PREDICTIVE CONTROL ALGORITHMS FOR
LINEAR AND NONLINEAR PROCESSES

(PREDICTIVE PID CONTROL, DECOUPLING, VOLterra
MODEL BASED NONLINEAR PREDICTIVE CONTROL)

Fakhredin Arousi
Ph.D. Dissertation

Scientific supervisors:

Fakhredin Arousi

Budapest, 2009
“To the soul of my Father
    To my Mother,
        To my Wife and my Sons
            To my Brothers and Sisters
                I dedicate this”
Contents

1. Introduction...........................................................................................................1

2. Predictive PID control algorithms.................................................................6
   2.1 Predictive PI(D) control structure.........................................................6
   2.2 Predictive PI control algorithm.............................................................7
       2.2.1 One-step-ahead predictive PI control.......................................8
       2.2.2 Long-range predictive PI control.............................................9
       2.2.3 Predictive PI control of a first-order process with dead time...............11
   2.3 Predictive PID control algorithm...........................................................15
       2.3.1 One-step-ahead predictive PID control..................................16
       2.3.2 Long-range predictive PID control.........................................17
       2.3.3 Predictive PID control of a second-order process with dead time........19
   2.4 Equivalence between the predictive PI(D) algorithm and the GPC algorithm.........................20
       2.4.1 Equivalence in case of extended horizon control.................20
       2.4.2 Equivalence in case of long-range horizon control.............22
       2.4.3 Simulation examples....................................................................24
   2.5 Tuning of predictive PI(D) algorithms..................................................28
   2.6 Robustifying effects applied for predictive PI(D) control Algorithms..................................................37
       2.6.1 Enhancing robustness of predictive PI(D) control using a noise model with polynomial T..................37
       2.6.2 Enhancing robustness with Smith predictor filter......................39

3. Decoupling in MIMO predictive control.......................................................45
   3.1 The control algorithm.............................................................................45
       3.1.1 Control without and with ideal decoupling.................................48
   3.2 Decoupling by decelerating the reference signal change......................52
   3.3 Weighting factor adjusting at reference signal change.......................55
   3.4 Control error dependent weighting factor adjusting............................57
   3.5 Application to distillation column model.............................................59
4. Nonlinear predictive control based on the Volterra model…………………..65

4.1 Suboptimal nonlinear Volterra model based predictive control algorithms applied to the model of a pilot plant…………………..65
  4.1.1 The parametric Volterra model and its predictive form……….66
  4.1.2 The cost function.................................................................68
  4.1.3 Minimization strategies for the cost function......................69
  4.1.4 The model of the ball and beam pilot plant......................71
  4.1.5 Identification results of the model.................................72
  4.1.7 Control simulation examples of the ball and beam system…..75

4.2 An iterative GPC-like nonlinear predictive control algorithm for the parametric Volterra model..............................77
  4.2.1 A new iterative control algorithm.................................77
  4.2.2 A case study: level control................................................80

5. Summary of the new scientific results...........................................85

6. Conclusion......................................................................................88

7. Bibliography...................................................................................90
Abstract

PID control algorithms are the most accepted in the process industries. Besides, predictive algorithms are also widely applied, where predicted error values are used to calculate the actual control signal. Predictive control algorithms provide good performance especially in case of big dead time and if the reference trajectory is known. As the operators of industrial control systems are familiar mostly with PID control and controller tuning, it would be beneficial to enhance the PID control algorithms with predictive properties while keeping the usual PID controller tuning rules. In this research predictive PID control algorithms are derived for first- and second-order systems with dead time. The structure of predictive PI(D) algorithms is shown. Tuning rules equivalent to GPC are presented. Tuning rules of thumb (Kuhn, Åström-Hägglund, etc.) can also be used with appropriate modification. As in predictive PI(D) control the effect of the parallel paths is added, in this case the continuous gain has to be divided by the number of the parallel channels. In case of plant/model mismatch the robustifying effect of the noise model filter \( T(q^{-1}) \) polynomial and of the robustifying filter in the Smith predictor-like scheme added to the predictive control structure is demonstrated. Simulation examples present the behavior of predictive PI(D) algorithms.

In control of MIMO process decoupling is an important question. Predictive control of MIMO system ensures decoupling if the input increments are not considered in the cost function. As inclusion of punishment of big control increments is important from practical point of view, improving of decoupling in this case has to be analysed. Several decoupling techniques are presented for TITO processes. In the first method the reference signal change is decelerated in order to make the control slower and to reduce the coupling effect. A new filter design is recommended for calculating the modified reference signal which suppresses the effect of the disturbance in the other control variable whose set-point was kept constant. In the second method different control error weighting factors are used for the two controlled outputs. The adoption of the weighting factors has to be synchronized to the reference signal change. A new automatic adoption procedure is introduced which makes the synchronisation superfluous by setting the weighting factor dependent on the control error.

Predictive control algorithms have been worked out mainly for linear processes. As in practice the plants generally have nonlinear characteristics, considering the nonlinearities in the control algorithms promises better control performance in the whole operating range. Parametric Volterra models can approximate well the behaviour of a nonlinear plant. Suboptimal nonlinear predictive control algorithms...
have been derived for the parametric Volterra model in Haber (1995), [27]. The algorithms are applied here for the model of a ball and beam system. The model is identified in the form of parametric Volterra model. The identification experiments show that the Volterra model gives a good approximation of the nonlinear process. The nonlinear predictive control algorithms are applied for the nonlinear process model. The simulations analyze the effect of the tuning parameters.

A new iterative nonlinear predictive control algorithm is presented here based on the quadratic parametric Volterra model. The algorithm uses a GPC-like structure. The control performance is demonstrated by a case study of level control of a two-tank system. The behavior of the new algorithm is compared with other suboptimal nonlinear algorithms.
Preface

PID control algorithms are the most accepted in the process industries. Besides, predictive algorithms are also widely applied, where predicted error values are used to calculate the actual control signal. Predictive control algorithms provide good performance especially in case of big dead time and if the reference trajectory is known.

The aim of this work was to contribute to the area of predictive control developing algorithms
- for predictive PI(D) control,
- for improving decoupling properties of MIMO predictive control,
- for nonlinear predictive control using Volterra models.

For applying predictive PI(D) controllers I used first- or second-order approximation of the plant with dead time. For these models the tuning of the parameters could be based on modified simple tuning rules of thumb. The aim was also to enhance the algorithms with some extensions to improve tracking and robustness properties.

For dynamic decoupling I applied adaptive tuning of the weighting factors in the cost function. I used these decoupling techniques for TITO processes.

I have suggested an iterative optimization method to control nonlinear systems identified as Volterra models. These algorithms contribute to better practical applicability of predictive control.

As the operators of industrial control systems are familiar with PID control and controller tuning, it would be beneficial to enhance the PID control algorithms with predictive properties while keeping the usual PID controller tuning rules. In this research predictive PID control algorithms have been derived for first- and second-order systems with dead time. The structure of predictive PI(D) algorithms is shown. Tuning rules are given. Enhancing robustness properties of the algorithms is also considered.

Some decoupling techniques are presented for TITO processes. By the first method the reference signal change is decelerated in order to make the control slower and to reduce the coupling effect. A new filter design is recommended for calculating the modified reference signal which suppresses the effect of the disturbance in the other control variable whose set-point was kept constant. In the second method different control error weighting factors are used for the two controlled outputs. The adoption
of the weighting factors has to be synchronized to the reference signal change. A new automatic adoption procedure is introduced which makes the synchronisation superfluous by setting the weighting factor dependent on the control error.

Predictive control algorithms have been worked out mainly for linear processes. As in practice the plants generally have nonlinear characteristics, considering the nonlinearities in the control algorithms promises better control performance in the whole operating range. Parametric Volterra models can approximate well the behaviour of a nonlinear plant. Suboptimal nonlinear predictive control algorithms have been derived for the parametric Volterra model in Haber (1995), [27]. The algorithms are applied here for the model of a ball and beam system. The model is identified in the form of parametric Volterra model. The identification experiments show that the Volterra model gives a good approximation of the nonlinear process. The nonlinear predictive control algorithms are applied for the nonlinear process model. The simulations analyze the effect of the tuning parameters.

A new iterative nonlinear predictive control algorithm has been worked out based on the quadratic parametric Volterra model. The algorithm uses a GPC-like structure. The control performance is demonstrated by a case study of level control of a two-tank system. The behavior of the new algorithm is compared with other suboptimal nonlinear algorithms.

I would like to express my appreciation and gratitude to my supervisors dr. Ruth Bars and Prof. Dr. Robert Haber for the outstanding supervision and well founded professional support during this work, which made this work possible at all. Also many thanks to dr. Ulrich Schmitz for the fruitful discussions and advices in programming and for Mr. Barta Andras for the discussions and his help in modeling the ball and beam system.

Finally special thanks to my family for their support, patience and understanding of family time lost during this work.
Notations

\( y(t) \)  continuous-time output signal (controlled variable)
\( y(k) \)  discrete-time output signal (controlled variable)
\( u(k) \)  discrete-time input signal (control variable)
\( y_r(k) \)  discrete-time reference signal
\( e(k) \)  control error
\( y_r(k + d + 1 + n_e | k) \)  reference signal predicted \( n_e \) +1 steps beyond the dead time
\( \hat{y}(k + d + 1 + n_e | k) \)  controlled output signal predicted \( n_e \) +1 steps ahead
\( \hat{e}(k + d + 1 + n_e | k) \)  predicted control error
\( k \)  actual discrete time
\( d \)  physical dead time
\( n_{e1} \)  beginning of the control error horizon beyond the dead time
\( n_{e2} \)  end of the control error horizon beyond the dead time
\( n_u \)  control horizon, the supposed consecutive changes in the control increment
\( \Delta u \)  actual and future manipulated variables
\( \lambda_u \)  factor punishing the control increments
\( H \)  matrix containing the points of the step response
\( m \)  number of the parallel paths \( m = n_{e2} - n_{e1} + 1 \)
SISO  single-input single-output system
MIMO  multi-input multi-output system
\( k_{ip}, k_{ia} \)  and \( k_{id} \)  are the coefficients of the proportional, the integrating and the derivative components of the controller
\( k_{pi} \)  vector containing the coefficients of the predictive PI controller
\( k_{PID} \)  vector containing the coefficients of the predictive PID controller
\[ G(q^{-1}) \] pulse-transfer function
\[ A(q^{-1}) \] denominator polynomial of the pulse-transfer function
\[ a_i \] coefficient of polynomial \( A(q^{-1}) \)
\[ B(q^{-1}) \] numerator polynomial of the pulse-transfer function
\[ B(q_1^{-1}, q_2^{-1}) \] two-dimensional polynomial of quadratic input terms
\[ b_i \] coefficient of the polynomial \( B(q^{-1}) \)
\[ C(q^{-1}) \] noise filter
\[ h_{PI} \] vector containing the points of the step response for predictive PI controller
\[ y_{rPI,\text{sum}} \] vector containing the reference signal
\[ \hat{y}_{\text{freePI,\text{sum}}} \] vector containing the free response
\[ \mathbf{H}_{n_{PI}} \] matrix of the forced response for predictive PI controller
\[ \mathbf{F}_{y_{\text{PI,\text{sum}}}}^{n_{PI}} \] matrix of the free response related to the past output values for predictive PI controller
\[ \mathbf{H}_{n_{PI}}^{P} \] matrix of the free response related to the past input values for predictive PI controller
\[ \mathbf{F}_{y_{\text{GPC}}}^{P} \] matrix of the free response related to the past output values for GPC controller
\[ \mathbf{H}_{\text{GPC}}^{P} \] matrix of the free response related to the past input values for GPC controller
\[ y_F(q^{-1}) \] filtered output signal
\[ \Delta u_F(q^{-1}) \] filtered input signal
1 Introduction

In the last 30 years predictive control has gained wide acceptance in the process industries besides PID control (Qin, Badgwell, 2003, [42]). Predictive control determines the control increment sequence minimizing a cost function considering the deviation between the reference signal and the output signal predicted in a future horizon. The future output signal is predicted based on the system model.

Giving a model of a system which describes its behavior from one or another point of view is very useful way to summarize the knowledge about the process. Process model can be obtained from prior knowledge understanding the process behavior, the physical principles governing its operation and giving a proper mathematical formulation for them. Generally nonlinear differential equation, or a set of first order nonlinear differential equation describe the system. Another way to get process models is by experimentation on the process. System identification is the experimental approach to process modeling. To obtain an adequate model describing the system it is beneficial to combine the methods of physical modeling and experimentation.

For predictive control predictive process models are required, giving future output estimation based on information on the input and output signals available up to the actual time point. If the system model is non-predictive, predictive transformation has to be executed.

In predictive control a quadratic cost function considering the deviation between the future reference signal and the predicted process outputs in a given future horizon is minimized under constraints. Without constraints in linear case GPC (Generalized Predictive Control) gives an analytic solution for the control algorithm (Clarke, 1987, [21]). From the calculated control sequence only the first one is applied and in the next sampling point the procedure is repeated (receding horizon strategy).

The predictive control algorithm is obtained by minimizing the cost function

\[ J = \sum_{n_1}^{n_d} \hat{e}^2(k+d+1+n_e | k) + \sum_{i=1}^{n_c} \lambda_i \Delta u^2(k+i-1) \]

where

\[ \hat{e}^2(k+d+1+n_e | k) = y_r(k+d+1+n_e | k) - \hat{y}(k+d+1+n_e | k) \]

1
\( y_r(k + d + 1 + n_e | k) \) is the reference signal predicted \( n_e \) steps over the dead time

\( \hat{y}(k + d + 1 + n_e | k) \) is the controlled output signal predicted \( n_e \) steps ahead

\( k \) is the actual discrete time

\( \hat{e} \) is the predicted future control error

\( d \) is the physical dead time

\( n_{e1} \) is the beginning of the control error horizon beyond the dead time

\( n_{e2} \) is the end of the control error horizon beyond the dead time

\( n_e \) is the control horizon, the supposed consecutive changes in the control increment

\( \Delta u \) denotes the actual and future manipulated variables

\( \lambda_e \) is the factor punishing the control increments

The output is predicted based on the system model. It is composed of the free response (the effect of the past inputs and outputs on the future output value) and the forced response (the effect of the actual and future control increments on the future output value).

The tuning parameters of predictive control algorithms are the prediction horizon, the initial point of the prediction horizon, the control horizon (number of the allowed consecutive changes of the control signal during the calculation) and the weighting parameters in the cost function.

Minimization of the cost function can be executed considering constraints. Without constraints minimization of the cost function gives the following analytical solution (GPC):

\[
\Delta u = \left[ H^T H + \lambda_e I \right]^{-1} H^T (y - y_{free})
\]

\( H \) is the matrix containing the points of the step response (Clarke, 1987, [21]).

Nowadays several industrial program packages support and ensure an appropriate environment for industrial applications of predictive control (e.g. Honeywell, 2007, [32]). These program packages provide possibilities for measurement and monitoring different process variables, execute unit step response measurements, identification, simulation, controller design, real-time control.

For controller design linear process models are preferred, as linear control theory provides well elaborated controller design methods. General controller design methods are still not available for nonlinear plants.
Big efforts are done for developing controller design methods handling nonlinearities. Nonlinear systems with continuous nonlinearities generally are linear in small, for small changes around the working points. Designing different controllers for different working points and switch among them when variation is observed in the working point leads to adaptive control realization. Controllers designed for an adequate nonlinear model of the system are more complex, but could provide better tracking and regulation properties than linear controllers in the whole operation range.

In case of big dead time in the process the performance of the control system will be slow, the PID controller can not accelerate the control system significantly. There are some discrete control algorithms as Smith predictor or dead-beat control, which provide faster performance than PID control for dead time systems, but these algorithms did not get really a wide industrial acceptance, especially because of their sensitivity to plant-model mismatch. Nevertheless, there are some methods to make the Smith predictor less sensitive to parameter uncertainties (Normey-Rico and Camacho, 2007 [41]).

In case of systems containing big dead time significantly faster control performance can be achieved with predictive control than with PID control. Also in case of known future reference trajectory predictive control provides better tracking properties.

Nevertheless it is difficult to introduce advanced control algorithms in practice, as PID algorithms also work well, and generally control problems can be solved with them.

Operators of industrial processes have a good expertise in PID controller tuning. Different tuning rules of thumb can be used, which are based mostly on the approximation of the system with a first-order model with dead time (Kuhn, 1995, Chien et. al., 1952, Åström, 2006, [17, 20, 36]). PID control algorithms can be equipped with predictive property if the error signal applied to calculate the control signal is not the actual error, but its predicted value in a future point or in a future horizon (Johnson, Moradi, 2001, 2005, [34, 35]). In this control system the operator will see PID control with embedded predictive properties.

Industrial processes generally are complex, containing interacting MIMO controller loops. It is required to select appropriate pairing of the input and output variables to ensure more effective control. There are different decoupling techniques.

It is known that multivariable predictive controllers can decouple the Multi-Input, Multi-Output (MIMO) processes well. The decoupling is perfect only if the control
increments are not weighted. In this case non-smooth control may occur. Weighting of the control increments in the cost function is important to keep the control variables within their limit values. Therefore practical methods for improving decoupling are required.

Most of the plants contain nonlinearities. The control algorithms generally consider linear, or linearised models around the working points. Predictive control algorithms based on nonlinear model of the plant would ensure better performance in the whole operating range, than algorithms using linear models. In the recent years there is an active research to develop nonlinear predictive control algorithms (Findeisen et. al [23]). One direction is to use Wiener, Hammerstein or Volterra models to approximate the nonlinear characteristics and to develop predictive controllers based on these models (Keviczky-Haber, 1999, [25,26], Lengyel,1998, [37], Haber et al., 1999, [ 29, 30, 31]).

**The objective of this thesis**

The aim of this work was to contribute to the area of predictive control developing algorithms

- for predictive PI(D) control,
- for improving decoupling properties of MIMO predictive control,
- for nonlinear predictive control using Volterra models.

For applying predictive PI(D) controllers I used first- or second-order approximation of the plant with dead time. For these models the tuning of the parameters could be based on modified simple tuning rules of thumb. The aim was also to enhance the algorithms with extensions to improve tracking and robustness properties.

For dynamic decoupling I applied adaptive tuning of the weighting factors in the cost function. I used these decoupling techniques for TITO processes.

I have suggested an iterative optimization method to control nonlinear systems identified as Volterra models.

These algorithms contribute to better practical applicability of predictive control.
Research methodology

For the three problems theoretical considerations have lead to new control algorithms.

Some parts of the research were based also on previous diploma projects (Arousi, 2000, [16], Lengyel, 1998, [37]). Discussions with former PhD student Ulrich Schmitz were especially fertilizing. The common research project with the Sevilla University, Engineering Department of Systems and Automation in 2008 also has promoted the approach to the nonlinear Volterra model based predictive control algorithm.

The performance of the algorithms has been tested by programs written in MATLAB. Besides the simulation examples case studies have been also analyzed using data from laboratory experiments (two-tank system, ball and beam, Dept. of Automation and Applied Informatics, BME, distillation column, Laboratory of Process Control, Cologne).

The research work has been done in tight cooperation with the Laboratory of Process Control, Institute of Process Engineering and Plant Design, University of Applied Science, Cologne.
2. Predictive PID control algorithms

The most widely used algorithms in practice are the PID control algorithms. The algorithms are simple, and with three effects (proportional, integrating and differentiating) generally the quality specifications prescribed for the control system can be met. The proportional path of the algorithm considers the effect of the actual error, the integrating part reacts according to the past history of the error signal, while the differentiating path acts considering the change in the error signal, taking into consideration the initial future trend.

Nevertheless in case of big dead time in the process the performance of the control system will be slow, the PID controller can not accelerate the control system significantly. There are some discrete control algorithms as Smith predictor or dead-beat control, which provide faster performance than PID control for dead time systems, but these algorithms did not get really a wide industrial acceptance, especially because of their sensitivity to plant-model mismatch. Nevertheless, there are some methods to make the Smith predictor less sensitive to parameter uncertainties (Normey-Rico and Camacho, 2007, [41]).

Predictive control algorithms, where predicted error values are used to calculate the actual control signal are also widely applied. Predictive algorithms provide good performance especially in case of big dead time and if the future reference trajectory is known. Applications of predictive control algorithms are supported by different industrial software packages. Nowadays besides PID control predictive control has gained increased acceptance in practical control systems.

As operators of industrial process control systems are familiar with PI(D) controllers and have expertise in PI(D) controller tuning, it would be advantageous to enhance the performance of PI(D) controllers with predictive properties, while applying the well accepted PI(D) tuning rules. In this way the operator gets a PI(D) controller with hidden predictive properties.

The properties of the two algorithms – predictive and PI(D) - can be combined. The idea of predictive PI(D) controllers presented here was initiated by (Katebi and Moradi, 2001, [35] and Johnson and Moradi, 2005, [34]).

2.1 Predictive PI(D) control structure

The widely applied PI(D) controllers calculate the value of the actual manipulated variable considering the actual error signal, which is the difference between the actual
reference signal and the actual output signal. A predictive PI(D) controller considers a predicted value of the error signal. Therefore it can provide faster performance than the usual PI(D) controller in case of big dead time. An extension of the predictive PI(D) controller is when a series of the predicted error values are taken into account, which are calculated as the difference of the predicted reference signal values and the predicted values of the output signal in a given prediction horizon. In this way parallel connected PI(D) controllers are used in the control system. (Katebi and Moradi, 2001, [35] suggested a predefined number of parallel connected PI(D) controllers with inputs of the predicted error signal values. For all controller paths the same PI(D) controller is applied. The block diagram of the predictive PI(D) control system is shown in Fig. 2.1.1, where \( \hat{e}(k + d + 1 + n_e | k) \) denotes the predicted values of the error signal, \( d \) is the dead time, \( n_{e1} \) is the first, while \( n_{e2} \) is the last point of the prediction horizon. The number of the parallel paths is \( m = n_{e2} - n_{e1} + 1 \). The predicted values of the reference and the output signal are denoted by \( y_r(k + d + 1 + n_e) \) and \( \hat{y}(k + d + 1 + n_e | k) \), respectively.

![Figure 2.1.1. Predictive PI(D) controller structure with parallel paths](image)

### 2.2 Predictive PI control algorithm

The form of a non-predictive discrete-time PI controller is

\[
u(k) = k_c e(k) + k_i \sum_{j=1}^{i} e(j)
\]

(2.2-1)
where \( e \) denotes the error signal and \( k_{cp} \), \( k_{ci} \) are the coefficients of the proportional and the integrating components of the controller.

Taking the difference on both sides of (2.2-1) at step \( k \) and \( k-1 \) leads to

\[
\Delta u(k) = u(k) - u(k - 1) = k_{cp}[e(k) - e(k - 1)] + k_{ci}e(k) = p_0 e(k) + p_1 e(k - 1)
\]  

(2.2-2)

where

\[
p_0 = k_{cp} + k_{ci} \quad \text{and} \quad p_1 = -k_{cp} \quad \text{or} \quad k_{cp} = -p_1 \quad \text{and} \quad k_{ci} = p_0 + p_1
\]  

(2.2-3)

In predictive PI control the manipulated variable is the sum of the controller outputs based on the predicted control errors with different prediction horizons. Applying the algorithm on a future error signal \( d + 1 + n_e \) step ahead the actual time point the corresponding control increment is obtained as

\[
\Delta u_n(k) = p_0\hat{e}(k + d + 2 + n_e | k) + p_1\hat{e}(k + d + 1 + n_e | k) 
\]  

(2.2-4)

Let us introduce the following vector notations:

\[
k_{pi} = [p_1, p_0]^T
\]  

(2.2-5)

\[
\hat{e}_{pi}(k + d + 1 + n_e | k) = [\hat{e}(k + d + 1 + n_e | k), \hat{e}(k + d + 2 + n_e | k)]^T
\]  

(2.2-6)

With these notations (2.2-4) can be written in the following form:

\[
\Delta u_n(k) = k_{pi}^T\hat{e}_{pi}(k + d + 1 + n_e | k) 
\]  

(2.2-7)

2.2.1 One-step-ahead predictive PI control

A special case of predictive control is extended horizon (one-step-ahead) predictive control, where \( n_x = n_{e1} = n_{e2} \). The future error signal is estimated as the difference between the future reference signal and the predicted output signal. The output prediction in a future point is obtained by the sum of the forced and the free response.
The forced response can be obtained in the knowledge of the points of the step response \( (h_{n+1}, h_{n+2}, \ldots) \). Supposing only one change in the control signal

\[
\Delta u(k) = \Delta u_{n_e}(k) = k_p^T \hat{e}_p (k + d + 1 + n_e | k) = k_p^T \begin{bmatrix}
y_r (k + d + 1 + n_e) - \hat{y}(k + d + 1 + n_e | k) \\
y_r (k + d + 2 + n_e) - \hat{y}(k + d + 2 + n_e | k)
\end{bmatrix}
\]

\[
= k_p^T \begin{bmatrix}
y_r (k + d + 1 + n_e) - h_{n+1} \Delta u(k) - \hat{y}_{free}(k + d + 1 + n_e | k) \\
y_r (k + d + 2 + n_e) - h_{n+2} \Delta u(k) - \hat{y}_{free}(k + d + 2 + n_e | k)
\end{bmatrix}
\]

Introducing the notations

\[
h_{PI,n_e} = [h_{n+1}, h_{n+2}]^T
\]

\[
y_{PI,n_e} (k + d + 1 + n_e) = [y_r (k + d + 1 + n_e), y_r (k + d + 2 + n_e)]^T
\]

and

\[
\hat{y}_{free,PI,n_e} (k + d + 1 + n_e | k) = [\hat{y}_{free}(k + d + 1 + n_e | k), \hat{y}_{free}(k + d + 2 + n_e | k)]^T
\]

equation (2.2-8) is written as

\[
\Delta u(k) = k_p^T [y_{PI,n_e} - h_{PI,n_e} \Delta u(k) - \hat{y}_{free,PI,n_e}]
\]

and \( \Delta u(k) \) is expressed as

\[
\Delta u(k) = [I + k_p^T h_{PI,n_e}]^{-1} k_p^T (y_{PI,n_e} - \hat{y}_{free,PI,n_e})
\]

### 2.2.2 Long-range predictive PI control

In long-range predictive PI control the manipulated variable is obtained considering the sum of the predicted control errors.

The control increment can be expressed as

\[
\Delta u(k) = k_p^T \sum_{n_e=n_1}^{n_2} \hat{e}_p (k + d + 1 + n_e | k)
\]
As $\Delta u(k), \Delta u(k+1), \ldots$ appear on the right side the equation above, an analytical expression for the control can be given considering some assumptions. With the assumption $\Delta u(k+i) = 0; \ i = 1, 2, \ldots $ (2.2-12) can be given in detailed form as

$$
\Delta u(k) = k_T^p \left[ \sum_{n_r=n_r_1}^{n_r_2} \sum_{n_i=n_i_1}^{n_i_2} h_{n_r+1} \right] - k_T^p \left[ \sum_{n_r=n_r_1}^{n_r_2} \sum_{n_i=n_i_1}^{n_i_2} h_{n_r+2} \right] - \Delta u(k) - k_T^p \left[ \sum_{n_r=n_r_1}^{n_r_2} \sum_{n_i=n_i_1}^{n_i_2} \hat{y}_{free}(k + d + 1 + n, k) \right]
$$

Let us introduce the following notations:

$$
\mathbf{h}_{pl} = \left[ \sum_{n_r=n_r_1}^{n_r_2} \sum_{n_i=n_i_1}^{n_i_2} h_{n_r+1} \right]^T \quad \text{and} \quad \mathbf{y}_{pl, sum} = \left[ \sum_{n_r=n_r_1}^{n_r_2} \sum_{n_i=n_i_1}^{n_i_2} y_{n_r+1} \right] \quad \text{and} \quad \hat{y}_{free, pl, sum} = \left[ \sum_{n_r=n_r_1}^{n_r_2} \sum_{n_i=n_i_1}^{n_i_2} \hat{y}_{free} \right]
$$

From (2.2-13) the control increment can be expressed similarly to (2.2-11) as

$$
\Delta u(k) = \left[ 1 + k_T^p \mathbf{h}_{pl} \right]^T k_T^p \left( \mathbf{y}_{pl, sum} - \hat{y}_{free, pl, sum} \right)
$$

I suggested two more strategies (introducing a simplified conception of control horizon in context of predictive PI(D) control):

1. $\Delta u(k+i) = 0; \ i = 1, 2, \ldots $ (discussed above, strategy 1).
2. $\Delta u(k) = \Delta u(k+1) = \Delta u(k+2) = \ldots = \Delta u(k+m-1)$; (equal control increments are supposed in the prediction horizon, strategy 2).
3. A given $n_u$ number for consecutive equal control increments is supposed (strategy 3).

In the three cases the calculation of $\mathbf{h}_{pl}$ is different.
In case 1 \[ h_{p1} = \left[ \sum_{n_e=n_{e1}}^{n_e} h_{n_e+1} \right] \] ; in case 2 \[ h_{p2} = \left[ \sum_{n_e=n_{e1}}^{n_e} h_{n_e+2} \sum_{n_e=n_{e1}}^{n_e} (m+1-n_e)h_{n_e+2} \right] \] ;

in case 3 \[ h_{p3} = \left[ \sum_{n_e=n_{e1}}^{n_e} h_{n_e+1} \sum_{n_e=n_{e1}}^{n_e} (n_u+1-n_e)h_{n_e+2} \right] \] (2.2-16)

where \( m \) is the number of the parallel paths.

For different systems the forced response can be calculated in the knowledge of the points of the step response, while the free response is obtained from the parameters of the model and from the past inputs and the actual and past output signals.

### 2.2.3 Predictive PI control of a first-order process with dead time

Aperiodic processes can be approximated well by a first-order process with dead time. In the process industries a lot of processes can be described by this model. In most cases the step response of the system can be measured easily even within industrial circumstances. A good, but slow control of this process can be achieved by a PI controller. Different practical tuning rules exist considering the parameters of the approximating first-order model of the process.

Applying predictive PI controller can improve the performance of the control system especially in case of significant dead time. In this case the control algorithm (2.2-15) can be expressed in analytical form.

The first-order system is described by the following CARIMA model:

\[
y(k) = \frac{b_1 q^{-1}}{1+a_1 q^{-1}} q^{-d} u(k) + \frac{1}{(1-q^{-1})(1+a_1 q^{-1})} v_r(k) \] (2.2-17)

The predicted output can be expressed as
\[
\hat{y}(k+d+n_r+1\mid k) = [h_{n+1} \ h_n \ \ldots \ h_1] \begin{bmatrix}
\Delta u(k) \\
\Delta u(k+1) \\
\vdots \\
\Delta u(k+n_r)
\end{bmatrix} + [f_{d+n+1,1} \ f_{d+n+1,2}] \begin{bmatrix}
y(k) \\
y(k-1) \\
\vdots \\
y(k-d)
\end{bmatrix}
\]

The first term on the right side of the equation gives the forced response, while the second and the third terms give the free response. If there is no dead time, i.e. \( d=0 \), the last term on the right side of the equation is missing.

\( h_1, h_2, \ldots, \) are the points of the step response, and \( f_{d+n+1,1}, f_{d+n+1,2} \) are the coefficients in row \( d+n+1 \) of the following \( \mathbf{f}_1 \) and \( \mathbf{f}_2 \) vectors:

\[
\mathbf{f}_1 = \begin{bmatrix}
1-a_i \\
1-a_i+a_i^2 \\
1-a_i+a_i^2-a_i^3 \\
\vdots
\end{bmatrix} \quad \mathbf{f}_2 = \begin{bmatrix}
a_i \\
(1-a_i)a_i \\
(1-a_i+a_i^2)a_i \\
\vdots
\end{bmatrix}
\quad (2.2-19)
\]

The points of the step response can be calculated from the parameters of the pulse transfer function using the following relationship:

\[
\frac{1}{1-q^{-1} + a_i q^{-2}} = h_1 q^{-1} + h_2 q^{-2} + \ldots
\]

Multiplying both sides of the equation by the denominator and comparing the coefficients the points of the step response are obtained as

\[
h_1 = b_1; \ h_2 = b_1(1-a_i); \ h_3 = a_i h_1 + (1-a_i) h_2 = b_1(1-a_i+a_i^2); \ldots; h_k = a_i h_{k-2} + (1-a_i) h_{k-1}
\quad (2.2-20)
\]

Let us define vector \( \hat{y}(k+d+1+n_r) \) as

\[
\hat{y}(k+d+1+n_r) = \begin{bmatrix}
\hat{y}(k+d+1+n_r) \\
\hat{y}(k+d+2+n_r)
\end{bmatrix}
\quad (2.2-21)
\]
Taking into consideration the predicted output equation, it can be expressed as

\[
\hat{y}(k + d + 1 + n_y) = \begin{bmatrix} \hat{y}(k + d + 1 + n_y) \\ \hat{y}(k + d + 2 + n_y) \end{bmatrix} = \begin{bmatrix} h_{n_y+1} & h_{n_y} & \cdots & h_2 & h_1 \\ h_{n_y+2} & h_{n_y+1} & \cdots & h_2 & h_1 \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k + 1) \\ \vdots \\ \Delta u(k + n_y) \\ \Delta u(k + 1 + n_y) \end{bmatrix} + \begin{bmatrix} f_{d+n_y+1,1} & f_{d+n_y+1,2} \\ f_{d+n_y+2,1} & f_{d+n_y+2,2} \end{bmatrix} y(k) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1) + \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix} y(k - 1)
\]

If there is no dead time, i.e. \(d=0\), the last term on the right side of equation (2.2-22) is missing.

Let us write (2.2-22) in the following form:

\[
\hat{y}(k + d + 1 + n_y) = H_{n_y} \Delta u_f + F_{p_{n_y}}^p y_p + H_{n_y}^p \Delta u_p
\]

where

\[
H_{n_y} = \begin{bmatrix} h_{n_y+1} & h_{n_y} & \cdots & h_2 & h_1 \\ h_{n_y+2} & h_{n_y+1} & \cdots & h_2 & h_1 \end{bmatrix};
F_{p_{n_y}}^p = \begin{bmatrix} f_{d+n_y+1,1} & f_{d+n_y+1,2} \\ f_{d+n_y+2,1} & f_{d+n_y+2,2} \end{bmatrix};
H_{n_y}^p = \begin{bmatrix} h_{n_y+2} & h_{n_y+3} & \cdots & h_{n_y+1+d} \\ h_{n_y+3} & h_{n_y+4} & \cdots & h_{n_y+2+d} \end{bmatrix}
\]

and

\[
\Delta u_f = \begin{bmatrix} \Delta u(k) \\ \Delta u(k + 1) \\ \vdots \\ \Delta u(k + n_y) \\ \Delta u(k + 1 + n_y) \end{bmatrix};
\Delta u_p = \begin{bmatrix} \Delta u(k - 1) \\ \Delta u(k - 2) \\ \vdots \\ \Delta u(k - d) \end{bmatrix};
\]

\[
y_p = \begin{bmatrix} y(k) \\ y(k - 1) \end{bmatrix};
\]

(2.2-24)
(Index $f$ refers to future data, while index $p$ indicates past effects.)

If $n_u = 1$, i.e. for the future control increments $\Delta u(k + i) = 0$ is supposed, only the first column of $H_{n_u}$ is considered, $H_{n_u} \equiv h_{PL,n_u} = [h_{n_u+1}, h_{n_u+2}]^T$. In the sequel this assumption is taken into account.

The control algorithm can be written according to (2.2-15), where the free response is obtained by the second and the third terms of the right side of (2.2-23).

In case of one-step-ahead predictive control the control increment is obtained as

$$\Delta u(k) = \left[1 + k_{p}^{T} H_{PL,n_u} \right]^{-1} k_{p}^{T} (y_{PL,n_u} - F_{m}^{p} y_{p} - H_{n_u}^{p} \Delta u_{p})$$

(2.2-25)

In case of long-range predictive control

$$\Delta u(k) = \left[1 + k_{p}^{T} H_{PL} \right]^{-1} k_{p}^{T} (y_{PL,sum} - \hat{y}_{free,sum})
\quad = \left[1 + k_{p}^{T} H_{PL} \right]^{-1} k_{p}^{T} (y_{PL,sum} - \sum_{n_{r}}^{n_{c+1}} F_{m}^{p} y_{p} - \sum_{n_{r}}^{n_{c+1}} H_{n_r}^{p} \Delta u_{p})$$

(2.2-26)

I gave the control algorithm in detailed form in (Arous et al., [1,2,3,4]) in case of strategy 1 as follows.

The control algorithm for strategy 1 can be written as

$$\Delta u(k) = k_{p} [y_{r}(k + d + 1 + n_{r}) - y_{r}(k + d + n_{r})] - k_{p} \Delta u(k) h_{m}
\quad - k_{p} \left[\hat{y}_{free}(k + d + 1 + n_{r}) - \hat{y}_{free}(k + d + 1 + n_{r}) \right]
\quad + k_{e} \left[\sum_{n_{r} = n_{c+1}}^{n_{r+1}} [y_{r}(k + d + 1 + n_{r}) - \hat{y}_{free}(k + d + 1 + n_{r})] - \sum_{n_{r} = n_{c+1}}^{n_{r+1}} h_{m+2} \Delta u(k) \right]$$

(2.2-27)

With some mathematical manipulations the control rule can be written as
\[ \Delta u(k) = k_p \left[ y_r (k + d + 1 + n_{e1}) - y_r (k + d + 1 + n_{e2}) - \hat{y}_{pree} (k + d + 1 + n_{e1}) + \hat{y}_{pree} (k + d + 1 + n_{e2}) \right] \]
\[ + \frac{k_i}{\Delta t} \sum_{n_{e1}}^{n_{e2}} \left[ y_r (k + d + 1 + n_{e1}) - \hat{y}_{pree} (k + d + 1 + n_{e1}) \right] \]
\[ k_u \]

where

\[ k_u = 1 + k_{ip} h_u + k_{ci} \sum_{n_{e1}}^{n_{e2}} h_{ue1} + k_{ci} \sum_{n_{e1}}^{n_{e2}} h_{ue2} \]
for strategy 1

In case of strategy 2 and 3

\[ k_u = 1 + k_{ip} \sum_{n_{e1}}^{n_{e2}} h_{ue1} + k_{ci} \sum_{n_{e1}}^{n_{e2}} (m - n_{e1} + 1) h_{ue1} \]
\[ k_u = 1 + k_{ip} \sum_{n_{e1}}^{n_{e2}} h_{ue1} + k_{ci} \sum_{n_{e1}}^{n_{e2}} (m - n_{e1} + 1) h_{ue2} \]
this is for strategy 2

2.3 Predictive PID control algorithm

The form of a non-predictive discrete-time PID controller is

\[ u(k) = k_p e(k) + k_i \sum_{i=1}^{k} e(i) + k_d \left[ e(k) - e(k-1) \right] \]  

(2.3-1)

\[ k_p, k_i \text{ and } k_d \] are the coefficients of the proportional, the integrating and the derivative components of the controller. Taking the difference on both sides of (2.3-1) at step \( k \) and \( k-1 \) leads to

\[ \Delta u(k) = u(k) - u(k-1) = k_p [e(k) - e(k-1)] + k_i e(k) + k_d [e(k) - 2e(k-1) + e(k-2)] = (k_p + k_i + k_d) e(k) + (-k_c + 2k_d) e(k-1) + k_d e(k-2) = p_0 e(k) + p_1 e(k-1) + p_2 e(k-2) \]  

(2.3-2)

where

\[ p_0 = k_p + k_i + k_d \]
\[ k_p = -p_1 - 2p_2 \]
\[ p_1 = -k_c - 2k_d \]
\[ k_d = p_1 + p_2 \]
\[ k_i \]

(2.3-3)
In predictive PID control the manipulated variable is the sum of the controller outputs based on the predicted control errors with different prediction horizons.

Applying the algorithm on a future error signal \( d + 1 + n_c \) step ahead the actual time point the corresponding control increment is obtained as

\[
\Delta u_{n_c}(k) = p_0 \hat{e}(k + d + 3 + n_c | k) + p_1 \hat{e}(k + d + 2 + n_c | k) + p_2 \hat{e}(k + d + 1 + n_c | k)
\]

(2.3-4)

The future error signals are predicted on the basis of the information available till the actual time point \( k \).

Let us introduce the following vector notations:

\[
k_{PID} = [p_2, p_1, p_0]^T \quad \text{(2.3-5)}
\]

\[
\hat{e}_{PID}(k + d + 1 + n_c | k) = [\hat{e}(k + d + 1 + n_c | k), \hat{e}(k + d + 2 + n_c | k), \hat{e}(k + d + 3 + n_c | k)]^T \quad \text{(2.3-6)}
\]

With these notations (2.3-4) can be written in the following form:

\[
\Delta u_{n_c}(k) = k^T_{PID} \hat{e}_{PID}(k + d + 1 + n_c | k) \quad \text{(2.3-7)}
\]

### 2.3.1 One-step-ahead predictive PID control

A special case of predictive control is extended horizon (one-step-ahead) predictive control, where only one control channel is considered \( n_c = n_{c1} = n_{c2} \).

In expression (2.3-7) the predicted error values are calculated as the difference between the future reference signals \( y_c(k + d + 1 + n_c) \) and the predicted output signals \( \hat{y}(k + d + 1 + n_c | k) \). As the predicted output values contain effects of the forced response and the free response, the actual and the next two future control increments do appear also on the right side of the expression. A closed form to calculate \( \Delta u(k) \) can be obtained simply only if some assumptions are considered for the future control increments, e.g. \( \Delta u(k + 1) = \Delta u(k + 2) = 0 \) is supposed (Katebi and Moradi, 2001, [35]). The forced response is calculated from the knowledge of the points of the step response \( \{ h, i = 1,2,3 \} \).
\[
\begin{align*}
\Delta u(k) = \Delta u_{n_e}(k) &= \mathbf{k}_{PID}^T \begin{bmatrix}
\hat{e}(k + d + 1 + n_e | k) \\
\hat{e}(k + d + 2 + n_e | k) \\
\hat{e}(k + d + 3 + n_e | k)
\end{bmatrix} \\
&= \mathbf{k}_{PID}^T \begin{bmatrix}
y_r(k + d + 1 + n_e) - \hat{y}(k + d + 1 + n_e | k) \\
y_r(k + d + 2 + n_e) - \hat{y}(k + d + 2 + n_e | k) \\
y_r(k + d + 3 + n_e) - \hat{y}(k + d + 3 + n_e | k)
\end{bmatrix}
\end{align*}
\]

(2.3-8)

where \( n_e \) is a chosen extension point beyond the dead time plus one \((d+1)\).

Let us introduce the following notations:

\[
\mathbf{h}_{PID,n_e} = [h_{n_e+1}, h_{n_e+2}, h_{n_e+3}]^T
\]

\[
y_{PID,n_e}(k + d + 1 + n_e) = [y_r(k + d + 1 + n_e), y_r(k + d + 2 + n_e), y_r(k + d + 3 + n_e)]^T
\]

and

\[
\hat{y}_{free_{PID,n_e}}(k + d + 1 + n_e | k) = [\hat{y}_{free}(k + d + 1 + n_e | k), \hat{y}_{free}(k + d + 2 + n_e | k), \hat{y}_{free}(k + d + 3 + n_e | k)]^T
\]

With these notations (2.3-8) is written as

\[
\Delta u(k) = \mathbf{k}_{PID}^T [y_{PID,n_e} - \mathbf{h}_{PID,n_e} \Delta u(k) - \hat{y}_{free_{PID,n_e}}]
\]

(2.3-10)

and \( \Delta u(k) \) is expressed as

\[
\Delta u(k) = \Delta u_{n_e}(k) = [1 + \mathbf{k}_{PID}^T \mathbf{h}_{PID,n_e}] \mathbf{k}_{PID}^T [y_{PID,n_e} - \hat{y}_{free_{PID,n_e}}]
\]

(2.3-11)

2.3.2 Long-range predictive PID control

In long-range predictive PID control the manipulated variable is obtained considering the sum of the predicted control errors. The control increment can be expressed as

\[
\Delta u(k) = \mathbf{k}_{PID}^T \sum_{n_e=n_{c1}}^{n_{e2}} \hat{e}_{PID}(k + d + 1 + n_e | k)
\]

(2.3-12)
The predicted error values are calculated as the difference between the future reference and the predicted output signals. As the predicted output signals contain the effect of the future control increments as well, again a closed form to calculate \( \Delta u(k) \) can be obtained simply only if some assumptions are considered for the future control increments, e.g. \( \Delta u(k + i) = 0; \ i = 1,2,... \) is supposed. This means that the control horizon is \( n_u = 1 \), i.e. only one change is supposed in the control signal. (Another assumption which leads to simple solution could suppose a given \( n_u \) number of subsequent equal changes in the control signal.)

With this assumption expression (2.3-12) can be given by means of (2.3-8) in detailed form as

\[
\Delta u(k) = k_{PID}^T \left[ \sum_{n_r+n_i=n_1}^{n_r+n_i=3} y_r(k + d + 1 + n_r) \sum_{n_r+n_i=n_1}^{n_r+n_i=3} h_{n_r+1} \sum_{n_r+n_i=n_1}^{n_r+n_i=3} h_{n_r+3} \right] - k_{PID}^T \left[ \sum_{n_r+n_i=n_1}^{n_r+n_i=3} \hat{y}_{free}(k + d + 1 + n_r) \sum_{n_r+n_i=n_1}^{n_r+n_i=3} \hat{y}_{free}(k + d + 2 + n_r) \sum_{n_r+n_i=n_1}^{n_r+n_i=3} \hat{y}_{free}(k + d + 3 + n_r) \right] \Delta u(k) - k_{PID}^T \left[ \sum_{n_r+n_i=n_1}^{n_r+n_i=3} \hat{y}_{free}(k + d + 1 + n_r) \sum_{n_r+n_i=n_1}^{n_r+n_i=3} \hat{y}_{free}(k + d + 2 + n_r) \sum_{n_r+n_i=n_1}^{n_r+n_i=3} \hat{y}_{free}(k + d + 3 + n_r) \right]
\]

(2.3-13)

Let us introduce the following notations:

\[
h_{PID} = \left[ \sum_{n_r+n_i=n_1}^{n_r+n_i=3} h_{n_r+1} \sum_{n_r+n_i=n_1}^{n_r+n_i=3} h_{n_r+2} \sum_{n_r+n_i=n_1}^{n_r+n_i=3} h_{n_r+3} \right]^T
\]

\[
y_{rPID,\text{sum}} = \left[ \sum_{n_r+n_i=n_1}^{n_r+n_i=3} y_r(k + d + 1 + n_r) \sum_{n_r+n_i=n_1}^{n_r+n_i=3} y_r(k + d + 2 + n_r) \sum_{n_r+n_i=n_1}^{n_r+n_i=3} y_r(k + d + 3 + n_r) \right]^T
\]

\[
\hat{y}_{\text{freePID,\text{sum}}} = \left[ \sum_{n_r+n_i=n_1}^{n_r+n_i=3} \hat{y}_{\text{free}}(k + d + 1 + n_r) \sum_{n_r+n_i=n_1}^{n_r+n_i=3} \hat{y}_{\text{free}}(k + d + 2 + n_r) \sum_{n_r+n_i=n_1}^{n_r+n_i=3} \hat{y}_{\text{free}}(k + d + 3 + n_r) \right]^T
\]

(2.3-14)

From (2.3-13) the control increment can be expressed as

\[
\Delta u(k) = \left[ 1 + k_{PID}^T h_{PID} \right]^{-1} k_{PID}^T \left[ y_{rPID,\text{sum}} - \hat{y}_{\text{freePID,\text{sum}}} \right]
\]

(2.3-15)
2.3.3 Predictive PID control of a second-order process with dead time

Second order processes can be controlled by PID control algorithm which can be applied both for aperiodic and oscillating processes. Processes with oscillating behavior frequently can be approximated by a second-order model containing two dominant conjugate complex poles. Enhancing the PID control with predictive properties will improve the control performance especially in the case of big dead time.

The CARIMA model of a second-order process with dead time is given by

\[ y(k) = \frac{b_1 q^{-1} + b_2 q^{-2}}{1 + a_1 q^{-1} + a_2 q^{-2}} q^{-d} u(k) + \frac{T(q^{-1})}{(1 - q^{-1})(1 + a_1 q^{-1} + a_2 q^{-2})} v_u(k) \] (2.3-16)

Supposing \( T(q^{-1}) = 1 \) I derived the general rule for the prediction of the output signal. The following relationship is given

\[ \hat{y}(k + d + n_e + 1 | k) = \begin{bmatrix} h_{n_e+1} & h_{n_e} & \ldots & h_1 \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k + 1) \\ \vdots \\ \Delta u(k + n_e) \end{bmatrix} + \begin{bmatrix} f_{d+n_e+1,1} & f_{d+n_e+1,2} & f_{d+n_e+1,3} \end{bmatrix} \begin{bmatrix} y(k) \\ y(k - 1) \\ y(k - 2) \end{bmatrix} \]

\[ + \begin{bmatrix} h_{n_e+2} & h_{n_e+3} & \ldots & h_{n_e+d+2} \end{bmatrix} \begin{bmatrix} \Delta u(k - 1) \\ \Delta u(k - 2) \\ \vdots \\ \Delta u(k - d - 1) \end{bmatrix} \] (2.2-17)

where \( h_i \) are the values of the step response and \( f_{d+n_e+1,1}, f_{d+n_e+1,2}, f_{d+n_e+1,3} \) are the elements in the \( d + n_e + 1 \)-th row and first, second or third column of the following matrix:

\[
F = \begin{bmatrix}
1-a_1 & a_1-a_2 & a_2 \\
(a_1-a_2)q^{-1}+a_1 & a_1 & a_2q^{-1} \\
(a_1-a_2)q^{-2}+(a_1-a_2)a_1 & (a_1-a_2)q^{-2} & a_2q^{-2} \\
(1-a_1)(1-a_1)q^{-3}+(a_1-a_2)a_2 & (1-a_1)(1-a_1)q^{-3}+a_2 & a_2(1-a_2)
\end{bmatrix}
\] (2.2-18)

On the right side of the predictive model the first term is the forced response, the second and third terms provide the free response.

For this model the predictive PID algorithm in (2.2-15) can be applied.
2.4 Equivalence between the predictive PI(D) algorithm and the GPC algorithm

For tuning of predictive PI(D) algorithms an equivalence between GPC and predictive PI(D) algorithms can be searched.

The GPC control algorithm is obtained by minimizing the cost function

$$J = \sum_{n_i=n_i,1}^{n_{1i}} \left[ \hat{e}^2_i(k + 1 + n_i) \right] + \sum_{i=1}^{n_{1i}} \lambda_u \Delta u^2(k + i - 1) \quad (2.4-1)$$

where $\lambda_u$ is a factor punishing the control increments.

Without constraints minimization of the cost function gives the following analytical solution:

$$\Delta u = \left[ H^T H + \lambda_u I \right]^{-1} H^T (y - y_{free}) \quad (2.4-2)$$

$H$ is the matrix containing the points of the step response

$$H = \begin{bmatrix} h_{(n_i,1)} & h_{(n_i,1-1)} & \cdots & h_{(n_i,n_i+1)} \\ h_{(n_i+1,1)} & h_{(n_i,2)} & \cdots & h_{(n_i,n_i+1)} \\ \vdots & \vdots & \vdots & \vdots \\ h_{(n_i,2)} & h_{(n_i,2-1)} & \cdots & h_{(n_i,n_i+1)} \end{bmatrix}$$

Only the first element of vector $\Delta u$ is used as a control input, and in the next time point the procedure is repeated (receding horizon strategy). As predictive PI(D) algorithms have been derived only for $n_u = 1$, for the GPC algorithm also this assumption is considered.

2.4.1 Equivalence in case of extended horizon control

I have derived the equivalence of the two algorithms in case of extended horizon (one-step-ahead) predictive control.
Comparing (2.4-2) and (2.2-11) or (2.4-2) and (2.3-11) predictive PI or PID algorithms can be given, which generate similar behavior to the GPC algorithm. In this case the two tuning parameters are \( n_e \) and \( \lambda_u \).

In case of the PID algorithm in (2.3-11) \( n_{e1} = n_e \) and \( n_{e2} = n_e + 2 \), so the prediction considers 3 future points from \( k + d + n_e + 1 \) till \( k + d + n_e + 3 \).

The equivalent predictive PID parameters are obtained by solving the following equation:

\[
\begin{bmatrix}
1 + k_{PID}^T & \lambda_u
\end{bmatrix}^{-1} k_{PID}^T = \left[ H^T H + \lambda_u I \right]^{-1} \sum_{n=0}^{N} h_n^2 + \lambda_u \left[ h_{n+1}, h_{n+2}, h_{n+3} \right] \tag{2.4-3}
\]

or in detail

\[
\frac{[p_2, p_1, p_0]}{1 + p_2 h_{n+1} + p_1 h_{n+2} + p_0 h_{n+3}} = \frac{[h_{n+1}, h_{n+2}, h_{n+3}]}{\lambda_u + h_{n+1}^2 + h_{n+2}^2 + h_{n+3}^2} = \frac{[h_{n+1}, h_{n+2}, h_{n+3}]}{N} \tag{2.4-4}
\]

where

\[
N = \lambda_u + h_{n+1}^2 + h_{n+2}^2 + h_{n+3}^2 \tag{2.4-5}
\]

If \( \lambda_u = 1 \), the trivial solution is \( p_0 = h_{n+3}, p_1 = h_{n+2}, p_2 = h_{n+1} \).

In other cases using some mathematical manipulations the following equation has to be solved:

\[
\begin{bmatrix}
p_0 \\
p_1 \\
p_2
\end{bmatrix} = \begin{bmatrix}
-h_{n+1} h_{n+3} / N & -h_{n+1} h_{n+2} / N & 1-h_{n+1}^2 / N \\
-h_{n+2} h_{n+3} / N & 1-h_{n+2}^2 / N & -h_{n+2} h_{n+1} / N \\
1-h_{n+3}^2 / N & -h_{n+3} h_{n+2} / N & -h_{n+3} h_{n+1} / N
\end{bmatrix}^{-1} \begin{bmatrix}
h_{n+1} / N \\
h_{n+2} / N \\
h_{n+3} / N
\end{bmatrix} \tag{2.4-6}
\]

In case of the PI algorithm in (2.3-1) \( n_{e1} = n_e \) and \( n_{e2} = n_e + 1 \), so the prediction considers 2 future points from \( k + d + n_e + 1 \) till \( k + d + n_e + 2 \). The equivalent predictive PI parameters considering also (2.2-11) are obtained by solving the following equation:
\[
\left[ 1 + k_p^T h_{pl,n_u} \right]^{-1} k_p^n = \left[ H^T H + \lambda_u I \right]^{-1} H^T = \left[ \sum_{n_u,n_u+1}^n h_n + \lambda_u \right]^{-1} [h_{n+1}, h_{n+2}]
\]  
(2.4-7)

In detail

\[
\frac{[p_1, p_0]}{1 + p_1 h_{n+1} + p_0 h_{n+2}} = \frac{[h_{n+1}, h_{n+2}]}{\lambda_u + h_{n+1}^2 + h_{n+2}^2} = \frac{[h_{n+1}, h_{n+2}]}{N}
\]
(2.4-8)

where

\[
N = \lambda_u + h_{n+1}^2 + h_{n+2}^2
\]
(2.4-9)

If \( \lambda_u = 1 \), the trivial solution is \( p_0 = h_{n+2} \), \( p_1 = h_{n+1} \). In other cases the following equation has to be solved:

\[
\begin{bmatrix}
   p_0 \\
   p_1 
\end{bmatrix} = \begin{bmatrix}
   -h_{n+1} h_{n+2} / N & 1 - h_{n+1}^2 / N \\
   1 - h_{n+2}^2 / N & -h_{n+2} h_{n+1} / N
\end{bmatrix}^{-1} \begin{bmatrix}
   h_{n+1} / N \\
   h_{n+2} / N
\end{bmatrix}
\]
(2.4-10)

As in case of the predictive PI(D) algorithms the sum of the reference signals in the prediction points is taken into account, in order to set the output to the required reference value the PI(D) gain has to be divided by the number of the parallel channels. That means that the values of \( p_0, p_1, p_2 \) have to be divided in case of extended horizon control by 2 for the PI, and by 3 for the PID algorithm (Arousi et al., 2008, [11]).

### 2.4.2 Equivalence in case of long-range horizon control

For the GPC algorithm the control increment is expressed as

\[
\Delta u(k) = \left[ H^T H + \lambda_u I \right]^{-1} H^T (y_r - y_{free}) = \left[ H^T H + \lambda_u I \right]^{-1} H^T \left[ y_r - F_{GPC}^p y_p - H_{GPC}^p \Delta u_p \right]
\]
(2.4-11)

where the free response is separated to the effect of \( y_p \), vector containing the actual and past output values and the effect of \( \Delta u_p \), vector containing the past inputs, respectively. \( n_u = 1 \) is taken into account.

For predictive PI(D) control the algorithm is
\[ \Delta u(k) = \left[ I + k_{PID}^T h_{PID} \right]^{-1} k_{PID}^T \left[ y_{PID, sum} - y_{PID, free, sum} \right] \]  

\[ = \left[ I + k_{PID}^T h_{PID} \right]^{-1} k_{PID}^T \left[ y_{PID, sum} - F_{PID}^p y_p - H_{PID}^p \Delta u_p \right] \]  

(2.4-12)

It is supposed that \( y_p \) and \( \Delta u_p \) are close to each other for the two algorithms. For longer prediction horizons this assumption can be accepted.

Considering zero reference signals the two algorithms can be compared as (Johnson and Moradi, 2005, [34]):

\[
\begin{bmatrix}
H' H + \lambda_n \mathbf{I} \\
\mathbf{I} \\
\end{bmatrix}^{-1} H' \left[ -F_{GPC}, -H_{GPC}^p \right] \begin{bmatrix} y_p \\ \Delta u_p \end{bmatrix} = \left[ I + k_{PID}^T h_{PID} \right]^{-1} k_{PID}^T \left[ -F_{PID}^p y_p - H_{PID}^p \right] \begin{bmatrix} y_p \\ \Delta u_p \end{bmatrix}
\]

(2.4-13)

Let us introduce the following notations:

\[
k_{GPC} = \begin{bmatrix} H' H + \lambda_n \mathbf{I} \\
\mathbf{I} \\
\end{bmatrix}^{-1} H'
\]

(2.4-14)

\[
k_0 = k_{GPC} \left[ -F_{GPC}^p, -H_{GPC}^p \right]
\]

(2.4-15)

\[
S = \begin{bmatrix} F_{PID}^p, H_{PID}^p \end{bmatrix}
\]

(2.4-16)

With these notations from (2.4-13) the following relationship is obtained:

\[
\left[ I + k_{PID}^T h_{PID} \right]^{-1} k_{PID}^T S = k_0
\]

(2.4-17)

and

\[
k_{PID}^T = k_0 \left[ S - h_{PID} k_0 \right]^{-1}
\]

(2.4-18)

But as the matrix in (2.4-18) generally is not invertible, the solution can be calculated by the following relationship (Johnson and Moradi, 2005, [34]):

\[
k_{PID}^T = k_0 (S - h_{PID} k_0)^T \left[ (S - h_{PID} k_0)(S - h_{PID} k_0)^T \right]^{-1}
\]

(2.4-19)

From \( k_{PID}^T \) the tuning parameters of the PI(D) algorithm can be calculated.
2.4.3 Simulation examples

Example 2.4.1 Calculation of the parameters of the GPC equivalent one-step-ahead predictive PID control

The transfer function of the plant is \( G(s) = \frac{1}{(1+s)(1+2s)} \), the sampling time is \( \Delta T = 1 \). The form of the pulse-transfer function is given by equation (2.3-16). The parameters of the pulse-transfer function are:

\[ b_1 = 0.1548; \quad b_2 = 0.0939; \quad a_1 = -0.9744; \quad a_2 = 0.2231. \]

The points of the step response are:

\[ h_1 = 0.1548; \quad h_2 = 0.3999; \quad h_3 = 0.6035; \quad h_4 = 0.7476; \quad h_5 = 0.8426; \quad h_6 = 0.9029; \]
\[ h_7 = 0.9405; \quad h_8 = 0.9637; \quad h_9 = 0.9779; \quad h_{10} = 0.9866; \ldots \]

The step response is shown in fig. 2.4.1a.

![Step response](image)

Figure 2.4.1a. Step response

For \( n_c = 5 \) and \( \lambda_c = 1 \) and \( \lambda_u = 10 \) the controller parameters are calculated according to (2.4-6) and are given in Table 2.4.1.

<table>
<thead>
<tr>
<th>( \lambda_u = 1 )</th>
<th>( \lambda_u = 10 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_0 ) 0.9637</td>
<td>0.0964</td>
</tr>
<tr>
<td>( P_1 ) 0.9405</td>
<td>0.0941</td>
</tr>
<tr>
<td>( P_2 ) 0.9029</td>
<td>0.0903</td>
</tr>
</tbody>
</table>
It is seen that for $\lambda_c = 1$ the values of the controller parameters are equal to the appropriate step response points. These parameters have to be divided by 3. The equivalence of the extended horizon GPC and predictive PID algorithms is demonstrated in Fig. 2.4.1b In the simulation a positive unit step reference signal acts at time point 5s, and a negative 0.5 step disturbance is applied at time point 30s at the process input. Prediction of the reference signal is taken into account. Fig. 2.4.2 shows the output and the manipulated signals. It has to be mentioned that for longer horizons the equivalence is better.

![Figure 2.4.1b](image)

Figure 2.4.1b Equivalence of predictive PID and GPC algorithms in case of one-step-ahead, extended horizon control
Example 2.4.2 Calculation of the parameters of the GPC equivalent long-range predictive PID control

The plant is the same as in the previous example. For $n_e = 1$ and $n_r = 10$ and $\lambda_u = 1$ the controller parameters calculated according to (2.4-19) are: $p_0 = 1.2818$, $p_1 = -0.4812$, $p_2 = 0.0947$ (which in case of long-range control are not divided, as (2.4-19) considers already the effect of the parallel paths). In the simulation a positive unit step reference signal acts at time point 10s, and a negative 0.5 step disturbance is applied at time point 25s at the process input. Prediction of the reference signal is taken into account. Fig. 2.4.2 gives the output and the manipulated variable signals for the GPC and the equivalent PID control.

Figure 2.4.2. Equivalence of predictive PID and GPC algorithms in case of long-range horizon control

Figure 2.4.3. Step responses of the third-order plant and its first-order approximation
Example 2.4.3 GPC equivalent predictive PI control of a third-order process approximated by a first-order process with dead time

The transfer function of the plant is

\[ G(s) = \frac{1}{(1 + 0.333s)} \].

The step response can be approximated by a first-order process with dead time as \( \hat{G}(s) = \frac{e^{-0.26s}}{1 + 1.25s} \). The first-order approximation is calculated considering the initial tangent of the step response. The step responses of the third-order plant and its first-order approximation are shown in Fig. 2.4.3.

The sampling time is \( \Delta T = 0.1 \) s. Determine the GPC equivalent PI controller parameters considering the first-order approximation if the parameters of the GPC algorithm are \( \lambda_u = 1, n_u = 1, n_{e1} = 1 \) and \( n_{e2} = 10 \), respectively. Let us remark, that the discrete dead time has to be an integer, and is approximated by 3. The parameters of the PI algorithm are: \( p_0 = 0.7723 \) and \( p_1 = -0.3826 \) or equivalently \( k_{ip} = 0.3826 \) and \( k_{ei} = 0.3897 \).

In the simulation a positive unit step reference signal acts at time point 10 s, and a negative 0.5 step disturbance is applied at time point 15 s at the process input. Prediction of the reference signal is taken into account. Fig. 2.4.4a/ shows the controlled output and the manipulated variable in case of GPC control and its equivalent predictive PI control supposing that the plant is exactly the same as the first-order model with dead time. In this case practically the two algorithms provide the same performance.

![Figure 2.4.4a. First-order process with dead time](image1)

![Figure 2.4.4b. Third-order process. The control algorithm considers its approximation by a first-order process with dead time.](image2)
Fig. 2.4.4b/ demonstrates the performance in case of plant/model mismatch, when the plant is of third-order and it is approximated by a first-order model with dead time. It is seen that in case of plant/model mismatch the control shows some oscillations.

Figure 2.4.4. Predictive PI and equivalent GPC control with parameters $\lambda_u = 1$, $n_{v1} = 1$ and $n_{v2} = 10$.

It has to be emphasized that the control system is not stable for all parameter settings. Especially in case of plant-model mismatch it can be sensitive for the appropriate tuning parameters. Appropriate choice of $\lambda_u$ and the prediction horizon generally could ensure stable performance for the given plant. The effect of plant/model mismatch could be compensated with additional robustifying considerations (see Section 2.6).

### 2.5 Tuning of predictive PI(D) algorithms

Besides the GPC equivalence other PI(D) controller tuning rules can also be applied for predictive PI(D) algorithms. Different tuning rules of thumb are known mainly for continuous PI(D) controllers. These rules can be used for digital controllers as well after discretization.

*I applied several tuning rules for predictive PI(D) control and I have shown how to modify them for predictive PI(D) control.*

The continuous PID control algorithm with ideal differentiating effect is given by the relationship

$$u(t) = K_c \left[ e(t) + \frac{1}{T_i} \int_0^t e(t) dt + T_d \frac{de(t)}{dt} \right]$$

(2.5-1)

The discrete-time PID algorithm for the control increment is expressed as

$$\Delta u(k) = p_0 e(k) + p_1 e(k - 1) + p_2 e(k - 2)$$

(2.5-2)

where

$$\Delta u(k) = u(k) - u(k - 1)$$

(2.5-3)
Discretization with sampling time $\Delta T$ can be executed in different ways. Using e.g. the trapezoid rule for approximate integration

$$
\Delta u(k) = K_c(e(k) - e(k - 1)) + \frac{K}{2T_i}(e(k) + e(k - 1)) + \frac{K T_d}{\Delta T}(e(k) - 2e(k - 1) + e(k - 2))
$$

Comparing (2.5-2) and (2.5-4) the following relationships are obtained:

$$
\begin{align*}
    p_0 &= K_c \left(1 + \frac{\Delta T}{2T_i} + \frac{T_d}{\Delta T}\right) ;
    p_1 &= -K_c \left(1 - \frac{\Delta T}{2T_i} + \frac{2T_d}{\Delta T}\right) ;
    p_2 &= K_c \frac{T_d}{\Delta T}
\end{align*}
$$

For tuning of the predictive PID control I suggest the following rule:

Parallel controller paths are applied. If the tuning is done based on continuous-time control, the controller gain has to be divided by the number of the paths, thus all the discrete-time controller parameters $p_0$, $p_1$, $p_2$ have to be divided by $m$ where $m = n_{e2} - n_{e1} + 1$.

There are different tuning rules for aperiodic processes. An aperiodic process can be described by the following transfer function:

$$
P(s) = \frac{K_p}{(1 + sT_1)(1 + sT_2) \ldots (1 + sT_p)} e^{-sT_d}
$$

where $K_p$ is the static gain, $T_1, T_2, \ldots, T_p$ are the time constants, and $T_d$ is the dead time. $T_\Sigma$ is defined as the sum of the dead time and the time constants:

$$
T_\Sigma = T_d + \sum_{i=1}^{p} T_i
$$

Rules of thumb suggested by Kuhn, (1995, [36]) for the coefficients of the continuous-time PI(D) controller with higher-order aperiodic processes are given in Table 2.5.1.

<table>
<thead>
<tr>
<th></th>
<th>PI</th>
<th>PID</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_c$</td>
<td>$0.5 / K_p$</td>
<td>$1 / K_p$</td>
</tr>
<tr>
<td>$T_i$</td>
<td>$0.5T_\Sigma$</td>
<td>$0.66T_\Sigma$</td>
</tr>
<tr>
<td>$T_d$</td>
<td>0</td>
<td>$0.167T_\Sigma$</td>
</tr>
</tbody>
</table>

It has to be emphasized that applying the rule for predictive control in $T_\Sigma$ the dead time $T_d$ is not considered, as predictive control based on predicted signals compensates the effect of the dead time. Therefore for systems containing significant
dead time predictive PI(D) control will result faster behavior than the conventional PI(D) control.

**Example 2.5.1 Predictive PI control of a third-order process approximated by a first-order model with dead time using Kuhn’s tuning rule**

The system is the same as in Example 2.4.3. The third-order process is approximated by a first-order system with dead time. The transfer function of the plant is:

\[ G(s) = \frac{1}{(1 + 0.333s)^3} e^{-0.26s} \]

The sampling time is \( \Delta T = 0.1 \).

The prediction horizon is given by \( n_{c1} = 1, n_{c2} = 5 \).

The parameters of the continuous-time PI controller according to Kuhn’s rules (Table 2.5.1) considering \( T_\Sigma = 1s \) are: \( K_c = 0.5 / K_p = 0.5 / 1 = 0.5 \) and \( T_i = 0.5T_\Sigma = 0.5 \cdot 1 = 0.5 \). Considering the discretization (2.5-5) and also dividing by \( m = n_{c2} - n_{c1} + 1 = 5 \) the parameters of the discrete-time controller are follows:

\[
p_0 = \frac{K_c}{m} (1 + \frac{\Delta T}{T_i}) = \frac{0.5}{5} (1 + \frac{0.1}{2 \cdot 0.5}) = 0.11; \quad p_1 = -\frac{K_c}{m} (1 - \frac{\Delta T}{T_i}) = -\frac{0.5}{5} (1 - \frac{0.1}{2 \cdot 0.5}) = -0.09
\]

Applying (2.2-3) the discrete tuning parameters are

\[ k_{cp} = -p_1 = 0.09; \quad k_{ci} = p_0 + p_1 = 0.11 - 0.09 = 0.02. \]

Including physical dead time \( T_d \) the tuning parameters are the same. Different dead times (0, 0.5, 1, 2 and 5) are considered in the process. In the simulation a positive unit step reference signal acts at time point 1s, and a negative unit step disturbance is applied at time point 15s. No prediction of the reference signal is taken into account.

Fig. 2.5.1 shows the controlled and the manipulated variables when the system is of first-order, and its model is accurate, also of first-order with the same parameters. It is seen, that the quality of the control with a stepwise reference signal change is the same for all dead time cases, the outputs are shifted appropriately, while the manipulated variable is the same. The disturbance rejection depends on the dead time. Fig. 2.5.2 gives the controlled and the manipulated variables when the system is of third-order with the dead times above, and the controller is designed according to the first-order approximation. The performance is worse than before, and also with bigger dead time the dynamics is also affected.
There are also other tuning rules based on the approximation of an aperiodic process by a first-order system with dead time (e.g. Chien-Hrones-Reswick, 1952, [20]).

Åström and Hägglund, (1995, [17]) recommended tuning rules considering also the so-called set-point weighting to attenuate too high manipulated variable values.

In the usual feedback control structure using PI(D) controllers the set-point and disturbance control cannot be tuned independently. The same dynamics is achieved for reference signal tracking and output disturbance rejection. On the other hand jumps in the reference signal may cause high values in the manipulated variable through the proportional and the differentiating channels. In case of high over-excitation the manipulated variable may saturate, causing windup phenomena. One solution to avoid too high manipulated signal values is applying a first-order filter after the reference signal. Another method is extending the traditional PI(D) terms by set-point weighting when calculating the manipulated variable, as recommended by Åström and Hägglund (1995, [17]). With these modifications the set-point tracking can be shaped independently from the rejection of a disturbance step. The initial jumps in the control signal can also moderated. Set-point weighting provides faster behavior than the set-point filter.

The original feedback structure is supplemented by an additional path affecting directly the manipulated variable from the reference signal. The structure of the control system is given in Fig. 2.5.3.
I extended the structure of the predictive PI(D) controller shown in Fig. 2.1.1 with set-point weighting. The modified structure is shown in Fig. 2.5.4.

Set-point weighting constant can be equal or can be different for the different paths.

The aperiodic process is approximated by a first-order process with dead time.

\[ G(s) = \frac{K_p}{1 + sT_T} e^{-sT_d} \]  

(2.5-8)

where the meaning of the variables is as follows:

- \( K_p \) static gain of the model,
- \( T_T \) time constant of the model,
- \( T_d \) apparent dead time,

furthermore
\( T_{T,63\%} = 0.63T_r \) is defined as the time to reach 63\% of the final value of the step response after the apparent dead time.

\( T'_c \) is a relative number, introduced to evaluate the ratio of the dead time and the time constant of the process:

\[
T'_c = \frac{T_L}{T_L + T_{T,63\%}}
\]

(2.5-9)

and \( K_r \) is the set-point weighting factor.

The suggested tuning rules for the continuous PI and PID controller by Åström and Hägglund (1995, [17]) are given in Table 2.5.2.

| \( K_c \) | PI \( \frac{0.29 T_{T,63\%}}{K_p T_L} \exp(-2.7T'_c + 3.7T''_L) \) | PID \( \frac{3.8 T_{T,63\%}}{K_p T_L} \exp(-8.4T'_c + 7.3T''_L) \) |
| \( T_i \) | \( 8.9T_L \exp(-6.6T'_c + 3T''_L) \) or \( 0.79T_{T,63\%} \exp(-1.4T'_c + 2.4T''_L) \) | \( 5.2T_L \exp(-2.5T'_c - 1.4T''_L) \) or \( 0.46T_{T,63\%} \exp(2.8T'_c - 2.1T''_L) \) |
| \( T_D \) | 0 | \( 0.89T_L \exp(-0.37T'_c - 4.1T''_L) \) or \( 0.077T_{T,63\%} \exp(5.0T'_c - 4.8T''_L) \) |
| \( K_r \) | \( 0.81\exp(-0.73T'_c + 1.9T''_L) \) | \( 0.40\exp(0.18T'_c + 2.8T''_L) \) |

When two suggestions are given for a parameter, these are close to each other. If both are calculated, a mean value can be taken into account.

For predictive control set-point weighting is applied for all the parallel paths (Arousi et al., 2007, [5], Arousi et al., 2008, [9]). \( K_c \) has to be divided by the number of the parallel paths.

There are other tuning rules, as well. Tuning can be done also based on frequency domain considerations, using e.g. pole cancellation technique.

**Example 2.5.2** Predictive PI(D) control of a third-order process approximated by a first-order system with dead time using the Åström-Hägglund tuning rules
The system is the same as in Example 2.5.1. It is approximated by a first-order process with dead time. The sampling time is $\Delta T = 0.1$ s. The prediction horizon is given by $n_1 = 1$, $n_2 = 5$.

The tuning parameters according to the Åström-Hägglund tuning rules (Table 2.5.2) for the continuous and the discrete case are given in Table 2.5.2. The calculation of the tuning has been executed considering the parameters of the first-order approximation of the system as follows: $T_r = 1.25$, $T_L = 0.26$, $T_{r,63\%} = 0.63T_r = 0.7875$,

$$T''_i = \frac{T_L}{T_i + T_{r,63\%}} = \frac{0.26}{0.26 + 0.7875} = 0.2482$$

Table 2.5.2: Controller parameters

<table>
<thead>
<tr>
<th></th>
<th>$K_c$</th>
<th>$T_i$</th>
<th>$T_D$</th>
<th>$p_0$</th>
<th>$p_1$</th>
<th>$p_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set-point weightings PI ($K_i = 0.7597$)</td>
<td>0.5645</td>
<td>0.52</td>
<td>0</td>
<td>0.6188</td>
<td>-0.51</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.2434</td>
<td>0.65</td>
<td>0.16</td>
<td>6.0054</td>
<td>-9.25</td>
<td>3.5894</td>
</tr>
</tbody>
</table>

The gain of the controller is divided by the number of the paths.

Figs 2.5.5 and 2.5.6 give the controlled and the manipulated variables for predictive PI and PID control, respectively, with tuning parameters calculated according to Kuhn’s and Åström-Hägglund rules applied for the first-order process with dead time. The control is implemented to the third order plant. It is seen, that in case of PI control the performance of the two algorithms is similar. With PID control the Åström-Hägglund method gives a more dynamic behavior.
With the Åström-Hägglund method reference signal tracking can be accelerated by increasing the factor $K_r$ providing set-point weighting. Fig. 2.5.7 shows the predictive PI control of the third-order process tuned according to the Åström-Hägglund rule, but with different values of set-point weighting $K_r$. With set-point weighting according to the rule the control behavior is close to the Kuhn tuning effect. Increasing the weighting the reference signal tracking could be somewhat accelerated. Disturbance rejection is not affected, as expected.

**Example 2.5.3** Predictive PI control of a third-order process approximated by a first-order model with dead time using Kuhn’s tuning rule
The linear process is the same as Example 2.5.1. The sampling time is $\Delta T = 0.1$. The tuning parameters are calculated according to Kuhn’s rules (Table 2.5.1). The discrete tuning parameters are $k_p = -p_1 = 0.09; k_i = p_0 + p_1 = 0.11 - 0.09 = 0.02$.

The prediction horizon is given by: $n_{e_1} = 1, n_{e_2} = 5$ for Fig. 2.5.8 and $n_{e_1} = 1, n_{e_2} = 60$ for Fig. 2.5.9.

The controller is designed considering the first-order approximating model. The plant is simulated according to the third-order description. In the simulation a positive unit step reference signal acts at time points 1 min and a negative unit step disturbance of amplitude -0.5 is applied at time point 15 or 30 min at the process input. Prediction of the reference signal is taken into account. Fig. 2.5.8 shows the controlled and the manipulated signals of the predictive and the non-predictive PI control algorithm with dead time $d = 2$ min. It is seen that the performance of the predictive PI controller is better. Fig. 2.5.9 shows the controlled and the manipulated signals of the predictive PI control algorithm for different assumptions for the control horizon. In strategy 1 $\Delta u(k + i) = 0; i = 1, 2, ...$ is supposed. In strategy 2 $\Delta u(k) = \Delta u(k + 1) = \Delta u(k + 2) = ... = \Delta u(k + m - 1)$ and in strategy 3 a given $n_u$ number for consecutive equal control increments is supposed (in strategy 1 $n_u = 1$ and in strategy 3 $n_u = 30$). It is seen that for strategy 2 the course of the output signal is slower than in case of strategy 1, and the manipulated variable is lower. Fig. 2.5.10 shows the controlled and the manipulated signals for three different prediction horizons, $n_{e_2} = 5, 10$ and 50, respectively. With bigger $n_{e_2}$ values the output signal starts with more steps before the change in the reference signal.

![Figure 2.5.9. Output and control signals of predictive PI for different control strategies](image1)

![Figure 2.5.10. Output and control signals of predictive PI for different prediction horizons](image2)
2.6 Robustifying effects applied for predictive PI(D) control algorithms

Control of a process is based on the process model. As generally there is mismatch between the process and its model, the real performance will differ from the ideal one. The tuning rules discussed above consider the first-order approximation of the process, so plant/model mismatch always exists. As these rules give calm performance, the algorithms generally will tolerate this mismatch. If there is a more significant mismatch, robustifying of the algorithms has to be considered. Control algorithms could be especially sensitive to mismatch in the dead time. In predictive PI(D) control increasing the number of the parallel paths generally reduces the effect of mismatch. Two more possibilities are considered here.

2.6.1 Enhancing robustness of predictive PI(D) control using a noise model with polynomial $T$

The CARIMA process model is given by the following equation:

$$ y(k) = \frac{B(q^{-1})}{A(q^{-1})} z^{-d} u(k) + \frac{T(q^{-1})}{A(q^{-1})(1-q^{-1})} v_{u}(k) $$

where

$$ B(q^{-1}) = b_0 + b_1 q^{-1} + ... + b_{n-1} q^{-(n-1)} $$
$$ A(q^{-1}) = 1 + a_1 q^{-1} + ... + a_n q^{-n} $$
$$ T(q^{-1}) = 1 + t_1 q^{-1} + ... + t_m q^{-m} $$

The output is affected by the input signal $u(k)$ and the disturbance. Generally $T(q^{-1})=1$ is considered, but if it is a polynomial, it can be treated also as a filter. It may attenuate the component of prediction error caused by plant-model mismatch, which is particularly important at high frequencies. The high frequency disturbances are present mainly due to the presence of high frequency components in unmodeled dynamics and unmeasurable load disturbances. If there is no unmodeled dynamics, the effect of polynomial $T$ is rejection of disturbances, with no influence on reference tracking. In this case polynomial $T$ can be used to detune the response to unmeasurable high-frequency load disturbances, preventing excessive control action. On the other hand, polynomial $T$ is used as a design parameter that may influence robust stability (Camacho and Bordons, 2004, [19], Maciejowski, J.M., 2002, [43]).

If $T(q^{-1})$ is a polynomial, the prediction equations are valid for the filtered signals $y^F(q^{-1}) = y(k)/T(q^{-1})$ and $\Delta u^F(q^{-1}) = \Delta u(k)/T(q^{-1})$, respectively. The polynomial $T$ can be chosen as
\[ T(q^{-1}) = A(q^{-1})(1 - \beta q^{-1})^{\alpha} \] (Camacho and Bordons, 2004, [19]), where \( \beta \) is close to the dominant roots of polynomial \( A(q^{-1}) \).

In the prediction equation the free response is substituted by the filtered values. The control algorithm will then give the filtered value of the control increment, which has to be filtered with the inverse filter to get the actual control increment. This filtering procedure has a robustifying effect in case of plant-model mismatch. The \( T(q^{-1}) \) polynomial can be used to make predictive PI(D) control more robust, as well (Arousi et al. 2008, [12]).

**Example 2.6.1 Predictive PI control of a third-order process, the effect of the filtering polynomial in the prediction equation**

The system and the tuning parameters are the same as in Example 2.5.1. Including the \( T(q^{-1}) \) polynomial in the prediction equations has a smoothing effect to the control performance. An appropriate choice of the \( T(q^{-1}) \) filtering polynomial could decrease the effect of plant/model mismatch. Let us choose a first-order filter as \( T(q^{-1}) = (1 - t_1 q^{-1})/(1 - t_1) \). Fig. 2.6.1 demonstrates the effect of the \( T(q^{-1}) \) polynomial (here \( t_1 = -0.7 \)). Fig. 2.6.2 shows the performance with different filter parameter values in case of \( T_1 = 2 \). It is expected, that with further tuning of the \( T \) polynomial the effect of the mismatch could be decreased further.

![Figure 2.6.1. Controlled and manipulated variables, the system is of third-order, the controller is based on first-order model, with filtering polynomial \( T(q^{-1}) = (1 - 0.7 q^{-1})/0.3 \)](image1)

![Figure 2.6.2. Controlled and manipulated variables, the system is of third-order, the controller is based on first-order approximation, with different filtering polynomials (with \( t_1 = 0 \), \( t_1 = -0.5 \), \( t_1 = -0.7 \)).](image2)

Fig. 2.6.3 demonstrates, that increasing the prediction horizon works also against plant/model mismatch.
2.6.2 Enhancing robustness with Smith predictor filter

Another possibility to make predictive PI(D) control more robust is the application of a robustifying filter in a control structure similar to Smith predictor as shown in Fig. 2.6.4. This solution has been suggested in (Camacho and Bordons, 2004), and also applied in (Normey-Rico and Camacho, 2007) where some robustness analysis is also given. The choice of the robustifying filter influences the control performance. The filter generally can be chosen as a first-order low-pass filter with unity gain.

Figure 2.6.3. Controlled and manipulated variables, the system is of third-order with dead time 2, the controller is based on first-order model, different prediction ranges \((m=0, 5, 50)\), with \(T(q^{-1}) = (1-0.7q^{-1})/0.3\)

Figure 2.6.4. Predictive PI controller with robust Smith predictor scheme
Robust stability can be given by the following relationship analyzed in the frequency domain (Morari and Zafiriou, 1989, [40]):

\[ |l(j\omega)| < \frac{1}{n_M(j\omega)} \quad \forall \omega \]  \hspace{1cm} (2.6-1)

where

\[ l = \frac{\Delta G}{G_n} = \frac{G - G_n}{G_n} \]  \hspace{1cm} (2.6-2)

is the relative additive uncertainty of the plant transfer function, \( G_n \) is the nominal transfer function of the plant, \( G \) is its actual value and \( \Delta G \) is the deviation between the actual and the nominal transfer functions in the frequency domain. \( M_n \) is the nominal closed-loop transfer function between the output and the reference signals.

With controller \( C \) and filter \( F \) (2.6.1) can be given as

\[ |\Delta G(j\omega)| < \frac{1 + F(j\omega)C_{sm}(j\omega)G_n(j\omega)}{C_{sm}(j\omega)} \quad \forall \omega \]  \hspace{1cm} (2.6.3)

With some block-diagram algebra Fig. 2.6.4 can be given in the form shown in Fig. 2.6.5.

Here the Smith-predictor-like controller is

\[ C_{sm} = \frac{C}{1 + CG_n(1 - Fe^{-st_c})} \]  \hspace{1cm} (2.6.4)

If \( C \) denotes the transfer function of the predictive PI controller, it can be expressed e.g. by using the GPC-PI equivalence and then by GPC-RST transformation relations. It is seen, that the filter influences the robustness properties.
Example 2.6.2 Predictive PI control of a third-order process approximated by a first-order model with dead time using the robust Smith predictor scheme

The system is the same as in Example 2.5.1. The tuning of the PI predictive controller is detuned: the parameters are not divided by the number of the paths, $k_{cp} = 0.45$ and $k_{ci} = 0.1$. If the process is the same as the first-order model, the control performance is appropriate, but taking into consideration the plant/model mismatch with the third-order process there are significant oscillations in the output signal (Fig. 2.6.6). Applying the first-order Smith filter according to the structure shown in Fig. 2.6.4 the oscillations are decreased and the control performance becomes acceptable (Fig. 2.6.7).

The Smith filter was $F(q^{-1}) = \frac{0.2q^{-1}}{1 - 0.8q^{-1}}$.

![Figure 2.6.6. Detuned PI controller without and with plant/model mismatch](image1)

![Figure 2.6.7. Detuned PI controller without and with plant/model mismatch using robustifying Smith filter](image2)
Example 2.6.3 Comparing the performance of the predictive PI control algorithm and predictive PI control algorithm enhanced by Smith predictor filter in case of mismatch in the model and in the dead time.

The controller is designed considering the first-order approximating model. The plant is simulated according to the third-order description. In the simulation a positive unit step reference signal acts at time point 1, and -0.5 step disturbance is applied at time point 15 at the process input. Prediction of the reference signal is taken into account.

\[ G(s) = \frac{1}{(1 + 0.333s)^3} = \frac{1}{1 + 1.25s} e^{-0.26t} \]

The tuning parameters of the predictive PI controller are calculated according to Kuhn’s rules, their values are

\[ k_{cp} = 0.45/m \] and \[ k_{ci} = 0.1/m, \ m=10. \]

The Smith filter was \[ F(q^{-1}) = \frac{0.2q^{-1}}{1 - 0.8q^{-1}} \].

Fig. 2.6.8 shows the controlled and the manipulated signals of the predictive PI control algorithm and the predictive PI control algorithm enhanced by Smith predictor filter with small mismatch in the dead time whose value was considered 6, while the controller was designed supposing value of 3. It is seen that the predictive PI control algorithm without the filter gives a little bit faster behavior, and the control signal is higher than with the filtered case.
Fig. 2.6.9 gives the controlled and the manipulated signals with still bigger mismatch in the dead time, whose value was considered 18, while the controller was designed supposing value of 3. The performance is worse than before. It is seen that the predictive PI control algorithm with the filter is more robust.

Example 2.6.4 Comparing the performance of the predictive PI control algorithm and predictive PI control algorithm enhanced by Smith predictor filter in the frequency domain

The linear process is the same as in the previous example. The sampling time is \( \Delta T = 0.1 \). The plant can be approximated by a first-order system with dead time. The controller parameters calculated according to (2.4-19) for \( n_{ci} = 1 \) and \( n_{e2} = 10 \) and \( \lambda_u = 0.5 \) are: \( k_{ep} = 0.7652 \) and \( k_{ei} = 0.7793 \)

The Smith filter was chosen as \( F(q^{-1}) = \frac{0.1 q^{-1}}{1 - 0.9 q^{-1}} \).

The polynomial in the RST equivalent scheme are:

\[
S = 6.5518 - 5.5518 q^{-1} \\
R \Delta = 0.5866 - 0.2175 q^{-1} + 0.0485 q^{-2} + 0.0448 q^{-3} - 0.4624 q^{-4} \\
T = 1 \\
and \quad \Delta = 1 - q^{-1}
\]
Fig. 2.6.10 gives the Bode diagram of the deviation between the actual and the nominal transfer functions and the Bode diagram of the closed-loop transfer function between the output and the reference signals, equations (2.6.3 and 2.6.4). It is seen, that with the Smith filter the robustness condition is fulfilled. Fig. 2.6.11 shows the effect of Smith filter in the time domain.

Stability is a key issue in predictive control. In its original formulation GPC dose not guarantee stability.

With GPC-PID equivalence we can get the tuning parameters of predictive PID control. If GPC is stable, predictive PID is also stable. Stability can be analysed in case of applying a robustifying filter in the frequency domain. With the empirical PID tuning rules the control is stable.

Predictive PID algorithms could be extended also by antireset-windup methodologies. One way for handling saturation could be to putting the dynamics of the controller in the feedback of the saturation.
3. Decoupling in MIMO predictive control

Industrial processes are usually complex systems with many inputs and state variables. Certain inputs are used as manipulated variables, while others act as disturbances.

It would be desired, that one manipulated variable would control one specific output variable, while the other one the second output variable, etc., without cross effects. But often the change in one manipulated variable causes change in more output variables – this is the undesired cross-coupling effect. There are different decoupling methods to reduce interaction. A usual conventional decoupling is given in Appendix A. Model predictive control can be applied to multidimensional systems. MIMO predictive control can be handle decoupling.

An appropriate pairing of the input and output variables is also important.

Let us consider the control algorithm for TITO (Two Input-Two Output) processes.

3.1 The control algorithm

The cost function of a TITO predictive control is

\[
J = \lambda_{y1} \sum_{n_e=0}^{n_{e1}} [y_{r1}^*(k + d_1 + 1 + n_e) - \hat{y}_i(k + d_1 + 1 + n_e | k)]^2 \\
+ \lambda_{y2} \sum_{n_e=0}^{n_{e2}} [y_{r2}^*(k + d_2 + 1 + n_e) - \hat{y}_2(k + d_2 + 1 + n_e | k)]^2 \\
+ \lambda_{u1} \sum_{j=1}^{n_{u1}} \Delta u_1^2(k + j - 1) + \lambda_{u2} \sum_{j=1}^{n_{u2}} \Delta u_2^2(k + j - 1) \Rightarrow \text{MIN} \frac{\Delta u(k)}{}
\]  

(3-1)

with the notations:

d_i: discrete (physical) dead time relative to the sampling time of the i-th output,

\( y_{r_i}(k + d_i + 1 + n_e | k) \) reference signal of the i-th output \( n_{e_i} \) steps over the dead time \( d_i \),

\( y_{r_i}^*(k + d_i + 1 + n_e | k) \) modified reference trajectory of the i-th output \( n_{e_i} \) steps over the dead time \( d_i \),

\( \hat{y}_i(k + d_i + 1 + n_e | k) \) predicted i-th output signal \( n_{e_i} \) steps over the dead time \( d_i \).
The tuning parameters of the control algorithm are:

\[ n_{e2i} - n_{e1i} + 1 \] the length of the prediction horizon for the i-th output

\[ n_{ui} \] the length of the control horizon of the i-th input (the number of the supposed consecutive changes in the control signal),

\[ \lambda_{i1}, \lambda_{i2} \] weighting factors of the control error of the i-th output,

\[ \lambda_{u1}, \lambda_{u2} \] weighting factors of the control increments of the i-th input,

\[ \Delta \mathbf{u}(k) = [\Delta u_i(k | k), \Delta u_i(k + 1 | k), \ldots, \Delta u_i(k + n_{ui} - 1 | k)]^T \]

are the actual control increments, which have to be optimized.

Define the vectors for the i-th output in the future time domain

\[ k + d_i + n_{e1i} \leq j \leq k + d_i + n_{e2i} \]

a) reference signal sequence

\[ y_{re} = [y_{re}(k + d_i + 1 + n_{e1i} | k), \ldots, y_{re}(k + d_i + 1 + n_{e2i} | k)]^T \]

b) predicted output signal sequence

\[ \hat{y}_i = [\hat{y}_i(k + d_i + 1 + n_{e1i} | k), \ldots, \hat{y}_i(k + d_i + 1 + n_{e2i} | k)]^T \]

The predicted output signal can be separated to free and forced responses

\[ \hat{y}_i = \hat{y}_{i, \text{forced}} + \hat{y}_{i, \text{free}} \]

where the predicted forced i-th output can be expressed as

\[ \hat{y}_{i, \text{forced}} = \sum_{j=1}^{M} H_{ij} \Delta \mathbf{u}_j \]  

with the unknown control sequence in the control horizon

\[ \Delta \mathbf{u}_j = [\Delta u_j(k | k), \Delta u_j(k + 1 | k), \ldots, \Delta u_j(k + n_{uj} - 1 | k)]^T \]  

and a matrix of step response coefficients of the process model
\[
H_{ji} = \begin{bmatrix}
  h_{ji}(n_{e1i}) & h_{ji}(n_{e1i} - 1) & \cdots & h_{ji}(n_{e1i} - n_{ui} + 1) \\
  h_{ji}(n_{e2i} - 1) & h_{ji}(n_{e2i} - n_{ui}) & \cdots & h_{ji}(n_{e2i} - n_{ui} + 1) \\
  \vdots & \vdots & \ddots & \vdots \\
  \vdots & \vdots & & \vdots \\
  h_{ji}(n_{e2i}) & h_{ji}(n_{e2i} - 1) & \cdots & h_{ji}(n_{e2i} - n_{ui} + 1)
\end{bmatrix}
\]  

(3-6)

where

\[ h_{ji}(k) = 0 \quad \text{if} \quad k < 0; \quad \forall i, \forall j \]

For a TITO process define the following vectors of signal sequences in the prediction time domain \( k + d_i + n_{ei} \leq j \leq k + d_i + n_{ei} \):

- \( y_r = [y_{r1}^T, y_{r2}^T]^T \) : modified reference signal,
- \( \hat{y} = [\hat{y}_1^T, \hat{y}_2^T]^T \) : predicted outputs,
- \( \hat{y}_{\text{forced}} = [\hat{y}_{1\text{forced}}^T, \hat{y}_{2\text{forced}}^T]^T \) : predicted forced outputs,
- \( \hat{y}_{\text{free}} = [\hat{y}_{1\text{free}}^T, \hat{y}_{2\text{free}}^T]^T \) : free responses

and

\[ \Delta u = [\Delta u_1^T, \Delta u_2^T]^T \quad \text{vector of all control input signals in the control horizon} \]

\[ k \leq j \leq k + n_{ui} - 1 \]

Now the predicted forced i-th output can be calculated as

\[ \hat{y}_{\text{forced}} = \sum_{j=1}^{2} H_{ji} \Delta u_j = H_{i} \Delta u \quad \text{where} \quad H_{i} = [H_{i1}, H_{i2}]^T \]

The vector of the predicted outputs is the sum of the predicted forced and free responses:

\[ \hat{y} = \hat{y}_{\text{forced}} + \hat{y}_{\text{free}}, \quad \text{with} \quad \hat{y}_{\text{forced}} = \begin{bmatrix} H_{i1} \\ H_{i2} \end{bmatrix} \Delta u = H_{i} \Delta u \]

The cost function becomes

\[ J = (y_r^* - \hat{y})^T \Lambda_y (y_r^* - \hat{y}) + \Delta u^T \Lambda_u \Delta u \Rightarrow \text{MIN} \]

\[ \Delta u(k) \]

with the weighting matrix of the control errors \( \Lambda_y = \text{diag}(\Lambda_{y1}, \Lambda_{y2}) \)

or in detailed form

\[ J = (y_r^* - H \Delta u - \hat{y}_{\text{free}})^T \Lambda_y (y_r^* - H \Delta u - \hat{y}_{\text{free}}) + \Delta u^T \Lambda_u \Delta u \Rightarrow \text{MIN} \]

\[ \Delta u(k) \]  

(3-7)
Unconstrained minimization of the cost function according to the whole sequence of input increments in the control time domain leads to

\[
\frac{dJ(\Delta u)}{d\Delta u} = -H^T \left[ \Lambda_y^T + \Lambda_y \right] y_r^* - H\Delta u - \hat{y}_{free} + \left[ \Lambda_u^T + \Lambda_u \right] \Delta u = 0
\]

which results in

\[
\Delta u = \left[ H^T \left( \Lambda_y^T + \Lambda_y \right) H + \left( \Lambda_u^T + \Lambda_u \right) \right]^{-1} H^T \left( \Lambda_y y_r^* - \hat{y}_{free} \right)
\]

(3-8)

All weighting matrices are diagonal

\[
\Lambda_{yi} = \lambda_i I \quad \text{and} \quad \Lambda_{ui} = \lambda_i I
\]

where \( I \) is the identity matrix.

As with diagonal matrices the transposed matrix is equal to the non-transposed one

\[
\Lambda_y^T = \Lambda_y \quad \text{and} \quad \Lambda_u^T = \Lambda_u
\]

then

\[
\Delta u = \left[ H^T \Lambda_y H + \Lambda_u \right]^{-1} H^T \Lambda_y \left( y_r^* - \hat{y}_{free} \right)
\]

(3-9)

According to the receding horizon technique only the actual control signals will be used and the computation is repeated in the next control step. Denote the actual control increments by

\[
\Delta u_{\text{actual}}(k) = [\Delta u_{1}(k), \Delta u_{2}(k)]^T,
\]

which can be expressed as

\[
\Delta u_{\text{actual}}(k) = \left[ 1, 0, \ldots, 0, 1, 0, \ldots, 0 \right]^T \Delta u(k)
\]

(3-10)

where the number of zeros is \( n_u1-1 \) and \( n_u2-1 \) respectively.

3.1.1 Control without and with ideal decoupling

In order to illustrate the problem of coupling a TITO process model (Fig. 3.1) was considered.
Fig. 3.1. TITO process model

$y_1$ and $y_2$ are the controlled variables (CV) and $u_1, u_2$ are the manipulated input variables (MV).

The sub-models are aperiodic first-order processes with different static gain $K_{pi}$, time constants $T_{ij}$, and dead time $T_{di}$. All processes have some ($n$) equal time constants:

P11: $K_{p11}=1.5$, $T_{11}=1.0$ min, $n_{11}=2$, $T_{d11}=0.1$ min
P12: $K_{p12}=0.5$, $T_{12}=0.5$ min, $n_{12}=4$, $T_{d12}=0.5$ min
P21: $K_{p21}=0.75$, $T_{21}=0.5$ min, $n_{21}=3$, $T_{d21}=0.8$ min
P22: $K_{p22}=1.0$, $T_{22}=2.0$ min, $n_{22}=1$, $T_{d22}=0.2$ min

Fig. 3.2 shows the unit step responses of the sub-models.

Fig. 3.2. Sub-models of the TITO process (top left:$P_{11}$, top right:$P_{12}$, bottom left:$P_{21}$, bottom right:$P_{22}$.
Fig. 3.3 shows the TITO predictive control without decoupling. The sampling time was $\Delta T=0.1 \text{ min}$ and the controller parameters are:

- start of control error horizons: $n_{c11}=n_{c12}=0$,
- end of control error horizons: $n_{c21}=n_{c22}=90$,
- length of control horizons: $n_{e11}=n_{e12}=30$,
- weighting factors of the control errors $\lambda_1=\lambda_2=1$,
- weighting factors of the control increments $\lambda_{u1}=\lambda_{u2}=0.5$.

The control scenario was:

- at $t=1 \text{ min}$ stepwise increase of the reference signal of CV1 by 1,
- at $t=10 \text{ min}$ stepwise increase of the reference signal of CV2.

In the further examples always the same control scenario is simulated with the nominal controller parameters given above.

![Graph showing the control scenario](image)

Fig. 3.3. TITO control without decoupling: (top: CV1, bottom: CV2)

The control of the set-value changes is fast with an overshoot of about 10%. There are changes of about 10 to 15% (related to the set-value changes) in the controlled variables whose set-value was kept constant.
A perfect decoupling can be achieved if the increments in the manipulated variables are not penalized, i.e. $\lambda_{u_1} = \lambda_{u_2} = 0$. Fig. 3.4 shows the control with these controller parameters. The other tuning parameters are the same as in the case of Fig. 3.3. A change in the set-value does not cause any change in the other control variable at the cost of very drastic changes in the manipulated variables. Thus this decoupling method is not practical.

![Fig. 3.4. TITO control with perfect decoupling without penalizing control increments (top: CV1, bottom: CV2)](image)

MIMO predictive control realizes perfect decoupling if the control increments are not weighted in the cost function. As this case can not be applied practically because of the drastic input changes which cause saturation, additional methods are required to improve decoupling properties of MIMO predictive control.

(Maurath et al. (1986), [39]) considered the predictive control of a TITO (Two Input, Two Output) process for set-point change only in one controlled variable while the other set-point is kept constant. The control aim is a relatively fast control in case of
set-point change while minimizing the control error of the other variable, i.e. minimizing the coupling effect. Three methods are recommended:

1. **Constrained control:**
   The controlled variable whose set-point was not changed is limited within a small range around its set-point, i.e. the control error of this variable is limited. The disadvantage is that the multivariable control algorithm has to be applied under constraints.

2. **Using a decelerated set-point change:**
   A slower change of the reference signal leads to less coupling effect. The stepwise change of the set-value can be replaced by a modified reference signal equal to the controlled variable of the decoupled case.

3. **Different weighting factors of the control errors:**
   The weighting factor of the control error of the variable whose set-point was not changed should be increased against that of the controlled variable whose set-point was changed. The advantage of the method is that the control algorithm can be designed and performed without constraints. It is disadvantageous, however, that the time point of the weighting factor change has to be synchronized to the change in the reference signal, thus this method requires a simple “signal detector”.

Among the three methods the change of the weighting factors can be realized the most easily. The question is whether the timing of the change of the weighting factor can be realized practically by synchronization to the set-point changes using a “set-point change detector”. An easy solution to this problem is setting the weighting factors as functions of the control error. With a stepwise change of the reference signal of a controlled variable the control error increases faster than the control error of the other variable whose set-point was kept constant. Consequently, if the weighting factor is set inverse proportional to the control error for both controlled variables, then after a stepwise change of a reference signal the weighting factor of the output whose set-point was not changed will be higher than the weighting factor of the output whose set-point was changed.

The new tuning procedure of the weighting factors works automatically, thus no extra synchronisation is necessary.

### 3.2 Decoupling by decelerating the reference signal change

It is expected that a slower change of the reference signal leads to decreased coupling effect. This is illustrated by Fig. 3.5 where the stepwise change of both set-values was filtered by a first-order filter with the time constant of $T_{r1}=1.5\text{min}$ and $T_{r2}=1\text{min}$. The
filter parameters were selected in such a way that the filtered reference signal approximates the controlled signal without decoupling (Fig. 3.2). As it is seen, any set-point change practically does not disturb the other controlled variable whose value should be unchanged.

In the next simulation the stepwise change of the set-value is replaced by a modified reference signal equal to the controlled variable of the decoupled case (Fig. 3.5) as recommended by (Maurath et al. (1986), [39]). (The remaining small coupling effects - before and after the reference step - the dead time and the very small oscillations after the control step have been removed from the old controlled signal of Fig. 3.5 for the new reference signal.)

Fig. 3.5. TITO control with reference signal filter (top: CV1, bottom: CV2; dashed: filtered reference signal)

Fig. 3.6 demonstrates the conditioning of the reference signal. It is forced to 0 before the set-point step and is forced to 1 after the settling time. Fig. 3.7 shows the control using the new reference signal both for small $\lambda_{y1}=\lambda_{y2}=1$ and a higher $\lambda_{y1}=\lambda_{y2}=100$
weighting factors of the control errors. The weighting factors have to be raised because now the reference signal is not a step but a slower signal.

Instead of storing the controlled variable as a modified reference signal the actual reference signal can be filtered in such a way that the filtered reference signal would approximate the controlled signal in the coupled case. This can be achieved if a filter is identified between the stepwise reference signal change and the controlled variable in the coupled case (Fig. 3.5) using a conventional LS-algorithm. A filter with order 3 was required to have a sufficient fitting between the reference signal change and the corresponding controlled variable. The estimated filters are:

$$y_{rl}^{f}(k) = \frac{0.0082 + 0.001055q^{-1} - 0.01332q^{-2}}{1 - 2.667q^{-1} + 2.423q^{-2} - 0.7405q^{-3}} y_{rl}(k)$$

Fig. 3.6. Conditioning of the control signal for the TITO control with the reference signal equal to the controlled signal in the decoupled case (top: CV1, bottom: CV2; solid: controlled signal, dashed: modified reference trajectory)
\[ y_{r2}^F(k) = \frac{2.308 \cdot 10^{-6} + 0.05613q^{-1} - 0.05421q^{-2}}{1 - 2.617q^{-1} + 2.304q^{-2} - 0.6843q^{-3}} y_{r2}(k) \]

This control is seen in Fig. 3.8. There is practically no difference whether the controlled signal from the coupled case or the filtered one was applied as a modified reference trajectory. However, the second case can be applied much easier because only some filter parameters and not the whole reference trajectory have to be stored.

### 3.3 Weighting factor adjusting at reference signal change

As mentioned already in the introduction increasing of the control error weighting factor of the variable whose set-point was kept constant reduces the control error in this variable. Fig. 3.10 illustrates this case. The controlled variables are as in the case
of the coupled control (Fig. 3.3). The weighting factors of both control errors were changed from $\lambda_1=\lambda_2=1$ to $\lambda_1=\lambda_2=100$ for that variable whose set-point was not changed in the moment of the set-point change. The duration of the change was 5 min which is a bit (about 2 min) longer than the settling time of the controlled process. The plots show that the two processes are completely decoupled.

The critical point of this method is the detection of the set-point change. In case of predictive control there are applications where the reference signal trajectories are given, so the changes in the reference signals are known in advance and stored. If the changes in the reference signals are not known a priori, there are several methods for detecting signal changes. However, we recommend in the next section an alternative method, which does not require any signal change detector or observer.

Fig. 3.8. TITO control with the optimally filtered reference signal (top: CV1, bottom: CV2; dashed: modified reference trajectory)

We increased the weighting factor in the case of Fig. 3.10 manually and kept its value constant at least for the duration of the settling time of the closed loop controlled
process. That means the weighting factor is decreased to its old value (before the set-point change) abruptly.

(Bego et al. (2000), [18]) applied a similar technique and decreased the weighting factor exponentially to its old value before the set-point change. They showed the effect of the choice of the starting value and the time constant of the exponential decrease, however, the parameters were tuned based on repeated simulations instead of giving tuning rules. Fig. 3.9 shows two alternative procedures: constant or decreasing weighting factor during the settling time after the set-point change in the other controlled variable. The exponential decrease ensures a smoother change of the weighting factor and should, therefore, be preferred against an abrupt change.

**Fig. 3.9. Weighting factor modification strategies**

### 3.4 Control error dependent weighting factor adjusting

The synchronisation at the set-value change can be performed automatically if the weighting factors are decentralized functions of the control errors. With a stepwise change of the reference signal of a controlled variable the control error increases faster than the control error of the other variable whose set-point was kept constant. Consequently, if the weighting factor is set inverse proportional to the control error for both controlled variables then after a stepwise change of a reference signal the weighting factor of the output whose set-point was not changed will be higher than the weighting factor of the output whose set-point was changed.

After some simulation trials the following dependence of the control error weighting factors and the control error seemed to be optimal:

\[ \hat{\lambda}_{yi} = \frac{\hat{\lambda}_{yi,\text{max}}}{(1 + |e_i(k)| \cdot \hat{\lambda}_{yi,\text{damp}})} \]  

(3-11)
Fig. 3.10. TITO control with the weighting factor change at set-point steps (top: CV1, MV1, $\lambda_{y1}$, bottom: CV2, MV2, $\lambda_{y2}$)
with $\lambda_{y1,\text{max}}=10$, $\lambda_{y2,\text{max}}=20$, $\lambda_{y1,\text{damp}}=100$ and $\lambda_{y2,\text{damp}}=100$.

The control is slightly slower than with the manual adaptation of the control error weighting (Fig. 3.10) but the control is still fast and the decoupling is very good (as before). The automatic adaptation of the control error weighting shows also a decrease of the other controlled signal whose set-value was kept constant which is an indicator of the remaining coupling effects. But these effects are very small and thus also the decrease of the control error weighting is small. From Fig. 3.12 one can see that the weighting factors of those controlled variable whose set-value was stepwise changed were temporarily significantly reduced. It has to be mentioned that the change of the $\lambda_{y1}$ and $\lambda_{y2}$ weighting factors approximates an exponential course (similarly to Fig. 3.9). (Remember that an exponential change of the weighting factors was preferred compared to a stepwise change.)

In Appendix A the usual conventional decoupling controller is designed for considered example (Figs. 3.1 and 3.2). The simulation shows that the derived decoupler in predictive control context provide better decoupling results.

### 3.5 Application to distillation column model

The algorithms were applied for control a model of a distillation pilot plant. Distillation is a thermal separation method of chemicals with different boiling points. In continuous binary distillation a feed is separated into two products. Vapour at the top of the column is cooled and condensed to liquid in the condenser. Top product is partially returned to the column as reflux flow. Liquid runs down the column (vapour up). The liquid reaching the bottom of the column is partially vaporized in a reboiler and is sent back up to the column. Part of the liquid in the column bottom is taken away as a bottom product.

Fig. 3.11 shows a typical column used for separating chemical petrol:

feed: chemical petrol from the desulfurisation, top product: light petrol, bottom product: heavy petrol.
Fig. 3.11. Piping and instrumentation scheme of the distillation column
Fig. 3.12. TITO control with error-dependent weighting factors (top: CV1, MV1, $\lambda_y^1$, bottom: CV2, MV2, $\lambda_y^2$)
Fig. 3.13 shows the process model between the manipulated and the controlled variables

- manipulated variables:
  - reflux flow,
  - heating power (duty),

- controlled variables:
  - top temperature,
  - bottom temperature.

All times are given in the transfer functions in minutes.

In the following simulation plots the following ranges of the variables were scaled to 0 to 100 %: (pressure compensated) top temperature (TOPPCT) 50 to 64 °C, bottom temperature (BOTPCT) 148 to 162 °C, reflux flow -200 to 520 tons day and heating power (duty) 3 to 10 MW. At time t=10 min the set point of the top temperature was decreased by 3°C and at time t=200 min the set point of the bottom temperature was increased by 2°C.

The multivariable control was simulated without constraints and with sampling time \( \Delta T=1 \) min, the prediction horizon was given \( n_{e1}=n_{e2}=0 \) and \( n_{e1}=n_{e2}=100 \), the control horizon was \( n_u=25 \), the weighting of the control error and of the control increments for the first and second manipulated variables \( \lambda_{y1}=\lambda_{y2}=1 \), \( \lambda_{u1}=0.0012 \) and \( \lambda_{u2}=30.8269 \), respectively. The simulation results are shown in Fig. 3.2.14.
Fig. 3.14 Control of the TITO distillation column model with constant control error weighting factors (CV1: TOPPCT, CV2: BOTPCT, MV1: Reflux and MV2: Duty)

It is seen that the set-point change disturbs the other variable, which should be remained constant. This coupling effect was suppressed by the method of changing the control error weighting factor as a function of the control error.

A good decoupling could be achieved with the weighting factors set according to (3.11) with \( \lambda_{y1,\max} = 1 \), \( \lambda_{y2,\max} = 1 \), \( \lambda_{y1,\text{damp}} = 25 \) and \( \lambda_{y2,\text{damp}} = 25 \). Fig. 3.15 shows the practically decoupled control.
Fig. 3.15 Control of the TITO distillation column model with control error dependent control error weighting factors (top: CV1: TOPPCT, CV2: BOTPCT, MV1: Reflux and MV2: Duty; bottom: $\lambda_{CV1}$, $\lambda_{CV2}$)
4. **Nonlinear predictive control based on the Volterra model**

Predictive control algorithms determine a series of the control signal minimising a cost function which calculates the difference between the reference signal and the output of the system model in a given future horizon. In the last decades predictive control algorithms have gained wide industrial acceptance.

In industrial practice the plants generally contain nonlinearities. Linear control algorithms provide good performance only in a small range around a working point. If the reference signal varies in a big range, control algorithms considering the nonlinear characteristics of the plant would ensure better control behaviour than the linear control algorithms.

Nowadays there is an intensive research in the area of nonlinear predictive control.

**4.1 Suboptimal nonlinear Volterra model based predictive control algorithms applied to the model of a pilot plant**

There are different ways of modelling nonlinear systems (Haber, (1995, [27]), Haber and Keviczky (1999,[26]).

Block-oriented description supposes seriesly connected static nonlinear and dynamic linear parts. In the Hammerstein model the static nonlinear characteristics precedes the dynamic linear part, while in the Wiener model the input signal acts first on the linear dynamics, and the linear output provides the input for the static nonlinear characteristics. A combination of the two models is the Wiener-Hammerstein model.

All these models can be approximated by the Volterra model, which besides the linear part considers the effect of cross-product terms of the shifted input signal in the calculation of the output signal.

The Volterra model is applied for predictive control of a nonlinear process. In this case predictive incremental transformation of the model is necessary. Minimisation of a quadratic cost function without constraints applying a quadratic Volterra model leads to an analytic solution. The control signal is calculated as a solution of a third degree algebraic equation.

The control algorithms are applied to the model of a ball and beam system.
4.1.1 The parametric Volterra model and its predictive form

The second-order parametric Volterra model is given by the following equation (Haber and Keviczky (1999, [25, 26]):

\[ A(q^{-1})y(k) = c_0 + B_1(q^{-1})u(k - d) + B_2(q_1^{-1}, q_2^{-1})u^2(k - d) \]  

(4-1)

where \( k \) denotes the current time point, \( d \) is the discrete dead time, \( A_1, B_1, \) and \( B_2 \) are polynomials of the backward shift operator \( q^{-1} \) of degrees \( n_{a1}, n_{b1} \) and \( n_{b2} \), respectively. In detailed form:

\[ y(k) = -\sum_{j=1}^{n_{a1}} a_j y(k - j) + c_0 + \sum_{i=0}^{n_{a2}} b_{1i} u(k - d - i) + \sum_{i=0}^{n_{b2}} b_{2ij} u(k - d - i) u(k - d - j) \]  

(4-2)

Shift the model equation \( d + n_e + 1 \) steps ahead

\[ A(q^{-1})y(k + d + 1 + n_e) = c_0 + B_1(q^{-1})u(k + 1 + n_e) + B_2(q_1^{-1}, q_2^{-1})u^2(k + 1 + n_e) \]  

(4-3)

This form is non-predictive. To apply the model for predictive control predictive transformation is required.

An incremental form where the predicted output depends on the input signal increment is advantageous for the derivation of the control algorithm, as the cost function contains the control increment and not the control signal itself. The current and the future control signals are expressed with the current and future control increments and with the old input signal \( u(k - 1) \) as follows

\[ u(k + i) = u(k - 1) + \sum_{j=0}^{i} \Delta u(k + j) \quad i = 0, 1, \ldots, n_e \]  

(4-4)

Let us define \( \Delta u^*(k) \) as follows:

\[ \Delta u^*(k) = u(k) - u(k - 1) \quad \text{if } k \geq 0, \]  

(4-5)

\[ \Delta u^*(k) = u(k) \quad \text{if } k < 0. \]

With the incremental form the predictive model can be expressed as (Haber (1992, 1995, [27], [28]))

\[ \hat{y}(k + d + 1 + n_e | k) = c_{01}^{(d+1+n_e-1)} + \delta_{i_1}^{(d+1+n_e-1)}(q^{-1})y(k) + \gamma_{1}^{(d+1+n_e-1)}(q^{-1})\Delta u^*(k + 1 + n_e) \]

\[ + \gamma_{2}^{(d+1+n_e-1)}(q_1^{-1}, q_2^{-1})\Delta u^2(k + 1 + n_e) \]  

(4-6)
where

\[
\delta^{(d_1+n_k)} (q^{-1}) = \alpha^{(d_1+n_k)} (q^{-1})
\]

\[
\gamma_{li}^{(d_1+n_k)} = \sum_{v=1}^{\min(i,n_k+2)} \beta_{lv}^{(d_1+n_k)}; \quad i = 1, \ldots, n_e + 2;
\]

\[
\gamma_{2li}^{(d_1+n_k)} = \sum_{v=1}^{\min(i,n_k+2)} \sum_{v'=v}^{\min(i,n_k+2)} \beta_{2lv}^{(d_1+n_k)}; \quad i = 1, \ldots, n_e + 2;
\]

\[
\gamma_{2ij}^{(d_1+n_k)} = \sum_{v=1}^{\min(i,n_k+2)} \left( \sum_{v'=v}^{\min(i,n_k+2)} \beta_{2v}^{(d_1+n_k)} + \sum_{v''=v''}^{\min(j,n_k+2)} \beta_{2v''}^{(d_1+n_k)} \right); \quad i = 1, \ldots, n_e + 2; \quad j = i + 1, i + 2, \ldots, n_e + 2;
\]

\[
\gamma_{2ij}^{(d_1+n_k)} = 0; \quad j < i
\]

\[
\gamma_{li}^{(d_1+n_k)} = \beta_{li}^{(d_1+n_k)} \quad \text{and} \quad \gamma_{2ij}^{(d_1+n_k)} = \beta_{2ij}^{(d_1+n_k)}, \quad j \geq i; \quad \text{if} \quad i \geq n_e + 3
\]  

(4-7)

The incremental form can be written also in the following form:

\[
\hat{y}(k + d + 1 + n_\rho | k) = p_0^{(d+1+n_\rho)} + p_1^{(d+1+n_\rho)} (q^{-1}) \Delta u(k + 1 + n_e) + p_2^{(d+1+n_\rho)} (q_1^{-1}, q_2^{-1}) \Delta u^2 (k + 1 + n_e)
\]  

(4-8)

The degree of the polynomials is \( \deg(P_0) = n_e + 1 \) and \( \deg(P_2) = [n_e + 1, n_e + 1] \). The coefficients are calculated as follows:

\[
p_0^{(d+1+n_\rho)} = c_0^p + \delta^{(d+1+n_\rho)} (q^{-1}) y(k)
\]

\[
+ \sum_{i=n_e+2}^{\max(n_e+2,n_k+2)} \gamma_{li}^{(d+1+n_\rho)} u(k + n_e + 1 - i)
\]

\[
+ \sum_{i=n_e+2}^{\max(n_e+2,n_k+2)} \sum_{j=0}^{\max(n_e+2,n_k+2)} \gamma_{2ij}^{(d+1+n_\rho)} u(k + n_e + 1 - j) u(k + n_e + 1 - i)
\]

\[
p_1^{(d+1+n_\rho)} = \gamma_{li}^{(d+1+n_\rho)} + \sum_{i=n_e+2}^{\max(n_e+2,n_k+2)} \gamma_{li}^{(d+1+n_\rho)} \Delta u(k + n_e + 1 - j); \quad i = 0, 1, \ldots, n_e + 1
\]

\[
p_2^{(d+1+n_\rho)} = \gamma_{2ij}^{(d+1+n_\rho)} \quad i = 1, \ldots, n_e + 1; \quad j = 1, 2, \ldots, n_e + 1.
\]  

(4-9)

In (4-8) \( p_0^{(d+1+n_\rho)} \) contains the effects of the past inputs and outputs on the future output signal, while \( p_1^{(d+1+n_\rho)} (q^{-1}) \Delta u(k + 1 + n_e) \) gives the linear and
$P_{z}^{(d+n_e)}(q_{1},q_{2}) \Delta u^2 (k + 1 + n_e)$ the quadratic terms depending on the actual and future control increments in the control horizon. The term $p_{0}^{(d+n_e)}$ does not depend on the actual and future control increments, which are unknown in the actual time point $k$.

(4-8) can be described also in the following form:

$$\hat{y}(k + d + 1 + n_e \mid k) = \hat{y}_0(k + d + 1 + n_e \mid k) + \hat{y}_1(k + d + 1 + n_e \mid k) + \hat{y}_2(k + d + 1 + n_e \mid k)$$

(4-10)

with

$$\hat{y}_0(k + d + 1 + n_e \mid k) = p_{0}^{(d+n_e)}$$

(4-11a)

$$\hat{y}_1(k + d + 1 + n_e \mid k) = [p_{1,0+n_e}^{(d+n_e)}, p_{1,1+n_e}^{(d+n_e)}, \ldots, p_{1,n_e+n_e}^{(d+n_e)}] \begin{bmatrix} \Delta u(k) \\ \Delta u(k + 1) \\ \vdots \\ \Delta u(k + n_e) \end{bmatrix}$$

(4-11b)

$$\hat{y}_2(k + d + 1 + n_e \mid k) = [\Delta u(k), \Delta u(k + 1), \ldots, \Delta u(k + n_e)].$$

(4-11c)

4.1.2 The cost function

A quadratic cost function $J$ is minimized, which considers the quadratic deviation of the reference signal $y_r$ and the output signal $\hat{y}$ predicted in a future horizon and punishes also the squares of the control increments

$$J = (y_r - \hat{y})^T \Lambda_y (y_r - \hat{y}) + \Delta u^T_f \Lambda_u \Delta u_f \Rightarrow \min_{\Delta u_f}$$

(4-12a)

or

$$J = \sum_{n_e=n_1}^{n_2} \Lambda_{y_{n_e}} [y_r(k + d + 1 + n_e) - \hat{y}(k + d + 1 + n_e \mid k)]^2 + \sum_{j=1}^{n_2} \Lambda_{u_j} \Delta u^2(k + j - 1) \Rightarrow \min_{\Delta u_f}$$

(4-12b)
The tuning parameters of the control algorithm are:

- $n_{e2} - n_{e1}$: the prediction horizon domain,

- $n_u$: the control horizon (the number of the supposed consecutive changes in the control signal),

- $\lambda_{e1}, \ldots, \lambda_{e2}$: weighting factors of the control error, usually assumed equal to 1 ($\lambda_y = 1$),

- $\lambda_{u1}, \ldots, \lambda_{un}$: weighting factors of the control increments, usually assumed to be equal (and denoted then by $\lambda_u$).

### 4.1.3 Minimization strategies for the cost function

In the knowledge of the future (predicted) reference signal the control error can be predicted and the cost function (4-12) can be minimized. With the calculated control inputs the output (controlled) signal achieves its reference value earlier than with no prediction.

Depending on the control strategy and whether there are constraints or not, different control algorithms can be derived.

- **Multidimensional optimization**
  
  If the number of allowed changes in the control horizon is greater than one, multidimensional optimization is required to compute the optimal control signal.

- **One-dimensional optimization**
  
  Using special manipulated variable parameterization strategies during the control horizon, the optimization problem can be reduced to one-dimensional (scalar) case. This is the case e.g. if only one change in the manipulated signal is allowed and in the subsequent points the change is zero, or all increments are assumed to be equal.

The suboptimal case with constant control signal or constant control signal increments during the manipulated variable horizon is considered.

As the predicted model output is a quadratic function of the input signal, and the cost function is a quadratic function of the control error and the control increments, the cost function is a 4th degree function of the control increments. In unconstrained case the minimization leads to the solution of a 3rd degree equation in the control increments

$$k_0 + k_1 \Delta u(k) + k_2 \Delta u^2(k) + k_3 \Delta u^3(k) = 0.$$  \hspace{1cm} (4-13)
The coefficients in the equation depend on the assumption related to the change of the control increments during the control horizon, see Haber et. al. (1999), [30].

a. Long-range optimal control with the assumption of constant control signal in the control horizon

The derivation of the cost function according to the control increment with the prediction horizon equation (4-8) and with the initial and final values of the prediction \( n_{x1}, n_{x2} \) supposing constant control signal during the prediction horizon leads to the following coefficients of the cubic equation (Haber et al., 1998; 1999a; 1999b)

\[
\begin{align*}
k_0 &= \sum_{n_x=n_{x1}}^{n_{x2}} \left[ p_0^{(n_x)} - y_r (k + d + 1 + n_x) \right] \cdot p_0^{(d+1+n_x)} \\
k_1 &= 2 \sum_{n_x=n_{x1}}^{n_{x2}} \left[ p_0^{(n_x)} - y_r (k + d + 1 + n_x) \right] \cdot p_1^{(d+1+n_x)} \cdot p_2^{(d+1+n_x)} + \sum_{n_x=n_{x1}}^{n_{x2}} \left[ p_1^{(d+1+n_x)} \right]^2 + \lambda_{n0} \\
k_2 &= 3 \sum_{n_x=n_{x1}}^{n_{x2}} \left[ p_1^{(d+1+n_x)} \cdot p_2^{(d+1+n_x)} \right] \\
k_3 &= 2 \sum_{n_x=n_{x1}}^{n_{x2}} \left[ p_2^{(d+1+n_x)} \right]^2
\end{align*}
\]

(4-14)

b. Long-range optimal control with the assumption of constant control increments in the control horizon

The derivation of the cost function according to the control increment with the prediction equation (4-8) supposing equal control increments during the prediction horizon leads to the following coefficients of the cubic equation (Haber, Bars and Lengyel, 1998; 1999a; 1999b)

\[
\begin{align*}
k_0 &= \sum_{n_x=n_{x1}}^{n_{x2}} \left[ p_0^{(n_x)} - y_r (k + d + 1 + n_x) \right] \cdot \left[ \sum_{i=1}^{n_x+1} p_i^{(d+1+n_x)} \right] \\
k_1 &= 2 \sum_{n_x=n_{x1}}^{n_{x2}} \left[ p_0^{(n_x)} - y_r (k + d + 1 + n_x) \right] \cdot \left[ \sum_{i=0}^{n_x+1} \sum_{j=0}^{n_x+1} p_{2ij}^{(d+1+n_x)} \right] + \sum_{n_x=n_{x1}}^{n_{x2}} \left[ \sum_{i=1}^{n_x+1} p_i^{(d+1+n_x)} \right]^2 + \sum_{j=0}^{n_x} \lambda_{n_j} \\
k_2 &= \sum_{n_x=n_{x1}}^{n_{x2}} \left[ p_2^{(d+1+n_x)} \right]^2
\end{align*}
\]

(4-15)
\[ k_2 = 3 \sum_{n_j=n_i}^{n_j+1} \left( \sum_{i=1}^{n_i+1} P_{li}^{(d+i+n_j)} \right) \left( \sum_{i=1}^{n_i+1} \sum_{j=1}^{n_j+1} P_{2j}^{(d+i+n_j)} \right) \]

\[ k_3 = 2 \sum_{n_j=n_i}^{n_j+1} \left( \sum_{i=1}^{n_i+1} \sum_{j=1}^{n_j+1} P_{2ij}^{(d+i+n_j)} \right)^2. \]

The control increment is to be chosen from among the solutions of the cubic equation. From among three real roots that one is chosen where the value of the cost function is the less. The real root is chosen if conjugate complex pairs also exist. The control signal should be inside the control limits.

4.1.4 The model of the ball and beam pilot plant

The algorithms above were applied to the ball and beam system BW500 in the laboratory of the Department of Automation and Applied Informatics. The nonlinear plant is represented by a ball lying on a beam with an adjustable angle. The ball can roll upon the beam within a distance of 1m. By a toothed belt, a toothed wheel and a clutch mounted on the shaft of a DC motor the beam can be driven such that the ball is stabilized at a pre assigned position. The aim is to control the position of the ball on the beam. The picture of the system is given in Fig. 4.1.1. The stabilization of the ball is accomplished by a digital controller. Based on measurements, the controller generates a suitable signal, which controls the DC-motor by an electronic servo amplifier. The measurements are the angle of the beam obtained by an incremental encoder and the ball position obtained by a CCD-camera. The system is ready to operate after connecting to the electronic actuator.

Fig. 4.1.1. Ball and beam system
Mathematical model of the ball and beam system is given in Appendix B. The system model and its real-time cascade PID control has been analysed and realised in the diploma project of Tziortzis, (2007, [33]). Here the model and the control of the beam velocity and angle will be considered. The simplified model of the beam is given in Fig. 4.1.2.

Due to friction the nonlinearity is of dead-zone type (Fig. 4.1.3).

The aim of the current initial investigation is to identify the model in parametric Volterra form and to apply the suboptimal Volterra control algorithm for position control of the model.

### 4.1.5 Identification results of the model

Off-line identification has been executed with the LS method. The system model with one and with two integrators has been identified considering different Volterra model degrees (memory). For the experiments the gain of the system was $K_p = 1$, and the dead zone $U_{c-} = 0.5$. The step-like input signal is shown in Fig. 4.1.4.
Identification results of the system with two integrators

The identification has been executed for different Volterra model degrees. As this is a parametric description, small order (2, 3, 4) could be enough, as the output signals contain also the effects of the past inputs.

The identified parameters with degree 1 of the polynomials are as follows (the identified polynomials are denoted by index 0):

\[ A_0 = 1 - 1.0103q^{-1} \]
\[ c_{00} = -3.1478 \]
\[ B_{10} = 1.6677q^{-1} \]
\[ B_{20} = 5.582q_1^{-1}q_2^{-1} \]

The identified parameters for degree 2 of the polynomials are as follows:

\[ A_0 = 1 - 2.0001q^{-1} + 1.0001q^{-2} \]
\[ c_{00} = -0.3411 \]
\[ B_{10} = 0.3762q^{-1} + 0.3759q^{-2} \]
\[ B_{20} = 0.0422q_1^{-1}q_2^{-1} + 0.0003q_1^{-1}q_2^{-2} + 0.041q_2^{-2} \]

The identified parameters for degree 4 of the polynomials are as follows:

\[ A_0 = 1 - 3.4403q^{-1} + 4.3807q^{-2} - 2.4406q^{-3} + 0.5002q^{-4} \]
\[ c_{00} = -0.0205 \]
\[ B_{10} = 0.3756q^{-1} - 0.1654q^{-2} - 0.3531q^{-3} + 0.1879q^{-4} \]
\[ B_{20} = 0.0428q_1^{-1}q_2^{-1} - 0.0189q_1^{-1}q_2^{-2} + 0.0403q_1^{-1}q_2^{-3} - 0.0403q_1^{-1}q_2^{-4} + 0q_1^{-2}q_2^{-2} - 0.0403q_1^{-2}q_2^{-3} + 0.0215q_1^{-2}q_2^{-4} + 0q_1^{-3}q_2^{-3} - 0.04q_1^{-3}q_2^{-4} + 0.0211q_1^{-4}q_2^{-4} \]

It is seen, that the cross terms are zeros or small to the coefficient of the quadratic terms. This is expected, as the system is given by a Hammerstein model.

Simulation results of the system with two integrators

Fig. 4.1.5 gives the output signal of the system and the identified model excited with the step-like input signal in case of memory 1. It seen that there is a deviation between the system and the model output. Fig. 4.1.6 shows the output signals of the model and the system excited with the step-like input signal in case of memory 2. The outputs of the system and the model are close to each other. With memory 4 the
coincidence is still better. Increasing the Volterra model degree the identification is improving.

Fig. 4.1.4. The input signal used for identification

Fig. 4.1.5. Output signals of the model and the system excited with step-like input signal in case of memory 1

Fig. 4.1.6. Output signals of the model and the system excited with step-like input signal in case of memory 2
4.1.7 Control simulation examples of the ball and beam system

The model with two integrators for memory 2 has been identified as

\[
y(k) = 2.0001y(k-1) - 1.0001y(k-2) - 0.3411 + 0.3762u(k-1) + 0.3759u(k-2) + \\
  + 0.0422u^2(k-1) + 0.0003u(k-1)u(k-2) + 0.041u^2(k-2)
\]

The suboptimal control algorithms with strategy 1 are applied for the Volterra system model. The reference signal changes between -12 and 12. The weighting factor of the control increments is 1.

Fig. 4.1.7 compares the controlled and the manipulated variables in case of two integrators for Volterra memory 2 and 4 with strategy 1. The prediction horizon is given by \( n_{e1} = 1 \) and \( n_{e2} = 5 \). Both cases show good control performance. With higher memory the manipulated variable is smoother. Fig. 4.1.8 compares the controlled and the manipulated variables in case of two integrators for different horizons with strategy 1 and memory 4. It is seen that by increasing the horizon results in smaller overshoot and smaller peaks in the manipulated variable.

![Fig. 4.1.7. Controlled and manipulated variables for two integrators, strategy 1, \( n_{e2} = 5 \)](image)
Fig. 4.1.8. Controlled and manipulated variables for two integrators, strategy 1, memory 4
4.2 An iterative GPC-like nonlinear predictive control algorithm for the parametric Volterra model

The system is modeled in the form of the parametric second-order Volterra model. A quadratic cost function calculating the differences of the future reference signal and the predicted output signal in the points of a given prediction horizon is minimized.

A new iterative solution for predictive control has been derived to obtain the series of the control signal based on the parametric Volterra model. This solution can be considered as an alternative of the iterative solution using the non-parametric Volterra model (Doyle et al. [22], Maner et al. [38]).

A case study of the level control of a two-tank system is discussed analyzing the performance of the different Volterra model based predictive control algorithms.

4.2.1 A new iterative control algorithm

In the following that case is considered when more and not necessarily equal changes are allowed in the control signal during the control horizon.

The output prediction can be described by a vector/matrix equation:

\[
\hat{y} = \hat{y}_0 + \hat{y}_1 + \hat{y}_2 = \hat{y}_0 + P_1 \cdot \Delta u + \hat{y}_2 = P_1 \cdot \Delta u + \hat{y}_{free+nlm}
\]  

(4-15)

where

\[
\hat{y}_{free+nlm} = \hat{y}_0 + \hat{y}_2
\]  

(4-16)

contains not only the free response (i.e. terms which are already known in the actual time point), but also the unknown, nonlinear quadratic terms of the actual and future manipulated signal increments.

The vectors give the performance of the system in a future horizon as
The predicted outputs can be composed of three terms:

\[
\hat{\mathbf{y}} = \hat{\mathbf{y}}_0 + \hat{\mathbf{y}}_1 + \hat{\mathbf{y}}_2
\]

where the constant term:

\[
\hat{\mathbf{y}}_0 = \begin{bmatrix}
\hat{y}(k + d + 1 + n_{c_1} | k) \\
\hat{y}(k + d + 2 + n_{c_1} | k) \\
\vdots \\
\hat{y}(k + d + 1 + n_{c_2} | k)
\end{bmatrix}
\]

the linear term:

\[
\hat{\mathbf{y}}_1 = \mathbf{P}_1 \cdot \Delta \mathbf{u} =
\begin{bmatrix}
\hat{y}(k + d + 1 + n_{c_1}) \\
\hat{y}(k + d + 2 + n_{c_1}) \\
\vdots \\
\hat{y}(k + d + 1 + n_{c_2})
\end{bmatrix}
\begin{bmatrix}
\Delta u(k) \\
\Delta u(k + 1) \\
\vdots \\
\Delta u(k + n_{c_2})
\end{bmatrix}
\]

the quadratic term:

\[
\hat{\mathbf{y}}_2 = \begin{bmatrix}
\hat{y}_2(k + d + 1 + n_{c_1} | k) \\
\hat{y}_2(k + d + 2 + n_{c_1} | k) \\
\vdots \\
\hat{y}_2(k + d + 1 + n_{c_2} | k)
\end{bmatrix}
\]

where the individual terms in vector \( \hat{\mathbf{y}}_2 \) are calculated according (4-11c).

The quadratic cost function can be minimized in different ways:
• by unconstrained or constrained numerical optimization of the cost function,
• by iterative usage of a quasi-linear unconstrained GPC (Generalized Predictive Control) algorithm (Clarke et al. [21]) and
• by an iterative usage of a quasi-linear constrained GPC algorithm applying e.g. quadratic programming.

An iterative algorithm was introduced in Doyle et al., (2001) [22], Maner et. al., (1996), [38] for the predictive control of the finite-response, non-parametric Volterra model, which performs the following steps:

1. Assume initial values for the actual and future manipulated signal increments ($\Delta u$) (based e.g. on the last control step).
2. Calculate the “free + nonlinear response” term $\hat{y}_{free+nlin}$ according to (4-16)
3. Calculate the new manipulated variable sequence
   - in the unconstrained case by
     $$\Delta u = [P_1^T \Lambda_p P_1 + \Lambda_m]^{-1} P_1^T \Lambda_n (y_n - \hat{y}_{free+nlin})$$
   - in the constrained case e.g. by quadratic programming considering the constraints
     $$\Delta u_{min} \leq \Delta u(k + j) \leq \Delta u_{max}; \quad j = 0,1,...,n_u - 1$$
     $$u_{min} \leq u(k + j) \leq u_{max}; \quad j = 0,1,...,n_u - 1$$
   These and other constraints can be considered easily in predictive control context (Camacho, Bordons, (1999,2004), [19], Maciejowski, J.M. (2002), [43]).
4. Check whether the new manipulated sequence differs from the previously calculated (or guessed) one. If the difference is larger than a predefined threshold vector, continue the iterative algorithm from Step 2 with the newly calculated future control signal sequence, otherwise terminate the iteration.

In Doyle et al. (2001), [22] and Maner et al. (1996), [38] a “mixed” predictive model description is used, that means incremental terms in the linear part ($\hat{y}_1$) and non-incremental terms in the quadratic part ($\hat{y}_2$). In Gruber et al. (submitted for publication,[24]) it is shown that this algorithm can be applied if the process is approximated and identified by a parametric (autoregressive) Volterra model and the control is realized based on the nonparametric Volterra series, using transformation equations between the parametric Volterra model and the finite response Volterra series (Haber and Keviczky,(1999), [25,26]). The new algorithm presented here is based on the publications mentioned and it has the following advantages:

• The control is based on the parametric Volterra model. (Parametric Volterra models approximate nonlinear dynamic processes with less parameters than the non-parametric (non-autoregressive) models using weighting function points.)
Both the linear and the quadratic terms of the controlled output are predicted based on system description with the control signal increments.

Also the iterative algorithm can be simplified by assuming equal control increments during the control horizon. This case, however, leads to the same results as the suboptimal strategy using one-dimensional minimization, presented before (Haber et al. (1999), [30]).

The iterative strategies of Doyle et al. (2001), [22] and Maner et al. (1996), [38] using the future control signal sequence and the new one using the future control signal increments in the quadratic part of the prediction equation provide similar results, however with completely different prediction equations. (The prediction equations differ also because of the difference between the models applied: finite-response or parametric.)

4.2.2 A case study: level control

Fig. 4.2.1 shows the scheme of a two-tank system. The inlet water quantity is denoted by $u_1$, the outlet quantities are $y_1$ and $y_2$, respectively. The water levels are denoted by $h_1$ and $h_2$. The cross sections of the tanks are $A_1$ and $A_2$. The model of the plant can be derived from physical equations.

Fig. 4.2.1. Scheme of a two-tank system
\[
\frac{dh_1}{dt} = \frac{u_1 - y_1}{A_1} = \frac{1}{A_1}u_1 - \frac{k_1}{A_1}\sqrt{h_1}
\]

\[
\frac{dh_2}{dt} = \frac{1}{A_2} (k_1\sqrt{h_1} - k_2\sqrt{h_2})
\]

(4.23)

\(k_1\) and \(k_2\) are constants. Relative units are used, all signals are related to their maximum values.

**Identification of the two-tank system in the form of a second-order Volterra model**

The model of the two-tank system is excited by a random signal composed of uniformly distributed random numbers on the interval \([0, 1]\). Each value is kept constant for 20 sampling points. The input and output data are collected and then offline LS parameter estimation is executed supposing a second-order parametric Volterra model.

The input and the output data of the level of the lower tank are shown in Fig. 4.2.2. The identified Volterra model of the tank is given by the following relationship:

\[
y(k) = 0.0017 + 1.5358y(k-1) - 0.6176y(k-2) + 0.0695u(k-1) - 0.0697u(k-2) - 0.0195u^2(k-1) + 0.0593u(k-1)u(k-2) + 0.0163u^2(k-2)
\]

Fig. 4.2.2. Input and output signals for identification
From this model a quadratic polynomial static characteristics is calculated in the following form:

\[ y_{\text{steady}} = 0.0208 - 0.0034u_{\text{steady}} + 0.6861u_{\text{steady}}^2 \]

The static characteristics and its approximation is shown in Fig. 4.2.3. Fig. 4.2.4. gives the output of the tank model and its Volterra approximation if the input is changed stepwise in every 150 sampling steps from 0 to 0.4, 0.5, 0.6, 0.3 and 0.5, respectively. It is seen that the approximation is acceptable. The fitting of the curves is better in the range where the static characteristics are tightly close to each other.

![Fig. 4.2.3. Static characteristics of the two-tank model and of its Volterra approximation](image)

**Predictive control of the two-tank system based on its Volterra model**

The identified Volterra model is used in the predictive control algorithm to calculate the manipulated variable. The output signal is calculated on the basis of the physical model of the plant. The reference signal changes from 0 to 0.2, then to 0.3 and to 0.2 again, and finally a big jump is acting changing the signal to 0.6. The iterative algorithm is applied with the following parameters: \( n_x = 1; \ n_u = 5; \ \lambda_c = 1; \ n_u = 1. \) Fig. 4.2.5. shows the output and the manipulated signals (full line). The control gives good performance in the whole range of the reference signal. In case of the last input change there is an acceptable overshoot. The control performance is compared with the suboptimal strategies when only one change is supposed in the manipulated
variable (dashed line), and with the case when equal changes are considered during the calculation (dotted line). It is seen that the iterative algorithm gives the fastest result. The other two suboptimal algorithms provide also good performance, the second algorithm supposing equal control increments provides the smoothest and slowest output signal. The running time of the suboptimal solutions is short, as they give analytic solution.

Fig. 4.2.4. The output of the tank model and of its Volterra approximation for stepwise input signal

Fig. 4.2.5. Predictive control of the two-tank system with the different algorithms
Fig. 4.2.6. shows the effect of the prediction horizon with the iterative algorithm. The tuning parameters are $\lambda_u = 1, n_u = 1, n_{r_1} = 1$ and $n_{r_2} = 1, 5, 10$.

It is seen that with longer horizon the control signal starts before the change of the reference signal and the output signal reaches its required value smoothly.

It can be concluded that the suggested nonlinear predictive algorithm is promising, providing good control results in the whole operating range.

Stability is not guaranteed, but during the iteration the convergence is indicated.
5. Summary of the new scientific results

In the last 30 years predictive control has gained wide acceptance in the process industries besides PID control. In case of systems containing big dead time significantly faster control performance can be achieved with predictive control than with PID control. Also in case of known reference trajectory predictive control provides better tracking properties. Nowadays several industrial program packages support an appropriate environment for industrial applications of predictive control.

I have addressed three topics related to predictive control.

1. In case of linear systems I have developed predictive PI(D) control algorithms. PI(D) control algorithms can be equipped with predictive property if the error signal applied to calculate the control signal is not the actual error, but its predicted value in a future point or in a future horizon. In this control system the operator will see PID control with embedded predictive properties. Aperiodic plants generally can be approximated by first-order models with dead time. I gave predictive PI(D) control algorithms supposing first-order and second-order models. The empirical tuning rules for PI(D) control can be modified for predictive PI(D) control. I applied set-point weighting to improve set-point tracking. I also investigated methods how to enhance robustness properties of the algorithms.

2. Industrial processes generally are complex, containing interacting MIMO controller loops. There are different decoupling techniques. Predictive control for MIMO processes ensures inherent decoupling if the cost function does not include terms punishing the control increments. But these terms are used to keep the control variables within their limit values. Enhancing decoupling properties of predictive control algorithms is important for practical applications. For improving decoupling properties of MIMO predictive control I applied adaptive tuning of the weighting factors in the cost function. I used these decoupling techniques for TITO processes and has shown their effectiveness in case of a model of a distillation plant.

3. Most of the plants contain nonlinearities. Predictive control algorithms based on nonlinear model of the plant would ensure better performance in the whole operating range, than algorithms using linear models. In recent years there is an active research to develop nonlinear predictive control algorithms. One direction is to use Wiener, Hammerstein or Volterra models to approximate the nonlinear characteristics and to develop predictive controllers based on these models. I have suggested an iterative optimization method to control nonlinear systems identified as Volterra models.
New Results

Thesis group 1: Predictive PI(D) control of linear SISO processes

1. I have introduced a simplified control horizon idea in the original formulation of predictive PI(D) control. I have derived predictive PI(D) control algorithms for simple plants with first- and second-order transfer functions with dead time based on their prediction equations.

2. I suggested simple tuning rules for predictive PI(D) controllers modifying the Kahn’s tuning rule for predictive PI(D) control. I applied tuning based on GPC-PID equivalence and gave a new calculation method for extended horizon control. I enhanced predictive PI(D) controllers by set-point weighting property introduced by Åström and Hägglund [17] for PID control.

3. I analysed the possibilities for enhancing robustness of predictive PI(D) controller using noise filter (T polynomial) and Smith predictor filter. I introduced the theory elaborated in [19, 41] for the predictive PI(D) control.

Related publications: [1,2,3,4,5,9,11,12,13,14]

Thesis group 2: Enhancing decoupling properties of MIMO predictive controllers

1. I used some decoupling techniques for TITO processes. With the first method the reference signal change is decelerated in order to make the control slower and to reduce the coupling effect. A new filter design is recommended for calculating the modified reference signal which suppresses the effect of the disturbance in the other control variable whose set-point was kept constant. With the second method the control error weighting factor is changed depending on the control error. The weighting factor of that variable, whose set-value was not changed is increased to suppress the decoupling effect. Instead of synchronizing this adoption to the set-point change using a signal detector, the weighting factor is set as a function of the control error. Thus an automatic adaption is possible.

Related publications: [7, 15]
Thesis 3: A new predictive control algorithm for nonlinear systems described by the parametric Volterra model

I have suggested an iterative optimization method to control nonlinear systems identified as Volterra models. The method is based on the parametric Volterra model. The optimization uses an iterative GPC-like technique, where the nonlinear prediction part is considered first as part of the free response. The performance of the control algorithm is demonstrated on level control of a two-tank model.

Related publications: [10, 13]
6. Conclusion

The most frequently applied controllers in the process industries are the PI(D) control algorithms. Control engineers and operators are familiar with the effects of the controller knobs and have an expertise in their tuning. There is a number of tuning rules based on simple process models. Such models can be obtained by simple measurements, e.g. by measuring the step response of the process, which generally can be executed under industrial circumstances. In case of significant dead time (transport delay) PI(D) algorithms provide slow control performance. To accelerate their response PI(D) algorithms can be furnished with predictive properties. The structure of predictive PI(D) algorithms is shown. Analytic expressions of predictive PI and PID algorithms are given for first- and second-order systems with dead time. Tuning rules equivalent to GPC are presented. Tuning rules of thumb (Kuhn, Åström-Hägglund, etc.) can also be used with some modifications. As in predictive PI(D) control the effect of the parallel paths is added, in this case the continuous gain has to be divided by the number of the parallel channels (this means that all the discrete controller parameter values are divided by the channel number). In case of plant/model mismatch the robustifying effect of the noise model filter \( T(q^{-1}) \) polynomial and of the robustifying filter in the Smith predictor-like scheme is added to the predictive control structure demonstrated. Simulation examples present the behavior of predictive PI(D) algorithms.

In control of MIMO processes decoupling is an important question. Predictive control of MIMO systems ensures decoupling if the input increments are not considered in the cost function. As inclusion of punishment of big control increments is important from practical point of view, improving of decoupling properties in this case has to be analysed.

Several decoupling methods were presented for TITO processes. In the first method the actual reference signal was substituted by a modified, decelerated reference signal in order to damp the disturbance caused in the other variable whose set-point was not changed. As an optimal modified reference signal the controlled signal in the not decoupled case was used. Instead of storing the whole reference trajectory a new method was recommended by identifying a reference signal filter and filtering with it the actual reference signal. In the second method the control error weighting factor of that variable, whose set-value was not changed is increased in order to suppress the decoupling effect. Instead of synchronizing this adoption to the set-point change by using a signal detector the weighting factor was set as a function of the control error. Thus an automatic adoption was possible.
Several simulations demonstrate the proper functioning of the proposed methods. The automatic adoption method is illustrated with the TITO model of a distillation column. While controlling the concentration of the distillation and the bottom product the usual coupling effect could be suppressed by a proper choice of the weighting factor adoption.

The procedures presented for TITO processes can be easily extended for MIMO systems of high dimensionality.

Most of the plants are nonlinear. Nevertheless in the practice generally linear control algorithms are applied considering the linearized model of the plant. Applying nonlinear control algorithms better performance is expected in the whole operating range.

Nonlinear Volterra model based predictive control algorithms derived by Haber (1995), [27] have been applied to control the nonlinear model of a ball and beam system. The system model was identified in the form of the parametric Volterra model. Matlab and Simulink programs have been written to realize the nonlinear control algorithms. Simulation results have shown the effectiveness of the control algorithms.

A new iterative nonlinear predictive control algorithm is presented here based on the quadratic parametric Volterra model. The algorithm uses a GPC-like structure. The control performance is demonstrated by a case study of level control of a two-tank system. The behavior of the new algorithm is compared with the suboptimal nonlinear algorithms derived in Haber (1995), [27]. The suggested nonlinear predictive algorithm is promising, providing good control results in the whole operating range.
7. Bibliography


[12] **Arousi, F.**, Schmitz, U., Bars, R., Haber, R. (2008). Robust predictive PI controller based on first-order dead time model. 17th IFAC World Congress, South Korea


91


[33] Ioannis Tziortzis (2007). Real time control of the ball and beam system. B.Sc. Diploma project, BME


Appendix A

Decoupling in control of MIMO systems

Realizing decoupling in control of MIMO processes, the design objective is to reduce control loop interactions by adding additional controllers called decouplers to a conventional configuration.

Fig. 1 shows the structure of the decoupling control scheme.

Fig. 1. Block diagram of decoupling control system

A structure of decoupling a MIMO control system for a process with two inputs and two outputs is shown in Fig. 1. (Seboreg et al. (1989)). Note that four controllers are used: two conventional feedback controllers, \( G_{c1} \) and \( G_{c2} \), plus two decouplers, \( D_{12} \) and \( D_{21} \). The input signal to each decoupler is the output signal from a feedback controller.

The transfer function matrix of the plant is
The decoupling controllers are designed to compensate for undesirable process interactions. For example, decoupler $D_{21}$ is to be designed to cancel $y_{21}$, which arises from the process interaction between $u_1$ and $y_2$. This cancellation will occur at the $y_2$ summation if the decoupler output $u_{21}$ satisfies

$$G_{p21}u_{11} + G_{p22}u_{21} = 0$$  \hspace{1cm} (1)$$

Substituting $u_{21} = D_{21}u_{11}$ and factoring gives

$$(G_{p21} + G_{p22}D_{21})u_{11} = 0$$  \hspace{1cm} (2)$$

$u_{11}(s) \neq 0$ since $u_{11}$ is a controller output that is time dependent. Thus, to satisfy Eq.1, it follows that

$$G_{p21} + G_{p22}D_{21} = 0$$  \hspace{1cm} (3)$$

Solving for $D_{21}$ gives the following an expression for the ideal decoupler

$$D_{21}(s) = -\frac{G_{p21}(s)}{G_{p22}(s)}$$  \hspace{1cm} (4)$$

In an analogous way we can derive a design equation for $D_{12}(s)$ by imposing the requirement that $u_{22}$ have no effect on $y_1$. Thus, the compensating signal $u_{12}$ and the process interaction due to $G_{p12}$ should cancel their effects in $y_1$

$$G_{p12}u_{22} + G_{p11}D_{12}u_{22} = 0$$  \hspace{1cm} (5)$$

The ideal decoupler is given by

$$D_{12}(s) = -\frac{G_{p12}(s)}{G_{p11}(s)}$$  \hspace{1cm} (6)$$

$$G_p(s) = \begin{bmatrix} G_{p11} & G_{p12} \\ G_{p21} & G_{p22} \end{bmatrix}$$
**Simulation example**

The transfer function matrix of the process discussed in subchapter 3.1.1 is considered.

\[
G_p(s) = \begin{bmatrix}
\frac{1.5}{(s+1)^2} e^{-0.1s} & \frac{0.5}{(0.5s+1)^4} e^{-0.5s} \\
0.75 e^{-0.8s} & \frac{1}{(2s+1)} e^{-0.2s}
\end{bmatrix}
\]

The ideal decouplers are

\[
D_{21}(s) = -\frac{G_{p21}(s)}{G_{p22}(s)} = -\frac{0.75(2s+1)}{(0.5s+1)^3} e^{-0.6} \quad D_{12}(s) = -\frac{G_{p12}(s)}{G_{p11}(s)} = -\frac{0.3333(s+1)^2}{(0.5s+1)^4}
\]

I designed two classic PI controllers for 60° phase margin as

\[
G_{c1} = \frac{0.3819(s+1)}{s} \quad ; \quad G_{c2} = \frac{5.2276(s+0.5)}{s}
\]

Fig. 2 shows the TITO predictive control without decoupling. The first and the third figure show the reference and output signals for the two controlled variables, while the second and the fourth figures give the manipulated variables. The effect of the coupling is seen. Changing of the first reference signal causes change in the second controlled variable and changing of the second reference signal influences the first controlled variable.

Fig. 3 gives the TITO predictive control with decoupling. It is seen that the effect of the coupling is reduced.

*Let us compare this method with the MIMO predictive control using the technique of control error dependent weighting factor adjusting (Fig. 3.10). It is seen that the performance of the MIMO predictive controller is better than the decoupling effect of the conventional decoupling method discussed above.*
Fig. 2. TITO control without decoupling: (top: CV1, MV1 bottom: CV2, MV2)
Fig. 3. TITO control with decoupling: (top: CV1, MV1 bottom: CV2, MV2)

Reference

Appendix B

1.1 Mathematical Model of the Ball and Beam System

The following denotations will be used to derive the mathematical model. The abbreviations have the following meaning:

- $m$: Mass of the ball
- $g$: Gravity
- $r$: Roll radius of the ball
- $I_b$: Inertia moment of the ball
- $I_w$: Inertia moment of the beam
- $M$: Mass of the beam
- $b$: Friction coefficient of the drive mechanics
- $K$: Stiffness of the drive mechanics
- $u(t)$: Force of the drive mechanics
- $l$: Radius of force application
- $l_w$: Radius of beam
- $x'$: Ball coordinates with respect to the beam
- $y'$: Ball coordinates with respect to the beam
- $\psi$: Angle of the ball to the beam
- $a$: Angle of the beam to the horizontal
A linear sliding friction in the drive mechanics is observable during a rotation, which is described by the friction coefficient $b$. The spring with the stiffness $K$ takes into account the delay behaviour of the driving belt which however may be neglected in the realized laboratory setup. With this the driving force $u(t)$ is the input variable of the system.

![Diagram of ball and rail system](image)

**Figure 2:** The different radius of the ball with respect to the rail of the beam

Because the ball does not roll in the plane, but in a groove (u-type profile), two different radius of the ball have to be considered on setting up the motion equations. One is the radius $R$ of the ball, the other is the roll radius $r$, given by the distance from the middle of the ball to the surface of the beam.

1.2
1.3 Motion equations of the system

**Motion equation of the beam**

$$mx^2 + I_b + I_w) \ddot{a} + (2mx \dot{x} + bl^2) \dot{a} + KI^2 a + (mr^2 + I_b) \frac{1}{r} \dot{x} - mgx \cos(a) = u(t)l \cos(a)$$

The following equation results, when ignore the second derivative of $x$.

$$mx^2 + I_b + I_w) \ddot{a} + (2mx \dot{x} + bl^2) \dot{a} + KI^2 a - mgx \cos(a) = u(t)l \cos(a)$$

Linearization of the previous equation about the beam angle, $a = 0$ gives the following linear approximation of the system.
\[ mx^2 + I_b + I_w \ddot{a} = u(t)l \]

Taking the Laplace transform of the previous equation, results to the following equation.
\[
(I_b + I_w)a(s)s^2 = u(s)l \Rightarrow a(s) = \frac{l}{(I_b + I_w) s^2}
\]

**Motion equation of the ball**

\[
\left( m + \frac{I_b}{r^2} \right) \ddot{x} + (mr^2 + I_b) \frac{1}{r} \dddot{a} - mx \dddot{a}^2 + mg \sin a = 0
\]

The second derivative of the input angle alpha \( a \) actually affects the second derivative of \( r \). However, this contribution was ignored. The Lagrangian equation of motion of the ball is then given by the following equation.

\[
\left( m + \frac{I_b}{r^2} \right) \ddot{x} - mx \dddot{a}^2 + mg \sin a = 0
\]

Linearization of this equation about the beam angle, \( a = 0 \) gives the following linear approximation of the system.

\[
\left( m + \frac{I_b}{r^2} \right) \ddot{x} = -mg \sin a
\]

Taking the Laplace transform of the previous equation, results to the following equation.

\[
\left( m + \frac{I_b}{r^2} \right) x(s)s^2 = -mg a(s) \Rightarrow x(s) = \frac{a(s)}{\left( \frac{I_b}{r^2} + m \right) s^2} \frac{1}{s^2}
\]

**Reference**

Ioannis Tziortzis (2007). Real time control of the ball and beam system. B.Sc. Diploma project, BME