Nonlinear PID control using Hammerstein or Wiener models

Erol, F.

Award date:
1999

Eindhoven University of Technology
Department of Electrical Engineering
Measurement and Control Group

Nonlinear PID control using Hammerstein or Wiener models

By F. Erol

Master of Science Thesis
Carried out from July 1998 to June 1999
Commissioned by prof.dr.ir. P.P.J. van den Bosch
Under supervision of dr.ir. Y.C. Zhu

Date: 15 June 1999

Abstract

The objective of this project is to study nonlinear process control using Hammerstein or Wiener models.

First closed-loop identification of Hammerstein and Wiener model is discussed. The Wiener model consists of two parts: a linear dynamic model followed by a nonlinear static part. The Hammerstein model has the same parts as the Wiener model but in reversed order. The nonlinear PID controller structures for the Hammerstein and Wiener models are proposed. By compensating the nonlinear static part of these models by its inverse, a linear PID controller can be used for controlling these models. The linear PID controller is designed by using the Internal Model Control approach.

The identification and control method is applied to a pH process. A mathematical representation of the pH process is derived. The pH process is modelled as a continuous stirred tank reactor (CSTR) by using mass balances. The derived model is not used for the purpose of controlling but for analysing the dynamic behaviour of that process. This process model can be approximated by a Wiener model. The pH process consists of two input streams of chemical solution, of which one input, a base solution is used as an input variable and the other a constant stream of an acid solution as a disturbance. The output of the process is the pH value inside the CSTR.

The pH process is identified by using the SISO Wiener model identification algorithm and the obtained model is used for controller design. Simulations with the linear and nonlinear PID controller are done and compared to each other. Finally, the linear PID controller and the nonlinear PID are tested on the pH process. The control objective is to keep the pH value in the CSTR at a desired value. The nonlinear PID controller can control the process in a larger range than the linear PID can. Implementation of these controllers was done by using a program that is called LabView.

Contents

1 INTRODUCTION 3

2 PARAMETRIC MODEL IDENTIFICATION 5

2.1 THE WIENER MODEL 5
2.2 SISO WIENER MODEL IDENTIFICATION 6
2.3 THE HAMMERSTEIN MODEL 15
2.4 SISO HAMMERSTEIN MODEL IDENTIFICATION 15
2.5 EXTRAPOLATING THE NONLINEAR FUNCTION 18

3 NONLINEAR PID CONTROLLER DESIGN 19

3.1 CONTROLLER STRUCTURE FOR WIENER MODEL 19
3.2 CONTROLLER STRUCTURE FOR HAMMERSTEIN MODEL 20
3.3 LINEAR PID CONTROLLER DESIGN 21

4 THE PH PROCESS 28

4.1 DESCRIPTION OF THE PH PROCESS 28
4.2 PHYSICAL MODELLING OF THE PROCESS 29
4.3 THE PH PROCESS APPROXIMATED AS A WIENER MODEL 33

5 IDENTIFICATION AND CONTROL OF THE PH PROCESS 35

5.1 THE IDENTIFICATION EXPERIMENT DESIGN 35
5.2 THE IDENTIFICATION SCHEME 36
5.3 CHOICE OF THE INPUT SIGNAL FOR IDENTIFICATION 41
5.4 IDENTIFICATION RESULTS 42
5.5 THE COMPARISON AND CHOICE OF THE MODEL 47
5.6 SIMULATION RESULTS 48
5.7 EXPERIMENTS OF CONTROLLING THE PH PROCESS 49
5.8 THE PERFORMANCE OF THE NONLINEAR PID CONTROLLER 51

6 CONCLUSIONS AND RECOMMENDATIONS 52

BIBLIOGRAPHY 53

APPENDIX 1. LABVIEW SCHEME OF THE NONLINEAR PID CONTROLLER 55

APPENDIX 2. EXTRAPOLATION PROGRAM 56

1 Introduction

Most models used for controller design are linear. However, many chemical processes are nonlinear. A mathematical model of a process is required for simulation and control of that process. The mathematical model can be understood as a mathematical description that defines the relationship between the input and the output of that process. There are two ways to process model building: physical modelling approach an identification approach. Physical modelling is deriving process models by using first principles (physical, chemical and biological laws).
The advantages of a physical model are:

- it provides physical insight in the underlying process, which is useful for understanding the process behaviour,
- the model can cover a wide range of process operations.

The disadvantages of physical modelling (for control) are:

- physical models of industrial processes are difficult to obtain and the cost of modelling is high,
- the accuracy of a physical model is often low and the model is too complex for use in control.

Process identification is to derive dynamic process models using process test data. Identification is often called black-box modelling, because physical laws are not explicitly used in this approach. When comparing to physical modelling, the advantages of identification are:

- low cost in modelling,
- high model accuracy for the tested operating range,
- the model is simple in structure.

An disadvantage of identification is that the model can only describe the process behaviour in the operating range that has been tested

We will study the identification of Wiener model and Hammerstein model. A Wiener model is a nonlinear model with a linear dynamic block followed by a static nonlinear function. The Hammerstein model has the same blocks but they are in reversed order.

In many chemical processes the nonlinearity of the steady-state gain is more significant than the nonlinearity of the time constants. This is why these types of nonlinear models are useful for chemical processes. An important advantage of these models is that they permit the use of standard linear controller design methods. This is possible because the static nonlinearity in the process can be cancelled by inserting the nonlinear inverse of the static nonlinearity in the proper location in the loop. For the Hammerstein model, the inverse nonlinear function should be located at the output of the controller. For the Wiener model, the inverse nonlinear function should be located at the output of the process. The controller sees only the linear dynamic part of the process, and conventional linear controller methods can be used. For the control of the linear part a linear PID controller is used. The linear PID controller is tuned by using the Internal Model Control (IMC) approach.

The process is a pH process at the laboratory of the System and Control Group of the Department of Chemical Technology and Applied Physics. The control objective is to keep the pH value in a continuously stirred tank reactor (CSTR) at a desired value.

The identification procedure for the Wiener model is applied on the pH process. Then the obtained model is used for simulating the linear and nonlinear PID controller. The linear and nonlinear PID controller is simulated using Simulink block models in the Matlab program. Finally the controller is implemented to control the pH process by using a program which is called LabView. The performance of the nonlinear PID controller is compared with that of the linear PID controller. This is done under the same conditions in the simulations and real measurements.

The report is organised as follows. In chapter 2, the identification method for the Wiener model is theoretically described. The identification of the Hammerstein model is discussed briefly. Also a way to extrapolating nonlinear functions is proposed. Chapter 3 describes the design of the nonlinear PID controller. The controller structures for the Wiener and Hammerstein are proposed and compared. Chapter 4 describes the pH process. A physical model of the pH process is derived. In chapter 5, a Wiener model the identification procedure is applied to the pH process. A nonlinear linear PID controller is designed and used for controlling the Wiener model. Chapter 6 contains the conclusions and the recommendations.

2 Parametric model identification

This chapter describes the parametric model identification of single-input single-output (SISO) Hammerstein and Wiener models (see [ZHU 98a] and [ZHU 98a]). Parametric models can be used for simulation and for control design. The model structure of these models will be discussed. The parameters of the chosen structure, which are nonlinear in the parameters, are obtained by minimising a cost function. The orders of the parts of these models are determined by looking at some error criterions. The model validation method is described.

Because of the similarity of the identification method for Wiener models and Hammerstein models, the identification of the Wiener model will be treated in detail and the Hammerstein model will be discussed briefly. The extrapolating of a nonlinear function is treated which can be useful when the model is used in a larger range than tested.

The main purpose of identification is control. The so-called ASYM method that has been used for identifying linear model [ZHU 98c] and Hammerstein model [Zhu 98b] will be applied to identify these models.

2.1 The Wiener model

A Wiener model is a nonlinear model with a linear dynamic block followed by a static nonlinear function. The Wiener model is a block-oriented nonlinear model. The basic idea of block-oriented model is that the system can be divided into blocks with static nonlinear blocks with dynamic linear terms that are combined into a system.
The Wiener model in discrete-time is given in figure 2.1. $G(q)$ represents a linear time-invariant transfer function with its input $u(t)$ and output $w_o(t)$. The input to the nonlinear static function $f(w)$ consist of $w_o(t)$ and the disturbance term $v(t)$. It is assumed that: 1) the nonlinear static function $f(w)$ is continuous, invertible and monotone; 2) the output of the linear part is disturbed by a stationary stochastic process $\{v(t)\}$.

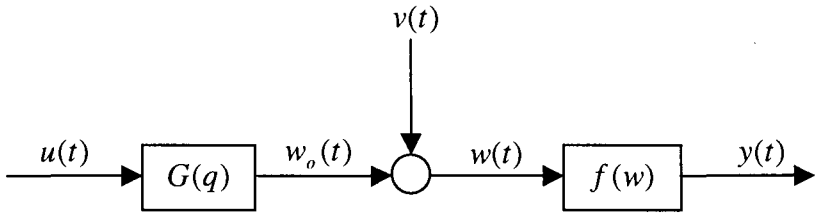


Figure 2.1 The Wiener model

The output of the model is as following: $y(t) = f[G(q)u(t) + v(t)]$. The disturbance term $v(t)$ is placed before the nonlinear block, which is different from the normal assumption that a stationary disturbance acts at the process output. This model implies asymmetrical output disturbance: the output disturbance is large when the process gain is large and is small when the gain is small. This assumption is more realistic from process operation point of view. The reason is that, a larger gain implies that the underlying process is sensitive and hence the disturbance will have strong influence at its output; a small gain means that the process is insensitive and the disturbance will have little influence at its output.

2.2 SISO Wiener model identification

Identification means the determination of the model (mathematical) of a dynamic system from input/output measurements. The knowledge of a model is necessary for the design and implementation of a control system.

There are two type of dynamic models:
- Nonparametric models (e.g. frequency response, step response);
- Parametric models (e.g. transfer function, differential or difference equation)

Often parametric models are needed because many control design methods require this kind of model description. Usually identified parametric models are more accurate than nonparametric models.

For the identification of a parametric model, it is necessary to parameterise the model. The parametric model has a restricted number of parameters in contrast with the nonparametric model. One of the most powerful and compact parameterisation for the linear and disturbance part is the so-called Box-Jenkins (BJ) model. The Box-Jenkins model structure is given by:

$$y(t) = G(q)u(t) + H(q)e(t) \tag{1}$$

where $G(q)$ and $H(q)$ are rational functions of q^{-1}.

When we apply this structure to the single-input single-output (SISO) Wiener model we get the following structure:

$$w(t) = G(q)u(t) + H(q)e(t) \tag{2}$$

with

$$G(q) = \frac{b_0 + b_1 q^{-1} + \ldots + b_n q^{-l}}{1 + a_1 q^{-1} + \ldots + a_n q^{-l}}, \quad H(q) = \frac{1 + c_1 q^{-1} + \ldots + c_n q^{-l}}{1 + d_1 q^{-1} + \ldots + d_n q^{-l}}$$

where:
- l is called the order of the linear part of the Wiener model;

- q^{-1} is the notation for the delay operator and the complex variable z^{-1};
- $H(q)$ is transfer function of a linear filter;
- $\{e(t)\}$ is a white noise sequence with zero mean value and variance λ^2;

For the parametrisation of the nonlinear block, it is necessary to model it by a function that gives a smooth and close to the true function. There are several ways to parameterise the nonlinear block.
The nonlinear block could be parameterised by using polynomial functions. The disadvantages of polynomial functions are:
- they do not extrapolate well;
- high degree polynomials has oscillatory behaviour between data values.

Another way to parameterise the nonlinear block is to use linear spline functions. The linear spline does not have the above disadvantages, but these types of functions are not smooth. This means that their derivatives are not continuous at their break points or "knots". To solve this problem we use cubic splines instead of linear splines.

Let K denote a set of knots $\{w_1, w_2, ..., w_m\}$, which are real numbers and are distinct and arranged in increasing order with:

$$w_1 = w_{\min} < w_2 < w_3 < ... < w_m = w_{\max} \tag{3}$$

A cubic spline function defined for all real numbers w is given as:

$$y(w) = \sum_{k=2}^{m-1} \alpha_k |w - w_k|^3 + \alpha_m + \alpha_{m+1} w + \alpha_{m+2} w^2 + \alpha_{m+3} w^3 \tag{4}$$

Where $[\alpha_2, \alpha_3, ..., \alpha_{m+3}]$ are fixed real numbers. The knot w_1 is not used in the function (4) but is used together with the knot w_m for calculating the other knots. Here m is called the number of knots, which can be seen as the "degree", or "order" of the cubic splines. Note that the number of parameters of the cubic splines is $m+2$. It is easy to verify that the first and second derivatives of the function are continuous and hence the function is smooth.

Note that the signal $w(t)$ can not be measured, hence an arbitrary gain may be distributed between the linear block and the nonlinear block. This means that the parameters of the model cannot be uniquely determined without further constraint. A solution to this problem is to fix the gain of the nonlinear block. For example one can let $\alpha_{m+1} = 1$.

An identification procedure has four steps, which also holds for the Wiener model:
1) **Test design.**
   An identification test must be designed by using excitation signals so that the parameters of the model in (2) and (4) are identifiable. If possible, the test should be designed so that the identified model is most suitable for control application. Denote the collected input/output data as:

$$\{u(1), y(1), u(2), y(2), \cdots, u(N), y(N)\}$$

where N is the number of samples.

2) **Parameter estimation.**
The parameters of the model must be designed using the input/output data from the test by minimising the loss function:

$$V_{PE} = \sum_{t=1}^{N} \varepsilon^2(t) \tag{5}$$

where

$$\varepsilon(t) = H^{-1}(q)\left[f^{-1}(y(t)) - G(q)u(t)\right] = \frac{D(q)}{C(q)}\left[f^{-1}(y(t)) - \frac{B(q)}{A(q)}u(t)\right]. \tag{6}$$

The error $\varepsilon(t)$ in (6) can be interpreted as one-step-ahead prediction error for the output of linear block $w(t)$, not for process output $y(t)$. The one-step-ahead prediction error is defined as the difference between the measured output and the predicted output based on information available at time instant $t-1$.

3) **Order selection.**
The orders n and m must be determined so that the obtained model is most accurate for control. The selection of the orders n and m is done by taken the combinations of these two orders and the best orders are determined using some criterion.

4) **Model validation.**
The identified model must be verified whether it is suitable for control. If not, redesign a new identification test. For applications such as control, it is not sufficient if the identified model is the best process approximation for the given data, it is more important to know whether the test is properly carried out and the data is informative enough for the purpose of control.

In the following we will discuss parameter estimation, order selection and model validation. The test design will be discussed in chapter 5.

Parameter estimation

The prediction error defined in (6) is highly nonlinear in model parameters. The prediction error is given by

$$\varepsilon(t) = y(t) - \hat{y}(t). \tag{7}$$

Direct minimisation of the loss function (V_{PE}) can be costly (the price of the model is associated with the effort to calculate it) and may be run into numerical problems. It is thus desirable to reduce this complexity by looking for some simpler numerical schemes, such as relaxation algorithm often used for Hammerstein model estimation. The essence of a relaxation algorithm is to use model structures in which the error is linear or bilinear in model parameters and one part of the parameters can be estimated by simple linear least-squares (LS) while fixing the other part of the parameters.

Any linear prediction error model structure can be approximated arbitrarily well by an ARX model or equation error model with sufficiently high order. The equation error model structure has the structure as the ARX model:

$$y(t) = \frac{B(q)}{A(q)}u(t) + \frac{1}{A(q)}e(t) \tag{8}$$

where $u(t)$ is the input, $y(t)$ the output and $e(t)$ is white Gaussian noise with zero mean.
Based on this fact, we shall approximate the linear part of the Wiener model in (2) by a high order ARX model:

$$w(t) = \frac{B^n(q)}{A^n(q)}u(t) + \frac{1}{A^n(q)}e(t). \tag{9}$$

Substituting $w(t)$ in (2) using the ARX model and taking the inverse of the nonlinear function on both sides yields the next equations:

$$y(t) = f\left[\frac{B^n(q)}{A^n(q)}u(t) + \frac{1}{A^n(q)}e(t)\right], \tag{10}$$

$$f^{-1}\{y(t)\} = \frac{B^n(q)}{A^n(q)}u(t) + \frac{1}{A^n(q)}e(t). \tag{11}$$

Multiplying $A^n(q)$ to both sides of (11) yields:

$$A^n(q)f^{-1}(y(t)) = B^n(q)u(t) + e(t). \tag{12}$$

The loss function for parameter estimation for model (13) becomes:

$$V_{ARX} = \sum_{t=1}^{N} \varepsilon^2(t) \tag{13}$$

where

$$\varepsilon(t) = A^n(q) f^{-1}(y(t)) - B^n(q)u(t). \tag{14}$$

Now the inverse nonlinear function $f^{-1}(y(t)) = w(t)$ in (14) can be parameterised by a cubic spline function in the similar way as for the nonlinear function $f(w) = y$. This yields the next equation:

$$w = \sum_{k=2}^{m_2-1} \gamma_k |y - y_k|^3 + \gamma_{m_2} + \gamma_{m_2+1} y + \gamma_{m_2+2} y^2 + \gamma_{m_2+3} y^3 \tag{15}$$

where m_2 is the number of knots, which can be seen as the "degree", or "order" of the inverse nonlinear function.

After substituting this function in (14), the next equation is obtained:

$$\varepsilon(t) = A^n(q)\left[\sum_{k=2}^{m_2-1} \gamma_k |y(t) - y_k|^3 + \gamma_{m_2} + \gamma_{m_2+1} y(t) + \gamma_{m_2+2} y(t)^2 + \gamma_{m_2+3} y(t)^3\right] - B^n(q)u(t). \tag{16}$$

The error $\varepsilon(t)$ is now linear in the parameters of $B^n(q)$ and bilinear in the parameters of $A^n(q)$ and $f^{-1}(y)$. This is much simpler than the original error form. Therefore, one can use the following relaxation algorithm for the estimation of the model.

After substituting the equation (14) in (13), the next equation is obtained:

$$V_{ARX} = \sum_{t=1}^{N} \left\{A^n(q) f^{-1}(y(t)) - B^n(q)u(t)\right\}^2. \tag{17}$$

The estimating of the inverse nonlinear function and the high-order ARX model is done with the following procedure.

First the initialisation procedure. There are two methods for initialisation of the equation (17):
1) Set $A^n(q) = 1$ and estimate $B^n(q)$ and $f^{-1}(y)$ sequentially using linear least squares.

2) Set $f^{-1}(y) = y$ and estimate $A^n(q)$ and $B^n(q)$ using linear least squares.

After that follows the iteration procedure:
Suppose that $\hat{A}^n_{(i)}(q)$, $\hat{B}^n_{(i)}(q)$ and $\hat{f}^{-1}_{(i)}(y(t))$ as the estimates from iteration i, then

1. Calculate $\hat{A}^n_{(i+1)}(q)$ and $\hat{B}^n_{(i+1,1)}(q)$ for fixed $\hat{f}^{-1}_{(i)}(y(t))$ by minimising:

$$V^{(i+1,1)}_{ARX} = \sum_{t=1}^{N} \left\{ A^n_{(i+1)}(q)\hat{f}^{-1}_{(i)}(y(t)) - B^n_{(i+1,1)}(q)u(t) \right\}^2. \qquad (18)$$

2. Calculate $\left[\hat{\gamma}_{2(i+1)}, \hat{\gamma}_{3(i+1)}, \ldots, \hat{\gamma}_{m_2+3(i+1)}\right]$ for fixed $\hat{A}^n_{(i+1)}(q)$ and $\hat{B}^n_{(i+1,1)}(q)$ by minimising:

$$V^{(i+1,2)}_{ARX} = \sum_{t=1}^{N} \left\{ \hat{A}^n_{(i+1)}(q)f^{-1}_{(i)}(y(t)) - \hat{B}^n_{(i+1)}(q)u(t) \right\}^2. \qquad (19)$$

3. Go back to 1. Stop when convergence no longer continuous.

Note that both steps in the relaxation algorithm are linear least squares problems, which are numerically simple and reliable.

Finally, the nonlinear function $f(y)$ is determined from the estimate of its inverse $\hat{f}^{-1}(y)$. This is an approximation problem and is done by using a linear least square estimation.

The error in the estimate consists of the components: one due to a too low model order (called bias) and one due to the noise (called variance). A bias, i.e. the true parameters are not equal to the estimated parameters, will occur if the process is not a member of the model set, e.g. for models of too low order. Then the estimated transfer function is different from the real one, even in the noiseless case. So the bias will in general decrease with increasing model order. The variance plays another role. In order to reduce the variance error, model reduction techniques can be used.

The obtained Wiener model with high order ARX linear part is unbiased. Model reduction is used for detection whether the found model is unnecessarily high order.
Assuming that the inverse nonlinear function is known exactly, the order n of the ARX model is sufficiently high and the input signal $u(t)$ is sufficiently rich (persistent exciting), then it can be shown (see Ljung, 1985) that the estimated frequency response of the high order model is unbiased and its error follows a Gaussian distribution with variance given as

$$\mathrm{var}\left[\hat{G}^n(e^{i\omega})\right] \approx \frac{n}{N} \frac{\Phi_v(\omega)\lambda^2}{\Phi_u(\omega)\lambda^2 - \left|\Phi_{ue}(\omega)\right|^2} \qquad (20)$$

where $\hat{G}^n(e^{i\omega})$ is the frequency response of the estimated high order ARX model, $G^o(e^{i\omega})$ is that of the true model, n is the order of the ARX model, N is the number of data points, $\Phi_v(\omega)$ is the power spectrum of output disturbance, λ^2 is the variance of white noise $\{e(t)\}$ that generated the output disturbance, $\Phi_u(\omega)$ is the power spectrum of the input $u(t)$ and $\Phi_{ue}(\omega)$ is the cross-

spectrum between the white noise $\{e(t)\}$ and $u(t)$ (due to feedback). This result holds for both open-loop and closed-loop tests.

The frequency response of the high-order estimates

$$\hat{G}^n(e^{i\omega_1}),\hat{G}^n(e^{i\omega_2}),\cdots,\hat{G}^n(e^{i\omega_n}),\ \text{where}\ \omega_k=\frac{k\cdot\pi}{n},\ k=1,\ldots,\ n \tag{21}$$

as the noisy observations of the true transfer function, we can then apply the maximum likelihood principle. Then based on (20), it can be shown that when $N\to\infty$, the asymptotic negative log-likelihood function for the process model is given by:

$$V=\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|\hat{G}^n(e^{i\omega})-G^l(e^{i\omega})\right|^2\frac{\Phi_u(\omega)\lambda^2-\left|\Phi_{ue}(\omega)\right|^2}{\Phi_v(\omega)\lambda^2}d\omega \tag{22}$$

where $G^l(e^{i\omega})$ is the frequency response of the reduced model to be calculated. One can justify this result by observing the fact that the errors are weighted by the inverse of their variances (neglecting the factor N/n); this will lead to an asymptotically efficient (minimum variance) estimate of the frequency response. Because the high-order reduced model (observation) follows a Gaussian distribution, one could expect that the reduced model is an asymptotic maximum likelihood estimate.

The minimisation of the loss function (22) needs nonlinear search. It can be done directly in the frequency domain; it can also be converted to a time-domain parameter estimation problem.
If disturbance model is needed in control, it can be obtained by a model reduction on $1/\hat{A}^n(q)$ using the same idea as for the process model.

Order selection

For finding the best model, four "orders" need to be determined:
1) the order n of the ARX model;
2) the number of knots m of the nonlinear function;
3) the number of knots m_2 of the inverse nonlinear function;
4) the order l of the reduced linear model.

The determination of the order m_2 of the inverse nonlinear function occurs during the parameter estimation with higher order ARX model. Many models are tested with different values for the order n and the number of knots m of the nonlinear function. For the selection of the number of knots m_2 of the inverse nonlinear function is taken the same number as for the m.

Model order selection is closely related to model validation which means to check whether a model is good enough for the intended use of the model. For the selection of the order for control is used an output error criterion.

For the purpose of control it is required to obtain a model between process input and output of which the simulation error or output error is small. The simulation error is the error between the measured output of the process and the simulated output of the model. The goodness of fit is measured by the sum of the squares of the simulation error or output error:

$$V_{oe} = \frac{1}{N}\sum_{t=1}^{N}\varepsilon_{oe}(t)^2 \tag{23}$$

where

$$\varepsilon_{oe}(t) = y(t) - \frac{\hat{B}(q^{-1})}{\hat{A}(q^{-1})} = y(t) - \hat{y}(t). \tag{24}$$

For making a choice of the best order for the linear part and the nonlinear static function of the Wiener model a criterion is evaluated on the estimation data set. The criterion [ZHU 98a] is called *final output-error criterion* (FOE) and is defined as:

$$FOE_y(m_2) = \frac{N+d}{N-d}V_{yoe} = \frac{N+(2n+m_2)}{N-(2n+m_2)}\frac{1}{N}\sum_{t=1}^{N}\left\{y(t) - \hat{f}\left[\hat{G}^n(q)u(t)\right]\right\}^2 \tag{25}$$

where N is the number of the estimation data, d is the number of parameters, and V_{yoe} is the output-error (or simulation error) criterion evaluated on the estimation data set.
The factor $\frac{N+d}{N-d}$ is used to correct for the overfit effect.

When the selected order n of the linear block is higher than two, model reduction is needed in order to use the model for controller design. In generally the order n higher than two can be used.

Model validation

Model validation is to check whether the identified model is suitable for control. If parameter estimation and order selection have been performed properly, the main task of model validation is to check if the identification test is carried out properly, and, if not, provide the design of new test.

The model validation method is considered only for the linear part of the Wiener model, by assuming that the nonlinear function is known perfectly. As discussed earlier, the high order ARX model is unbiased, its frequency response follows a Gaussian distribution with variance given by (20).

Therefore, a stochastic 3-σ upper bound of the errors can be defined for the high order model

$$\left|\hat{G}^n(e^{j\omega}) - G^o(e^{j\omega})\right| \le \hat{\Delta}(\omega) = 3\sqrt{\frac{n}{N}\frac{\Phi_v(\omega)\lambda^2}{\Phi_u(\omega)\lambda^2 - \left|\Phi_{ue}(\omega)\right|^2}} \qquad \text{w.p. } 99\% \qquad (26)$$

It is recommended to use Akaike's ratio $\frac{N+d}{N-d}$ in estimating $\Phi_v(\omega)$ in order to correct for the overfit when estimation data is used. The same upper bound will also be used for the reduced model $\hat{G}^l(e^{i\omega})$, because model reduction will in general reduce the model error. Note that 3-σ bound is related to the two test design variables, input and the test time. Hence it can be used effectively in model validation and test (re)design.

Two validation methods will be proposed based on the upper bound.

Validation Method 1, Grading the Model

This is done by comparing the relative sizes of the bound and of the model. Denote $[\omega_1, \omega_2]$ as the frequency band that is important for control, then the model quality can be graded as follows:

1) Grade A, very good model, if

$$\bar{\Delta}(\omega) \le 20\%\left|\hat{G}^n(e^{j\omega})\right|, \text{ for } \omega \in [\omega_1, \omega_2].$$

2) Grade B, good model, if

$$20\%\left|\hat{G}^n(e^{j\omega})\right| \le \bar{\Delta}(\omega) \le 50\%\left|\hat{G}^n(e^{j\omega})\right|, \text{ for } \omega \in [\omega_1, \omega_2].$$

3) Grade C, marginal model, if

$$50\%\left|\hat{G}^n(e^{j\omega})\right| \le \bar{\Delta}(\omega) \le 90\%\left|\hat{G}^n(e^{j\omega})\right|, \text{ for } \omega \in [\omega_1, \omega_2].$$

4) Grade D, poor good model, if

$$\bar{\Delta}(\omega) \ge 90\%\left|\hat{G}^n(e^{j\omega})\right|, \text{ for } \omega \in [\omega_1, \omega_2].$$

Models with grade A and B can be used for process control. When models with C or D grades are obtained, often redesign of test is necessary. According to the upper bound formula (26) one can design a new test by following some rules:

Rule 1. Double the input signal amplitude will half the error over the whole frequency range. This is most effective if the signal amplitude is not yet constrained.

Rule 2. Double the test time will reduce the error by a factor of $\sqrt{2}$ over the whole frequency range. This rule can be used when the signal amplitude is already constrained.

Rule 3. If PRMS signals are used, double the average signal switch time will half the error at low frequencies and double the error at high frequencies. This rule can be used when the error is too big only at low frequencies.

Validation Method 2, Robust stability/Performance Analysis

Assume that the inverse of the nonlinear function is used in the controller to compensate for the nonlinearity. Then linear robust control theory can be used to carry out the robustness analysis for the linear part of the controller using the error bound. If the model passes the robustness test, it can be used in control. If the model cannot pass the robustness test, the analysis will indicate at which frequencies the model is not sufficiently accurate. This information will be used for the redesign of identification test.

2.3 The Hammerstein model

A Hammerstein model [ZHU 98b] is also a block-oriented nonlinear model. The Hammerstein model contains the same elements as in the Wiener model but in the reverse order. The nonlinear Hammerstein model in discrete time is given in figure 2.2.

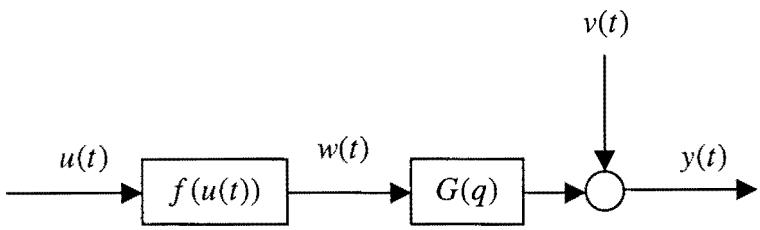


Figure 2.2 The Hammerstein model

2.4 SISO Hammerstein model identification

Since the linear and nonlinear part of the Hammerstein model is the same as in the Wiener model, the linear time-invariant transfer function $G(q)$ is parameterised in the same way as for the Wiener model. The static nonlinear function $f(u)$ can be parameterised by using a polynomial function or a cubic spline function.
The nonlinear function, parameterised by using a polynomial function, can be written as:

$$f(u(t)) = \beta_1 u(t) + \beta_2 u^2(t) + ... + \beta_m u^m(t). \tag{27}$$

For parameterisation of the linear and disturbance part is used the Box-Jenkins model. See equation (28) and (29).

$$G(q) = \frac{b_0 + b_1 q^{-1} + ... + b_n q^{-l}}{1 + a_1 q^{-1} + ... + a_n q^{-l}} = \frac{B(q)}{A(q)} \tag{28}$$

$$v(t) = H(q)e(t) = \frac{1 + c_1 q^{-1} + ... + c_n q^{-l}}{1 + d_1 q^{-1} + ... + d_n q^{-l}} = \frac{C(q)}{D(q)} e(t). \tag{29}$$

When we apply these equations to the single-input single-output (SISO) Hammerstein model we get the following structure:

$$y(t) = \frac{B(q)}{A(q)} \sum_{k=1}^{m} \beta_k u^k(t) + \frac{C(q)}{B(q)} e(t) \tag{30}$$

where:

- l is called the order of the linear part of the Hammerstein model;
- q^{-1} is the notation for the delay operator and the complex variable z^{-1};
- $H(q)$ is transfer function of a linear filter;
- $\{e(t)\}$ is a white noise sequence with zero mean value and variance λ^2;
- n is the order of the polynomials of the numerator and the denominator of $G(q)$ and $H(q)$
- $[\beta_1, \beta_1, ..., \beta_m, a_1, ..., a_l, b_0, b_1, ..., b_l, c_1, ..., c_n, d_1, ..., d_l]$ are the parameters to be estimated.

The cubic spline function is given in equation (31) with its knots $\{u_1, u_2, ..., u_m\}$.

$$w(u) = \sum_{k=2}^{m-1} \beta_k |u - w_k|^3 + \beta_m + \beta_{m+1} u + \beta_{m+2} u^2 + \beta_{m+3} u^3 \tag{31}$$

The identification procedure for the Hammerstein model consists of four steps:

1) **Test design.**
   An identification test must be designed by using excitation signals so that the parameters of the model in (.) and (.) are identifiable. If possible, the test should be designed so that the identified model is most suitable for control application. Denote the collected input/output data as:
   $$\{u(1), y(1), u(2), y(2), \cdots, u(N), y(N)\}$$
   where N is the number of samples.

For the identification of linear models, it is sufficient to consider only frequency content of the test signal like binary test signals or generalised binary noise (GBN) signals. For the identification of Hammerstein models must be used other test signals, to determine the coefficients of the nonlinear function. Since the binary signal has two levels (two amplitude values), it can only identify the nonlinear polynomial of degree two. Thus for determining the nonlinear polynomial of degree m, the amplitude of the test signal should have at least $m+1$ levels.

2) **Parameter estimation.**
The parameters of the model must be designed using the input/output data from the test by minimising the loss function:

$$V_{PE} = \sum_{t=1}^{N} \varepsilon^2(t) \tag{29}$$

where

$$\varepsilon(t) = H^{-1}(q)[(y(t) - G(q)f(u(t))] = \frac{D(q)}{C(q)}\left[y(t) - \frac{B(q)}{A(q)} \sum_{k=1}^{m} \beta_k u^k(t) \right]. \tag{30}$$

The same relaxation algorithm in section 2.2 can be used. Note that the nonlinear block can be estimated directly.

3) **Order selection.**
The orders l and m must be determined so that the obtained model is most accurate for control. Again, we use FOE criterion.

4) **Model validation.**
The identified model must be verified whether it is suitable for control. If not, redesign a new identification test. The upper error bound of linear part is used for the purpose.

2.5 Extrapolating the nonlinear function

The identified Hammerstein or Wiener model is only valid in its test range. When the model is used outside the test range, the model error can be arbitrarily large which can cause poor performance or even instability of the control system. To prevent this from happening, nonlinear function can be extrapolated manually to a larger range based on process knowledge.

The extrapolation can be done by adding additional points to outside the tested range of the nonlinear function. In the case of the Wiener model identification, only the inverse nonlinear should be extrapolated since the nonlinear function is determined from its inverse. By using process knowledge the nonlinear function can be extrapolated in any direction. The extrapolation procedure is done with Matlab and is as following. First the estimated inverse nonlinear function, with his coefficients and knots, is plotted, say with x number of points. By using a priori knowledge, the direction of the function is determined. The points, which are used for the extrapolation, say e, are then added to the x points. Then a new plot is made with these new points. The extrapolated function is fitted again with cubic splines and new coefficient and knots are determined for further use.

The points e can be added by clicking points with the mouse button in a graphical user interface (GUI) program inside the identification procedures. So the direction of the nonlinear function outsides its range can be modified.

An example is given in figure (2.3). The curve represents a nonlinear function, which is not extrapolated. The straight curve is the extrapolated nonlinear function. A smoother function can be obtained by adding more points. The extrapolation program is given in appendix 2.

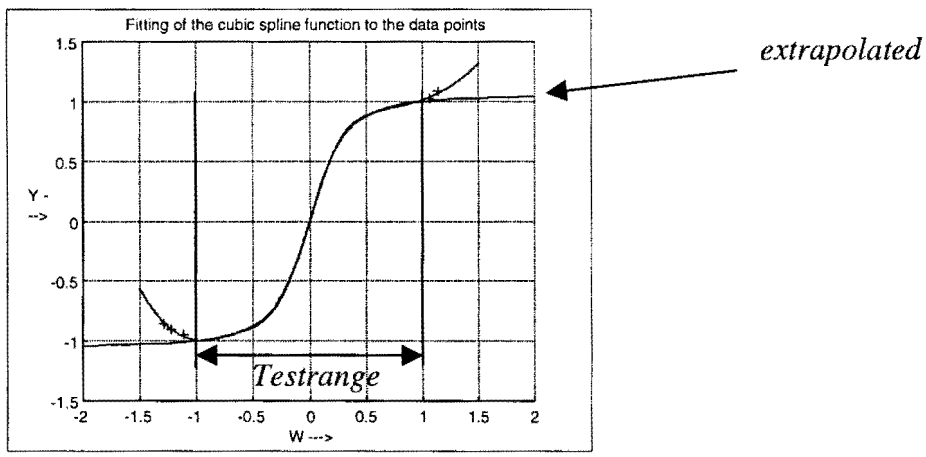


Figure 2.3. Extrapolation of nonlinear function

3 Nonlinear PID controller design

This chapter describes the nonlinear PID controller structure for the Wiener model. By compensating the nonlinear static part of the Wiener model by its inverse [ALD 96], a linear PID controller can be used for controlling the Wiener model. In this way the linear PID controller sees only the linear part of the Wiener model and linear control theory can be used. A comparable control structure can also be applied for the Hammerstein model.
The linear PID controller is designed by using the Internal Model Control approach. The robustness of these controllers can be verified by tuning the controller with different closed-loop time constant. By doing this way, the speed of the controllers can be increased or decreased.

Many nonlinear processes can be controlled succesfully with linear controllers such as PID that are tuned in the neighbourhood of an operating point. These control algorithms do not usually take into account the nonlinearities of the process and may give poor performance for certain processes. Nonlinear PID controllers can give better performance for strongly nonlinear processes. The nonlinear function describes the nonlinearities of a process. By using a-priory knowledge of a process that has to be controlled, nonlinearities of a process can be determined and extrapolated as discussed in chapter 2. The nonlinear functions cannot be represented exactly the nonlinearities by cubic splines with knots. Normally there is the nonlinearity mismatch error. The nonlinearity mismatch error depends on the number and position of the knots. The nonlinear PID controller is simulated and applied to a chemical process in chapter 5. Also a linear PID is simulated and tested on a real system. The results of these controllers are compared to each other and one can see that the nonlinear PID controller has better results.

3.1 Controller structure for Wiener model

Wiener models can be controlled by a control loop structure like in figure 3.1. This structure is discussed by [ALD 96]. The descriptions L and N represents respectively linear transfer function and nonlinear static function. The description N^{-1} represents the inverse of the nonlinear static function.

The nonlinear static part of the Wiener model is compensated by using the inverse of this nonlinear static part in the controller. This compensation will cancel the effect of the nonlinear static function and hence a linear controller can be designed for the linear part using the linear control theory.

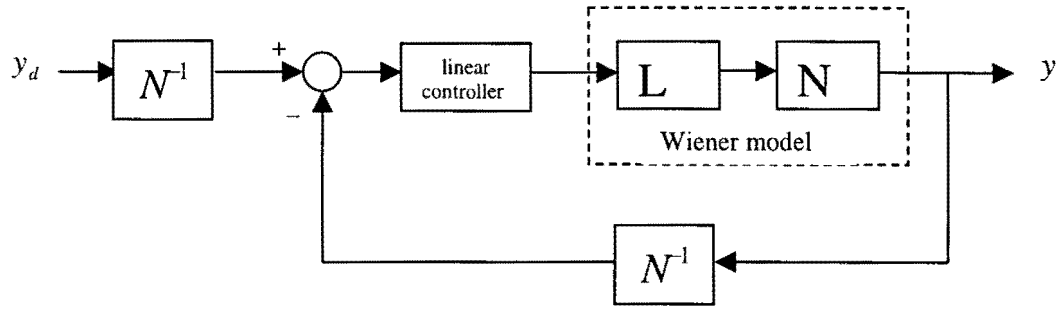


Figure 3.1 Controller structure for Wiener model

Consider the nonlinear PID control structure in more detail shown in figure 3.2, where $u(t)$ denote the process input, $w(t)$ the unmeasurable input to the nonlinear function and $y(t)$ the process output. It is assumed that the dynamic linear transfer function is stable with settling time T_s and the nonlinear static function is continous. The $y_d(t)$ is the desired output or the setpoint.

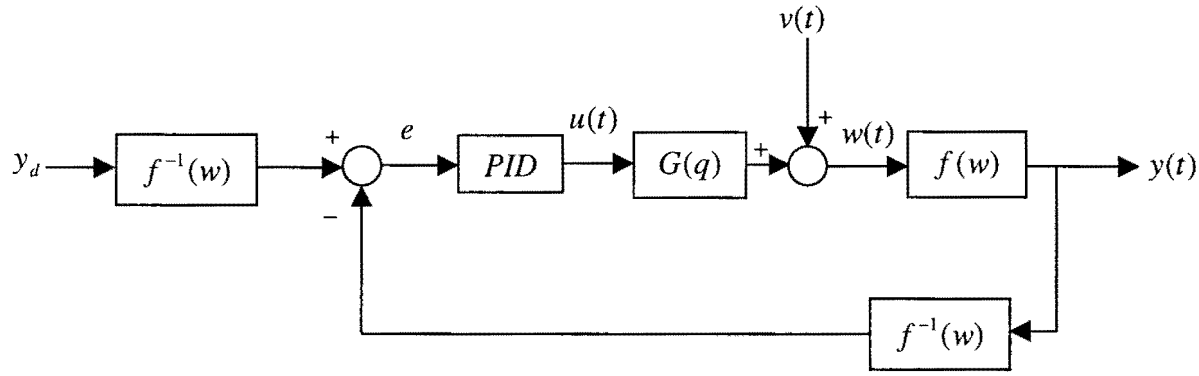


Figure 3.2 Nonlinear PID control structure for Wiener model

The inverse nonlinear static function is determined in the identification algorithm as described in the second chapter. There is also an inverse nonlinear function added after the setpoint $y_d(t)$. This is done because the output $y(t)$ passes directly through the identified inverse nonlinear function. In order to compare the setpoint with the desired output, the setpoint must also pass the inverse nonlinear function.

3.2 Controller structure for Hammerstein model

Hammerstein models can be controlled by a control loop structure like in figure 3.5. The nonlinear static part of the Hammerstein model is compensated by using the inverse of this nonlinear static part in the controller. This compensation will cancel the effect of the nonlinear

static function and hence a linear controller can be designed for the linear part using the linear control theory.

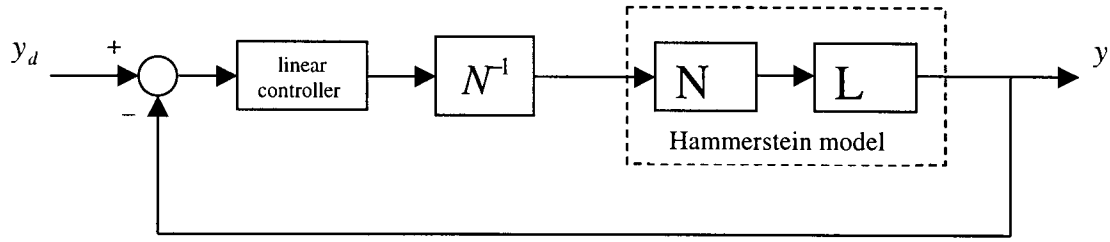


Figure 3.3 Controller structure for Hammerstein model

Consider the nonlinear PID control structure in more detail shown in figure 3.4, where $u(t)$ denote the process input, $w(t)$ the unmeasurable input to the nonlinear function and $y(t)$ the process output. It is assumed that the dynamic linear transfer function is stable with settling time T_s and the nonlinear static function is continous. The $y_d(t)$ is the desired output or the setpoint.

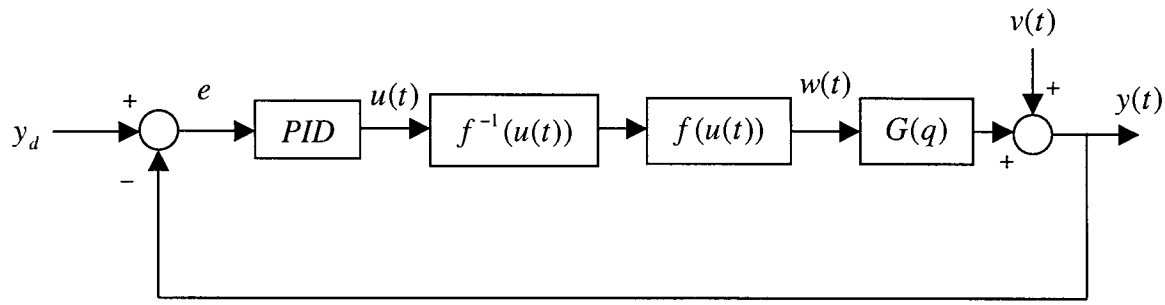


Figure 3.4 Nonlinear PID control structure for Hammerstein model

The inverse nonlinear static function is determined in the identification algorithm as described in the second chapter. For the control of the Hammerstein model is only one inverse nonlinear static function needed. When the two nonlinear functions compensate each other exactly, the control signal will be the same. The control structure for the Hammerstein model is much easier than the structure of the Wiener model.

3.3 Linear PID controller design

The PID control algorithm is the most widely used in the process industries [AST 95]. The PID controller generates a control signal $u(t)$ by operating on the error e. The error is the difference between setpoint and measured value $y(t)$. This control signal is then applied to the process. The main function of the integral action is to make sure that the process output agrees with the setpoint in steady state. With integral action, a small positive error will always lead to an

increasing control signal, and a negative error will give a decreasing control signal no matter how small the error is. The integral time constant is τ_i. The action of a controller with proportional and derivative action maybe interpreted as if the control is made proportional to the predicted process output, where the prediction is made by extrapolating the error by the tangent to the error curve (see figure 3.5). The prediction time is τ_d. The proportional action is proportional to the control error. The proportional gain is K_c.

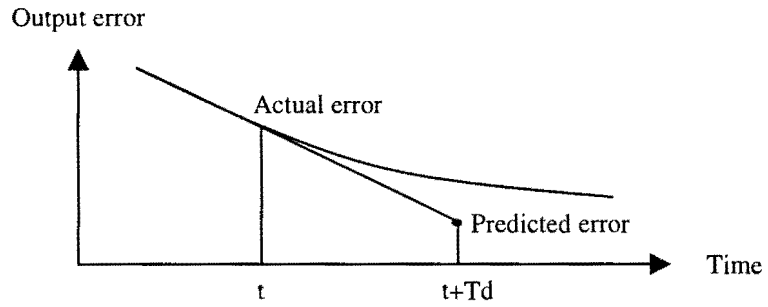


Figure 3.5. Output error curve

For the design of the linear PID controller for the process whose dynamics are represented by the first-order-plus-time-delay model is used the Internal Model Control (IMC) approach [LAM 99]. Consider the process whose dynamic behaviour is represented by:

$$Y(s) = G(s)U(s) + D(s) \tag{31}$$

with the block diagram shown in figure 3.6. Here D represents the unmeasured disturbances on the process output Y.

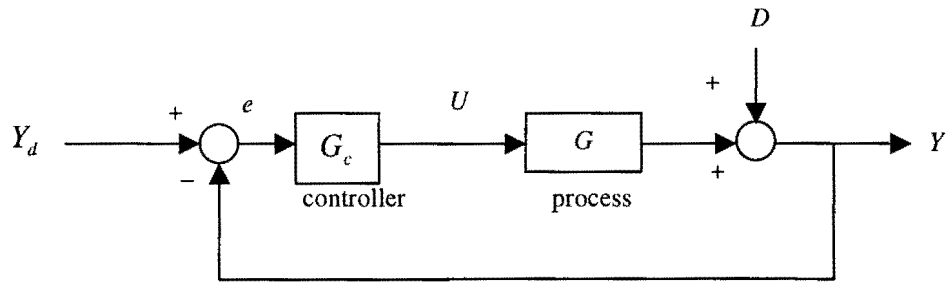


Figure 3.6 Block diagram controller and process

If it is desired to have perfect control, in which the output tracks the desired setpoint Y_d perfectly, the control action required to achieve this is obtained by substituting $Y = Y_d$ in equation (31):

$$Y_d(s) = G(s)U(s) + D(s) \tag{32}$$

and then solving for $U(s)$ to obtain:

$$U(s) = \frac{1}{G(s)}[Y_d(s) - D(s)] \tag{33}$$

If both $D(s)$ and $G(s)$ are known, then for any given Y_d, equation (33) provides the controller that will achieve perfect control. Since in reality, $D(s)$ is unmeasured, and unknown, and since $G(s)$ may not always be known perfectly, but only modelled approximately, we may now take the following strategy for implementing equation (33):

1 Assuming that $\tilde{G}$ is the best model of the plant dynamics G, then the best estimate of D is obtained by subtracting the model prediction, $\tilde{G}(s)U(s)$, from the real plant output Y to yield the estimate:

$$\tilde{D} = Y - \tilde{G}(s)U(s) \tag{34}$$

2. Choose the notation:

$$C(s) = \frac{1}{\tilde{G}(s)} \tag{35}$$

and rewrite equation (33) as:

$$U(s) = C(s)[Y_d(s) - \hat{D}(s)] \tag{36}$$

where $\hat{D}$ is the estimate of D given by equation (34). A block diagrammatic representation of equation (34) and (36) takes the form shown in figure 3.7. This structure is known as the "Internal Model Control" structure.

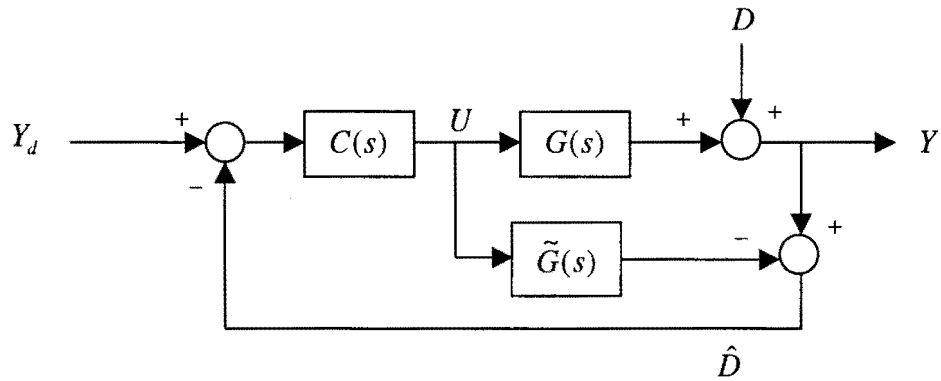


Figure 3.7 The Internal Model Control structure

From figure 3.7 the following closed-loop transfer function relation can be derived:

Relations of U to Y_d and D:

$$U = \frac{C}{1 + C(G - \tilde{G})}(Y_d - D) \tag{37}$$

Relations of Y to Y_d and D:

$$Y = D + \frac{GC}{1 + C\left(G - \tilde{G}\right)}\left(Y_d - D\right) \tag{38}$$

or

$$Y = \frac{GC}{1 + C\left(G - \tilde{G}\right)}Y_d + \frac{1 - \tilde{G}C}{1 + C\left(G - \tilde{G}\right)}D \tag{39}$$

The relationships between the conventional feedback controller (see figure 3.6 for the structure), $G_c(s)$, and the internal model controller, $C(s)$, can established as:

$$G_c(s) = \frac{C(s)}{1 - C(s)\tilde{G}(s)} \tag{40}$$

$$C(s) = \frac{G_c(s)}{1 + G_c(s)\tilde{G}(s)} \tag{41}$$

Now follows the IMC controller design procedure. First we factorises the model of the process $\tilde{G}$ into two parts:

$$\tilde{G} = \tilde{G}_+ \tilde{G}_- \tag{42}$$

where $\tilde{G}_+$ contains all the noninvertible parts (time delays and the right half-plane zeros) with a steady state gain of 1 and $\tilde{G}_-$ the invertible part.
Second we define the controller specification as:

$$C(s) = \frac{1}{\tilde{G}_-} f(s) \tag{43}$$

where $f(s)$ is a filter usually of the form:

$$f(s) = \frac{1}{(\lambda s + 1)^n} \tag{44}$$

with parameters λ and n chosen to ensure that $C(s)$ is proper (i.e., the numerator order is less than, or at most equal, the denominator order).

The IMC controller, $C(s)$, can be converted to the conventional form, $G_c(s)$, for implementation.
For first- and second-order process models $G(s)$ we select the filter $f(s)$ as:

$$f(s)=\frac{1}{\lambda s+1} \text{ or } f(s)=\frac{\left[2\lambda-\tilde{G}'_{+}(0)\right]s+1}{(\lambda s+1)^2} \tag{45}$$

for processes with an integrator.

To get a linear PID controller we have to approximate the exponential function e^{-2s}. An approximation of this function is obtained with the first-order Padé approximation:

$$e^{-\phi s} \approx \frac{1-\frac{\phi}{2}s}{1+\frac{\phi}{2}s}. \tag{46}$$

To get the controller in a PID form we write equation (42) as:

$$C(s)=\frac{1}{s}F(s) \text{ with } F(s)=\frac{s\tilde{G}_{-}(s)}{f^{-1}(s)-\tilde{G}_{+}(s)}. \tag{47}$$

The equation (46) of $C(s)$ can be expanded in a Taylor series in s and yields:

$$C(s)=\frac{1}{s}\left[F(0)+F'(0)s+\frac{F''(0)}{2}s^2+\ldots\right]. \tag{48}$$

In the expansion we can see that the first three terms can be interpreted as an ideal PID controller:

$$C(s)=K_c\left(1+\frac{1}{\tau_i s}+\tau_d s+\ldots\right) \text{ with } K_c=F'(0),\ \tau_i=\frac{F'(0)}{F(0)} \text{ and } \tau_d=\frac{F''(0)}{2F'(0)}. \tag{49}$$

The approximation is exact if the order of the controller is one or two.

The PID algorithm in equation (48) is seldom used in practice because much better performance is obtained by the modified algorithm. The modified PID controller is described by:

$$C(s)=K'_c\left(1+\frac{1}{s\tau'_i}\right)\left(1+s\tau'_d\right). \tag{50}$$

The controller given by equation (48) is called non-interacting, and the one given by equation (49) interacting. The reason is that in the controller (48) the integral time τ_i does not influence the derivative part, and the derivative time τ_d does not influence the integral part (see equation (49)). The parts are thus non-interacting. In the interacting controller, the derivative time τ'_d does influence the integral part. Therefore, the parts are interacting. The interacting controller can always be presented as a non-interacting controller, whose coefficients are given by:

$$K_c = K_c' \frac{\tau_i' + \tau_d'}{\tau_i'} \tag{51}$$

$$\tau_i = \tau_i' + \tau_d' \tag{52}$$

$$\tau_d = \frac{\tau_i' \tau_d'}{\tau_i' + \tau_d'} \tag{53}$$

An interacting controller of the form (49) corresponds to a non-interacting controller (48) can be found only if:

$$\tau_i \geq 4\tau_d .$$

Then,

$$K_c' = \frac{K_c}{2}\left(1 + \sqrt{1 - 4\tau_d/\tau_i}\right) \tag{54}$$

$$\tau_i' = \frac{\tau_i}{2}\left(1 + \sqrt{1 - 4\tau_d/\tau_i}\right) \tag{55}$$

$$\tau_i' = \frac{\tau_i}{2}\left(1 + \sqrt{1 - 4\tau_d/\tau_i}\right). \tag{56}$$

Thus the coefficients of the linear PID controller, in interacting form, are calculated in this way. For the implementation of the linear PID controller is used an modified interacting form:

$$C(s) = K_c'\left(1 + \frac{1}{s\tau_i'}\right)\left(\frac{1 + s\tau_d'}{1 + s\tau_d'\alpha}\right) \tag{57}$$

where α is a small quantity, typically less than 0.1.
To obtain the exact derivative of a signal is impossible in practice; therefore the derivative term of the controller (49) is modified. The modification can be interpreted as the ideal derivative filtered by a first-order system with the time constant $\tau_d'\alpha$.

In this way the PID controller algorithm can be implemented on the computer. The algorithm of the controller is implemented in discrete-time. The controller structure is given in figure 3.8:

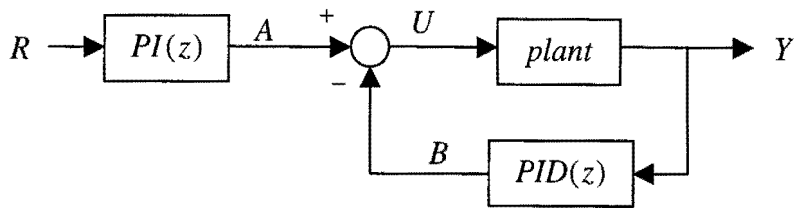


Figure 3.8 The PID controller structure

There is added a PI controller, to obtain a smooth signal for the desired signal Y_d. When the setpoint (or desired signal) makes a stepchange, thus when the input signal is a step signal, the difference between the output signal and the setpoint becomes too large. Therefore the setpoint signal is made smooth to get a smaller difference.

The structure shown in figure 3.8 can be transformed in another form that is more useful. This structure has a separated derivative action, see figure 3.9. The reason is that when a step change is made in the setpoint, the derivative action makes a sudden change in controller output – called "derivative kick." To avoid this kick, the derivative action is separated [OGU 94].

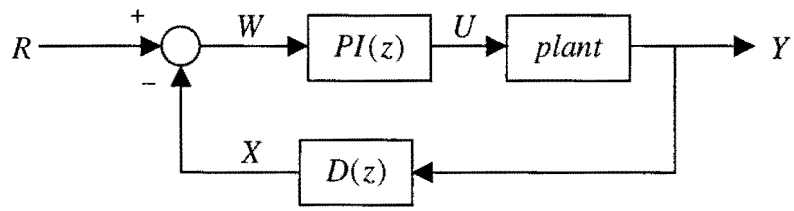


Figure 3.9 Transformed PID controller structure

To show that these two controller structures (from figure 3.8 and 3.9) are the same, we give the closed-loop transfer function relation of each structure:

Relations of R to Y for the controller structure in figure 3.8:

$$\frac{Y}{R} = PI\frac{p}{1+PID.p} \tag{58}$$

Relations of R to Y for the controller structure in figure 3.9:

$$\frac{Y}{R} = \frac{PI.p}{1+PI.D.p} \tag{59}$$

4 The pH process

The pH process is considered in the literature [PAJ 84] to have the structure of a Wiener model with the linear element describing the mixing dynamics of the reagent streams of a continuous stirred tank reactor (CSTR). The nonlinear element represents a static nonlinear titration curve, which gives pH as a function of the chemical species. This chapter describes the pH process and a mathematical representation of this process is derived. The pH process is modelled as CSTR by using mass balances [LAM 98]. The model that is derived is not used for the purpose of controlling but for analysing the dynamic behaviour of that process. This model of the pH process is actually used as a-priori information for designing the identification experiments.

Since this process is highly nonlinear it is useful for representing it as a Wiener model. The pH process exhibits a logarithmic behaviour.

4.1 Description of the pH process

The pH process consists of a continuous stirred tank reactor (CSTR) with two input streams and one output stream. The scheme is shown in figure 4.1.

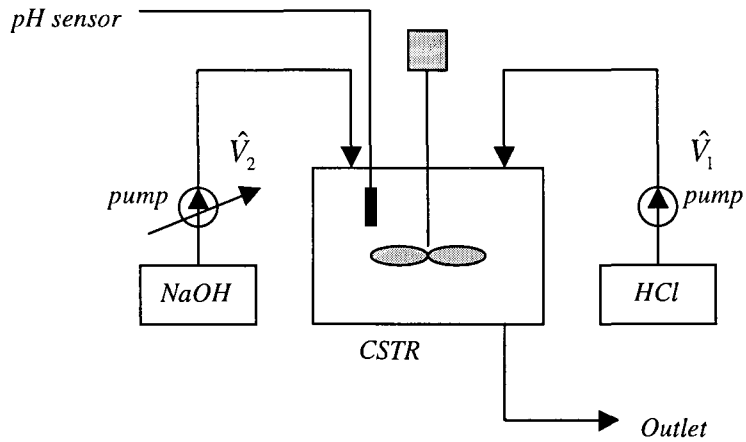


Figure 4.1 The pH process scheme

The first input flow consists of solution of strong acid (HCl) and the second flow consists of a solution of strong base $(NaOH)$. The first flow has a constant given volumetric rate $(\hat{V}_1)$ and the rate of the second flow $(\hat{V}_2)$ adjusted using a controlled pump. These two flows react with each other and produce a pH value. The pH of the solution inside the CSTR is measured by using a pH sensor.

The base flow rate $(NaOH)$ is used to control the pH value of the solution inside the tank. The acid flow (HCl) is held constant or acts as a disturbance.

4.2 Physical modelling of the process

For controlling or understanding a process there is information needed about the dynamic behaviour of a process: time-dependent behaviour of a process. To obtain this information we need a mathematical model of the process, which describes the dynamic behaviour of the process. This mathematical representation of the process is usually given in terms of a set of mathematical equations (differential), whose solution yields the dynamic behaviour of the process.

The construction of a model for a process depends on the application for which the model is to be used. Different models are used for different purposes. For example, a model that is used for the control of a process shall be different from a model that is used for the analysis of the same process. Here the model is constructed for analysing the process. The process is modelled as a CSTR in which the reaction of the acid and base take place.

The process variable pH is the negative logarithm of the hydrogen ion (c_{H^+}) concentration. Equation (60) mathematically describes the definition of pH.

$$pH = -\log_{10}(c_{H^+}) \tag{60}$$

$$k_w = c_{H^+} \cdot c_{OH^-} \tag{61}$$

with

k_w = *water equilibrium constant* $(\approx 10^{-14}\ mol^2/l^2)$

c_{H^+} = *concentration of* H^+ *in the reactor* $[mol/l]$

c_{OH^-} = *concentration of* OH^- *in the reactor* $[mol/l]$

Hydrogen ions exist in all streams that contain water or an acid, and can be visualised as a single, positive charged nucleus, which is a proton (H^+). Equation (61) shows that the product of the hydrogen and hydroxyl ion (c_{OH^-}) concentration is a constant for water at a given temperature. This constant can be applied to all reactions involving aqueous solutions, and it will become the basis for the pH scale.
In a neutral solution (pure water) these two concentrations are equal: the H^+ concentration is equal to $10^{-7}\ [mol/l]$ and the OH^- concentration is equal to $10^{-7}\ [mol/l]$.

The pH scale is a way to classify how strong an acid or base is. The pH scale is a scale from 0 to 14. On the scale, 0 is the strongest acid, 14 is the strongest base, and a neutral solution is located at 7.

A solution in which the concentration of hydrogen ion is higher is called an acidic solution (pH<7); and one in which the hydroxyl ion is higher is said to be alkalic (pH>7).

The electroneutrality balance is written as follows:

$$\sum_i c_i \cdot Z_i = 0 \tag{62}$$

with $c_i =$ *concentration of ion i* $\left[mol/m^3\right]$

$Z_i =$ *charge of ion i.*

The charge of the hydroxyl ion is equal in magnitude and opposite in sign to the hydrogen ion. At pH value of 7, the hydrogen ion concentration equals the hydroxyl ion concentration so that the net charge of the water is zero.
Figure 3.2 shows that the hydroxyl ion concentration increases by a factor of 10 for each unit increase in pH [MCM 94].

pH	Hydrogen ion	Hydroxyl ion
0	1.0	0.00000000000001
1	0.1	0.0000000000001
2	0.01	0.000000000001
3	0.001	0.00000000001
4	0.0001	0.0000000001
5	0.00001	0.000000001
6	0.000001	0.00000001
7	0.0000001	0.0000001
8	0.00000001	0.000001
9	0.000000001	0.00001
10	0.0000000001	0.0001
11	0.00000000001	0.001
12	0.000000000001	0.01
13	0.0000000000001	0.1
14	0.00000000000001	1.0

Figure 3.2 pH value table for hydrogen and hydroxyl ion concentration

Starting from the mass balances for the construction of the mathematical model of the CSTR, the total mass balance of the CSTR (in kg.):

$$\frac{dm}{dt} = \hat{m}_1 + \hat{m}_2 - \hat{m}_3 = \rho_1 \hat{V}_1 + \rho_2 \hat{V}_2 - \rho_3 \hat{V}_3 \tag{63}$$

with $\hat{m}_1$, the mass component of the first inlet flow in kg,

$\hat{m}_2$, the mass component of the second inlet flow in kg,

$\hat{m}_3$, the mass component of the outlet flow in kg.

Assuming that the densities of the inlet flows and of the reactor are the same, the mass balance can be written as:

$$\frac{dV}{dt} = \hat{V}_1 + \hat{V}_2 - \hat{V}_3 \qquad \text{with } m = \rho \cdot V \tag{64}$$

The component mass balances (in moles) for the four involved components, with $n_i = c_i \cdot V$:

for H^+: $$\frac{dn_{H^+}}{dt} = c_{H^+,1}\hat{V}_1 + c_{H^+,2}\hat{V}_2 - c_{H^+}\hat{V}_3 \tag{65}$$

for OH^-: $$\frac{dn_{OH^-}}{dt} = c_{OH^-,1}\hat{V}_1 + c_{OH^-,2}\hat{V}_2 - c_{OH^-}\hat{V}_3 \tag{66}$$

for Na^+: $$\frac{dn_{Na^+}}{dt} = c_{Na^+,2}\hat{V}_2 - c_{Na^+}\hat{V}_3 \tag{67}$$

for Cl^-: $$\frac{dn_{Cl^-}}{dt} = c_{Cl^-,1}\hat{V}_1 - c_{Cl^-}\hat{V}_3 . \tag{68}$$

The neutralization reaction inside the tank is described by the reversible chemical reaction:

$$H^+ + OH^- \Leftrightarrow H_2O .$$

The electroneutrality balance of the CSTR:

$$c_{H^+} + c_{Na^+} = c_{OH^-} + c_{Cl^-} \tag{69}$$

Now we can write an expression for c_{H^+} as a function of c_{Na^+} and c_{cl^-} by substituting the equations (64) and (69). This yields:

$$c_{H^+} = \frac{-(c_{Na^-} - c_{Cl^-}) + \sqrt{(c_{Na^-} - c_{Cl^-})^2 + 4k_w}}{2} \tag{70}$$

The state variables, which characterise the behaviour of the CSTR:

$$\underline{x(t)} = \begin{bmatrix} c_{Na^+} \\ c_{cl^-} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} . \tag{71}$$

Since we are interested in the pH value of the reactor that is caused by the manipulated inlet flow $NaOH$, the input and output variable are respectively defined as:

$$\underline{u(t)} = \left[\hat{V}_2\right] = [u] \text{ and } \underline{y(t)} = [pH] = [y]. \tag{72}$$

The inlet flow HCl can be considered as the disturbance of the model. Since the pH of this flow is relatively low, we can approximate $c_{cl^-,1} = c_{H^+,1}$, so that:

$$\underline{d(t)} = \left[c_{cl^-,1}\right] = [d]. \tag{73}$$

The state-space equations for c_{Na^+} and c_{cl^-} are derived and are as follow:

$$\frac{dc_{Na^+}}{dt} = \frac{\left(c_{Na^+,2} - c_{Na^+}\right)\hat{V}_2 - c_{Na^+}\hat{V}_1}{V} \tag{74}$$

$$\frac{dc_{cl^-}}{dt} = \frac{\left(c_{cl^-,1} - c_{cl^-}\right)\hat{V}_1 - c_{cl^-}\hat{V}_2}{V} \tag{75}$$

The meaning of the used notations:

c_{Na^+} = concentration of Na^+ in the reactor

c_{cl^-} = concentration of Cl^- in the reactor

$c_{Na^+,2}$ = concentration of Na^+ of the inlet flow of the $NaOH$ -solution

$c_{cl^-,1}$ = concentration of Cl^- of the inlet flow of the HCl -solution

V = volume of the contents of the reactor

$\hat{V}_i$ = volumetric flowrate of flow i.

The derived equations (74) and (75) can be written in the vector form:

$$\frac{d}{dt}\underline{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \underline{f(x,u)} = \begin{bmatrix} f_1(x_1,u) \\ f_2(x_2,u) \end{bmatrix} = \begin{bmatrix} \dfrac{(c_2 - x_1)u - x_1 c_1}{V} \\ \dfrac{(d - x_2)c_1 - x_2 u}{V} \end{bmatrix} \tag{76}$$

where $c_1 = \hat{V}_1$ and $c_2 = c_{Na,2}$ are constants and assumed constant during the operation.
The output vector as a function of the states of the model is written as:

$$\underline{y} = [g(x_1,x_2)] = \left[-\log\left(\frac{-(x_1 - x_2) + \sqrt{(x_1 - x_2)^2 + 4k_w}}{2} \right) \right]. \tag{77}$$

Equation (76) and (77) represents a nonlinear state space model of the pH process.

4.3 The pH process approximated as a Wiener model

The pH process in the CSTR can be decomposed into two separate parts: a static nonlinear relation (the state/output map) and a mildly nonlinear (bilinear) dynamic part [NOR 98].
The static part of the pH process manifests itself as the titration curve of the pH process. The titration curve is the most important for the design and troubleshooting of pH control systems. The titration curve is a plot with pH for the Y-axis and the difference between the acid and base concentration for the abscissa. For a strong acid that provides a single hydrogen ion from each molecule and for a strong base that provides a single hydroxyl ion from each molecule, the difference in acid and base concentration equals the difference between hydrogen and hydroxyl ion concentration. Figure 4.3 shows a titration curve for a strong acid and strong base with the abscissa labelled both with acid and base and hydrogen and hydroxyl ion concentration. The titration curve is symmetrical about the 7 pH point, which is the neutral point. Note that when the difference in acid and base or hydrogen and hydroxyl concentration is zero, the slope is steepest.

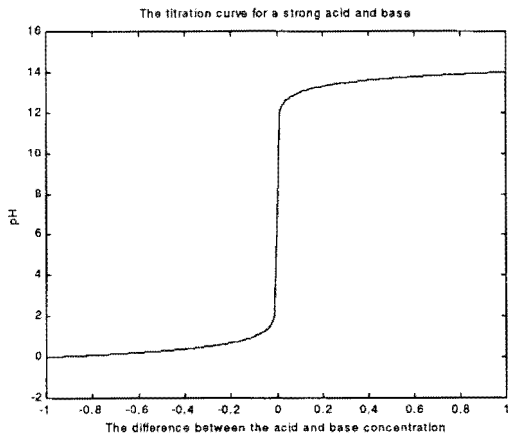


Figure 4.3 Titration curve

The pH process is considered to have the structure of a Wiener model. The nonlinear static part of the pH process represents the static nonlinear titration curve.
Denote the input of the nonlinear static function as $w = x_1 - x_2$ and by substituting this in equation in (77), yields:

$$y = f(w) = -\log\left(\frac{-w + \sqrt{w^2 + 4k_w}}{2}\right). \tag{78}$$

The dynamic part of the pH process, which is given by the state-space equations for the concentrations of Na^+ and Cl^- in the reactor in (74) and (75), can be approximated by the linear part of the Wiener model.

Although the CSTR is the major dynamic element, there are often elements whose dynamics behaviour affects that of the total system. These elements include the final control element, the electrode system and a small dead time due the mixing, injection and measurement time lags. It will be tedious to include all these effects in physical modelling. On he other hand, an identified pH process model will include these dynamics if they are significant.

5 Identification and control of the pH process

The goal of the identification procedure is to obtain a model that can approximate the pH process in a CSTR. The pH process is nonlinear as described in chapter 4. The SISO Wiener model identification technique (chapter 2) is applied to the pH process. The closed-loop identification experiment is designed by using a linear PID controller in the control loop. The tuning method for the linear PID controller is treated. The way of implementing of the controller is described.

The experimental setup (for identification) and preliminary tests are discussed. These tests include plots of step responses and what is necessary the input and output signals for the identification algorithm. The choice of the input signals for the identification tests are discussed. The various Wiener models are validated by using an error criterion.

The nonlinear PID controller is applied to the pH process that approximates the Wiener model. Also the linear PID is proposed to the pH process. The controllers are first simulated and compared to each other in Simulink block diagrams in Matlab. After that, the linear and the nonlinear PID controllers are applied to the pH process and experiments are made to determine the performance of the nonlinear PID controller.

5.1 The identification experiment design

The experimental setup is depicted in figure 5.1. The basic principle of the setup is: a constant flow of *HCl* solution enters the continuously stirred tank reactor (CSTR). A pH sensor is used to measure the pH value of the CSTR contents. The pH sensor measures the pH from 0 to 14. The CSTR is continuously injected with a *NaOH* solution to keep the pH value at a certain setpoint.

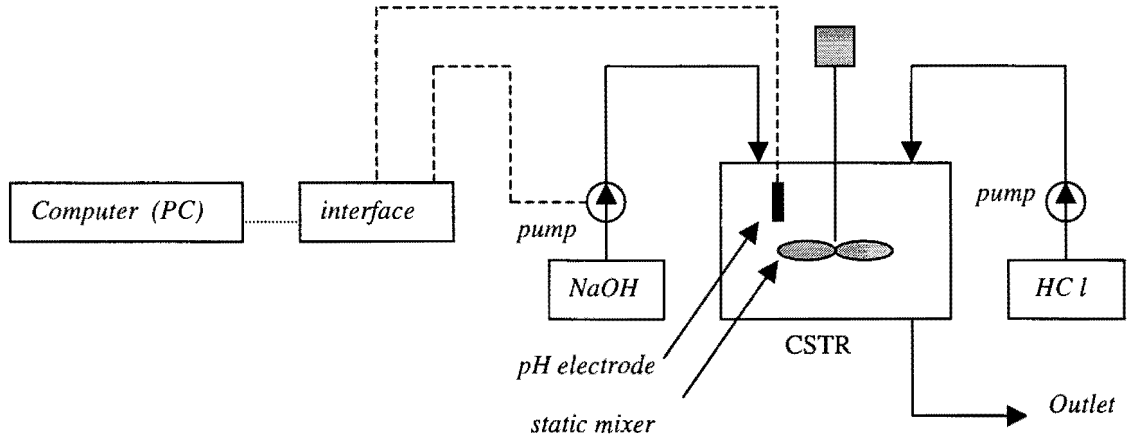


Figure 5.1 The experimental setup

For the control of the *NaOH* flow is used a personal computer in combination with an interface. The interface is connected to the process (CSTR). The *NaOH* flow is controlled with a pump that is connected to the interface. The interface sends a control signal between 0 and 3 Volts to the *NaOH* pump. A signal in this range causes the pump to give a flow in the range of 0 – 20 ml/s linearly.

The pH sensor continuously measures the relevant pH value and the corresponding sensor signal is determined such that it can be read by the A/D converter (in the interface) and converted from continuous-time to discrete-time signals. The processor in the computer handles the measurement data and computes control actions for the actuator (the final control element represents the actuator). The D/A converter (in the interface) translates the discrete commando's into analog signals that enters the actuator.

5.2 The identification scheme

The identification of the pH process is carried out in closed-loop operation by using a linear PID controller. The experiment is represented by the block scheme in figure 5.2.

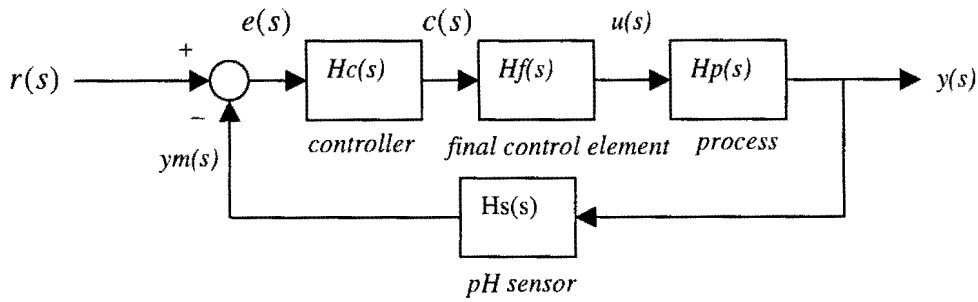


Figure 5.2 Block scheme of the experiment

In the CSTR it is assumed that the *HCl* flow is constant. There is only one input and one output, thus a SISO system. The input is the *NaOH* flow and the output is of the system is the pH. The disturbance, which is not drawn in the scheme, is the change in the *HCl* flow. The controller is the PC, the final control element the *NaOH* pump and the sensor the pH electrode.
In figure 4.1 we see that the controller can only influence the *NaOH* flow.

Since the processor in the computer operates with floating-point numbers, the measured error $(e(s))$ between the reference $(r(s))$ and the output of the plant is converted into digital form by the A/D converter at sampling interval k defined by the synchronisation clock. The computer

interprets the converted signal $(e(s))$ as sequence of numbers that it processes using a control algorithm, and generates a new sequence of numbers $(u(k))$representing the control. By means of the D/A converter, this sequence is converted into an analog signal, which is maintained constant between the sampling interval by a zero order hold (ZOH). The set of A/D converter, computer and D/A converter behave the same as an analog controller (such as a PID type), when using the use of a high sampling frequency. In this way the controller algorithm can be implemented on the computer. The algorithm of the controller is implemented in discrete-time.

The controller is a feedback controller. The sensor measures the pH, which is the output of the system $(ym(s))$ and sends it to the PC with a time lag through the interface. In the computer the error is computed. The error is the difference between the setpoint (=reference or the desired output) and the actually measured pH value of the system (=CSTR).
After the computer (=controller) has determined the error it decides what to do: whether to increase the *NaOH* flow or to decrease it to reach the desired pH value. For doing this the computer uses the control algorithm.

Before designing the linear PID, which will be used for the closed-loop identification of the pH process, an empirical model of the pH process must be developed. We need also information about the sampling frequency for data collection.
When we choose a sampling frequency that is too low then it will lead to loss of information, since the dynamic of the process may be then too fast to be observed. A suitable choice for the sampling frequency can be found with:

$$Ts = \frac{T_s}{30 \sim 100} \qquad \text{with } T_s = \text{settling time.} \tag{79}$$

Most chemical processes can be approximated by first-order-plus-time-delay:

$$Hp(s) = \frac{y_m(s)}{c(s)} = \frac{Ke^{-t_d s}}{\tau s + 1} \tag{80}$$

with K = gain
t_d = deadtime
τ = time constant.

To get these values of the model in equation (80), an open-loop experiment is done. These open-loop experiments are done by putting in a step disturbance at the input of the final control element $(c(s))$ and record the output variable $(ym(s))$as a function of time.
This measured output data $ym(s)$ is fitted to the equation in (80). This open-loop experiment is done with different step sizes. The results of the experiment are shown in figure 5.3.

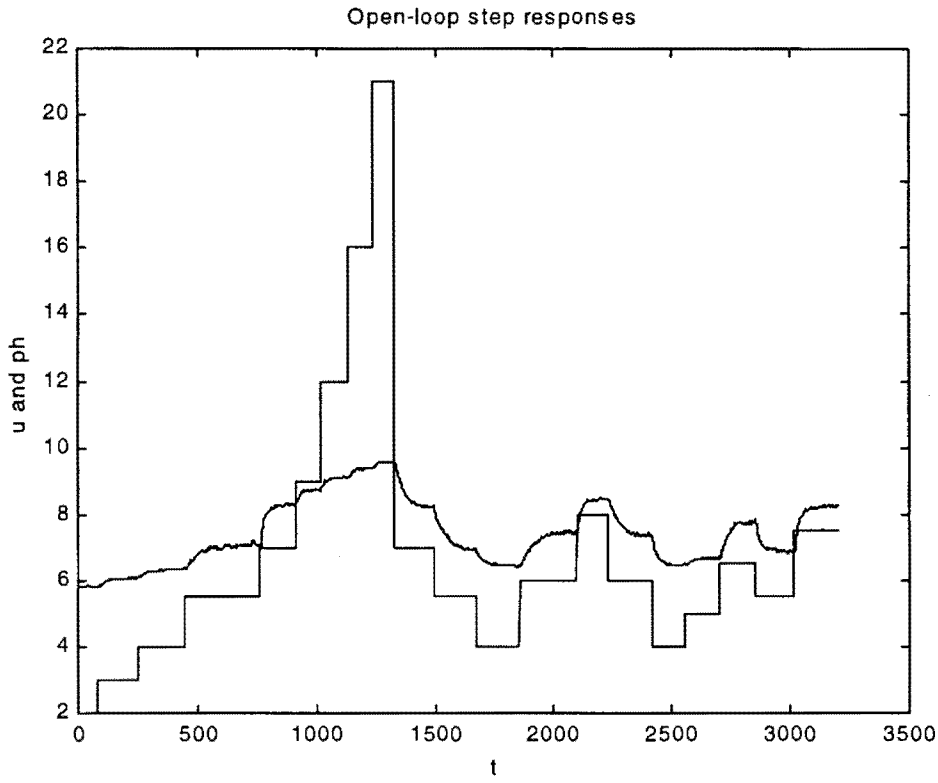


Figure 5.3 The open-loop experiment result

Inside this the block shaped signal represents the output signal of the final control element, thus $u(s)$ or the process input. The other signal, obtained from the pH sensor, represents the process output $ym(s)$ or the real pH value.

One can see that the step responses are nonlinear. From this we can find the first-order-plus-time-delay model parameter values by taking the step response with the highest time constant and fitting the step response to the process model (80).

The best first-order-plus-time-delay model parameter values that is found:

gain K = 0.4910
deadtime t_d = 2sec
time constant $\tau = 26.7284$

$$Hp(s) = \frac{y_m(s)}{c(s)} = \frac{Ke^{-t_d s}}{\tau s + 1} = \frac{0.4910e^{-2s}}{26.7284s + 1} \tag{81}$$

By considering the first-order-plus-time-delay model, the settling time T_s and the Ts is determined as:

$$T_s = 4 \cdot \tau + \tau_d = 4 \cdot 26.7284 + 2 = 108.9136 \text{ sec.}$$

$$Ts = \frac{T_s}{30 \sim 100} = \frac{108.9136}{30 \sim 100} = 3.63 \sim 1.089 \text{ sec.} \approx 1 \text{ sec. thus the sample frequency is 1 Hz.}$$

With these results started with the design of the linear PID controller, which will be used for the identification.

Now follows a brief description of the implementation procedure [LAM 99] of the linear PID controller structure from figure 3.9. First the coefficients of the IMC parameters K_c, τ_i and τ_d are calculated with the formulas (80). The calculations are made with Matlab files.
The parameter λ of the filter in (44) that can be adjusted during the calculations of the IMC parameters determines the speed of the response of the system. The factor $1/\lambda$ is approximately equal to the closed-loop bandwidth, which must always be smaller than the bandwidth over which the process model is valid.

Second, the controller structure from figure 3.9 in s-domain is transformed into z-domain.

The differential equations of the PI part can be written as:

$$PI(s) = K\left(\frac{\tau_i s + 1}{\tau_i s + a}\right) \tag{82}$$

$$\text{with } s = \frac{\ln(z)}{T_s} \Rightarrow PI(z) = K_d \cdot \left(\frac{\frac{\tau_i \ln(z)}{T_s} + 1}{\frac{\tau_i \ln(z)}{T_s} + a}\right) = K_d \cdot \left(\frac{\ln(z) + \frac{T_s}{\tau_i}}{\ln(z) + \frac{aT_s}{\tau_i}}\right) = K_d \cdot \left(\frac{z - e^{-\frac{T_s}{\tau_i}}}{z - e^{-\frac{aT_s}{\tau_i}}}\right), \tag{83}$$

(the last step is obtained by finding the roots for the numerator and the denominator like $\ln(z) + \frac{T_s}{\tau_i} = 0 \Rightarrow z = e^{-\frac{T_s}{\tau_i}}$ and writing it as $z - e^{-\frac{T_s}{\tau_i}}$)

and applying of $H(s)\big|_{s=0} = H(z)\big|_{z=1}$ we get $PI(s)\big|_{s=0} = PI(z)\big|_{z=1} \Rightarrow \frac{K}{a} = K_d \cdot \left(\frac{1 - e^{-\frac{T_s}{\tau_i}}}{1 - e^{-\frac{aT_s}{\tau_i}}}\right)$

$$\Rightarrow K_d = \left(\frac{1 - e^{-\frac{aT_s}{\tau_i}}}{a}\right)\left(\frac{K}{1 - e^{-\frac{T_s}{\tau_i}}}\right) \qquad \text{and if } a \to 0: \; K_d = \lim\left(\frac{1 - e^{-\frac{aT_s}{\tau_i}}}{a}\right)\left(\frac{1}{1 - e^{-\frac{T_s}{\tau_i}}}\right),$$

$$\Rightarrow K_d = \left(\frac{\frac{T_s K}{\tau_i}}{1 - e^{-\frac{T_s}{\tau_i}}} \right).$$

$$\text{Thus } PI(z) = K_d \cdot \left(\frac{z - e^{-\frac{T_s}{\tau_i}}}{z - e^{-\frac{aT_s}{\tau_i}}} \right) = \left(\frac{\frac{T_s K}{\tau_i}}{1 - e^{-\frac{T_s}{\tau_i}}} \right) \left(\frac{z - e^{-\frac{T_s}{\tau_i}}}{z - 1} \right) = \left(\frac{\frac{T_s K}{\tau_i}}{1 - e^{-\frac{T_s}{\tau_i}}} \right) \left(\frac{1 - e^{-\frac{T_s}{\tau_i}} z^{-1}}{1 - z^{-1}} \right). \tag{84}$$

For the D action we can do the same way as for the PI controller, the result of this:

$$D(s) = \left(\frac{\tau_d s + 1}{\tau_d \alpha s + 1} \right) = \frac{x(s)}{y(s)} \tag{85}$$

$$D(z) = K_D \cdot \left(\frac{z - e^{-\frac{T_s}{\tau_d}}}{z - e^{-\frac{T_s}{\tau_d \alpha}}} \right) = \left(\frac{1 - e^{-\frac{T_s}{\tau_d \alpha}}}{1 - e^{-\frac{T_s}{\tau_d}}} \right) \left(\frac{1 - e^{-\frac{T_s}{\tau_d}} z^{-1}}{1 - e^{-\frac{T_s}{\tau_d \alpha}} z^{-1}} \right). \tag{86}$$

The corresponding differential equations of the $PI(z)$ and $D(z)$ are:

$$PI(z) = \frac{u(z)}{w(z)} \rightarrow \quad u(n) = k_{PI}.w(n) - k_{PI}.e^{-\frac{T_s}{\tau_i}}.w(n-1) + u(n-1) \tag{87}$$

$$D(z) = \frac{x(z)}{y(z)} \rightarrow \quad x(n) = k_D.y(n) - k_D.e^{-\frac{T_s}{\tau_d}}.y(n-1) + e^{-\frac{T_s}{\tau_d \alpha}}.x(n-1). \tag{88}$$

In these forms the differential equation (85) and (86) are implemented in the computer by using a program that is called LabView. This program uses a graphical programming language to create programs in block diagram form. LabView has extensive libraries of functions and subroutines for most programming tasks. LabView includes conventional program development tools, so you can set breakpoints, animate the execution to see how data passes through the program, and single-step through the program to make debugging and program development easier.
The total scheme of the implementation of the nonlinear PID controller is given in appendix 1.

5.3 Choice of the input signal for identification

As described in the second chapter, the excitation signals for the identification test must be designed so that the parameters of the model are identifiable. These tests should be designed so that the identified model is most suitable for control application.
The shape of the test signals used in the identification experiment has an influence on the results. In order to determine which type of excitation signals can be used to identify coefficients of the nonlinear static part of the Wiener model the two following test designs are made.

1. The generalised binary noise (GBN) test: this like a PRBS (pseudo-random binary sequences), they are sequences of rectangular pulses, modulated in width, that approximate a discrete-time white noise and thus have a spectral content rich in frequencies. The length of the sequence and the average switching time of the GBN can be defined. The average switching time of the GBN signals is between $T_s/4$ and T_s, where T_s denotes the process 98% settling time. The GBN signal has only two amplitude levels, which can be defined.

2. Staircase signals as excitation signals: This type of input signals consists of different amplitude levels and different width of the stairs. By using a-priori knowledge, obtained from the open-loop step responses, the input range of this signal can be defined properly. Also the length of the sequence, the width of the stairs and the number of the stairs can be defined. Denote $\tau \approx T_s/4$, then the width of the stairs can be chosen as a multiple of the rise time τ of the process. This excitation signal will lead to identifying more nonlinearities of the process.

A part of the designed excitation signals is given in figure 5.4.

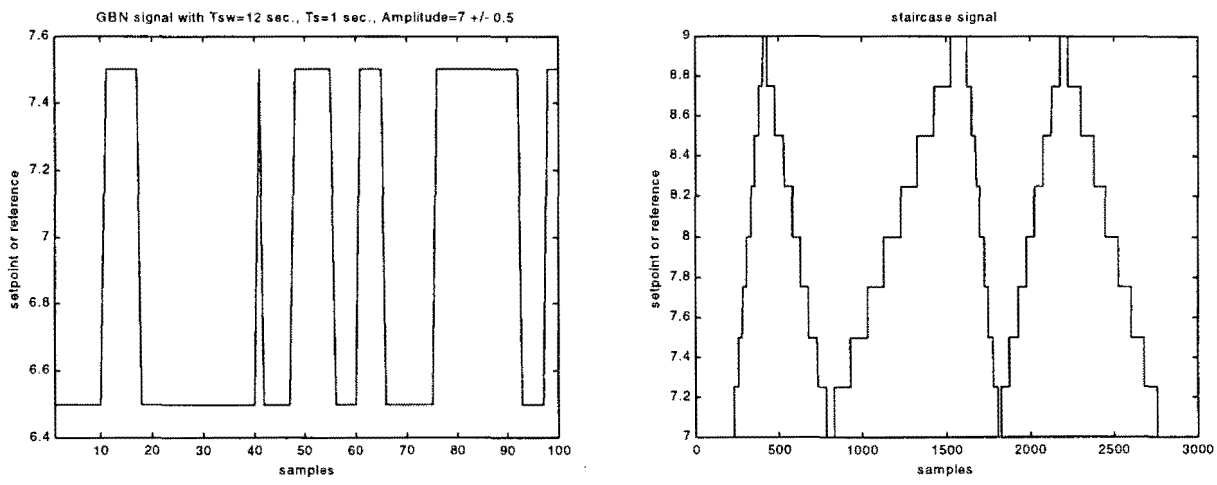


Figure 5.4 excitation signals

5.4 Identification results

Using the test signals, the actual identification experiment is done. First the identification experiments are done with GBN signals with different switching times and are excited with different reference or setpoint values. In the table in figure 5.5 the GBN signals are given. The adjustments for the identification experiment are as following. For the pH process: the constant stream of the HCl solution was adjusted on 14 ml/s, the volume of the tank $V = 514ml$, the pH values for the solutions: $NaOH \rightarrow 10pH$ and $HCl \rightarrow 4pH$.
The coefficients of the linear PID controller:

$K_c = 9.02728$, $\tau_i = 26.7284$ $\tau_d = 0.3249$ and $\lambda = 4$.

Experiment number	Number of samples	Sampling time Ts	Average switching time Tsw	Amplitude
1	2500	1 sec.	12 sec.	7 +/- 0.5
2	2500	1 sec.	12 sec.	7 +/- 0.75
3	2500	1 sec.	12 sec.	7 +/- 0.5
4	2500	1 sec.	12 sec.	7 +/- 0.75
5	2500	1 sec.	10 sec.	7 +/- 0.5
6	2500	1 sec.	10 sec.	7 +/- 0.75
7	2500	1 sec.	12 sec	7 +/- 0.5
8	2500	1 sec.	12 sec.	7 +/- 0.75

Figure 5.5 GBN test signal adjustments

We have thus eight data sets of the input and output of the pH process. The data sets are used to estimate the models of the pH process. In order to analyse the results, some experiments are repeated again. Each data set is divided into an estimation data set and a validation data set. The estimation data set is used to identify a model of the pH process. The validation data set is used to see if the model behaviour is close to that of the pH process.

The stored input and output of the estimation data set signals are entered to the identification algorithm. This is done with the program Matlab. As discussed in chapter 1 in the estimation algorithm three orders needs to be determined for order selection. In order to get the best model, all possible combinations of these three orders are tested. For the order n of the ARX model (this is the linear part of the Wiener model) is chosen: $n = 1$ to $n = 10$. For the number of knots of the nonlinear function m is chosen: $m = 9$ to $m = 21$ with steps of 2. The number of the knots of the inverse nonlinear function m_2 is the same as for the nonlinear function, thus $m = m_2$.

With this we can make in total 70 combinations and that means 70 different models. These combinations are tested for all the identification experiments shown in the table in figure 5.5.

A part of the measured input and output signal from experiment number 7 is given in figure 5.6. The input signal is the output signal of the final control element, thus the amount of the *NaOH* flow in ml/s.

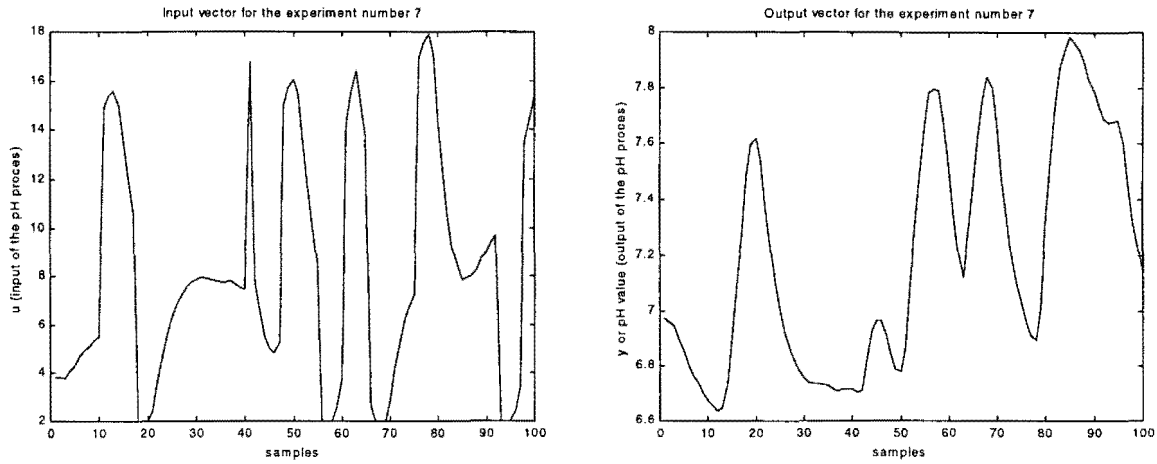


Figure 5.6 part of the measured input and output signal from experiment number 7

The identification experiment with the staircase signal is done. The number of stairs of the staircase signal is eight and the height of each step is 0.25. The width of the stairs from the beginning to the end respectively: τ up, 2τ down, 4τ up, τ down, 2τ up and 3τ down.
Whereby:

$$\tau = 25\text{sec}.$$

The stored input and output of the estimation data set signals are entered to the identification algorithm that is made with the program Matlab. (A part of the measured input and output signal is shown in figure 5.7). The results of the best model will be discussed in the next section.

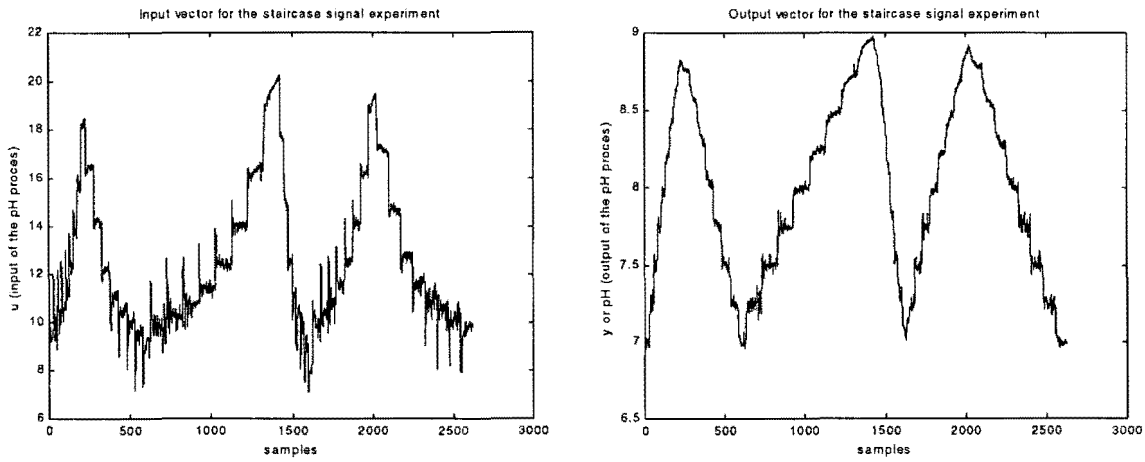


Figure 5.6 part of the measured input and output signal of staircase test signal

The identified models are also simulated inside a Matlab program. The simulations are done with Simulink, to simulate dynamic systems by using block diagrams. By doing open-loop simulations we can compare the simulated output of the pH process with the measured output. For the linear

transfer funtion is used a discrete filter and the nonlinear static function is constructed with subsystems in one block. The simulations are simulated in discrete time. The Simulink model is shown in figure 5.7.

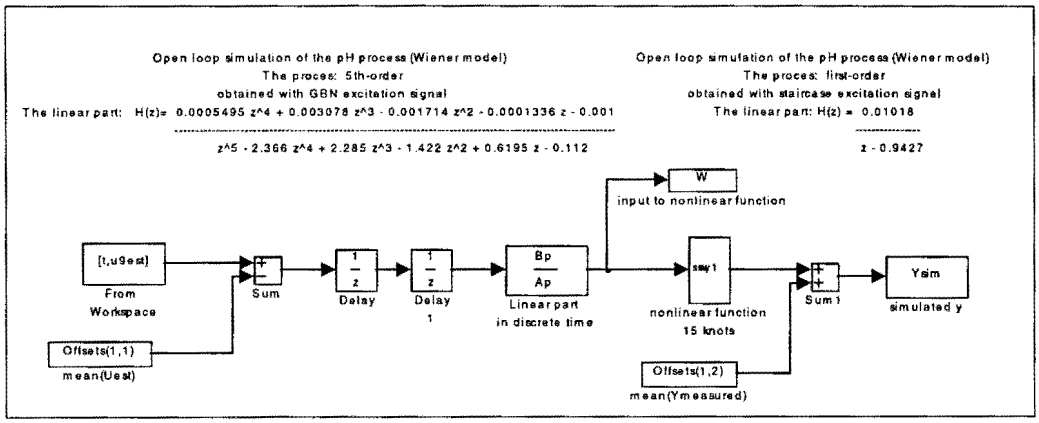


Figure 5.7 The Simulink model

In figure 5.8 is given the result of the open-loop simulation of the Wiener model whereby the GBN excitation signal is used. The input vector is the same estimated data set that is used to for the identification of the model. In figure 5.9 is given the result of the open-loop simulation of the Wiener model whereby the staircase excitation signal is used. In this case the number is not 15 but 11 and the order of the linear part of the Wiener model is 1 instead of 5.

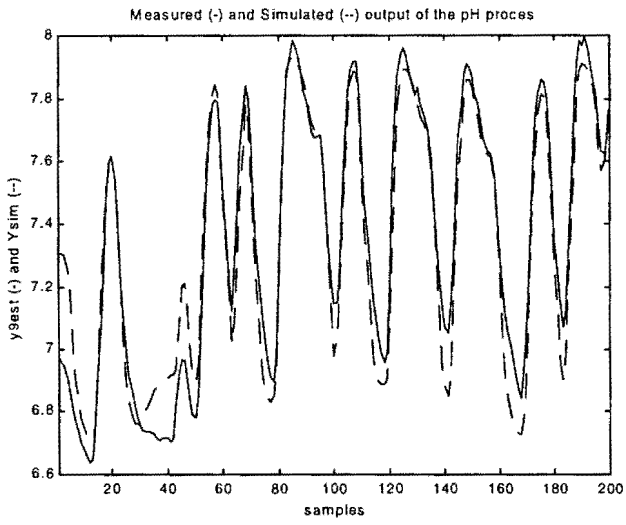


Figure 5.8 Simulation of the Wiener model

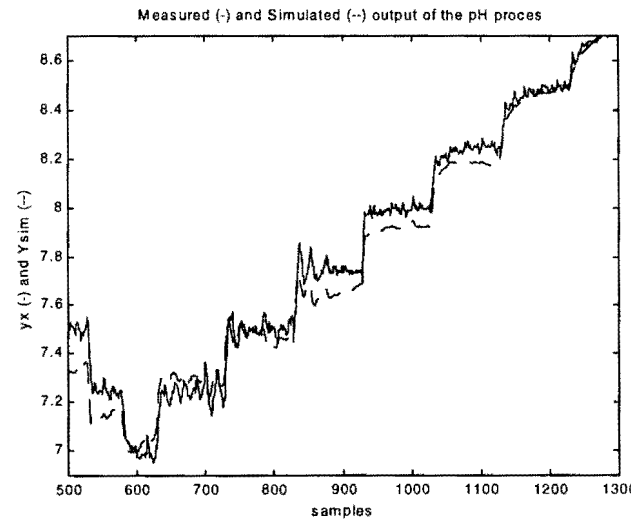


Figure 5.9 Simulation of the Wiener model

The simulated nonlinear static function of the Wiener model is shown in figure 5.10.

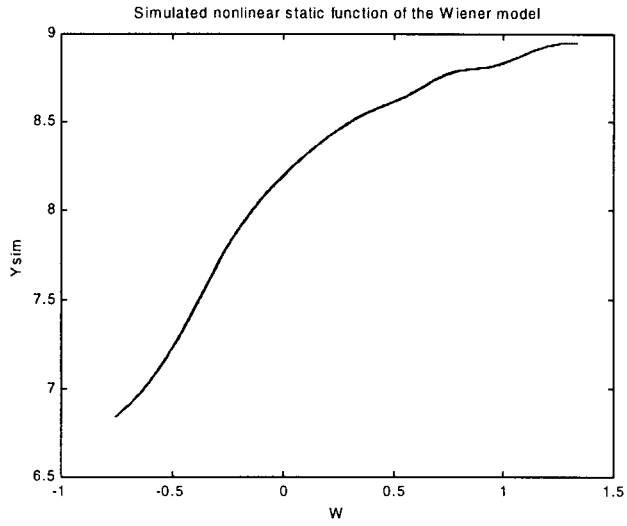


Figure 5.10 Simulated nonlinear static function of the Wiener model

For selection of the best Wiener model, order selection is done. The *final output-error criterion* (FOE) is determined for all the experiments from the table in figure 5.5. The results are given in table 5.11. Experiment number 7 gives the best model. The results of the FOE for experiment number 7 is shown in figure 5.12 and the best model is model number 32 with FOE=0.0078. The smallest value for the FOE is the model with order $n = 5$ for the linear part and $m = m_2 = 15$ for the nonlinear static function and its inverse function. The number of the samples of the validation data set is 1250.

Experiment number	1	2	3	4	5	6	7	8
knots	21	11	9	11	9	9	15	9
order	4	5	2	4	3	3	5	4
FOE	0.0132	0.0182	0.0234	0.0172	0.0150	0.0114	0.0078	0.0107

Figure 5.11 Order selection

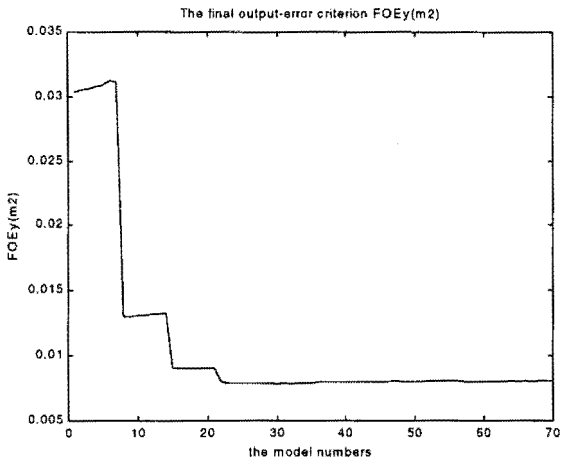


Figure 5.12 Result of the FOE

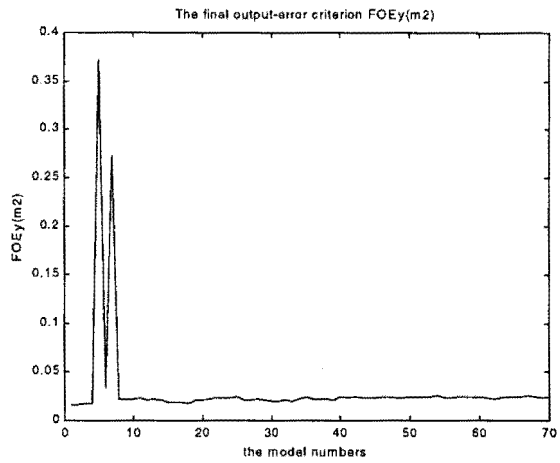


Figure 5.13 Result of the FOE

The result of the FOE for the obtained model with staircase signal is given in figure 5.13. The best model is model number 2 with FOE =0.0165. The smallest value for the FOE is the model with order $n = 1$ and $m = m_2 = 11$. The number of the samples of the estimation data set is 2626. The simulation is done with the same block diagram in Simulink as in figure 5.7. The result is given in figure 5.14.

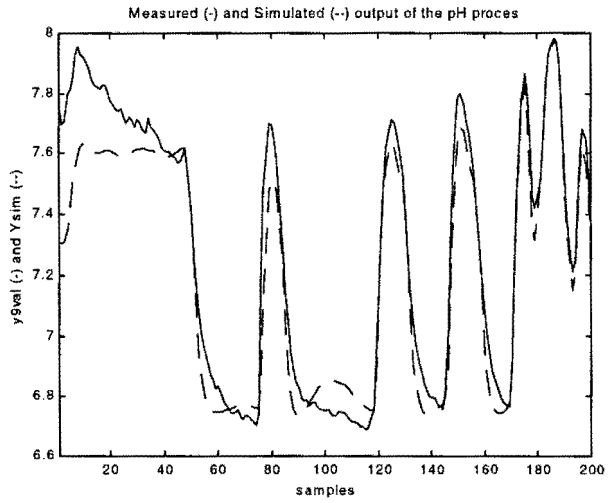


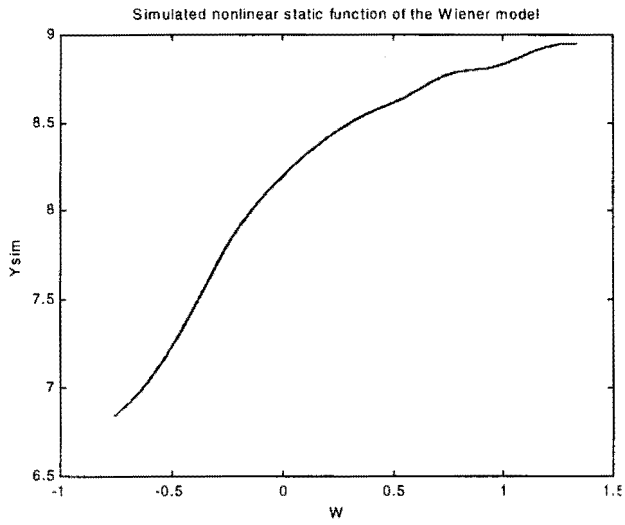
Figure 5.14 Simulation result with estimation data

In this experiment the data set is not divided into an estimation data set and a validation data set. The whole data set is used to identify a model of the pH process. Since the Wiener model has more parameters (parameters for the linear model and for the nonlinear function) than a linear model for the same process, the duration of identification test requires more time and thus more samples are needed.

Because of the limitation of the testing time of the pH process, we can only do measurements for a restricted time. After a time the *NaOH* and *HCl* solutions in the tanks get empty.
So there is no validation data set and model validation is not applied.

5.5 The comparison and choice of the model

The model, which is identified by using the GBN signal as excitation signal (say modelA), is compared with the model by using the staircase signal (say modelB). The FOE of modelA is smaller than modelB. For the purpose of control we have to look at some other things than only the smallest FOE. We have to compare the simulated nonlinear static functions of both Wiener models. See figures 5.15 (of modelB) and 5.16 (of modelA) for the functions.



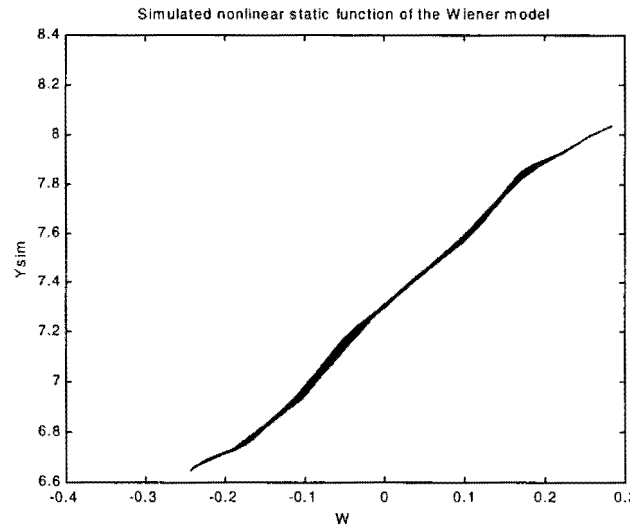


Figure 5.15 Nonlinear function of modelB

Figure 5.16 Nonlinear function of modelA

The nonlinear static function of modelA has the shape of a linear function and also the range of the output (the y-axis) of the function is small. That means that this model (modelA) can be tested only in the range of $y = pH \approx 6.6 - 8$. The nonlinear static function of modelB shows a real nonlinear function, and has a larger output range ($y = pH \approx 6.7 - 9$).

ModelA is maybe more accurate but is not sufficient for testing a Wiener model. Since these models have a nonlinear gain, an input signal must be used which also demonstrates the response of the process to a range of amplitude changes.
Therefore modelB is chosen and will be used in the simulations and measurement for the pH process.

5.6 Simulation results

The linear PID controller and the nonlinear PID controller are simulated with Simulink. A step input of magnitude 0.4 is applied to the linear PID controller structure of figure 5.18 and the response is shown in figure 5.17.

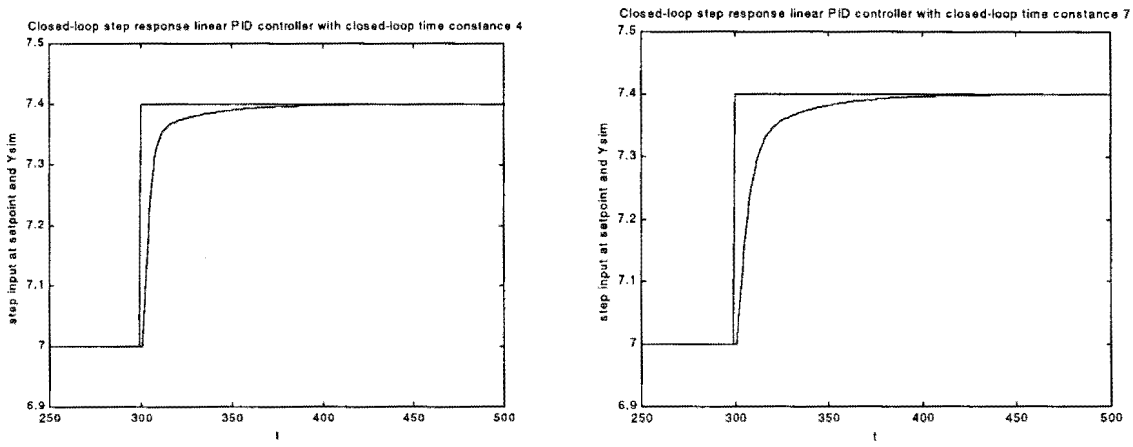


Figure 5.17 Simulation results of step responses

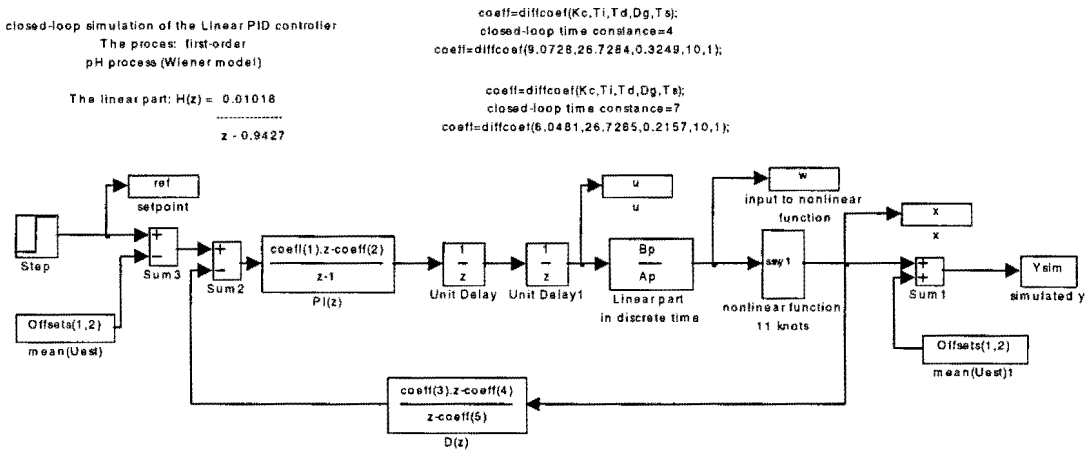


Figure 5.18 The linear PID controller structure in Simulink

The nonlinear PID controller controller structure in Simulink is given in figure 5.19. The same step input signals are used and the step responses are shown in figure 5.20.

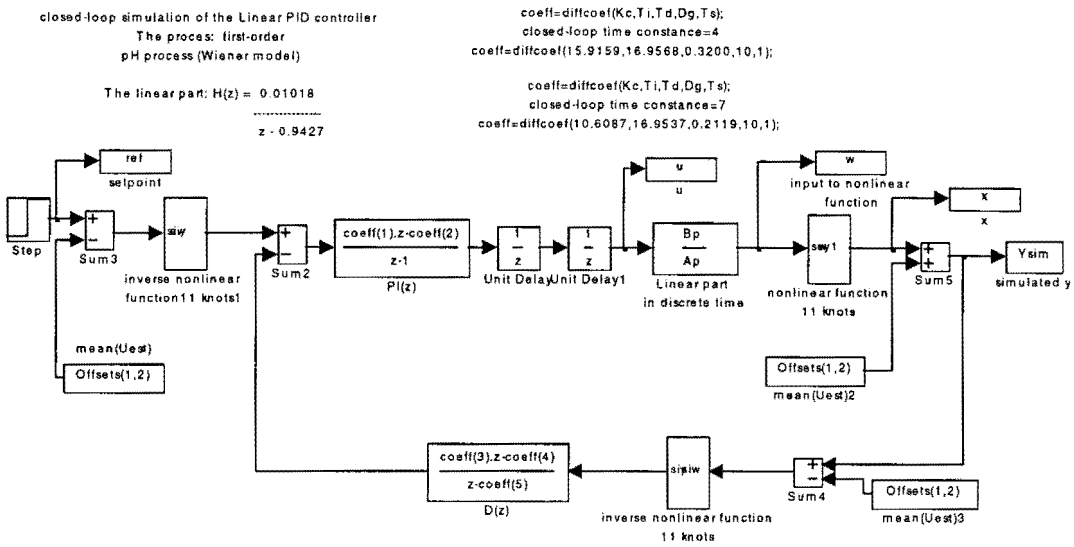


Figure 5.19 The nonlinear PID controller controller structure in Simulink

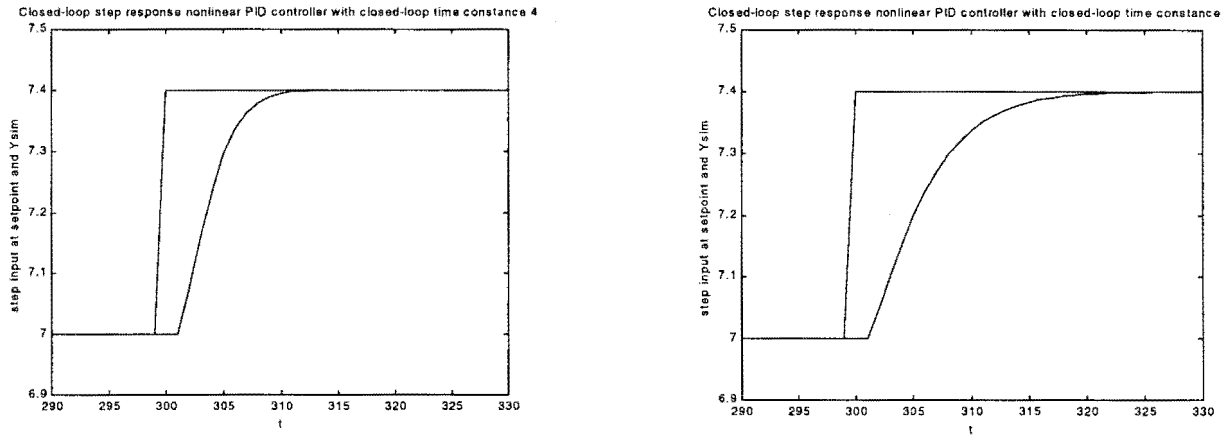


Figure 5.20 Simulated step responses

5.7 Experiments of controlling the pH process

The simulated nonlinear PID controller is implemented in the computer and experiments are done with the pH process. Step inputs are used to check the performance of the nonlinear PID controller at different nonlinear points in the titration curve. The same experiment is done for the linear PID controller. The experiments must be done with the same *NaOH* and *HCl* solutions to compare the results of the measurements. The determination of the robustness of the linear and nonlinear PID controller is done. The analysing of the robustness is experimentally determined by using different values for the closed-loop time constant. The controllers are getting slower

when the closed-loop time constant is increased. First the linear PID controller is applied to the pH process and after that the nonlinear PID controller with both tuned at a closed-loop time of 4. The second experiment is done at a closed-loop time of 7. The choice of the step input values is made by looking first at the range whereby the nonlinear static function is valid for the nonlinear PID controller.

The results are given in figure 5.21; the left picture is from the linear PID controller and the right picture of the nonlinear PID controller.

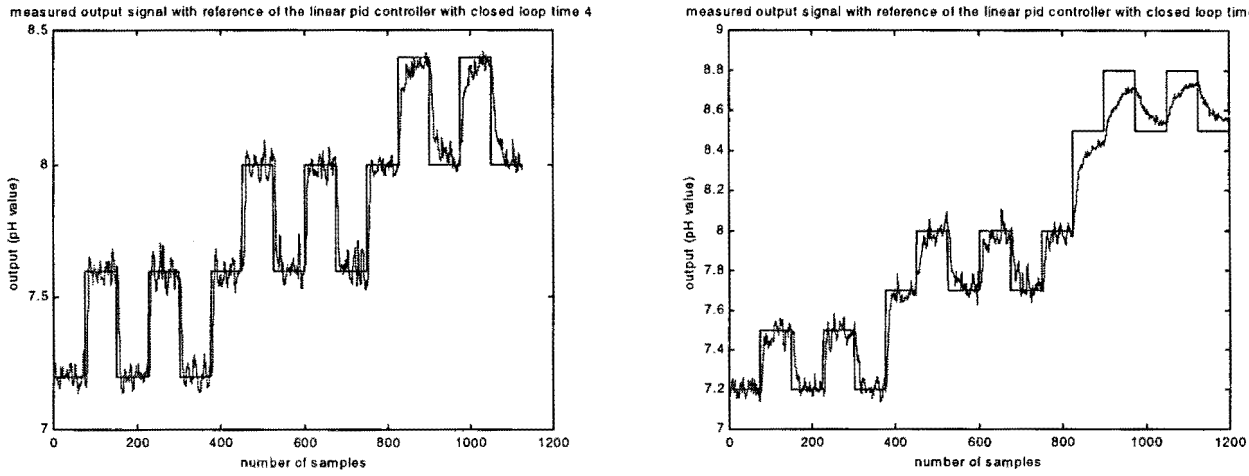


Figure 5.21 Measured step responses

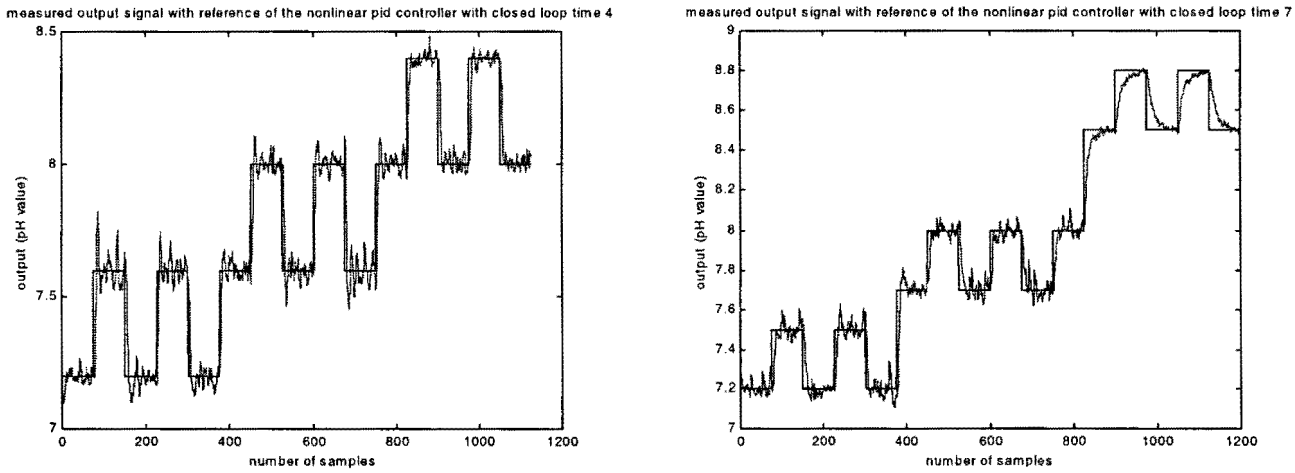


Figure 5.22 Measured step responses

5.8 The performance of the nonlinear PID controller

When we compare the linear PID controller and the nonlinear PID, we can see from the simulations with the Wiener model, that the nonlinear PID controller is much faster.
In figure 5.23 are the same simulation results given in one picture. One can determine that the step responses of the nonliner PID controller are much faster (the first two responses counted from left).

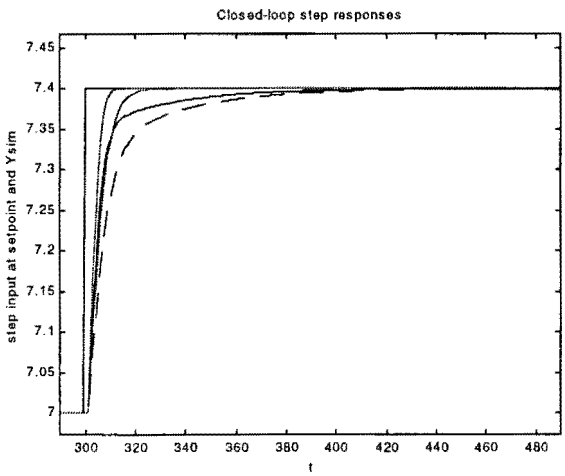


Figure 5.23 Simulation results of step responses

The robustness can be analysed practically by using different values for the closed-loop time constant. One can see in the simulations with the linear and nonlinear PID controller that the response speed of the output of the process is changing.

The linear and the nonlinear PID controller coefficients were adjusted with the adjustments used in the simulations with Simulink. The measurement results show that the responses of the step inputs of the nonlinear PID controller are faster, even when the controller speed is decreased.
The nonlinear PID controller can give only good results when it is used in the range of the nonlinear static function. Outside its range it becomes instable.

6 Conclusions and recommendations

Conclusions

The goal of the project was to study nonlinear process control using Wiener or Hammerstein models. This project included theoretical and practical aspects: black box identification for one of these models, software and designing a nonlinear PID controller by using a linear PID controller and simulating of the nonlinear PID controller. The black box identification was applied on a chemical process in a laboratory, a pH process. The identification procedure showed that is was possible to estimate a model of the pH process and this was succesfully used to determine a model for simulations and to design the nonlinear PID controller.

Using the inverse nonlinear static function in the control loop, a Wiener model can be controlled by using a linear PID controller. The PID controller is tuned by using the Internal Model Control approach. This demonstrates simple block-oriented models can improve control performance for nonlinear processes.
The nonlinear PID controller of the pH process has a better performance than a linear PID controller.

Recommendations

Although the results were good enough to simulate and design a nonlinear PID controller, some improvements could have resulted in a better Hammerstein or Wiener model, which will improve the control of nonlinear processes.

- Improvement of the design of the excitation signal for identification is possible. A better distribution and more amplitude levels can lead to model of the pH process. This will result in a better inverse nonlinear function.
- Study the use of Hammerstein or Wiener models for Model Predictive Control. These models can be incorporated into model predictive control schemes in a way that effectively removes the nonlinearity from the control problem.
- Adaptive control can be applied for controlling Wiener models.
- Study the model structure that combines Hammerstein and Wiener models.

Bibliography

[LAM 99] Lammerts, J.
AUTOMATIC PID CONTROLLER TUNING BASED ON CLOSED-LOOP IDENTIFICATION.
Eindhoven: Eindhoven University of Technology, Measurement & Control group M.Sc. Thesis, 1999.

[ZHU 98a] Zhu,Y.C.
PARAMETRIC WIENER MODEL IDENTIFICATION FOR CONTROL.
Eindhoven: Eindhoven University of Technology, Faculty of Electrical Engineering, 1998.
EUT Report

[ZHU 98b] Zhu,Y.C.
IDENTIFICATION OF HAMMERSTEIN MODELS FOR CONTROL USING ASYM.
Eindhoven: Eindhoven University of Technology, Faculty of Electrical Engineering, 1998.
EUT Report.

[ZHU 98c] Zhu,Y.C.
MULTIVARIABLE PROCESS IDENTIFICATION FOR MPC: THE ASYMPTOTIC METHOD AND ITS APPLICATIONS.
Journal of Process Control, Vol. 8 (1998), No. 2, p. 101-115.

[LAM 98] Lammerts, I.M.M.
DOCUMENTATION FOR THE LABORATORY COURSE. Dynamic Systems and Control Technology.
Eindhoven: Eindhoven University of Technology, Systems and Control group, Departments of Chemical Technology and Applied Physics, Version 14 May, 1998.

[NOR 98] Norquay, S.J., A. Palazoglu and J.A. Romagnoli.
MODEL PREDICTIVE CONTROL BASED ON WIENER MODELS.
Chemical Engineering Science, Vol. 53 (1998), No. 1, p. 75-84.

[ALD 96] Alduwaish, H., M.N. Karim and Chandrasekar.
USE OF MULTILAYER FEEDFORWARD NEURAL NETWORKS IN IDENTIFICATION AND CONTROL OF WIENER MODEL
IEE Proc.-Control Theory Appl., Vol. 143 (1995), No. 3, p. 255-258.

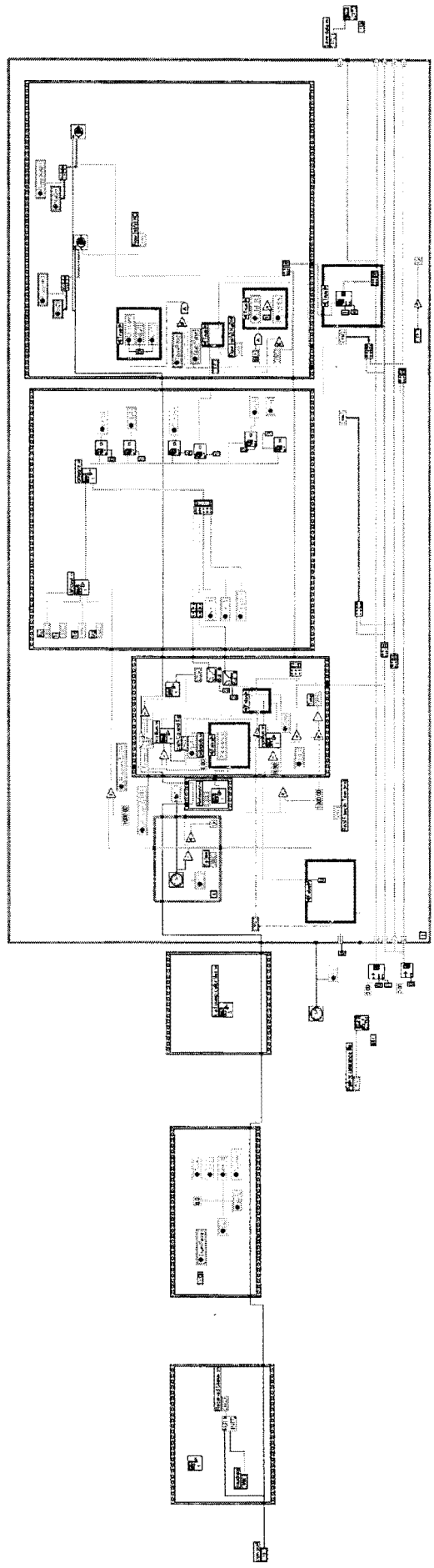
[AST 95] Åström, K. and T. Hägglund.
PID CONTROLLERS: THEORY, DESIGN, AND TUNING.
Sweden: Department of Automatic control, Lund institute of Technology, 1995.

[MCM 94] McMillan, G.K.
PH MEASUREMENT AND CONTROL.
Second edition. An independent Learning Module from the Instrument Society of America, 1994

[OGU 94] Ogunnaike, B.A. and Harmon Ray, W.
PROCESS DYNAMICS, MODELING AND CONTROL.
New York: Oxford University Press, 1994

[PAJ 84] Pajunen, G.A.
APPLICATION OF A MODEL REFERENCE ADAPTIVE TECHNIQUE TO THE IDENTIFICATION AND CONTROL OF WIENER TYOE NONLINEAR PROCESSES.
Helsinki: Doctoral Dissertation, Department of Electrical Engineering, Helsinki University of Technology, 1984.

Appendix 1. LabView scheme of the nonlinear PID controller



Appendix 2. Extrapolation program

```
----------------------------------------------------------main m-file: extrapce.m----------------------------------------
function [We,Ye,ke,ce]=extrapce(W,Ypro,nK,Wmin,Wmax)

%       [We,Ye,ke,ce]=extrapce(W,Ypro,nK,Wmin,Wmax)
% This function extrapolates the nonlinear function (cubic spline function)
% by clicking points at the nonlinear function. The data points must be
% clicked in the axis.The extrapolated function contains the same number of knots and coefficients.
%
% Input arguments:
%       W:      Row vector of the input values of the nonlinear function
%       Ypro:   Row vector of the output values of the nonlinear function
%       nK:     The number of knots for the nonlinear function
%       Wmin:   begin value for plotting of the nonlinear function
%       Wmax:   end value for plotting of the nonlinear function
%
% Output arguments:
%       We:     Row vector of the input values of the nonlinear function
%       Ye:     Row vector of the new Y values
%       ke:     Row vector of knot sequence for the extrapolated nonlinear function
%       ce:     Row vector of cubic spline coefficients for the extrapolated nonlinear function


clear We Ye ke ce
clg;

hold;
N=size(Ypro,1);

[cn,kn,Ynh] = cspline2(W,Ypro,nK);  % This function fits the function Yn = f(W)

Wj=Wmin:0.004:Wmax;
Wj=Wj(:);
Yb=csplfun2(cn,kn,Wj);              % This function evaluates cubic spline function
plot(Wj,Yb);                        % Plotting of the nonlinear function
hold on
xlabel('Y --->'); ylabel('W --->')
title('Fitting of the cubic spline function to the data points');
hold;

klik                                % clicking of the points (see klik.m)

[ke,ce,We,Ye] = extrapne(kn,cn,X,Y,nK,W,Ypro);   % This function extrapolates the clicked points
Wj=Wmin-1:0.004:Wmax+1;
Wj=Wj(:);
Yc=csplfun2(ce,ke,Wj);              % This function evaluates the extrapolated cubic spline function
grid on
hold on
plot(Wj,Yc,'b');            % Plotting of the extrapolated nonlinear function
xlabel('Y --->'); ylabel('W --->')
title('Fitting of the cubic spline function to the data points');
```

```
----------------------------------------------------------------m-file: extrapne.m----------------------------------------------------
function [kn,cn,Wnh,Yn] = extrapne(k,c,Xc,Yc,nK,Yf,Wf)

%       [kn,cn,Wnh,Yn] = extrapne(k,c,Xc,Yc,nK,Yf,Wf)
%       This function extrapolates the clicked points
%
%       Input arguments:
%       k:              Row vector of knot sequence (extra knots included)
%       c:              Row vector of cubic spline coefficients for the nonlinear function
%       Xc:             Row vector of the clicked points
%       Yc:             Row vector of the clicked points
%       nK:             The number of knots for the nonlinear function
%       Wf:             Row vector of the input values of the nonlinear function
%       Yf:             Row vector of the output values
%
%       Output arguments:
%       kn:             Row vector of knot sequence for the extrapolated points
%       cn:             Row vector of cubic spline coefficients for the extrapolated nonlinear function
%       Wnh:            Row vector of the input values of the nonlinear function
%       Yn:             Row vector of the new (extrapolated) Y values

[Xcs,Ycs,I]=sorteer(Xc,Yc);             % Xc and Yc will be sorted --> Xcs and Ycs
Wf=Wf';
Yf=Yf';
Xcs=Xcs';
Ycs=Ycs';

for b=1:length(Xc)                      % The sorted clicked points Xcs and Ycs
 Yf=[Yf,Xcs(b)];                        % will be added to the other points (Wn and Yn).
 Wf=[Wf,Ycs(b)];
end
hold;
klik                                    % Clicking of points in the left side
                                        % of the nonlinear function.

Xcl=X;                                  % X and Y are the co-ordinates of the clicked
Ycl=Y;                                  % points that is saved in the in the function click
[Xcsl,Ycsl,I]=sorteer(Xcl,Ycl);         % Xcl and Ycl will be sorted --> Xcsl and Ycsl
Xcsl=Xcsl';
Ycsl=Ycsl';

for b=1:length(Xcsl)
 Yf=[Xcsl(length(Xcsl)-b+1),Yf];
 Wf=[Ycsl(length(Xcsl)-b+1),Wf];
end
Wn=Wf';
Yn=Yf';
[cn,kn,Wnh] = cspline2(Yn,Wn,nK);       % This function fits the function Yn = f(W)
                                        % by a cubic spline using LS.
```

```
----------m-file: cspline2.m----------
function [C,K,Yh] = cspline2(X,Y,nK)
%
%    [C,K,Yh] = cspline2(X,Y,nK)
%
% This function fits the function Y = f(X) by a cubic spline using LS. The knots
% on X will be unitformly distributed in [min(X) max(X)]
%
% Input arguments:
%        X:      Vector of independent variable (abscissas, input)
%        Y:      Vector of function velues (cordinates, output)
%      nK:       The number of knots on X, must be odd, Xmin is 1st, Xmax last
%
% Output arguments:
%        C:      Row vector of linear spline coefficients
%        K:      Row vector of knot sequence (values betwen Xmin and Xmax)
%        Yh:     Estimate of Y using the linear spline
%

N = length(X);

% Form the knot sequence K and regression matrix C_FI.
% This is different from the book, but it extrapolates better
K = zeros(1,nK);
C_FI = zeros(N,nK);
minX = min(X);
maxX = max(X);
dltX = (maxX-minX)/(nK-1);
K(1) = minX;
K(nK) = maxX;
C_FI(:,1:2) =[X ones(N,1)];
for j = 2:nK-1
  K(j) = minX + (j-1)*dltX;
  C_FI(:,2+(j-1)) = (abs(X-K(j))).^3;
end

% Estimate cubic spline coefficients C
th = C_FI\Y;
C = th';

% Calculate Yh
Yh = C_FI*C';

----------m-file: csplfun2.m----------
function Yh = csplfun2(C,K,X)
%
%    Yh = csplfun2(C,K,X)
%
% This function evaluates cubic spline function.
% ?????? X values are limited in [K(1) (K(nK)] ??????
%
% Input arguments:
%        C:      Row vector of cubic spline coefficients
%        K:      Row vector of knot sequence (values betwen Xmin and Xmax)
```

```
%       X:      Vector of independent variable (abscissas, input)
%
%  Output arguments:
%       Yh:     Estimate of Y using the cubic spline
%

X = X(:);
N = length(X);
nK = length(K);

% Limite the value in X by K(1) and K(nK)
%IndMin = find(X < K(1));
%X(IndMin) = K(1)*ones(length(IndMin),1);
%IndMax = find(X > K(nK));
%X(IndMax) = K(nK)*ones(length(IndMax),1);

C_FI = zeros(N,nK);
C_FI(:,1:2) =[X ones(N,1)];
for j = 2:nK-1
  C_FI(:,2+(j-1)) = (abs(X-K(j))).^3;
end

%  Calculate Yh
Yh = C_FI*C';
```

----------m-file: klik.m----------

```
% This part of the program is the clicking of the points
hold;

for n=1:1000
   [x(n),y(n),button]=ginput(1);
   if button==1; plot(x,y,'+r'), end
   if button==2; break; end
end
[x(1:n-1); y(1:n-1)]
x=x(1:n-1);
y=y(1:n-1);
Y=y';
X=x';
%save X.tmp X -ascii
%save Y.tmp Y -ascii
```

----------m-file: sorteer.m----------

```
function [Xs,Ys,I]=sorteer(X,Y)

% [Xs,Ys,I]=sorteer(X,Y)
% This function is for sorting two vectors
% X is the input vector which must be sorted
% Xs is the sorted input vector
% Y is the output vector which must be sorted
% Ys is the sorted output vector
% I is the index of the sorted vector
[Xs,I]=sort(X);
for i=1:length(X)
Ys(i)=Y(I(i));
end
```