IFAC PROFESSIONAL BRIEF

Hands-on PID autotuning: a guide to better utilisation

A. Leva leva@elet.polimi.it

Dipartimento di Elettronica e Informazione
Politecnico di Milano, Italy

C. Cox chris.cox@sunderland.ac.uk

Control Systems Centre, School of Computing Eng. and Technology
University of Sunderland, UK

A. Ruano aruano@ualg.pt

Centre for Intelligent Systems, Faculty of Science & Technology
University of Algarve, Portugal

Abstract

PID regulators are the backbone of most industrial control systems. The problem of determining their parameters, then, is of great importance in the professional control domain. To simplify this task and reduce the time required for it, many PID regulators nowadays incorporate *autotuning* capabilities, i.e. they are equipped with a mechanism capable of computing the 'correct' parameters automatically when the regulator is connected to the field. Due to the importance of the problem, a very wide variety of PID autotuners have been developed and are currently available on the market.

This monograph is aimed at control professionals who want to approach the problem of choosing and applying a PID autotuner in a knowledgeable and effective way. This goal is pursued by inducing an understanding of the theory of autotuner operation rather than by presenting an exhaustive menu of techniques and commercial products. As a consequence, this volume does not lead to a mere selection guide but to a set of concepts that, once mastered, allows the reader to evaluate the effectiveness of a specific industrial autotuner for a particular application. A significant space is also devoted to the review of PID control principles, essentially for less experienced readers. Some industrial products are presented, to illustrate how the concept introduced for classifying and selecting autotuners reflect in the real world, and samples of current research are given based on the authors' experience.

Foreword

This monograph is principally aimed at professional engineers who want to know how to choose and apply, much more effectively, one of the many PID autotuners currently available. In addition, it should also be of interest to anyone who would like to improve their knowledge of PID control and autotuners. The focus of the information is on how the various autotuners work. The primary aim is to provide the right kind of detail users require in order to select the methodology most appropriate for their current application. Let us begin by sketching out the monograph's organisation, in order to help the reader understand the rationale behind this particular Professional Brief.

Section 1 features some basic concepts including the necessary definitions to establish the terminology employed. Section 2 reviews the basics of PID control and pays particular attention to design issues and the formulation of a general transfer function that encapsulates many of the industrial PID controller structures often quoted in the technical literature. It is clear that this is an important prerequisite. However, because of space limitations this section is necessarily brief. If the reader feels completely unfamiliar with the content of section 2, then, it is recommended that the references, identified in the monograph, be consulted before addressing the core of the subject.

Once a firm grasp of PID control basics is achieved, Section 3 can be digested. This section explains how PID controllers can be tuned 'on site'. This introduces the importance of experimentation, especially when this leads to the derivation of a description of the process behaviour from measured I/O data. The rationale of Section 3 is to clarify that once a 'process description' has been obtained there are many methods available for determining the PID controller parameters. Another important feature is that at the end of this section the reader has all of the information required to tune the PID controller 'by hand'. This kind of knowledge should be invaluable later when assessing the various autotuner functionalities. In summary, at this stage the reader will have encountered several tuning methods, presented so as to be implemented by a human (not yet automatically). The main goal of this section is to understand the range and variety of the various tuning methods that are available to the practitioner.

Section 4 addresses the problems that arise when the tuning is automated and briefly describes some of the solutions that are commonly exploited in industrial applications. The importance of obtaining steady state information is emphasised, together with the need for signal conditioning to improve the accuracy of the process description. The expectation is that the understanding gained from this section should lead to a more informed choice if the reader has to select a particular system.

The focus of Section 5 is a (short) presentation of a sample of industrial autotuners. This review is not meant to be exhaustive (reference to product documentation is given when required), but rather to illustrate how the autotuner features previously considered and the proposed classification, reflect in the real world. If the goal of the monograph has been attained, the reader should now be capable of reading a product description and understand the advantages and disadvantages of that configuration when it is to be applied in a particular situation.

Sections 6 and 7 are devoted to brief descriptions of some 'research' autotuners, chosen to illustrate the authors' experience. Section 6 refers to research on 'classical' autotuning, while section 7 deals with the very promising alternative provided by the use of soft computing. The purpose of Section 8 is to provide a summary of the steps, covered in the earlier sections, that leads to a 'route map' that should be followed for the purpose of autotuner selection. In Section 9 some brief concluding remarks are reported.

1. Introduction

The basic concepts of (PID) autotuning

Many definitions of autotuning have been suggested. For the purpose of this volume, we prefer to define the object devoted to the autotuning operation rather than the operation itself. Thus, we say that

> *an autotuner is something capable of computing the parameters of a regulator connected to a plant (i.e. of tuning that regulator) automatically and, possibly, without any user interaction apart from initiating the operation.*

The autotuner may reside within the regulator or anywhere in the overall control system. No matter where it is, any regulator whose parameters can be computed automatically is said to have *autotuning capability*. Note that, strictly, the autotuner is not part of the regulator: when no autotuning is in progress, the computation of the control signal in no sense depends on the autotuner's presence. It is also sensible at this stage to distinguish between autotuning and *adaptation* or *adaptive control*. In the latter case, the regulator parameters are computed without operator intervention, while in the autotuning context the system may at best suggest the operator to retune, not initiate a tuning operation. Complete treatment of adaptive systems can be found e.g. in (Isermann et al., 1992).

In selecting an autotuner, then, the user is encouraged to read the associated documentation and establish when tuning occurs and how it is initiated. Under this point of view, we shall distinguish four cases.

(1) Tuning is initiated by the user as a deliberate decision, either explicitly or by making some manoeuvre which the documentation states to initiate a tuning (e.g. turning up power or modifying the set point). If this is the case, it is important that the connection between the manoeuvre and the tuning phase can be broken if desired, i.e. for example that the 'tune when set point is changed' option can be disabled in the regulator configuration and inhibited temporarily with some signal from outside the regulator, to prevent e.g. that necessary plant manoeuvres cause two interacting loops to begin a tuning operation together.

(2) Tuning is initiated by the user as a deliberate decision, but the regulator can suggest a retune. If this is the case, it is important that the suggestion logic be documented and configurable.

(3) Tuning occurs automatically when some condition occurs, e.g. the error becomes 'too big' for 'a certain time'. If this is the case, it is even more important that the logic be

precisely documented and configurable. Moreover, it must be possible to disable this functionality in the regulator configuration and to inhibit it temporarily from outside the regulator.

(4) Tuning occurs continuously.

Cases (1) and (2) are to be classified as autotuning, case (4) is clearly continuous adaptation, case (3) is somehow hybrid but, if the logic is properly configured, it is much more similar to autotuning than to continuous adaptation. It is important, when selecting an autotuner, to understand in which category it falls so as to forecast how it will possibly interact with the rest of the control system. It is also necessary to read the documentation carefully, since an expression like 'expert tuning mode', 'supervised self adaptation' or something similar, does not provide any information on the actual mechanisms involved.

In this work we shall focus primarily on PID autotuning for single loop situations and say almost nothing on continuous adaptation and more complex structures. This might appear a limitation, so a justification is in order. First, the statistics reported in (Control Engineering, May 1998) state that single loop controllers contribute 64% of all loop controllers, suggesting that "many customers still want a controller to handle only one or two loops so as to provide a more manageable process in case of failure". In addition, the two most desired features of a loop controller are the PID algorithm and "startup self-tuning", which in our terminology means autotuning initiated at power up. As such, becoming familiar with autotuning for single loop controllers means becoming proficient in a fundamentally relevant situation. Then, no matter how complex a plant is, control system optimisation is a skyscraper whose foundations are the low level loops. As such, learning to (auto)tune the individual loops means mastering the very basic fundamentals of the overall construction. Finally, discussing the issues we have chosen to omit would really lead far beyond the scope and extent of this work, so we must acknowledge that we are covering a small section of the competencies required for setting up control systems in the 'general' case.

An effective grasp of what we mean can be obtained e.g. from (EnTech, 1994), a short document that the authors strongly recommend to anyone involved in industrial control, or from (Bialkowski, 2000). However, accepting these limitations will allow for a self-contained discussion; some problems will necessarily be left open, and it is the authors' hope that the reader will be encouraged and stimulated by them. Some suggestions for further reading will be given, when useful, to guide those who desire a wider or deeper knowledge.

1.1. How autotuners work and how they can be broadly classified

To start understanding how an autotuner works, consider how a human would act when tuning a regulator in the field. Basically, he would (a) observe how the process behaves, maybe stimulating it somehow, and – more or less consciously – turn this knowledge into a *description of the process behaviour*, (b) convert his ideas on how the closed loop should function, taking into account the limitations of the process description determined in (a), into a *de-*

scription of the desired closed loop behaviour, (c) decide what the regulator parameters must be to achieve this desired behaviour. Formalising these steps will establish a procedure, which from now on will be termed a *tuning method*. Hence, an autotuner is an implementation of a tuning method made so as to be capable of running automatically. Let us now take a closer look at these steps.

1.1.1. The way process data is obtained and treated

Step (a) provides the *process data*. This can be done by stimulating the process deliberately or just observing how it behaves during normal operation. In the former case we have an *experiment based* autotuner, in the latter a *non experiment based* one. Experiment based autotuners can stimulate the process in various ways: in open or closed loop, with different signals injected in different places, and so on. Some approaches try to reduce the process perturbation by taking as 'experiments' normal manoeuvres, e.g. set point changes. Whatever method is adopted, it is necessary to ensure that process data are significant enough to allow regulator tuning. This is a very complex and critical problem. Suffice now to say that, quite intuitively, this problem is simpler in experiment based autotuners.

Once process data are available, they must be turned into a description of the process. To this end, two main directions can be followed. The first is to employ the data for obtaining the *process model*. An extremely wide variety of model types are used in autotuning, ranging from state space models to transfer functions, convolution models, stochastic models, up to neural networks and so on. To clarify the panorama, then, we can say that in the autotuning context

> *a model is something that can be simulated and, in so doing, reproduces the process data it has be drawn from with sufficient accuracy, so that it can be expected to capture the process behaviour precisely enough to allow forecasting that of the closed loop once the regulator is tuned.*

In this case, we have a *model based* autotuner. The other approach is to employ the process data immediately for the subsequent tuning. In this case we have a *non model based* autotuner. Also in the non model based case, a wide variety of process descriptions are used: points of the Nyquist curve, points or characteristic values of some relevant responses in the time domain (e.g. the static gain, overshoot and settling time of the step response) and so forth. In our description, the relevant fact is that these various representations cannot be simulated. Non model based autotuners can also be termed *characteristics based autotuners*.

1.1.2. The way specifications are accepted or produced

Step (b) corresponds to agreeing the *control specifications*. Clearly these can depend on what has been observed in (a), in that a given requirement can be realistic or not depending on the process physical characteristics and limits evidenced by its description. In particular, specifications must take into account what is the control effort required for their satisfaction. Specifications may involve

(i) requirements on the controlled variable's behaviour, typically expressed in terms of settling time, maximum overshoot, response time, rejection of disturbances, bandwidth and so on;

(ii) requirements on the control variable's behaviour, basically aimed at keeping the control energy as low as possible;

(iii) requirements on the loop degree of stability and robustness, typically in terms of a required phase, gain and/or magnitude margin;

(iv) constraints on the controlled variable, e.g. alarm levels;

(v) constraints on the regulator dynamics given by measurement noise, which typically calls for reducing the high frequency gain;

(vi) constraints on the control variable in terms of value and rate saturations.

It is worth pointing out that, for autotuner designers, control specifications are one of the most difficult issues. In fact, rigorously speaking, specifications should take into account not only the process information obtained in (a) but also the objectives of the tuning and the role of the regulator in the overall control system, which cannot be determined from experiments. This leads to some important considerations which are relevant not only for designing an autotuner but also for selecting one. Consider the following short discussion on the matter.

The tuning objectives

The goal of any autotuner is to achieve the 'best' control. However, it is often unclear what is meant by 'best control'. Any control engineer knows that there are a number of trade-offs when synthesising a regulator; for example, bandwidth and degree of stability are normally opposite desires. Autotuners can accommodate such trade-offs, but there exist other choices that must be made to orient the automatic tuning. Maybe the most important one is to decide exactly what the best control is. Low overshoot? Fast settling? Quick set point tracking? Efficient disturbance rejection? In addition, these desires often conflict with one another, and a good autotuner must be so flexible in accepting specifications to allow the skilled user to obtain exactly what he wants, yet remaining simple enough to avoid confusing less knowledgeable users and robust enough in the face of incorrect choices.

The role of the regulator in the overall control system

When tuning the regulator in a single, stand-alone loop, the main problem for the autotuner designer is given by the extremely variable level of understanding the users may have. For example, especially in very low-end products, so little user knowledge is commonly assumed that specifications must be generated entirely by the autotuner itself: this is normally achieved on the basis of the process descriptions, with methods too complex to be presented here. As a consequence, autotuners can accept specifications at very different levels of complexity: some do not require them at all, some provide the user with basic control, e.g. asking whether a 'fast' or 'low-overshoot' tuning is preferred, some allow the user to input numeric specifications, e.g. a desired settling time, directly. In synthesis, an important characteristic

of an autotuner is the *level of specifications control* it allows the user to have. It must be noted that the validity of user specifications must somehow be checked, thus that full control is normally available only on products conceived for complex and/or large scale control systems, where the presence of skilled personnel can be assumed.

On the other hand, tuning a regulator which is part of a more complex control system also requires tackling the problem of loop interactions. This has a number of consequences, starting from the fact that one has to decide the order in which loops need tuning. For example, in cascade control anybody would tune the inner loop first and then the outer with the inner closed, such that the outer bandwidth is sufficient for the correct process operation but lower than that required for the inner. However, such choices clearly require knowledge of the overall control system structure. It is then apparent that autotuner designers have to face some very tough problems in making their products usable. Quite naturally, the result is the availability of products that are more or less open with respect to the user control allowed, thus more or less powerful if well used and dangerous if not. When choosing an autotuner for a given application, especially if several interacting loops are involved, it is important to be able to select one that allows the right level of user control and to ensure that the human skills required for using that autotuner correctly are available.

1.1.3. The way parameters are computed

Step (c) corresponds to invoking a procedure having the process and desired closed loop descriptions as input and the regulator parameters as output. This procedure, then, implements what is commonly referred to as the *tuning rules*.

Tuning rules can be constructed in several ways. One requires a process model and resorts to choosing the regulator parameters so that the *expected closed loop behaviour* (that forecast using the process model) either be as similar as possible to that of a *reference model* or enjoy some time or frequency domain properties such as a prescribed settling time or a prescribed position of its poles. In the former case we have a *model based, model following autotuner* (the desired closed loop description being in fact a model), in the latter we have a *model based, non model following* one, which – since it imposes some characteristics of the closed loop and not a model for it - could also be called *model based, characteristics following*. Another possibility, not requiring a model, is to choose the regulator parameters so that some *expected closed loop characteristics* (computed using the process description, which is not a model) reflect the user's desires: an example of a 'characteristic' is the request that the open loop Nyquist curve contain a prescribed point, which is typical of relay based autotuning. In this case, we have a *non model based, characteristics following* autotuner. A third way to construct tuning rules is trying to emulate human reasoning. This leads to the so-called *rule based* tuning methods. In rule based tuning both the process behaviour and the desired closed loop behaviour descriptions can be in the form of a model or not, since human reasoning can be used (thus imitated) in any case.

All these approaches involve, more or less explicitly, solving some system of equations. This can be accomplished in many ways, from analytical methods to numeric approximations. In particular, characteristic-based tuning can lead to a number of techniques for pursuing its goals, including pattern recognition based methods and soft computing.

It is worth emphasising that the classification we are proposing is slightly different from that commonly adopted in the literature, which distinguishes only 'model based' and 'rule based' (auto)tuning: the former is when a process model is involved explicitly, the latter when no model is used explicitly and the tuning system tries to emulate human expertise (Åström and Hägglund, 1995, p. 237, 241). In our opinion, this classification is well suited for classifying autotuners under a *methodological* point of view but, especially if used by non specialists, may introduce some confusing elements. Hence, we prefer a description that reflects the *operational* aspects of autotuners more in detail and, though remaining simple and broad, allows us to point out clearly those differences that may not be methodological but have a significant relevance on what must be understood when using that autotuner. A final remark on the classification we are proposing is that all the steps of the autotuning process are tightly coupled: for model based autotuning a process model is in order, for time domain characteristics to be imposed the required process information must be available, and so on. Moreover, a certain tuning policy is more suited for some purposes than for others, e.g. pole/zero cancellation tends to produce sluggish transients for load disturbance rejection. It is then important, when choosing an autotuner, to understand at least the basics of the tuning rules it contains.

1.2. Where and why PID autotuning is useful

Before saying anything in this respect, it must be made clear that a sufficiently skilled human with sufficient process knowledge, data and time available can outperform any autotuner in any situation. Autotuning can provide a tremendous improvement in setting up and maintaining control systems provided that it is viewed as an aid to human skill, not a substitute for it. This improvement can take place in several directions, and in the following we shall point out the three major ones.

1.2.1. The importance of tuning PID loops correctly

Several studies report that the majority of regulators are mounted in the field and set into operation using their default parameters. This often results in poor operation. According to (EnTech, 1994), 80% of the control loops not only do not provide any benefit, but even increase variability. Of these situations, 30% are due to incorrect regulator tuning. As a consequence, many loops end up being left in manual mode. This implies at least two things. First, the majority of applications do no appear to be critical at least as far as stability is concerned. Second, which is far more relevant to us, the importance of having PID loops tuned correctly is often underestimated.

In fact, few loops are independent. A poorly tuned loop means more hassle for the upper levels of the control hierarchy, where (intuition should be enough for guessing this) interactions

are even more evident. Thus, having a loop tuned incorrectly often results in the necessity of taking more complex solutions at the higher hierarchy levels or of reducing the overall expectations.

This might seem obvious, but the authors have quite often came across situations where a few badly tuned PID controllers turned a simple problem into a very difficult one. In fact, when setting up the higher levels of the control system appears particularly difficult, almost certainly a part of the responsibility falls on some underlying loops. The problem is that these loops must be identified, and above all that the only way for avoiding problems is to tune every loop 'as well as possible', which is actually very time consuming. Autotuners allow the user to spot these problems automatically, to tune a loop quickly, and even to perform cyclic loop retuning for progressively improving the system performance. They can be used in the plant startup phase or during its operation, providing flexible tools for maintenance.

Of course, especially in large-scale systems, it is important that autotuners be highly integrated in the overall control and supervision system. The main goal of these remarks is then to clarify that, especially in complex systems, loop autotuning must not be considered an ancillary functionality, both when choosing a control product and when using it; rather, it is an important feature to observe when selecting a controller.

1.2.2. The importance of having standardised tuning policies

Repeatability is an issue in plant construction, commissioning and maintenance. It means clarity, less errors, reduced costs. In one word, it is necessary because also undesired situations are in some sense repetitive. Briefly, and intuitively, these problems can be substantially smoothed by the adoption of standardised procedures; in this respect, a systematic use of autotuning can be very useful.

1.2.3. The process knowledge that can be gathered from the use of autotuning

Finally, we would like to focus on an often neglected advantage provided not by autotuners directly, rather by the adoption of a systematic control system design and maintenance policy where they are used extensively and cleverly. Quoting from (Åström and Hägglund, 1995), p. 232,

> *"[…] poor behaviour of a control loop can not always be corrected by tuning the regulator. It is absolutely necessary to understand the reason for the poor behaviour. […] Remember–no amount of so-called 'intelligence' in equipment can replace real process knowledge."*

This can lead to two interpretations. First, autotuning must be employed with the awareness that invoking it unconditionally every time a loop behaviour is not satisfactory can actually hide process malfunctions. For example, if an actuator is progressively reaching an unserviceable condition, adapting the regulator over and over can keep the loop inside its operating

limits up to the moment when the actuator definitely trips. This is, by the way, one of the major reasons why continuous adaptation features must be used sparingly and wisely. On the other hand, every autotuning operation gathers process information and is a snapshot of process operating conditions. If process data are made available by autotuners to the plant information system, it is possible – and not difficult – to compare them during the various tuning operations.

Using autotuners systematically involves making many experiment on the plant. Besides tuning the regulators, these can produce a wealth of process information, thus refining the process knowledge by completing qualitative, human impressions with quantitative data. This requires that the autotuners employed be open towards the exchange of information and compatible in this respect with the total system. This degree of compatibility is then another useful parameter for selecting an autotuner.

2. The basics of PID control

Definitions, performance features and design fundamentals

2.1. Introduction

PID control research supports a massive literature. It would be impossible to do justice to all the published results in a volume such as this. In addition, because of varying definitions, assumptions and terminology it is sometimes difficult to make a direct comparison of two seemingly similar algorithms. A common result is that the improved features highlighted in a particular publication are not reproduced in the problem that one is examining. The aim of this section is to establish our notation, to agree and identify the different PID controller structure representations and to define and illustrate the key performance features that are regularly employed when quantifying the response of the final closed loop control system.

2.2. Controller Modes

The PID control law computes the control signal as the sum of three contributions, which are termed the Proportional (P), Integral (I) and Derivative (D) actions. Quite often the P, I and D actions are also referred to as 'control(ler) modes'.

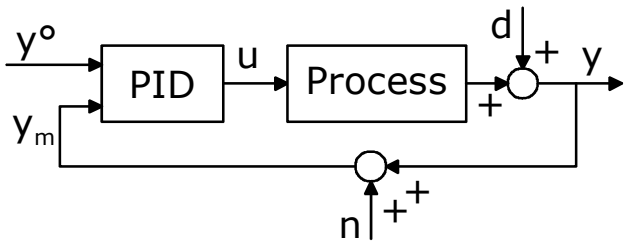


Figure 1: generic PID control scheme.

Throughout the volume we shall refer to the generic PID control scheme of Figure 1. Here y° is the reference signal, u is the control signal and is assumed to be limited by two bounds u_{min} and u_{max}, y is the controlled variable, y_m is the measurement of it fed back to the PID regula-

tor, d and n a are a disturbance and a measurement noise, respectively. Note that in a real control system there are usually more than two sources of disturbances and noise. However, for our purpose it is effective to classify all of them in the two categories suggested by figure 1. More precisely, then, we can state the following.

- There are disturbances that do not make y_m differ from y. These include all the actions on y (not y_m) other than the control signal u, thus correspond to physical actions on the system and not to phenomena that just affect measurements. For analysis and synthesis purposes all these disturbances can be treated as output disturbances, i.e. as d in figure 1. For example, if the process is described by a transfer function P(s), a load disturbance (i.e. an additive one on u) with Laplace transform $D_{load}(s)$ (we denote the Laplace transform of signals with the corresponding uppercase letter) is equivalent in figure 1 to an output disturbance D(s) with Laplace transform $P(s)D_{load}(s)$.
- There are disturbances that do make y_m differ from y. We can group all of them under the collective name of 'measurement noise' because noise is their most common source. These disturbances do not correspond to physical actions on the system; conversely, they account for the imperfections of measurement (whatever their reasons may be), signal transmission and so forth. All these disturbances can be treated as n in figure 1.

Quite intuitively, counteracting phenomena that affect a physical quantity and phenomena that only affect its measurement are different problems. That is why in the analysis and in the synthesis of a control system d and n are treated differently, as will be shown later. It is worth noting that in the literature there are several other ways to classify disturbances, and that in some products' documentation there appears to be some confusion on the matter. As a consequence, the reader is encouraged to master this (or any equivalent) classification of disturbances, because making wrong assumptions on how a disturbance must be counteracted may have a significant adverse affect on the controller (auto)tuning .

As for the error, some definitions are consequently in order to clarify notation. The control objective is to lead y (not y_m) to y°, so the 'true' error is defined as $e_t(t)=y^\circ(t)-y(t)$. However the regulator can only act on the measurement y_m of y, since n is by definition unknown (in fact one could give, for example, a stochastic description of it, but this is beyond our scope). Hence, we can define also an 'apparent' error $e_a(t)=y^\circ(t)-y_m(t)$. This is the 'error' the regulator will try to make as small as possible, and is a good representation of e_t if n is small. It is important to keep in mind the distinction between real and apparent error, because it has some important consequences that will be discussed later. Curiously enough, however, also this distinction is sometimes unclear, especially in the literature concerning less sophisticated products.

To minimise possible confusion, we shall adopt the following notation. When referring to the 'error seen by the regulator', which will be our point of view when describing all control laws and most tuning methods, we shall use the symbol 'e' and the reader must remember that this is the apparent error. When the distinction is necessary, we shall adopt the symbols

'e_t' and 'e_a' explicitly and the reader must remember that expectations are on e_t whilst available measurements are described by e_a .

A very common specialisation of the scheme in figure 1 is that shown as Figure 2. Here the process is assumed to be described by a linear, time invariant dynamic system in the form of the transfer function P(s) while R(s) is the transfer function of the (PID) regulator, which is considered linear as well. Moreover, the regulator input is the (apparent) error. This means that the regulator has 'one degree of freedom' (1-d.o.f.) in that the transfer functions from Y°(s) to U(s) and from Y_m(s) to U(s) differ only by sign, so it is not possible to specify how the control will react to a set point change and to a measurement change (i.e. for example to a disturbance) separately. Such regulators are frequently termed 'error-input' in the professional literature.

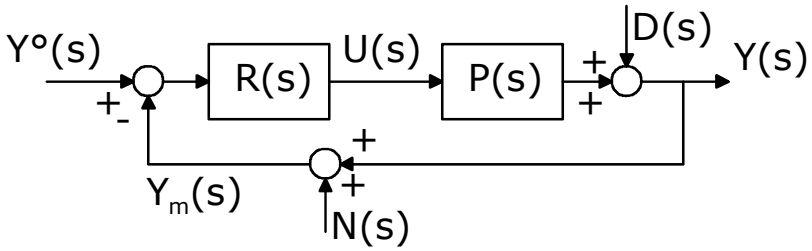


Figure 2: generic linear 1-d.o.f. (PID) feedback control scheme.

In the simplest PID regulator the control signal is then computed as

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt}$$

or equivalently (and with a more common notation) as

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int e(t)dt + T_d \frac{de(t)}{dt} \right) \quad (1)$$

which corresponds, when expressed in the form of the transfer function from e to u, to

$$R_{PID}(s) = K\left(1 + \frac{1}{sT_I} + sT_d \right) \quad (2)$$

and is frequently termed the 'ideal PID'.

2.2.1. Proportional Mode

The mode that is almost universally present is the proportional or P mode. With reference to (1), then, the control law in this case is given by

$$u_P(t) = Ke(t) + u_b \quad (3)$$

where $u_P(t)$ is the (proportional) controller output, i.e. the P action itself, K is the controller gain and u_b is a bias or reset value. The P action makes the control proportional to the error. Hence it obeys to the intuitive principle that, the bigger e is, the bigger the control action must be to lead y close to y°.

The P action depends only on the instantaneous value of the error and is nonzero only if e is nonzero. In other words, the P action is ideally zero at steady state, but only provided that the required steady state can be reached with zero control. If this is not the case it will be necessary to 'reset' u, i.e. to add a constant term to it so that it maintains the required steady state; if only the P action is used, this is the role of u_b. However, the reset can also be accomplished by the I action, and that is why the in older regulators this action is also called 'automatic reset'. Taking into account the control signal limitation it is possible to define the proportional band of the controller, i.e. the range of error values where u can be made proportional to e without exceeding the limits.

The proportional band P_b is then given by $(u_{max}-u_{min})/K$. Note that if K » 1 the control output is only linear for small errors. However, the proportional controller's response is instantaneous and (normally) produces a quick response from the process. Unfortunately the P action is very prompt in responding also to measurement noise. Hence it must be used wisely, since it can contribute to excessive actuator upset. Finally, note that the P action gives its main contribution at the beginning of transients; then, as y approaches y°, it tends to vanish (unless set point weighting is used in it, see later on).

2.2.2. Integral Mode

Integral (or reset) action produces a controller output that is proportional to the accumulated error. The control law in this case is given by

$$u_I(t) = \frac{K}{T_i}\int e(t) + u(0) \tag{4}$$

where T_i is the integral or reset time constant and u(0) is the value of the controller output when t=0. Note that u_I also depends on the controller gain. Because u_I is proportional to the 'sum' of the system errors, integral action is referred to as a 'slow mode'. Åström and Hägglund (1995) point out that integral action can also be viewed as a device that automatically resets the bias term u_b of a proportional controller. This follows immediately by considering that at steady state with y=y° the P action is zero except for u_b. In other words, the I action guarantees zero steady state error because, whenever e is the input of an integrator, there cannot be any steady state if e is nonzero.

The I action does not vanish with e; on the contrary, if e remains constant, it varies linearly obeying to the principle that, if y 'does not start moving' towards y°, the control action exerted must become stronger and stronger. Thus, the I action does not consider only the present e but also its past history, and that is another way of explaining how it provides the reset. Note that at any steady state with y=y° the control will be made only of I action (unless

set point weighting is used, see later). The I action is slower in responding to e and cannot have abrupt variations (being the state of an integrator). However it plays a crucial role in governing the way steady states are reached.

2.2.3. Derivative Mode

The final mode is the derivative (or rate) action. Here the control is proportional to the rate of change of the error signal. It follows that whenever the error signal is constant, the derivative signal contributes zero. The control law in this case is given by

$$u_D(t) = KT_d \frac{de(t)}{dt} \tag{5}$$

Where T_d is the derivative or rate time constant. Problems may arise when the error signal is entrenched in (high-frequency) noise or when step changes in the set point occur, since in these cases derivative action will generate large amplitude signals. However, most real systems have simple fixes to limit any 'harmful' effects *a priori*, such as imposing that the D action cannot provide more than a specified percentage of the overall control u. Derivative action is referred to as a 'fast mode' that generally improves the loop stability. It is often said that the D action 'anticipates the future'. This is another way of saying that it makes u depend on the direction and speed of the variations of e. In fact, with reference to Figure 3, it can be stated that the D action depends on the forecast variation of e 'T_d ahead': T_d then determines how far in the future this forecast is projected, while K acts as a further proportional factor between the forecast and the corresponding action.
The D action is the quickest to react (unfortunately, also to measurement noise) and helps only if the forecast is good, i.e. if T_d is not too big with respect to the time scale of the error dynamics (compare cases A and B in figure 3). That is why T_d must be limited and, because T_i is a measure of the closed loop time scale, T_d is normally requested to be 'quite a bit smaller' than T_i.

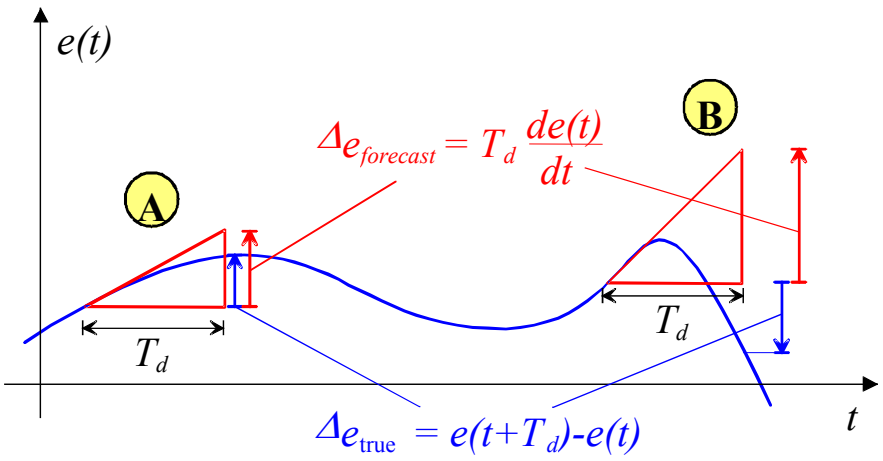


Figure 3: explanation of the role of the D action.

2.3. Control performance assessment in the time domain

The most natural way of assessing the performance of a controller is to formulate prescriptions on the transients it must produce. A variety of response characteristics can be used to this end, such as the most common ones indicated in Figure 4. Or, the requirements could be in the form that 'the closed loop response to a set point step must be as similar as possible to that of this model'. Thus, time domain assessment can be used both in model following and in characteristics following autotuners.

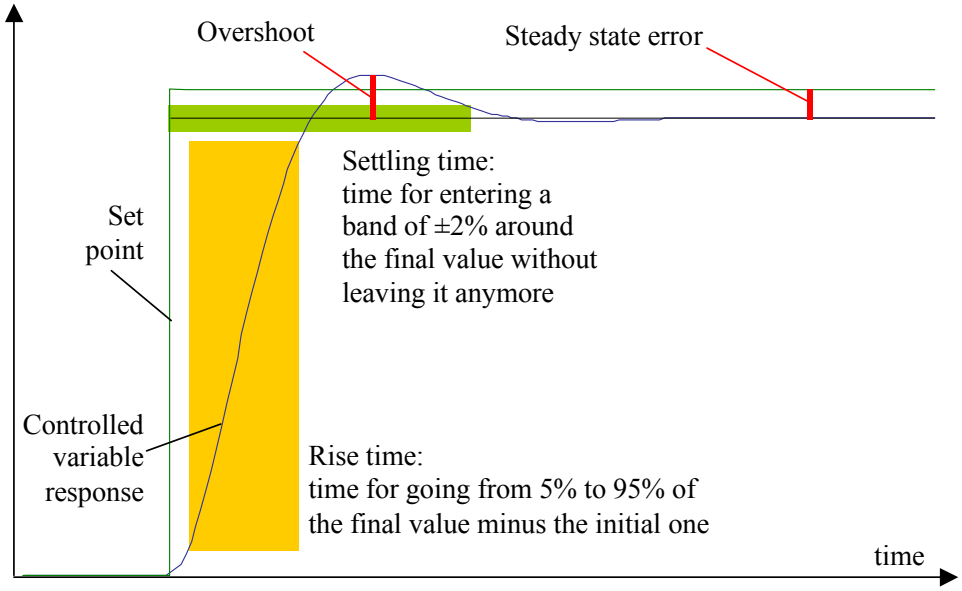


Figure 4: typical requirements for the closed loop step response in the time domain.

Quite intuitively, similar requirements could be made on the load disturbance response and/or on the transients of the control variable. We omit details for brevity, but it is worth noting that time domain assessment is very close to the way people not familiar with control theory tend to evaluate the control performance. As such, it is a very natural way of compiling a specification.

Time domain assessment, however, may not be easy to automate because it often implies recognising local characteristics of a response that can be compromised by spurious measurements. An alternative is to use integral indexes computed over time like the ISE (Integral of the Squared Error); these are treated later on in this work. In addition, characteristics like the degree of stability are easier to assess in the frequency domain.

2.4. Control performance assessment in the frequency domain

Assessing the controller performance in the frequency domain is maybe less intuitive but in general leads to more powerful analysis and synthesis tools, provided that the required process information is available. Moreover, in this context also the degree of stability and the re-

jection of noise and disturbances can be assessed within a unitary framework. The block diagram considered is that shown as figure 2, i.e. the process and the regulator are assumed linear and described by the transfer functions P(s) and R(s), respectively.

The most used media for frequency domain assessment are the Nyquist and Bode diagrams. Given a transfer function G(s), the associated Nyquist diagram is the plot, in the complex plane, of the image through G of the positive imaginary semiaxis, i.e. the locus defined by $G(j\omega)$, $0<\omega<\infty$. In closed loop control, it is particularly interesting to observe the Nyquist plot of the open loop transfer function $L(s)=R(s)P(s)$ - see figure 2 - because this allows two very important definitions to be made. These are the phase margin, PM and the gain margin, GM, illustrated in Figure 5.

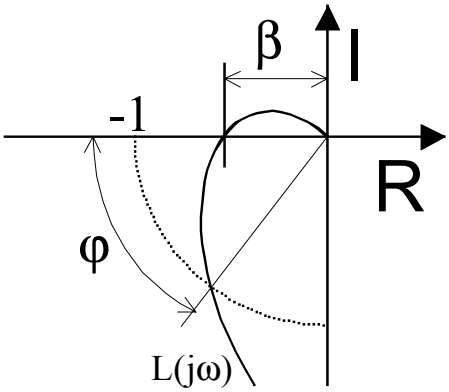


Figure 5: gain and phase margins as seen on the Nyquist diagram.

The GM is defined in dB as $-20\log_{10}(|\beta|)$, whilst the PM is φ and is normally expressed in degrees. The system is marginally stable when both GM and PM are zero. For a system to be stable both GM and PM have to be positive. The Bode diagrams are two graphs drawn to a base of $\log_{10}(\omega)$. If the frequency response of L(s) can be written as

$$L(j\omega) = |L(j(\omega)|e^{j\phi(\omega)}$$

then the vertical axis on one graph (the 'magnitude diagram') is the log-modulus, LM. The LM is expressed in dB and defined as

$$LM = 20\log_{10}|L(j\omega)|$$

The vertical axis on the second graph (the 'phase diagram') is simply the phase, $\varphi(\omega)$ normally expressed in degrees. Examples of Bode diagrams are presented as Figure 6, which also indicates how GM and PM appear. The frequency where $|L(j\omega)|=1$ is termed 'cutoff frequency' and indicated in figure 6 with ω_c. The frequency where $\arg(L(j\omega))=-180°$ is termed 'ultimate frequency' and indicated in figure 6 with ω_{180}. Note that a low-pass behaviour has been assumed for L(s), as suggested by physical considerations in any case of interest. Also, it has been assumed that ω_c is properly defined, i.e. that there is only one frequency where $|L(j\omega)|=1$. We omit theoretical details for brevity, but a proper definition of how ω_c can be achieved by correct regulator design is a required feature for any control system.

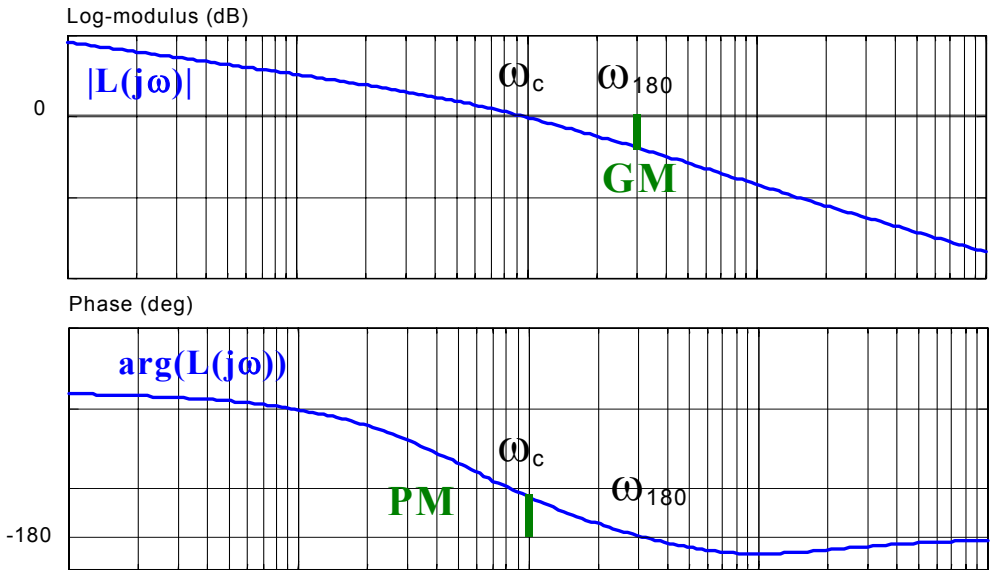


Figure 6: gain and phase margins as seen on the Bode diagrams.

The magnitude Bode diagram is widely used for regulator synthesis because many control specifications can be easily expressed using it. In fact, with reference to the scheme of figure 2, the transfer function from Y°(s) to Y(s) is L(s)/(1+L(s)), that from D(s) to Y(s) is 1/(1+L(s)) and that from N(s) to Y(s) is -L(s)/(1+L(s)). Hence for $\omega \ll \omega_c$ (i.e. where $|L(j\omega)| \gg 1$) $L(j\omega)/(1+L(j\omega)) \approx 1$ and $1/(1+L(j\omega)) \approx 1/L(j\omega)$, while for $\omega \gg \omega_c$ (i.e. where $|L(j\omega)| \ll 1$) $L(j\omega)/(1+L(j\omega)) \approx L(j\omega)$ and $1/(1+L(j\omega)) \approx 1$. This means that requirements in terms of response speed, disturbance and noise rejection can be expressed in terms of $|L(j\omega)|$ very naturally, as depicted in figure 7, while stability requirements are easily checked on the Bode phase diagram. In detail, if the closed loop dominant time constant must be between T_1 and T_2, if a disturbance D(s) in the band (ω_1,ω_2) must be attenuated at least by the quantity A_D (in dB) and if a noise N(s) in the band (ω_3,ω_4) must be attenuated at least by the quantity A_N (in dB), the Bode magnitude diagram of L(jω) must fulfil the constraints of figure 7.

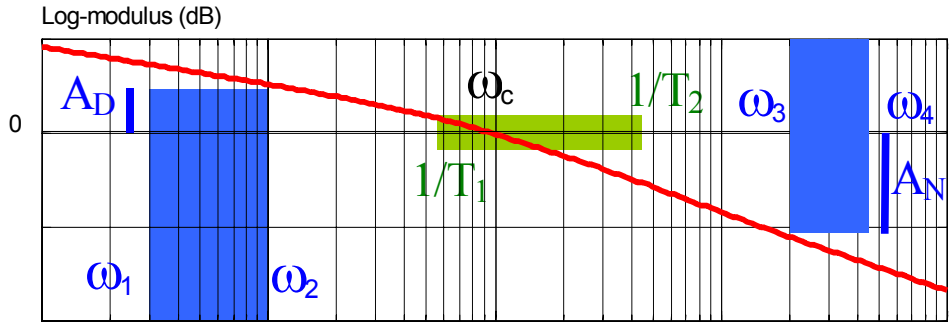


Figure 7: the Bode magnitude diagram as assessment tool.

Notice the different role of D(s) and N(s). Disturbances representing actions on the (physical) variable Y(s) can be counteracted by feedback if their frequency content does not extend

above the cutoff frequency ω_c and rejection is better if $|L(j\omega)|$ is bigger in that band. Noises only affecting the measurement $Y_m(s)$ can be counteracted if their frequency content is above the cutoff frequency ω_c and rejection is better if $|L(j\omega)|$ is smaller in that band. A response speed requirement means that $|L(j\omega)|$ must intersect the 0 dB axis in a given interval. This formalises and quantifies the intuitive idea that the control loop must exert strong feedback up to the cutoff frequency in order to follow the set point and reject disturbances, then introduce strong attenuation to prevent (high frequency) measurement noise from upsetting the system.

It is clear that frequency domain assessment can be used in model following autotuners, the desired closed loop description being in this case a desired $L(j\omega)$; Figure 7 has also sketched how it can be used in characteristics following autotuners, since the desired characteristics can be turned into *features* of a desired $L(j\omega)$. In any case, this approach requires a process model unless the decision is taken to assess the desired characteristics of $L(j\omega)$ on the basis of some conveniently measured points of its Nyquist diagram (this is typical of relay based tuning).

2.5. Modern design issues and the accommodation of plant uncertainty

Up to now it has been assumed that the process under control is perfectly described by a linear, time invariant model.

In the real world, this is rarely (not to say never) the case. The process model will only approximately capture the real plant behaviour because of time–variances, nonlinearities, unmodelled dynamics, sensor noise and unpredictable disturbances (Doyle *et al*, 1992; Rohrs *et al*, 1993; Goodwin *et al*, 2001; Åström and Hägglund, 2000). For these reasons we must be aware of how modelling errors will influence the process description and then, of course, the performance of the closed-loop control system.

These ideas have led initially to a range of investigations captured under the global heading of 'sensitivity analysis' but nowadays presented under the more general framework of uncertainty and robustness (Sanchez–Pena and Sznaier, 1998). Central to the new formulations is the concept of plant uncertainty described in terms of a nominal plant model plus a multiplicative or additive perturbation. This leads to definitions of robust stability and robust performance. Robust stability infers the capability of maintaining stability in the face of plant variations and modelling errors. Robust performance preserves a specified level of response in spite of plant variations and modelling errors.

The following sections are based on extracts from the books and papers mentioned above. The intention is to follow the framework introduced by Åström and Hägglund, 2000. The purpose is to provide the reader with basic knowledge of modern controller design and assessment methods, that can account for model errors and uncertainty in a more complete and rigorous way than could be done with more classical indexes such as the cutoff frequency

and the gain and phase margins. Control synthesis and assessment methods that account for model errors explicitly are often collectively called ‘robust control’. This means that they can ensure some properties (e.g. stability and performance) in a robust way, i.e. so that these properties continue to hold in spite of model errors, process variations and/or uncertainty provided these remain within quantified bounds.

2.5.1. The block diagram and some important transfer functions

The control systems we are dealing with are assumed to be characterised by the block diagram of figure 8.

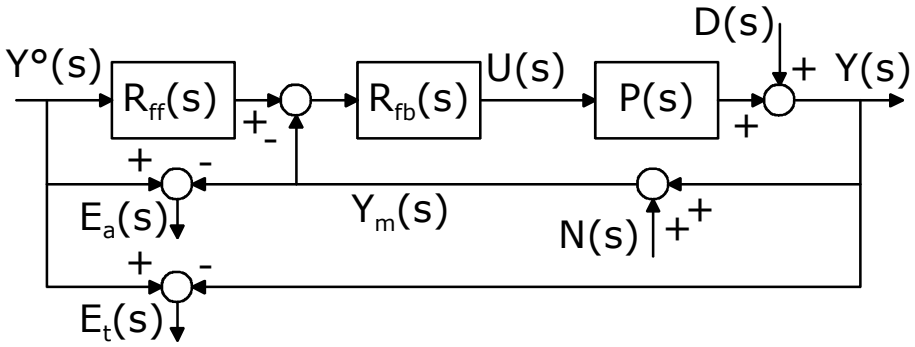


Figure 8: generic feedback control scheme highlighting the true and apparent errors.

In figure 8, the process is assumed SISO (Single Input, Single Output) and described as a linear system with transfer function P(s). We assume that P(s) is not perfectly known and that only an approximation of it is available. When relevant, we shall denote this approximation with $P_o(s)$ and with the term ‘nominal model’.

The controller is also linear but, notice, with two degrees of freedom (2-d.o.f.). In fact, the transfer function $R_{fb}(s)$ describes the feedback from the process output measurement $Y_m(s)$ to the control signal U(s) while the series of $R_{ff}(s)$ and $R_{fb}(s)$ represents the feedforward from the set point Y°(s) to U(s). Hence, in the 2-d.o.f. case, these two signal paths can be made different and the control reactions to the set point and to disturbances can be dealt with in a (partially) separate way. A 1-d.o.f. formulation is obviously obtained by setting $R_{ff}(s)=1$, see figure 2. Clearly, in the 2-d.o.f. case the open loop transfer function is defined as $L(s)=R_{fb}(s)P(s)$ and in nominal conditions it is $L_o(s)=R_{fb}(s)P_o(s)$. The choice of $R_{fb}(s)$ leads to the structure embraced by many of the texts introducing analysis and design methodologies at undergraduate level.

The relationships between the external input signals - Y°(s), the disturbance D(s) and the measurement noise N(s) - and the control system outputs - the process output Y(s), its measurement $Y_m(s)$ and the control signal U(s) - can be expressed in nominal conditions by

$$\begin{bmatrix} Y(s) \\ Y_m(s) \\ U(s) \end{bmatrix} = \begin{bmatrix} S_o(s) & -T_o(s) & R_{ff}(s)T_o(s) \\ S_o(s) & S_o(s) & R_{ff}(s)T_o(s) \\ -C_o(s) & -C_o(s) & R_{ff}(s)C_o(s) \end{bmatrix} \begin{bmatrix} D(s) \\ N(s) \\ Y°(s) \end{bmatrix} \quad (6)$$

The transfer functions $S_o(s)$, $C_o(s)$ and $T_o(s)$ are called the nominal sensitivity, the nominal complementary sensitivity and the nominal control sensitivity, respectively, and are defined as follows:

$$S_o(s) = \frac{1}{1 + R_{fb}(s)P_o(s)}, T_o(s) = \frac{R_{fb}(s)P_o(s)}{1 + R_{fb}(s)P_o(s)}, C_o(s) = \frac{R_{fb}(s)}{1 + R_{fb}(s)P_o(s)} \quad (7)$$

2.5.2. Stability definitions

The classical concept of stability of a linear time invariant system described as a transfer function is well known. This will be defined here as input-output stability and demands that the roots of the nominal characteristic equation $1+R_{fb}(s)P_o(s)=0$ all lie in the left half of the s-plane.

The second concept is that of internal stability. The requirement for internal stability is that the nine transfer functions in (6) are stable. This corresponds to requiring that the nominal system is input-output stable and also that in it there is no pole/zero cancellation in the right half plane.

2.5.3. Frequency domain design definitions for SISO systems with plant uncertainty

The frequency domain design of robust feedback control systems involves fitting some conveniently chosen frequency responses to constraints derived from specifications and from some quantification of uncertainty and model errors.

To this end, the functions $S_o(s)$ and $T_o(s)$ defined in (7) play an important role. Let us have a look at them under this perspective, assuming that both P(s) and L(s) have a low-pass aspect and considering for simplicity the 1-d.o.f. case, i.e. $R_{ff}(s)=1$.

With these hypotheses, in nominal conditions the transfer function from Y°(s) to the *true* error $E_t(s)$ is $S_o(s)$, that from D(s) to $E_t(s)$ is $-S_o(s)$ and that from N(s) to $E_t(s)$ is $T_o(s)$. Moreover, the transfer function from Y°(s) to Y(s) is $T_o(s)$, that from D(s) to Y(s) is $S_o(s)$ and that from N(s) to Y(s) is $-T_o(s)$, see (6) and figure 8.

Generalising the facts stated when talking about Bode diagrams (see figure 7), three frequency ranges of importance emerge as shown in Figure 9.

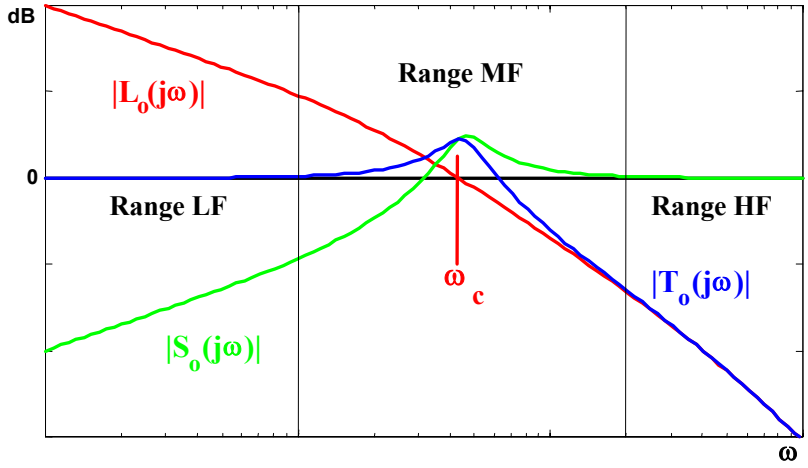


Figure 9: important ranges in frequency domain design.

- At low frequencies (range LF) the disturbance D(s) will be attenuated, while the noise N(s) will pass through on the controlled variable (thus on the error) with no attenuation. Recall that in this frequency range $|L_o(j\omega)|\gg 1$, hence $S_o(j\omega)=1/(1+L_o(j\omega))\approx 1/L_o(j\omega)$ and $T_o(j\omega)=L_o(j\omega)/(1+L_o(j\omega))\approx 1$.
- At high frequencies (range HF), conversely, the noise N(s) will be attenuated while the disturbance D(s) will pass through. Recall that in this frequency range $|L_o(j\omega)|\ll 1$, hence $S_o(j\omega)=1/(1+L_o(j\omega))\approx 1$ and $T_o(j\omega)=L_o(j\omega)/(1+L_o(j\omega))\approx L_o(j\omega)$.
- In the boundary area between these frequency ranges, i.e. around the cutoff frequency (range MF), there is usually some amplification of both noise and disturbance.

As for the set point, its frequency components below the cutoff frequency will pass through on the controlled variable (making the error small in that band) while those at higher frequencies will be attenuated (allowing a larger error in that band), see again figure 9. This is, by the way, another explanation of the connection between cutoff frequency and response speed. From these considerations, we will now show that performance and robustness specifications can be given in terms of desired shapes of $|S_o(j\omega)|$ and $|T_o(j\omega)|$ in a very straightforward and expressive way.

Nominal performance

The first issue to deal with is nominal performance, which means requesting that in nominal conditions (i.e. without model errors, disturbances and uncertainty) the error produced by any set point signal belonging to a characterised set be ‘smaller’ than a prescribed amount. One way of characterising such a set is to say that the set point signal has finite energy and that its frequency distribution is quantified by a function $W_1(j\omega)$. More precisely, this means that the set point signals of interest are all both Fourier- and Laplace-transformable, which implies that the two transforms coincide on the imaginary positive semiaxis, and that denoting these transforms by $Y^\circ(j\omega)$

$$\left|Y^{\circ}(j\omega)\right| \le \left|W_1(j\omega)\right| \quad \forall\omega$$

holds. Requesting that, in nominal conditions, any frequency component of the set point signals of interest produce an error frequency component with amplitude (in the corresponding unit of measure) less than unity simply means imposing that

$$\left|S_o(j\omega)\right| \le \left|\frac{1}{W_1(j\omega)}\right| \quad \forall\omega\,.$$

This condition can also be understood visually by looking at figure 9 and stating that 'shaping' $|S_o(j\omega)|$ in the frequency range where any set point signal of interest may have components is a performance specification because the plot of $|S_o(j\omega)|$ expresses how much the frequency components of the set point are attenuated on the error. Of course requesting that the error frequency components' amplitude be less than the unity is purely conventional: for a tighter constraint on the error it is enough to select a proportionally larger $|W_1(j\omega)|$.

It follows, then, that the designer can specify performance by defining desired the shape of $|S_o(j\omega)|$. Moreover, from the definition of $S_o(j\omega)$, the previous inequality may be expressed as

$$\left|W_1(j\omega)\right| < \left|1 + L_o(j\omega)\right| \quad \forall\omega\,. \tag{8}$$

In words, this inequality will be satisfied if a circle of radius $|W_1(j\omega)|$ centred on the point -1 of the s-plane does not include the point $L_o(j\omega)$, for all ω. Equation 8 specifies necessary and sufficient conditions for 'nominal performance', see Doyle et al (1992) for a more extensive discussion.

Robust stability

A similar condition can be derived for robust stability. According again to Doyle et al (1992), a controller provides robust stability if it provides internal stability for a family of plants that embrace all of the conceived plant variations and modelling errors.

To define robust stability it is then necessary to introduce the idea of model uncertainty. This is the 'difference' between the nominal model and the true system. Model uncertainty can be described either as an additive or multiplicative perturbation. For the latter case the true (and unknown) plant transfer function P(s) can be written as

$$P(s) = P_o(s)\left[1 + \Delta(s)W_2(s)\right] \tag{9}$$

where $P_o(s)$ is the nominal model, the weighting function $W_2(j\omega)$ provides a means of quantifying the 'frequency distribution' of the model uncertainty (Sanchez-Pena and Sznaier, 1998) and $|\Delta(j\omega)|<1$. It is worth noting that for a rigorous treatment of this matter some further technical hypotheses are necessary. For example, it is required that P(s) and $P_o(s)$ have the same number of right half-plane poles for any $W_2(s)$ and admissible $\Delta(s)$ and that there be no right half-plane cancellations between $R_{fb}(s)$ and P(s) for any $W_2(s)$ and admissible $\Delta(s)$, see Doyle et al (1992) for a discussion that we omit for brevity.

Here we prefer to express the model error as a multiplicative perturbation because it allows one to write

$$L(s) = L_o(s)\left[1 + \Delta W_2(s)\right]$$

where $L_0(s)=R_{fb}(s)P_0(s)$, which simplifies computations. This information can be used to assess closed-loop stability in the presence of plant uncertainty. Assuming the nominal loop, i.e. the one described by $L_o(s)$, to be internally stable, it turns out that the robust stability condition is

$$\left|L_o(j\omega)\left[1 + \Delta(j\omega)W_2(j\omega)\right] - L_o(j\omega)\right| < \left|1 + L_o(j\omega)\right| \ \forall\omega, \ \forall\left|\Delta(j\omega)\right| < 1$$

i.e.

$$\left|T_o(j\omega)\right| < \left|\frac{1}{W_2(j\omega)}\right| \ \forall\omega\,. \tag{10}$$

Also in this case, a visual explanation can be given. By the Nyquist criterion, if the nominal (closed) loop is stable and, for any $W_2(s)$ and admissible $\Delta(s)$, there are no right half-plane cancellations and $P(s)$ and $P_o(s)$ - thus $L(s)$ and $L_o(s)$ - have the same number of right half-plane poles, the perturbed (closed) loop is stable too iff the Nyquist plots of $L_o(j\omega)$ and of $L(j\omega)$ make the same number of turns around the point -1. If the perturbation is so big as to destroy stability, then, there must be at least one frequency $\bar{\omega}$ for which the corresponding point $L(j\bar{\omega})$ of the perturbed open loop Nyquist diagram is far away from the nominal one $L_o(j\bar{\omega})$ more than the distance from $L_o(j\bar{\omega})$ to the point -1. This means that, the larger $|W_2(j\omega)|$ is in a certain band, the further $L_o(j\omega)$ must stay from the point -1 to guarantee that stability is preserved.

In force of (10), this condition can be expressed on $L_o(j\omega)$ easily. In words, the inequality (10) will be satisfied if a circle of radius $|W_2(j\omega)|$ centred on the point $L_o(j\omega)$ of the s-plane does not include the point -1, for all ω; see again Doyle et al (1992) for a complete discussion.

Robust performance

The result of equations 8 and 10 can be combined to yield e.g.

$$\left|W_1(j\omega)S_o(j\omega)\right| + \left|W_2(j\omega)T_o(j\omega)\right| < 1 \quad \forall\omega\,.$$

If this is satisfied, it is guaranteed that the system achieves the required performance under nominal conditions, expressed with $W_1(j\omega)$, and that stability is preserved for any model uncertainty or perturbation not exceeding the limits expressed by $W_2(j\omega)$. It must be noted that the problem of combining equations (8) and (10) is far more complex than the very simplistic sketch presented herein, see Doyle et al (1992) once again. Nevertheless, this is the 'practical' condition usually specified for robust performance of SISO systems. This condition can be expressed graphically requiring that the two discs in figure 10 do not intersect for all ω.

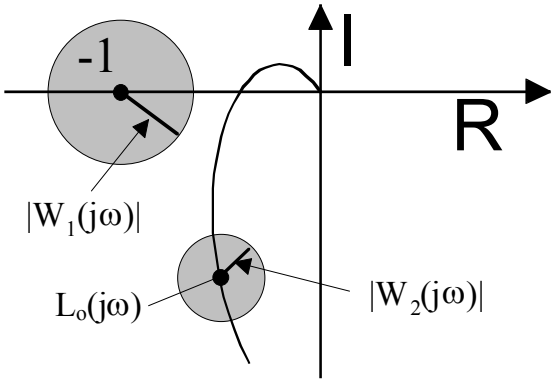


Figure 10: assessing robust performance on the basis of $S_o(j\omega)$ and $T_o(j\omega)$.

2.6. Concluding remarks on stability, performance and robustness assessment

It should now be apparent that the problem of assessing the stability, performance and robustness characteristics of a control system has been continually evolving since its first (intuitive) setting in the time domain and also since the introduction of the stability and performance indexes in the frequency domain (the phase margin and the cutoff frequency). The previous sections have just given a quick overview of recently established results; a complete treatment on the matter can be found with specific reference to the PID case in Åström and Hägglund (1995) and, more in general, in Doyle *et al.* (1992) and Morari and Zafiriou (1989). Under the restrictions of this volume, if the goal of the previous sections has been attained the reader should now be able to master the following concepts.

- Assessment in the time domain typically refers either to local characteristics of a response (e.g. the settling time or the overshoot of the closed loop step response) or to some integral index computed over time (e.g. the ISE) for a certain response. It is very intuitive for the non specialist user but sometimes tricky to automate. Autotuners using this approach may require no model of the process but, in this case, cannot provide rigorous guidance for modifying the specifications if the required result cannot be obtained. In addition, issues like the degree of stability or the attenuation of noise are not easily cast into this framework.
- Assessment in the frequency domain with classical indexes (the phase margin and the critical frequency) can only be understood by people with at least a minimum of theoretical control engineering knowledge. Autotuners using this approach can employ a process model or not, and only in the latter case are they able to forecast the effect of a modification in the requirements. In any case, all the results provided by such autotuners are based on the assumption that the process model (if any) is exact, so the only way to achieve some 'robustness' is to reduce the requirements on the cutoff frequency and/or to increase those on the phase margin *a priori*.

- Assessment methods accommodating model uncertainty are seldom used in autotuners because they require both a process model and some measurement of its uncertainty. Obtaining this information automatically is very difficult. However, if an autotuner reveals the model that has been employed for the tuning (some nowadays do), a knowledgeable user can determine the nominal system's characteristics and also quantify the amount of disturbances and model errors that can be tolerated.

According to the authors' experience, several people who tune regulators in the field appear to consider their parameters just as 'knobs to be moved' depending on what characteristic of the loop response needs improving. This may be enough in that context but does not suffice for evaluating an autotuner effectively. For understanding how an autotuner will behave it is necessary to abandon ideas like 'the regulator gain must depend on the step response overshoot', which are misleading because the entire regulator depends on the entire process dynamics, facts like the overshoot being just an external evidence of what these dynamics are.

From now on, to simplify the presentation, we shall employ the simplified scheme of figure 1 (instead of figure 8) and almost always refrain from analysing the effects of disturbance and noise and the difference between the real and apparent error. Nevertheless, we strongly encourage readers to become familiar with the subject of the preceding sections and to employ this knowledge for evaluating the results of tuning operations, or at least of critical ones. In fact, if the results obtained with a given autotuner have not been satisfactory, it is always a good idea to examine these results with the analysis methods presented.

2.7. Realistic PID Structures

Åström and Hägglund (1995) have covered PID control in great detail. They have explored different algorithms besides the ideal control law (2), provided detailed solutions to the integral wind-up problem and looked in some depth at several important operational scenarios. This brief section is simply intended to introduce three important themes that will be relevant in different parts of this book. Here we shall briefly concentrate on antiwindup, controller properness and set point weighting.

2.7.1. Antiwindup

It is perhaps unrealistic to assume that the system actuator will never ever hit an end-stop. One possible effect of such a constraint on the PID controller is that the integrator may 'wind-up' and produce a very large signal usually leading to poor dynamic system performance. The remedy is that the I action must never be allowed to exceed the control saturation limits. This is the basic principle of antiwindup, which can be implemented in several ways (a complete discussion would not fit in this volume). It is, in any case, a nonlinear regulator feature: that is why it is seldom considered in autotuning, being in general sufficient to rely on the antiwindup mechanism of the underlying regulator however it is implemented.

2.7.2. Controller properness

Another issue is that the D part of the PID controller in the ideal form (1) is not proper. To overcome this, it is commonly implemented as

$$U_D(s) = \frac{sKT_d}{1 + sT_d / N} E(s)$$

This is often referred to as 'using a real derivator'. In this way, N becomes another parameter of the PID that has to be selected. It is worth noting that a high N makes the implementation of the D action similar to a true derivative but it also increases the high frequency gain, thus increasing noise sensitivity.

2.7.3. Set point weighting in the P and D modes

Set point weighting in the P mode means that the proportional action is computed as

$$U_P(s) = K\left(bY^\circ(s) - Y_m(s)\right)$$

where b is a further parameter whose role is to limit the control step that may arise as a consequence of an error step. An error step, in turn, is most likely to arise as a consequence of a set point step, since the process response is rarely instantaneous. Some remarks are now in order. First, b has practically no influence as steady state is approached, since at this point it is the I action that dominates. Then, if $b \neq 1$ the P action is nonzero at steady state, since there $y=y^\circ$. Finally, since b can limit control bumps, it is particularly useful in the inner loops of cascade controls, where the set point is not under direct operator control. Parameter b is seldom considered in practice; several regulators do not encompass it, while others just permit one to select for it a value of 0 or of 1. Some advanced tuning techniques make use of b.

Set point weighing in the D mode means that the derivative action is computed as

$$U_D(s) = \frac{sKT_d}{1 + sT_d / N}\left(cY^\circ(s) - Y_m(s)\right)$$

where c is a further parameter whose role is to limit the control spike that may arise as a consequence of an error step, also with a proper controller. Here too, some remarks are useful. Since in general y° is seldom modified, especially in process applications, setting c=0 does not modify the controlled system dynamics significantly. In fact, c influences only the (few) instants when y° varies, because when it is constant its derivative is zero and computing the D action by deriving e or –y is the same. Clearly this does not hold e.g. for the inner loops of cascade controls, or whenever y° may vary continuously. Some advanced tuning techniques use c. Most often, however, it is set to zero leading to the so-called 'output derivation PID'.

2.7.4. The ISA PID

A number of PID structures have been proposed in the literature. In this volume we have to choose one for discussing general aspects, though we shall quote some of the others when

dealing with autotuners that use them. As a general form for the realistic PID regulator we choose the ISA one (Åström and Hägglund, 1995) due to its great generality. The general form of this control law is

$$U(s) = K\left[bY^{\circ}(s) - Y_m(s) + \frac{1}{sT_i}\left(Y^{\circ}(s) - Y_m(s)\right) + \frac{sT_d}{1+sT_d/N}\left(cY(s)^{\circ} - Y_m(s)\right)\right] \quad (11)$$

The notation is as previously defined with the inclusion of the set-point weights b and c in the proportional and derivative actions, already introduced. Similarly, the derivative part is made proper by adding a pole with time constant proportional to T_d via parameter N, as discussed above. Another way of seeing that these additional three parameters give added flexibility to the controller implementation is to notice that they correspond to the 2-d.o.f. realisation shown as figure 11

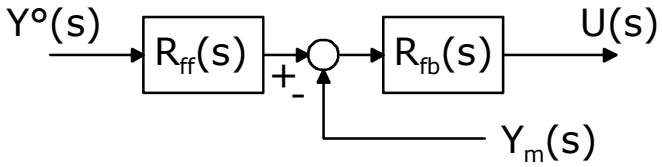


Figure 11: a PID regulator with 2 degrees of freedom.

where

$$R_{fb}(s) = K\left(1 + \frac{1}{sT_i} + \frac{sT_d}{1+sT_d/N}\right),$$
$$R_{ff}(s) = \frac{1 + s(bT_i + T_d/N) + s^2T_iT_d(c + b/N)}{1 + s(T_i + T_d/N) + s^2T_iT_d(1 + 1/N)} \quad (12)$$

Two facts are worth pointing out. First, $R_{fb}(s)$ is a real (i.e. made proper with N) 1-d.o.f. PID. Thus it can be tuned with virtually any method, including the (numerous) old ones that refer to this structure. Second, once N is fixed by tuning $R_{fb}(s)$, b and c can only modify the zeros of $R_{ff}(s)$. Hence, once stability and disturbance rejection have been dealt with in the synthesis of $R_{fb}(s)$, b and c can be safely used for improving the set point tracking.

The ISA form is a good structure for considering N, b and c as true regulator parameters and for taking profit from them in the synthesis phase. This is a quite modern issue, but some autotuners do start ‘reasoning’ this way. At present, there is considerable research effort on this and other associated matters.

As a final remark, note that the presence of the set point weights in the ISA PID can also be interpreted as a very specialised form of feedforward compensation, because the scheme of figure 11 can also be drawn as

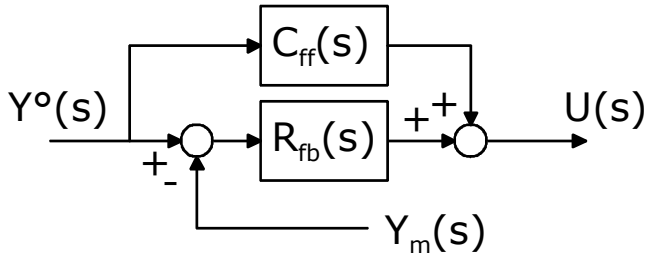


Figure 12: set point weighting viewed as feedforward compensation.

where

$$C_{ff}(s) = R_{fb}(s)(R_{fb}(s) - 1) = \cdots = K(b-1)\frac{1 + sT_d\left(\frac{1}{N} + \frac{c-1}{b-1}\right)}{1 + sT_d / N}. \qquad (13)$$

2.8. Two widely used extensions of the PID controller

As anticipated, many controllers are not applied to processes 'alone'. On the contrary, these controllers are assembled to form more complex structures like cascade control, feedforward/feedback schemes, and so forth. In this volume there is not the space for treating control structures, though this is a very important subject deeply connected with the recent research on autotuning: for a detailed discussion, the interested reader can refer to (Åström and Hägglund, 1995) and to the bibliography given therein.

Nevertheless, at least one control structure has become so popular (especially in process control, which is in some sense the preferred domain of autotuning) to be considered not really a structure surrounding a regulator but an extension of the regulator itself. This is the Smith predictor, used for controlling processes with significant delay and at present offered by several industrial controllers and autotuners.

The Smith predictor is then presented in the following together with a somehow simplified version of it called the 'predictive PI', which is also offered by several autototuners. The aim of this section is not to provide complete theoretical coverage on these extensions, rather to make the reader understand their rationale, possibilities and pitfalls, so as to be able of evaluating the operation of an autotuner encompassing them.

2.8.1. The Smith Predictor

The Smith predictor has been first proposed in (Smith, 1957). Since then it has been extensively applied to the control of processes with significant dead time, i.e. when in the process response the delay dominates the rational dynamics. The Smith predictor is shown in the scheme of Figure 13.

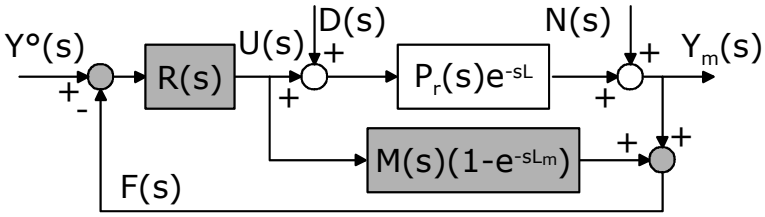


Figure 13: the basic scheme of the Smith predictor.

The rationale is as follows. Suppose the process is described by a rational model $P_r(s)$ cascaded to a delay L. If an (almost) exact approximation M(s) of $P_r(s)$ and an (almost) exact estimate L_m of L are available, then in the scheme of figure 13 the transfer function from the control signal u to the feedback signal f is very close (ideally, equal) to $P_r(s)$. Moreover, in the ideal case and without disturbances D(s) and N(s),

$$Y_m(s) = Y(s) = \frac{Y(s)}{U(s)}\frac{U(s)}{F(s)}F(s) = P_r(s)e^{-sL}\frac{1}{P_r(s)}F(s) = e^{-sL}F(s).$$

Hence, by using this scheme, the (PID) regulator R(s) can be tuned taking into account only the rational dynamics of the process, i.e. $P_r(s)$. Without modelling errors, noise and disturbances - which implies that $Y_m(s)=Y(s)$ - the resulting behaviour of Y(s) will be that of F(s) (which would be the controlled variable if there were no delay) just delayed by L. The name 'predictor' comes from the block $M(s)(1-e^{-sL_m})$, which actually generates the prediction F(s) of Y(s) compensating for the delay.

Of course the Smith predictor cannot counteract disturbances D(s) and N(s) simply because it cannot predict their effects, and it is not a robust scheme because all its rationale lies on the availability of a quite accurate process model. Nevertheless it is widely used especially in process control, so that several regulators and autotuners encompass it. In fact, as will be clarified later, adopting the Smith predictor is a viable way to extend a (model based) autotuner conceived for process with rational dynamics to cases where the delay is dominant. With a very crude simplification, once the rational model and the delay estimate have been obtained, it suffices to tune a PID on the former and then insert it in the scheme of figure 13, M(s) being the model used for the tuning and L_m the delay estimate. The main pitfall of this approach is that the model must be more accurate than normally required for model based tuning of standard PID regulators.

2.8.2. The Predictive PI (pPI)

A specialised version of the Smith predictor is even more widely employed in process control applications. Recalling the definition of $R_{fb}(s)$ in (12), the (1-d.o.f.) pPI regulator in its simplest form is given by

$$U(s) = K\left(1+\frac{1}{sT_i}\right)\left(Y°(s) - Y_m(s)\right) - \frac{1}{sT_i}\left(1-e^{-sL_m}\right)U(s), \qquad (14)$$

where L_m is an estimate of the process delay. This is a specialisation of the scheme in figure 13 with $R(s)=K(1+1/sT_i)$, i.e. a PI, and $M(s)=1/sT_i$. The pPI can also be interpreted as a standard PI - the first term in (14) - plus a correction based on the effects of the control action that have not yet appeared on the controlled variable due to the process delay. The resulting regulator block diagram is shown in Figure 14. Notice the similarity with the standard PI, that corresponds to the same scheme replacing the term e^{-sLm} with the unity.

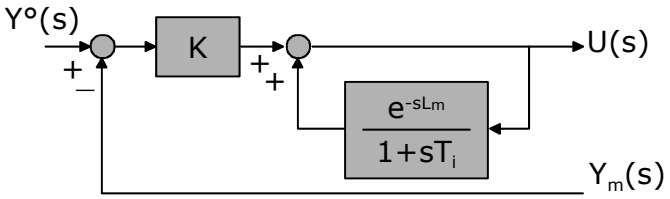


Figure 14: the basic scheme of the predictive PI (pPI).

Many variations of the pPI have been proposed, see e.g. (Åström and Hägglund, 1995). For our purposes it suffices to say that this scheme is a less powerful (but also less critical and easier to tune with commonly available process data) declination of the Smith predictor idea, often available in industrial regulators.

3. The basics of PID tuning

Obtaining simple process descriptions from data and computing the PID parameters

To select an autotuner, it is of great help if one is capable of tuning. Hence, after the review of PID control principles made in the previous section, here we shall illustrate the major PID tuning methods. These can be used by a human (and it is recommended that anyone involved in control have some experience of these) and, when automated, form the backbone of many of the more common autotuners. The tuning process as sketched out in the introduction will be the path followed in this section. We shall first describe how to obtain a description of the process behaviour and then how to compute the PID parameters on the basis of it and of the desired specification.

3.1. The extreme basics of data-based process description

The process descriptions used for autotuning must be obtained from I/O data. In the great majority of cases, a process description is obtained by performing an experiment on the process, i.e. by stimulating it *deliberately*. In other cases, it is obtained by merely observing the process input and output during normal operation. This approach is far less frequent because it requires one to make sure that the observed output behaviour is actually caused by the input, which is not easy and may require measuring additional signals. If there is no certainty in this respect, the identification mechanism may try to explain as an effect of the input some fact that is actually caused by something else, and conclusions on the process dynamics drawn in this way can be far from reality.

The simple structure of the PID regulator (especially in the 1-d.o.f. case) calls for simple process descriptions, so that first or second order models or small sets of characteristics are typically used. In this section we present a very brief review of some modelling and identification techniques used in autotuners, limiting the scope to experiment based methods since they cover almost the totality of cases and treating the other methods would really not be practicable. Since this presentation cannot be exhaustive for apparent reasons, we only introduce methods adopting the most common approaches, i.e. involving an open loop experiment, the recording of the so obtained response and the processing of this for obtaining the process description.

In most cases, a step response will be used. Step experiments are the most common strategy in experiment-based autotuners. For the reader willing to experiment, they are also easy to apply in the field: it suffices to switch the regulator to manual, wait until a reasonably steady state is reached, then change the control variable suddenly by an amount sufficient to make the response obtained easily distinguishable from noise. In addition, step tests permit the process to be maintained under reasonable control without perturbing it excessively or leading it to the stability boundary, as required e.g. by the closed loop Ziegler-Nichols method (treated later on). Finally, a step experiment lasts as long as the process takes to settle, thus it is one of the shortest ways of getting all the required information.

In some cases the step is applied in closed loop (to the set point or to the control signal), but this requires that a regulator be already present. Hence this is only used for refining an existing tuning or in autotuners with a ‘pretune’ mode. Closed-loop step experiments clearly provide a description of the closed loop system. That of the process must be obtained from it, which is a nontrivial task and is not treated in this work. For a more extensive discussion including closed-loop identification, frequency domain methods and so on, the reader can refer to Åström and Hägglund (1995) and to the references given therein, particularly in the bibliography of chapter 2.

Step experiments are very common both in model based autotuners (because it is straightforward to identify a model on the basis of a step response record) and in characteristic based ones, since several interesting features of the process dynamics emerge clearly from some features of the step response such as the apparent delay, the settling time and so on.

Another widely used method for stimulating the process is the use of random-like signals, e.g. a PRBS (Pseudo Random Binary Sequence). In practical applications this is common only when a model has to be identified by parameter optimisation techniques (see later). Finally, several experiment based autotuners use the so called ‘relay identification’. As will be explained later, almost any stable process subject to relay feedback enters a permanent oscillatory state, from which some characteristics in the time or (especially) in the frequency domain are straightforward to obtain. Relay feedback has the important feature that it does not require one to open the loop, which makes it particularly useful in some applications.

3.1.1. Model based process description

In the great majority of process loops, applying a step to the control variable causes the controlled variable to reach a steady state and does not provoke an instantaneous variation of it. This means that the process model seen by the regulator can be described by an asymptotically stable, strictly proper transfer function. In a few loops, a control step causes the controlled variable to asymptotically assume a ramp-like behaviour: this case is commonly referred to as 'runaway', 'integrating' or 'non self-regulating' processes and can be described by models with a pole at the origin of the s-plane. These facts are in good accordance with experience, since any practitioner would classify the step responses he may encounter more or less as depicted in Figure 15. Other cases (e.g. an oscillatory response with significant delay) may exist, but they are unlikely to appear in practice. For simplicity, in this section we do not consider noise and disturbances unless explicitly stated.

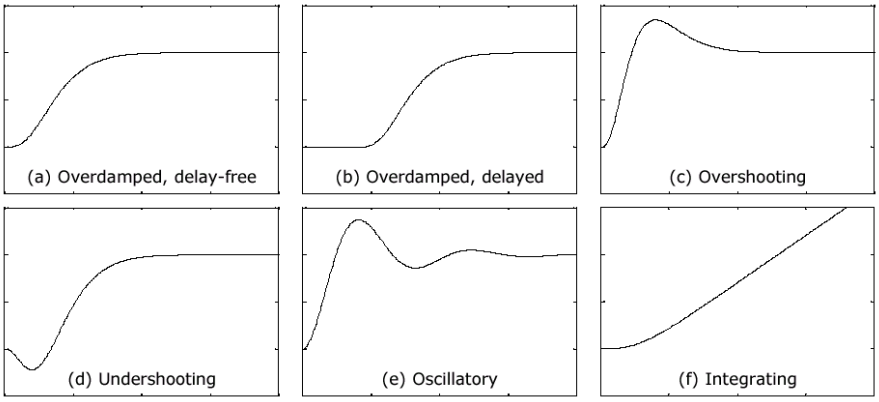


Figure 15: classification of step responses.

First-order models

Overdamped responses can be well represented with a first order model plus delay (or 'dead time', leading to the acronym FOPDT), i.e. with a transfer function in the form

$$M(s) = \mu \frac{e^{-sL}}{1+sT} \tag{15}$$

Many methods exist for identifying such models; an extensive review can be found in chapter 2 of (Åström and Hägglund, 1995). Here we present one of the most widely used, namely the method of areas. Given the step response record $y_s(t)$, one must first compute the gain μ by dividing the response total swing by the input step amplitude A_s and the *unit* step response $y_{us}(t)$ as $y_s(t)/A_s$. Then, denoting by t_{end} the final experiment time, i.e. assuming that from t_{end} on $y_{us}(t)=\mu$, it is necessary to compute in sequence the three quantities

$$A_0 = \int_0^{t_{end}} (\mu - y_{us}(t))dt,\ t_0 = \frac{A_0}{\mu},\ A_1 = \int_0^{t_0} y_{us}(t)dt$$

where the areas A_0 and A_1 motivate the method's name as depicted in figure 16a. Finally, the other two parameters of the model are obtained as

$$T = \frac{eA_1}{\mu}, L = \frac{A_0 - eA_1}{\mu}$$

and setting L=0 should the computed value be negative (which can happen if the real delay is small).

The method of areas is very powerful, remarkably noise-insensitive (since it uses integration) and quite accurate. Moreover, it has the ability of estimating the delay without obliging the user to define thresholds for deciding when the process response has started moving. This ability suggests its use also for (moderately) oscillatory, undershooting or overshooting responses, provided that after computing μ the parts of $y_{us}(t)$ greater than it be 'mirrored' with respect to μ and those below zero be truncated to zero. This is in some sense a trick based more on empirism than on rigorous reasoning. It is shown synthetically in figure 16b, where the process response is indicated by the dashed line and that used for computing A_0 and A_1 by the solid line.

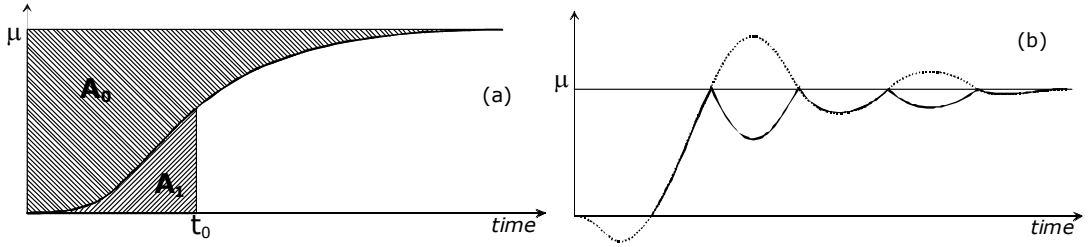


Figure 16: the method of areas (a) and the response to be used (b).

Second-order models

Second order models can describe both overdamped and oscillatory responses. In the former case a delay can also be included for better fitting, leading to a model in the form

$$M(s) = \mu \frac{e^{-sL}}{(1+sT_1)(1+sT_2)} \tag{16}$$

which, with the same rationale as above, is termed SOPDT.

For oscillatory responses, conversely, the advised model structure is

$$M(s) = \frac{\mu}{1 + 2\frac{\xi}{\omega_n}s + \frac{s^2}{\omega_n^2}} \tag{17}$$

It is apparent that an overdamped response can be described also by a FOPDT model. The advantage of the SOPDT is a better phase accuracy, because in the FOPDT case any lag not

explained by the first order rational dynamics causes the estimation of a larger delay (which can deteriorate the subsequent tuning results).

SOPDT models of the type (16) can be identified from the step response in several ways. The simplest one is to determine μ as in the FOPDT case, then L as the intercept on the time axis of the tangent drawn from the unit step response in the maximum slope point. Finally, the remaining parameters T_1 and T_2 are computed by fitting two points of the model unit step response, whose expression (with $T_1>T_2$) is

$$\mu\left(1+\frac{T_2 e^{-\frac{t-L}{T_2}}-T_1 e^{-\frac{t-L}{T_1}}}{T_1-T_2}\right),$$

to the measured response (recall that at this stage μ and L are known). This must be done numerically but is not a complex task. Traditionally, the two points used for the fitting are the ones where the measured response reaches 33% and 67% of its final value.

For oscillatory responses that cannot be described well enough by FOPDT models, i.e. when the oscillation is evident and cannot be confused with a moderate overshoot, models in the form (17) are to be used. To identify them, it is necessary to measure the period T_o of the oscillation and the first two peaks a_1 and a_2 as indicated in figure 17 (μ is to be determined as above). By the way, the presence of a visible second peak is a good clue for suggesting that this is the right model to use.

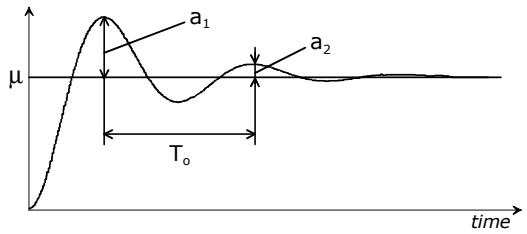


Figure 17: estimating second order oscillatory models.

Once these quantities are available, the remaining parameters of the model are computed as

$$\xi=\frac{1}{\sqrt{1+\left(\frac{2\pi}{\log(a_2/a_1)}\right)^2}},\ \omega_n=\frac{2\pi}{T_o\sqrt{1-\xi^2}}$$

Models for integrating processes

It is very uncommon that integrating processes also exhibit oscillatory behaviours. As such, for describing them it is possible to use FOPDT or overdamped SOPDT models multiplied by 1/s. Moreover, an integrating process produces responses that are similar to those shown for asymptotically stable processes provided a *pulse*, not a step, is applied. More precisely, if

a model M(s) has no poles (nor zeros) in s=0 and $y_s(t)$ is its response to a step of amplitude A, then $y_s(t)$ is also the response of M(s)/s to an *ideal* pulse of *area* A. Hence, one can identify a model for an integrating process in two ways. One is to apply a step and wait that the process output moves for say 5 times the (previously observed) noise band, then remove the step (thus overall applying a non ideal pulse) and wait for settling. The response is then used for identifying a FOPDT or SOPDT model as above, remembering to normalise dividing by the *area* (not the amplitude) of the pulse. The required model is the identified one multiplied by 1/s. The other way is to apply a step and wait for the process response to become a straight line (unfortunately this typically means more perturbation). The response is then differentiated numerically and treated as above for identifying a FOPDT or SOPDT model. The required model is the identified one multiplied by 1/s.

More complex models

If the model must represent complex dynamics, it must be correspondingly complex. For example, some of the responses shown at the beginning of this section apparently call for the presence of zeros. In addition, it is not always a good practice to use a model with delay if it is physically known that no delay exists, thus that the observed dead time is due to high-order rational dynamics. Approximating such processes with delay models permits surely to achieve the regulator tuning, but generally with worse performance than could be obtained.
Unfortunately, for more complex models less general and simple estimation methods exist. It is then often necessary to employ numerical parameter optimisation, and even scratching the surface of this subject would extend far beyond the scope of this volume. However, almost any package for engineering mathematics nowadays offers such features included in a reasonably friendly interface, so that identifying a model given its structure is an ability that can be learned with moderate effort.

3.1.2. Characteristic based process description

Time domain characteristics

Time domain characteristics are, in synthesis, those sketched out in Figure 4: the gain or the presence of an integrator, the settling and rise times, the overshoot and undershoot and so on, naturally completed with the times at which any relevant response fact (e.g. the maximum) occurs. Apparently all these characteristics are almost immediate to obtain from a step test. For a human they are immediate *de facto*, while in an automatic tuning process the only (though not simple) problem is to replicate human insight and to eliminate the effects of noise, which can make the recognition of a characteristic more or less 'blurred'.

Frequency domain characteristics

Frequency domain characteristics, conversely, are very straightforward to obtain by means of a relay experiment. The rationale is that if a process with Nyquist curve $P(j\omega)$ is subject (as in figure 18a) to relay feedback with amplitude D (i.e. whose output is ±D) and hysteresis E (i.e. whose switching points are at ±E), a permanent oscillation of the process output occurs. This oscillation has the frequency ω_{ox} where $P(j\omega)$ intersects the critical point locus of the

relay, which is a straight line parallel to the real negative axis, located in the third quadrant of the complex plane and depending on the hysteresis entity as shown in figure 18b. Noise and disturbances are omitted here for simplicity.

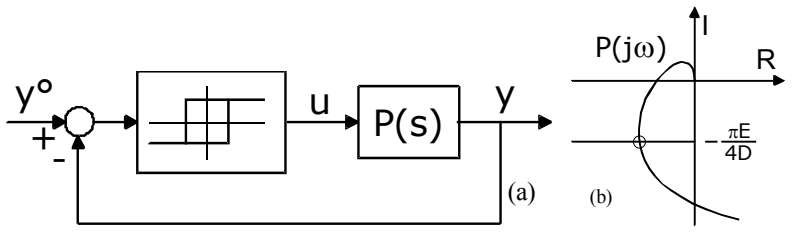


Figure 18: relay feedback (a) and oscillation characteristics (b).

This allows to identify one point of the process Nyquist curve, since $P(j\omega_{ox})$ is the point indicated above with the circle: its magnitude is related to the amplitude A of the controlled variable oscillation by $|G(j\omega_{ox})| = \pi A/4D$ and its phase can be easily deduced knowing that its real part is $-\pi E/4D$. Notice that if a relay without hysteresis is employed this phase is $-\pi$. An extension of the basic relay feedback idea is to insert a time delay between the relay and the process (Leva, 1993). In this case the relay has no hysteresis or, realistically, has the minimum for avoiding spurious switchings due to noise, its critical point locus being still approximated by the real negative axis. Here the point identified is the intersection of the Nyquist curve $\widetilde{P}(j\omega)$ with the real negative axis. Due to the delay τ, however, a point of $\widetilde{P}(j\omega)$ corresponds to the point of $P(j\omega)$ given by $\widetilde{P}(j\omega)/e^{-j\omega\tau}$. Thus, by modifying τ, several points of $P(j\omega)$ can be obtained. This use of relay feedback, then, provides a set of characteristics in the frequency domain given by several points of the process Nyquist curve.

3.2. Most common approaches to PID synthesis

We now present some model based, some characteristics based and (more briefly) some rule based methods for the synthesis of PI(D) regulators. This is not meant to be an exhaustive list, rather a reasoned selection of well established and useful techniques. These methods, and the autotuners that use them, may require the user to select some design parameters. Another aim of this section is then to explain at least the basic rationale of the methods, so as to allow a conscious use of the autotuners based on them.

3.2.1. Model based synthesis

The key feature of model-based methods is precisely that the process description is a model, which is available as a by-product of the synthesis. Thus, these methods rely on some identification technique that must provide a simple, fixed-structure model due to the necessity of deriving simple tuning rules. Albeit approximate, however, this model has to represent the process well enough to allow 'sensible' forecasting of the tuning results. When model-based methods are used manually this is very useful for selecting the possible design parameters. When they are used in an autotuner, it is important to check whether the identification results

are made available to the user: if this is the case, they can be a valuable source of information for process diagnosis.

The Haalman method

The Haalman method (Haalman, 1965) refers to the ideal, 1-d.o.f. PID, i.e. to the control law (2). Its basic idea is to start from a FOPDT or SOPDT model and to select the controller parameters so that $L(s)=2e^{-sL}/3Ls$, which corresponds to a cutoff frequency ω_c of 2/3L and a phase margin φ_m of 50° approximately. The objective is then to make the closed loop behave like L(s)/(1+L(s)). Hence, this is a model following method.

Once L(s) has been assigned, the regulator parameters are computed by applying the relationship R(s)=L(s)/M(s) where M(s) is the process model transfer function, which means that the model poles and zeros are cancelled. If a FOPDT process model is used it is advised to select a PI tuned with the formulae

$$K = \frac{2T}{3\mu L},\ T_i = T$$

while if the process model is SOPDT a PID is selected and the tuning formulae are

$$K = \frac{2(T_1 + T_2)}{3\mu L},\ T_i = T_1 + T_2,\ T_d = \frac{T_1 T_2}{T_1 + T_2}$$

The Haalman method is well suited for processes with overdamped response and significant delay. In fact, being ω_c inversely proportional to L, the requested response might become too fast if L is small.

A modified version of the method has been proposed in (Scattolini and Schiavoni, 1995) so as to ensure, for a FOPDT process model and with a PI regulator, a minimum phase margin φ_m and a maximum cutoff frequency ω_c, taking of course the most restrictive constraint. This leads to

$$K = \min\left(\frac{T(\pi/2 - \varphi_m)}{\mu L}, \frac{T\omega_c}{\mu}\right),\ T_i = T$$

where φ_m and ω_c become design parameters. A cue for selecting them is to fix φ_m to a reasonable minimum (say 50°) by default, while ω_c can be computed by imposing that the (expected) closed-loop settling time, which equals $5\omega_c$, be 'β times smaller' than that of the process model. In the FOPDT case, where the model settling time can be expressed as L+5T, this means setting

$$\omega_c = \frac{5\beta}{L + 5T}$$

where β can range from 4 to 10. Note that it can be interpreted as an acceleration factor, which makes its understanding quite easy also for non specialists. The modified Haalman

method does not necessarily impose L(s), thus it is a characteristics following method where the characteristics providing the desired closed loop behaviour are φ_m and ω_c.

The Symmetric Optimum (SO) method

Despite being older than the Haalman method, the SO one (Kessler, 1958a-b) - which also refers to the ideal, 1-d.o.f. PID (2) - contains several ideas that have been widely developed in the following years. The most important one is to assume that the process model be

$$M(s) = \mu \frac{e^{-sL}}{\prod_{k=1}^{m}(1+sT_k)\prod_{h=1}^{n}(1+sT_h)}$$

i.e. either FOPDT (m=1) or SOPDT (m=2) but with some other poles accounting for unmodelled dynamics. It is also assumed that the time constants T_k are dominant, i.e. that

$$T_k >> \sum_{h=1}^{n}(1+sT_h) \quad \forall k\,.$$

The quantity

$$T_{um} = L + \sum_{h=1}^{n}(1+sT_h)$$

can then be interpreted as the time constant of a transfer function representing the unmodelled dynamics, which is a very clever (albeit rough) way to account for model mismatch. The SO method takes as approximate model

$$M'(s) = \mu \frac{e^{-sL}}{(1+sT_{um})\prod_{k=1}^{m}(1+sT_k)}$$

and designs the regulator so that the cutoff frequency be $1/2T_{um}$ (thus reducing the demand as the mismatch increases) and that the open loop magnitude $|R(j\omega)M'(j\omega)|$ has a slope of -20 dB/dec in the frequency interval from $1/4mT_{um}$ to $1/T_{um}$. Hence, this is a characteristics following method. For a SOPDT model with $T_1 \gg T_2$, the SO tuning formulae are

	K	T_i	T_d
PI	$\frac{T_1}{2\mu T_{um}}$	$4T_{um}$	
PID ($T_2 \geq 4T_{um}$)	$\frac{T_1 T_2}{8\mu T_{um}^2}$	$16T_{um}$	$4T_{um}$
PID ($T_2 \geq 8T_{um}$)	$\frac{T_1(T_2+4T_{um})}{8\mu T_{um}^2}$	T_2+4T_{um}	$\frac{4T_2 T_{um}}{T_2+4T_{um}}$

The SO method performs very well provided that the process delay is small since the time constants T_k must also dominate L, thus it is especially suited for electromechanical systems. Moreover it is keen to generate low frequency regulator zeros, i.e. overshoots in the set point

responses. This can be avoided by filtering the set point or, equivalently, by set point weighting.

Many evolution of the SO method have been proposed (there is an extensive discussion e.g. in Åström and Hägglund, 1995, pp. 166-172).

The Dahlin method or λ-tuning

Given a FOPDT process model, the Dahlin method (Dahlin, 1968) aims at making the transfer function from the set point to the controlled variable resemble that of a first-order model with unity gain, the same delay as the process model and a specified time constant, which becomes a design parameter. Denoting this time constant with λ (which motivates the method's name) this corresponds to tuning the regulator so that it can be approximated by

$$R(s) = \frac{1+sT}{\mu\left(1+s\lambda-e^{-sL}\right)}$$

If the term e^{-sL} is replaced with its (1,0) Padé approximation, i.e. 1-sL, this approximation turns out to be a PI. Conversely, if a (1,1) Padé approximation - i.e. (1-sL/2)/(1+sL/2) - is used, a PID is obtained. In synthesis, then, the Dahlin tuning formulae are

$$K = \frac{T}{\mu\left(L+\lambda\right)},\ T_i = T$$

for the PI and

$$K = \frac{T+L/2}{\mu\left(L+\lambda\right)},\ T_i = T+L/2,\ T_d = \frac{TL/2}{T+L/2}$$

for the PID. The method refers to the ideal, 1-d.o.f. PID (2) and is apparently model following. It is a good technique but requires a reasonable choice of λ. However, experience would soon convince any user that a bigger λ (i.e. less performance) is required when the process-model mismatch is more significant. Therefore, for making these methods really useful, some model error information should be gathered and used. This is dealt with at present by several research paths.

As a final remark, note that λ-tuning has significant relationships with the pPI control law: an interesting discussion on this matter can be found in Åström and Hägglund (1995), pp. 156-158.

The 'kappa-tau' (or 'KT') method

This method computes the parameters of the 2-d.o.f. ISA PID control law (11) apart from N and in the output derivation case (i.e. c=0). It requires to identify a FOPDT model if the process is not integrating, or a FOPDT one plus a factor 1/s if it is.

The information used is then given by the model parameters μ, T and L, by the presence or absence of the term 1/s (the parameters' meaning is of course different) and by the request of a PI or PID regulator. A further specification is the required magnitude margin M_s, defined as

$$M_s = \max_{\omega} \left| \frac{1}{1 + L(j\omega)} \right|$$

for which the two values of 1.4 (conservative tuning) or 2.0 (more aggressive tuning) are advised. Given all this, and defining the process normalised gain α and normalised delay τ as

$$\alpha = \mu \frac{L}{T}, \tau = \frac{L}{L+T}$$

the PI(D) regulator parameters are computed as

$$K = \frac{A_0}{\alpha} e^{(A_1\tau + A_2\tau^2)}, \quad T_i = LB_0 e^{(B_1\tau + B_2\tau^2)},$$
$$T_d = LC_0 e^{(C_1\tau + C_2\tau^2)}, \quad b = D_0 e^{(D_1\tau + D_2\tau^2)},$$

where the coefficients A_i, B_i, C_i and D_i come from the following table, taken from (Åström and Hägglund, 1995).

Integr.	No	No	No	No	Yes	Yes
Contr.	PI	PI	PID	PID	PI	PI
M_s	1.4	2.0	1.4	2.0	1.4	2.0
A_0	0.29	0.78	3.8	8.4	0.41	0.81
A_1	-2.7	-4.1	-8.4	-9.6	-0.23	-1.1
A_2	3.7	5.7	7.3	9.8	0.019	0.76
B_0	8.9	8.9	5.2	3.2	5.7	3.4
B_1	-6.6	-6.6	-2.5	-1.5	1.7	0.28
B_2	3.0	3.0	-1.4	-0.93	-0.69	-0.0089
C_0			0.89	0.86		
C_1			-0.37	-1.9		
C_2			-4.1	-0.44		
D_0	0.81	0.48	0.4	0.22	0.33	0.78
D_1	0.73	0.78	0.18	0.65	2.5	-1.9
D_2	1.9	-0.45	2.8	0.051	-1.9	1.2

These coefficients were derived by applying dominant pole design to many different processes and then interpolating the results to obtain compact tuning relationships. Thus, this is a model following method with the peculiarity of using interpolation. One important remark is that the normalised delay, sometimes called the 'controllability index', can be taken as a measure of how difficult to control a process is.

The KT method is a very good tool, simple to use and suitable for many different situations. Also a frequency response version of this method exists.

The Internal Model Control (IMC) method

The IMC scheme, first proposed in (Morari and Zafiriou, 1989), has found a number of successful applications. To briefly explain its rationale, consider the block diagram of Figure 19

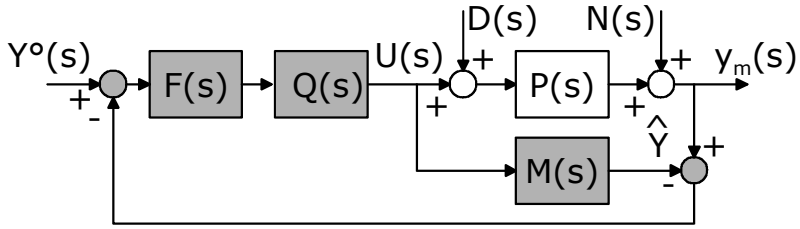


Figure 19: IMC block diagram.

where P(s) is the transfer function of the process (which we assume to be asymptotically stable, thus excluding integrating processes), M(s) is the process model, Q(s) and F(s) are asymptotically stable transfer functions, at this stage arbitrary; y_m and $\hat{y}$ are the true (measured) and nominal controlled variables, i.e. the outputs of the process and of the model respectively. Note that the feedback signal is the difference y_m-$\hat{y}$, which motivates the method's name in that the regulator (the grey blocks) contains a model of the process *explicitly*. Note also that the IMC scheme closely resembles that of a Smith predictor. In fact, the IMC is a generalisation of the Smith and of many other schemes, casting them all together in a unified framework. The IMC scheme corresponds to a classical (1-d.o.f.) feedback one if the regulator is

$$R(s) = \frac{F(s)Q(s)}{1 - F(s)Q(s)M(s)}$$

and it can be proven that, under the hypothesis P(s)=M(s), it is internally asymptotically stable iff M(s), Q(s) and F(s) are asymptotically stable. Hence, the IMC provides a parameterisation of all the regulators which stabilise a control system containing a (known) asymptotically stable process.

Coming to its practical use, the IMC synthesis method is a two-step procedure (all the details omitted here can be found in Morari and Zafiriou, 1989). First Q(s) is determined to optimise the system's response to the reference signal of interest, without any robustness consideration, assuming P(s)=M(s) and with the sole constraint that Q(s) be asymptotically stable. The best policy is to choose Q(s) as an approximated inverse of M(s), namely that of its minimum-phase part. Then, to ensure robustness, the low-pass “IMC filter” F(s) is introduced. The structure and the parameters of F(s) are chosen to achieve a reasonable balance between robust stability and performance. For simplicity, F(s) is often chosen of the first order and (of course) with unity gain, i.e.

$$F(s) = \frac{1}{1+s\lambda}$$

Parameter λ can be interpreted as the closed-loop time constant of the control system if P(s)=M(s) and if P(s) is minimum-phase. More in general, it can be thought as the dominant closed-loop time constant. In any case λ is the design parameter of the method, which determines the control system bandwidth and degree of robustness. Being in the ideal case T(s)=F(s)Q(s)M(s), the IMC is a model following method trying (roughly speaking) to cancel M(s) with Q(s) so as to impose the closed-loop dynamics F(s).

Applying the IMC approach to PID tuning is very straightforward. Here we present a modified version (Leva and Colombo, 2001b) of the original IMC-PID method reported in (Morari and Zafiriou, 1989). This method computes the parameters of the 2-d.o.f. ISA PID (11) apart from the weights b and c but including N, which can improve the sensitivity (see Leva and Colombo, 2001b for a discussion). If the objective is disturbance and noise rejection, then, this method does all the job. If also set point tracking is an issue, several methods exist for computing the weights (which do not interact with stability and disturbance rejection, as discussed). One such method is proposed in (Leva and Colombo, 1999), but also computing them with the (simpler) KT formula has proven to be satisfactory in practice.

In synthesis, then, the method consists of identifying a FOPDT model and then applying the IMC technique by choosing

$$Q(s) = \frac{1+sT}{\mu}, \; F(s) = \frac{1}{1+s\lambda}$$

and by replacing the process delay by its (1,1) Padé approximation (1-sL/2)/(1+sL/2). The regulator turns out to be a real PID given by

$$T_i = T + \frac{L^2}{2(L+\lambda)}, \; K = \frac{T_i}{\mu(L+\lambda)}, \; N = \frac{T(L+\lambda)}{\lambda T_i} - 1, \; T_d = \frac{\lambda L N}{2(L+\lambda)} \qquad (18)$$

The main concern in using the IMC-PID method is the choice of λ. This concern is shared by the Dahlin method, which can be interpreted as an ancestor of the IMC procedure. As anticipated, λ is a knob for trading stability and robustness against performance. It has been proven (Leva and Colombo, 2001b) that, given a process model and an estimate of the corresponding model error, a lower bound for λ, i.e. an upper bound for the performance request, can be found beyond which stability cannot be ensured anymore.

There exist also methods for estimating the model error from measured data (see e.g. Leva and Colombo, 2000) but these would lead us beyond the scope of the volume, thus we just quote the fact as a suggestion for interested readers. As a practical rule of thumb, anyway, one can reason (both for the IMC-PID and for the Dahlin method) as described in the following. A far more extensive discussion, involving cues for selecting the most appropriate controller structure, is reported in (Åström *et al.*, 1992).

- Decide whether rational dynamics dominate the delay or vice versa. This can be made e.g. by computing the normalised delay and assuming the delay as dominant if it is greater than a given threshold (one is a good threshold, see Åström *et al.*, 1992).
- If the delay is not dominant, say if $\tau \leq 0.25$, set λ to a fraction of the model time constant (e.g. 1/10 to 1/2 depending on the required acceleration).
- If the delay is moderately dominant, say if $0.25 < \tau \leq 0.75$, set $\lambda = 1.5(L+T)$.
- If the delay is definitely dominant, consider delay compensation (e.g. with a Smith Predictor scheme or with a pPI) if possible, otherwise choose a greater λ than in the previous cases: $3(L+T)$ is a fairly good first guess.

It must be noticed that these are only rules of thumb. In every practical case, performance can be improved by trying different values of λ in the field. For apparent stability reasons, this should be made starting with a high (i.e. conservative) value and then reducing it.

Optimisation methods

There are a number of methods sharing the following, simple rationale. If the closed loop behaviour must be made similar to that of a given model, tune the PID by minimising with respect to its parameters a cost function J containing the difference between the response of the loop (forecast with the process model) and that of the model to be followed. A frequently used function is the ISE, i.e.

$$J = \int_0^{t_{end}} \left(y_{(forecast)}(t) - y_{(mod.\,to\,be\,followed)}(t) \right)^2 dt$$

This reasoning can be extended to the characteristics following context provided that the cost function does not involve any model to be followed: for example, it may simple contain the error and be

$$J = \int_0^{t_{end}} \left(y^{\circ}(t) - y_m(t) \right)^2 dt$$

which can be viewed as a characteristic of the loop behaviour. It is important to note that these methods require the user to specify what response must be considered. For example, minimising an ISE for the set point response may lead to different results than minimising it for the load disturbance response. Also, the ISE is not the only cost function employed.

Several examples of such methods can be found in (Åström *et al.*, 1993; Åström and Hägglund, 1995) and in the references given therein.

3.2.2. Characteristics based synthesis

The key feature of non model based methods is that the process description is not a model, rather some characteristic values of it in the time or frequency domain. In deriving some

methods a model may be involved, but this is *not* meant to represent the process so as to allow forecasting the tuning results.

Here too, we present a brief and not exhaustive list of methods, to improve the knowledge of them and of autotuners based on them. For apparent reasons, non model based synthesis methods are intrinsically non model following.

The Ziegler-Nichols methods

The Ziegler-Nichols rules (Ziegler and Nichols, 1942) are the first (thus historical) example of a method for automating the PID synthesis. Their rationale is to impose a decay ratio of 0.25 to the set point step response, assuming that the process can be (very roughly) described by an integrator plus a time delay at least in the band of interest.

As such, this is a characteristics following method. Ziegler and Nichols proposed some formulae for computing the PID parameters on the basis of characteristic values of either the process open-loop step response or the sustained oscillation induced by a proportional regulator of convenient gain.

The first method consists in determining the quantities a and b based on an open-loop step response record as depicted in figure 20.

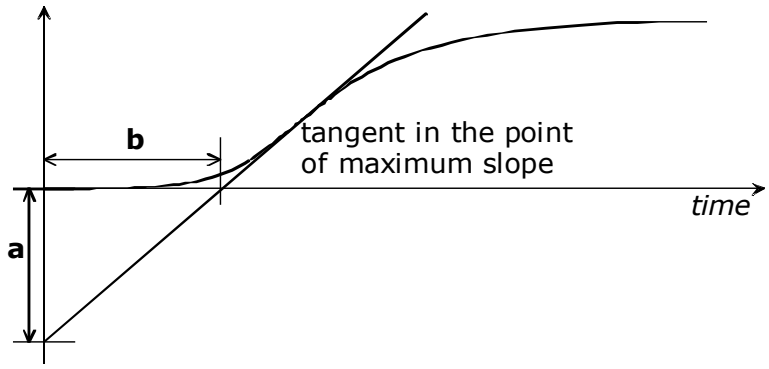


Figure 20: the Ziegler-Nichols method.

Then, the PI(D) parameters can be computed from the following table:

	K	Ti	Td
P	1/a		
PI	0.9/a	3b	
PID	1.2/a	2b	b/2

The second method requires to apply proportional control and increase the controller gain until the process output reaches a sustained oscillation. Denoting with T_u the period of the oscillation and with K_u the regulator gain yielding it, the PI(D) parameter are computed as

	K	Ti	Td
P	$0.5K_u$		
PI	$0.4K_u$	$0.8T_u$	
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$

Relay-based methods

A number of methods exist for PID tuning that are termed this way because they use the typical information provided by a relay experiment, i.e. one or more point of the process Nyquist curve. As such they are all characteristics based methods, and in the great majority of cases they are also characteristics following. A somehow historical example can be found in (Åström and Hägglund, 1984), another is (Leva, 1993). In fact, the most natural way of using one point of the process Nyquist curve characterised by a frequency ω_1, a magnitude P_1 and a phase φ_1, is to impose that

$$R(j\omega_1)P_1e^{j\varphi_1} = e^{j(\varphi_m-\pi)}$$

This complex equation yields two (real) regulator parameters and means that the open loop Nyquist curve $L(j\omega)$ will contain the point $e^{-j(\varphi m-\pi)}$, i.e. that the loop will have cutoff frequency ω_1 and phase margin φ_m. For tuning an ideal PID in the 1-d.o.f. form a third equation is required, which is typically obtained by imposing the T_i/T_d ratio, i.e. setting $T_i = \alpha T_d$ where α becomes a design parameter useful for limiting the high frequency regulator gain.

The main concern in using relay based methods is that the cutoff frequency emerges as a result of the relay experiment, being the frequency at which the oscillation arises. As such, the amount of relay hysteresis adopted becomes relevant in determining the loop cutoff frequency, and the problem is that the relationship between these two quantities is far from trivial. Inserting a delay in series with the relay can help in this respect, because modifying this delay along the experiment can make the oscillation arise at a prescribed frequency. This means that the design parameters are the cutoff frequency and the phase margin, which is a widely accepted choice. The problem is that several delay modifications may be required, resulting in a long tuning phase.

A very important advantage of relay based methods, however, is that the local characteristics of the open loop Nyquist curve can be imposed exactly in that it is guaranteed that this curve contain the point $e^{-j(\varphi m-\pi)}$: the problem is what the overall behaviour of this curve will be, but having a way for imposing at least its local behaviour around the cutoff with great precision is a good feature.

3.2.3. Rule based synthesis

According to the proposed classification, in rule based synthesis there is actually no explicit 'process description', neither as a model nor as a set of characteristics. The goal behind this class of methods is to mimic human intuitive reasoning rather than time or frequency domain computations. Under this framework lie expert and fuzzy systems.

These methods, which are not suitable for manual tuning, but can only be applied in an automated manner will be described further on, in the section devoted to Soft-Computing methods.

3.3. Choosing the controller structure

This section is aimed at giving some very basic guidelines for understanding when P, PI, PID or more complex control strategies are recommended. Borrowing from section 3.9 of (Åström and Hägglund, 1995), where a deeper analysis is reported, we can start with the following statements.

- PI control is sufficient when 'tight control' is not required (i.e. when the *detailed aspect* of the controlled variable and control signal transients is not an issue) or when the process dynamics exhibits an apparent first-order behaviour (single time constant and no dead time). This is easy to guess from a step response and quite frequent in practice (e.g. in level control of single tanks). Moreover, the I action can be excluded if zero steady-state error is not required, which sometimes happens e.g. in internal loops of cascade controls.
- PID control is sufficient when the process dynamics looks second-order and is significantly delay-free. A typical case is temperature control, when one time constant comes from the body whose temperature is controlled and the other from the sensor. The D action is particularly beneficial if the two time constants differ significantly, as is common in the case quoted. It must be kept in mind that a large D action also amplifies measurement noise unless properly filtered. Thus, if the regulator at hand does not allow control on the derivative filter, it may be better to reduce bandwidth expectations and use a PI.
- For processes where one of the two previous statements hold and if tight control is not required, there is a very little benefit in using more complex controllers.
- PID control may be inadequate when tight enough control is required for processes with long dead times, high-order dynamics or oscillatory modes. If this is the case it is often necessary to resort to more complex and/or specialised controllers, which are not treated in this volume.

An excellent and far more detailed discussion on the matter of this section can also be found in Åström *et al*., 1992 and in the papers quoted therein, to which the interested and control-theoretically curious reader is referred. The discussion starts by defining some dimensionless quantities that can be used for indicating a preferred controller structure and also for predicting the main characteristics of the closed-loop behaviour.

Though referring to regulators tuned with the closed-loop Ziegler/Nichols rules, this discussion can be used for drawing first-cut conclusions on the preferred controller structure also in the general case provided that the process dynamics can be described by low order models precisely enough. It must be kept in mind that *in any case* these are just first-cut rules: it is in

no sense guaranteed that the suggested controller structure is the 'best' one, but if an autotuner allows to choose the structure, the one provided by these rules is a good first guess. In the following we shall then resume the guidelines given in Åström *et al*., 1992, which at the authors' experience can be used effectively in all the cases of practical interest. The first thing to note in this respect is that user-guided controller structure selection is seldom available in autotuners but can be very useful if properly employed. Furthermore, any recommendation on the controller structure is deeply connected with some on the tuning policy. This means, quite intuitively, that when a problem is so critical that the controller structure cannot be chosen arbitrarily this also indicates that some ways of tuning that controller are not recommended. The quantities considered in Åström *et al*., 1992 refer to a FOPDT model in the form (15) if the process is not integrating or to one in the form

$$M(s) = \mu_v \frac{e^{-sL}}{s(1+sT_v)} \tag{19}$$

if it is integrating, thus they are consistent with the (simple) model identification guidelines we have given before. These quantities are the 'normalised process gain for processes without integration' k_1, the 'normalised process gain for processes with integration' k_2, the 'normalised dead time for processes without integration' θ_1 and the 'normalised dead time for processes with integration' θ_2, defined with reference to (15) or (19) as

$$k_1 = \frac{\mu}{|M(j\omega_u)|}, \quad k_2 = \frac{\mu_v}{\omega_u |M(j\omega_u)|}, \quad \theta_1 = \frac{L}{T}, \quad \theta_2 = \frac{L}{T_v},$$

where ω_u is the 'ultimate frequency', i.e. the smallest one such that the model phase is $-\pi$. Note that names like these are used elsewhere with different meanings, so we have adopted the quantity names and symbols used in Åström *et al*., 1992 to avoid confusion.

The directions for controller structure selection are given in the following table. The complete interpretation of this table would be too lengthy to carry out here. For the purpose of this work, it is better to draw from it some operational cues for the selection and use of an autotuner. Hence, we can state the following.

- When the table dictates that the I action is optional, recall that it must be included if zero steady-state error is required.
- If tight control is not required the only significant recommendation is to use set point weighting with integrating processes that, if the integrator is removed, show a dynamics dominated by the time constant T_v (case 5). Integrating processes whose dynamics (removing the integrator) is dominated by the dead time can be managed with a P or PI regulator (case 4).

If tight control is required, things are far more complex. Also in this case, however, some directions can be given.

- When feedforward compensation is required recall that set point weighting is a (very specialised) form of it, see (13). Hence, in the single-loop PID framework treated in this volume, choose a PID with this feature (e.g. an ISA one) and possibly an autotuner employing the weights. If the latter is not available tune for load disturbance rejection (where feedforward has no influence) and then try to adapt the weights manually, recalling again (13).
- When dead time compensation is required it would be advisable to use regulators with this feature (e.g. comprising a Smith predictor or a pPI). There also exist autotuners for these regulators, not treated in this volume for space limitations. If no predictor is available, choose a tuning policy aimed basically at high damping and then try some manual adjustments, e.g. slightly increasing the gain or decreasing the integral time.

		Tight control not required	Tight control required		
			High measurement noise	Low saturation limit	Low meas. noise and high sat. limit
1	$\theta_1>1$, $k_1<1.5$	I	I + FFCE + DTCR	PI + FFCE + DTCR	PI + FFCE + DTCE
2	$0.6<\theta_1<1$, $1.5<k_1<2.25$	I or PI	I + FFCR	PI + FFCR	PI + FFCR + DTCR or PID + FFCR + DTCR
3	$0.15<\theta_1<0.6$, $2.25<k_1<15$	PI	PI	PI or PID	PID
4	$\theta_1<0.15$, $k_1>15$ or $\theta_2>0.3$, $k_2<2$	P or PI	PI	PI or PID	PI or PID
5	$\theta_2<0.3$, $k_2>2$	PD + SPWE	PPTR	PD + SPWE	PD + SPWE

FFCR Feed Forward Compensation Recommended
FFCE Feed Forward Compensation Essential
DTCR Dead Time Compensation Recommended
DTCE Dead Time Compensation Essential
SPWE Set Point Weighting Essential
PPTR Pole Placement Tuning Required

4. The typical autotuning process

Automating the steps of tuning methods

In this section we shall describe the most important aspects of the autotuning process. That is, after the reader has been provided with the background of how a PID can be tuned on the basis of process information gathered from the field, it will be explained how this operation can be automated, what are the major problems that arise in doing this and what are the solutions taken for these problems in industrial autotuners. Understanding this material will then become another source of information for selecting an autotuner and for using it in the most effective way.

4.1. Obtaining the process behaviour description automatically

4.1.1. The needs: steady-state and control-relevant dynamics determination

First, it may be necessary to determine the static process behaviour. Limiting the scope to situations that may arise in practice, in a model based context this means sensing whether the process is integrating or not and in the latter case estimating its gain. In a characteristics based context this may mean several things: the gain or the steady state output value are normally considered as characteristics too, and in most cases some lexical variable is used for stating that the process is integrating. So, as long as static behaviour is considered, the model and the characteristics based contexts are quite similar.

However, the matter is less trivial than it may appear for at least two reasons. Under the point of view of the autotuner designer, obtaining static information is not an easy task especially in noisy cases and when nonlinearities are involved. Under the point of view of the user, static information is always useful but its actual importance depends on the characteristics of the specific control problem.

This fact, which is the more relevant in this context, will be now illustrated with an example. Consider the two processes

$$P_1(s) = \frac{1}{(1+s)^2}, P_2(s) = \frac{0.5}{s}$$

Notice that at ω=1 both have magnitude 0.5 and phase –90°: thus, we can say that these two processes, though they are completely different as for the static behaviour because one is integrating and one not, in the frequency domain 'have the same local behaviour' around ω=1.

Suppose the regulator must be tuned to achieve ω_c=1 and φ_m=60°. This can be done with a PI by imposing that

$$R(j1)P(j1) = e^{-j120°}$$

which, by the way, is the most common practice in relay tuning. In both cases, since $P_1(j1)=P_2(j1)$, the result will be $K = T_i = \sqrt{3}$. Figure 21 depicts the resulting open loop Nyquist plots, the magnitude plot of the transfer function from the load disturbance to the controlled variable (the process output) and the closed loop responses of the controlled variable to a set point and to a load disturbance unit step.

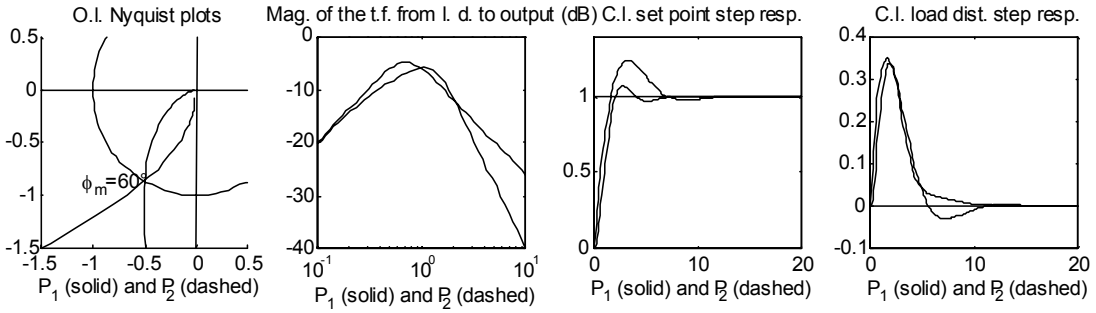


Figure 21: different processes with the same local behaviour and local requests on L(jω).

It is apparent that in both cases the required ω_c and φ_m have been attained, that the resulting set point responses are quite different and that the load disturbance responses are not so different. All these facts are because a *local* request (assigning one point of the open loop Nyquist diagram) has been made, and the effects of it depend on the *overall* loop dynamics, i.e. on the overall aspect of P(jω). In particular, the integrator makes the two Nyquist plots outside the unit circle (i.e. at low frequencies) completely different, while this difference is significantly 'smoothed' (the magnitude maxima are comparable and at similar frequencies) in the transfer function from load disturbance to output unless at high frequency, where it is not relevant in any case.

Observing the figure, we can now make some remarks. Should the tuning process have been made taking into account the presence or absence of the integrator, it would have been possible e.g. to increase the requested phase margin so as to obtain a lower overshoot. This, however, would also have slowed the load disturbance response. Or, the overshoot in the set point response could have been cured with set point weighting, but this requires a 2-d.o.f. regulator. And in any case, one point of P(jω) – which is the only information used in the example for tuning – is not enough if not only ω_c and φ_m but also the aspect of the obtained transients are important, i.e. if tight control is required. In this case, either static information must be gathered prior to tuning or it must be (implicitly) obtained after, measuring the resulting overshoot to compute the set point weight. From this example, we can learn the following (and general) lessons.

- Static process information is always useful, though in general not easy to obtain and leads to longer identification phases (e.g. it cannot come from a single relay experiment).
- Static information can be more or less relevant depending on the regulator structure, on the tuning policy and on the characteristics of the control problem at hand, especially on whether tight control is required or not. This means that a skilled user needing to select an autotuner should ask himself at least the following questions: do I have mostly a set point tracking or a disturbance rejection problem? Do I need tight control or not? Do I have integrating processes to deal with or not? Is the regulator 1-d.o.f. or 2-d.o.f.? Does the autotuner use static information or not? This means also that the autotuner should be documented well enough to allow these questions to be answered.

It is very unlikely, however, that these autotuner characteristics do emerge from the product documentation clearly enough. This is a pity since it adversely affects good product selection and utilisation, which ends up in diminishing the users' confidence in the autotuning technology as a whole. However it can be counteracted by users at least in two ways, which are in some sense two further lessons learned:

- Encouraging autotuner manufacturers to enrich their documentation so that it will be informative for knowledgeable users, by convincing them that this can make a product more successful.
- Learning to ask themselves the questions above, to seek their answers in the documentation and, if they are not there, to find them by experimenting with the autotuner. In fact, in the authors' experience, once the user has clarified to himself what he wants to know on a given autotuner, simple experiments with it in a laboratory are enough. As a further remark, experiments in the field are *not* always the right way to answer these questions, nor to evidence general (thus conceptually reusable) facts.

The second need is to gather information on the control-relevant process dynamics. This task is even more difficult because, contrary to steady state information, it is necessary that some clue on what are the 'control-relevant' dynamics be provided by the user or by the identification. In fact, the concepts of 'dominant' and 'control-relevant' dynamics are often confused by many users, and this too leads to poor autotuner utilisation. Also in this case, we illustrate the idea with an example. Consider the process

$$P(s) = \frac{1+0.5s}{(1+s)^3}$$

Identifying a FOPDT model for it with the method of areas leads to the result depicted on the left in Figure 22; on the right are the results of different PID tuning operations.

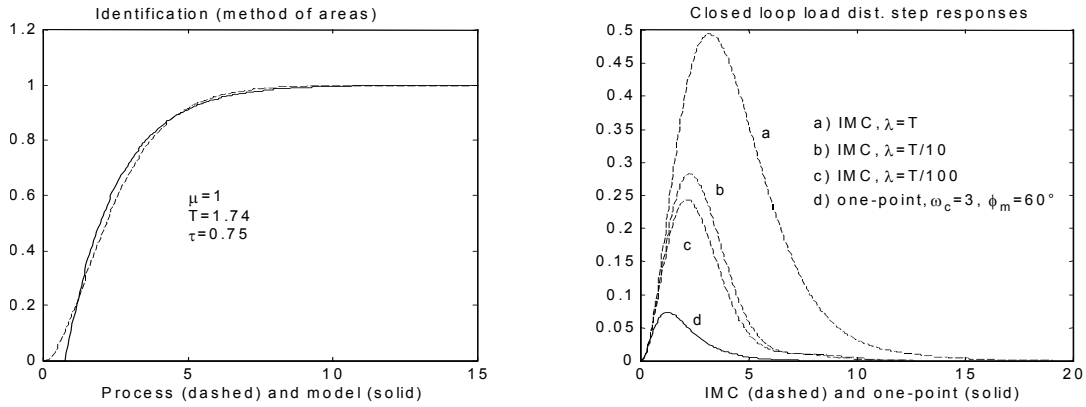


Figure 22: different tuning operations on the same process.

Setting T=1.74 captures the dominant dynamics well enough. However this requires the introduction of a delay (which is not physical). Moreover, the FOPDT structure 'hides' the effect of the zero. As a result, the cancellation policy of the IMC method will always produce

an integral time larger then needed, so that only the gain will be used for opening the bandwidth to achieve good disturbance rejection. The gain is however limited by the delay, so that even very demanding speed specifications (e.g. λ=T/100) will not push the obtained result beyond a certain limit. Notice also the 'saturation' in the achieved performance: the improvement obtained between λ=T/10 and λ=T/100 is smaller than that from λ=T to λ=T/10. It is also apparent that in this case tuning with a 'one-point' method, i.e. using e.g. a relay based autotuner, can achieve far better results provided one stipulates a good specification in terms of ω_c and φ_m. Taking some general lessons also from this example, then, leads to the following statement:

- Not always do the dominant and control-relevant dynamics coincide. The latter depend also on what are the control needs. The problem is particularly relevant for processes with complex dynamics, where low order models may be inadequate. In such cases, if tight control is required, it is better not to use model based autotuners unless they have the capability of selecting the model structure or the set of characteristics to detect and/or measure, or at least to let the user select these. Note also that model or characteristic structure selection is a very powerful feature but requires a significant degree of understanding.
- Advanced autotuners with model or characteristic structure selection typically 'hide' this capability by asking the user what is the nature of the controlled variable (e.g. temperature, level, flow) and sometimes some other basic information on the process, e.g. whether the pressure of a liquid or of a gas is to be controlled. On the basis of the user response they can then make some - generally loose - assumptions on the structure of the process dynamics (in the model based case) or on which characteristics have to be expected and quantified (in the characteristics based case). For example, in level control the process is frequently integrating, in pressure control there is frequently a quick overshoot followed by smooth convergence to the steady state, and so on. Apparently, it is necessary to provide this information correctly. It would be also useful to know *which* assumptions the autotuner makes, but this is seldom revealed – mostly due to justified know how protection – and is also quite difficult to figure out by experimenting with the autotuner. The advice, when selecting one, is then to try providing some of the requested information incorrectly. This should not be done in the field, of course, but is very useful when testing the autotuner, e.g. simulating the transfer function seen by the regulator with a PC. In so doing, one can always obtain a quite clear idea of which information is more critical, and sometimes (with good knowledge of the control theory and some luck) also imagine at least the basic assumptions made by the autotuner.
- If one has a clear idea on what some closed loop characteristics must be, it is better to use a characteristics following autotuner having those among the ones it is capable of ensuring.
- If one has no well defined desires, a good idea is to use a model based autotuner (which may sometimes achieve modest results but is very unlikely to provoke e.g. instability) and observe its outcome. In our example it would have been evident that the IMC produced too large an integral time and that the disturbance transient could be accelerated.

After the autotuning operation, an experienced user could easily refine the tuning (an easier job than doing it from scratch).

Three remarks are now in order. First, the considerations we have made apply both to model based and to characteristics based autotuning. No matter what the description of the process behaviour is, certain quantities yield information on certain facts, and the knowledgeable user must be able of understanding if these facts are the right ones to learn about for solving his problem. In the great majority of cases where the authors have seen autotuners not behaving satisfactorily, some wrong assumptions in this respect had been made when selecting the autotuners themselves.

Second, when thinking of how an autotuner obtains the description of the process behaviour, users tend to focus their attention primarily on "what it does to the process" and to worry mostly about the required perturbations. This is important but is not the only aspect: it is equally important to figure out "what it learns about the process" and to discriminate whether the information learned are enough and adequate for solving the control problem. In other words, first one has to worry about what the autotuner must know, then select one that learns what is required with the minimum process upset: doing the reverse (i.e. limiting the accepted upset *a priori*) is keen to limit the achievable results, and in the authors' knowledge it is another very frequent flaw in control systems design involving the use of autotuners.

Third, one of the reasons why a skilled human can outperform any autotuner is that gathering process information and converting it into a description of the process behaviour automatically is a very difficult task. Any human, looking at any response, can learn much more than any automatic system. This means that, for evaluating an autotuner, it is very important to see how rich, flexible and possibly configurable the process description employed is.

Of course, all these considerations apply to complex problems and where tight control is required. There is no need to say that these cases are a minority, and that in all the others almost any autotuner can do a decent job. In other words, the examples presented herein should not make the reader think that autotuners need an amount of human effort comparable to that required for tuning the regulators manually. We are discussing these problems because the aim of this section is to identify difficult cases, explain why they are difficult (i.e. when and why the way an autotuner is selected and used may be critical) and giving clues for solving them. In this respect we can conclude that no process description is 'always' good, so that if an autotuner does not work properly also the identification phase, and not only the specs or the tuning method, must be diagnosed.

4.1.2. Improving the accuracy in obtaining the process description

On-line outlier removal

During the identification phase, spurious phenomena may affect the measurements. This must be counteracted somehow, otherwise *any* process description may be erratic. To give a scheme, we can distinguish two main cases: fast phenomena spoiling only a few samples and

slower phenomena acting for a time comparable with the process dynamics. The former case is the typical effect of noise or transmission flaws, the second may also occur when 'some other manoeuvre' is made on the process during a tuning operation. Of course the distinction between 'fast' and 'slow' is relative, but in the majority of practical case it is quite easy to distinguish the possible sources of outliers and to classify them in these two categories. We shall refer to them as 'instantaneous' and 'lasting' outliers. Rigorously speaking there is a third type of outlier, given by phenomena with a time scale *longer* than that of the process dynamics. However this problem is traditionally treated in the framework of 'detrending', so here it is dealt with in the corresponding section.

The effect of instantaneous outliers is relevant especially for model based autotuners and for the determination of process characteristics in the frequency domain. In fact, it is intuitive that a single spurious sample is more unlike to cause an erroneous detection e.g. of a settling time than to affect the result of the FFT of an output record. In addition, instantaneous outliers are quite easy to eliminate: most autotuners impose a threshold on the measurement variation and reject samples that exceed it. This threshold is normally computed on a statistical basis, and in some cases this mechanism is configurable. Also filtering (see later) may help.

Lasting outliers are more difficult to eliminate and can affect any type of process description. The first precaution to take is to ensure that during the identification phase nothing happens that affects the process output other than the possible input stimulation made by the autotuner. In the opposite case, any identification method would try to explain these effects as input-output process dynamics, thus making an unavoidable mistake of unpredictable entity. Of course this is a preventive approach, but little else can be made. It is however important that when using the autotuner these situations be detected (usually by the user, who must then be conveniently trained) so that the autotuning process can be aborted simply and without consequences, which is another feature to look for when selecting an autotuner. It is important to note that user training in this respect is simple but necessary: the authors have seen a number of failed autotuning operations in which the only problem was that plant operators did not know that at least the loop they were tuning had to be 'left alone' during the tuning operation. This is very simple to explain and understand and solves the great majority of that kind of problem. It is also obvious for anyone with a minimum knowledge on the matter, but curiously is not always transmitted to operators.

Filtering

Filtering is used mainly to eliminate the effects of noise, including instantaneous outliers. The filters used are normally lowpass, and it is desirable that their bandwidth be configurable. Some systems determine it automatically by recording the process output while the control is kept constant. If the filter is configurable, it must be considered that any identification procedure will identify the process *in series with the filter*, so that an incorrect choice of it may deteriorate the identification results.

On the basis of considerations similar to those made for outliers, it is intuitive that the potential negative effects of filtering are more noticeable in model based autotuners, and especially in those that employ optimisation or prediction based techniques.

Detrending

A process response is said to have a superimposed 'trend' if it is affected by very slow phenomena, lasting more than the dynamics that are relevant to the response itself. It is important to note that these phenomena may be completely exogenous or depend (at least partially) on the possible stimulation given to the process. In other words, the qualifying aspect of a trend is only the fact of being slow and typically of modest entity, not the fact of being exogenous.

To explain this with an example, we may consider the response of the temperature of a body to a step in the power of its heater. If this response is measured while the temperature of the surrounding air is varying very slowly, it will have a trend. However, the air temperature variation could be mostly exogenous or be also significantly due to the thermal exchange with the body itself and/or to heater losses. In a very simplified case, this means assuming either that the air temperature increases linearly with time or that the temperature of 'far-off' air stays constant while that of 'surrounding' air increases due to the exchanges with the body and with far-off air. These two situations correspond to the models

$$\begin{cases} C_{body}\dfrac{dT_{body}}{dt} = P_{heater} - K_{bodyair}\left(T_{body} - T_{air}\right) \\ T_{air} = T_{a0} + T_{agrad}t \end{cases}$$

and

$$\begin{cases} C_{body}\dfrac{dT_{body}}{dt} = P_{heater} - K_{bodyair}\left(T_{body} - T_{surrair}\right) \\ C_{air}\dfrac{dT_{surrair}}{dt} = K_{bodyair}\left(T_{body} - T_{surrair}\right) - K_{surrfar}\left(T_{surrair} - T_{farair}\right) \\ T_{farair} = T_{a0} \end{cases}$$

Implementing and simulating these two models (where the meaning of symbols should be self-explanatory) in the Simulink environment produces the results depicted in Figure 23, where the first model and its transients are reproduced in the upper half while the lower half is devoted to the second model and to the transients obtained with it.

It can be easily observed that, when the air thermal capacity is bigger than that of the body, even if the thermal exchange coefficient from body to surrounding air is larger than that from surrounding air to far-off air (a quite realistic situation), the surrounding air temperature response is slow. So slow in fact that in the time scale of the body temperature response it is

practically a straight line. As a result, without quite detailed process knowledge it is apparently impossible to distinguish one case from the other.

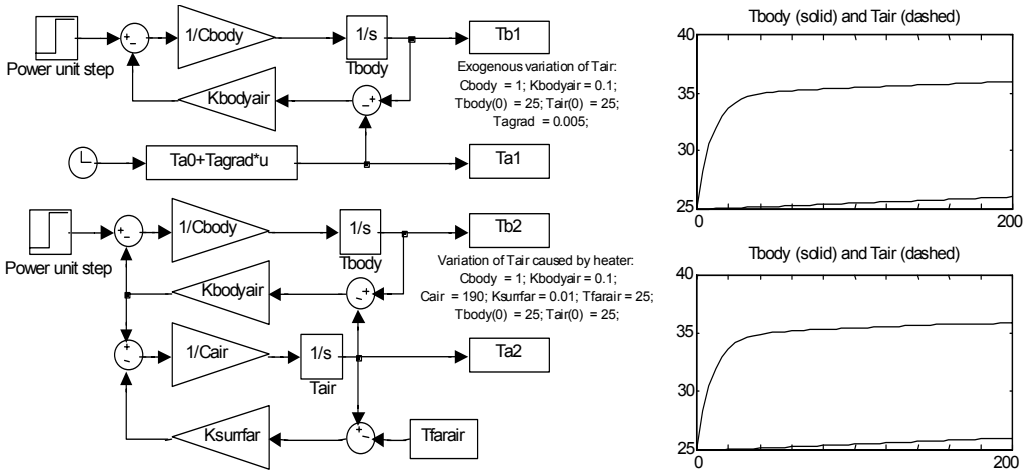


Figure 23: trends caused by exogenous and endogenous phenomena.

In the autotuning context detailed process knowledge is seldom available, and strictly speaking assuming its availability is in contradiction with the principle itself of autotuning. So the problem is not to understand the reasons of a trend or to distinguish it from the effects of unmodelled dynamics (corresponding in the example to the air thermal dynamics if the first model is used), rather to recognise the presence of a trend and to eliminate its effects on the identification of the process behaviour description.

Coming back to the example, it should be clear that when controlling the body temperature the relevant dynamics are those of the body. These do emerge from the step response but, considering the time scale of the body temperature response, it seems that no steady state is reached. So, trends may be viewed as another reason why detecting static characteristics may be tricky.

The effects of absent or incorrect detrending can vary a lot depending on all parts of the autotuner's operation. It is important to remember that in any case they must not be considered a marginal aspect. Suppose, for example, that in the simple case described a PI had to be tuned from a first order model, that the time constant T of this model be computed as 1/5 of the measured settling time, that its gain μ be obtained from the steady state value of the step response and that tuning be made by pole/zero cancellation so that the closed loop settling time is equal to half the measured open loop one. This implies that $T_i=T$ and $K=2/\mu$. Note that this is a quite crude procedure, but especially for very low end products it is not so far from reality.

In the example, which is deliberately extreme, the response of the body temperature (in the second more realistic model) settles when that of the air temperature does, in approximately $1.2 \cdot 10^5$ seconds. The amplitude of the corresponding transient of the body temperature is ap-

proximately 110 degrees. This means μ=110 and T=24000, while the detrended response of the body temperature provides approximately μ =12 and T=10. Denoting with (a) the PI controller tuned with the detrended response and with (b) the other, the closed loop responses to a unit set point step are shown in Figure 24: notice the different time scales. Apparently the PI (b) 'regulates the wrong dynamics', but if nobody has told it what are the relevant ones it can only do what it has been tuned for. Realistic cases are not so extreme, but it should be clear that a 'dumb' tuning policy can easily fail and that the absence of detrending is a fairly evident symptom of dumbness.

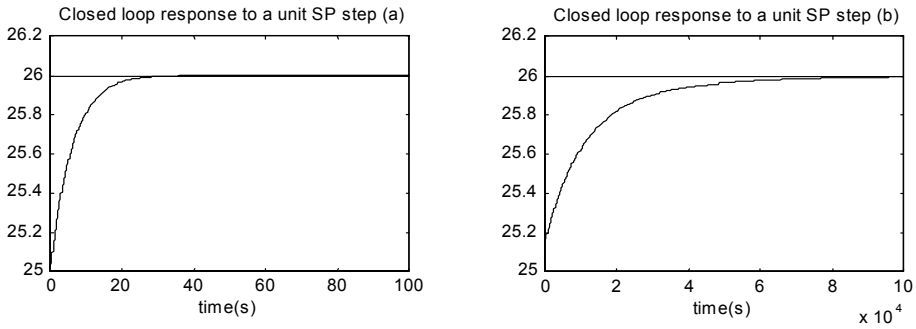


Figure 24: closed loop responses with and without detrending.

To generalise from the example, we can easily state that the solution to this problem is to instruct a possible autotuner as follows: I know that a steady state *must* be reached by the step response, so when you record it and see it goes asymptotically to a straight line, consider *this* as a steady state because there must be something else (I don't care what) that acts like 'moving some base value' for the response (note that, observing the phenomenon from the body, that is exactly what happens with the air temperature in both cases). So, identify the slope of this trend and use it to subtract a straight line from the response you have: that is the response to consider for describing the process, because that is the one containing the relevant phenomena.

For doing this it suffices to say 'the process is not integrating', which is an information that most advanced autotuners accept and that it is important that the user be able of providing consciously. Note that the same could be said if a (slowly) oscillating trend were observed, due, say, to cyclic climatic variations throughout the experiment. In this case the user should instruct the autotuner that 'the process has no oscillatory dynamics' and a periodic trend would have to be detected and subtracted.

To help non specialists, advanced autotuners normally do not ask whether the process is integrating, oscillating or not. Instead, they adopt the same policy described for model or characteristics structure selection, requesting the user to provide some basic information on the nature of the loop. This information can also be used for deciding which aspects of the response used for the identification are probably spurious, and correspond to trends that must be removed. We can then notice again that for judging an advanced autotuner it is important

to understand (as completely as possible) what assumptions on the process dynamics, thus on the expected responses, it makes depending on the information provided by the user.

4.2. Accepting (and checking) user specifications

Contrary to (inexperienced) intuition, this is one of the most important parts of an autotuner. Specifications concern stability, robustness, disturbance rejection and set point tracking. The first thing to know in this respect is whether the autotuner exploits the 2-d.o.f. structure of the PID or not, because only in the former case can stability, robustness and disturbance rejection issues be separated from tracking ones. In any case, the main problem for autotuners is that possible inconsistent user requests must be dealt with. This makes industrial products adopt, in extreme situations, one of the three approaches described in the following together with their advantages and potential pitfalls.

4.2.1. Autotuners with no specs

Such autotuners are extremely simple to use, do not aim at particularly sophisticated results but can cope with most problems satisfactorily. The only way to make them fail is to use them in a situation where their identification mechanism (which is normally very simple) can fail significantly, like we have shown in the body temperature example. Fortunately, in practical cases this is not a difficult precaution to take on the sole basis of (conscious) common sense.

4.2.2. Autotuners with lexical (word) specs

These products allow more user control in that they accept specs like ‘minimise the overshoot’, ‘achieve the shortest settling’ or ‘minimise an index’ like e.g. the ISE. This normally reflects in characteristic following tuning made by optimisation, where the spec dictates the cost function to be used. Some autotuners of this kind allow the user to select the regulator structure, others choose it on the basis of measured data. For these autotuners the identification phase is more critical than in the previous case, and that is why more sophisticated techniques are used. It is important to note that this approach is not suited for tight control, because in general it is not easy to forecast how achieving a characteristic is compensated for in terms of the others and how minimising an index reflects on the shape of the achieved transients.

This is particularly true if the regulator structure is selectable, as we now show with an example. Consider a process described by a delay-free first order transfer function. If a PI regulator is tuned for it, *no matter which synthesis policy is adopted*, the open-loop transfer function will be approximated around ω_c by $L(s) = \omega_c/s$. If a PID is adopted, L(s) will have a second zero above ω_c, i.e. it will be approximated by $L(s) = \omega_c(1+s\alpha/\omega_c)/s$, where $\alpha \in (0,1)$ is the high frequency open-loop gain. Suppose now that the (lexical) spec is to minimise the ISE for a step variation of the set point, which in the PI and PID cases turn out to be

$$\mathrm{ISE}_{\mathrm{PI}} = \int_0^\infty e^{-2\omega_c t} dt = \frac{1}{2\omega_c}, \quad \mathrm{ISE}_{\mathrm{PID}} = \frac{1}{(1+\alpha)^2} \int_0^\infty e^{-2\frac{\omega_c}{1+\alpha} t} dt = \frac{1}{2\omega_c (1+\alpha)}.$$

It is apparent that the ratio between the PI and the PID ISE is 1+α, thus the PID appears better in any case. It is also apparent, however, that this ISE improvement is paid in terms of a higher value of the open-loop high frequency gain. Moreover, the bigger this reduction is, the longer and closer to the 0 dB axis the plateau of |L(jω)| after ω_c has to be, which means poor robustness.

Coming to the quality of the output transients, the situation is depicted in Figure 25, where a normalised time $\tau = \omega_c t$ has been introduced for convenience. The ISE reduction provided by the PID is essentially due to the prompt response in the first instants (which requires a bigger control spike), while the settling time *increases*.

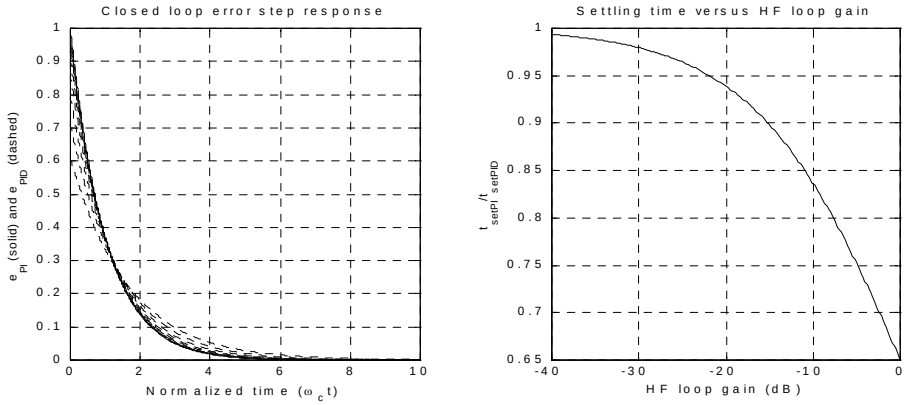


Figure 25: PI versus PID tuning with ISE minimisation.

In this case, minimising the same index means different things if different controllers are selected. Generalising, then, we can say that

- Optimising integral indexes like the ISE is a good policy for achieving 'compromise' solutions where no tight objective dominates all the others, but can sometimes produce unexpected results. It must be kept in mind that an optimiser achieves its goal *whatever this means*, and that if one has desires not directly reflected into an index, problems like this are quite likely to arise.
- In general, autotuners based on index minimisation (and, more generally, on lexical specs) require the user to select the correct index (or spec) and the correct regulator structure for the problem at hand. Index and structure selection is often done automatically, as already discussed, but autotuners like these are not the best choice if tight control is required.

4.2.3. Autotuners with numeric specs

These allow the maximum user control and are an almost obliged choice if tight control is required. Identification is very important and specification correctness even more. In fact, this is the only case where user requirements can prove inconsistent.

This problem is more relevant in autotuners where specs are directly related with 'control-theoretical' characteristics. For example, where the user is allowed to select the cutoff frequency and/or phase margin explicitly. There is no need to prove that these two specs (which are quite common) can easily be inconsistent, so the autotuner must make a choice and fulfil only one of them. The problem is that there is no way of ensuring that this choice will reflect the user requirements. As such, we can state the following:

- Autotuners with numeric specs are to be used wisely. They are powerful tools for tight control, especially if they use the 2-d.o.f. structure, but must be used by people capable of giving specs correctly or at least not in an inconsistent way.
- Before adopting such an autotuner, test several others (if possible) selecting apparently inconsistent requests, see what they do and choose the 'most reasonable' one for the problem at hand: here, there is really no general criterion. It would be appreciated that documentation were available to allow one to forecast the autotuner behaviour with an inconsistent request, but this is rarely the case (and, we must admit, it is very difficult to document these facts satisfactorily).

4.3. Computing and validating the regulator parameters

At this point, an autotuner can compute the PID parameters with the tuning method it is based on. After this step, it is necessary to check the obtained result against some validation criteria. These can be broadly classified in two categories.

4.3.1. Criteria for checking the tuning results

Since in solving the equations required for tuning it is often necessary to employ numerical techniques and even quite crude approximations, it is generally required to check that the desired results have been attained. In model based autotuners it is also possible to forecast the loop behaviour before actually modifying the regulator parameters. For obvious reasons this is not possible in the characteristic based case. In this case, sometimes a small perturbation is given with the tuned regulator and if results are not satisfactory the previous tuning is restored. In the experiment based case this check is often done to recover from the experiment perturbation (this is also useful in the model based case). Product documentation normally says almost nothing on these checks, which is sometimes a pity because their appreciation allows one to understand the autotuner's operation more precisely. In some cases, these checks require additional process perturbation.

4.3.2. Criteria for checking the 'aspect' of the obtained regulator

As a final step, the regulator is checked against some parameter consistency rules; for example, most autotuners verify that Ti>Td. Also the implementation feasibility is checked, typically verifying, say, that no regulator time constant is smaller than 'few times' the sampling time. Finally, very high-end products also check the sensitivity of parameters to specifications, to avoid the situation where a small change in requirements results in a completely different regulator. This is not required, strictly, but nevertheless users are keen to think that the dependence of parameters on specs must be in some sense continuous. Also these checks are normally documented up to a very limited extent, but for selecting a product this is less important.

5. Examples of industrial autotuners
Applying the obtained knowledge to evaluate a product

Throughout the previous sections we have acknowledged the features of the various types of autotuners with the aim of helping the reader understand their operation and, above all, to select one on the basis of the characteristics of his particular control problem. In this section we briefly present some industrial products. Firstly, to show how the features that have been highlighted earlier can be detected in the manufacturers' documentation. Secondly, to see how this information can then be evaluated as for its suitability for a given application.

This section is in two parts. The first part looks at autotuners that are supplied by the controller manufacturers. The second explores some of the new software environments that generally offer analysis and diagnostics as well as loop tuning capabilities. In either case the lists are not extensive but simply represent examples typical of those likely to be encountered by practicing control engineers.

An excellent review of the loop tuning tools available, about five years ago, was presented in an article in the November issue of Control Engineering (VanDoren, 1997). This section complements that article and also provides web site details of the controllers and tuning environments cited, for those readers who wish to obtain more information.

5.1. Controllers with 'built-in' tuning capability

5.1.1. The Foxboro EXACT

In this 'historical' autotuner (Bristol, 1986) the tuning operation is initiated automatically whenever the error exceeds a user-specified threshold. When this occurs it is assumed that either a set point change or a disturbance might have occurred. In both of these cases the error transient may resemble that of Figure 26 (a) and (b), respectively. Heuristic logic – details are not published - is used to decide whether a 'proper' transient has occurred and to compute the peaks e_1, e_2 and e_3 by means of pattern recognition techniques.

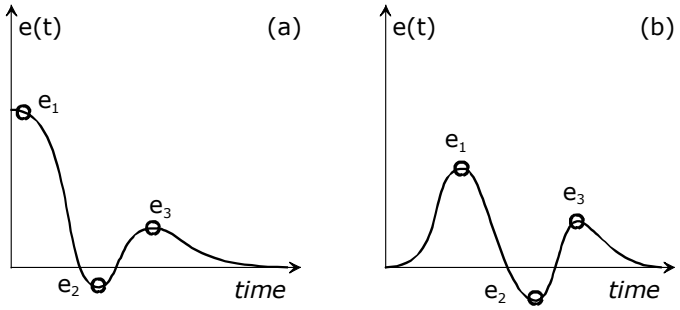


Figure 26: examples of error transients for (a) a set point change and (b) a load disturbance.

Tuning is completed once specified values of 'damping' and 'overshoot' are satisfied, i.e. the two quantities $(e_3-e_2)/(e_1-e_2)$ and $|e_2/e_1|$ match given values. User specifications are given in terms of these quantities: notice that the terms 'damping' and 'overshoot' are *not* used with their normal interpretation. The autotuner uses a complex set of heuristic rules also for this purpose.

The EXACT requires quite a lot of prior information for tuning. For example, it is necessary to tell it how long to wait for an error peak before concluding that it will not occur. The most important thing, however, is in order to speed up convergence, initial regulator controller parameters must be provided. To help the user supply this information, the EXACT has a 'pretune' mode based on an open-loop step test and on the Ziegler-Nichols rules.

The EXACT is not experiment based apart from the pretune phase and from the set point modification case. It is characteristic based (it uses the error response peaks), as well as characteristic following (maximum allowed damping and overshoot in the sense stated above). It features a rule base (it uses heuristics and pattern recognition) and specifies a 1-d.o.f. PID structure. It is a good tool but difficult to completely understand and sometimes to 'drive', given the complexity of the heuristics involved and the limited amount of documentation on it (see the Exact technical information), and Foxboro for controllers using this technique.

5.1.2. The Honeywell AccuTune

This autotuning method, described in (Åström *et al.*, 1993) is implemented in several controllers, see e.g. (Honeywell controllers) where several powerful devices are described with trend removal capabilities, set point programming and so on. It can only be used for stable processes.

It is initiated by setting the controller to manual, then driving the controlled variable to a steady state 'a little away' from the set point before switching to automatic. This initiates a step experiment, where the control step amplitude is calculated (but no information is supplied on how) in order to force the process variable back to the set point. The experiment leads to a first or second order model (the selection is automatic), from which the PID parameters are computed by convenient rules based on cancellation and influenced by the

process order and by the presence of a significant process delay. The PID is then set to automatic and, once the set point is reached, a 'fine tuning' of its parameters is carried out on the basis of steady-state levels. The user can decide whether tuning must be made only on set point changes or also on transients that are assumed to be caused by load disturbance, and which is the minimum set point change that will trigger the tuning.

Thus, the AccuTune is model based with automatic structure selection. It also is experiment based, and characteristic following (it cancels the model poles aiming at fast response with constraints on the phase margin and on the high frequency regulator gain). It uses steady-state information and leaves quite limited user intervention possibilities. It also refers to the 1-d.o.f. PID structure and possesses a continuous adaptation feature. For a list of controllers using this technique, see Honeywell controllers).

5.1.3. The Yokogawa SLPC

This autotuner (see e.g. Yokogawa, 1993) is initiated on user demand and experiment based (a step added to the control signal in closed loop). It is based on a FOPDT model identified by optimisation on the basis of the captured response. It requires the user to specify the 'type' of the response in terms of maximum overshoot (no overshoot, 5%, 10% or 15%), thus it is characteristics following.

Tuning is decided by rules developed statistically on a large number of simulations. These rules minimise an integral index (ITAE, ISE, and so on) chosen on the basis of the response type requested. The SLPC refers to a 2-d.o.f. PID structure where R_{ff} is set by the user and does not use steady-state information. It has a 'pretune' mode based (as usual) on an open-loop step test and has also a continuous adaptation feature. For a list of controllers incorporating this technique see the Yokogawa page.

5.1.4. The ABB ECA 600

The auto-tuning mode has to be enabled by the operator. The method is based on a relay experiment with hysteresis in closed loop. The value of controlled variable should be close to the set-point, when the auto-tuner is initiated. The procedure starts by inserting a small step to the system and this causes the system to oscillate. The step-value has to be selected in advance.

The tuning function itself has similarities with the manual Ziegler-Nichols method, precise information about the tuning method is not available. The auto-tuner adjusts the amplitude so that the process value will not be greater than the level that is necessary to isolate the measurement noise. The auto-tuner is able to judge if the derivative element is necessary. An adaptive procedure is available which can be used as a fine-tuner. For further details, see the ABB web site.

5.1.5. The OMRON E5AK/E5EK

The Omron E5AK/E5EK calculates suitable PID parameters using a fuzzy-self-tuning function. The controller itself determines when tuning is necessary. Three different modes are selectable: step response tuning (SRT), disturbance tuning (DT), hunting tuning (HT). In the SRT mode the controller parameters are tuned, when a new or different set-point change occurs. In the DT mode the PID-parameters are adjusted so that the controller output stays within the target range that has to be specified in advance. In the HT mode the controller parameters are amended once hunting occurs. HT will be enabled when four or more extreme control values occur, provided SRT is not being executed.

5.1.6. The National Instruments Autotuning PID

This quite recent autotuner is initiated on user demand, experiment based (it uses relay feedback on the set point), characteristics based (the process description is given by the ultimate gain and frequency, see e.g. (Åström and Hägglund, 1984), characteristics following with lexical specs (the user has to select the controller type - P, PI or PID - and 'normal', 'fast' or 'slow' tuning).

Parameters are computed by Ziegler/Nichols-like formulae, described in the product documentation. The regulator has an ISA-like form: there is no derivative filter but a (fixed) one on the process variable, output derivation is used and set point weighing in the P action is available: in the online help the weight is called 'relative emphasis of disturbance rejection to set point tracking'. Gain scheduling and a static nonlinear characteristic on the error are also available.

While presenting all the advantages and pitfalls of autotuners obtaining local process information from relay feedback (see the corresponding section), this one has two peculiar features that are worth pointing out. First, tuning is accomplished by the used via a 'wizard', i.e. a sequence of interactive steps guided by windows carrying the necessary instructions and the usual 'next/previous/cancel' buttons to navigate back and forth in the procedure. Interestingly enough, in this wizard the recognition of the steady state prior to the relay test and that of the permanent oscillation are made by the user. This type of interactivity, where human insight is employed for decisions that are particularly difficult to automate, seems to be a trend in new autotuners. It can lead to good results but requires at least a minimum of insight.

The second feature is that both for the controller and for the autotuner the source code (in the 'G' graphical programming language) is visible. This means that a conscious user can inspect it and learn a lot about the autotuner's operation (see the LabVIEW PID Control Toolset User Manual).

5.1.7. Additional Controller Information

The following table reports a short (and in no sense exhaustive) list of other controllers with autotuning and/or adaptation capabilities, together with the web sites where additional information can be obtained.

Controller	Manufacturer	Law	GS	AT	Ad	FF	Web site
ECA 600	ABB	PI(D)pPI	•	•	•	•	http://www.abb.com/
1/16 DIN Universal Process Controller	Athena	P-I-D		•			http://www.athenacontrols.com/pages/tempproc.html
UCD3863	Bristol-Babcock	PID	•		•	•	http://www.bristolbabcock.com/productliterature
2408 PID Controllers	Eurotherm	PID	•	•	•	•	http://www.eurotherm.com/products.htm
Series 2000	FGH	PID		•	•		http://www.fgh.co.uk/
762CNA	Foxboro	P-I-D		•	•		http://www.foxboro.com/m&i/controllers_recorders/specs_controllers.htm
UDC 5000 Ultra	Honeywell	PID		•	•	•	http://catalog.sensing.honeywell.com/
TC16	Leeds&Northrup	PID		•	•		http://www.procinst.com/leeds&northrup-tc8-16.htm
E5AK	Omron	PID			•		http://www.omronsupport.net/knowhow
EC300	Toshiba	PID		•	•		http://www.tic.toshiba.com/
SLPC	Yokogawa	PID		•			http://www.yokogawa.com/MCC/ys80.htm

GS Gain Scheduling
AT AutoTuning
Ad Adaptation
FF FeedForward

5.2. Independent Software Tools

There has been an explosion in the number of loop tuning packages designed to facilitate both novice and experienced user. A limitation of some of the earlier systems was that connection between the PC (containing the tuning software) and the controller relied on an ADC/DAC interface card in the PC and knowledge of the controllers communication protocol which was not always easily accessible. Some systems also used ‘current clamps’ that had to be physically connected to the input and output terminals of the controller to provide the necessary tuning information. Traditionally, each software or application developer was required to write a custom interface to allow data exchanges between the various hardware devices. Most modern systems still provide one or both of the above options but much more reliance is placed on exploiting OPC server technology. OPC is the standard for plant floor communications between data servers and client applications. The OPC specification is a non-proprietary technical specification that defines a set of standards based upon Microsoft’s OLE/COM technology. The application of the OPC standard makes process interoperability straightforward by defining a common, high performance interface that can be reused by

SCADA, control and custom client applications. Some of the features present in these new tools include:

- Data conditioning prior to modelling (outlier and 'bad data' removal)
- Process model derivation (transfer function or frequency response)
- PID controller design including (a) empirical 'look-up' table (b) Åström-Hägglund (c) IMC and (d) optimal methods
- Performance checking through system simulation
- Robustness analysis
- Looking simultaneously at more than one loop
- Process diagnosis to detect (a) hysteresis, (b) stiction (c) oversized/undersized valves (d) excessive measurement noise (e) need for gain scheduling
- What-if simulations
- Report preparation facilities
- Lots of excellent graphics support

Because of space restrictions the intention is to identify some of the more well known products by their web site addresses and this information is tabulated below. At these sites a considerable amount of information is available plus downloads of the software.

Software tools	Web site
Tune Wizard PID Controller Tuning, Process Diagnostics, Loop Simulation	www.tunewizard.com
RaPID Robust Advanced PID Control	www.mathworks.com or www.ismc.be
PROTUNE™ Complete software hardware system that records, troubleshoots and analyses dynamic data	www.protuner.com
EXPERTUNE® PID tuning, analysis and simulation software	www.expertune.com or 169.207.153.173
Control Arts Inc. Model Identification and PID tuning software	www.controlartsinc.com
Rockwell Software RSLoop Optimizer	www.software.rockwell.com
BESTune PID Controller Auto-Tuning software	bestune.50megs.com/

6. Some samples of the current research

How advanced autotuning concepts are put into operation

This section is aimed at presenting some 'research' autotuners taken basically from the authors' experience, so as to provide the reader with at least some samples of what is being done. The descriptions reported here tend to be longer than those of industrial products because the references provided are written in a more 'academic' style.

6.1. A relay autotuner exploring the process Nyquist curve

This autotuner is presented in (Leva, 1993). It first performs a (short) double step test to see if the process is integrating, then identifies one point of its Nyquist curve by a relay experiment. A delay is cascaded to the relay to explore several points, as discussed. This search is aimed at finding the frequency where the process magnitude is 1/10 of the static gain (in the non integrating case) or 0.1 (in the integrating case).

Tuning is made by 'moving' this point onto the unit circle with the user-specified phase margin. The method ensures a 'reasonable' cutoff and a regulator magnitude of at least 20 dB at the cutoff, which results in good disturbance rejection.

The autotuner is initiated on demand, characteristic based, characteristic following, and uses static information. An example autotuning session is shown in figure 27.

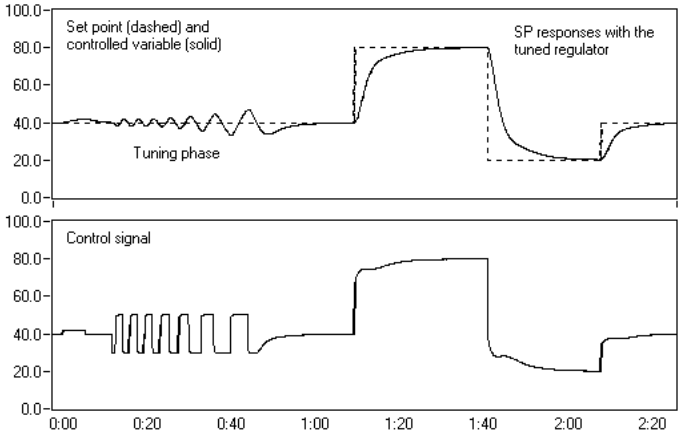


Figure 27: autotuning session.

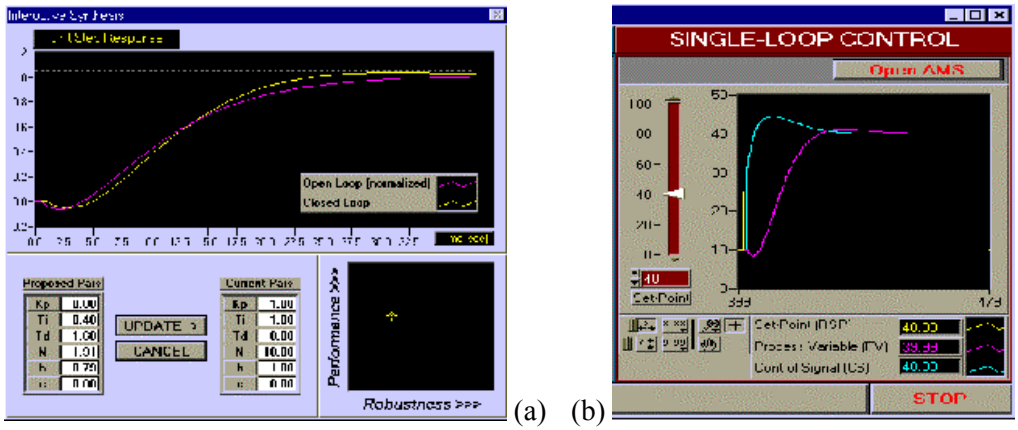
6.2. A robust autotuner based on the IMC and on optimisation

This autotuner is described in (Leva and Colombo, 2001a) and works for asymptotically stable processes. It is based on the identification of a FOPDT model in the form (15) by an open loop step test and the method of areas. Subsequently, the IMC formulae (18) are used for synthesising the block $R_{fb}(s)$, see figure 11.

Parameter λ can be chosen by the user, but a limit on it is imposed on the basis of an estimate of the additive model error magnitude obtained in the identification phase with the method described in (Leva and Colombo, 2000). The computation of this limit on λ is discussed in (Leva and Colombo, 2001b); in so doing, λ becomes a stability degree user request over a minimum forced by the autotuner on the basis of data. After $R_{fb}(s)$ is tuned, a convolution model of the loop part of the control system is computed. This contains $R_{fb}(s)$ and the response data, not the FOPDT model, so that no structural hypotheses on the process dynamics are involved.

Finally, $R_{ff}(s)$ is tuned by minimising with respect to b and c the ISE between the control system response (computed with the convolution loop model) and that of a first order transfer function with unity gain and delay equal to that of the FOPDT model, whose time constant becomes the user set point tracking performance request and cannot interact with stability, robustness and disturbance rejection because it only affects $R_{ff}(s)$. This method for optimising the weights b and c is described in (Leva and Colombo, 1999).

This autotuner is then experiment based, model based, model following (by cancellation for R_{fb} and by optimisation for R_{ff}), initiated on demand. The user interface of the autotuner has been designed so that the user can modify the stability and performance request and see the achieved results (computed with the convolution model) on line; this has proven to be highly appreciated by several people with very different degree of control culture. An example session with the autotuner is shown in figure 28.



(a) (b)

Figure 28: the autotuner's user interface (a) and results (b).

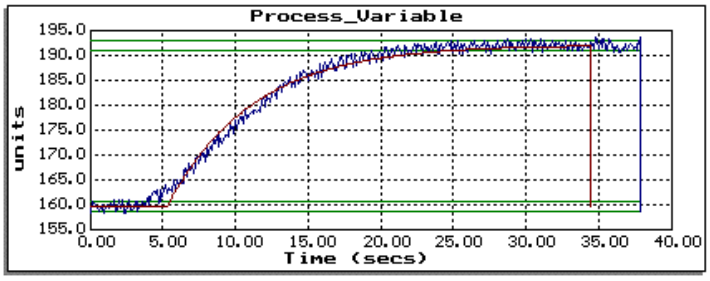
6.3. The MasterTune CAD Software

The **MasterTune** software has been developed to help in the automatic tuning of PI, PID and pPI controllers. The kernel of the **MasterTune** software (Cox *et al.*, 1994) for the PI and PID cases is the Åström-Hägglund (1984) autotuner. However, it has a family of additional refinements that make it easy to use whilst at the same time producing consistent behaviour at least equivalent to that obtained when using well tried empirical formulae (Cox *et al.*,1997). The main features of the software are briefly outlined next. During the tuning

phase the values of the various test parameters used are normally set to default values by the software. If the user so chooses, these values can be modified to suit individual requirements. The parameters which can be modified include:

- The percentage overshoot: this is the amount of overshoot the closed loop system will exhibit when compensated by the tuned PI controller. The percentage overshoot is used to determine the phase margin required by the design equations.
- The relay characteristics: the amplitude (to control the size of the limit cycle oscillations during tuning) and the hysteresis (used to prevent false switching caused by noisy signals).
- Tolerance between peaks: used by the software to detect when the tuning phase has been completed.
- Constraints: control over the allowable level variations of both the process and manipulated variables.

The process analysis phase is the preliminary step of the tuning procedure. Here an open loop step test is conducted and a FOPDT model automatically calculated using the characteristic area method (Nishikawa et al. 1984). The process inputs and outputs are visually displayed to the process operator throughout this phase. On completion of the test, the model is overlaid onto the process reaction curve allowing an informed judgement to be made about the quality of the model (figure 29).




```
Estimated process parameters :
    Gain 3.259   Tau 5.855s   Dead time 3.381s
Recommended relay parameters :
    hysteresis 1.005   height 2.991
Predictive-PI settings :
    Xp 325.916   Ti 5.855s
Est. cycle period :
    31.270
```

Figure 29: typical process analysis result

PI QuickTune

If the time delay is not appreciable in relation to the time constant the software recommends that a PI controller be used. This fast tune facility uses the transfer function obtained in the process analysis phase to calculate the controller settings using some empirical formula, typically those recommended by Cox et al (1997) - reference. This feature can be used to establish a satisfactorily tuned control loop in a very short period of time (relevant of course to the dynamics of the process).

Predictive PI (pPI)

If the time delay is large compared to the principal time constant, then a PI controller will not be able to yield satisfactory performance. In this situation the software recommends that the pPI strategy is used and calculates the controller parameters. For implementation of the pPI algorithm it is recommended to use a programmable process controller.

Setting the relay characteristics

If, as recommended, the fast tune procedure is by passed in favour of relay feedback autotuning, then the relay characteristics must be set. The process analysis phase calculates a value for the relay height that will produce a limit cycle with an amplitude approximately equal to that set in the 'Set test parameters' phase above. Additionally, the software will analyse the level of noise imposed on the process variable and use this value to suggest a level of hysteresis.

This stage relieves the process operator of the task of specifying the relay characteristic. The operator only needs to specify the maximum process variable swing allowed by the closed loop system. Further, it should be noted that this stage may only be required when tuning a process for the first time. In subsequent sessions, the relay parameters can be entered directly based on previously recorded values.

AutoTune (PI strategy)

When the autotune in invoked, the relay and integrator force the process variable to oscillate with the specified amplitude, see figure 30. When the tuning phase is completed as indicated by a flag, a PI controller is calculated which will result in a system with the requested level of overshoot

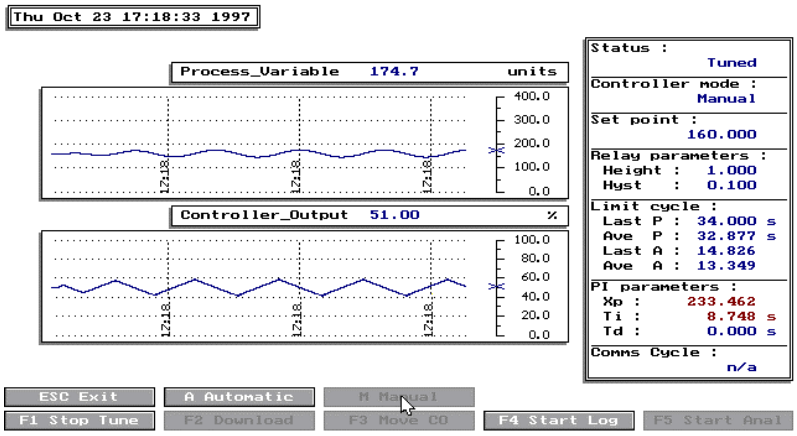


Figure 30: CAD software screen dump showing Autotuning in process

Evaluation

Once the appropriate controller has been selected and tuned, the evaluation stage can be implemented. Here the controller is put into automatic mode and, if desired, a step change can be induced in the set-point (see figure 31). This, as well as each other stage in the tuning cycle, can be logged and saved in a data file to provide a full record of the tuning cycle.

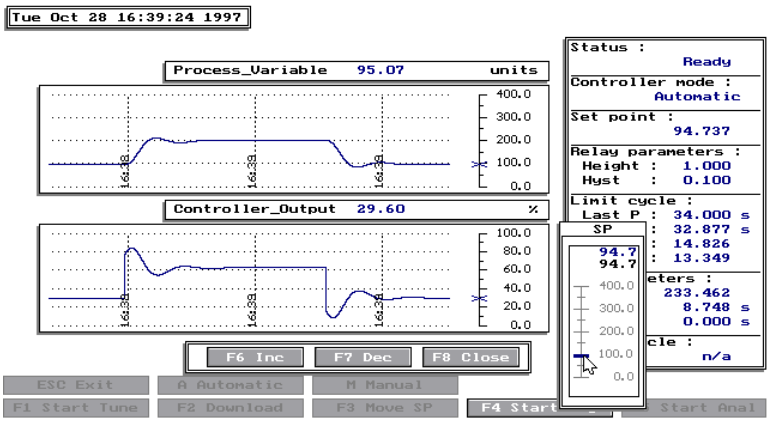


Figure 31: CAD software screen dump showing evaluation of PI controller

The **MasterTune** software incorporates many attractive features that provide an automatic medium for PID controller parameter determination. Another attractive feature is the ability to download the parameters directly into the actual process controller once the values are accepted. When this stage is complete the PC is disconnected form the controller. A new tuning cycle with a different controller begins once the communications link is again established.

7. Soft-Computing methods for PID Autotuning
A novel alternative and a very promising research field

7.1. What is Soft-Computing?

Soft-computing is a relatively recent collection of methodologies, which have been inspired by natural phenomena. The term 'soft' was coined by L. A. Zadeh (Zadeh, 1965) as opposed to conventional 'hard'-computing, in order to emphasise their tolerance for imprecision and uncertainty. The core methodologies of soft-computing are *fuzzy logic*, *neural networks* and *evolutionary computing*. Although these methodologies have different genesis, they can be seen as complementary rather than competitive. Recently, hybrids systems, such as neuro-fuzzy, neuro-genetic or even neuro-fuzzy-genetic systems have been proposed, and it is our feeling that this trend will increase in the future. Before describing some applications in PID autotuning, let us briefly introduce each methodology.

Artificial neural networks were inspired by the human brain operation. They consist of processing elements (the *neurons*), each one with very small processing capability, densely inter-

connected in a *network* through the use of *weights*, each neuron acting independently of the others. A neural network, as the brain, has the capability of *learning*. There are a large variety of neural networks, employed for different purposes, and using different paradigms of learning (the interested reader can consult, for instance (Haykin, 1999), (Principe et al, 2000). Here we only mention the ones used in the chosen applications.

In PID autotuning, they are used for *nonlinear function approximation* purposes. The most widely know neural networks are *Multilayer Perceptrons* (MLPs). They have, as the name indicates, a layered structure, consisting of an *input layer* (a buffer), one or more *hidden layers* where typically *sigmoidal* functions are used, and an *output layer*, where linear functions are usually employed. Although different learning algorithms can be employed, the most well-known is the *error back-propagation* (BP) algorithm. This belongs to the class of *supervised learning* algorithms (supervised means that a set of desired outputs is available for a set of inputs, and the role of the learning algorithm is to minimise the sum of the squared errors between these two quantities), and the BP algorithm performs its task that by computing, in an iterative fashion, the gradient of the criterion with respect to the weights and adapting them in the direction opposite to the gradient. The name of the algorithm comes from the fact that the gradient is computed by propagating the error from the output towards the input, therefore 'back'-propagating it. If the set of examples is fixed through all the iterations and the gradient is computed for the whole set, then we have *batch learning* or *training*, otherwise we have *pattern-based learning*. Of course for on-line applications this is the procedure adopted, and learning becomes *adaptation*.

B-spline neural networks have also been employed for PID autotuning. They have also a three-layered structure, but the basis functions employed have a *compact support*, which means that they are active only for specific sub-domains within the larger domain covered by the inputs. A more detailed description of these networks is beyond the scope of this work and the interested reader is conducted to (Brown and Harris, 1994) for more information. We only mention that these networks have interesting properties for adaptation (performing on-line learning in a neighbourhood around an operation point minimally affects the information stored for the whole application domain) and can be interpreted as fuzzy systems.

Fuzzy systems were first introduced by Zadeh (Zadeh, 1965) as a means for handling and processing vague, linguistic information (as humans do). It allows variables to be partial members of a particular set and uses generalisation of conventional Boolean algebra to manipulate this information. Fuzzy information is represented by a set of fuzzy rules:

$$\text{IF } \left(x_1 \; is \; A_1^i\right) \text{ AND ... AND } \left(x_n \; is \; A_n^i\right) \text{ THEN } \left(y \; is \; B^j\right) \quad (c_{ij})$$

The terms A_k^i are linguistic variables which represent vague terms such as 'small', 'medium' or 'large', defined on the input and output variables. Each rule maps the antecedent, formed by the intersection of the n univariate linguistic statements $\left(x_k \; is \; A_k^i\right)$, to the consequent, formed by a single univariate linguistic statement $\left(y \; is \; B^j\right)$. Associated with is rule is a variable (c_{ij}) which describes the confidence in the particular rule being true. To implement

a fuzzy algorithm, the fuzzy sets need to be defined and the functions used to implement the fuzzy operators chosen. To be used in the real world, a numerical *crisp* value needs to be *fuzzified*, i.e, the *degree of membership* of each of the linguistic fuzzy sets needs to be calculated, and, on the other hand, a real-valued output is obtained by defuzzifying the fuzzy output set, which is formed from the contributions of each fuzzy rule. There are different types of fuzzy models. The most common are the Mamdani fuzzy model (Mamdani, 1975), whose global structure was described above, and the Takagi-Sugeno model (Takagi and Sugeno, 1985), where the consequents are polynomials. There are various excellent introductory books in fuzzy systems, such as (Driankov *et.al.*, 1993).

Evolutionary algorithms are founded on Darwinian principles of evolution of species. Different algorithms follow under this umbrella, such as *Genetic Algorithms* (GAs), Evolution Strategies, and Evolutionary Programming. Among these, GAs have been, since their introduction by Goldberg (Goldberg, 1989), the mostly used and actually the only ones used for PID autotuning. GAs are a powerful stochastic method for performing global search and optimisation. Instead of working with only one solution, which is evolved iteration-by-iteration, they employ a population of possible solutions to the problem, and it is this population that is evolved over different *generations*. Each particular individual is expressed in a particular genetic code and is assigned a *fitness value*, which describes how well it performs for the problem at hand. Evolution is performed by applying *genetic operators* for the current population. These include *selection* (based on the fitness values assigned to the individuals within the current population, only some of them are selected for propagating its genes for subsequent populations), *crossover* (which involves the exchange of genetic material between two parents to create new off-springs) and *mutation* (performing random changes to individual chromosomes). Once the new generation is obtained, the process goes on iteratively for a specified number of generations.

Genetic algorithms are robust algorithms, which look for *global optima*, instead of *local optima*, as gradient-based algorithms do. They have been found to cope well with noise, discontinuities, and are very well-suited for performing *multi-objective* (MO) optimisation. The interested reader can be found more detailed information in another IFAC Professional Brief – Genetic Algorithms in Control Systems Engineering (Fleming and Purshouse, 2002), and the references there within.

7.2. Neural Networks approaches to PID autotuning

The first (historically) approaches are due to (Swiniarski, 1990) and (Lightbody and Irwin, 1991). Both exploit the approximation capabilities of MLPs to approximate the mapping between samples of the step response to the PID (PD in the second case) gains. The first approach uses the open-loop output samples, while the second uses the closed-loop output under PD control of a nominal plant. Both approaches suffered from several problems, the most important being requiring a very large of samples (and hence neural network inputs) and being obviously dependent on the sampling time.

Several authors have employed neural networks for adaptive PID control. As this is not the theme of this work, this route will not be detailed. We shall just point out that they usually two different neural networks: one, acting as a plant model (generally determined off-line), and that is used on-line to back-propagate the error between the reference and the plant output to the second neural network, which supplies the PID gains. This latter network is adapted on-line in order to minimise the square of the error between the reference and the output at each time instant. References to approaches in this route can be found in (Omatu et. al., 1995 and 1999).

Ruano and co-workers (Ruano et. al., 1992) employed MLPs in a characteristic-based autotuner. Here, they evaluated the process transfer function, in the real axis, in specific points. These were used, as they can be obtained, on-line, using integral measures of the reference step response, in the open loop case, or the step response and control signal, under PID control. This enabled to identify the plant, on-line, that belonged to a large domain of plant transfer functions and for each transfer function type, a large range of its parameters. By 'sampling' this plants domain, and determining, for each example, the PID gains, according to a user-defined criteria (in this paper, the ITAE criterion was used), a set of examples was obtained for off-line training three MLPs, each one responsible for a PID parameter. After these steps were realized off-line, the on-line operation consisted on applying a reference step, determining the identification measures using the output, or the output and the control signals, apply those to the MLPs, and finally the PID values to the controller. Figure 32 illustrates an example of this method.

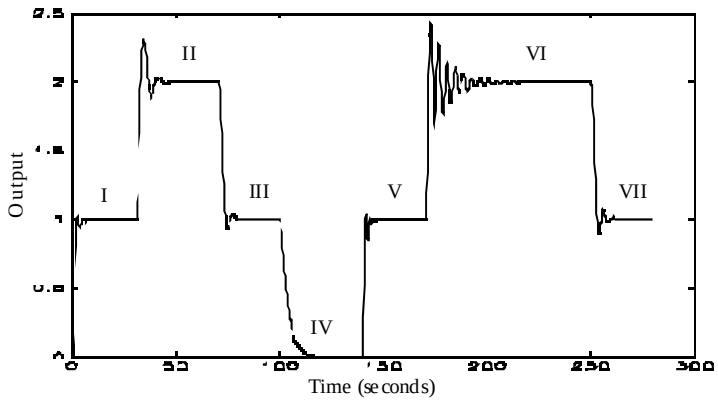


Figure 32: results of PID autotuning

The plant was allowed to change between steps, once the transients related to the previous step had vanished. The initial value of the PID parameters was computed from the open loop step response. Further PID values were computed from the closed-loop step response. The following sequence of changes to the plant was investigated:

$$\frac{e^{-s}}{s+1} \rightarrow \frac{e^{-s}}{(s+1)^2} \rightarrow \frac{1}{(s+1)^2(0.5s+1)} \rightarrow \frac{1}{(s+1)^2(0.5s+1)^2}$$

At time 0 the loop was closed, and step I was applied to the reference input. The transfer function of the plant was

$$\frac{e^{-s}}{s+1}.$$

When step II was applied, the plant had been changed to

$$\frac{e^{-s}}{(s+1)^2}.$$

Since the PID parameters were tuned for the first plant, a considerable overshoot is present. The plant remains unchanged when step III is applied and better results are obtained, since the PID has retuned as a consequence of the observation of the results of the last step. In step IV a plant with transfer function of

$$\frac{1}{(s+1)^2(0.5s+1)}$$

is assumed. The response is now typically overdamped. The controller retunes and a faster response is now obtained in step V. In step VI the plant had changed to

$$\frac{1}{(s+1)^2(0.5s+1)^2},$$

and an oscillatory response is obtained. The controller is again retuned for step VII, and again a better response was achieved.

This approach has been subsequently refined, in order improve its performance as a result of on-line operation. In the previous approach, the neural networks were only trained off-line, their weights remaining fixed subsequently. In (Ruano and Azevedo, 1999) an additional neural network (the criterion network) was incorporated, to capture the mapping between the identification measures and the PID parameters to the performance criterion, the ITAE. As this criterion can be computed on-line, this neural network can be adapted on-line, therefore improving its knowledge about the system as a result of on-line operation. This enables to adapt the original neural networks – PID networks (responsible for delivering the parameters to the PID controllers) - using the optimal PID values resulting from an on-line optimisation, using the criterion network to supply estimates of the ITAE criterion as function of the PID parameters. In this approach, due to their useful properties for on-line adaptation, B-spline networks are used.

Figure 33 illustrates the performance of the auto-tuner, when the plant time delay (here a FOPDT plant is considered) changes between $1.0 \rightarrow 1.05 \rightarrow 1.1 \rightarrow 1.05 \rightarrow 1.0 \rightarrow \cdots$. The transitions occur at every 10th reference step. The purple symbols indicate the case where the learning rate employed for the PID network (λ_{PID}) and the ITAE network (λ_{ITAE}) were 0.02 and 0.5, respectively, and the green symbols the case where $\lambda_{PID} = 0.05$ and $\lambda_{ITAE} = 0.2$.

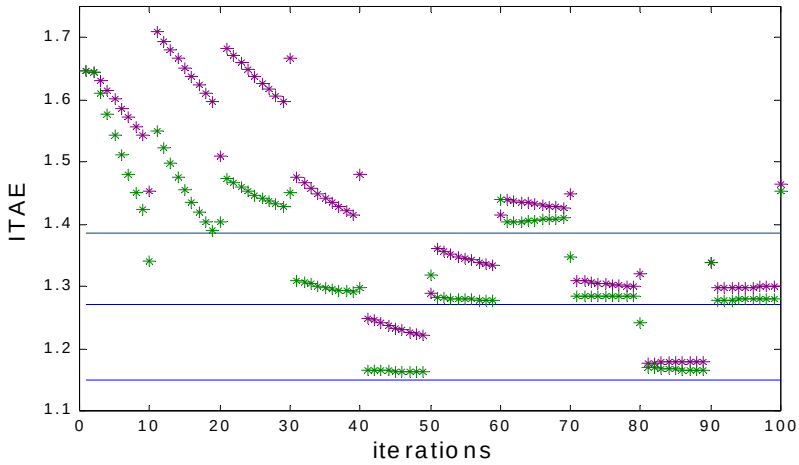


Figure 33: On-line performance of the PID autotuner

It can be seen that the ITAE is converging to its optimal value, indicated by the three solid lines (each one for each plant) in the figure.

7.3. Fuzzy Logic approaches to PID autotuning

There are a significant number of contributions related with fuzzy PI or PID-like controllers, which will not be mentioned here. As in the neural network approaches, most of the applications focus in adaptive PID control. The interested reader can consult, for instance, the works of (Zhao et al, 1993), (Pfeiffer and Isermann, 1994), (Visioli, 1999), (Visioli, 2001), the LabVIEW PID Control Toolset User Manual and a commercial controller, Honeywell UDC 6300, incorporating fuzzy logic.

Hyeong-Pyo Hong et. al. (1992) proposed a scheme where some characteristics of the step response (first-peak, ratio of settling time, ratio of time constant and ratio of the first peak) were identified and converted into linguistic values. Using a set of six fuzzy rules, the fuzzy inference engine computes fuzzy values, which, defuzzified, are used as adjusting (multiplying) factors for the current PID parameters.

A similar scheme is presented in (Iwasaki et. al., 1993). Again, characteristics of the reference step response (overshoot, time duration of the overshoot, ratio between the integral parameter and the rise time, saturation time) are fuzzified and used as an input to a fuzzy system which outputs the PID parameters. The main difference is that, previously, the process is classified according to the ratio of the delay time to the time constant (a FOPDT model is assumed) in three different classes: lag, middle and dead-time plant, and for each case a specific fuzzy rule set is used.

An example of a neural-fuzzy approach to PID supervision is described in (Henriques *et al*, 1999). Here they address the problem of the PID control of a solar power plant. One charac-

teristic of this problem is that the operating point varies throughout the day, causing changes in the plant dynamics which can not obviously be controlled. Different PID controllers were designed for different operating points, based on neural network plant models. Based on measured data related with the solar radiation and the reference temperature, a fuzzy supervisor selects, among the available controllers, the one which achieves the maximum output value, among the rules that have been fired simultaneously.

7.4. Genetic approaches to PID autotuning

Being genetic algorithms powerful optimisation methods, it is not surprising that they have been applied for PID tuning. The first known (by the authors) approach to PID autotuning is due to (Jones and Oliveira, 1995). Assuming that the process is modelled by an ARMA process, they use first a genetic algorithm to estimate the model parameters based on the closed-loop step response. Then they use this model to determine, again using genetic algorithms, the PID gains that minimise a cost function such as ISE, IAE or ITAE. The same authors extended this work to use co-evolutionary design in (Jones and Oliveira, 2000). Different authors have used genetic algorithms for PID tuning, the major problem being to automate the techniques for on-line operation.

As mentioned before, genetic algorithms are well suited for performing multi-objective optimisation. This approach was introduced in (Lima and Ruano, 2000) for PID autotuning, where they compared genetic algorithms with standard optimizers for the simultaneous optimisation of criteria related with reference tracking and disturbance rejection. This scheme is being extended to accommodate additional criteria, and integrated in the learning scheme proposed in (Ruano and Azevedo, 1999). As an example of this technique, figure 34 illustrates an example of a closed loop step response of a SOPTD plant, where the PID tuning was obtained from a multi-objective genetic optimisation, considering criteria such as the ITAE, overshoot, rise-time, control peak value, and additional criteria related with noise rejection.

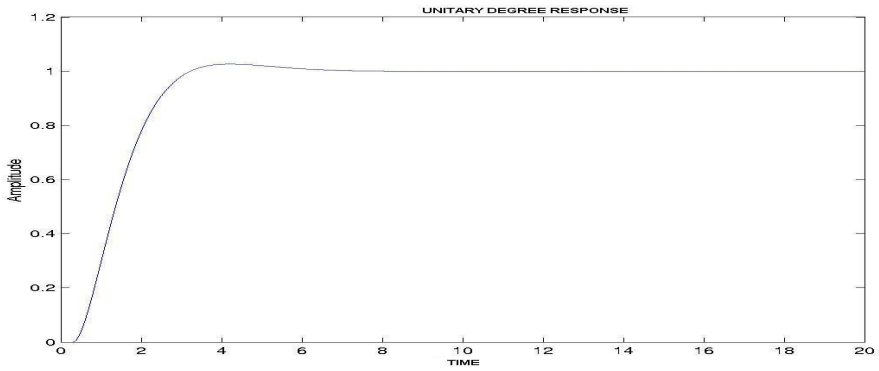


Figure 34: a step response obtained with MOGA PID autotuner

8. Selecting an autotuner

How the concepts introduced can be joined to form a *modus operandi*

Selecting an autotuner is never easy and the role of experience is so important that the knowledge gathered from any work like this can only serve as a starting point. A 'way of thinking' for this selection process can be obtained by recalling the numerous 'selection oriented' statements made throughout the volume, but we think it is useful to complete this knowledge with some final guidelines. Assuming that the problem is to choose one or more autotuners to be employed in a sufficiently well identified application or type of applications (a quite realistic scenario), experience recommends to proceed more or less as follows.

- Identify the characteristics of the control problem(s) involved in the application in the terms introduced here (tight control or not, dominance of tracking or rejection, degree of nonlinearity, and so on). In particular, identify loops that require gain or parameter scheduling, because performing several (auto)tuning operations in different conditions is one of the most efficient ways for selecting the scheduling characteristics.
- Find out the really critical ones, taking into account also past process experience. Concentrate efforts on these, since for the others the choice of the autotuner will not be crucial. This means that for non critical loops regulators can be chosen in any other way and the autotuner they encompass (if any) can be used without worrying particularly of its characteristics.
- Identify who will use the autotuners for the critical loops, which provides information on what level of knowledge the autotuners to be chosen can or cannot require on the part of the user.
- At this point all the information is available for choosing the best autotuning policies for all the critical loops with the guidelines provided along this volume, plenty of product information and the consciousness required for interpreting it.
- If this choice appears to call for adopting too many different types of product, figure out the cost of system complexity versus performance and flexibility. This means pondering whether adopting the same product for two different problems (one more suited for it than the other) is likely to produce so big a performance loss to make the added complexity of two products acceptable. In these considerations, be extremely careful in considering compatibility among products, and preferably make sure they all respect a well established communication standard. This is very important because using several autotuners in an integrated process environment almost always requires them to communicate somehow.
- In extremely complex cases, consider that nowadays selecting an autotuning policy does not always mean selecting a product. There are several SCADA environments providing 'off-line tuning', i.e. capable of making an experiment on the process and then compute the PID parameters in the computer running the SCADA with plenty of different meth-

ods. Parameters can then be downloaded to the regulator if the communication channel allows to do so, otherwise input manually.

- In even more complex cases, consider implementing a set of autotuning policies tailored for the application(s) at hand in a SCADA environment with user programming features. This can be done only with a deep knowledge of autotuning (much deeper than that achievable from this volume) but in some (few) cases it is the only realistic alternative.

9. Conclusions

The goal of this work is to help control professionals to select a PID autotuner effectively. We have decided to pursue this goal by inducing consciousness on autotuners' theory of operation rather than by presenting a large number of products 'flatly'. As a consequence, this volume does not lead to a selection table but to a set of concepts that, once mastered, allows to evaluate how much a product fits an application.

We have proposed a classification of autotuners that is slightly different from the one most widely adopted in the academic literature. This must not be taken as a criticism to that classification, however. Our point is that the proposed one reflects more precisely the *operational* aspects of autotuning, being based on the three questions (a) what type of process information it uses? (b) how is the desired behaviour of the control system specified? (c) how does it act for achieving its objectives? As such, we think that this classification may be less suited for methodological discussions but is more useful for selecting a product.

The monograph has devoted a significant space to the review of PID control principles. This has been done essentially for less experienced readers. We did not intend to provide a complete treatment of PID control, rather to recall the most relevant facts with a notation consistent with the rest of the work. The same rationale is behind the section devoted to tuning methods, with the difference that mastering these is very important (not to say crucial) for selecting an autotuner properly.

Some industrial products have been presented to illustrate how the concept introduced for classifying and selecting autotuners reflect in the real world. Some samples of the current research have also been given based on the authors' experience.

We sincerely hope that our intent has been successful.

References

Åström, K.J. and T. Hägglund (1984). *Automatic Tuning of Simple Regulators with Specifications on Phase and Amplitude Margins*. Automatica, **20**, pp. 645-651.

Åström, K.J. and T. Hägglund (1995). *PID Controllers: Theory, Design and Tuning – Second Edition*. Instrument Society of America.

Åström and Hägglund (2000). *The Future of PID Control*. Proc. IFAC Workshop on Digital Control - Past, Present and Future of PID Control (PID 00), Terrassa (E).

Åström, K.J., C.C. Hang, P. Persson and W.K. Ho (1992). *Towards Intelligent PID Control*. Automatica, **28** (1), pp. 1-9.

Åström, K.J., T. Hägglund, C.C. Hang and W.K. Ho (1993). *Automatic Tuning and Adaptation for PID Controllers: a Survey*. Control Engineering Practice, **1** (4), pp. 699-714.

Bialkowski, W.L. (2000). *Control of the Pulp and Paper Making Process*. In S. Levine (Ed.), "Control System Applications", CRC Press, pp. 43-66.

Bristol, E.H. (1986). *The EXACT Pattern Recognition Adaptive Controller, a User-oriented Commercial Success*. In Narendra (Ed.), "Adaptive and Learning Systems", Plenum Press, pp. 149-163.

Brown, M. and C. Harris (1994) *Neurofuzzy Adaptive Modelling and Control*. Prentice-Hall.

Control Engineering (May 1998). *Single-Loop Controllers Dominate Marketplace*.

Cox, C.S., W.J.B. Arden and A.F. Doonan (1994). *CAD Software Facilitates Tuning of Traditional & Predictive Control Strategies.* Proc. ISA/94-Advances in Instrumentation and Control, **49** (2), pp. 241-250, Anaheim, CA.

Cox, C.S., P.R. Daniel and A. Lowdon (1997). QUICKTUNE: A Reliable Automatic Strategy for Determining PI and pPI Controller Parameters Using a FOPDT Model. Control Eng. Practice, **5** (10), pp 1463-1472.

Dahlin, E.G. (1968). *Designing and Tuning Digital Controllers*. Instrumentation and Control Systems, **41** (6), pp. 77-81.

Doyle, J.C., B.A. Francis and A.R. Tannenbaum (1992). *Feedback Control Theory*. MacMillan.

Driankov, D., H. Hellendoorn and M. Reinfrank (1993). *An Introduction to Fuzzy Control*. Springer Verlag.

EnTech™ (1994). *Competency in Process Control – Industry Guidelines, version 1.0.*

Fleming, P.J. and R.C. Purshouse (2002). *Genetic Algorithms in Control Systems Engineering*. IFAC Professional Brief.

Goodwin, G.C., S.F. Graebe and M.E. Salgado (2001). *Control System Design*. Prentice-Hall.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Haalman, A. (1965). *Adjusting Controllers for a Deadtime Process*. Control Engineering, July, pp. 71-73.

Haykin, S. (1999). *Neural Networks: a Comprehensive Foundation - 2nd edition*. Prentice-Hall.

Henriques, J., A. Cardoso and A. Dourado (1999). *Supervision and C-Means Clustering of PID Controllers for a Solar Power Plant*. Int. Journal of Approximate Reasoning, 22, pp. 73-91.

Hong H-P, S-J Park, S-J Han, K-Y Cho, Y-C Lim, J-K Park and T-G Kim (1992). *A Design of Autotuning PID Controller using Fuzzy Logic*. 1992 Int. Conf. on Industrial Electronics, Control, Instrumentation and Automation, pp. 971-976.

Isermann, R., K.H. Lachman and D. Matko (1992). *Adaptive Control Systems*. Prentice-Hall.

Iwasaki, T., A. Morita and H. Maruyama (1993). *Fuzzy Autotuning with Model Classification*. Japanase Journal of Fuzzy Theory and Systems, **5** (3), pp. 435-446.

Jones, A.H. and P.B. Oliveira (1995). *Genetic Auto-Tuning of PID Controllers*. Galesia 95, **13**, pp. 141-145.

Jones, A.H. and P.B. Oliveira (2000). *Co-Evolutionary Design of PID Control Structures*. Proc. IFAC Workshop on Digital Control - Past, Present and Future of PID Control (PID 00), Terrassa (E), pp. 205-213.

Kessler, C. (1958a). *Das Symmetrische Optimum, Teil I* (in German). Regelungstechnik, **6** (11), pp. 395-400.

Kessler, C. (1958b). *Das Symmetrische Optimum, Teil II* (in German). Regelungstechnik, **6** (12), pp. 432-436.

Leva, A. (1993). *PID Autotuning Algorithm Based on Relay Feedback*. IEE Proceedings-D, **140** (5), pp. 328-338.

Leva, A. and A.M. Colombo (1999). *Method for Optimising the Set-Point Weights in ISA-PID Autotuners*. IEE Proceedings - Control Theory and Applications, **146** (2), pp. 37-146.

Leva, A. and A.M. Colombo (2000). *Estimating Model Mismatch Overbounds for the Robust Autotuning of Industrial Regulators*. Automatica , **36**, pp. 1855-1861.

Leva, A. and A.M. Colombo (2001a). *Implementation of a Robust PID Autotuner in a Control Design Environment*. Transactions of the Institute of Measurement and Control, **1**, pp. 1-20.

Leva, A. and A.M. Colombo (2001b). *IMC-based Synthesis of the Feedback Block of ISA-PID Regulators*. Proc. ECC 2001, Porto (P).

Lima, J.M. and A.E Ruano (2000). *Neuro-Genetic PID Autotuning: Time Invariant Case*, IMACS Journal of Mathematics and Computers in Simulation, **51**, pp. 287-300

Lightboby, G. and G. Irwin (1991). *Neural Networks for Nonlinear Adaptive Control*. Proc. 1st IFAC Conference on Algorithms and Architectures for Real-Time Systems.

Mamdani, E. H. and S. Assilian (1975). *An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller*. Int. J. of Man-Machine Studies, pp. 1-13.

Morari, M. and E. Zafiriou (1989). *Robust Process Control*. Prentice-Hall.

Nishikawa, Y. Sannomiya, N. Ohta, T. and Tanaka H. (1984). *A Method for Auto-tuning of PID Control Parameters*, Automatica, **20** (3), pp. 321-32.

Omatu, S., M. Khalid and R. Yusof (1995). *Neuro-Control and their Applications*. Springer-Verlag.

Omatu, S., T. Fujinaka, Y. Kishida and M. Yoshioka (1999). *Self-Tuning Neuro-PID for SIMO systems*, Proc. European Control Conference (in CD-Rom).

Pfeiffer, B.M and R. Isermann (1994). *Self Tuning of Classical Controllers with Fuzzy Logic*. Mathematics and Computers in Simulation, **37**, pp. 101-110.

Príncipe J., N. Euliano and W. Lefebvre (2000). *Neural and Adaptive Systems: Fundamentals through Simulations*. John Wiley & Sons.

Rohrs, C.E., J.L. Melsa and D.G. Schultz (1993). *Linear Control Systems*. McGraw-Hill.

Ruano, A.E., P.J. Fleming and D.I. Jones (1992). *A Connectionist Approach to PID Autotuning*, IEE Proceedings-D, **139** (3), pp. 279-285.

Ruano, A. E. and A.B Azevedo, (1999) *B-Splines Neural Networks Assisted PID Autotuning*. International Journal of Adaptive Control & Signal Processing, **13** (4), pp. 291-307.

Sanchez–Pena, R.S. and M. Sznaier (1998). *Robust Systems: Theory and Applications*. John Wiley & Sons.

Scattolini; R. and N. Schiavoni (1995). *Regolatori PID e Metodi Classici di Taratura* (in Italian). Proc. ANIPLA Workshop on Advanced PID Regulators for Industrial Processes, Milano (I).

Smith, O.J.M. (1957). *Close Control of Loops with Dead Time*, Chemical Engineering Progress, **53**, pp. 217-219.

Swiniarski, R. (1990). *Novel Neural Network based Self-Tuning PID Controller which uses Pattern Recognition Technique*.Proc. IEEE Automatic Control Conference, **3**, pp. 3023-3024.

Takagi, T. and M. Sugeno (1985). *Fuzzy Identification of Systems and its Applications to Modelling and Control*, IEEE Transactions on Systems, Man & Cybernetics, **15** (1), pp. 116-132

Van Doren, V.J. (1997). *PC-based Control Package Includes Everything*, Control Engineering, November 1997.

Visioli, A. (1999). *Fuzzy Logic Based Set-point Weighting for PID Controllers.* IEEE Transactions on Systems Man & Cybernetics, Part A, **29**, pp.587-592.

Visioli, A. (2001). *Tuning of PID Controllers with Fuzzy Logic*. IEE Proceedings-D, **148** (1), pp. 1-8.

Zadeh, L. A. (1965). *Fuzzy Sets*. Inform. Control, **8,** pp. 338-353.

Ziegler, J.G. and N.B. Nichols (1942). *Optimum Settings for Automatic Controllers*. Trans. ASME, **64**, pp. 759-768.

Zhao, Z., M. Tomizuka and S. Isaka (1993). *Fuzzy Gain Scheduling of PID Controllers*. IEEE Transactions on Systems Man & Cybernetics, **23** (5), pp. 1392-1398.

About the authors

Alberto Leva was born in 1964 in Milano, Italy. In 1989 he received the MSc (Laurea) Degree in Electrical Engineering from Politecnico di Milano, Italy. In 1991 he joined the Department of Electronics and Information of the Faculty of Engineering at Politecnico di Milano, where in 1997 he became Assistant Professor of Automatic Control. His main research interests are process modelling and simulation (especially of power plants), automatic tuning of industrial regulators (with particular emphasis on application-oriented issues) and the development of innovative tools for education in Automatic Control. He serves as reviewer for several international journals including Automatica, Control Engineering Practice, IEE Proceedings - Control Theory and Applications, IEEE Transactions on Education, Journal of Process Control, Simulation, and for various international conferences.

Chris Cox was born in 1939 in Kingston-on-Thames, England. His subsequent life and education was mainly in the North East of England. He received a BSc(Hons) Degree in Electrical and Electronic Engineering in 1963 from the University of Durham and a MSc in Control Engineering from the University of Newcastle in 1996. He is presently Professor of Control Engineering at the University of Sunderland. He is a Chartered Engineer, Member of the IEE and Fellow of the Institute of Measurement of Control. His main research interests include tuning of traditional and predictive control, system identification and intelligent instrumentation with particular emphasis on chemical dosing philosophies at clean and dirty water treatment works. He also has an interest in the development of tools for the education of undergraduate students. He has acted as an editor for a number of special features and his reviewer experience has been very diverse including journals such as the IEE Proceedings-Control Theory and Applications, Simulation, Industrial and Engineering Chemistry Research and International Journal of Condition Monitoring and Diagnostic Engineering Management.

Antonio Ruano was born in 1959 in Espinho, Portugal. He received the First Degree in Electronic and Telecommunications Engineering from the University of Aveiro, Portugal, in 1982, the MSc in Electrothecnic Engineering from the University of Coimbra, Portugal, in 1989, and the PhD degree in Electronic Engineering from the University of Wales in 1992. In 1992 he joined the Department of Electronic Engineering and Computing of the Faculty of Sciences & Technology of the University of Algarve, where in 1996 he become Associate Professor of Automatic Control. His main research interests are neural control (and particularly its applications to PID autotuning), environmental control and parallel processing techniques applied to real-time control. He is the Coordinator of the Centre for Intelligent Systems and has over 100 research publications. He is Associate Editor for Automatica, he is a member of the Editorial Board of International Journal of Systems Science, and serves as reviewer for other international journals and conferences.