DEVELOPMENT AND APPLICATION OF THE AUTOTUNING PID CONTROLLER

Master's thesis

Zagreb, 1999
This master's thesis was written at the Department of Control and Computer Engineering in Automation of the Faculty of Electrical Engineering and Computing, University of Zagreb.

Supervisor: Professor Nedjelko Perić, Ph. D.

The thesis counts 96 pages.

Number of the thesis:
Master's thesis evaluation committee included:

- Professor Zoran Vukić, Ph. D. – chairman, FEEC, Univ. of Zagreb
- Professor Ivica Mandić, Ph. D. – member, FEEMENA, Univ. of Split
- Professor Nedjelko Perić, Ph. D. – supervisor, FEEC, Univ. of Zagreb

Master's thesis exposition committee included:

- Professor Zoran Vukić, Ph. D. – chairman, FEEC, Univ. of Zagreb
- Professor Ivica Mandić, Ph. D. – member, FEEMENA, Univ. of Split
- Professor Nedjelko Perić, Ph. D. – supervisor, FEEC, Univ. of Zagreb

Date of exposition: October 28, 1999.
I wish to thank my supervisor Professor Nedjelko Perić and co-
supervisor Ivan Petrović, Ph.D. for guidance and encouragement
to write this thesis.

My thanks go to all my friends from the Department who created
an inspiring ambient for work on the thesis.

*To my family. This work is a small token of gratitude for all you
gave me.*
CONTENTS

1 INTRODUCTION ............................................................................................................. 1

2 PID CONTROL ............................................................................................................... 3

2.1 Forms of the PID controller ...................................................................................... 4

2.2 Practical issues in the application of PID control ..................................................... 8

2.3 Tuning methods for PID controllers ......................................................................... 11

2.3.1 Ziegler-Nichols tuning rules ................................................................................. 14

2.3.2 PID tuning based on integral criteria ................................................................... 17

2.3.3 Cohen-Coon tuning rules ...................................................................................... 19

2.3.4 PID tuning based on gain and phase margin specifications ................................ 20

2.3.5 Approximate pole placement method: Lambda tuning ........................................ 21

2.3.6 Approximate pole placement method: Dominant pole design ......................... 22

2.3.7 Magnitude optimum and symmetric optimum tuning methods ......................... 25

2.3.8 PID tuning in the framework of Internal Model Control .................................... 29

2.4 Usage of PI controller in dead-time compensating controllers ............................. 32

3 AUTOTUNING PID CONTROLLER .............................................................................. 36

3.1 An overview of adaptive control ................................................................................ 36

3.2 Basic autotuning algorithm ...................................................................................... 39

3.3 Modifications of the basic autotuning algorithm ....................................................... 45

3.3.1 Relay experiment with adjustable dead time ....................................................... 45

3.3.2 Estimation of mathematical model parameters using relay experiment .............. 48

3.3.3 Autotuning in the presence of constant load disturbance .................................... 49

3.3.4 Addition of the preliminary identification phase .................................................. 52

3.3.5 Other modifications of the basic autotuning algorithm ........................................ 59

3.4 Examples of adaptive PID controllers ..................................................................... 60

4 SIMULATION STUDY OF THE AUTOTUNING PID ALGORITHM .......................... 65

4.1 Simulation and testing of the preliminary identification procedure ....................... 65

4.2 Choice of dead-time compensating controller ...................................................... 69
1 INTRODUCTION

This master's thesis addresses the practical problem of finding a high-performance controller, able to control as wide a range of processes as possible, and requiring minimum tuning effort from the operator. Furthermore, the eventual result of this thesis should be completely designed software product, stand-alone module for programmable logic controller (PLC).

In order to achieve these goals several possible designs were considered, and the autotuning controller introduced by the famous paper of Åström and Hägglund (1984), was selected. An appealing property of the autotuning controller is that it resolves two problems that can be regarded as the core problems in the field of control engineering, that is, the identification of the controlled process and the design of a suitable controller. The autotuning controller efficiently resolves both of these problems. Simple relay experiment is used for the identification, and a classical PID controller is used for the control of the process.

Autotuning has been applied in diverse commercial controllers, and industrial experience has proven its usefulness. Åstrom et al. (1993) note that autotuning is a highly desirable and useful controller feature. It is especially useful in commissioning of the control system, as it shortens the start-up time. During normal operation of the plant, autotuning can be used for regular re-tuning of the controller and for setting a parameter table for gain scheduling.

In spite of the obvious advantages of the autotuning controller, the question arises why would anyone bother to work on a half-century old control algorithm such as PID? Many controllers have been proposed in the scientific literature, but PID controller is applied in the majority of the control loops in the process industry. According to some studies, more than 95% of control loops are of the PID type, of which most are actually of the PI type (Åstrom and Hägglund, 1995). Its popularity eventually led to the recognition of many other advantages of PID control. For example, Marsili-Libelli (1981) recognises structural advantages of PID controllers in constant disturbance cancellation, perfect tracking of a constant set point, and reduced sensitivity to parameter variations.

However, as a control strategy PID controller with constant parameters is not suitable for the application in next cases (Åstrom and Hägglund, 1995):

- High quality control (tight control) of higher order processes;
- Control of processes with large dead time;
- Control of processes with oscillatory modes;
- Control of processes with time-varying parameters.

In these cases advanced control strategies have to be employed. Sometimes, a PID controller is incorporated into such control strategy, as in some dead-time compensating controllers (Normey-Rico et al., 1997; Hägglund, 1996) or in some self-tuning controllers (Radke and Isermann, 1987).
1. Introduction

With respect to previous considerations, the objectives of this paper are to:

• Review and systematise the existing techniques in PID control;
• Evaluate different autotuning controllers;
• Extend autotuning algorithm with new capabilities;
• Implement the extended algorithm as a stand-alone software module in PLC.

The material for this master's thesis evolved from work with several students on their graduation theses (Tomić, 1997; Šilj, 1997; Juretić, 1998; Kousek, 1999) and from papers presented on scientific gatherings (Perić et al., 1997; Petrović et al., 1998; Branica et al., 1999).

Let me give a brief outline of the thesis at this point. This first chapter is an introduction to the subject matter. The second chapter gives an overview of techniques in PID control. It proceeds with a discussion about the theoretical basis and the implementation of PID controllers and pays special attention to techniques of controller anti-windup and bumpless parameter transfer. Moreover, this chapter includes considerations of issues in the digital implementation of PID controller and reviews several PID design techniques. Most commonly used design techniques, both classical and modern, are commented. The chapter concludes with an insight into the application of PI control in dead time compensating controllers and with a test the controllers' robustness.

The third chapter outlines the autotuning algorithm and points out its relation to other adaptive techniques. It explains the basic autotuning algorithm and its usage and proposes some modifications in detail. The chapter details the process parameter estimation for two typical process models by relay experiment, presents the derivation of preliminary identification procedure, and, finally, gives some examples of the adaptive usage of the PID controller, commenting on the advantages and features.

The fourth chapter describes the simulation of the autotuning PID controller and pays particular attention to details of the preliminary identification algorithm. The chapter further explains the choice of dead time compensating controller for incorporation into the autotuning algorithm. Functioning of the autotuning algorithm is explored through functional description and by simulation examples.

The fifth chapter describes the implementation of the algorithm in PLC and the implementation difficulties and trade-offs related to development environment constraints. Behaviour of the implemented PLC module is checked through several experiments on laboratory processes.

The thesis concludes with the sixth chapter. Conclusive remarks consider the usability of the implemented PLC module. Pointers regarding further work are also given. Finishing sections of the work are literature, abstract, and a short biography of the author.
This paper investigates the PID control of the Single-Input Single-Output (SISO) processes. The basic configuration of the control system is shown in the block diagram below (Fig. 2.1). The block diagram gives the notation used for the signals in the control system.

The error signal $e(t)$ is formed as the difference between the reference signal $r(t)$ and the output signal $y(t)$. The PID controller acts on the error signal to form control signal $u(t)$. Some PID controllers with modified structure also use output signal to form control signal. Disturbance $z(t)$ can be introduced into the process at different points. Furthermore, various noise sources (e.g. process noise, measurement noise) can be present in the control system, but it is adequate to model noise influence on the control system with the signal $n(t)$, which is added to the output of the process to form the output signal.

The block marked »Process« in Figure 2.1 includes all the elements of the control system which are considered as the parts of the process: actuator, plant, and sensor. The mathematical model of the process can be very complex, with complicated static and dynamic description. The identification of a complex model requires a lot of engineering effort. Since the performance obtained from control system with the PID controller is limited, many PID controller tuning methods use simple models which have similar complexity as the PID controller. These models require simple identification experiments and capture dominant dynamic properties. Usual representation of these models is low-order (first or second order) transfer function in Laplace domain. On the other hand, some tuning methods for PID controllers were developed for more complex process models (e.g. higher order models, models with non-linear characteristics) because of successful and widespread use of PID controllers in industry (Persson, 1992). A transfer function modeling the process is generally represented as follows:

$$G_p(s) = K \frac{1 + b_1 s + b_2 s^2 + \ldots + b_m s^m}{s^n (1 + a_1 s + a_2 s^2 + \ldots + a_m s^m)} e^{-T_t s}, \text{ with } m \leq n,$$

where $T_t$ is dead time, $m$ and $n$ are degrees of complex variable polynomials, and $k$ is the number of integrators present in the process. The type of the process model is determined by the
exponent $k$, so that a process without an integrator ($k=0$) is called »type 0 process«, a process with an integrator ($k=1$) is called »type 1 process«, and so on.

Two common process models are used in this study. The first model is the First Order with Dead Time (FODT) model, often used for the description of chemical processes, given by the transfer function:

$$G_p(s) = G_t(s) = \frac{K_{e}e^{-T_{p}s}}{1+T_{s}s}. \quad (2-2)$$

The second model is frequently employed to describe electromechanical processes. It consists of an integration and a first order lag:

$$G_p(s) = G_t(s) = \frac{K_{2}}{s(1+T_{2}s)}. \quad (2-3)$$

A PID controller consists of the three terms: proportional (P), integral (I), and derivative (D). Its behavior can be roughly interpreted as the sum of the three term actions: the P term gives a rapid control response and a possible steady state error; the I term eliminates the steady state error; and the D term improves the behavior of the control system during transients. This description of the term actions matches the actual behavior of the PID control system, when it is used for some processes and for some excitation signals (for example, type 0 process and step set-point change).

A PID-type controller can be implemented variously. The implementation choice depends on the structure of the process, design specifications, and PID controller’s features. The following subsections describe how controllers apply the PID control law through the review of different PID controller forms and implementation aspects. The section proceeds with an outline of different tuning rules and explains the usage of PI controllers in dead-time compensating controllers.

### 2.1 Forms of the PID controller

Different forms of PID controller reflect the development of the PID algorithm in different technologies and its use in diverse control systems. Besides, some PID forms ensure better performance and behavior of the control system than others. The textbook version of the PID control law in the time domain is:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} = K_p e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt}, \quad (2-4)$$

where $K_p$ is proportional gain, $K_i$ is the gain of the integral term, $K_d$ is the gain of the derivative term, $T_i$ is the integral time constant, and $T_d$ is the derivative time constant.

The Laplace transformation of equation (2-4) gives the transfer function of the PID controller:
Since the numerator of the PID controller transfer function in (2-5) has a higher degree than the denominator, the transfer function is not causal and as such can not be realized. The form (2-5) of the PID controller is modified in such way as to make the controller. It is achieved through the addition of a lag to the derivative term:

\[ G_R(s) = \frac{U(s)}{E(s)} = K_p (1 + \frac{1}{T_R s} + \frac{T_D s}{1 + \frac{T_D}{N} s}), \quad (2-6) \]

where \( T_D/N \) is the time constant of the added lag. Divisor \( N \) in (2-6) determines the gain \( K_{HF} \) of the PID controller in the high frequency range:

\[ K_{HF} = \lim_{\omega \to \infty} G_R(j\omega) = K_p(1 + N). \quad (2-7) \]

The gain \( K_{HF} \) must be limited because measurement noise signal \( n(t) \) often contains high frequency components and its amplification should be limited. Usually, the divisor \( N \) is chosen in the range \( 3 \div 10 \) (Hang et al., 1991).

The form of the PID controller defined by (2-5) is called the parallel form, because proportional, integral, and derivative term act simultaneously on the error signal and the control signal is the sum of the term actions. It is also called non-interactive form of the PID controller (see Figure 2.2-a.).

Another form of the PID controller is the series or interacting form (Fig. 2.2.b.) with transfer function:

\[ G_{RS}(s) = K_p(1 + \frac{1}{sT_{IS}T_{DS}})(1 + sT_{DS}), \quad (2-8) \]

This form of the PID controller has a simple representation in the frequency domain, since all roots and zeros of \( G_{RS}(s) \) are real and correspond to the inverses of the break frequencies. This form of the PID controller is also referred to as a classical form of the PID controller, since it
conforms to the structure of PID controllers when these were implemented in the pneumatic technique.

Based on equations (2-6) and (2-8), the relations for converting the parameters between the parallel and the series form of the PID controller are:

\[ K_{P_{PS}} = \frac{K_p}{2} (1 + \sqrt{1 - 4\frac{T_D}{T_I}}), \]  
\[ T_{I_{PS}} = \frac{T_i}{2} (1 + \sqrt{1 - 4\frac{T_D}{T_I}}), \]  
\[ T_{D_{PS}} = \frac{T_I}{2} (1 - \sqrt{1 - 4\frac{T_D}{T_I}}), \]

and can be used only if \( T_i \geq 4T_D \), i.e. when poles and zeros of the parallel form are real.

As output of the above forms of the PID controller is the total value of the control signal \( u(t) \), they are called positional PID algorithms. Some actuators such as a motor may use the increment or derivative of the control signal as an input signal, because they have built-in integral action. PID controllers with such an output are termed velocity (continuous version) or incremental (discrete version) PID controllers. An advantage of an incremental version of the PID algorithm is that it allows straightforward implementation of the algorithm extensions like anti-windup scheme and bumpless parameter switch (Isermann, 1989) which are described in the following subsection.

![Fig. 2.3. Two-degrees of freedom PID controller](image)

Standard PID controllers act on the error signal \( e(t) \) and give the control signal \( u(t) \) as the output. Such configuration of the controller uses the same parameters as in responding to set-point change (tracking) and to load disturbance (regulating). The two functions of the control system often impose contradictory demands on the value of the controller parameters. The contradiction is resolved by a trade-off in the controller's design. The trade-off can be recognized
in the responses of the control system to set-point change and to load disturbance, which can not have the same quality. In order to avoid this trade-off a modification of the PID controller structure was devised (Fig. 2.3.). Signal channels for reference signal and for measurement signal are separated, and a set of weights \((F_P, F_I, F_D)\) in the channel of the reference signal is introduced. Weights in the channel of the reference signal allow design, which arbitrarily assigns zeros of the closed-loop transfer function, and therefore define dynamic behavior of the closed-loop system to set-point change.

The PID controller with set-point weighting is tuned in two steps:
1. Parameters are tuned for good regulation;
2. Weights \(F\) are adjusted in order to set zeros of the closed-loop transfer function and thus to improve the tracking behavior of the control system.

Controllers that allow such separation of the design for regulating and for tracking are called two-degree-of-freedom controllers. Eitelberg (1987) and Hippe et al. (1987) gave some recommendations for the appropriate choice of weights and proposed the use of filters instead of constant weights.

The introduction of arbitrary weights in the reference channel of the PID controller gives design more freedom and renders it more complicated. In some PID controller implementations (Stojić and Petrović, 1986; Hang et al., 1989) weights are set to \(F_D=0\) and \(F_P=0\) in order to avoid derivative and proportional bumps, which are present in response to step set-point change. Furthermore, the weight of integral term is set to \(F_I=1\), which ensures that the steady state error for unit step set-point change equals zero.

The standard form of the PID controller, similar to the structure depicted in Figure 2.3, and recommended by Instrument Society of America, is given below (Åstrom and Hägglund, 1995):

\[
U(s) = K_d [(F_P R(s) - Y(s)) + \frac{1}{s T_I} (R(s) - Y(s)) + \frac{s T_D}{1 + s T_D / N} (F_D R(s) - Y(s))] \quad (2-12)
\]

Frequently, only a part of a PID controller is used. Åstrom and Hägglund (1995) have noted that most control loops are of the PI type. As a rule, the PI controller is used for processes of the first order, or for processes not requiring tight control. A PD controller can be used for processes which contain integrators, and which do not have constant load disturbances, since PD controller can not compensate for it. The application of a P controller is limited to simple control tasks.

Today, almost all control strategies are implemented as digital algorithms in various devices such as Programmable Logic Controllers (PLCs), Digital Signal Processors (DSPs), and in other microprocessor-based equipment. To become applicable in such equipment, the PID control algorithm has to be discretized. Using Euler integration method – rectangular integration, the discrete version of the positional algorithm (2-4) is calculated as:

\[
u(k) = K_p \left[ e(k) + \frac{T}{T_i} \sum_{i=0}^{k-1} e(i) + \frac{T}{T} (e(k) - e(k-1)) \right], \quad (2-13)
\]

where \(k\) denotes discrete time instant, and \(T\) is the sampling time. Recursive equation describing the incremental version of the PID algorithm is obtained when the equation (2-13) for the time instant \(k-1\) is subtracted from the same equation for the time instant \(k\):
\[
\Delta u(k) = u(k) - u(k-1) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2),
\]
(2-14)

where:
\[
q_0 = K_P \left(1 + \frac{T_D}{T}\right),
\]
\[
q_1 = K_P \left(1 + 2 \frac{T_D}{T} - \frac{T}{T_I}\right)
\]
and
\[
q_2 = K_P \frac{T_D}{T}.
\]

Other relations for parameters \( q_i \) in (2-14) are obtained if a different integration method (e.g. trapezoidal method) is used. These discrete approximations of the continuous PID controller are valid only if the sampling time \( T \) is sufficiently short in comparison to the time constants of the controller. Otherwise, when sampling time \( T \) is not much shorter than the time constants of the controller, connection with continuous PID controllers is dropped and \( Z \)-transform form of the PID controller is used (Isermann, 1989). Discrete controller of the second order with an integrator has a transfer function in the \( Z \)-domain:
\[
G_c(z) = \frac{q_0 + q_1 z^{-1} + q_2 z^{-2}}{1 - z^{-1}}.
\]
(2-15)

Polynomial coefficients \( q_i \) in (2-15) have to satisfy the following relations (Isermann, 1989):
\[
q_0 > 0,
\]
\[
q_1 < -q_0,
\]
\[
-(q_0 + q_1) < q_2 < q_0,
\]
so that the obtained digital controller has a dynamic behavior of the continuous PID controller.

### 2.2 Practical issues in the application of PID control

All controllers are designed to work with processes which have some physical constraint: valves have a limited operating range (0%-100%), pumps have limited power, motors have a maximum moment, and so on. These limitations can be regarded as non-linearities in the process and have to be considered in the application of the controller. Since many of these limitations appear at the input of the actuator, they are referred to as input limitations and are modeled with a non-linear element having a saturation characteristic, as shown in Figure 2.4. It is assumed that the disturbance \( z(t) \) adds to the input signal of the process and adequately models actual disturbances in the control system. Beside input magnitude limitations, actuators often have defined rate limitations, or maximum rates at which the control signal \( u(t) \) can be changed.

When the controller output signal \( u_0(t) \) exceeds the upper limit \( u_{max} \), or when it falls below the lower limit \( u_{min} \) of the operating range, its value changes in the input limitation element so that the controller output signal \( u_0(t) \) and the process input \( u(t) \) do not coincide. Shouldn’t the two coincide, the integrator in a controller with integral action would produce an inaccurate and highly excessive value which would cause oscillation and slowing down of the transient response. In other words, the effect would be a large overshoot and a long settling time.
This behavior is called the integrator windup. Moreover, the feedback loop during the windup behaves as if it were broken. There are several anti-windup algorithms to avoid adverse effects of the integrator windup on the control system performance. This paper describes the most frequently used anti-windup algorithms. The reader may further consult comparative studies of these and other algorithms in Bohn and Atherton (1995) and Vrančić (1997).

![Fig. 2.4. Input limitation as a part of the control system.](image)

Figure 2.5 shows the structure of the linear feedback anti-windup algorithm. It relies on the assumption that it is possible to measure both the controller output signal $u_0(t)$ and the process input signal $u(t)$. If the measurement of the process input signal is not possible, the simulation of the saturation element in the controller can be used. A new signal $e_{aw}(t)$ is added as an additional input to the integrator of the controller. It is active when there is difference between the controller output $u_0(t)$ and the process input $u(t)$. It acts in direction opposite to the windup effect. The rate of anti-windup action is defined with the constant $T_{AW}$ which can be explained as a time constant of this action. Åstrom and Hägglund (1995) calculated its value as follows: $T_{AW} = \sqrt{T_I T_D}$.

![Fig. 2.5. Structure of the linear feedback anti-windup algorithm.](image)

Another anti-windup algorithm, suitable for discrete implementation, is the conditional integration algorithm. It allows integration of the error signal $e(t)$ in the integrator element.
provided that some conditions imposed on the signals present in the control system are met. Otherwise, the integration is not permitted. The condition that stops the integration can be expressed as (control signal $u_0(t)$ is in the saturation) AND (the sign of the integral increment moves state of integrator deeper into saturation). Furthermore, it is advisable to add a small hysteresis, so that control signal does not oscillate around the limit value.

Similar deterioration of the control system performance happens when the source of the control signal $u(t)$ is changed, for example, when the controller is substituted with another, or when it is switched from manual to automatic mode. The switch between two different modes of control system operation is called plant-input substitution (Peng et al., 1996). A bump in the control signal $u(t)$ reflects such plant-input substitution unless the switching controllers are properly prepared. The effect can be avoided by using bumpless transfer techniques. These techniques calculate the states of the substitution controller before the switch happens, so that bump in the control signal does not occur. Peng et al. (1996) offer a good survey of bumpless transfer techniques.

Kothare et al. (1994) have introduced a unified theoretical framework for the study of anti-windup and bumpless transfer (AWBT). It is based on the approach which, as a first step, designs the linear (PID) controller ignoring non-linear input characteristics of some elements in the control loop and then adds AWBT compensation to minimize negative effects of these characteristics on the control loop performance. They have shown that their theory applies in the analysis of all anti-windup techniques and bumpless transfer, and that two matrices are enough to parametrize these characteristics.

An important issue in the implementation of discrete control algorithms is the choice of sampling time. That choice depends on the control-loop dynamics and should follow the recommendation given in the Shannon’s theorem. Since there are many signals and elements in the control loop with different dynamic properties, it is not always clear how to choose the sampling time. For the discrete PID controller, Isermann (1989) relates the sampling time $T$ to the settling time $T_{95\%}$ of the process (time required for the response to reach 95% of its final value):

$$\frac{T_{95\%}}{T} \approx 5 \div 15.$$  \hspace{1cm} (2-16)

Some rules of thumb have been established for relating sampling time $T$ to the parameters of PI and PID controllers (Åstrom and Wittenmark, 1990):

- **PI controller:**
  $$\frac{T}{T_I} \approx 0.1 \div 0.3 ;$$  \hspace{1cm} (2-17)

- **PID controller:**
  $$\frac{TN}{T_D} \approx 0.2 \div 0.6 ,$$  \hspace{1cm} (2-18)
where $N$ is divisor constant from equation (2-6). Previous relations should serve as guidelines and, if necessary, should be adjusted for the particular use.

Additionally, the discrete implementation of the PID controller raises several other issues which have to be addressed:

- Effects of finite word length;
- Signal quantization effects;
- Signal conditioning and prefiltering problems.

Solutions and trade-offs concerning these issues have been addressed in many textbooks (e.g. Åstrom and Wittenmark, 1990; Isermann, 1989) and in specialized literature (e.g. Åstrom and Steingrimsson, 1991).

### 2.3 Tuning methods for PID controllers

Controller tuning methods provide the controller parameters in the form of formulae or algorithms. They ensure that the obtained control system would be stable and would meet given objectives. These methods require certain knowledge about the controlled process. This knowledge, which depends on the applied method, usually translates into a transfer function. The objectives which should be achieved by the application of the control system are associated with the following control system features (Persson, 1992):

- Regulating performance;
- Tracking performance;
- Robustness;
- Noise attenuation.

Often, the desired objectives put contradictory demands on the values of the controller parameters, so that various trade-offs have to be made. The objectives can be stated in many ways such as through:

- Specifications within the time domain;
- Specifications within the frequency domain;
- Robustness specifications;
- Other specifications.

The specifications within the time domain give some values related to the shape of control system signals in the time domain. Figure 2.6 shows a typical output signal $y(t)$, a response to set-point change. Specification values within the time domain are marked on it: overshoot $\sigma_m$, undershoot $\sigma_u$, rise time $t_r$, time of first maximum $t_m$, settling time $t_e$, and steady state error $e_{ss}$. Similar specifications within the time domain are used to describe characteristics
of the control system response to load disturbance: peak perturbation $\sigma_{dm}$ and disturbance settling time $t_{dc}$.

![Fig. 2.6. Specifications in time domain.](image)

The specifications within the frequency domain define some values related to the frequency characteristics of transfer functions of various elements in the control system. Bandwidth of a closed-loop control system with transfer function $G(s)$ is the lowest frequency $\omega_b$ for which below relation holds:

$$\left| \frac{G(j\omega_b)}{G(0)} \right| = \frac{1}{\sqrt{2}}$$  \hspace{1cm} (2-19)

The gain margin $A_M$ of the control system, described with the open-loop transfer function $G_O(s)$, is defined as the inverse of the open-loop gain at the phase crossover frequency $\omega_\pi$:

$$A_M = \frac{1}{\left| G_O(j\omega_\pi) \right|},$$  \hspace{1cm} (2-20)

where the frequency $\omega_\pi$ is defined as the lowest frequency with

$$\arg[G_O(j\omega)] = -\pi.$$  \hspace{1cm} (2-21)

The phase crossover frequency $\omega_\pi$ is also called the ultimate frequency of the control system.

The phase margin $\gamma$ of the control system is defined as the phase of the open-loop transfer function $G_O(s)$ at the gain crossover frequency $\omega_c$:

$$\gamma = \arg[G_O(j\omega_c)] + \pi,$$  \hspace{1cm} (2-22)

where the frequency $\omega_c$ is defined as the lowest frequency with

$$\left| G_O(j\omega_c) \right| = 1.$$  \hspace{1cm} (2-23)

Maximum sensitivity $M_s$ of the control system, also called modulus margin, is defined as:

$$M_s = \max_\omega \left| \frac{1}{1 + G_O(j\omega)} \right|,$$  \hspace{1cm} (2-24)
and can be interpreted as the inverse of the shortest distance between the critical point $C(-1, i0)$ in the Nyquist plane and the Nyquist curve. The above definition (2-24) for the maximum sensitivity $M_c$ makes it possible to relate $M_s$ to gain and phase margin of the control system (Persson, 1992):

$$A_r \geq \frac{M_c}{M_s - 1},$$  \hspace{1cm} (2-25)

$$\gamma \geq 2 \arcsin \left( \frac{1}{2M_s} \right),$$ \hspace{1cm} (2-26)

Figure 2.7 gives an example with the Nyquist curve of a process and specifications within the frequency domain.

Robustness specifications define allowed deviation of the process parameters from nominal values. The control system should retain designed stability and performance in the range of these deviations. The parameter deviations from nominal values can be defined as multiplicative or additive parameter uncertainty characteristics, expressed in the frequency domain (Morari and Zafiriou, 1989). Besides, the robustness of the control system can be specified in terms of gain margin $A_r$, phase margin $\gamma$, and as maximum allowed sensitivity $M_s$ of the control system.

Other specifications which define a control system requirements include the description of the process constrains and other implementation-related specifications.

![Nyquist curve](image)

Fig. 2.7. Specifications in frequency domain.

Some controller tuning methods include recommendations of a suitable controller structure and its parameters. This paper describes tuning methods for the fixed structure of PID controllers. P, PI, and PD controllers are considered as special cases of PID controller. The tuning methods for PID controllers can be grouped according to their nature and usage, as follows:
• Heuristic methods evolved from practical experience in PID controller tuning;
• Frequency methods employ frequency characteristics of the controlled process to tune PID controller parameters;
• Analytical methods calculate PID controller parameters from analytical or algebraic relations that define control system by direct calculation;
• Loop-shaping methods seek to shape the open-loop transfer function of the control system into a desirable form;
• Optimization methods obtain PID controller parameters from different optimization algorithms;
• Methods in which PID controller represents a restriction of possible controller structure (e.g. PID controller tuning in the framework of Internal Model Control);
• Methods for tuning a PID controller which functions as a part of an advanced control strategy (e.g. usage of PI controller in dead-time compensating controllers).

The above groups do not sharply distinguish and some methods may belong to more than one group. An important criterion in the evaluation of the presented tuning methods is the suitability of a particular method for the on-line usage. This especially refers to the possibility to use a particular method for autotuning.

### 2.3.1 Ziegler-Nichols tuning rules

Ziegler and Nichols have introduced a useful methodology for controller tuning. It consists of a simple experiment with a controlled process and extracts some of its features. Once the experiment is completed, the method provides tables by which it is possible to calculate the controller parameters. The tuning tables were developed through numerous experiments which involved different processes. The goal of the design was to find a controller which gives the quarter amplitude damping (QAD) ratio of the control systems in response to load disturbance. QAD ratio is achieved when the ratio of the first overshoot and the first undershoot of the control system response equals ¼ (Åstrom and Hägglund, 1995). This design specification arises from empirical observations and has been used traditionally, but gives too oscillatory control systems. Ziegler and Nichols considered P, PI, and PID controllers in their work.

The first experiment consists of measuring apparent dead-time $T_{ZN1}$ and the maximum slope of the response on the process reaction curve (response to step set-point change) $K_{ZN1}$. The measurements are shown in Figure 2.8.a, and relations for obtaining controller parameters in Table 2.1.a.
2. PID control

In the second experiment, the process is controlled with a proportional controller. The gain of the controller is gradually increased until the control system reaches stable oscillations on the stability limit. The value of the controller gain $K_U$ is called the ultimate gain, and the oscillation period $T_U$ is called the ultimate period. The two values serve as the basis for the calculation of the controller parameters (see Table 2.1.b). Figure 2.8.b shows a typical process output $y(t)$ during such an experiment.

![Diagram](image)

**Fig. 2.8. Ziegler-Nichols experiments: output signal of the control system.**

In the second experiment, the process is controlled with a proportional controller. The gain of the controller is gradually increased until the control system reaches stable oscillations on the stability limit. The value of the controller gain $K_U$ is called the ultimate gain, and the oscillation period $T_U$ is called the ultimate period. The two values serve as the basis for the calculation of the controller parameters (see Table 2.1.b). Figure 2.8.b shows a typical process output $y(t)$ during such an experiment.

<table>
<thead>
<tr>
<th>Controller</th>
<th>$K_P$</th>
<th>$T_I$</th>
<th>$T_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>$1/(T_{ZNI} K_{ZNI})$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PI</td>
<td>$0.9/(T_{ZNI} K_{ZNI})$</td>
<td>$3 T_{ZNI}$</td>
<td>-</td>
</tr>
<tr>
<td>PID</td>
<td>$1.2/(T_{ZNI} K_{ZNI})$</td>
<td>$2 T_{ZNI}$ $T_{ZNI}/2$</td>
<td>-</td>
</tr>
</tbody>
</table>

**Tab. 2.1. Ziegler-Nichols relations for calculating controller parameters.**

Takahashi has developed similar relations (Table 2.2) to calculate the discrete P, PI, and PID controller parameters. These relations use the incremental form of the PID controller, as follows:

$$u(k) = u(k-1) + K_P \left[ y(k-1) - y(k) + \frac{T}{T_I} e(k) + \frac{T_D}{T} \left[2y(k-1) - y(k-2) - y(k)\right]\right]. (2-27)$$

An additional parameter to the original relations in Table 2.2 is the sampling time $T$. In relation to other time constants, the sampling time $T$ has to be small enough for the relations to produce useful results (Isermann, 1989).
Many authors assessed the performance of the control systems with controllers tuned according to Ziegler-Nichols (ZN) rules (e.g. Åström and Hägglund, 1995; Hang et al., 1991; Thomas, 1991). The comparison of the two ZN methods shows that the second can be regarded as better, since the first method fails to make it clear how to measure apparent dead time $T_{ZN1}$. Moreover, for many processes the controller gain obtained with the first method is 25%–40% higher than the gain obtained with the second method, giving more oscillatory response to set-point change. Hang et al. (1991) showed that the performance of the controllers tuned according to ZN rules depended strongly on the value of the process dead-time, and that ZN rules often gave poor damping and excessive overshoot in response to set-point change. ZN tuning rules for PI controller show worse performance than rules for the PID controller and even produce unstable control systems for some processes (Thomas, 1991).

<table>
<thead>
<tr>
<th>Controller</th>
<th>$K_P$</th>
<th>$T/T_i$</th>
<th>$T_D/T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PI</td>
<td>$0.9 - \frac{0.135T}{K_{ZN1}(T_{ZN1} + T)}$</td>
<td>$0.27T$</td>
<td>-</td>
</tr>
<tr>
<td>PID</td>
<td>$1.2 - \frac{0.3T}{K_{ZN1}(T_{ZN1} + T)}$</td>
<td>$0.6T$</td>
<td>$0.5$</td>
</tr>
</tbody>
</table>

a) Experiment with process reaction curve;

<table>
<thead>
<tr>
<th>Controller</th>
<th>$K_P$</th>
<th>$T/T_i$</th>
<th>$T_D/T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>$0.5 K_U$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PI</td>
<td>$K_U \left( 0.45 - 0.27 \frac{T}{T_U} \right)$</td>
<td>$0.54 \frac{K_U T}{K_p T_U}$</td>
<td>-</td>
</tr>
<tr>
<td>PID</td>
<td>$0.6 K_U \left( 1 - \frac{T}{T_U} \right)$</td>
<td>$1.2 \frac{K_U T}{K_p T_U}$</td>
<td>$\frac{3K_U T}{40 K_p T_U}$</td>
</tr>
</tbody>
</table>

b) Experiment with process on the stability limit;

Tab. 2.2. Relations for controller tuning developed by Takahashi.

Hang et al. (1991) addressed these problems and proposed refinements of the original ZN2 method using the normalized gain of the process:

$$\kappa = K_p K_U,$$

where $K_p$ is the steady state gain of the process.

Normalized dead time of the process is defined with:

$$\Theta = \frac{T_i}{T_p},$$

where $T_i$ is the apparent dead time, and $T_p$ is the dominant time constant of the process. For many processes values (2-28) and (2-29) are inversely proportional, so when the normalized gain of the
2. PID control

process is large, the normalized dead time is small. Strong dependence between values (2-28) and (2-29) on one hand, and overshoot in the response to set-point change of the control system on the other suggested (Hang et al., 1991):

1 – The use of set-point weighting in order to reduce excessive overshoot for processes with small dead-time (2.25<\(\kappa\)<15):

- \(F_D=0; F_I=1\);
- Weight \(F_P\) is adjusted according to:
  - Specification overshoot \(\sigma_m=10\%:\)
    \[F_P = \frac{15 - \kappa}{15 + \kappa};\]
  - Specification overshoot \(\sigma_m=20\%:\)
    \[F_P = \frac{36}{27 + 5\kappa}.

2 – Set-point weighting and modification of the relation for integral time constant \(T_I\) of the PID controller for processes with large dead-time (1.5<\(\kappa\)<2.25):

- Specification overshoot \(\sigma_m=20\%:\)
  \[F_D=0; F_I=1; F_P = \frac{8}{17} \left(\frac{4}{9} \kappa + 1\right);\]
  \[T_I = \frac{2}{9} \kappa T_U.\]

3 – New formulae for PI controller tuning for processes with 1.2<\(\kappa\)<15:

- Specification overshoot \(\sigma_m=10\%:\)
  \[K_p = K_U \frac{5}{6} \frac{12 + \kappa}{15 + 14\kappa};\]
  \[T_I = \frac{1}{5} \frac{T_U \left(\frac{4}{15} \left(\kappa + 1\right)\right)}{15 + 14\kappa}.\]

Most authors agree that ZN tuning rules can quickly give approximate values of the controller parameters, but require fine-tuning for better performance of the control system. The ZN2 experiment led to the idea of relay experiment, which is an identification technique customarily used in autotuning.

2.3.2 PID tuning based on integral criteria

Methods based on integral criteria for tuning PID controller involve searching for the minimum of the cost function \(I\) in the general form:

\[I = \int_0^\infty [e(t)]^n dt,\]

(2-30)

where \(e(t)\) is the error signal.
Optimum controller parameters and the minimum of the penalty function $I$ is found when its partial derivatives, in respect to controller parameters, equal zero. Equations for the calculations of the PID controller parameters are:

$$
\frac{\partial I}{\partial K_p} = 0, \quad \frac{\partial I}{\partial T_i} = 0, \quad \frac{\partial I}{\partial T_d} = 0.
$$

(2-31)

Generally, the set of equations (2-31) can not be solved analytically but numerically. Usually the choice of a particular function $f$ and exponent $n$ leads to formation of the following criteria (2-30):

- Integral Error (IE):
  $$f[e(t)] = e(t), \quad n=0;$$
- Integral Absolute Error (IAE):
  $$f[e(t)] = |e(t)|, \quad n=0;$$
- Integral Time multiplied Absolute Error (ITAE):
  $$f[e(t)] = |e(t)|, \quad n=1;$$
- Integral Squared Error (ISE):
  $$f[e(t)] = e(t)^2, \quad n=0;$$
- Integral Squared Time Error (ITSE):
  $$f[e(t)] = e(t)^2, \quad n=1;$$
- Integral Time square multiplied Squared Error (IT$^2$SE):
  $$f[e(t)] = e(t)^2, \quad n=2.$$

For some choices of the integrand in (2-30), it is possible to give physical interpretation of the cost integral $I$ (Persson, 1992). IE criterion is suitable for the static processes with non-oscillatory behavior. For example, this criterion is suitable for the control of a process with an output leading to a storage tank. An example of the appropriate use of IAE criterion is the octane control in gasoline production (Persson, 1992). Minimization of the total control energy in control systems leads to the ISE criterion. Time-weighted integrand functions, with $n>0$ in definition equation (2-30), given by the ITAE and ITSE criteria do not have physical significance (Seborg et al., 1989). These criteria penalize errors that persist for long periods of time. The ITAE criterion can be used for obtaining very conservative controller settings.

It is important to note that the error signal $e(t)$, used for optimization, can be a result of set-point change or of load disturbance. It is, therefore, possible to obtain two sets of parameters: one optimized for set-point change and the other for load disturbance.

A useful property of a PID controller is that the parameter $K_i$ (2-4) is directly related to IE for unit step load disturbance through (Åstrom and Hägglund, 1995):

$$I_{IE} = \int_0^\infty e(t) dt = \frac{1}{K_i} \frac{T_i}{K_p} = \frac{1}{K_i} \frac{T_i}{K_p}.$$  

(2-32)

For non-oscillatory processes, IE and IAE criteria have the same values, and for well-damped oscillatory processes IE can be regarded as an approximation of IAE. Criteria for the above mentioned processes can be minimized through maximization of the parameter $K_i$. 
2. PID control

<table>
<thead>
<tr>
<th>Parameters optimized for</th>
<th>$K_P$</th>
<th>$T_I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set-point change</td>
<td>$0.361 K_U$</td>
<td>$0.083 \left(1.935 \kappa + 1\right) T_U$</td>
</tr>
<tr>
<td>Load disturbance</td>
<td>$\frac{1.892 \kappa + 0.244}{3.249 \kappa + 2.097} K_U$</td>
<td>$\frac{0.706 \kappa - 0.227}{0.7229 \kappa + 1.2736} T_U$</td>
</tr>
</tbody>
</table>

Tab. 2.3.a Tuning rules for a PI controller.

<table>
<thead>
<tr>
<th>Parameters optimized for</th>
<th>$K_P$</th>
<th>$T_I$</th>
<th>$T_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set-point change</td>
<td>$0.509 K_U$</td>
<td>$0.051 \left(3.302 \kappa + 1\right) T_U$</td>
<td>$0.125 T_U$</td>
</tr>
<tr>
<td>Load disturbance</td>
<td>$\frac{4.434 \kappa - 0.966}{5.12 \kappa + 1.734} K_U$</td>
<td>$\frac{1.751 \kappa - 0.612}{3.776 \kappa + 1.388} T_U$</td>
<td>$0.144 T_U$</td>
</tr>
</tbody>
</table>

Tab. 2.3.b. Tuning rules for a PID controller.

Zhuang and Atherton (1993) performed ITSE optimizations for the FODT model. After that, they studied relations between controller parameters and the ultimate gain and ultimate period of the process. They found out that the parameters of PI and PID controllers can be expressed in terms of the ultimate gain $K_U$, ultimate period $T_U$, and normalized gain $\kappa$ of the process. These tuning rules are summarized in Table 2.3 and are prepared for an application in the autotuning procedures based on relay experiment.

2.3.3 Cohen-Coon tuning rules

The Cohen-Coon tuning method is based on the FODT model (2-2) with main design specification for quarter amplitude decay (QAD) ratio in response to load disturbance. The design objectives (Persson, 1992; Åstrom and Hågglund, 1995) were to maximize the gain and minimize the steady-state error and QAD for P and PD controllers. The parameters of the PI controller were obtained through minimization of the IE criteria and demand for QAD response. The parameters for PID controller were calculated with the same objectives as for the PI controller. The positioning of the additional controller pole was on the negative real axis. It is placed at the same distance from the origin as the two complex poles of the controller.

Relations for controller parameters in Table 2.4 are given in terms of parameters:

$$\alpha = \frac{K_T}{T_i},$$

$$\tau = \frac{T_i}{T_1 + T_i},$$

which are calculated from the parameters of the FODT model (2-2).
Tab. 2.4. Cohen-Coon controller tuning rules.

Åstrom and Hägglund (1995) observed that the Cohen-Coon tuning method suffers from a too small decay ratio, which results in low damping and high sensitivity of the closed-loop system. The method can be applied for on-line PID controller tuning, if the parameters of the FODT model are known.

### 2.3.4 PID tuning based on gain and phase margin specifications

PID tuning methods based on gain and phase margin specifications (GPM methods) involve solving definition equations for gain and phase margins, given by (2-20)-(2-23). Generally, these equations are non-linear and complicated for solving. Therefore, usual design methods based on these specifications are solved numerically or graphically, using Bode diagrams (Perić, 1998).

Ho et al. (1995a) analyzed the control system consisting of a PI controller and a FODT process with transfer function (2-2). When transfer functions of these dynamic elements are put into (2-20)-(2-23), arctan function appears in relations determining gain and phase crossover frequency:

\[
\frac{1}{2} \pi + \arctan \omega_c T_i - \arctan \omega_c T_i - \omega_c T_{ii} = 0, \quad (2-35)
\]

\[
A_K = K_p K_i = \omega_c T_i \sqrt{\frac{\omega_c^2 T_i^2 + 1}{\omega_c^2 T_i^2 + 1}}, \quad (2-36)
\]

\[
K_p K_i = \omega_c T_i \sqrt{\frac{\omega_c^2 T_i^2 + 1}{\omega_c^2 T_i^2 + 1}}, \quad (2-37)
\]

\[
\gamma = \frac{1}{2} \pi + \arctan \omega_c T_i - \arctan \omega_c T_i - \omega_c T_{ii}. \quad (2-38)
\]

In order to simplify the procedure of solving these non-linear equations, the following approximation was introduced:
The approximation and solving equations (2-35)-(2-38) for PI controller parameters gives:

\[ K_P = \frac{\omega_\pi}{A_1 K_1}, \]  
\[ T_I = \left(2\omega_\pi - \frac{4\omega_\pi^2 T_{nl} + 1}{T_i}\right)^{-1}, \]  
where \( \omega_\pi \) is calculated through:

\[ \omega_\pi = \frac{A_\gamma + \sqrt{A_\gamma(A_\gamma - 1)}}{(A_\gamma - 1)T_{nl}}. \]  

The parameters for PID controller are derived when the model of the process is second order with dead time (SODT) model. Equations (2-40) and (2-41) are used for parameters \( K_{PS} \) and \( T_{IS} \) in the series form of the PID controller and the derivative time constant \( T_{DS} \) is used to compensate for the smaller time constant.

It is important to note that it is not possible to achieve arbitrary gain and phase margin through the design procedure. The achievable specifications depend on the process characteristics and on the values of valid controller parameters. Reasonable values, recommended by Ho et al. (1995a) are \( A_\gamma = 4 \) and \( \gamma = 60^0 \). The method is valid for the FODT processes with the \( T_{nl}/T_1 \) ratio below 1.

Fung et al. (1998) studied exact solutions for the set of equations (2-35)-(2-38) describing control loops with a PI controller and a general linear process. They have proposed a graphical method for finding PI controller parameters. The method checks the solvability of the GPM method problem and is a good analytical tool.

Approximation (2-39) of \( \arctan(x) \) was successfully applied in a self-tuning PID controller (Ho et al., 1997). This approach to the design of PID controllers is, therefore, particularly useful in the context of adaptive control and autotuning.

### 2.3.5 Approximate pole placement method: Lambda tuning

Pole placement is an approach to control design which places all closed-loop poles in the desired places. Poles can be placed by employing a feedback controller or by using a state feedback controller which can be accompanied with an observer (Åstrom and Wittenmark, 1990). The methodology has been developed for processes of an arbitrary degree and can produce high-order controllers and observers. In order to use pole placement methodology for tuning PID controllers, only few of the closed-loop poles are placed in the desired places. One of the methods based on this principle is the Lambda tuning method.
Lambda tuning method is an approximate pole placement method which was originally devised by Dahlin and Highham to control processes with time delays (Panagopoulos et al., 1997). The controller is designed in such a fashion as to place a pole of the closed-loop system at the location $s=-\lambda$. The method was named after the parameter $\lambda$ which is the inverse of the desired closed-loop time constant $T_{CL}=1/\lambda$.

The method can be applied to obtain parameters of the PI controller for the control of a process modeled by the FODT model (2-2). The parameters can be calculated with or without cancellation of the process pole by the controller zero (Panagopoulos et al., 1997). In the derivation of the tuning rules, a complex exponential which models the process time delay is approximated with $e^{sT_t} \approx 1-sT_t$. Tuning rules are given in Table 2.5.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>PI controller with pole-zero cancellation</th>
<th>PI controller without pole-zero cancellation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_R$</td>
<td>$\frac{1}{K_1 T_{ti} + T_{CL}}$</td>
<td>$\frac{1}{K_1} \left( \frac{T_{ti} T_{CL} + 2T_{ti} T_{CL} - T_{CL}^2}{(T_{ti} + T_{CL})^2} \right)$</td>
</tr>
<tr>
<td>$T_I$</td>
<td>$T_i$</td>
<td>$\frac{T_{ti} T_{CL} + 2T_{ti} T_{CL} - T_{CL}^2}{T_{ti} + T_{CL}}$</td>
</tr>
</tbody>
</table>

Tab. 2.5. Lambda tuning rules for PI controller.

Cancellation of process poles by the controller zero results in the presence of uncontrollable modes in the closed-loop system, which may lead to poor performance if these modes are excited. This effect is particularly apparent in the control systems with the cancelled modes slower than the dominant modes (Åstrom and Hägglund, 1995; Hang, 1989). Besides, pole-zero cancellation applied in the systems with large signal non-linearities in the loop can produce large transient signals (Clark, 1988).

### 2.3.6 Approximate pole placement method: Dominant pole design

Dominant-Pole Design (DPD) methods find controller parameters which place the dominant poles of the closed-loop system in specified locations. In other words, those methods can be viewed as a translation of the problem of finding controller parameters into the problem of placing dominant poles in desired locations. The number of dominant poles to be placed depends on the number of free parameters, that is, on the number of controller parameters. A PI controller allows placement of two dominant poles, and a PID controller of three dominant poles. For these controllers, locations of the closed-loop dominant poles are parameterized with (Åstrom and Hägglund, 1995):

- PI controller:

\[ p_{1,2} = \omega_n \left( -\zeta \pm j\sqrt{1-\zeta^2} \right), \quad 0<\zeta<1; \quad (2-43) \]
2. PID control

– An additional pole location for PID controller:

\[ p_3 = -k_0\omega_n. \] (2-44)

Pole locations (2-43) and (2-44) in \( s \)-plane are depicted in Figure 2.9, where angle \( \alpha \) is determined through \( \alpha = \arccos(\zeta) \).

![Fig. 2.9. Locations of dominant poles for PI and PID controllers.](image)

Calculation of the closed-loop poles involves solving characteristic equation of the control system:

\[ 1 + G_R(s)G_P(s) = 0. \] (2-45)

If the required pole location is \( p_1 \), and the controller \( G_R(s) \) is a PI controller, the characteristic equation of the control system is:

\[ 1 + \left( \frac{1}{T_i} + \frac{1}{T_p} \right) G_P(p_1) = 0, \] (2-46)

and it should be solved for the controller parameters \( K_P \) and \( T_I \). To simplify calculation of the characteristic equation transfer function \( G_P(s) \), modeling the process, is parameterized with:

\[ G_P(s) \big|_{e^{i\omega_n e^{i(\pi - \alpha)}}} = G_P(\omega_n e^{i(\pi - \alpha)}) = a(\omega_n, \alpha) e^{i\phi(\omega_n, \alpha)} \] (2-47)

Parameter functions \( a(\omega_n, \alpha) \) and \( \phi(\omega_n, \alpha) \) can described as ‘frequency characteristics’ of the process on the ray with angle \( \alpha \) in \( s \)-plane, where \( a \) can be considered as ‘gain characteristic’ and \( \phi \) can be considered as ‘phase characteristic’. By putting parameterizations (2-43) and (2-47) in the characteristic equation (2-46), and solving it, following relations for PI controller parameters are obtained:

\[ K_P = -\frac{\sin(\phi(\omega_n, \alpha) + \alpha)}{a(\omega_n, \alpha) \sin(\alpha)}, \] (2-48)
Dominant pole locations of the control system are set to $p_1$ and $p_2$ with suitable choice of $\omega_n$ and $\zeta$, and through calculation of the PI controller parameters according to (2-48) and (2-49). Similar relations can be derived for parameters of the PID controller.

Direct choice of pole locations $p_i$ may lead to poorly damped control systems, which is avoided through a kind of optimization procedure. The optimization procedure is defined by specifications in time domain or frequency domain. Generally, value $\omega_n$ in DPD methods is used to achieve specified performance objective, and damping $\zeta$ is used to set the required robustness of the control system. Persson (1992) has proposed four methods, as follows:

- Two pole method (2PM) for a PI controller design; $\omega_n$ is chosen through maximization of the parameter $K_i$, which minimizes IE of the control system, and $\zeta$ is chosen through specification of the desired maximum sensitivity $M_s$. The usual $M_s$ values keep in the range 1.5-2.2, where values below $M_s=1.8$ produce non-oscillatory control systems;
- Modified PI controller design (MPI) takes into account that maximization of $K_i$ may at times produce a control system with too large an overshoot in response to load disturbance. This is why the parameter $\omega_n$ is increased until the value of the parameter $K_i$ drops to 0.8 $K_{i\text{max}}$;
- Three pole method (3PM) for a PID controller design; parameters $\omega_n$ and $\zeta$ are chosen as in the 2PM and the parameter $k_0$ is selected from values ranging between 0.5 and 1; the method poses moderate computational demands;
- Modified controller method (MCM) for a PID controller design uses the 2PM to calculate the PI controller parameters with maximum sensitivity $M_{s1}$. After that, the parameter $K_D$ is increased, and $\omega_n$ changes to keep $K_i$ maximal until the required $M_{s2}$ is obtained. This method is very computationally demanding and does not work well for resonant systems.

The position of a pole, which is defined by the D term lag (divisor $N$ in (2-6)), should not be a part of the optimization procedures, because it might acquire atypical values. The lag should be considered as a noise filter, and thus, as an implementation issue.

Persson (1992) observed that the maximum sensitivity $M_s$ of the control system, defined as a design objective, produces controllers with similar responses in the time domain for different processes. On the other hand, common measures for robustness – gain and phase margin, set as design objectives – can produce controllers with very dissimilar responses in the time domain for different processes (Fung et al., 1998). This makes the maximum sensitivity a very good choice for a design objective.

DPD methods rely on the knowledge of the process, that is, of the transfer function which models it, and are off-line in their nature. However, the design can be made in advance for some models and the results of the design can be used for on-line computation of controller parameters.
2.3.7 Magnitude optimum and symmetric optimum tuning methods

Magnitude optimum (MO) and symmetric optimum (SO) are two loop-shaping tuning methods extensively employed by the German company Siemens. The first step in the application of these methods is to determine appropriate transfer function which models the process. Once the transfer function is determined, the controller is able to shape the open-loop transfer function in a desired.

MO tuning method was devised with the objective to obtain a control system with a frequency as close to unity and as flat as possible for the maximum bandwidth (Umland and Safiuddin, 1990; Deur, 1999). Its mathematical expression states the requirements posed on the closed-loop transfer function $G_C(s)$:

$$G_C(0)=1, \quad \omega_n = \lim_{\omega \to 0} \frac{d^n(G_C(j\omega))}{d\omega^n} = 0,$$

for as many $n$ as possible. The desired open-loop transfer function is:

$$G_{O1}(s) = \frac{\omega_n^2}{s + 2\zeta\omega_n},$$

where $\zeta$ is the damping of the closed-loop system and $\omega_n$ determines the closed-loop dynamics, that is, the speed of response. For example, the PI controller is employed when it is possible to approximate the model of the process with the transfer function:

$$G_p(s) = \frac{K}{(1+T_1s)(1+T_2s)},$$

with $T_2<T_1$. By analyzing (2-50)-(2-53), and by setting $\zeta=0.707$, PI controller parameters are calculated (Åstrom and Hägglund, 1995)

$$K_p = \frac{T_1}{2K_T},$$

$$T_I = T_1,$$

with $\omega_n=0.707/T_2$. The dominant pole is cancelled by the PI controller zero, and the closed-loop dynamics are determined the smaller time constant $T_2$ of the process.

MO design method optimizes the closed-loop transfer function $G_C(s)$ between the reference and the output signal. It often cancels the process poles by the controller zeros, which can lead to poor performance of the control system in response to load disturbance (Umland and Safiuddin, 1990).

Vrančić (1997) used a multiple integration of the open-loop step response to identify the process model and optimized the PID controller parameters according to the MO principle. The method is termed Modulus Optimum Multiple Integration (MOMI). It is suitable for the proportional processes with a time delay (type 0 processes) which can be modeled with following transfer function:
2. PID control

\[ G_p(s) = K_{PR} \frac{1 + b_2 s + b_3 s^2 + \ldots + b_m s^m}{1 + a_1 s + a_2 s^3 + \ldots + a_n s^n} e^{-sL}. \]  

(2-56)

Integrals, used in the procedure, are defined as:

\[ I_1(t) = \int_0^t \left( K_{PR} - \frac{y(\tau)}{\Delta U} \right) d\tau, \quad A_1 = I_1(\infty), \]  

(2-57)

\[ I_2(t) = \int_0^t (A_1 - I_1(\tau)) d\tau, \quad A_2 = I_2(\infty), \]  

(2-58)

\[ I_3(t) = \int_0^t (A_2 - I_2(\tau)) d\tau, \quad A_3 = I_3(\infty), \ldots \]

where \( K_{PR} \) is the identified gain of the process, \( y(t) \) is the response to the applied step, \( \Delta U \) is the amplitude of the applied step, and \( A_i \) are integrated areas. Parameters of the PID controller are calculated in terms of the first five integrated areas (\( A_1 - A_5 \)) as (Vrančić, 1997):

\[ K_p = \frac{A_1}{2(A_1 A_2 - A_3 K_{PR} - T_D A_i^2)}, \]  

(2-59)

\[ T_i = \frac{A_2}{A_2 - T_D A_i}, \]  

(2-60)

\[ T_D = \frac{A_3 A_4 - A_2 A_5}{A_3 A_3 - A_1 A_5}. \]

In practice, significant identification errors occur due to load disturbances and process non-linearities. In order to minimize the effects of these errors, simplified PID tuning formulae are given in terms of the first three integrated areas (\( A_1 - A_3 \)) (Vrančić, 1997):

\[ K_p = \frac{0.5}{\frac{A_1}{T_i} - K_{PR}}, \]  

(2-61)

\[ T_i = \frac{A_2 - \sqrt{A_2^2 - 4 \rho A_1 A_3}}{2 \rho A_i}, \]  

(2-62)

\[ T_D = \rho T_i, \]  

(2-63)

where \( \rho \) is set to \( \rho = 0.2–0.25. \) The advantage of this method is that it allows calculation of the PID controller parameters merely by measuring the open-loop step response of the process. The procedure is suitable for on-line calculation of the PID controller parameters.

The objective of the SO method, which was originally proposed by Kessler (1958), is to obtain an open-loop transfer function of the below formula:

\[ G_{O2}(s) = \frac{a \omega_c^2}{s^2} \frac{(s + \frac{\omega_c}{a})}{(s + a \omega_c)}, \]  

(2-64)

where \( \omega_c \) is the gain crossover frequency and \( a \) is related to the phase margin of the control system through (Perić, 1979, 1998):
\[ \gamma = 2 \arctan \left( \frac{a - 1}{a + 1} \right), \quad (2-65) \]

or conversely through:

\[ a = \frac{1 + \sin \gamma}{\cos \gamma}. \quad (2-66) \]

The method maximizes the phase margin of the control system and leads to symmetrical phase and amplitude characteristics, as can be observed in Figure 2.10. The second multiplicant has the transfer function of a phase-lead network, which provides required phase uplifting at the frequency \( \omega_c \). Figure 2.10 shows the amplitude and gain characteristics of the control system with the open-loop transfer function equal to (2-64). In the example, the parameter \( a \) was set to 4.

![Gain and phase characteristics of a control system tuned according to symmetrical optimum.](image)

Fig. 2.10. Gain and phase characteristics of a control system tuned according to symmetrical optimum.

For example, if the process can be modeled with the transfer function \( G_2 \) (2-3):

\[ G_2(s) = \frac{K_2}{s(1 + T_z s)}, \]

it is suitable for the application of the SO tuning method. The procedure leads to a PI controller with the following settings (Perić, 1979):

\[ K_P = \frac{1}{a K_2 T_z}, \quad (2-67) \]

\[ T_I = a^2 T_z, \quad (2-68) \]
and $\omega_c = \frac{1}{a T_2}$. The common choice for the parameter $a$ is 2, which gives the phase margin of the control system $\gamma \approx 37^\circ$. As with the MO method, the speed of response of the control system obtained by the SO method is related to the time constant $T_2$ of the process. $T_2$ is considered a lower time constant, since the integrator in (2-3) can be regarded as the time constant with a very high value.

The SO method is designed to give a good response to load disturbance, but the response of the control system to set-point change has large overshoot. The overshoot is commonly reduced through the usage of a two-degree-of-freedom controller or with a prefilter (Åstrom and Hägglund, 1995).

The MO and SO tuning methods are widely used in the cascade control systems, especially to control motor drives (Perić, 1979; Deur, 1999). In such an application, the inner control loop uses a PI controller, designed according to the MO method, with a measured current signal as the feedback signal. The outer loop consists of a PI controller tuned using the SO method and a measured drive speed signal as the feedback signal. As noted previously, overshoot of the control system is reduced with the prefilter, which compensates the zeros of the closed-loop transfer function.

In both previous methods (MO and SO), the speed of the control system response is related to the smaller, non-dominant time constant of the process. Voda and Landau (1995) have therefore proposed a tuning method based on the knowledge of:

1 – the value of the sum of non-dominant time constants ($T_\Sigma$); and
2 – the ratio between the process gain and the dominant time constant. The method was inspired by the SO method and is termed Kessler-Landau-Voda (KLV) tuning method.

The region of frequency characteristics of the process where $T_\Sigma$ can be determined is the region where the phase characteristic equals $-135^\circ$ (Voda and Landau, 1995). The frequency where the phase characteristic equals $-135^\circ$ is denoted with $\omega_{-135}$, and the process gain at that frequency is denoted with $|G(\omega_{-135})|$. Using these process characteristics, the tuning rules are as follows:

– For the PI controller:

$$K_p = \frac{1}{3.5|G(\omega_{-135})|},$$

$$T_i = \frac{4.6}{\omega_{-135}},$$

– For the PID controller:

$$K_p = \frac{4 + \beta}{4} \frac{\beta}{2\sqrt{2}|G(\omega_{-135})|},$$
In equations (2-71) – (2-73) factor \( \beta \) defines the ‘acceleration’ of response of the control system using a PID controller in relation to the response of the control system using a PI controller. It is defined as the ratio between corresponding rise times:

\[
\beta = \frac{t_{r-PD}}{t_{r-PI}}.
\]

Recommended values for the factor \( \beta \) range between 1 and 2.

The KLV method is valid for the type I processes and for the type 0 processes in which dominant time constant is at least four times as high as the sum of non-dominant time constants. For type 0 processes that do not satisfy the required ratio between time constants, a control system tuned according to the KLV method is stable, but has poor dynamic performance. This tuning method is applicable for on-line usage and autotuning, because it was developed as a part of an autotuning procedure (Beasançon-Voda and Roux-Buisson, 1997).

### 2.3.8 PID tuning in the framework of Internal Model Control

Internal Model Control (IMC), thoroughly described by Morari and Zafiriou (1989), is a general design procedure for obtaining controllers that ‘optimally’ meet requirements for stability, performance, and robustness of the control system. Instead of choosing fixed control structure, and then finding optimal parameters for that structure, IMC postulates a model, states desirable control objectives, and then finds appropriate controller structure and parameters (Rivera et al., 1986). The concept of IMC is based on the simulation of the process model \( G_M(s) \) within the control structure. Figure 2.11 shows the arrangement of IMC. If the model of the process \( G_M(s) \) perfectly matches the process \( G_P(s) \), and load disturbance is not present, the output of the model cancels the output of the process annulling thus the feedback signal. In such case, the process is controlled in an open loop. The feedback signal and hence feedback control, exist only if there is the model mismatch or load disturbance \( Z(s) \). In other words, IMC is a control strategy in which controller \( C(s) \) is designed with the ease of open-loop design, while it retains the benefits of closed-loop control.
IMC design is made through following steps. The first is to factor the transfer function modeling the process:

\[ G_M(s) = G_M^+(s) \frac{G_M^{-}(s)}{1}, \]  

where \( G_M^+(s) \) contains only the left half plane poles and zeros, and \( G_M^{-}(s) \) contains all time delays and the right half plane zeros. After that, the controller \( C(s) \) is defined with:

\[ C(s) = (G_M^-(s))^{-1}G_F(s), \]  

where \( G_F(s) \) is a filter which guarantees that the controller \( C(s) \) is realizable. It also obtains the desired robustness and defines the closed-loop dynamics. The usual form of the filter is:

\[ G_F(s) = \frac{1}{(1+T_Fs)^n}, \]  

but other forms can also be used.

The IMC design procedure can be used to design conventional feedback controllers. Figure 2.11 shows the relation between a conventional feedback controller \( G_R(s) \) and IMC controller \( C(s) \), which may be expressed with the below formula:

\[ G_R(s) = \frac{C(s)}{1 - C(s)G_M(s)} \]  

or inversely:

\[ C(s) = \frac{G_R(s)}{1 + G_R(s)G_M(s)} \]  

For the particular choice of model \( G_M(s) \) and filter \( G_F(s) \), IMC design procedure leads to PID controller (Morari and Zafiriou, 1989). Performance and robustness trade-off of the obtained control system is handled through the value of the adjustable parameter \( T_F \), which determines the dominant time constant of the closed-loop system. For example, a low value of the parameter \( T_F \) produces fast response of the control system, but also results in low robustness margin.

The FODT model (2-2), can be used within the frame of IMC, but the part of the transfer function modeling dead time \( e^{-sT_d} \) has to be replaced with Pade approximations (Rivera et al., 1986). Furthermore, the exponent \( n \) in the denominator of the filter transfer function (2-77) is set to 1. Pade approximation of the zero order is:

\[ e^{-sT_d} \approx 1, \]
and leads to an IMC PI controller with the following parameters:

\[ K_p = \frac{T_i}{K_i T_F}, \quad (2-81) \]

\[ T_I = t_1, \quad (2-82) \]

and the recommended value for the filter time constant \( T_F > 1.7 \ t_1 \).

A PI controller that uses a better model of the dead time results in improved behavior of the control system. It is obtained when the parameters of the controller are set to (Rivera et al., 1986):

\[ K_p = \frac{2T_i + T_{i1}}{2K_i T_p}, \quad (2-83) \]

\[ T_l = T_i + \frac{T_{i1}}{2}, \quad (2-84) \]

with the recommended value for the filter time constant \( T_F > 1.7 \ t_1 \).

The first order Padé approximation:

\[ e^{-\frac{s}{t_{11}}} \approx \frac{1 - s T_{i1}/2}{1 + s T_{i1}/2}, \quad (2-85) \]

in the FODT model and IMC design lead to a PID controller with parameters (Rivera et al., 1986):

\[ K_p = \frac{2T_i + T_{i1}}{K_i (2T_F + T_{i1})}, \quad (2-86) \]

\[ T_l = T_i + \frac{T_{i1}}{2}, \quad (2-87) \]

\[ T_D = \frac{T_I T_{i1}}{2T_F + T_{i1}}, \quad (2-88) \]

and the recommended value for the filter time constant \( T_F > 0.8 \ t_1 \).

PD and PID controllers are obtained as result of IMC design when the process can be modeled with integration and a time constant (2-3). Parameters of these controllers are:

– PD controller

\[ K_p = \frac{1}{K_i T_F}, \quad (2-89) \]

\[ T_D = T_2; \quad (2-90) \]

– PID controller

\[ K_p = \frac{2T_F + T_2}{K_i T_F^2}, \quad (2-91) \]

\[ T_l = 2T_F + T_2, \quad (2-92) \]

\[ T_D = \frac{2T_F T_2}{2T_F + T_2}. \quad (2-93) \]
Seborg et al. (1989) concluded that the main advantages of IMC design are:

1. Model uncertainty is explicitly considered;
2. Trade-off between performance and robustness of the control system is clearly defined.

The principal drawback of the method is that the process poles are cancelled with controller zeros according to (2-76), which results in sluggish response to load disturbance, as described in subsection 2.3.5.

IMC tuning rules are expressed in terms of process model parameters and can be applied after the identification of the process model. Such models can be obtained as a part of an autotuning procedure.

2.4 Usage of PI controller in dead-time compensating controllers

One area of process control in which PID controllers fail to produce satisfactory results is the control of processes with large time delays. Time delay is considered large when its value exceeds the dominant time constant of the process. This type of dynamic behavior, termed time delay or dead time, is present in processes involving transport of materials such as rolling mills in metal industry and is a common result of composition analysis in chemical industry.

The assertion that PID control is inadequate for the control of processes with large dead time is based on two arguments:

1. the derivative action of the PID controller, needed for prediction, amplifies noise;
2. the open-loop gain has to be small rendering the performance of the control system poor.

The derivative action of the PID controller can be used to predict, for instance, future changes of the process output based on the error signal. This type of prediction is called linear extrapolation and is expressed in the PD part of the PID controller as:

\[ u(t) = K_p (e(t + T_d)) = K_p (e(t) + T_p \frac{de(t)}{dt}), \tag{2-94} \]

where \( e(t) \) is the error signal. In (2-94), the control signal is calculated using the predicted value of the error signal \( e(t+T_d) \) which is presumed to be linear in time. Linear extrapolation can not be applied to control systems with noise in the measurement signal, since differentiation amplifies noise. Accordingly, the derivative term of the PID controller is often switched off, and only the PI part of the controller is employed. However, linear extrapolation can be successfully applied in temperature control loops, where noise levels are not too high (Hägglund, 1996). The second problem is that a large time delay introduces a phase lag proportional to the value of the time delay. As a result, the open-loop gain has to be small in order to preserve stability. Small open-loop gain system causes slow response and renders the performance of the control system poor (Laughlin et al. 1987).
Another approach to control of processes with large time delays is to incorporate the model of the system as in Internal Model Control described in subsection 2.3.8. The Smith predictor, shown in Figure 2.12, applies that approach. The model of the process is divided into two parts: one for modeling the dynamic behavior $G_M(s)$ of the process and the other for modeling time delay $e^{-sT_{tm}}$. The transfer function of the control system with the Smith predictor is:

$$G_X(s) = \frac{Y(s)}{R(s)} = \frac{G_R(s)G_P(s)e^{-sT_p}}{1 + G_R(s)G_P(s) + G_R(s)G_P(s)e^{-sT_p} - G_R(s)G_M(s)e^{-sT_m}}. \tag{2-95}$$

The denominator of the transfer function (2-95) which is the characteristic equation of the closed-loop control system, plainly shows that when the modeling is exact ($G_M(s)=G_P(s)$) the two last terms are cancelled. In such case, the closed-loop transfer function becomes:

$$G_X(s) = \frac{Y(s)}{R(s)} = \frac{G_R(s)G_P(s)}{1 + G_R(s)G_P(s)} e^{-sT_p}, \tag{2-96}$$

and the controller $G_R(s)$ can be designed as if the process did not contain dead time. In other words, the controller $G_R(s)$ can be designed just for the part of the process modeled by the transfer function $G_M(s)$. The cancellation of dead-time influence on the dynamic behavior of the control system is characteristic of dead-time compensating controllers.

The main drawback of the Smith predictor is that the performance and the stability of the control system are very sensitive to inaccurate modeling of the process, especially to the inaccurate modeling of dead time. Furthermore, it is very difficult to tune the Smith predictor because it involves precise identification of the process model. Schneider (1988) gave a good comparison of control performance obtained by pole-cancellation in the PI controller and the Smith predictor.
One important advantage of PID control over other control strategies, including the Smith predictor, is that operators are familiar with the tuning procedures for PID controllers. These involve finding of only three parameters. In comparison, the tuning of the Smith predictor involves identification of a suitable process model, and then tuning of the controller. The Smith predictor, which is based on the FODT model and on the PI controller, has five parameters and is very complicated to tune and operate. In order to simplify the tuning procedure of the Smith predictor, Hägglund (1996) has proposed a restriction of the choice of the PI controller and FODT model parameters. This type of the Smith predictor is called the predictive PI (PPI) controller. Figure 2.13 shows the structure of the PPI controller with the filter $F(s)$ set to 1.

In the PPI controller, the parameters of the PI controller are related to the FODT model parameters as follows:

$$K_R = \frac{\kappa}{K_M}, \quad (2-97)$$

$$T_t = \frac{1}{\tau} T_M, \quad (2-98)$$

where $\kappa$ and $\tau$ are calculated from the desired performance of the closed-loop control system. Based on these parameters, the characteristic equation of the PPI is:

$$s^2 + \frac{1+\kappa}{T_M} s + \frac{\kappa\tau}{T_M^2} = 0. \quad (2-99)$$

Parameter $\tau$ determines the speed of response (rise time) of the closed-loop system. When $\tau$ is larger than 1, the closed-loop system is slower, and when $\tau$ is smaller than 1 the closed-loop system responds more quickly than a system operating in the open loop. When $\kappa$ and $\tau$ are set to 1, and when the process exactly matches the FODT model used in the predictor, the closed-loop system has a double pole at $s=-1/T_M$ in the $s$-plane. Such choice of closed-loop pole positions ensures aperiodic response of the control system.

Hägglund (1996) compared the performance (IAE) of the classical PI and the PPI controller in presence of a constant load disturbance. A simple analysis has shown that the relation between the IAE values corresponding to each controller, is:
The equation (2-100) implies that the PPI IAE is 50% lower than the IAE of the classical PI controller for systems with large time delay, i.e. when $T_{tm} \gg T_M$. Furthermore, Normey-Rico et al. (1997) stated that the PPI controller could be interpreted as equivalent to the General Predictive Controller (GPC), possessing all its advantages.

The PPI controller solves the problem of operational complexity of the Smith predictor since it decreases the number of controller parameters, but it does not solve the problem of high sensitivity of the Smith predictor due to the inaccurate modeling of the process. In order to increase the robustness of the PPI controller Normey-Rico et al. (1997) introduced a filter in the PPI structure. Figure 2.13 shows such filtered PPI (FPPI) controller. In order to preserve the simple structure of the PPI filter, $F(s)$ is chosen to be the first-order lag with static gain equal to one:

$$F(s) = \frac{1}{1 + T_F s}.$$  \hspace{1cm} (2-101)

In Normey-Rico et al. (1997) the robustness of the FPPI controller is analyzed with respect to different values of the filter time constant $T_F$. The result is the recommendation for its value:

$$T_F = \frac{T_{tm}}{2}.$$  \hspace{1cm} (2-102)

The described approach to control of processes with large time delays is not suitable for control of processes that have astatic characteristics (processes with type above 0), because constant load disturbance produces a steady-state error. These processes have to be controlled with modified Smith predictor structures, suggested in works of Watanabe and Ito (1981), and Åstrom et al. (1994).

The described PID controllers are basic components of many control systems operating in industry. It is worthwhile to automate some of the described tuning procedures. The next step in the development of autotuning PID controller is to review several variants of the algorithm. Such review is given in the third chapter.
3 AUTOTUNING PID CONTROLLER

Autotuning was developed as a technique that addressed many practical problems of control engineering, which were not adequately solved by the existing techniques. This chapter will give a brief review of several techniques of adaptive control and will focus on the basic autotuning algorithm and some modifications thereof. The chapter concludes with several examples of adaptive usage of PID controller.

3.1 An overview of adaptive control

Control systems are designed to minimize the effects of process variations and environment influences on the quality of process control. Sometimes these variations and influences are significant to the extent that the conventional linear controllers with constant parameters are unable to control the processes successfully. The main tasks of the controller in such situations are to retain stability and maintain the desired performance of the control system. Two control engineering approaches have been devised to achieve these demanding goals: robust control and adaptive control. In robust control, the controller is fixed, which satisfies requirements for all models belonging to a certain class. If this class of models encompasses all expected process and environment variations, the designed controller solves the problem. On the other hand, an adaptive controller adjusts itself to changes in the course of operation. It recognizes changes in the process and in the environment, and adapts the structure and the parameters of the controller accordingly. Adaptation mechanism further automates the control of the process by performing tasks usually made by the control engineer and thus extends the idea of the feedback loop. It acts in the adaptation loop of an adaptive control system, as shown in Figure 3.1.

Research of control paradigms resulted in the development of the following adaptive control techniques (Åström, 1996):

- **Gain scheduling** is an adaptive control technique in which controller parameters change depending on some measured (scheduling) variables, which are directly related to the operating conditions. Sets of controller parameters are stored in a table in a table created at the time of design, and each set is used in one part of the operating range. The controller parameter switch is made through bumpless transfer techniques. It is important to note that the adaptation is not direct, since scheduling variables may not be related to the performance of the control system.

- **Model reference adaptive systems** (MRAS) use the model of the system that provides the desired response to the applied reference signal. The desired response signal is subtracted from the actual response signal and the obtained signal is used in a gradient scheme of parameter
adjustment. This adaptive technique directly updates controller parameters, without a separate design stage, that is, it belongs to the group of direct adaptive techniques.

- **Self-tuning regulator** (STR) is an indirect adaptive technique. It identifies the model of the process, and designs the controller on the basis of the identified model. The procedure is repeated continuously. The identification is usually recursive, employing different identification procedures such as least squares methods and maximum likelihood methods. The application of identification procedures requires special attention, because the identification is performed in a closed loop. One of the problems encountered in this scheme is that persistent excitation signal is required for successful identification, which may be unacceptable for the process. Various design techniques are employed in the design stage of self-tuning controllers such as pole placement, LQG, and $H^\infty$. The self-tuning technique has been applied the design of the PID controller (Radke and Isermann, 1987; Ho et al., 1997).

- **Autotuning** (Åstrom and Hägglund, 1984; Perić et al., 1997) is an adaptive technique that is put in motion by the user, or colloquially by a ‘push button’. In other words, it does not operate continuously in the adaptation loop, but only when there is need for tuning or re-tuning. This technique repeats the steps taken by a control engineer during design of the controller. Firstly, a simple experiment is performed, which determines some characteristics of the process. After that, the controller parameters are calculated on the basis of the obtained data and the designed controller is started. This feature of modern controllers is particularly useful during commissioning of control systems. Furthermore, autotuning can be used to build the table of controller parameters for gain scheduling.

![Fig. 3.1. General structure of an adaptive control system.](image)

The above mentioned adaptation techniques should not be viewed as competitive, but rather as a set of different tools used for solving different tasks in control engineering. The structure of the adaptation system, as shown in Figure 3.1, is very general and includes only the main features of the described adaptation techniques.
The above classification of adaptive techniques, in fact, retraces the steps of scientific development of adaptive control. Most of these techniques are model-based and require knowledge of the process model. Model-based techniques give good control performance for set-point changes, but poorly handle situations in which the main excitation signals in the system are load disturbances. On the other hand, many industrial adaptive controllers are rule-based (Hägglund and Åstrom, 1997). Rule-based controllers do not require the process model, but perform adjustments using a predefined set of rules. The advantage of these algorithms is that set-point changes and load disturbances are treated in the same way. This type of adaptive control works well for isolated set-point changes and load disturbances, but problems arise when two consecutive changes occur.

Many model-based adaptive algorithms have been proposed in scientific literature, but the application of these algorithms in process industry is limited because of their sensitivity to operational conditions in the industrial environment. To resolve these problems Hägglund and Åstrom (1997) suggested the use of a control shell that would supervise the execution of adaptive algorithms, and would make them more robust. A shell such as this requires the following functions:

- **Initialization and pre-tuning** is required so that the adaptive controller is started with an appropriate set of parameters. Parameters such as time scale and apparent dead time are needed for setting suitable sampling time and choosing measurement filter time constants. These parameters can be obtained from the operator or from a pre-tuning identification experiment. It is also possible that these parameters are present in the adaptive controller from the previous runs of the algorithm on the same process. As observed by Hägglund and Åstrom (1997), the initialization procedure should be divided in modules which may be used in different parts of an adaptive algorithm.

- **Excitation detection** determines whether the reference signal is sufficiently persistent to set to motion identification and adaptation. The detection uses the filtered controller output signal \( u(t) \) and the process output signal \( y(t) \). The procedure is described in subsection 3.3.3.

- **Load disturbance detection** is the part of the detection procedure that uses the filtered controller output signal \( u(t) \) and the process output signal \( y(t) \) to detect load disturbance and stop the process of identification and adaptation. The aim is to avoid erroneous identification of the process model.

- **Oscillation detection** has the purpose to halt adaptation of the controller when oscillations in the control system occur and to alarm the operator. Hägglund (1995) proposed an algorithm which enables detection of load disturbance using IAE and measures the number of load disturbance occurrences in a defined time period. If the number of occurrences exceeds a certain limit, the algorithm sets off the alarm. The feature is particularly useful for coupled control loops where one oscillating control loop may give rise to oscillations in the whole control system.

- **Signal saturation** has to be detected in order to employ proper anti-windup scheme (described in section 2.2) and to stop identification and adaptation.
- **Mode transitions** have to be implemented using bumpless transfer techniques. Modes of controller operation can include the manual, autotuning, gain scheduling, and adaptive mode. It is of utmost importance to handle different transitions safely and to avoid unpredictable transitions or undefined states of the controller.

- **Noise detection** in the pre-tune phase is required for appropriate design of filters for noise filtering. Frequency characteristics and the amplitude of noise signals in the control loop should be measured. Noise bandwidth, measured in Hertz, is approximately equal to the average rate of crossings over steady-state value per second (Åstrom and Hägglund, 1995).

### 3.2 Basic autotuning algorithm

The autotuning algorithm, which was originally proposed in the well-known paper by Åstrom and Hägglund (1984), is an automated version of the second Ziegler-Nichols experiment (described in subsection 2.3.1). The algorithm is the result of a need for a simple identification which would provide data for a PID controller design. The difference is that the autotuning algorithm performs these tasks automatically. Instead of using proportional controller to control the process and to reach limit cycle oscillations, the process is controlled by a relay. Such usage of relay is possible since most industrial processes, having phase lag of at least $-180^\circ$ at high frequencies, will enter limit cycle oscillations under the relay control. The algorithm measures the amplitude and the frequency of the limit cycle, which is the information used for the design of PID controller. Åstrom and Hägglund (1984) proposed a PID controller design based on amplitude and phase margin specifications. However, most PID design techniques described in chapter 2 can be used in autotuning with some modifications of the algorithm. Figure 3.2 shows the setup for the basic autotuning algorithm. The relay experiment is on when the switch $S$ is in the position 0. After the completion of the experiment, PID controller parameters are calculated, switch $S$ is set to position 1, and the designed PID controller is started.

---

**Fig. 3.2. Setup for basic autotuning algorithm.**
The relay experiment is analyzed through the description of the closed-loop system consisting of a relay and a linear process. Since relay is a non-linear element, its behavior is characterized by a set of non-linear equations. Figure 3.3 shows the input-output relation of a relay element. The principle of harmonic linearization (Kuljača et al., 1997) serves to simplify the analysis. The principle assumes that the process has low-pass characteristics and that higher harmonics are filtered out. Moreover, only oscillations of the basic harmonic are considered, which leads to the concept of describing function. Describing function is a relation that gives amplitude gain and phase shift of the basic harmonic, which are introduced into the control loop by a non-linear element. The input variable of a describing function is amplitude $A$ of limit cycle oscillations. The form of the function is calculated similarly to the calculation of the Fourier coefficients (Kuljača et al., 1997). The describing function of a relay element without hysteresis ($\varepsilon = 0$) is:

$$N(A) = \frac{4d}{\pi A}.$$

The equation (3-1) does not contain the imaginary part, so that phase shift is not introduced in the control loop by the relay. This is the characteristic of describing functions of all non-linear elements which are single-valued. The describing function of a relay with hysteresis is:

$$N(A) = \frac{4d}{\pi A^2} \sqrt{A^2 - \varepsilon^2} - j \frac{4d\varepsilon}{\pi A^2},$$

where the introduced phase shift depends on the size of hysteresis and is expressed as the imaginary part of the equation.

During the limit cycle oscillations, the output signal is propagated through all elements of the closed-loop system. The propagation of oscillations is described by the equation of harmonic equilibrium:

$$y(j\omega) = y(j\omega)N(A)G_p(j\omega).$$

With the oscillation signal $y(j\omega) \neq 0$, harmonic equilibrium (3-3) leads to the characteristic equation of the system containing a linear and a non-linear element:

$$0 = 1 + N(A)G_p(j\omega).$$
From (3-4) it follows that the amplitude and the frequency of limit cycle oscillations are defined at the point $A_\varepsilon$ on the Nyquist curve (Figure 3.4). The point $A_\varepsilon$ is determined as the intersection of the Nyquist curve and the negative inverse of the describing function:

$$G_p(j\omega) = \frac{-1}{N(A)}.$$  \hspace{1cm} (3-5)

The negative inverse of the describing function of a relay with hysteresis follows from (3-2):

$$\frac{-1}{N(A)} = -\pi \frac{\sqrt{\varepsilon^2 - \varepsilon'^2}}{4d} - j \frac{\pi \varepsilon}{4d}. \hspace{1cm} (3-6)$$

Hence, this experiment procedure identifies the point $A_\varepsilon$ on the Nyquist curve $G(j\omega)$ by measuring the period and the amplitude of limit cycle oscillations. Relay hysteresis and amplitude of limit cycle oscillations define the angle $\varphi_\varepsilon$ between the negative real axis and the line, on which the $A_\varepsilon$ point is positioned:

$$\varphi_\varepsilon = \arctan \frac{\varepsilon}{\sqrt{\varepsilon^2 - \varepsilon'^2}}. \hspace{1cm} (3-7)$$

When the relay element does not contain hysteresis, the experiment procedure identifies the critical point $A_-$ of the process which is on the negative real axis in the Nyquist plane (Figure 3.4). Oscillations of the system in this point define critical gain:

$$K_c = \frac{4d}{\pi A}, \hspace{1cm} (3-8)$$

where $A$ is the measured amplitude of oscillations. The frequency of oscillations in the point $A_-$ is called critical frequency of the process. Figure 3.4 shows that this frequency is identical to the phase crossover frequency $\omega_{\pi}$, defined by (2-21). The terms will be used interchangeably.

![Fig. 3.4. Negative inverse of the describing function and Nyquist curve of the process in the Nyquist plane.](image)

The control of oscillations is performed using the iterative scheme for the calculation of the output amplitude $d$ of the relay element:
\[ d_{n+1} = d_n - (A_n - A_D) \frac{d_n - d_{n-1}}{A_n - A_{n-1}}, \]  

(3-9)

where \( A_D \) is desired amplitude value, \( n \) the iteration number, \( A_i \) the measured amplitude in the \( i^{th} \) iteration, and \( d_i \) is output amplitude of relay element in the \( i^{th} \) iteration. Several oscillations of the output signal constitute one iteration in this scheme. The iterative scheme has quadratic convergence rate near the solution, so it finds needed output amplitude of the relay element after only few iterations.

PID controller can be interpreted as a dynamic element which moves one point of the Nyquist curve to the arbitrary position (Åstrom and Hägglund, 1995). The proportional gain of the controller is used to move that point in the radial direction from the origin, while derivative and integral gains are used to move the point in the orthogonal direction. These directions are given in respect to the shape of the Nyquist curve. The relay experiment identifies the critical point (or point \( A_\varepsilon \)) of the Nyquist curve, in order for the PID controller to move it in the desired direction. The desired phase margin \( \gamma \) may serve as the design specification.

When the critical point \( A_\varepsilon \) is identified by the relay experiment, it is moved to the point on the unity gain circle in the Nyquist plane, with the angle equal to \( -\pi + \gamma \). The open-loop transfer function of the control system is:

\[ G_O(s) = K_p \left( 1 + \frac{1}{T_i s} + T_D s \right) G_p(s). \]  

(3-10)

The required phase advancement at the crossover frequency \( \omega_c \) equals the desired phase margin \( \gamma \) and is introduced into the control loop by the designed PID controller. Consequently, the controller time constants have to satisfy the following relation:

\[ \omega_c T_D = \frac{1}{\omega_\pi T_i} = \tan \gamma. \]  

(3-11)

Similar to Ziegler-Nichols tuning rules, the proportional relation is established between integral and derivative time constants:

\[ T_i = c T_D \]  

(3-12)

According to Ziegler-Nichols tuning rules described in subsection 2.3.1, the constant \( c \) is set to 4. Equations (3-11) and (3-12) are used to calculate the value of the derivative time constant:

\[ T_D = \frac{\tan \gamma + \sqrt{4 + \tan^2 \gamma}}{2 \omega_\pi}. \]  

(3-13)

From the requirement that the gain of the open-loop system at critical frequency \( \omega_\pi \) equals unity:

\[ |G_O(j \omega_\pi)| = \left| \frac{K_p}{K_c} (1 + j \tan \gamma) \right| = 1, \]  

(3-14)

the gain of PID controller is calculated according to:

\[ K_p = K_c \cos \gamma. \]  

(3-15)

The same PID design procedure can be used when the \( A_\varepsilon \) point on the Nyquist curve is identified, but the angle \( \phi_\varepsilon \) has to be much smaller than the angle defined by the desired phase margin \( \gamma \). If
that constraint is not satisfied, the angle $\varphi_e$ has to be incorporated in equations (Perić et al., 1997) for calculating PID controller parameters. Therefore, equations (3-11),(3-13), and (3-15) are modified into:

$$\omega_e T_D - \frac{1}{\omega_e T_I} = \tan(\gamma - \varphi_e),$$

$$T_D = \frac{\tan(\gamma - \varphi_e) + \frac{4}{\omega_c} + \tan^2(\gamma - \varphi_e)}{2\omega_e},$$

$$K_p = \frac{4d}{\pi A} \cos(\gamma - \varphi_e),$$

where $\omega_e$ is the frequency and $A$ is the amplitude of limit cycle oscillations.

In addition, the desired amplitude margin $A_r$ of the control system can be used as a specification for the design of the PID controller. Parameters of the PID controller are calculated using the following relations (Åstrom and Hägglund, 1984):

$$K_p = \frac{K_C}{A_r},$$

$$T_D = \frac{1}{\omega_c^2 T_I},$$

where integral time constant $T_I$ can be chosen freely. The proportional gain of the PID controller, set according to (3-19), guarantees the required amplitude margin $A_r$, and the choice (3-20) of derivative time constant ensures that the phase of the control system is not changed at the critical frequency $\omega_c$.

Usually, the relay hysteresis is used to suppress the effect of noise on limit cycle oscillations. The hysteresis should exceed the noise amplitude, if oscillations are to result from the oscillating process dynamics, and not from noise.

The described autotuning algorithm has several features that make it very practical:

- The identification experiment is performed in the closed loop, which allows control of amplitude of limit cycle oscillations. The ZN2 experiment, however, renders impossible the control of the amplitude of limit cycle oscillations which can attain unacceptable values.
- The basic operations of the algorithm are not complex and can be easily programmed in a microprocessor-based control equipment (e.g. measurement of amplitude and frequency of limit cycle oscillations).
- The algorithm makes commissioning of the control system simpler and does not require expert knowledge.
- Regular re-tuning of the PID controller through autotuning improves long-term performance of the control system.
• It can be used as a start-up algorithm for more advanced controllers such as the self-tuning controller. Furthermore, a PID controller obtained by autotuning can be used as a robust alternative for these controllers. It may be employed when adaptation has to be stopped and provide more robust control.

While having these desirable properties, basic autotuning algorithm also has many unresolved issues and drawbacks that have to be properly addressed.

The relay experiment has to be performed in the operating point because of the validity of the PID controller parameters that are obtained through autotuning. Often, these parameters are valid only in a limited operating range. Consequently, for successful application of an autotuning algorithm, it is necessary to devise a scheme for reaching the operating point. It can within the initialization procedure of the algorithm’s supervision shell, through manual control, or through some kind of automatic procedure.

Since an autotuning algorithm should not require operator with expert knowledge, only simple parameters should be used as its input parameters. Furthermore, there should be as few input parameters as possible. These parameters should include approximate time scale and the desired amplitude of limit cycle oscillations.

A problem may arise due to the use of relay element during identification experiment. The analysis of the relay experiment in this section presumes the existence and uniqueness of limit cycle oscillations, which cannot be guaranteed for all processes. Some related issues have been addressed in the paper by Åström and Hägglund (1984). For example, a process that has only two integrators controlled by a relay without hysteresis would enter limit cycle oscillations with an arbitrary period. Such process behavior during the relay experiment is undesirable and renders the autotuning algorithm inapplicable. The problem of existence and uniqueness of limit cycle oscillations has been thoroughly discussed in the Ph.D. thesis by Johansson (1997). The precise characterization of the class of the processes that converge to a limit cycle oscillation with unique parameters under relay control is still an open problem of control engineering.

In the basic autotuning algorithm, PID controller is designed to apply a simple rule for moving the ultimate point of the process to a specified position. Such procedure gives acceptable results for many processes, but it is possible to achieve better performance if the employed design rule is optimized for the controlled process. Even better results can be obtained if the relay experiment is suited for the controlled process. Many modifications of the basic autotuning algorithm, implementing the ideas of optimization of autotuning according to the type of the controlled process, have been proposed in literature (Ho et al., 1996a; Voda and Landau, 1995; Majhi and Atherton, 1998). However, these modifications of the basic autotuning algorithm require a priori knowledge of the type of the controlled process.
3.3 Modifications of the basic autotuning algorithm

Modifications of the basic autotuning algorithm were introduced in order to resolve its drawbacks, to widen the area of application, and to make it more robust and accurate.

Many proposed modifications use the dynamic element $W(s)$ connected in series with the relay when performing the relay experiment. Figure 3.5 shows such autotuning setup.

![Fig. 3.5. Setup for a modified autotuning algorithm.](image)

Ho et al. (1996a) connected the derivative element $W(s) = s$ to the output of the relay and used it for autotuning of type 1 processes. This modification cancels the integrator in the process with the derivative element and identifies the point on the Nyquist curve with phase lag of $270^\circ$. The identified point is the intersection of the Nyquist curve of the process and the positive part of the imaginary axis in the Nyquist plane. The use of the derivative element shortens the time needed for one oscillation, and hence the time required to complete the experiment. The simulation and the experimental results by Ho et al. (1996a) showed that the time needed for the relay experiment was halved.

3.3.1 Relay experiment with adjustable dead time

In the description of the basic autotuning algorithm, it was mentioned that the position of the identified point $A_\epsilon$ depended on the size of relay hysteresis $\epsilon$. The adjustable size of hysteresis allows identification of different points on the Nyquist curve, with the phase ranging from $-180^\circ$ to $-90^\circ$ (the points of the Nyquist curve positioned in the third quadrant of the Nyquist plane). An equivalent effect can be achieved by the insertion of the element with variable dead time after relay element without hysteresis into the control loop, that is, by choosing $W(s) = e^{-sT}$. Beasançon-Voda and Roux-Buisson (1997) used this modification of the basic autotuning algorithm to identify the point on the Nyquist curve of the process with the phase equal to $-135^\circ$. The identification consists of measuring the frequency where the phase equals $-135^\circ$ and measuring the gain of the process at that frequency. The knowledge of the process gain and frequency is used to design the PID controller. The design procedure was
developed by Voda and Landau (1995) and is described in subsection 2.3.7. The process thus identified through the relay experiment becomes:

\[ G'(j\omega) = G(j\omega)e^{-sT_tv}, \]  
(3-21)

and is called the equivalent process. The phase lag of the equivalent process \( \phi \), as shown in Figure 3.6., equals the sum of the process phase \( \phi_0 \) and the phase lag introduced by the dead time element \( \phi_v \):

\[ \phi = \phi_0 + \phi_v = \phi_0 - \omega T_v. \]  
(3-22)

It is important to note that points \( A \) and \( A_e \) are positioned on the same frequency of corresponding Nyquist curves. Furthermore, these points correspond to the same gain, since the dead time element has gain equal to the unity at all frequencies. It is, therefore, possible to obtain the amplitude and the frequency of limit cycle oscillations in the point \( A_e \) on the Nyquist curve by measuring these values in the point \( A \) on the Nyquist curve of equivalent process.

\[ T_{tv} = \frac{\phi_d - \phi}{\omega}, \]  
(3-23)

where \( \phi_d \) is the desired phase (in this procedure equal to -135°) and \( \phi \) is the phase of the identified point of the equivalent process (in this procedure equal to -180°). A modification of the previous relation leads to iterative equation for finding the needed dead time (Beasançon-Voda and Roux-Buisson, 1997):
\[ T_{n+1} = \frac{\varphi_d - \varphi}{\omega} = T_{n1} + c\left( \frac{\varphi_d - \varphi}{\omega} - T_{n1} \right) = T_{n1} + c\left( \frac{\pi}{4\omega} - T_{n1} \right), \] (3-24)

where \( i \) denotes the number of iterations, \( \omega \) denotes the measured frequency of oscillations in an iteration, and \( c \) is the weight term with the range \( 0 < c \leq 1 \). This iterative procedure should coincide with the iterative procedure of adjusting the output amplitude of the relay element \( d \) defined by (3-9).

The equivalence between the relay experiment with adjustable hysteresis and the relay experiment with adjustable dead time can be expressed by equaling the angle \( \varphi_e \) (Fig. 3.4.) and angle \( \varphi_L \) (Fig. 3.6.). This equivalence can be used to establish relations between the ‘equivalent’ values of hysteresis and dead time:

\[ \varepsilon = A \sin \varphi_e = A \sin \omega T_{n1}, \] (3-25)

and:

\[ T_{n1} = \frac{\varphi_e}{\omega} = \frac{1}{\omega} \arctan \frac{\varepsilon}{\sqrt{A^2 + \varepsilon^2}} = \frac{1}{\omega} \arcsin \frac{\varepsilon}{A}. \] (3-26)

The main advantage of adjusting dead time instead of hysteresis in the relay experiment that it involves simpler calculation and has better accuracy (Beasançon-Voda and Roux-Buisson, 1997).

A similar autotuning approach has been proposed in the work by Leva (1993). Variable dead time element is used to find a suitable point on the Nyquist curve to move it to the unit circle by employing the appropriate PI/PID controller. The designed controller should ensure the desired phase margin, used as the design specification. The suitable point is defined through the amplitude and phase constraints. The element consisting of dead time and time lag defines the additional dynamics inserted into the closed loop during the relay experiment:

\[ W(s) = \frac{e^{-s T_{e}}}{1 + T_f s}, \] (3-27)

where \( T_f \) is calculated in relation to sampling time \( T \).

At the beginning of the experiment, the additional dead time \( T_{n1} \) is set to zero and is gradually increased. A test performed after each iteration seeks to determine whether the point fulfills the prescribed conditions. A similar check is run for the consistency of the possible PI/PID controller. Unless the conditions are met and the controller started, the relay experiment is repeated with longer additional dead time.

The advantages of this approach are that it is not computationally demanding and requires little a-priori knowledge about the process. One of the main reasons why the method with variable dead time performs better than the method with variable hysteresis is that the former can not be used for processes which have the phase above \(-\pi/2\) in whole frequency range. Furthermore, Leva (1993) suggested that this variant of autotuning algorithm should not be applied to processes that are weakly damped.
3.3.2 Estimation of mathematical model parameters using relay experiment

The relay experiment can be used to estimate parameters of a known mathematical model. The relay experiment measures two values: the amplitude and the frequency of limit cycle oscillations and is able to directly identify models which include the two parameters. This study considers two process models: the FODT model (2.2) and the integrator with a lag (2.3) and investigates the possibility to estimate model parameters using the relay experiment.

The analysis examines the behavior of the process during relay experiment using the characteristic equation of the closed loop (3-4). The FODT model is inserted as $G_p(s)$ into the equation (3-4), and the gain and phase components are separated. The phase equation gives (Perić et al., 1997):

$$\omega T_{t_l} + \arctan \omega T_1 = -\pi + \varphi_e,$$  \hspace{1cm} (3-28)

and the gain equation leads to:

$$K_1 = \frac{\pi A}{4d} \sqrt{1 + \omega^2 T_1^2}. \hspace{1cm} (3-29)$$

It is not possible to solve the previous set of equations and to find parameters of the process model, since the two equations have three unknowns ($T_{t_l}$, $T_1$, $K_1$). This is why additional data are required to identify all parameters of the FODT model.

The additional data can be obtained by using a non-symmetrical relay element in the relay experiment (Ho et al., 1996a). The non-symmetrical relay element has different amplitudes ($d_+$ and $d_-$) for positive and negative values of the output signal, as depicted in Figure 3.7.

Intermediate values, used for a simpler calculation of the FODT model parameters are:

$$x_1 = -2e \frac{d_- - d_+}{\pi A}, \hspace{1cm} (3-30)$$

$$x_2 = \frac{d_- - d_+}{\pi A} \left[ \sqrt{A^2 - (y_o - e)^2} + \sqrt{A^2 - (y_o + e)^2} \right]. \hspace{1cm} (3-31)$$

and these make possible the calculation of the model parameters as follows (Ho et al., 1996a):
\[ K_1 = \frac{y_0}{u_0}, \]  
(3-32)

\[ T_i = \frac{1}{\omega^4} \sqrt{K_1^2 (x_1^2 + x_2^2) - A^2}, \]  
(3-33)

\[ T_i = \frac{1}{\omega} \left( \pi - \arctan \frac{x_1}{x_2} + \arctan \frac{x_1}{x_2} \right), \]  
(3-34)

where \( y_0 \) and \( u_0 \) are DC components determined from the areas under signals \( y(t) \) and \( u(t) \) during limit cycle oscillations, and \( \omega \) and \( A \) are the frequency and the amplitude of limit cycle oscillations.

The parameters of the second process model (2-3), the integrator with a lag can be identified directly with the relay experiment. The relations for the calculation of the model parameters, obtained by measuring the amplitude and the frequency of limit cycle oscillations, are (Perić et al., 1997):

\[ T_2 = \frac{1}{\omega} \frac{\sqrt{A^2 - \epsilon^2}}{\epsilon}, \]  
(3-35)

\[ K_2 = \frac{\pi A}{4d} \frac{A \omega}{\epsilon}. \]  
(3-36)

The estimation of model parameters during the relay experiment permits the use of different PID controller design techniques in the design phase of the autotuning algorithm. Some of the suitable design techniques have been described in the second chapter.

### 3.3.3 Autotuning in the presence of constant load disturbance

As mentioned earlier, adaptive algorithms require a supervision shell that would make them more robust for application in industrial environment. One of the tasks of the proposed supervision shell is to detect constant load disturbance. This feature is important in both phases of an autotuning algorithm: experimental and control. When load disturbance is present in the control system during the relay experiment inaccurate parameters of limit cycle are identified. It is, therefore, advisable to compensate influence of load disturbance during experiment (Hang et al., 1993). In addition, PID controllers tuned according to rules optimized for set-point change often have poor performance when load disturbance occurs. The use of a two-degree-of-freedom PID controller, as described in the second chapter, often solves this problem. Another possibility, proposed by Petrović et al. (1998), is to employ load disturbance detection scheme introduced by Hägglund and Åstrom (1997).

The presence of constant load disturbance during the relay experiment creates asymmetrical oscillations. As a consequence, this asymmetry introduces errors into the estimation of the ultimate gain and frequency. Asymmetry can be detected when consecutive
Oscillation half periods differ significantly. A non-symmetrical relay element (Figure 3.7) (Hang et al., 1993) has the purpose to restore symmetry to oscillations and the validity of the estimation. The employment of the non-symmetrical relay element redefines the values characterising the relay output, as follows:

\[ d' = \frac{d_+ + d_-}{2}, \]  
\[ d_0 = \frac{d_+ - d_-}{2}, \]  

where \( d' \) can be regarded as the output amplitude of the relay and \( d_0 \) is the bias around which the relay output oscillates. The bias \( d_0 \) is used to cancel constant load disturbance. Iterative cancellation through biasing, as described in Hang et al. (1993), can easily be incorporated in an autotuning algorithm. The procedure works for load disturbances with magnitude not exceeding the output amplitude \( d' \) of the relay. The cancellation may take several steps until the desired accuracy is achieved.

The load disturbance detection scheme detects the presence of load disturbance through the characteristics of control and output signal during transients. The control signal and the output signal are filtered in band-pass filters (BP) and the obtained signals are processed in a simple logic block. The logic block then detects whether there was set point change or load disturbance (Hägglund and Åstrom, 1997). The logic signal is used to switch PID controllers, of which one is optimized for set point change and the other for load disturbance (Petrović et al., 1998; Šilj, 1997). The switch between the two controllers should rely on bumpless techniques described in section 2.2. Figure 3.8 shows the detection and control setup.

![Fig. 3.8. Block diagram for detection of set-point change and load disturbance.](image-url)
Band-pass filters $G_{fu}$ and $G_{fy}$ have the transfer functions of the form (Petrović et al., 1998):

$$G_f(s) = \frac{K_f s}{(s + \omega_f)(s + N_f \omega_f)},$$  \hspace{1cm} (3-39)

where cut-off frequency $\omega_f$ is calculated from the dominant time constant $T_p$ of the process:

$$\omega_f = \frac{1}{T_p},$$  \hspace{1cm} (3-40)

and the gain of the filter $K_f$ is calculated as:

$$K_f = \frac{1}{K_{PR} N_f \omega_f^2},$$  \hspace{1cm} (3-41)

where $K_{PR}$ is the static gain of the process and $N_f$ is the constant used to fine-tune the shape of the logic signal $l(t)$ (Petrović et al., 1998). Its recommended value should keep in the between 0.5 and 2. The filters are band-pass filters, so that static values are eliminated on one side of the frequency characteristic, and the noise influence is suppressed on the other side.

![Fig. 3.9. Detection of set-point change and load disturbance: filtered control signal $u_f$ and filtered process output signal $y_f$.](image)

Figure 3.9 shows the waveforms of filtered control and output signal after set point change. When set point change occurs, filtered signals $u_f$ and $y_f$ change in the same direction, provided that the process has a positive static gain and that the process is non-minimum phase. However, when load disturbance occurs, filtered signals $u_f$ and $y_f$ change in the opposite direction. This happens because load disturbance is negated, as a part of feedback signal, before it is noticed in the control signal. This property can be used to produce the logic signal in a logic block which processes filtered signals. The functioning of the logic block can be expressed by using the C syntax:
3. Autotuning PID Controller

```c
if((abs uf)<=n_level)||(abs yf)<=n_level) out=1.0;
else { if(sign uf)==sign yf) out=1.0;
else out=0.0; }
```

where `n_level` is the estimated noise level, `uf` and `yf` are filtered signals, and `out` is the output of the logic block. The value of the logic signal equal to 1 denotes set point change, and the value equal to 0 denotes load disturbance. The logic activity threshold `n_level`, related to the estimated noise level, is introduced in order to prevent shifts between the two sets of PID controller parameters as a possible consequence of the noise signal fluctuations.

Another function of the supervision shell suitable for the use in the autotuning algorithm suggested by Petrović et al. (1998) is the outlier removal. The outliers are removed through the application of the median function with three arguments. The function finds the median value using the current process output sample and two previous measurements. The median value is the output value of the function and the same procedure is repeated for each sampling step. It introduces a slight delay in the measured signal, which can be compensated through a higher sampling rate. The function is very useful in the autotuning framework, since outliers can disturb the process control significantly and can induce erroneous measurements of the limit cycle amplitude during the relay experiment.

### 3.3.4 Addition of the preliminary identification phase

Preliminary identification is an additional control phase prefixed to the basic autotuning algorithm. Its aim is to determine the type of the controlled process and to make crude estimates of its parameters. It determines the type of the relay experiment, which is used in the ensuing stage of the algorithm. The obtained values serve to set the initial parameters of the relay. This procedure shortens the time needed for the relay experiment, because the relay parameters are already adjusted to approximate parameters of the process at the beginning of the experiment. Furthermore, the results of preliminary identification may serve to determine the process dead time. It is important to note that preliminary identification is made during the phase of reaching the operating point, and it can be viewed as a normal start-up procedure of an autotuning algorithm. The inclusion of the preliminary identification phase to the basic autotuning algorithm has been outlined in Perić et al. (1997) and considered in detail in Branica et al. (1999).

In the phase of preliminary identification the controlled process is excited with a ramp signal. The slope of the ramp signal is selected according to the user's choice of the expected time scale of the process. The preliminary identification is supervised through monitoring of the control signal `u(t)` and of the output signal `y(t)`. If the control signal reaches the upper limit value, the procedure is stopped and repeated with a slower ramp. Additionally, when the output signal does not pass a given threshold in preset time, calculated from the process time scale, the procedure is repeated with a faster ramp. Figure 3.10 shows the typical control signal `u(t)` and process output `y(t)` in the phase of preliminary identification.

Values of the process output signal `y(t)` are collected during the experiment. The collection of data should start when transients in `y(t)` die out and should stop before `y(t)` reaches
the operating point. The responses of time lags in the process are considered as transients in the output signal. When \( y(t) \) reaches 30% of the operating range \( (y_{100}-y_0) \), the time instant \( t_{30} \) is regarded suitable to start data collection and the time instant \( t_{60} \) suitable to end it (Fig. 3.10.).

Completed the collection, the obtained data set is fitted to presumed models, the parameters of the models are calculated, and the type of the process is determined. Furthermore, the initial parameters of the relay for the experiment are calculated. After the calculations, the process is driven into the operating point \( y_{100} \) by a conservative controller which is designed using the model of the process obtained in the preliminary identification. The P controller is used for astatic processes (type 1 processes) and the PI controller for static processes (type 0 processes). The relay experiment starts when the process output reaches the desired operating point.

As described earlier, the controlled process involves two standard models. The first is the FODT model (2.2):

\[
G_1(s) = \frac{K_i e^{-sT_i}}{1 + T_i s}, \tag{3-42}
\]

and the second consists of an integrator and a time constant (2.3):

\[
G_2(s) = \frac{K_2}{s(1 + T_2 s)}. \tag{3-43}
\]

When excited with the ramp signal:

\[
u(t) = K_0 t, \tag{3-44})
\]

the response of the FODT model is:
3. Autotuning PID Controller

\[
y_1(t) = \begin{cases} 
0; & t < T_{i1} \\
K_iK_0 \left( (t - T_{i1}) - T_{i1} \left( 1 - e^{\frac{t - T_{i1}}{T_{i1}}} \right) \right); & t > T_{i1}
\end{cases}
\]  
(3-45)

and the response of the second model is:

\[
y_2(t) = K_2K_0 \left[ \frac{t^2}{2} - tT_2 + T_2^2 \left( 1 - e^{\frac{t}{T_2}} \right) \right].
\]  
(3-46)

The response of the first model (3-45) is a linear function with a transient, of the second (3-46) a parabolic function with a transient. The difference (linear function and parabolic function) in response serves to determine the type of the process. The transient responses that are present as exponential terms in (3-45) and (3-46) diminish after some time (5-6 time constants \(T_i\) or \(T_2\)) after which (3-45) and (3-46) approach their asymptotes, that is, steady-state responses to the ramp excitation signal (3-44):

\[
y_{1_a}(t) = K_iK_0[t - (L_i + T_i)] = A_i + B_i,
\]  
(3-47)

\[
y_{2_a}(t) = K_2K_0 \left( \frac{t^2}{2} - tT_2 + T_2^2 \right) = A_2t^2 + B_2t + C_2.
\]  
(3-48)

Asymptotes of the process responses are used in least squares (LS) procedure to calculate the model parameters. Least squares procedure finds the parameters by minimizing the sum of square errors defined by:

\[
SSE = \sum_{i=0}^{n-1} [y_i - y_a(t_i)]^2
\]  
(3-49)

where \(t_i\) denotes equidistant time instants between \(T_{\text{start}}(t_{30})\) and \(T_{\text{end}}(t_{60})\), \(y(i)\) is the measured value at time instant \(t_i\), and \(n\) denotes the total number of samples collected during the procedure. \(SSE_1\) denotes SSE of the first model, whereas \(SSE_2\) denotes SSE of the second model.

The LS procedure is performed in a standard manner, as described by Biles and Swain (1980). Normal equations of the procedure for the FODT model are obtained by taking partial derivatives of SSE:

\[
SSE_1 = \sum_{i=0}^{n-1} [y(i) - y_a(t_i)]^2
\]  
(3-50)

in respect to the parameters of the model:

\[
\frac{\partial SSE_1}{\partial A_i} = \sum_{i=0}^{n-1} \left[ y(i) - A_i t_i^2 - B_i t_i \right] = 0,
\]  
(3-51)

\[
\frac{\partial SSE_1}{\partial B_i} = \sum_{i=0}^{n-1} \left[ y(i) - A_i t_i - B_i \right] = 0.
\]  
(3-52)

The following abbreviations denote time moments used in the calculation:

\[
U_0 = n, \quad U_k = \sum_{i=0}^{n} t_i^k, \quad V_k = \sum_{i=0}^{n} y_i t_i^k, \quad W_k = \sum_{i=0}^{n} y_i^2 t_i^k,
\]  
(3-53)

that are collected at each time instant.
The parameters of asymptote $y_{1a}(t)$ are calculated by inserting abbreviations (3-53) in (3-51) and (3-52), and by solving the obtained equations:

$$A_i = \frac{V_1 U_0 - V_0 U_1}{U_2 U_0 - U_1^2}, \quad (3-54)$$

$$B_i = \frac{V_0 U_2 - V_1 U_1}{U_2 U_0 - U_1^2}, \quad (3-55)$$

These parameters can be converted into the parameters of the FODT model:

$$K_i = \frac{A_i}{K_0} = \frac{1}{K_0} \frac{V_1 U_0 - V_0 U_1}{U_2 U_0 - U_1^2}, \quad (3-56)$$

$$T_i + T_n = -\frac{B_i}{A_i} = \frac{V_1 U_1 - V_0 U_2}{V_1 U_0 - V_0 U_1}. \quad (3-57)$$

Equations (3-56) and (3-57) can be used to calculate the FODT parameters, but it is possible to improve calculation by moving the coordinate system used in the calculation from the axis pair $t$-$y$ to the axis pair $t'$-$y'$, as shown in Figure 3.11.

![Fig. 3.11. LS procedure: moving the coordinate system for the first model.](image)

In the new coordinate system asymptote of the first model becomes:

$$y_{1a}'(t) = A_1' t', \quad (3-58)$$

and SSE becomes:

$$SSE_1' = \sum_{i=0}^{n-1} \left[ y(t_i) - A_1' t_i \right]^2. \quad (3-59)$$

By repeating the LS procedure defined by steps (3-50)-(3-53), the parameter $A_i'$ is obtained as:

$$A_i' = \frac{V_1'}{U_2'}, \quad (3-60)$$
where time moments $V_1'$ and $U_2'$ are calculated in the new coordinate system, according to (3-53). The second parameter of the model is obtained from relations between the coordinates of the new and the old coordinate system:

$$B_1' = y_{\text{start}} - A_1'T_{\text{start}}, \quad (3-61)$$

where $B_1'$ is an auxiliary variable. As in (3-56) and (3-57), the parameters of the FODT model are calculated as:

$$K_1 = \frac{A_1'}{K_0}, \quad (3-62)$$

$$T_i + T_{ni} = -\frac{B_1'}{A_1'}, \quad (3-63)$$

New formulae (3-60)-(3-63) are much simpler and more accurate since subtractions are avoided in the denominators of equations (3-54) and (3-55). Besides, this improvement ensures that the calculated time moments have lower values, since the values of the output signal and time instant are lower in the new coordinate system and so is the possibility of a calculation overflow.

The same change of the coordinate system of the second model results in a new form of asymptotic parabola of response (Fig. 3.12):

$$y_{2a}'(t) = K_2K_0'r'(t+T_2') = A_2'r'^2 + B_2'r', \quad (3-64)$$

where $T_2'$ is an auxiliary constant without physical interpretation. This form of response asymptote is more suitable for the linear LS procedure because the form given by (3-48) has a non-linear relation between parameters $B_2$ and $C_2$, which requires optimization (Hamming, 1973).

![Fig. 3.12. LS procedure: moving of coordinate system for the second model.](image)
The equation defining \(SSE_2\) model is the base for LS estimation of the model parameters:

\[
SSE_2' = \sum_{i=0}^{n} \left[ y_i' - A_i' t_i'^2 - B_i' t_i' \right] .
\]  

(3-65)

The LS procedure is rerun to establish relations between the parameters of the asymptote as defined by (3-64):

\[
A_i' = \frac{V_i' U_2' - V_i' U_1'}{U_1'^2 - U_2'^2},
\]

(3-66)

\[
B_i' = \frac{V_i' U_3' - V_i' U_4'}{U_3'^2 - U_4'^2}.
\]

(3-67)

Again, time moments \(V_i'\) and \(U_i'\) are calculated in the new coordinate system using relations (3-53). Upon (3-64), the gain of the second model is calculated as:

\[
K_2 = \frac{A_i'}{K_0},
\]

and the auxiliary constant is:

\[
T_2' = \frac{B_2'}{A_2'} = \frac{V_2' U_3' - V_1' U_4'}{V_2' U_2' - V_1' U_3'}.
\]

(3-69)

The estimation of the time constant \(T_2\) is based on the observation that the minimum of parabolic asymptote \(y_{2a}(t)\) is reached at the time instant \(t = T_2\) in the original coordinate system, as can be observed in Figure 3.12 and in the definition equation (3-48). Thus, the relation for calculating the time constant \(T_2\) is (Fig. 3.12):

\[
T_2 = T_{\text{start}} - \frac{T_2'}{2}.
\]

(3-70)

The next step in preliminary identification is to decide which of the used models is more adequate to describe the process. SSE is calculated for both models using equations (3-59) and (3-65):

\[
SSE_1' = W_0' - 2 A_1' V_1' + A_1'^2 U_2',
\]

(3-71)

\[
SSE_2' = W_0' - 2 A_2' V_2' - 2 B_2' V_1' + A_2'^2 U_4' + 2 A_1' B_2' U_3' + B_2'^2 U_4'.
\]

(3-72)

These values cannot be directly compared, since the second model is a parabolic function and usually fits better to the response data than the linear function of the first model. So, the procedure temporarily assumes that the second is the ‘true’ model of the process. After that, the procedure checks relative deviation of the first model from the ‘true’ model. Unless there is a substantial deviation, it is assumed that the first model is suitable for the description of the process. Therefore, the assertion:

\[
\frac{|SSE_2' - SSE_1'|}{SSE_2'} > r_{\text{threshold}}.
\]

is used to make the decision. If the assertion (3-73) holds and the first model deviates sufficiently from the ‘true’ model, the second model is more appropriate for modeling the controlled process.
Furthermore, the procedure checks the validity of the estimated process constants; and a model with a negative time constant is ruled out. If both models fail the test, the procedure signals an error.

The final part of preliminary identification consists of driving the process into the desired operating point. It is done by employing the PI controller for processes modeled by the first model and P controller for processes described by the second model. The parameters of these controllers are calculated on the basis of estimated parameters of the process model.

The processes that can be described by the FODT model are driven into the desired operating point by using a PI controller tuned according to the $\lambda$-tuning design method (subsection 2.3.5). Since it is not possible to distinguish the time constant and dead time in the estimated sum ($T_1 + T_d$) (equation (3-63)) it is assumed that the time constant $T_1$ equals one third of this sum and that the dead time $T_d$ equals two thirds of the sum. To obtain conservative design closed-loop time constant is set to $T_c = 2T_1$. With these specifications, pole cancellation $\lambda$-tuning method produces the following parameters of the PI controller:

$$K_p = \frac{1}{4K_1},$$  \hspace{1cm} (3-74)

$$T_p = \frac{1}{3}(T_1 + T_d),$$  \hspace{1cm} (3-75)

where $K_1$ and $(T_1 + T_d)$ are values estimated during previous stages of preliminary identification. This choice of PI controller parameters was tested for extreme process parameters and showed conservative control behavior for all cases.

The gain of the P controller used for processes modeled with the second model is set in such manner as to equal the relative damping ratio $\zeta$ of the closed-loop system to $\frac{\sqrt{2}}{2}$. The closed-loop transfer function is:

$$G_c(s) = \frac{K_pK_2}{T_2s^2 + s + K_pK_2}.$$  \hspace{1cm} (3-76)

Setting the gain of the controller to the value:

$$K_p = \frac{1}{2T_2K_2},$$  \hspace{1cm} (3-77)

achieves the required relative damping ratio. The resulting time constant of the closed-loop system is:

$$T_c = \sqrt{2T_2}.$$  \hspace{1cm} (3-78)

Subsection 3.3.2. describes the use of the relay experiment in the estimation of mathematical model parameters of the controlled process. It is not able to estimate all three parameters of the FODT model. An additional equation is needed to make this identification possible. The equation (3-63) can be used together with equations (3-28) and (3-29)which define the behavior of the FODT process under relay control. These equations form a system of three equations and three unknowns which can be easily solved.
3.3.5 Other modifications of the basic autotuning algorithm

One useful modification of the basic autotuning algorithm handles autotuning of unstable FODT processes. Majhi and Atherton (1998) described such autotuning setup, as follows:

\[ G(s) = \frac{K_0 e^{-\tau s}}{1 - T_1 s}. \]  

The model has an unstable pole at the position \( s = \frac{1}{T_1} \). Successful estimation with the ordinary relay experiment is possible for processes with parameters satisfying the condition:

\[ \frac{T_{dl}}{T_1} < \ln 2 = 0.693. \]

The inclusion of the PD element in a minor feedback loop during relay experiment enables the estimation of parameters for processes with larger dead time (Fig. 3.13). Such estimation is very accurate because it is based on integrals of input and output signals and is not very sensitive to noise.

The PD element can also be used during the control phase (\( S_1 = 1 \) and \( S_2 \) is on), which gives the PI-PD structure (Fig. 3.13) an edge over other PID controllers in control of unstable FODT processes (Majhi and Atherton, 1998).

Since the basic autotuning algorithm is based on the identification of the ultimate point of the process, many authors have considered the usage of the algorithm with various frequency response methods. Wang et al. (1999) proposed an application of the relay experiment for non-parametric identification of the process frequency response. Discrete Fourier Transform (DFT) in the form of fast DFT algorithm (FFT) is employed to estimate the process frequency response. The estimated frequency response serves to design a PID controller in the tuning phase of the algorithm.

In order to get accurate results, process input and output signals have to be divided in stationary and transient parts. The output signal is represented as the sum of these parts:
\[ y(t) = y_s(t) + y_t(t). \] (3-81)

Similarly, the process input signal is represented as:

\[ u(t) = u_s(t) + u_t(t). \] (3-82)

The stationary and transient parts of signals, expressed as addends in (3-81) and (3-82), are separated by logging signals for a period of time that is longer than the transient response and by subtracting the last period of responses from the recorded signals. After the division, proposed algorithm performs separate DFTs on signals and obtains the frequency responses of corresponding parts. The frequency response of the process is calculated as their combination using the below relation (Wang et al., 1999):

\[ G(j\omega) = \frac{Y(j\omega)}{U(j\omega)}, \] (3-83)

where \(\omega\) are frequencies between 0 and the critical frequency \(\omega_c\). The number of frequency points depends on the number of oscillations which have to be recorded, so that measurement includes complete transient response.

The main advantage of this procedure is that multiple frequency response points are accurately identified by a single experiment, which shortens the testing time.

### 3.4 Examples of adaptive PID controllers

The main objective of this study was to develop a stand-alone PLC software module for the autotuning PID controller. This section gives several examples of the adaptive use of the PID controller and seeks to evaluate controllers with similar capabilities and to compare their characteristics. The aim is to distinguish common and useful characteristics which can eventually be integrated into a PLC software module.

Ho et al. 1996b described a self-tuning PID controller that relied on information from the relay experiment. The controller uses the relay experiment to identify the ultimate point of the process and to initialize the PID controller. Afterwards, several points on the Nyquist curve are tracked and, if necessary, updated. The tracked points \(G(j\omega)\) are picked out in the frequency range around the ultimate frequency \(\omega_u\), as shown in Figure 3.14. The frequency band of estimated points should be as wide as possible and the recalculation of the frequencies should be simple. In other words, frequencies \(\omega_i\) are selected to form geometrical progression in the frequency domain. Tracking of points \(G(j\omega)\) is used to accurately estimate the process ultimate point which can change during operation due to variations in the process parameters or due to the presence of non-linearity in the process. When the parameters of the ultimate point are changed, tracked points \(G(j\omega)\) are used to prediction a new ultimate point by interpolation or by extrapolation. In such case, the frequencies \(\omega_i\) are recalculated and the estimation is repeated.
with the new set of points $G(j\omega_i)$. The PID controller parameters are retuned after each valid update of the ultimate point estimate.

![Nyquist curve](image)

Fig. 3.14 Estimated and tracked points of the Nyquist curve.

Each point $G(j\omega_i)$ is estimated in the least-squares procedure which uses filtered control signal $u(t)$ and filtered process output signal $y(t)$. Signals $u(t)$ and $y(t)$ are filtered in a set of band-pass filters of the form (Ho et al., 1996b):

$$G_{BPi}(s) = \frac{2\zeta s}{s^2 + 2\zeta \omega_i s + \omega_i^2},$$

where $\omega_i$ are BP filters center frequencies and $\zeta$ is the relative damping factor of these filters.

The estimation and adaptation are activated when the condition requiring the presence of sufficient excitation in the control system for a prescribed period is met.

The proposed self-tuning algorithm (Ho et al., 1996b) is very robust because it identifies process parameters in the middle frequency range, so that the estimates are not affected by high-frequency noise or by low-frequency disturbances. In addition, the design of the algorithm includes such supervision features as initialization, stability check, excitation check, validation, and limits imposed on the controller parameters.

Radke and Isermann (1987) recognized the need to combine features of the adaptive and PID controller. A controller thus combined would be able to work in two modes of operation:

- Initial tuning (autotuning);
- Continuous tuning (self-tuning).

The project of designing adaptive controller with these properties was accomplished on the microcontroller DMR-16, which is based on microprocessor 8086/8087. The problem was divided in the two tasks that are executed iteratively: control and optimization. The control task executes the PID algorithm, defined by (2-27), at each sampling step. The optimization task adapts PID controller and requires a lot of computation. It has low priority and its execution is spread over several sampling periods.
3. Autotuning PID Controller

The optimization task involves a procedure for the identification of the process parameters and a procedure for optimization of the PID controller parameters. The identification follows a two-stage square root least-squares algorithm. This type of recursive least-squares algorithm has excellent numerical properties for larger execution periods. New parameters of the PID controller are calculated using the identified process model. The goal of calculation is to minimize the quadratic performance criterion (Radke and Isermann, 1987):

\[ S = \sum_{i=0}^{N} \left[ e_i^2 + cK_p^2 \Delta u_i^2 \right], \]  

(3-85)

where \( e_i \) is the error signal at the time instant \( i \), \( c \) is the weighting factor, \( K_p \) is the process gain, and the difference \( \Delta u_i \) is defined as \( \Delta u_i = u_i - u_{i-1} \). The quadratic performance criterion is minimized in respect to PID controller parameters and can be reformulated in the time domain and in the \( Z \)-domain. The minimization employs the hill-climbing algorithm of Hooke and Jeeves (also called pattern search algorithm; Biles and Swain, 1980).

The implementation of a parameter-adaptive PID controller (Radke and Isermann, 1987), involves monitoring of pre-conditions for the tuning procedure in order to identify errors in the starting period and indicate them to the operator. Minimum sampling time equals 65ms and minimum execution time is 4 s for this implementation, which is acceptable for a wide range of industrial processes.

Poulin et al. (1996) suggested the design of an autotuning and adaptive PID controller (AAC) that is based on the recursive explicit identification procedure, followed by a PID controller design derived from several tuning rules. The applied tuning rule depends on the type of the identified process.

Data acquisition system of the controller consists of an analog anti-aliasing LP filter for measuring the process output \( y(t) \) and of a pair of digital BP filters for filtering the control signal \( u(t) \) and the measured process output signal. These filters are used to reduce noise and to attenuate the effect of low frequency disturbances on the identified process model. The identification is based on the damped least-squares (DLS) algorithm, which is an extended version of the recursive least-squares (RLS) algorithm. The identified model is the second order transfer function in the \( Z \)-domain with dead time (Poulin et al., 1996):

\[ G_M(z) = \frac{b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}}z^{-d}, \]  

(3-86)

where \( d \) represents dead time in the number of sampling periods.

Prior to starting the controller, user has to specify process class. One of the three process classes can be chosen: stable, unstable and process with an integrator. If the process belongs to the class of stable processes open-loop autotuning is performed, otherwise autotuning is made in the closed loop. Further characterization of the process is made during autotuning and identification, so that proper tuning rule can be applied after autotuning. Additionally, parameters of the supervision shell are adjusted: noise band is measured, dead time is estimated, and cutoff frequencies for data acquisition system are set. When autotuning is completed, the adaptation mechanism is activated. The comparison (Poulin et al., 1996) with several
commercial adaptive controllers has shown that the described controller generally performed better.

A paper by Åstrom et al. (1993) and a book by Åstrom and Hägglund (1995) give an overview of the main features of commercial adaptive PID controllers much used and produced for industry by various manufacturers. This paper singles out a few.

Foxboro EXACT 760/761 is a single-loop adaptive controller which uses the step-response analysis for autotuning and a pattern recognition technique together with heuristic rules in the adaptation mode. Controller ECA400, made by Alfa Laval Automation, uses relay autotuning and model-based adaptation. UDC 6000, a controller produced by Honeywell, has an adaptive function, which consists of model-based and rule-based procedures, and can only be applied for stable, type 0 processes. Yokogawa controllers, SLPC-181 and SLPC-281 have an autotuning feature based on the step-response analysis and the adaptive mode of operation which is model-based.

All described controllers have an autotuning feature, which can be followed by continuous adaptive control. These controllers differ in techniques employed to perform these tasks and in the type of processes they handle. Furthermore, all controllers have a supervision shell and features for treating noise.

A well-known German company Siemens introduced a ‘PID self-tuner’ software package to the SIMATIC S7 product family (SIEMENS, 1997a). Since developed autotuning PID controller is implemented as a software module in same PLC family, it is worthwhile to explore the capabilities of this self-tuner.

The package can be used to tune various PID modules, both software and hardware, in the SIMATIC system. Primarily, these PID modules are used without the ‘PID self-tuner’ block and are tuned by the commissioning personnel. The package is mainly applied in temperature control, fluid level control, and flow control. The self-tuner package warrants four modes of operation:

• The initial controller tuning: process identification after a sufficiently large step set-point change and tuning of PI or PID controller;

• Controller adaptation to an identified process: optimization of an already tuned controller by a step set-point change with small amplitude;

• Variable controller structure: switching between PI/PID and PD controller structure in order to avoid large overshoot during set-point change with a controller designed for good response to load disturbances;

• Manual controller: cancellation of other modes of operation.

The main shortcoming of the package is that it can be used for rather a narrow class of processes. The controlled process should be stable with reasonably low gain and short time lags, which rules out the processes type above 0. Moreover, it should have almost linear characteristics in the operating range and should allow control by a positive monopolar signal. The signal-to-noise ratio should be adequately high.
These examples suggest that a developed auto-tuner should be applicable for a wide class of processes. It should be apt to control type 0 and type 1 processes, as well as processes with large dead time as the most common in industry. Moreover, the autotuning procedure should be able to tune different PID controllers in the SIMATIC family.

Since the main shortcoming of many adaptive controllers is their sensitivity to industrial environment, it is very important to make them as robust as possible. The incorporated supervision shell should employ sensitive and accurate detection, especially of important operating conditions. These conditions, together with significant controller states, should be indicated to the operator. The supervision shell should be able to treat noise in such fashion as to be able to tolerate moderate noise signals in the control system. Adaptable BP filters could be employed to treat noise in the measured signals. In addition, it should allow simple expansion of the algorithm with self-tuning capability.

In order to evaluate the described PID autotuning methods, these were simulated in a MATLAB/SIMULINK software package. Chapter four brings the main results and the developed autotuning PID controller.
4. SIMULATION STUDY OF THE AUTOTUNING PID ALGORITHM

The purpose of the computer simulation performed in this study was to evaluate the features of different autotuning algorithms and to check the several extensions of the basic algorithm. The simulation puts the algorithm on a simple and quick test with a wide range of model processes. Moreover, it can be viewed as the first step in the designing an autotuning PID controller, because the main functional blocks of the algorithm take shape in the process of creating simulation models.

The choice of the programming environment for the computer simulation fell on MATLAB/SIMULINK package of the company The MathWorks because of its versatility and extensibility. MATLAB evolved into a complete computer-aided development environment from a simple interactive matrix calculator which was based on LINPACK and EISPACK software libraries. SIMULINK is a set of functions in the MATLAB terminology ‘toolbox’ which is used to model, simulate, and analyze dynamical systems in the MATLAB framework (The MathWorks, 1998). MATLAB version 5.2 and SIMULINK version 2.2 was used in this study. MATLAB development environment allows efficient performance of the following tasks: mathematical computation, algorithm development, simulation and prototyping, data analysis and visualization, real-time control, and application development (The MathWorks, 1997a). In other words, it seems that MATLAB programming package provides a nearly optimal programming environment for the development and testing of control algorithms.

The first topic to be addressed in this chapter is the implementation of preliminary identification procedure. Follow comments on the choice of dead-time compensating controller which is used as the part of the autotuning controller in the second section. The third section details the simulation model of the autotuning PID controller, the main parts of the controller, and how it functions.

4.1 Simulation and testing of the preliminary identification procedure

Preliminary identification is a new, additional feature of the proposed autotuning controller. Its properties required investigation since the derivation of the procedure involved several approximations. Testing, furthermore, included noise sensitivity as well as the behavior of the procedure for processes that differed from the model processes incorporated into the procedure. The pseudocode is given below in a MATLAB-like syntax with comments referencing to related equations. The procedure is executed at each step of the simulation. Functions implementing safety measures related to the slope of the ramp signal are omitted in this version of the procedure pseudocode.
Pseudocode of the preliminary identification procedure:

```plaintext
switch PIstate % switch depending on the state of preliminary identification block
  case PIramp % apply ramp signal with a given slope
    calculate and apply ramp_signal
    if process_output > (0.3 * operating_point) then
      store t_start, y_start
      PIstate := PImeasurement
  case PImeasurement % measurement between Tstart and Tend
    calculate and apply ramp_signal
    t_new := current_time - t_start
    y_new := process_output - y_start
    calculate modified time weighted moments with t_new, y_new
    if process_output > (0.6 * operating_point) then
      calculate parameters of the first model: K1, T1+Tt1 % eq.(3-62) and (3-63)
      calculate SSE of the first model: SSE1 % eq.(3-71)
      calculate parameters of the second model: K2, T2 % eq.(3-68) and (3-70)
      calculate SSE of the second model: SSE2 % eq.(3-72)
      detect type of the process with SSE1 and SSE2 % eq.(3-73)
      PIstate := PIcontrol
      % check if time constants are larger than 0
      if T1+Tt1 < 0 then
        process_type := 2
      if T2 < 0 then PIstate=PIerror
      if T2 < 0 then
        process_type := 1
      if T1+Tt1 < 0 then PIstate=PIerror
      if process_type == 1 then wait_time := 2 * T1+Tt1
      else wait_time := 2 * T2
  case PIcontrol % attaining operating point
    error := operating_point-process_output
    if process_type == 1 then
      calculate state of integrator in PI controller with error
      calculate output of PI controller with error
    else
      calculate output of P controller with error
    if (process_output in 2% bounds around operating_point) then
      if (current_time - time_enter_bound) > wait_time then
        PIstate := PIend
      else
        time_enter_bound := current_time
  case PIend % preliminary identification finished
```
The pseudocode of the preliminary identification distinguishes three active states of the procedure as different values of variable $PI_{state}$. These are:

1. Generation of ramp excitation and waiting;
2. Generation of ramp excitation and measurement;
3. Control of the process.

These states follow the description of the procedure given in section 3.3.4. When the procedure is in the first state, it generates ramp excitation signal $u(t)$ and waits until the process output signal $y(t)$ reaches 30% of the operating point. At that moment, the current time and process output value are stored and the algorithm switches to the second state in which time weighted moments of the process output signal $y(t)$ are accumulated, as defined by the equations (3-53). When the process output signal $y(t)$ reaches 60% of the operating point, calculations are performed, process parameters are estimated, and the type of the process is determined. After that, P or PI controller drives the process into the operating point. When the process output signal $y(t)$ reaches operating point, the procedure waits for the control signal $u(t)$ to settle before completing the execution. The waiting time is set to double the value of the estimated time constant of the process.

![Fig. 4.1. Procedure of the preliminary identification of the FODT process.](image-url)
Figure 4.1 shows signals during the preliminary identification of the FODT process: process output signal $y(t)$, control signal $u(t)$, and state variable $\text{PIstate}$. The parameters of the process were: $K_{p1}=2$, $T_{p1}=6s$ and $T_{q1}=4s$. The preliminary identification correctly determined the type of the process, and estimated the following values: $\hat{K}_1 = 2$ and $\hat{T}_1 + \hat{T}_q = 10$. State transitions of the procedure are visible in the changes of the $\text{PIstate}$ variable.

The preliminary identification gives excellent results for this type of processes, even when a moderate noise signal is present in the measured process output signal $y(t)$.

Preliminary identification was also tested on processes with an integrator and a time lag defined by transfer function (2-3). For example (Fig. 4.2), it identified the process with parameters $K_{p2}=2$ and $T_{p2}=8s$. The type of the process was correctly determined and the estimated parameters were $\hat{K}_2 = 2.1$ and $\hat{T}_2 = 8.8s$.

For this type of processes, the procedure is very sensitive to the presence and the amplitude of the added measurement noise, so that errors in the estimation of the process parameters equaling 25% are not uncommon. A random number generator, coupled with a weakly dumped second-order filter, simulated the measurement noise signal. The sensitivity of the procedure to actual noise signals in industrial control systems has to be determined during the testing phase after PLC implementation of the autotuning algorithm. However, the results of the preliminary identification for this process type are not used to tune PID controllers. Instead, the tuning is based on the results of the relay experiment with adjustable dead time. In other words, in this type of processes, moderately inaccurate results of preliminary identification do not affect the correctness of a PID controller design.
Another important issue related to the accuracy of the procedure is the number of sampling steps during measurement (the second state of the procedure). The accuracy of the procedure improves with the number of sampling steps. It is determined by the slope of ramp excitation signal and by sampling time. A lower slope of the excitation ramp prolongs the measurement time and, increases the number of samples. Similarly, the shortening of sampling time increases the number of measurement steps. The simulation showed that a few hundred (300–700) points sufficed for successful preliminary identification for the first type of processes. The second type of processes, however, required more measurement steps (500–1200). These parameters of the procedure should be selected during experiments with PLC implementation of the complete autotuning algorithm.

Preliminary identification was also simulated with processes that did not match the models incorporated in the procedure. The models included additional time lags. The procedure identified the value of time constant that was approximately equal to the sum of the lag constants present in the process. This time constant can be interpreted as a ‘substitute’ time constant of the process. The relay experiment and controller tuning would verify the correctness of the identified value.

For both types of processes, attaining the operating point with internal control algorithm (P or PI) of the procedure was successful.

### 4.2 Choice of dead time compensating controller

Section 3.4 concluded with a statement that a developed autotuning PID controller it should be able to control processes with large dead time. In order to achieve that, dead time compensating controller has to be integrated into the autotuning controller. Comparison simulations, testing the robustness of several such controllers, have been performed prior to the integration of a dead time compensating controller into the autotuning controller. The robustness of the selected controller is an important issue because the identification procedure of the autotuning algorithm which includes preliminary identification and relay experiment can give parameters of the process that are not accurate. Besides, the developed controller can be used for processes that do not match the FODT model, so that inaccurate modeling can not be avoided.

This study compares the Smith predictor (with the FODT model and a PI controller acting on error signal \(e(t)\)), the PPI controller, and the FPPI controller. The theoretical basis and characteristics of the tested controllers are given in section 2.4.
Figures 4.3 and 4.4 show the responses of the control systems to the step set change at $t=10s$ and to step load disturbance at $t=50s$. The control involved a FODT process (2-2) with parameters: $K_p=2$, $T_p=3s$ and $T_{dp}=8s$, and with added measurement noise. In this example, compared dead time compensating controllers had the model parameters equal to the parameters of the process.
Figure 4.3 shows the response of the control system with the Smith predictor. Since in this controller the proportional part of the PI controller acts on the error signal $e(t)$, one may observe that the step in the reference value passed on to the control signal $u(t)$. This step represents a bump on the actuator which should be reduced.

The same excitation signals were applied to the FPPI controller, the response of which is shown in Figure 4.4. A similar response of the control system was obtained with the PPI controller. The step in the control signal $u(t)$ is avoided, so that control signal is formed as gradually changing signal. Although transient response of the control system is slower than the transient response of the control system with Smith predictor, such behavior is more desirable and the configuration with the proportional part of PI controller acting only on the output signal is preferred.

The FODT process was used with the same parameters for the test that followed. Process models, incorporated into the PPI and FPPI controllers, had the following model parameters: $K_{m1}=2$, $T_{m1}=3.75s$ and $T_{tm1}=10s$. In other words, the inaccuracy of the model time constant and model dead time was 25%, with model parameters larger than the actual parameters of the process. Figure 4.5 shows the response of the control system with the PPI controller and Figure 4.6 the response of the control system with the FPPI controller. The response of the control system with the PPI controller showed a significant degradation in control performance with respect to the previous example. The control signal was sluggish and oscillatory while attaining a steady state. The first undershoot equaled 23% and the control signal $u(t)$ contained noise from the measured process output signal $y(t)$. The FPPI controller (Fig. 4.6) showed better performance; the first undershoot equaled 14% and the control signal was much smoother, so that the effect of noise on the control signal was considerably lower. However, it also took long to attain a steady state.
Similar simulations showed that increasing of the process model inaccuracy would destabilize the control system with the PPI controller, whereas the control system with the FPPI controller would retain stability, but would lose on control performance. This effect is more apparent for processes with larger normalized dead time. The level of model inaccuracy which unacceptably degrades (or destabilizes) the performance of the control system depends on the value and on the inaccuracy of a particular process parameter.

![Graph](image1)

Fig. 4.6. Response of the control system with the FPPI controller with inaccurate parameters of the process model.

Usually, the PPI controller responds more quickly than the FPPI controller because of the added lag in the filter $G_F(s)$. This filter, however, substantially improves the robustness and reduces the effect of noise on the control system. This test showed that the FPPI controller was more successful than the PPI controller in controlling an inaccurately modeled process.

When the controlled process cannot be modeled with the FODT model, inaccuracies of the process model incorporated in the FPPI controller are inevitable. Figure 4.7 shows the response of the control system with the FPPI controller in a simulation of such situation. The controlled process consisted of the third-order lag with dead time:

$$G_p(s) = \frac{2e^{-8s}}{(3s+1)^2(0.75s+1)}.$$  

The model parameters in the FPPI controller were set to $K_m=2$, $T_m=6s$ and $T_{tm}=10s$. The response was aperiodic, and it showed that FPPI controller successfully controlled the process with a suitable choice of the model parameters. It can be concluded that the FPPI controller behaves well in comparable situations.
4. Simulation Study of the Autotuning PID Algorithm

The presented simulations, as well as the experience with the FPPI controller (Kousek, 1999), confirmed its good properties (see also the paper by Normey-Rico et al., 1997). Consequently, this controller is incorporated in the developed autotuning controller.

4.3 Simulation of the autotuning PID algorithm

The simulation of the autotuning PID controller was performed in the MATLAB/SIMULINK development environment. Figure 4.8 shows the functional scheme of the algorithm, and Figure 4.9 shows the SIMULINK block diagram.

The SIMULINK extension mechanism was used to simulate complex parts of the autotuning algorithm. The extension allows simple incorporation of additional blocks with the user-defined behavior into the simulation environment. There are two possibilities to accomplish this task: the first is to use M scripting files and the second to use the MEX extension mechanism. M files define modules with required behavior in the MATLAB scripting language and are interpreted at each simulation step. The MEX extension mechanism defines the C language interface to simulation engine and is used to produce compiled modules which are executed much quicker than the modules implemented in the scripting language. SIMULINK features for accelerated simulation execution such as external simulation mechanism (The MathWorks, 1998) and real time workshop (The MathWorks, 1997b) were employed in the final stages of the simulation study of the algorithm.
The functional scheme brings the key functional units (Fig. 4.8.) which emphasize the important behavior of the algorithm, while the block diagram (Fig. 4.9.) depicts all connections, blocks, and data stores participating in the simulation of the algorithm. As can be observed, the simulation required a complicated block diagram. The MATLAB execution list of the block diagram consisted of more than 200 units. This chapter comments the behavior of the algorithm with respect to the functional scheme and refers to the block diagram.

The control of the process involves three phases:

- Preliminary identification;
- Adaptive relay control;
- Control phase.

Switch S1 controls the transition from one phase to another, selecting appropriate control signal. The first phase of the autotuning controller consists of preliminary identification. The second phase includes a relay experiment. In the third phase, a suitably designed controller controls the process. A Suitable controller is selected by proper setting of the switch S2.

Fig. 4.8 Functional scheme of the autotuning controller.

Details about the first phase, preliminary identification, are in section 4.1. The second phase consists of a relay experiment. It is the most important and a very sensitive part of the
autotuning algorithm and requires careful design. The relay experiment is performed according to the type of the process, which is determined during the preliminary identification. Ordinary relay experiment is performed if the FODT process is detected, as described in section 3.2. Otherwise, if the process can be modeled with an integrator and a time lag, a modified relay experiment with adjustable dead time will be executed according to recommendations given in subsection 3.3.1. During the experiment, the relay amplitude $d$ and dead time $T_{tv}$ change simultaneously. Consequently, the relay element and dead time element have to be implemented in such manner as to allow simple reconfiguration.

A very important precondition for a successful relay experiment is the accurate measurement of the amplitude and of the period of limit cycle oscillations (Tomić, 1997). The accuracy is achieved by employing measures for noise removal and by collecting the oscillation data at appropriate time instants. These measures prevent misinterpretation of noise fluctuations as oscillations of the process output. Several simulation blocks perform this measurement during simulation. The measurement procedure can be observed in the block diagram through the signal flow from block ‘Remove noise’ to block ‘Amplitude and period measurement’.

![Figure 4.10. Measurement of amplitude of limit cycle oscillations.](image)

Figure 4.10 shows an example of the collection of oscillation data. A signal following the current maximum of the process output signal $y(t)$ (signal max in block diagram) is depicted together with the logic signal (signal getmax in block diagram) which signalizes the appropriate time instant to collect the data. One may note that noise is practically eliminated by this measurement approach.

Another problem with the relay experiment is that a change in relay parameters entails a change in parameters of limit cycle oscillations. Since limit cycle oscillations gradually change to a new steady state, several oscillation periods (2-3) after the change of relay parameters should not be included in the measurement. This significantly improves the accuracy of the measurement, yet also prolongs the experiment.
Fig. 4.9. SIMULINK block diagram of the autotuning controller.
4. Simulation Study of the Autotuning PID Algorithm

The length of the relay experiment is determined by the convergence of the relay adjustment procedure and by permissible tolerance in the oscillation parameters from the desired values. If the tolerance were too small, the relay adjustment procedure would last too long. Simulations demonstrated that tolerance of 10%–20% is a reasonable trade-off between the required accuracy and the length of the relay experiment.

The selection of an appropriate control strategy and the tuning of the respective controller follow at the completion of the relay experiment. The task is accomplished by the block denoted ‘Calculate controller parameters’ in the block diagram (Fig. 4.9). If the preliminary identification procedure detected the type 1 process which can be modeled with the model defined by (2-3), the PID controller is employed and is tuned according to the KLV method described in section 2.3.7. When the process can be modeled with the FODT model (2-2), the estimated time constant is compared to the estimate of dead time. The FPPI controller is tuned and started if the comparison shows that the dead time estimate is larger than the estimated time constant. However, if the estimated time constant of the process is larger than the dead time estimate, the selection falls on the PID controller. It is tuned according to the ISTE criterion, as explained in section 2.3.2.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>model of FODT process</th>
<th>est. of preliminary identification</th>
<th>est. after relay experiment</th>
<th>controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (Fig. 4.11)</td>
<td>2 6 8</td>
<td>1.96 13.27</td>
<td>1.96 5.23 8.03</td>
<td>FPPI</td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td>1.97 9.41</td>
<td>1.97 6.40 2.05</td>
<td>PID-ISTE</td>
</tr>
</tbody>
</table>

Table 4.1. Results of autotuning simulations with the FODT model processes.

Blocks ‘Decomposed FPPI controller’ and ‘Decomposed PID controller’ in the block diagram are simulation subsystems of these controllers. This form was used to easily change controller gains from the tuning block. Moreover, the reset function is incorporated in these controllers, which ensures smooth transition to the control phase. The ‘Switch’ block selects the proper control signal and works as the S2 switch from the functional scheme.

The chapter further describes a simulation of the extended autotuning algorithm and the results of simulations with typical process models. Table 4.1 summarizes the results of two exemplary simulations with the FODT models. These simulations show that process parameters were correctly estimated and that the autotuning algorithm selected the appropriate control strategy for the process.

Figure 4.11 shows the control signal $u(t)$ and the response of the FODT process $y(t)$ during simulation of the autotuning algorithm (Experiment 1. from Table 4.1). The autotuning algorithm correctly detected the need for the FPPI controller and tuned its parameters. The responses to set-point change ($t=400s$) and to load disturbance ($t=475s$) show that the FPPI controller performed well.
Comparable results were obtained from simulations with type 1 models, defined by (2-3). Figure 4.12 shows a typical control signal and response of the type 1 process during simulation of the autotuning algorithm. The parameters of the process model were $K_2=2$ and $T_2=5s$. One may note that the autotuning algorithm detected the proper type of the process in the preliminary identification procedure, performed the relay experiment with adjustable dead time, and successfully tuned the PID controller parameters.
It can be concluded that the extended autotuning algorithm exhibited acceptable and satisfactory control behavior when controlling various processes.

The described details of the extended autotuning algorithm provide a good insight into the structure of the algorithm, so that it can be successfully implemented in PLC. The implementation is detailed in the next chapter.
5 IMPLEMENTATION AND EXPERIMENTS WITH THE AUTOTUNING PID ALGORITHM

One of the main results of this thesis is a stand-alone software module, which implements the extended autotuning algorithm. The implementation is presented in this chapter.

The autotuning PID controller was implemented in a SIEMENS PLC of the product family SIMATIC S7-300 (SIEMENS, 1997b). It was programmed in programming environment STEP 7 (SIEMENS, 1997c) in statement list language (STL; SIEMENS, 1997d). The language has many characteristics of an assembly language and differs considerably from the simulation environment, which was used to develop the algorithm. The difference necessitated some changes in the design of the algorithm.

The first section of this chapter describes adaptations of the autotuning algorithm required for successful implementation in the PLC. The implemented module was tested on laboratory processes and the results of several experiments are presented in the second section.

5.1 PLC implementation of the autotuning PID algorithm

The autotuning PID controller is implemented as a set of function blocks in PLC STEP 7 programming environment (SIEMENS 1997c; SIEMENS 1997d). These blocks follow the functional division of the algorithm. Function blocks are scheduled through the use of several flags which define the current state of the controller. During operation of the controller its current state is checked at each sampling step in if-then decision structure, which dispatches calls to appropriate function blocks.

The implementation of preliminary identification procedure in the PLC conforms to the description of the procedure given in previous chapters. The only difference is the incorporation of noise and stationary offset measurement at the beginning of the procedure. The measured noise band is used in later stages of the autotuning PID controller for setting dead zones that prevent the influence of noise fluctuations on the behavior of the controller. The starting stationary offset was neglected in simulations, but has to be taken into account during the actual control of processes.

The relay element is implemented with adjustable parameters. In addition, the use of different dynamical elements (integrator, derivator, variable dead time element, etc.) in conjunction with the relay can easily be accomplished.

Since the simulation showed that the autotuning algorithm is very sensitive to the accuracy of the measurement of limit cycle parameters, it had to be implemented carefully. One of the problems encountered was that the oscillation parameters slowly converged to the stationary values when the relay parameters changed and especially when the changes were
significant. This transient causes erroneous measurements and should be excluded from them. The first few oscillation periods after the change are not included into the measurement in the PLC implementation of the autotuning algorithm. The number of excluded oscillation periods from measurement is a configurable parameter and is usually set to two or three.

Another problem with the measurement is related to the choice of sampling time. It is important that the choice does not adversely affect the accuracy of the measurement. It would be degraded if the sampling time were too long, that is, if too few points were sampled during one oscillation period. Several relay experiments showed that 20 to 30 sampling steps during one period of oscillation ensure adequate accuracy.

The PID controller function block, readily available in STEP 7 development environment, was the central part of the autotuning controller. It contained three algorithms for PID control which allowed several configurations. The function block ‘CONT_C’ used in this study was designed to control technical processes with a continuous input and a continuous output (SIEMENS, 1997b). This PID algorithm is of positional type and has a parallel form. In addition, includes the input for feedforward control and a switch for manual control. A useful feature of this function block is a dead zone that acts at the input signal of the controller, attenuating the influence of measurement noise during the control phase.

![Fig. 5.1. Arrangement of circular buffer emulating dead-time element.](image)

One of the important parts of the implementation of the autotuning PID controller is the realization of the dead time element. The dead-time element has a double function in the autotuning algorithm; it is used for the relay experiment with adjustable dead time and for the FPPI controller during the control phase. It, therefore, requires flexible implementation. Dead time is implemented as a circular buffer with two index-pointers pointing at the input (head) and at the output (tail) of the dead time element, and where each cell of the buffer holds data for one sampling step. Since the array containing a
circular buffer has to be declared in advance and because PLC has limited memory, the size of
the array is limited. This limitation is simulated through the use of modulo function after each
increase in value of the index-pointers at each sampling step. Consequently, dead time that can
be simulated with such software block is limited. This fact should be taken into account in the
selection of sampling time. Namely, too short a sampling time might require too long an array.
The dead time value is adjusted through by changing the difference between head and tail index-
pointers in the number of cells. Figure 5.1 shows a circular buffer emulating dead time element
before and after the wrapping of the input index-pointer caused by modulo function.

This representation of adjustable dead time element enforces a different form of iterative
equation for adjusting dead time (3-24) during the relay experiment:

\[ T_{ni+1} = T_{ni} + c \left( \frac{\pi}{4\omega_i} - T_{ni} \right), \]

where \( i \) denotes the number of the iteration, \( T_{ni} \) is dead time at the \( i^{th} \) iteration, \( \omega_i \) is the
oscillation frequency at the \( i^{th} \) iteration, and \( c \) is weight term. It is changed into the equation for
the calculation of required difference \( k_{i+1} \) of cells between the head and tail index-pointers by
dividing the values denoting dead time and frequency by sampling time \( T \):

\[ k_{i+1} = \text{int}\left( k_i (1 - c) + \frac{k_{osc,i}}{8} c \right), \quad (5-1) \]

where \( k_{osc,i} \) is the oscillation period represented by the number of sampling steps at the \( i^{th} \)
iteration and \( k_i \) is the required difference at the \( i^{th} \) iteration. The first value \( k_0 \) required for
iteration is determined from the results of preliminary identification as:

\[ k_0 = \text{int}\left( \frac{\pi}{4} \frac{T_2}{T} \right). \quad (5-2) \]

The described autotuning algorithm and the adaptations required for the PLC implementation
was downloaded to SIMATIC PLC CPU 314IFM (SIEMENS, 1997b). The minimum sampling
time in which the controller was able to perform the required actions was 20 ms. Since the
algorithm requires changeable sampling time, which is not supported in STEP 7 programming
environment, it was emulated with the waiting loop.
5. Implementation and experiments with the autotuning PID algorithm

5.2 Experiments with laboratory processes

The first laboratory process on which the autotuning controller was tested is a laboratory blower process PT326 produced by a company FEEDBACK (FEEDBACK, a). The main parts of the laboratory process are the heater, fan, air influx valve, temperature sensor, and measurement electronics (Fig. 5.2.).

The behavior of the process can be approximated with the FODT model with parameters in the range $K_1=0.5 \div 1.2$, $T_1=0.4s \div 0.8s$, and $T_{t1}=0.15s \div 0.3s$. These parameters were obtained through the step-response identification experiment. The exact values of these parameters depend on the operating point, defined by the opening of the air influx valve, by environment temperature, and by the position of the temperature sensor. The values of the input signal $u(t)$, which sets the power of the air heater, may range from 0 to 10 V.

![Fig. 5.2. Schematic layout of the laboratory blower process.](image)

![Fig. 5.3. Autotuning experiment with the laboratory blower process; the air influx valve is set to $50^\circ$.](image)
The range of the output signal $y(t)$, which corresponds to the measured temperature keeps between $-2$ V and $12$ V. The process is very sensitive to air movements, represented by low frequency disturbances. It is usual for the noise signal with the amplitude $0.15$ V to be present in the measured output signal.

Figure 5.3 shows the results of the experiment with the laboratory blower process when the air influx valve opening was set to $50^\circ$ and Figure 5.4 when the air influx valve opening was set to $70^\circ$. The results of these experiments are summarized in Table 5.1.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>noise band[V]</th>
<th>oscillations parameters</th>
<th>process parameters</th>
<th>PID controller parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ampl [V]</td>
<td>per [s]</td>
<td>type</td>
<td>$K_I$</td>
</tr>
<tr>
<td>1. Fig. 5.3.</td>
<td>0.10</td>
<td>0.51</td>
<td>0.60</td>
<td>FODT</td>
</tr>
<tr>
<td>2. Fig. 5.4.</td>
<td>0.05</td>
<td>0.53</td>
<td>0.74</td>
<td>FODT</td>
</tr>
</tbody>
</table>

Table 5.1. Summarized results of autotuning experiments with the laboratory blower process.

In both tests autotuning controller successfully performed the relay experiment, identified the process parameters, and tuned the PID controller. The adjustment of the relay during the experiment lasted only four iterations, confirming the quick convergence of the procedure. The controller parameters during the autotuning procedure show that the controller behavior adapts to the process parameters. This can be observed in the fourth and the fifth column in Table 5.1. The estimated parameters of the process are in the range of parameters obtained by step-response identification methods. A change in the air influx valve opening was correctly identified as a decrease in static gain $K_I$ and increase in time lag $T_I$. In addition, these experiments
demonstrated that the controller was able to complete the procedure in spite of a strong noise signal, exhibiting thus good robustness. The dead zone of the PID controller attenuated the effect of noise on its behavior. These observations confirm that the PLC implementation of the autotuning algorithm can be successfully applied for the control of processes that can be modeled with the FODT model.

The second set of tests involved the ‘PCS 327’ process simulator (FEEDBACK, b). It consists of several operational amplifiers, summing elements, and a dead time element. In addition, it has a non-linear unit for simulating elements with non-linear characteristics such as backlash, hysteresis, and deadband. It can be configured to simulate various combinations of time lags, integrators, non-linear elements, and dead time. The noise signal was very weak in the output signal of this process simulator.

The laboratory process simulator was set to simulate a process with an integrator and a time lag (2-3) with parameters $K_2=1$ and $T_2=1$ s. Figure 5.5 shows the behavior of the control system.

![Figure 5.5. Response of the autotuning PID controller and the type 1 process.](image)

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Ampl [V]</th>
<th>per [s]</th>
<th>$K_P$</th>
<th>$T_I$</th>
<th>$T_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 5.5.</td>
<td>1.05</td>
<td>7.14</td>
<td>0.65</td>
<td>3.42</td>
<td>0.76</td>
</tr>
<tr>
<td>KLV method with $\beta=1$; according to eq. (2-71)–(2-73)</td>
<td>0.62</td>
<td>5.00</td>
<td>0.80</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2. Results of autotuning experiments with the type 1 process.
The preliminary identification procedure correctly identified the type of the process. Followed the relay experiment with adjustable dead time. It took five iterations of the relay and dead time adjustment to attain limit cycle oscillations with desired parameters. The PID controller was then correctly set according to KLV tuning method. This can be verified by comparing the parameters of the PID controller tuned during experiment with the parameters obtained by the calculation (the last row in Table 5.2). The responses of the tuned PID controller to set-point change \((t=163s)\) and load disturbance \((t=184s)\) are described at the end of the experiment (Fig. 5.5). The overshoot of 19% and the rapid compensation of load disturbance illustrate the acceptable performance of the tuned controller in response to set-point change.

\[ G_p(s) = \frac{e^{-s}}{(3s+1)(s+1)(0.5s+1)}, \quad (5-3) \]

The following test involved the experiment with the process:

![Graphs showing system response](image)

Fig. 5.6. Response of the autotuning PID controller and the type 0 process, defined by equation \((5-3)\).

<table>
<thead>
<tr>
<th>oscillations parameters</th>
<th>process parameters</th>
<th>PID controller parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment</td>
<td>ampl [V] per [s]</td>
<td>Type</td>
</tr>
<tr>
<td>Fig. 5.6.</td>
<td>0.51 3.93</td>
<td>FODT</td>
</tr>
</tbody>
</table>

Table 5.3. Results of autotuning experiment with the lab process simulator defined by equation \((5-3)\).
The performance of the autotuning controller was acceptable; the relay adjustment was short and the obtained PID controller displayed satisfactory behavior, although a bit too oscillatory. This controller can be employed for similar stable, non-oscillatory type 0 processes.

The above experiments show that the autotuning algorithm in the PLC implementation completed all tasks successfully and with acceptable control behavior. The preliminary identification procedure properly detects the type of the process and selects the type of the relay experiment accordingly. The relay experiments converge to limit cycle oscillations with desired parameters after a few iterations of the adjustment procedure. To sum up, the controller proved to be robust. Moreover, it demonstrated the capability to tune the PID controller parameters for processes with models not incorporated into the algorithm.

The above tests suggest that the implemented autotuning controller is capable to control processes with similar dynamic characteristics.
6 CONCLUSION

This paper sought to describe the development and implementation of the autotuning PID controller. The development consisted of several phases: the review of PID control, study of autotuning controllers, extension of the existing autotuning algorithms, and simulation. Simulation provided the necessary understanding of the algorithm structure. The algorithm is implemented as a stand-alone software module in an industrial PLC.

The proposed extension of the autotuning algorithm, that is, the preliminary identification, uses the phase of reaching the operating point for crude estimation of the process parameters by employing the least-squares procedure. Besides, it is used to detect the type of the process, which enables automatic adjustment of the autotuning procedure to the detected type. Simulation and experimental results show that the inclusion of preliminary identification in the autotuning algorithm shortens the time required for the relay experiment. Together with the data obtained in the relay experiment, it improves the accuracy of the estimated process parameters required to design the controller.

Another benefit of employing preliminary identification is the possibility to estimate the process dead time, when the process can be modeled with the FODT model. The estimated dead time is compared with the estimated dominant time constant of the process. Dead time compensating controller is set to operation if the comparison shows that dead time exceeds the dominant time constant. A filtered predictive PI (FPPI) controller proved to be a suitable choice for a dead time compensating controller. It is very robust and can tolerate inaccuracies in the autotuning algorithm. The simulation further demonstrated good control properties of the autotuning procedure with the FPPI controller.

The implemented PID autotuning controller was tested on several laboratory processes. The experimental results confirm good properties of the autotuning controller observed during simulation. The controller was tested on several laboratory processes that matched the models incorporated in the controller. It successfully tuned controller parameters in all cases. Furthermore, other tests confirmed that it is able to control processes that do not match model processes incorporated into the controller and which can be approximated with these models. The above suggests that the extended autotuning controller may find application in processes with similar dynamical characteristics and that the controller is suitable for an industrial application.

This controller should not be applied to control processes with strong non-linear characteristics. Preliminary identification presumes response of the linear system and large signal non-linearities in the response of the controlled process would induce errors in the procedure such as determination of the type of the process which would lead to incorrect behavior of the algorithm. This problem of the autotuning controller could be solved with further improvements of the algorithm’s supervision shell that would detect any irregular behavior.

Although the autotuning controller is based on approximate identification and tuning methods, it shows good control properties. It successfully performs the relay experiment, estimates the process parameters and tunes the PID controller for different processes.
Furthermore, it demonstrates adequate robustness of the procedure in the presence of noise. To conclude by referring to my introduction, this study achieved its objectives.

There are different possibilities to rework and extend this controller. One way would be to broaden the class of processes – such as processes with oscillatory modes and unstable processes – to which the controller could be applied. Moreover, the controller should be extended in such way as to be able to control multi-input multi-output (MIMO) processes. Many authors proposed different autotuning procedures for the MIMO processes, but most of these procedures require a complicated experimental setup. The problem is that the idea of the relay experiment can not be directly transformed for use with the MIMO processes.

The review of several commercial adaptive PID controllers, given in section 3.4, showed that these controllers have incorporated different adaptive features. Adaptive features such as gain scheduling and self-tuning should also be included in the upgrade of the autotuning controller.


7 LITERATURE


ABSTRACT

This paper presents the development and implementation of the autotuning PID controller and reviews the properties of PID as the basic control algorithm and of different PID tuning methods.

The study delves into the original autotuning algorithm and its modifications, and proposes the introduction of the preliminary identification and use of dead time compensating controller within the autotuning framework. The author derives the procedure of the phase of preliminary identification and checks the extensions through simulation. The paper describes in detail the successful performance of the extended autotuning algorithm for various model processes and its implementation in an industrial PLC. The implemented autotuning PID controller was tested on several laboratory processes. The tests confirm the good performance of the controller already demonstrated by simulation.

Keywords: autotuning controller, dead time compensation, least squares method, PID controller, relay experiment, simulation, supervision shell

SAŽETAK

U ovom radu je prikazan razvoj i implementacija samopodesivog PID regulatora. Taj regulator koristi PID algoritam kao osnovni algoritam upravljanja. Obrađena su svojstva PID algoritma i razne metode podešavanja njegovih parametara.


Prošireni samopodesivi regulator implementiran je u industrijskom programirljivom logičkom kontroleru. Istaknuti su neki detalji implementacije. Implementirani samopodesivi PID regulator je provjeren na više laboratorijskih procesa. Ovi testovi su pokazali dobra svojstva regulatora, i time potvrdili rezultate simulacije.

Ključne riječi: PID regulator, kompenzacija mrtvog vremena, samopodesivi regulator, relejni pokus, metoda najmanjih kvadrata, nadzorna ljuska, simuliranje.
BIOGRAPHY

Ivan Branica was born on February 10, 1970 in Zagreb where he finished the elementary and secondary school.

In 1988, he enrolled in the Faculty of Electrical Engineering and Computing and spent that academic year serving in the army. In 1989, he started to attend courses at the FEEC, and in the 5th semester he took automatics as his major subject. He graduated on July 3, 1995 with a paper entitled “Recursive Identification of Process Parameters Using Instrumental Variable Method” (in Croatian: “Identifikacija parametara procesa rekurzivnom metodom pomoćnih varijabli”).

Since September 1, 1995 he has been working as a research assistant in the Department of Control and Computer Engineering in Automation of the FEEC. He started his postgraduate studies at the FEEC within the branch of Automatics in 1996.

Ivan Branica is the co-author of three scientific papers presented on international conferences and one scientific paper published in Croatia.

ŽIVOTOPIS


Ivan Branica koautor je tri znanstvena rada objavljena u inozemstvu i jednog znanstvenog rada objavljenog u domaćem znanstvenom časopisu.