structures and Algorithms*; Springer: London, UK, 2007.
2. Zarzycki, K.; Ławryńczuk, M. LSTM and GRU neural networks as models of dynamical processes used in predictive control: A comparison of models developed for two chemical reactors. *Sensors* **2021**, *21*, 5625. [CrossRef] [PubMed]
3. Andrei, A.M.; Bildea, C.S. Linear Model Predictive Control of Olefin Metathesis Process. *Processes* **2023**, *11*, 2216. [CrossRef]
4. Huyck, B.; De Brabanter, J.; De Moor, B.; Van Impe, J.F.; Logist, F. Online model predictive control of industrial processes using low level control hardware: A pilot-scale distillation column case study. *Control. Eng. Pract.* **2014**, *28*, 34–48. [CrossRef]
5. Sokólski, P.; Rutkowski, T.A.; Ceran, B.; Złotecka, D.; Horla, D. Event-Triggered Communication in Cooperative, Adaptive Model Predictive Control of a Nuclear Power Plant's Turbo-Generator Set. *Energies* **2023**, *16*, 4962. [CrossRef]
6. Simmini, F.; Caldognetto, T.; Bruschetta, M.; Mion, E.; Carli, R. Model Predictive Control for Efficient Management of Energy Resources in Smart Buildings. *Energies* **2021**, *14*, 5592. [CrossRef]
7. Ducaju, J.M.S.; Llobregat, J.J.S.; Cuenca, A.; Tomizuka, M. Autonomous Ground Vehicle Lane-Keeping LPV Model-Based Control: Dual-Rate State Estimation and Comparison of Different Real-Time Control Strategies. *Sensors* **2021**, *21*, 1531. [CrossRef]
8. Vu, T.M.; Moezzi, R.; Cyrus, J.; Hlava, J. Model Predictive Control for Autonomous Driving Vehicles. *Electronics* **2021**, *10*, 2593. [CrossRef]
9. Shi, J.; Li, K.; Piao, C.; Gao, J.; Chen, L. Model-Based Predictive Control and Reinforcement Learning for Planning Vehicle-Parking Trajectories for Vertical Parking Spaces. *Sensors* **2023**, *23*, 7124. [CrossRef]
10. Eskandarpour, A.; Sharf, I. A constrained error-based MPC for path following of quadrotor with stability analysis. *Nonlinear Dyn.* **2020**, *98*, 899–918. [CrossRef]
11. Rodriguez-Guevara, D.; Favela-Contreras, A.; Gonzalez-Villarreal, O.J. A qLPV-MPC Control Strategy for Trajectory Tracking of Quadrotors. *Machines* **2023**, *11*, 755. [CrossRef]
12. Aggelogiannaki, E.; Sarimveis, H. Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models. *Comput. Chem. Eng.* **2008**, *32*, 1225–1237. [CrossRef]
13. Mendez, A.P.; Whidborne, J.F.; Chen, L. Wind Preview-Based Model Predictive Control of Multi-Rotor UAVs Using LiDAR. *Sensors* **2023**, *23*, 3711. [CrossRef]
14. Li, P.; Tong, X.; Wang, Z.; Xu, M.; Zhu, J. Sensorless Model Predictive Control of Single-Phase Inverter for UPS Applications via Accurate Load Current Estimation. *Sensors* **2023**, *23*, 3742. [CrossRef] [PubMed]

15. Fu, H.; Yao, W.; Cajo, R.; Zhao, S. Trajectory Tracking Predictive Control for Unmanned Surface Vehicles with Improved Nonlinear Disturbance Observer. *J. Mar. Sci. Eng.* **2023**, *11*, 1874. [CrossRef]
16. Ławryńczuk, M. Nonlinear Predictive Control Using Wiener Models: Computationally Efficient Approaches for Polynomial and Neural Structures. In *Studies in Systems, Decision and Control*; Springer: Cham, Switzerland, 2014; Volume 389.
17. Bloemen, H.H.J.; Chou, C.T.; Boom, T.J.J.; Verdult, V.; Verhaegen, M.; Backx, T.C. Wiener model identification and predictive control for dual composition control of a distillation column. *J. Process. Control.* **2001**, *11*, 601–620. [CrossRef]
18. Cervantes, A.L.; Agamennoni, O.E.; Figueroa, J.L. A nonlinear model predictive control system based on Wiener piecewise linear models. *J. Process. Control.* **2003**, *13*, 655–666. [CrossRef]
19. Gómez, J.C.; Jutan, A.; Baeyens, E. Wiener model identification and predictive control of a pH neutralisation process. *Proc. IEE Part D Control. Theory Appl.* **2004**, *151*, 329–338. [CrossRef]
20. Kalafatis, A.D.; Wang, L.; Cluett, W.R. Linearizing feedforward–feedback control of pH processes based on the Wiener model. *J. Process. Control.* **2005**, *15*, 103–112. [CrossRef]
21. Al Seyab, R.K.; Cao, Y. Nonlinear model predictive control for the ALSTOM gasifier. *J. Process. Control.* **2006**, *16*, 795–808. [CrossRef]
22. Arto, V.; Hannu, P.; Halme, A. Modeling of chromato-graphic separation process with Wiener-MLP representation. *J. Process. Control.* **2001**, *78*, 443–458. [CrossRef]
23. Ławryńczuk, M.; Söffker, D. Wiener structures for modeling and nonlinear predictive control of proton exchange membrane fuel cell. *Nonlinear Dyn.* **2019**, *95*, 1639–1660. [CrossRef]
24. Ławryńczuk, M. Identification of Wiener models for dynamic and steady-state performance with application to solid oxide fuel cell. *Asian J. Control.* **2019**, *21*, 1836–1846. [CrossRef]
25. Zhang, C.; Meng, X.; Ji, Y. Parameter estimation of fractional Wiener systems with the application of photovoltaic cell models. *Mathematics* **2023**, *11*, 2945. [CrossRef]
26. Mahfouf, M.; Linkens, D.A. Non-linear generalized predictive control (NLGPC) applied to muscle relaxant anaesthesia. *Int. J. Control.* **1998**, *71*, 239–257. [CrossRef]
27. Patel, A.M.; Li, J.K.J. Validation of a novel nonlinear black box Wiener system model for arterialpulse transmission. *Comput. Biol. Med.* **2017**, *88*, 11–17. [CrossRef] [PubMed]
28. Schoukens, M.; Rolain, Y. Parametric MIMO parallel Wiener identification. In Proceedings of the 2011 50th IEEE Conference on Decision and Control/European Control Conference CDC-ECC, Orlando, FL, USA, 12–15 December 2011; pp. 5100–5105.
29. Schoukens, M.; Rolain, Y. Parametric identification of parallel Wiener systems. *IEEE Trans. Instrum. Meas.* **2012**, *61*, 2825–2832. [CrossRef]
30. Schoukens, M.; Rolain, Y. Parallel Wiener identification starting from linearized models. In Proceedings of the 2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings, Graz, Austria, 13–16 May 2012; pp. 1411–1415.
31. Hermansson, A.W.; Syafiie, S. Model predictive control of pH neutralization processes: A review. *Control. Eng. Pract.* **2016**, *45*, 98–109. [CrossRef]
32. Åkesson, B.M.; Toivonen, H.T.; Waller, J.B.; Nyström, R.H. Neural network approximation of a nonlinear model predictive controller applied to a pH neutralization process. *Comput. Chem. Eng.* **2005**, *29*, 323–335. [CrossRef]
33. Dougherty, D.; Cooper, D. A practical multiple model adaptive strategy for single-loop MPC. *Control. Eng. Pract.* **2003**, *11*, 141–159. [CrossRef]
34. Galán, O.; Romagnoli, J.A.; Palazoglu, A. Real-time implementation of multi-linear model-based control strategies–an application to a bench-scale pH neutralization reactor. *J. Process. Control.* **2004**, *14*, 571–579. [CrossRef]
35. Grancharova, A.; Kocijan, J.; Johansen, T.A. Explicit output-feedback nonlinear predictive control based on black-box models. *Eng. Appl. Artif. Appl.* **2011**, *24*, 388–397. [CrossRef]
36. Ławryńczuk, M. Modelling and predictive control of a neutralisation reactor using sparse Support Vector Machine Wiener models. *Neurocomputing* **2016**, *205*, 311–328. [CrossRef]
37. Mahmoodi, S.; Poshtan, J.; Jahed-Motlagh, M.R.; Montazeri, A. Nonlinear model predictive control of a pH neutralization process based on Wiener-Laguerre model. *Chem. Eng. J.* **2009**, *146*, 328–337. [CrossRef]
38. Nebeluk, R.; Ławryńczuk, M. Computationally efficient nonlinear model predictive control using the L1 cost-function. *Sensors* **2021**, *21*, 5835.
39. Janczak, A. Identification of Nonlinear Systems Using Neural Networks and Polynomial Models: A Block-Oriented Approach. In *Lecture Notes in Control and Information Sciences*; Springer: Berlin, Germany, 2004; Volume 310.