Chapter 5
Modelling and MPC of the Neutralisation Reactor Using Wiener Models

Please cite the book:
Maciej Ławryńczuk: Nonlinear Predictive Control Using Wiener Models: Computationally Efficient Approaches for Polynomial and Neural Structures. Studies in Systems, Decision and Control, vol. 389, Springer, Cham, 2022.

Abstract This Chapter discusses simulation results of MPC algorithms based on Wiener models applied to the neutralisation reactor. At first, the process is shortly described and identification of the Wiener model is discussed. Polynomials and neural networks are used in the nonlinear static block of the model. Effectiveness of both model classes is compared. Implementation details of different MPC algorithms are given. Next, MPC algorithms are compared in terms of control quality and computational time in the classical set-point following task and, additionally, some constraints are imposed on the predicted value of the controlled variable.

5.1 Description of the Neutralisation Reactor

Lest us consider a neutralisation (pH) reactor [8]. The reactor is schematically shown in Fig. 5.1. A base (NaOH) stream q_1, a buffer ($NaHCO_3$) stream q_2 and an acid (HNO_3) stream q_3 are mixed in a constant volume tank. The process has one input variable which is the base flow rate q_1 (ml/s) and one output variable which is the value of pH. Changes of the buffer and acid streams may be treated as disturbances of the process, but in this Chapter, they are assumed to be constant.

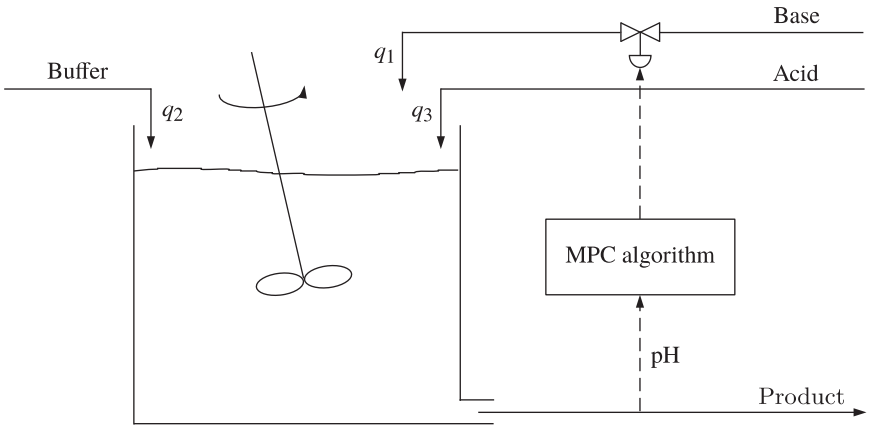

Fig. 5.1 The neutralisation reactor control system structure

Table 5.1 The neutralisation reactor: the parameters of the first-principle model

$K_1 = 6.35$	$W_{a_1} = -3.05 \times 10^{-3}$ mol	$W_{b_1} = 5 \times 10^{-5}$ mol
$K_2 = 10.25$	$W_{a_2} = -3 \times 10^{-2}$ mol	$W_{b_2} = 3 \times 10^{-2}$ mol
$V = 2900$ ml	$W_{a_3} = 3 \times 10^{-3}$ mol	$W_{b_3} = 0$ mol

The continuous-time fundamental model of the process is comprised of two ordinary differential equations

$$\frac{\mathrm{d}W_\mathrm{a}(t)}{\mathrm{d}t} = \frac{q_1(t)(W_{\mathrm{a}_1} - W_\mathrm{a}(t))}{V} + \frac{q_2(W_{\mathrm{a}_2} - W_\mathrm{a}(t))}{V} + \frac{q_3(W_{\mathrm{a}_3} - W_\mathrm{a}(t))}{V} \tag{5.1}$$

$$\frac{\mathrm{d}W_\mathrm{b}(t)}{\mathrm{d}t} = \frac{q_1(t)(W_{\mathrm{b}_1} - W_\mathrm{b}(t))}{V} + \frac{q_2(W_{\mathrm{b}_2} - W_\mathrm{b}(t))}{V} + \frac{q_3(W_{\mathrm{b}_3} - W_\mathrm{b}(t))}{V} \tag{5.2}$$

and one algebraic output equation

$$W_\mathrm{a}(t) + 10^{\mathrm{pH}(t)-14} - 10^{-\mathrm{pH}(t)} + W_\mathrm{b}(t)\frac{1 + 2 \times 10^{\mathrm{pH}(t)-K_2}}{1 + 10^{K_1-\mathrm{pH}(t)} + 10^{\mathrm{pH}(t)-K_2}} = 0 \tag{5.3}$$

State variables W_a and W_b are reaction invariants. The parameters of the above first-principle model are given in Table 5.1. The values of process variables in the nominal operating point are given in Table 5.2.

Fig. 5.2 depicts the structure of the continuous-time fundamental model of the neutralisation reactor in Simulink. It may be used to act as the simulated process. Of course, for this purpose the differential equations (5.1), (5.2) and the nonlinear relation (5.3) may be also solved directly in MATLAB, without the necessity of using Simulink.

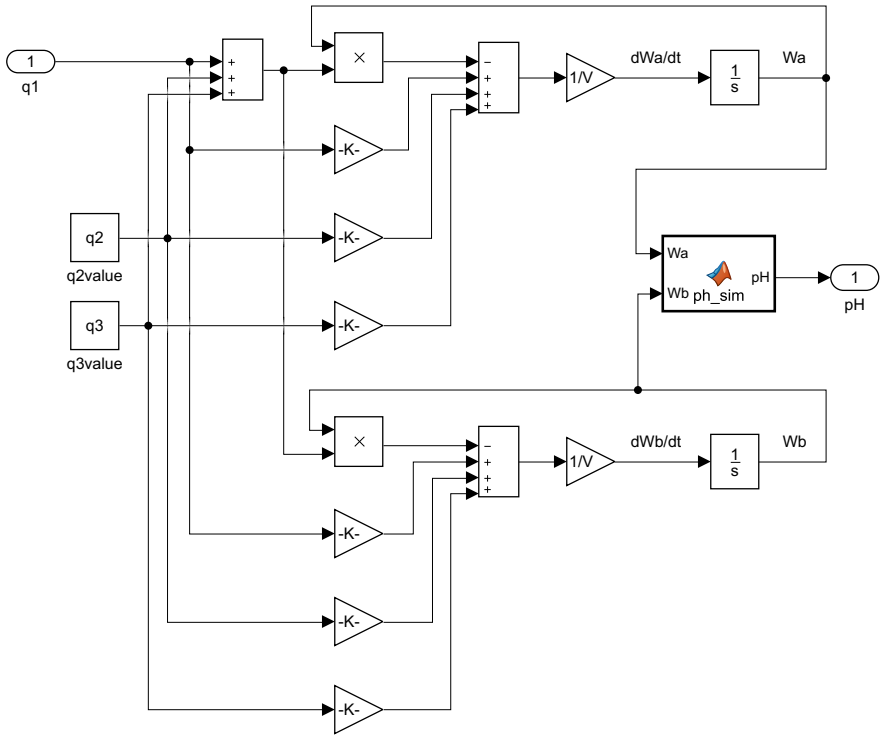

Fig. 5.2 The neutralisation reactor: the structure of the continuous-time fundamental model in Simulink

Table 5.2 The neutralisation reactor: the nominal operating point

$q_1 = 15.55$ ml/s	pH = 7
$q_2 = 0.55$ ml/s	$W_a = -4.32 \times 10^{-4}$ mol
$q_3 = 16.60$ ml/s	$W_b = 5.28 \times 10^{-4}$ mol

Example step responses of the process are depicted in Fig. 5.3. The excitation signal is

$$q_1(t) = \begin{cases} \bar{q}_1 & \text{if } t < 50 \text{ sec.} \\ \bar{q}_1 + \delta q_1 & \text{if } t \geq 50 \text{ sec.} \end{cases} \tag{5.4}$$

where $\bar{q}_1$ denotes the value of the variable q_1 in the nominal operating point. It is clear that both steady-state and dynamic properties of the pH reactor are nonlinear. Firstly, for the positive and negative steps, the process gains are different and the gains depend on the amplitude of the input step. Secondly, time-constants of all

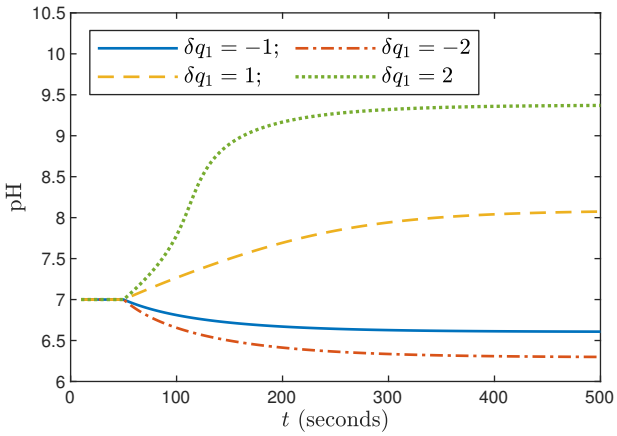

Fig. 5.3 The neutralisation reactor: example step responses

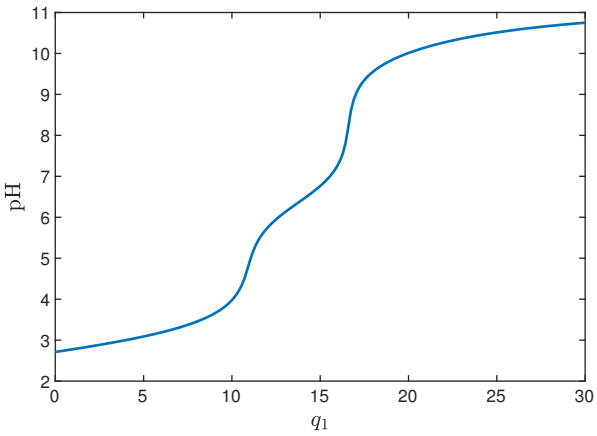

Fig. 5.4 The neutralisation reactor: the steady-state characteristic

the steps are different. The steady-state characteristic of the neutralisation reactor is depicted in Fig. 5.4.

In general, good control of the neutralisation process is necessary in chemical engineering, biotechnology and waste-water treatment industries [11]. Since both steady-state and dynamic properties of the neutralisation process are nonlinear, it is difficult to control by the classical linear control methods (e.g. PID), in particular when the set-point or other operating conditions change significantly and fast. In addition to its industrial importance, the neutralisation process is a classical benchmark used to evaluate different nonlinear model structures and control methods. Due to nonlinearity of the process, adaptive control techniques may be used, in particular, a model reference adaptive neural network control strategy [20], an adaptive nonlinear

output feedback control scheme containing an input-output linearising controller and a nonlinear observer [10], an adaptive nonlinear Internal Model Controller (IMC) [14] and an adaptive backstepping state feedback controller [25]. An alternative is to use multi-model controllers, e.g. a multi-model PID controller based on a set of simple linear dynamical models [2], a multi-model robust H_∞ controller [7], or fuzzy structures, e.g. a fuzzy PI controller [6], a fuzzy PID controller [12] and a fuzzy IMC structure [13]. An adaptive fuzzy sliding mode controller is presented in [3], a nonlinear IMC structure is discussed in [22]. Another options are: a neural network linearising scheme cooperating with a PID controller [20], a model-free learning controller using reinforcement learning [24] and an approximate multi-parametric nonlinear MPC controller [9].

Of course, the neutralisation process may be controlled by MPC algorithms. A multiple-model control strategy based on a set of classical linear MPC controllers is described in [4, 7]. A neural network trained off-line to mimic the nonlinear MPC algorithm may also be used [1]. A continuous-time MPC algorithm using a piecewise-linear approximation, which simplifies implementation, is discussed in [23]. When a nonlinear model is used directly in MPC for prediction, we obtain the nonlinear MPC-NO optimisation problem solved at each sampling instant on-line. Applications of the MPC-NO algorithm to the neutralisation process are reported in [21, 19]. An application of the neural Wiener model (a network of the MLP type is used as the static nonlinear part of the model) in the MPC-NPSL, MPC-NPLT and MPC-NPLPT algorithms is described in [16]. An interesting alternative to the neural network is the LS-SVM nonlinear approximator discussed in [17]. An excellent review of possible MPC approaches to the neutralisation process is given in [11]. Although the neutralisation reactor is typically considered in the SISO configurations, in some studies, the MIMO version of the process is used. A version of the MPC-NPSL algorithm, in which the model is not linearised in the simplified way, but the full linear approximation is calculated from the Taylor expansion, is described in [18, 15]. Unlike numerous works, not the Wiener but Hammerstein model structure is used. Although model accuracy is worse in comparison with that of the Wiener one, the resulting MPC algorithm works very well; all inaccuracies are compensated by the negative feedback mechanism present in MPC. Finally, a multilayer control system structure may be used in which the optimal set-points for the MPC algorithm are calculated on-line from an additional set-point optimisation problem [15].

5.2 Modelling of the Neutralisation Reactor for MPC

In the case of the neutralisation reactor, we will use for prediction in MPC some empirical input-output models, not the fundamental state-space model. If the fundamental model were used in MPC, it would be necessary to solve repeatedly on-line the state differential equations (5.1)-(5.2) and the nonlinear algebraic output equation (5.3). In order to find Wiener models, the fundamental state-space model is used

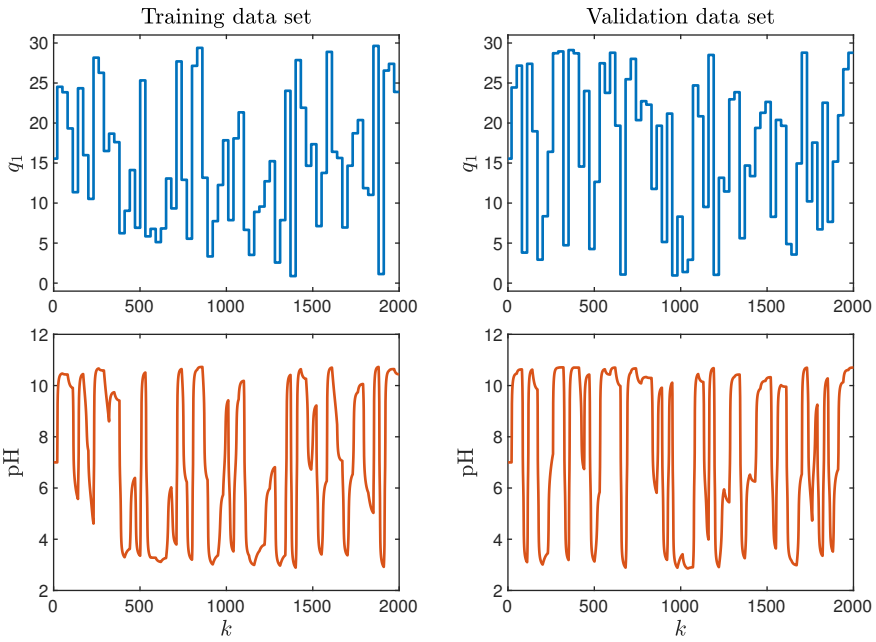

Fig. 5.5 The neutralisation reactor: open-loop simulations (the training and validation data sets)

to generate 2000 samples of the process output variable when a series of steps of random amplitude is applied as the input signal. The Runge-Kutta algorithm of the order 45 is used to solve the differential equations. The sampling time is $T_s = 10$ seconds. Two sets of data are generated: the training data set and the validation one. Fig. 5.5 depicts the manipulated and the controlled variables from the first and the second sets, respectively. For model identification, the process variables are scaled in the following way

$$u = (q_1 - \bar{q}_1)/15, \;\; y = 0.2(\mathrm{pH} - \overline{\mathrm{pH}}) \tag{5.5}$$

where $\bar{q}_1$ and $\overline{\mathrm{pH}}$ denote values of process variables at the nominal operating point (Table 5.2).

We consider the following models of the pH reactor:

a) the linear model,
b) the Wiener model with a polynomial static nonlinear block,
c) the Wiener model with a neural static nonlinear block,

All dynamical models have the second order of dynamics [16]. It means that the linear model is

$$y(k) = b_1 u(k-1) + b_2 u(k-2) - a_1 y(k-1) - a_2 y(k-2) \tag{5.6}$$

and the dynamic blocks of both types of the Wiener model are

$$v(k) = b_1 u(k-1) + b_2 u(k-2) - a_1 v(k-1) - a_2 v(k-2) \tag{5.7}$$

In the first structure of the Wiener model, we use polynomials in the nonlinear static part of the model. Such polynomial Wiener models may be determined in MATLAB from input-output data shown in Fig. 5.5 by means of the function `nlhw` which may be used to find general Hammerstein-Wiener models (a linear dynamic part sandwiched by two nonlinear static blocks) the structure of which is shown in Fig. 2.8. Syntax of the `nlhw` function is

```
sys = nlhw(Data,Orders,InputNL,OutputNL)
```

The estimated model is returned as the structure `sys`. `Data` is the data set used for model identification, `Orders` specifies the delay and the order of dynamics of the linear dynamic block, `InputNL` and `OutputNL` determine the types of static nonlinear approximators used in the input and output nonlinear static blocks. A few variants of nonlinear blocks are possible: piecewise linear functions, sigmoid or custom networks defined by the user, wavelet networks, saturations, dead zones, polynomials, constant unit gains. To obtain a Hammerstein model, a unit gain must be chosen as `OutputNL`. Conversely, to obtain a Wiener model, a unit gain must be chosen as `InputNL`. The nonlinear part of the polynomial Wiener model is

$$y(k) = g(v(k)) = \sum_{i=0}^{K} c_i v^i(k) \tag{5.8}$$

where K denotes the degree of the polynomial and c_i are coefficients.

In the second structure of the Wiener model, we use the sigmoid neural network as the nonlinear static part of the model. Such neural Wiener models may also be determined in MATLAB from input-output data by means of the function `nlhw`. The nonlinear part of the neural Wiener model is

$$y(k) = g(v(k)) = d + PL(v(k) - r) + \sum_{i=1}^{K} a_i \varphi((v(k) - r)Qb_i + c_i) \tag{5.9}$$

where the transfer function is the sigmoid one

$$\varphi(v(k)) = \frac{1}{1 + \exp(-v(k))} \tag{5.10}$$

For the SISO process, the scalar parameters are: a linear coefficient L, the linear subspace P, the nonlinear subspace Q, the offset d and a mean value of the data r. K is the number of nonlinear nodes. The parameters are: a_i, b_i and c_i. We may simplify

the notation by using the following representation of the nonlinear block

$$y(k) = g(v(k)) = d^{\mathrm{nn}} + l^{\mathrm{nn}}(v(k) - r^{\mathrm{nn}}) + \sum_{i=1}^{K} a_i^{\mathrm{nn}} \varphi((v(k) - r^{\mathrm{nn}})b_i^{\mathrm{nn}} + c_i^{\mathrm{nn}}) \quad (5.11)$$

where $l^{\mathrm{nn}} = PL$ and $b_i^{\mathrm{nn}} = Qb_i$ for $i = 1, \ldots, K$ are scalars, auxiliary superscripts "nn" are added to all parameters of the nonlinear block.

Table 5.3 gives the values of model errors for different model configurations. All errors are defined as the sum of squared differences between the data samples and the output of the model for the whole data sets [5]. To select the model finally used in MPC, we take into account the validation errors for dynamic data (E_{val}) and the validation errors for steady-state data ($E_{\mathrm{val}}^{\mathrm{ss}}$). The steady-state error is calculated for 200 equidistant points in the domain of q_1. The linear model has large errors. As far as the polynomial Wiener model is concerned, the structures of the degree $K = 2, 3, \ldots, 9, 10, 15, 20$ are compared. The model with the polynomial degree $K = 5$ is chosen because it gives a good compromise between accuracy and complexity. Moreover, increasing the degree of the polynomial does not give better results. As far as the neural Wiener model is concerned, the structures with $K = 1, 2, \ldots, 10, 15, 20$ hidden nodes are compared. In general, the neural Wiener models are more precise than the polynomial ones. Furthermore, for the polynomials of high degree, in particular for $K = 10, 15, 20$, large errors are obtained, whereas the neural Wiener models are much more precise. It is important that in the case of the neural Wiener model, the errors do not grow significantly when the number of hidden nodes is increased. The neural Wiener model with five hidden units is chosen since it gives very low values of errors and it has a moderate number of parameters (22).

Fig. 5.6 depicts the dynamic validation data set vs. the outputs of four models (the linear model, the polynomial Wiener model of the degree $K = 5$, the polynomial Wiener model of the degree $K = 15$ and the neural Wiener model containing $K = 5$ hidden nodes). Fig. 5.7 depicts the relation between the validation data vs. the outputs of the compared models. As the numerical data indicate, the linear model is very imprecise, the polynomial Wiener model of the degree $K = 5$ is good, the polynomial Wiener model of the degree $K = 15$ is very bad and the neural Wiener model containing $K = 5$ hidden nodes is excellent.

Finally, it is interesting to compare the real steady-state characteristic of the neutralisation reactor vs. the characteristic of its empirical models. Such a comparison is shown in Fig. 5.8 for the linear model, the polynomial Wiener model with different degree of the polynomial and the neural Wiener model with a different number of the hidden nodes. The obtained results correspond with the values of the steady-state error $E_{\mathrm{v}}^{\mathrm{ss}}$ given in Table 5.3. We can see that the neural Wiener models make it possible to achieve very good steady-state modelling. It is practically impossible for the polynomial Wiener model, i.e. when the degree of the polynomial is low, the

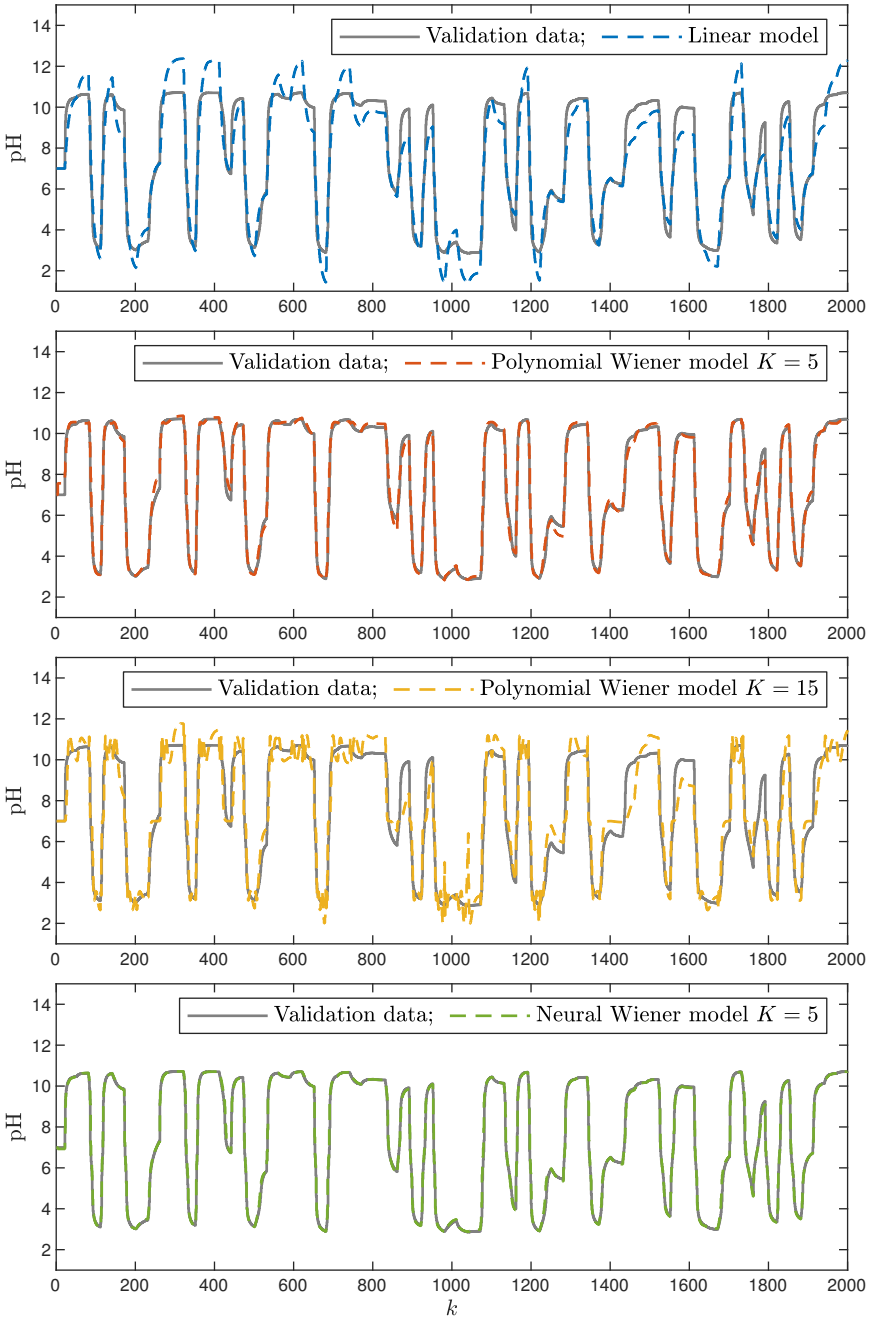

Fig. 5.6 The neutralisation reactor: the validation data set vs. the outputs of four models (the linear model, the polynomial Wiener model of the degree $K = 5$, the polynomial Wiener model of the degree $K = 15$ and the neural Wiener model containing $K = 5$ hidden nodes)

Table 5.3 The neutralisation reactor: comparison of linear, polynomial Wiener models of the degree K and neural Wiener models containing K hidden nodes in terms of the number of parameters (n_{par}), errors for dynamic data (E_{train} and E_{val} denote the errors for the training and validation data sets, respectively) and errors for the validation steady-state data (E_{v}^{ss}); for the Wiener models the number of training epochs (n_{train}) are given

Model	K	n_{par}	n_{train}	E_{train}	E_{val}	$E_{\text{val}}^{\text{ss}}$
Linear	–	4	–	4.6849×10^{1}	5.5100×10^{1}	7.2282
Polynomial Wiener	2	7	100	3.5293×10^{1}	5.1654×10^{1}	1.2027×10^{1}
Polynomial Wiener	3	8	22	1.0896×10^{1}	1.0647×10^{1}	1.7528
Polynomial Wiener	4	9	15	9.6340	1.0468×10^{1}	3.2236
Polynomial Wiener	5	10	55	7.1403	6.4198	5.3075×10^{-1}
Polynomial Wiener	6	11	100	7.1369	6.4166	5.4547×10^{-1}
Polynomial Wiener	7	12	100	6.9839	6.2962	6.6179×10^{-1}
Polynomial Wiener	8	13	63	6.4389	5.9981	2.8954
Polynomial Wiener	9	14	63	6.4388	6.0000	2.7576
Polynomial Wiener	10	15	81	5.3186	5.3038	3.5245×10^{1}
Polynomial Wiener	15	20	7	6.1889×10^{1}	5.2801×10^{1}	2.7649×10^{4}
Polynomial Wiener	20	25	7	1.5017×10^{2}	1.6791×10^{2}	8.6413×10^{5}
Neural Wiener	1	10	13	7.4231	6.6865	5.2167×10^{-1}
Neural Wiener	2	13	20	6.9491	6.3098	4.9063×10^{-1}
Neural Wiener	3	16	67	4.2745	4.7204	1.2290
Neural Wiener	4	19	100	1.6546	2.4682	4.2338×10^{-2}
Neural Wiener	5	22	150	1.6495	2.4640	3.8260×10^{-2}
Neural Wiener	6	25	200	1.6489	2.4657	4.9552×10^{-2}
Neural Wiener	7	28	200	1.6482	2.4620	4.0388×10^{-2}
Neural Wiener	8	31	250	1.6250	2.4933	3.8154×10^{-2}
Neural Wiener	9	34	250	1.6568	2.4762	4.4784×10^{-2}
Neural Wiener	10	37	250	1.6202	2.4675	4.0583×10^{-2}
Neural Wiener	15	52	250	1.6187	2.4925	4.6630×10^{-2}
Neural Wiener	20	67	250	1.6038	2.5171	3.9830×10^{-2}

steady-state model characteristic does not have enough degrees of freedom; when the degree of the polynomial is high, numerical problems occur (ill-conditioning).

5.3 Implementation of MPC Algorithms for the Neutralisation Reactor

The following MPC algorithms are compared:

1. The classical LMPC algorithm based on a linear model (three example models, obtained for different operating points, are considered).
2. The classical MPC-inv algorithm.
3. The MPC-SSL and MPC-NPSL algorithms.