POLITECNICO MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

Problem-tailored model parametrization for the autotuning of event-based controllers

Tesi di Laurea Magistrale in
Automation and Control Engineering - Ingegneria dell'Automazione

Author: **Marco Zamuner**

Student ID: 975725
Advisor: Prof. Alberto Leva
Academic Year: 2022-23

Abstract

This thesis is about the (automatic) tuning of industrial modulating controllers – most typically, PIDs – also considering their event-based realization, i.e., implementations in which the control signal is not computed periodically but when some triggering mechanism requires a new value for it.

More precisely, the work refers to model-based tuning, i.e., on tuning techniques that first use the collected process input/output data to identify a process model and then exploit that model to tune the regulator.

In this context, two problems are addressed. The first one comes from the effect on the tuning quality of the procedure used to parametrize the model form data, the second one from the non parameter-free nature of event triggering mechanisms.

On the first problem, the thesis presents and discusses a technique to select the best combination of model parametrization procedure and controller tuning rule (in both cases within given sets) so that the resulting control be optimal with respect to a quality index of choice, here too within a given set.

On the second problem, a methodology is proposed to extend tuning rules conceived in the continuous time domain – hence naturally keen to periodic, fixed-rate implementations – to the event-based setting.

Simulation examples are reported to back up the statements made, and as a conclusion of the work, future research directions – most notably, a joint treatise of the two problems above – are outlined.

Keywords: Autotuning, Event-Based Control, Model Parametrization Procedure, Digital Controller, Tuning Quality Indices

Abstract in lingua italiana

Questa tesi tratta della taratura (automatica) di controllori industriali – tipicamente PID – considerando anche la loro realizzazione a eventi, cioè implementazioni in cui l'azione di controllo non è calcolata periodicamente ma solo quando un meccanismo di generazione degli eventi ne richiede un nuovo valore.

Più in dettaglio, questo lavoro si riferisce a metodi di taratura basati su modello, cioè tecniche di taratura che utilizzano i dati di input/output per identificare un modello del processo, che è usato per la successiva taratura del regolatore.

In questo contesto vengono affrontati due problemi. Il primo è legato a come la procedura utilizzata per parametrizzare il modello dai dati influenza la qualità della taratura; il secondo riguarda la taratura dei parametri di controllo che sono relativi al meccanismo di generazione degli eventi.

Per il primo problema questa tesi propone una tecnica per selezionare la miglior combinazione tra il metodo di parametrizzazione del processo e la regola di taratura del controllore (scelti in set predefiniti) in modo tale che il controllo risultante sia ottimo dal punto di vista di alcuni indici di qualità, anch'essi scelti in un insieme dato.

Per il secondo problema, si è utilizzata una metodologia per estendere al contesto a eventi delle regole di taratura originariamente sviluppate per essere utilizzate nel tempo continuo e di conseguenza per implementazioni a passo fisso o periodico.

Il lavoro presenta degli esempi di simulazione a supporto delle soluzioni proposte e si conclude con la descrizione di alcune direzioni per la ricerca futura, in particolare in merito a come integrare i due problemi sopra menzionati.

Parole chiave: Regole di Taratura, Controllo a Eventi, Metodi di Identificazione, Controllo Digitale, Indici di Qualità della Taratura

Contents

Abstract **i**

Abstract in lingua italiana **iii**

Contents **v**

1 Introduction **1**

2 Brief Literature Review **3**

3 Background **7**

3.1 Model-Based PI and PID Tuning . . . 7

3.1.1 PI controller . . . 8

3.1.2 PID controller . . . 9

3.2 Event-Based Realization . . . 11

3.3 Test Benchmark . . . 17

4 The Proposed MPP-TR Selection Technique **19**

4.1 Design . . . 19

4.1.1 Model Parametrization Procedure . . . 19

4.1.2 Tuning Quality Indices . . . 20

4.1.3 Offline Part of the Technique . . . 22

4.1.4 Online Part of the Technique . . . 24

4.2 Benchmark Assessment . . . 26

5 Going Digital and then Event-Based **37**

5.1 Event-Based Realization Technique . . . 37

5.2 Event-Based Realization Assessment . . . 38

6 Conclusions, Open Issues, Future Work **51**

Bibliography **53**

A Appendix A **59**

List of Figures **65**

List of Tables **67**

Acknowledgements **69**

1 | Introduction

This thesis is about autotuning industrial controllers, specifically of the model-based type, and offers two main contributions. First, it evidences the role of the technique used to parametrize the used process model, and makes that technique an integral part of the controller synthesis procedure together with the tuning rule of choice. Second, with reference to event-based (as opposite to fixed-rate) digital controller realizations, it addresses the problem of extending tuning rules conceived in the continuous time so as to also determine the controller parameters that refer to the event triggering mechanism.

When designing a controller, one would like to obtain the best performances with the least effort; there are many techniques that help with such a purpose, and autotuning is a widely used and well studied example. Autotuning gives through simple equations, based on mathematical models and experimental data interpolation, a starting point for the tuning parameters. These parameters can be improved if the performance does not satisfy the requirements. From an industrial point of view this represents a time saving operation and consequently an economic advantage.

In modern control systems – think for example of the so called “Industry 4.0” paradigm – the presence of potentially overloaded communication channels (for example, wireless) and in general the need for reducing the burden imposed to the used networks, collectively call for *event-based* controller realizations. In such implementations the control signals are not computed periodically but “when necessary”, thereby reducing transmissions and also undue (small) actuator movements. For battery-operated devices this approach allows to save energy so improve the life cycle.

It is quite evident that the *scenario* just sketched pose new challenges to the (automatic) tuning of industrial controllers. On one hand, it is nowadays possible to identify process models online even with low-end hardware, which increases the applicability and importance of model-based tuning but at the same time makes the procedure used to compute the process model parameters critical for the uniformity and the quality of the tuning results. On the other hand, in the event-based case, the controller and its triggering mechanism form a unit that must be synthesized in a unitary manner, as an incorrect operation of the said triggering would impair the tuning results.

In this work we consider the setting outlined above and evidence two problems, one general and one related to an event-based realization.

- Given the same data and tuning rule, changing the model parametrization procedure has a relevant impact on the results. Hence it is useful to be able to select the best tuning procedure, for which we mean a parametrization procedure-tuning rule couple, to fit the problem at hand; this in turn entails defining/choosing a selection indices.
- A tuning procedure should also be capable of providing the control law parameters that pertain to the event-based realization. As we shall point out, certain kinds of such realizations lend themselves particularly to the context we are building, and these are the ones where "event-related" parameters have the least influence on important properties like stability.

This thesis attempts to help solve the two problems above, and in the light of the considerations just reported, its contributions can be better qualified as follows.

- Based on a literature review and some consequent considerations, a set of indices is selected and motivated for tuning quality evaluation in the case of regulatory and servo tuning [43].
- Based on the said indices, a technique is proposed to exploit the gathered process I/O data to select the "best" tuning procedure for a given problem. The proposed technique is evaluated by applying it to a well assessed literature benchmark.
- Employing a particular event-based realization paradigm, extension are proposed to some tuning procedures in a view to also determining their event-related parameters. This extension was to date tested in structurally nominal conditions, benchmark assessment is underway.

2 | Brief Literature Review

The work by Ziegler and Nichols [53], dating back to 1942, is considered by the majority of the researchers the starting point of auto-tuning (PID) control. The authors formulated two methods, based one on an open-loop experiment (a step test) and the other on a closed-loop test (proportional feedback to generate a permanent oscillation condition).

The resulting tuning methods were in fact significantly empirical, and as such the research community saw room for improvement, and numerous authors started to develop their own tuning rules on a more or less rigorous basis. This research has led throughout the years to a large number of tuning rules [35] some of which are employed in different industrial products [27].

The auto-tuning research is still an open field, also because control techniques have evolved beyond PID, while at the same time model identification has progressed and learning-base methodologies have emerged; for example, (auto)tuning methods can now be integrated with neural networks [24] and applied to gain scheduling or model predictive control [1].

This thesis focuses on model-based auto-tuning, i.e., on tuning techniques that first use the collected process Input/Output (I/O) data to identify a process model (most typically, a transfer function) and then exploit that model to tune the regulator; the counterpart (not addressed herein) is data-based tuning, where the collected I/O information is used directly to tune the controller, without any model as intermediate step.

It is important to notice that tuning procedures most frequently run on low-end hardware and in any case the tuning time must be as short as possible to reduce the process upset. As a consequence tuning rules should be simple, and explicit ones (i.e., rules that compute the controller parameters as functions of model parameters and specification without solving any implicit system of equations) are strongly preferred.

As a direct consequence the used models should be simple, and their structure end up being dictated by that of the controller to tune — for example, in the case of a PID, a second order model is the choice of election.

Besides simplifying the *scenario*, the point above has a subtle but important effect. As the model structure is chosen *a priori* and not based on the collected data, a potentially significant process-model structural mismatch is inevitable, hence the procedure used to determine the model parameters from data has a relevant impact on the overall tuning procedure.

Curiously enough, in the literature this aspect is seldom considered, so that most papers that present a tuning rule do not even specify which model parametrization procedure was used to obtain the shown results. A relevant example to support the statement just made is the comprehensive work "Handbook of PI and PID Controller Tuning Rules" by A. O'Dwyer [35], that collects and analyses several hundreds of tuning rules for PI and PID controllers. Considering the PI case, it turns out that only 42% of the shown rules were presented by their authors together with some information on the Model Parametrization Procedure (hereafter MPP for short); the rest of the rules were discussed either with no consideration of the MPP, or simply taking the process model as known.

In the literature one can find many MPPs, and quite intuitively it was sometimes observed that the quality – and also the uniformity – of the tuning results highly depends on the parametrization accuracy. It is however difficult to quantify the said accuracy, so that most works of the type just mentioned ultimately resort to empirical best practices [30].

This MPP impact on tuning has been analysed in [25], where the closed-loop system in structurally nominal conditions is compared to the one obtained from various MPPs, making it clear that the parametrized system is worse than the structurally nominal one. In [42] the compound of Model Parametrization Procedure and Tuning Rule (hereafter the MPP-TR compound) is studied through some quality performance indices. Such indices are a useful tool that allows to compare tuning results [2] from various methods. Several indices can be defined depending on which aspect of the obtained control result is most relevant; the set can include for example the ISE (integral square error), the IAE (integral absolute error), the settling time, the overshoot, and so forth. Some such indices are more relevant in set-point tracking, others in disturbance rejection; some quantify the high-frequency control sensitivity and this the effect of measurement noise, others consider the position of the controller zeros with respect to the dominant process (model) poles, and the list could continue. The main point here is that previous research shows that the MPP influences the control quality and – most important – that the "optimal" MPP for a given quality index depends on the particular problem at hand.

Summing up on this first aspect, a way to select the best MPP-TR compound given the recorded I/O data and the quality index to optimise is highly desirable.

Carrying on, another research area that is growing in recent times is Event-Based (EB) control. This is testified by survey work like [3], where, through the analysis of 2299 research papers from 1999 to 2018, the authors report a large publication increment during this time span. This interest in EB control is also fuelled by the industry trend to make distributed control system nodes able to update and retrieve information to and from other nodes in a network.

In the literature there are many techniques to design an EB industrial controller, typically of PI(D) type [41, 49, 52], with applications e.g. to wireless control valves [7], unmanned aerial vehicle guidance [47], and more.

The EB approach can be implemented in both wired and wireless systems, and is gaining particular importance in the latter case owing to the need for rapid plant reconfiguration that is typical of modern manufacturing. Also and more in general, in a system where there are tight requirements on the information exchanged, the ability to reduce the traffic load is important to prevent delays and data losses. In these situations EB control can yield benefits because each node/device transmits only when an event occurs.

The advantages of the EB approach are numerous; for example in a distributed sensor network some sensors are installed where the power/data cable can not be placed due to difficulties in maintenance, hence they must work on battery. In battery-supplied nodes the most energy consuming part is the radio transmitter, which consumes some mW with respect to μW of the low power part hence keeping the radio always on is not an option. An EB realization of the node behaviour can turn on the radio only when needed, therefore the life-cycle of the device increases as a consequence.

Implementing a controller in an EB *scenario* is not as simple as designing the same controller in a classical (fixed-rate) one, as noted in several works such as that by Tiberi *et al.* [49]. In [13] Durand and Marchand compare different EB controller realization, observing that different set of control parameters for different implementations will give better performances. The point here is that any mechanism to generate events has parameters, and therefore any tuning procedure for an EB controller should provide these parameters together with those strictly relative to the control law. Some solutions have been proposed, like the one in [26] on which we build in this thesis.

Summing up on this second aspect, there is the need to extend tuning rules – and as a consequence MPP-TR compounds – to the EB case; strictly, one would like to select together with MPP-TR also the *way* to generate events and not only the event-related parameters once the way is chosen, but we leave this to future research.

To conclude, from the brief literature review above it is clear that two (intertwined) problems are open. One is how to choose the MPP-TR compound properly for the control quality index of choice. The other is how to extend a tuning rule to the EB context. This thesis addresses the two said problems individually, in a view to first help solving them and then provide the foundations for future research to comprehend them jointly.

3 | Background

In this chapter we are introducing and explaining the elements of the proposal.
In the first section we list all the tuning rules that are used in the following chapters.
The second section introduces a generic Event-Based (EB) realization before analysing a specific technique based on multitransmission.
In the last section, the Åström benchmark systems are presented.

3.1. Model-Based PI and PID Tuning

We focus on model-based tuning in order to have uniformity on the results and on the model parametrization procedure, we work only in normalized conditions and with normalized rules as in [21], where possible, to ease the comparison. The availability of the model allows us to forecast the tuning results and also to set up quality indices that employ the model parameters as in [22].

All the rules selected use the First Order Plus Dead Time process (FOPDT) to tune the parameters. We consider the process in nominal conditions.

$$P(s) = \mu \frac{e^{-sD}}{1 + sT} \tag{3.1}$$

We have unitary gain $\mu = 1$ and to make the results comparable we used the normalized time delay θ such that it will range from 0 to 1.

$$\theta = \frac{D}{D + T} \tag{3.2}$$

Some rules are parameter-free, some are not; for a fair comparison a uniform procedure was introduced to select the tuning parameter λ because in literature most of the times the parameter selection procedure is not specified. The parameter λ can be interpreted as the closed-loop time constant. However, this matter needs further investigation in the future.

$$\lambda = \frac{T + \frac{D}{5}}{k_a} \tag{3.3}$$

Where k_a is the acceleration factor.

We will use the two most common structures of controller, PI (Proportional-Integral controller) and PID (Proportional-Integral-Derivative controller). The PI is simply a PID without the derivative action. Each part of the controller has a different purpose. The proportional part determines the ratio of the output response to the error signal, it simply scales the error between the set-point and the output by a factor K_C. The integral part reduces the steady-state error but increases the overshoot. The integral response will continually increase over time unless the error is zero because it sums (integrates) the error term over time. The integral time T_i is a scaling factor for the integral operation. The derivative part determines how fast the error varies. The derivative response is proportional to the slope of the error variable, the derivative time parameter T_d will cause the control system to react more strongly to changes in the error and it will increase the speed of the overall control system response.

Now we present the different tuning rules selected.

3.1.1. PI controller

The PI tuning rule refers to the asymptotically stable FOPDT and the one-degree-of-freedom PI controller

$$C(s) = K_C\left(1 + \frac{1}{sT_i}\right) \tag{3.4}$$

The rules selected are:

- IMC-PI, formulæ Morari and Zafiriou [33], Braatz [8], Leva and Colombo [19]. Indicated here as 'IMC';
- SIMC rules by Skogestad [44, 45], indicated with 'Sko';
- IMC improved PI, 'Rivera PI' [39], here 'Riv';
- 'Direct Synthesis for disturbance' method by Chen and Seborg [10], termed here 'Dsd';
- Formulæ used in ABB Easy-Tune and reported in Li *et al.* [28], indicated with 'ABB';
- Rules based on the minimization of the IAE (Integral of Absolute Error) given by Lopez *et al.* [31], here 'LSM';
- Formulæ by Daniel and Cox [12], a specialisation of Vrančić *et al.* [50], indicated here as 'D&C';

- Method 31 or 32 for $0 < D/T < 2$, closed loop response overshoot $< 5\%$ from Mann *et al.* [32], termed with 'Mann';
- Method 1 from Hägglund and Åström [15], here 'H&A';
- Method 1 by Lee *et al.* [18], indicated with 'Lee';
- Method 1 for $T > D$ from Isaksson and Graebe [16], here 'I&G';
- Method 1 from Smith [46], termed here as 'Smith';

	K_C	T_i
IMC	$\frac{T}{\mu(D+\lambda)}$	T
Sko	$\frac{T}{\mu(D+\lambda)}$	$min(T, 4(D+\lambda))$
Riv	$\frac{T+D/2}{\mu\lambda}$	$T + D/2$
Dsd	$\frac{T^2+TD-(\lambda-T)^2}{\mu(D+\lambda)^2}$	$\frac{T^2-TD-(\lambda-T)^2}{D+\lambda}$
ABB	$\frac{1.164}{\mu}\left(\frac{D}{T}\right)^{0.977}$	$\frac{60}{40.44}T\left(\frac{D}{T}\right)^{0.68}$
LSM	$\frac{0.758}{\mu}\left(\frac{D}{T}\right)^{-0.861}$	$\frac{T}{1.02-0.323\frac{D}{T}}$
D&C	$\frac{1}{2\mu}\frac{T^3+T^2D+0.5TD^2+0.167D^3}{T^2D+TD^2+0.667D^3}$	$\frac{T^3+T^2D+0.5TD^2+0.167D^3}{T^2+TD+0.5D^2}$
Mann	$\frac{0.51T}{\mu D}$	T
H&A	$\frac{1}{\mu}\left(0.14+0.28\frac{T}{D}\right)$	$0.33D + \frac{6.8DT}{10D+T}$
Lee	$\frac{T+\frac{D^2}{2(\lambda+D)}}{\mu(D+\lambda)}$	$T + \frac{D}{2(\lambda+D)}$
I&G	$\frac{T+0.25D}{\mu\lambda}$	$T + 0.25D$
Smith	$\frac{T}{\mu D}$	T

Table 3.1: PI tuning rules

3.1.2. PID controller

The PID tuning rules are based on two control structures, with and without filtration of the derivative part. The difference between these two implementation is that the filtered one performs a filtering action of the noise that a sensor could add. It is more similar to how a real PID will be developed. The non-filtered controller is the academic PID structure.

The PID structure with filtered derivative part is:

$$C(s) = K_C\left(1 + \frac{1}{sT_i} + \frac{sT_d}{1 + sT_d/N}\right) \tag{3.5}$$

The rule that uses this controller structure is IMC-PID [20, 40], indicated as 'IMC':

$$\begin{aligned} T_i &= T + \frac{D^2}{2(\lambda + D)}, & K_C &= \frac{T_i}{\mu(\lambda + D)} \\ N &= T\frac{\lambda + D}{T_i\lambda} - 1, & T_d &= \frac{\lambda D N}{2(\lambda + D)} \end{aligned} \tag{3.6}$$

The academic PID takes the form:

$$C(s) = K_C \left(1 + \frac{1}{sT_i} + sT_d\right) \tag{3.7}$$

The tuning rules are:

- Method 2 approximated quarter decay ratio from Connell [11], here 'Connell';
- Method 1 from Moros [34] attributed to Oppelt, indicated here as 'Moros';
- Method 2 by Liptàk [29], termed as 'Liptàk';
- Formulæ from Sree *et al.* [37], here 'Sree';
- Method 1 for minimum IAE, $0.05 < D/T < 6$ from Wang *et al.* [51], indicated with 'Wang';
- Method 2 $D/T < 0.33$ from Fruehauf *et al.* [14], here 'Fruehauf1';
- Method 2 $D/T \geq 0.33$ from Fruehauf *et al.* [14], here 'Fruehauf2';
- Method 1, $\lambda > 0.T, \lambda \geq 0.8D$ by RIvera *et al.* [39], termed as 'Riv';
- Method 1 from Lee *et al.* [18], here 'Lee';

	K_C	T_i	T_d
Connell	$\frac{1.6T}{\mu D}$	$1.6667D$	$0.4D$
Moros	$\frac{1.2T}{\mu D}$	$2D$	$0.42D$
Liptàk	$\frac{0.85T}{\mu D}$	$1.6D$	$0.6D$
Sree	$\frac{1}{\mu}\left(\frac{T}{D}+0.5\right)$	$T+0.5D$	$\frac{0.5D(T+0.1667D)}{T+0.5D}$
Wang	$\frac{Ti\left(0.7645+\frac{0.6032}{D/T}\right)}{\mu(T+D)}$	$T+0.5D$	$\frac{0.5TD}{T+0.5D}$
Fruehauf1	$\frac{0.56T}{\mu D}$	$5D$	$0.5D$
Fruehauf2	$\frac{0.5T}{\mu D}$	T	$0.5D$
Riv	$\frac{Ti}{\mu(\lambda+0.5D)}$	$T+0.5D$	$\frac{TD}{2T+D}$
Lee	$\frac{Ti}{\mu(\lambda+D)}$	$T+\frac{D^2}{2(\lambda+D)}$	$\frac{D^2}{2(\lambda+D)}\left(1-\frac{D}{Ti}\right)$

Table 3.2: PID tuning rules

3.2. Event-Based Realization

An EB controller updates its control law only when an event is triggered, this means that something has changed in the system and the old control action does not meet the specifications.

In the realization of an EB controller there are some elements to figure out, first of all how to move from continuous-time to discrete-time then how to manage, generate and trigger an event.

For the first problem in literature [9, 17] there are different discretization methods that allow to express a continuous-time system into a discrete-time one e.g. Euler methods, Runge-Kutta methods, Tustin. These discretization methods can be divided in two groups implicit methods and explicit methods. Both have their pros and cons, but explicit methods tend to develop instability for large step size [6].

In an EB implementation an event can occur at any time, and this brings about some problems such as the well known Zeno behaviour. Other problems could be for example if two events occur at the same time which one has priority? Also when the controller is in the middle of a task and an event occurs should it continue with the previous task or abort it? This is a major problem in the implementation of an EB system and some solutions can be adopted, for example creating a priority hierarchy for the events.

In order to avoid such problems, and thanks to the inherently clocked nature of any industrial (digital) control system, we decided to use a periodic EB control. This approach can be treated as a fixed-rate control where some steps are skipped thanks to an event-triggering mechanism. The event-triggering mechanism generates the events. Given the fixed-rate nature of the controller and the event-triggering mechanism an event can occur only at a multiple of a quantum q that is a finite amount of time.

There are many ways to compute the quantum, we choose to constrain the phase margin reduction caused by the digital controller realization to be less than a prescribed $\Delta\varphi_m$.

$$q = \frac{\Delta\varphi_m}{k_c\omega_c} \tag{3.8}$$

Where ω_c is the nominal cutoff frequency, and k_c is a parameter that ranges from $1/2$ to $3/2$.

The event-triggering mechanism can be implemented in different ways e.g. integrate and fire or send on delta, but it mainly depends on the specific implementation and technologies adopted. The send on delta, SoD, policy triggers an event when in a multiple of q the difference in magnitude between the current measurement y and the past one y_{old} is greater than a certain threshold δ_y. A drawback of this policy is that it is sensible to noise, and spikes in the measurement will trigger an event. In integrate and fire the signal is integrated and when a certain threshold is passed an event is generated, it is less sensitive to noise but how the threshold is selected is less clear with respect to SoD.

Now listed the main elements of a generic EB realization we report the results taken from a particular technique based on multitransmission [26].

This technique considers the controller a switching system. The switching signal σ is a boolean variable that changes status when an event is triggered; σ distinguishes between a Running mode (R, $\sigma = 1$) and an Holding mode (H, $\sigma = 0$).

When the kth event is triggered the controller commutes from H-mode to R-mode and the event generator (e.g. a sensor) transmits the present value $y(k)$ and the last v_p past values $y(k-1), \dots, y(k-v_p)$. When an event is triggered the sensor will unconditionally transmit at step q for the further $v_f - 1$ instants then σ is set equal to 0.

In R-mode the new value of $u(k)$, control signal, is computed and actuated, hence an event causes v_f subsequent computations of u that is kept constant between its computations. In H-mode the last value of u is kept.

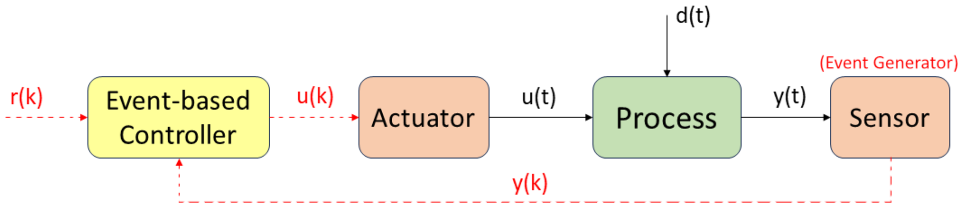


Figure 3.1: Event-based loop

The technique aims to compute the event-related parameters that do not affect critical parameters such as stability and to do so it has to ensure the stability of the closed-loop system composed by a Linear Time-Invariant (LTI) asymptotically stable strictly proper process P and an asymptotically stable switching controller C.

In order to apply this technique we have to discretize the process, the controller is already in the discrete-time domain. The discretization method should not affect significantly the results.

The process P has order n_P, input $u(k)$ and output $y(k)$, it can be described by the matrices A_P, b_P, c_P, d_P such that

$$\begin{aligned} A_P &= diag\{\lambda_{P_i}\}, & b_P' &= [\rho_{P_1} \cdots \rho_{P_{n_P}}] \\ c_P &= [1 \cdots 1], & d_P &= 0 \end{aligned} \tag{3.9}$$

Where $\{\lambda_{P_i}\}$ is the set of eigenvalues and $\{\rho_{P_{n_P}}\}$ is the set of the corresponding residues.

The switching linear controller has order n_C, with output $u(k)$, and input $e(k) = r(k) - y(k)$, $r(k)$ is the set-point. The controller is described in the I/O form as

$$\begin{aligned} u(k) =& u(k-1), & \sigma = 0 \\ u(k) =& \alpha_{C_1} u(k-1) \cdots + \alpha_{C_{n_C}} u(k-n_C) & \\ & + \beta_{C_0} e(k) \ldots \beta_{C_{m_C}} e(k-m_C), & \sigma = 1 \end{aligned} \tag{3.10}$$

Where $m_C < n_C$.

The following hypotheses are assumed true.

- A_P is Schur with real distinct eigenvalues;
- The closed-loop (LTI) system with C in R-mode is asymptotically stable.
- In the transition from H-mode to R-mode at a certain $k = k_{HR}$ C receives the values $y(k_{HR}), \ldots, y(k_{HR} - m_C)$.

- The H-mode can be maintained for an arbitrarily large but finite number of N_H of subsequent steps.

Under the hypotheses the following statements hold true.

- For the sequence of steps, even of infinite length, in H-mode the 2-norm of the closed-loop system state is limited.
- The ratio between the 2-norm of the closed-loop system state at the end and at the beginning of a sequence of steps in H-mode is overbound by

$$\sqrt{1+(n_C-1)\min_{i=1,\dots,n_P}\left(\frac{1-\lambda_{Pi}}{\rho_{Pi}}\right)^2} \tag{3.11}$$

- A_σ is the dynamic matrix of the switching closed-loop system, there is a finite minimum R-mode dwell time such that the origin of the state-space of the autonomous linear system

$$x(k) = A_\sigma x(k-1) \tag{3.12}$$

  is a globally asymptotically stable equilibrium.

The proof is available on the paper [26], we report the principal tools that will be used in chapter 5.

The closed-loop state state can be written as

$$x(k) = \begin{bmatrix} x_P(k) \\ x_C(k) \end{bmatrix} \tag{3.13}$$

Where x_P and x_C are the states of P and C.

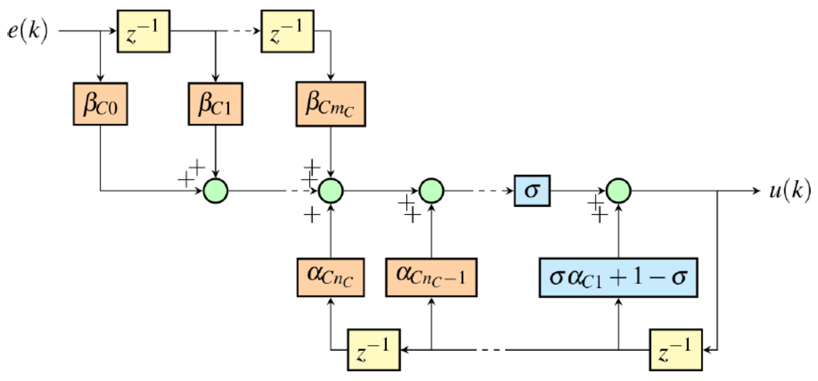


Figure 3.2: Switching representation of the controller

The controller C can be expressed in state-space form as

$$A_{C\sigma} = \begin{bmatrix} \sigma(\alpha_{C1}-1)+1 & \sigma\alpha_{C2} & \cdots & \sigma\alpha_{Cn_C-1} & \sigma\alpha_{Cn_C} \\ & & & & 0 \\ & & I_{n_C-1} & & \vdots \\ & & & & 0 \end{bmatrix}$$

$$b_{C\sigma} = \begin{bmatrix} \sigma \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$c'_{C\sigma} = \begin{bmatrix} \sigma(\beta_{C1}+\beta_{C0}\alpha_{C1}-1)+1 \\ \sigma(\beta_{C2}+\beta_{C0}\alpha_{C2}) \\ \vdots \\ \sigma(\beta_{Cn_C-1}+\beta_{C0}\alpha_{Cn_C-1}) \\ \sigma(\beta_{Cn_C}+\beta_{C0}\alpha_{Cn_C}) \end{bmatrix}, \quad d_{C\sigma} = \sigma\beta_{C0} \tag{3.14}$$

With $\sigma = 0$ for H-mode and $\sigma = 1$ R-mode, if $m_C < n_C$ then β_C coefficients are zero. Note that (3.14) is a minimum realization of C and it does not represent the actual operation of the controller as per (3.10), in H-mode u is held and no values of y are received. When transitioning from H to R the transmission of the m_C past values of y properly set the state x_C as C has evolved.

The closed-loop switching dynamic matrix is

$$A_\sigma = \begin{bmatrix} A_P - b_P d_{C\sigma} c_P & b_P c_{C\sigma} \\ -b_{C\sigma} c_P & A_{C\sigma} \end{bmatrix} \tag{3.15}$$

and takes as base for the state space the one corresponding to x_P and x_C as per (3.9) and (3.14).

In case of H-mode A_σ becomes A_H (3.16). The upper left triangular submatrix $A_{H_{11}}$ has dimension $n_p + 1$ and it is diagonalizable because its eigenvalues are the unity and those of A_P, that are distinct eigenvalues for hypothesis.

$$A_H = \begin{bmatrix} \lambda_{P1} & 0 & \cdots & 0 & \rho_{P1} & | & 0 & \cdots & 0 \\ 0 & \lambda_{P2} & & 0 & \rho_{P2} & | & 0 & \cdots & 0 \\ \vdots & & \ddots & \vdots & \vdots & | & \vdots & & \vdots \\ 0 & \cdots & 0 & \lambda_{Pn_P} & \rho_{Pn_P} & | & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & 1 & | & 0 & \cdots & 0 \\ - & - & - & - & - & - & - & - & - \\ 0 & \cdots & \cdots & 0 & 1 & | & 0 & \cdots & 0 \\ \vdots & & & \vdots & \vdots & | & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 0 & | & \cdots & 1 & 0 \end{bmatrix} \tag{3.16}$$

The diagonalizing matrix of $A_{H_{11}}$ is $T_{H_{11}}$ and T_H is built as

$$T_H = \begin{bmatrix} T_{H11} & O_{n_P+1\times\{n_C-1\}} \\ O_{n_c-1\times\{n_P+1\}} & I_{n_C-1} \end{bmatrix} \tag{3.17}$$

T_H is used as a change of base to compute the R/H-mode dynamic matrix $\tilde{A_\sigma} = T_H^{-1} A_\sigma T_H$ for the closed-loop system with state $\xi = T_H^{-1} x$. From now on $\tilde{A_\sigma}$ is the closed-loop R/H-mode dynamic matrix "in the ξ base".

The maximum multiplicative expansion that the 2-norm of the closed-loop state in the ξ base can undergo along a sequence of H-mode steps, even infinite, is

$$E_{H,\xi} = \sqrt{1 + (n_C - 1) \min_{i=1,\dots,n_P} \left(\frac{1-\lambda_{Pi}}{\rho_{Pi}}\right)^2} \tag{3.18}$$

which is the sought overbound that does not change since the ordering of eigenvalues and residues in (3.9) is arbitrarily.

Given, by hypothesis, the closed-loop stability of the system in R-mode, A_R, then the 2-norm of the iterated matrix will contract for k large enough, $\|A_R^k\|_2 \to 0$ for $k \to \infty$. Consequently, there is a quantity Δ_R such that

$$E_{H,\xi} \|\tilde{A_R}^k\|_2 < 1 \quad \forall k \geq \Delta_R \tag{3.19}$$

In conclusion there will be an H-mode sequence of arbitrary length that will expand the state of the system at a rate of $E_{H,\xi}$ followed by an R-mode contraction one of length at least equal to Δ_R.

Δ_R is an event-based parameter and it will be used to determine the minimum time of transmission from the last event received. It is the lower bound that guarantees not to loose important information about control, if an event is triggered during the Δ_R-time of transmission the transmission time will reset.

We can add a mechanism that will generate an event even if the sensor does not measure any difference on the output. A time-out mechanism is suitable for this task, if an event is not triggered and the timeout is overtaken then a time-out event is created to update the system. This can be thought as a safety measure to keep the system always alive to avoid malfunctions.

3.3. Test Benchmark

We decided to use the first five benchmark classes of [4] because they are well representative of most scenarios that can be found on a real implementation and are standard systems that are well suited for parametric studies.

$$P_1(s) = \frac{1}{(s+1)^\alpha} \qquad \alpha = 2, 3, 4, 5, 6, 7, 8; \tag{3.20}$$

$$P_2(s) = \frac{1}{(s+1)(1+\alpha s)(1+\alpha^2 s)(1+\alpha^3 s)} \qquad \alpha = 0.05 : 0.05 : 0.95; \tag{3.21}$$

$$P_3(s) = \frac{1-\alpha s}{(s+1)^3} \qquad \alpha = 0.1 : 0.1 : 5; \tag{3.22}$$

$$P_4(s) = \frac{1}{1+s\alpha} e^{-s} \qquad \alpha = 0.1 : 0.1 : 10; \tag{3.23}$$

$$P_5(s) = \frac{1}{(1+s\alpha)^2} e^{-s} \qquad \alpha = 0.1 : 0.1 : 10; \tag{3.24}$$

The first class (3.20) is widely used as test cases by controller manufacturers, for large values of α the system behaves like a ones with long delays. P_2 processes have four poles spaced through the parameter α, we decided not to consider the case $\alpha = 1$ because it is represented by the previous class. These systems (3.22) have an unstable zero and three equal poles, the performances deteriorate as α increases. The fourth class is the FOPDT, it is used in all autotuning rules (subsection 3.1.1 for PI and 3.1.2 for PID). For large α it represents lag dominated systems. The last class (3.24) is similar to P_4 but with a higher frequency roll off. For all classes we decided to not consider the degenerate cases, the pure time delay ($\alpha = 0$ in (3.23), (3.24)) or the single pole process ($\alpha = 1$ in (3.20)).

4 | The Proposed MPP-TR Selection Technique

This chapter is about taking process I/O data and selecting the best procedure (parametrization procedure-tuning rule compound) given the servo or regulatory problem, acceleration factor where applicable. The involved indices are computed analytically wherever possible, and in any case the entire treatise is carried out in the continuous time, hence in this part of the overall proposal there is no role for the event-based realization, not even for the digital nature of the real controller.

4.1. Design

The model-based tuning rules use the process in the FOPDT form to determine the tuning parameters, but the benchmark processes that we will use to test the tuning rules have different structures. In order to compute the equivalent FOPDT model from one of the benchmark classes we need a model parametrization procedure.

We select and present some tuning quality indices that are suitable for industrial use and allow to compare directly different tuning rules.

We also present one technique composed by an offline part, used to evaluate the quality indices and compute the tables and an online part that can be used directly on the field to quickly determine which rules is the best. The online part of the technique cannot be used standalone because we need the tables computed in the offline part, hence one run of this part of the technique is necessary.

4.1.1. Model Parametrization Procedure

In literature there are many model parametrization procedures that allow to express an unknown process in the FOPDT form. Most of them require a step response of the process others use a frequency analysis in order to return the parameters necessary to tune the controller.

The two parametrization procedures selected are the Method of Areas [36], and the Sundaresan and Krishnaswamy method [48], for brevity percentage procedure, both are step response procedures.

The method of areas (M_1) expresses the process step response $y_{us}(t)$ and symbolically computes [23] the two integrals:

$$A_0 = \int_0^{\infty} (y_{us}(\infty) - y_{us}(t))dt, \qquad A_1 = \int_0^{A_0/y_{us}(\infty)} y_{us}(t)dt \tag{4.1}$$

where $y_{us}(\infty) = lim_{t\to\infty} y_{us}(t)$, then set

$$\mu = y(\infty), \qquad T = e\frac{A_1}{\mu}, \qquad D = \frac{A_0}{\mu} - T \tag{4.2}$$

The percentage procedure (M_2) selects two instants t_1 and t_2 that correspond to the 35.3% and 85.3% of the total process step amplitude, respectively. Then the parameters are:

$$\mu = y(\infty), \qquad T = \frac{2}{3}(t_2 - t_1), \qquad D = 1.3t_1 - 0.29t_2 \tag{4.3}$$

We selected these two procedures because the parameter computation is very different, this affects the normalized time delay and, as a consequence, the indices evaluation.

4.1.2. Tuning Quality Indices

If we want to determine the optimal MPP-TR compound rule we have to use some tuning quality indices in order to compare the results.

The two most common control scenarios are servo control or set-point tracking and regulatory control or disturbance rejection. In literature, it is known that the problems of set-point tracking and disturbance rejection require different control tuning.

In servo problems the controller must be able to follow a variation of the set-point reference $y_0(s)$, figure 4.1. The controller designed for regulatory problems has the purpose to attenuate the disturbances that come from different sources, represented in the figure 4.1 as an external input $d(s)$.

In case of disturbance rejection the controller has to be robust but the set-point response would be too nervous with a big overshoot or many oscillations that cause stress on an actuator. In servo control the tuning can be done through cancellation of the dominant dynamic causing a weak disturbance rejection which means introducing an offset in the tracking that can be difficult to recover.

Given the block diagram, figure 4.1, where $C(s)$ is the controller, $P(s)$ is the process, servo control takes the complementary sensitivity function $T(s)$ while disturbance rejection takes the control sensitivity function $K(s)$.

$$T(s) = \frac{y(s)}{y_0(s)} = \frac{P(s)C(s)}{1 + P(s)C(s)} \tag{4.4}$$

$$K(s) = \frac{y(s)}{d(s)} = \frac{P(s)}{1 + P(s)C(s)} \tag{4.5}$$

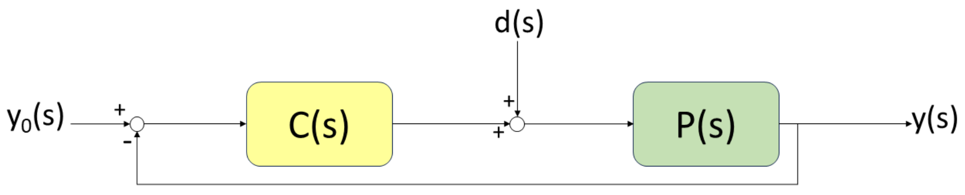


Figure 4.1: Block diagram of the considered closed-loop system

In this thesis we selected the following indices for set-point tracking:

- ISE, integral square error, indicated here as ISE_{sp};
- Maximum overshoot, here Ovr;
- 99% settling time.

For disturbance rejection we have:

- ISE, integral square error, indicated here as ISE_{dr};
- Maximum absolute error, here $|e|$.

The selected indices are well representative of the respective problems, they are also suitable for industrial use as criterion for selecting the best controller for an application.

The settling time is the time required for the response to reach and stay within a range of 2% of the final value. It tells how long a system takes to stabilize to the new reference after a set-point change.

The maximum overshoot, Ovr, is the maximum peak value of the response curve measured from the final steady-state value. It is used to understand how strong the control action is, bigger values of overshoot are produced by stronger control actions.

$$Ovr = \frac{y_{max} - y_{\infty}}{y_{\infty}} \tag{4.6}$$

The maximum absolute error will say what is the maximum value taken with respect to the steady-state, y_{ss} when a disturbance is introduced. We take the absolute value because we do not know if the disturb will cause a larger positive or negative error. A controller that minimizes the error will be more robust to disturbances.

$$|err| = |y_{max/min} - y_{ss}| \tag{4.7}$$

ISE integrates the square of the error, hence penalises large errors more than smaller ones. A system designed to minimize this index usually shows a fast decrease in a large initial error, the result is a system with a fast oscillatory response and a poor relative stability [38]. The minimization of this index involves the minimization of power consumption for some systems.

$$ISE = \int_0^{\infty} e^2(t)dt \tag{4.8}$$

Where $e(t)$ is the error.

4.1.3. Offline Part of the Technique

We now describe the offline part of the proposed technique, that is, how the selection tables for online use are created; the operations are summarized below.

1. Select a benchmark class;
2. Make the class parameter change in a predefined range;
3. Compute the equivalent FOPDT model through an MPP;
4. Compute for each model the normalized estimated delay, θ_{si} as (3.2);
5. Tune the controller for each model and tuning rule;
6. Compute the indices;
7. For each MPP create a table of the best tuning rule per index.

In the figure 4.2 is reported the flowchart of the offline part of the evaluation technique and how the results can be represented and stored in the rule $nI[A, C]$.

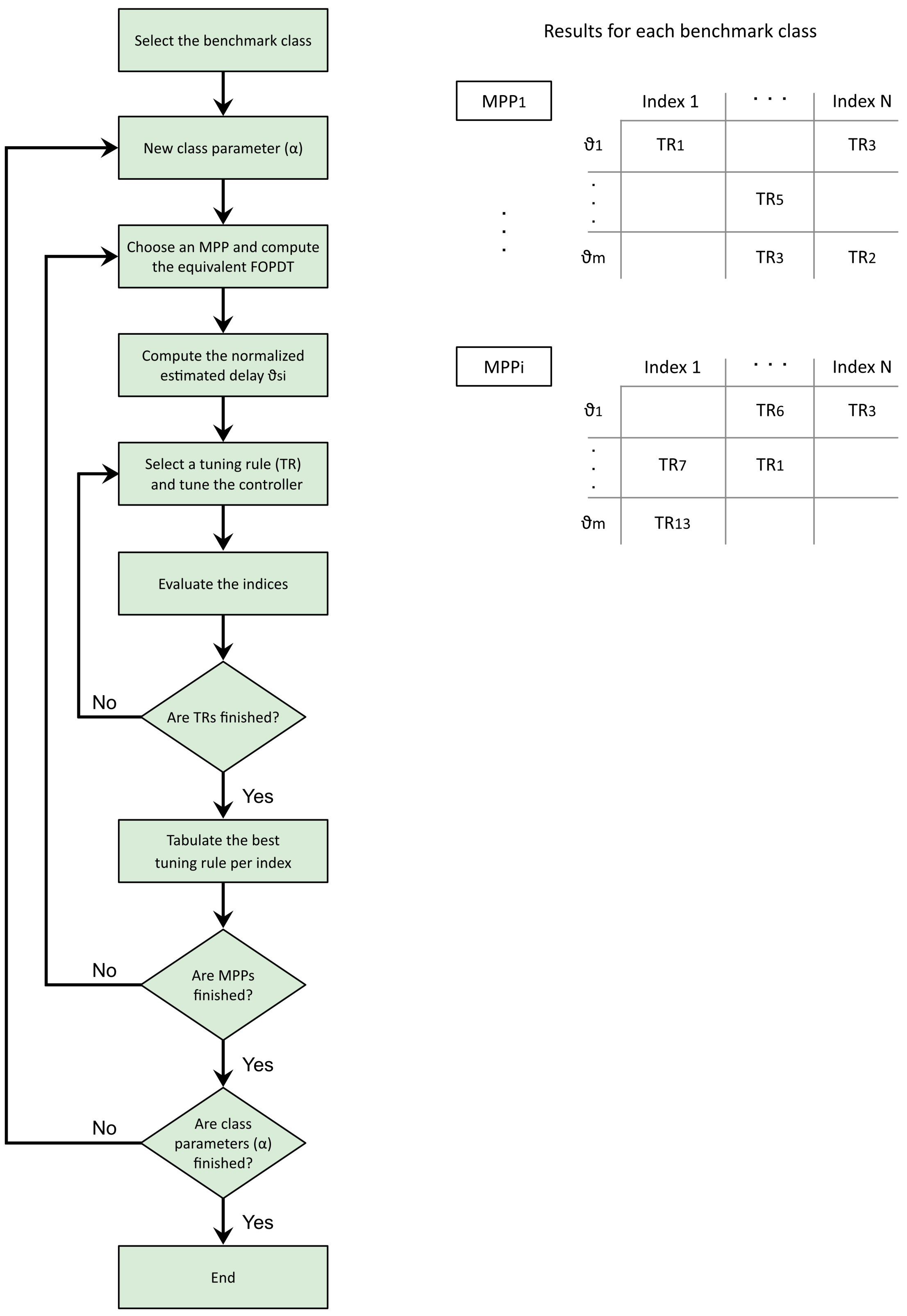

Select the benchmark class
New class parameter (α)
Choose an MPP and compute the equivalent FOPDT
Compute the normalized estimated delay ϑsi
Select a tuning rule (TR) and tune the controller
Evaluate the indices
Are TRs finished?
No
Yes
Tabulate the best tuning rule per index
Are MPPs finished?
No
Yes
Are class parameters (α) finished?
No
Yes
End
Results for each benchmark class
MPP1
Index 1
Index N
ϑ1
TR1
TR3
TR5
ϑm
TR3
TR2
MPPi
Index 1
Index N
ϑ1
TR6
TR3
TR7
TR1
ϑm
TR13

Figure 4.2: Flowchart of the offline part of the evaluation technique

This is an offline technique because it uses the benchmarks to compute the tables.

Each table of the model parametrization procedure-tuning rule (MPP-TR) is codified, through interpolation or a threshold method, in the rule $nI[A,C]$ where n is the benchmark class, I is the index selected and $[A,C]$ gives the aggressive (A) or conservative (C) approach.

All the tuning rules that are too unstable or too low damped for a certain θ_{si} are discarded from the pool of results.

Given the same step response the two procedures return different values of θ_{si} hence the results are not directly comparable. A solution to this problem is to always compute the two θ_{si} then choose the one that better exploits the application. The greater normalized delay matches the most conservative implementation while the smaller one matches the most aggressive one as noted in [5], this implies that in an aggressive implementation a conservative rule is the best choice for having the response not too nervous reducing the risk of saturation.

4.1.4. Online Part of the Technique

This section describes how the information created in the offline part are used online to actually select the MPP-TR compound for a given problem.

1. Perform the process response;
2. Compute the equivalent FOPDT and θ with all MPPs hence there will be n_{MPP} models and n_{MPP} θ;
3. For each MPP:
   (a) Identify in which benchmark class the process can be approximated and select which table to use; at the present state of the work this needs to be done manually, but research is underway and ideas are already available about how to automate the selection;
   (b) Decide the θ approach (aggressive, conservative...);
   (c) Select the tuning rule from the table and save the corresponding index value;
4. Select the best index and get the optimal MPP-TR compound.

Here another problem arises, how to understand which benchmark class does a process belong to. We can look for similarities between the real process data and the model for example we know that if the process step response presents an initial undershoot it will belong to the third benchmark class, if it has a significant dead time then it will be class four or five, in other cases it will be class one or two. Another solution is to look at the difference between the real process data and the model, if the difference is below a certain threshold then it is more likely to have identified the class. This problem will not be treated in this thesis, we assume that we know which class the unknown process belongs to.

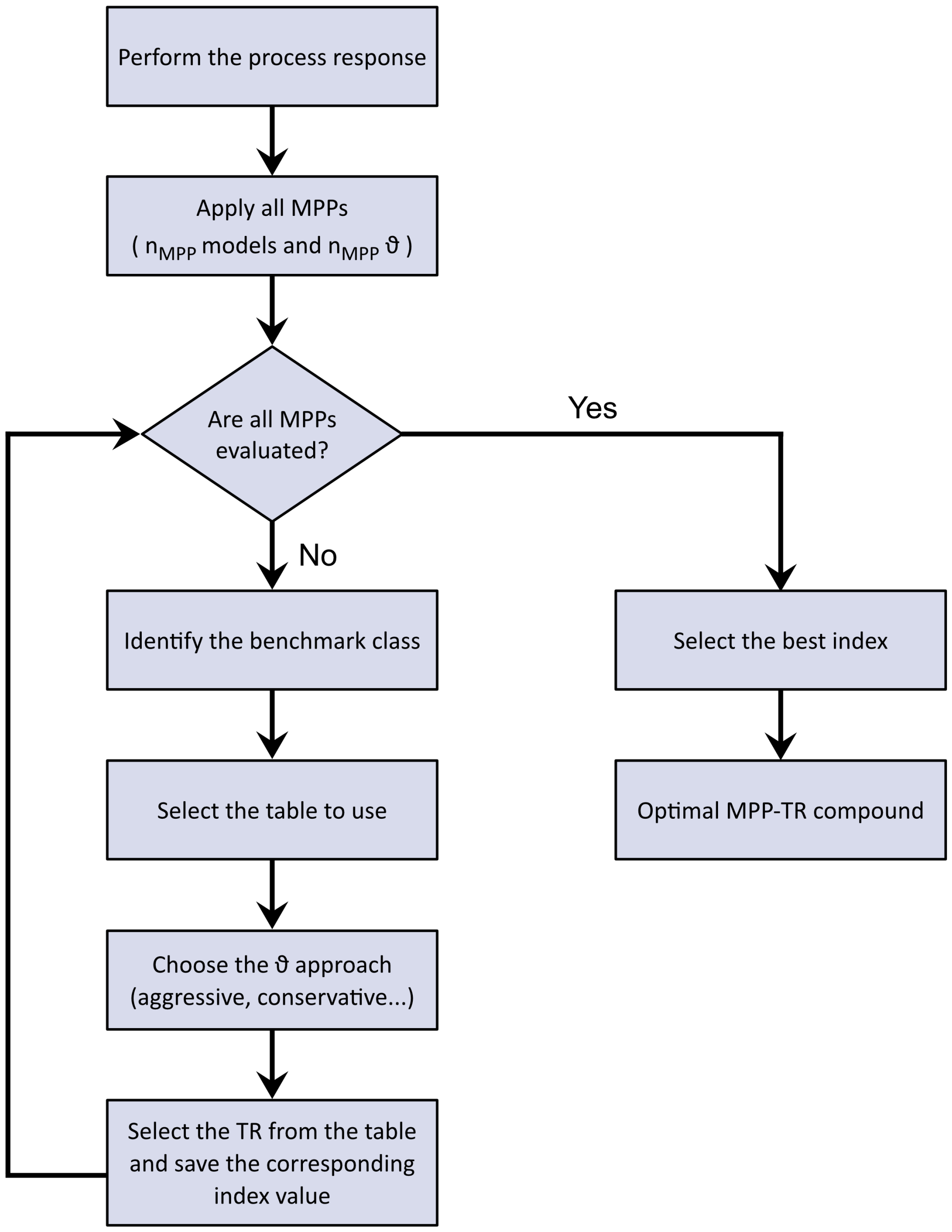

Perform the process response
Apply all MPPs
(n_{MPP} models and n_{MPP} ϑ)
Are all MPPs evaluated?
Yes
No
Identify the benchmark class
Select the table to use
Choose the ϑ approach (aggressive, conservative...)
Select the TR from the table and save the corresponding index value
Select the best index
Optimal MPP-TR compound

Figure 4.3: Flowchart of the online part of the evaluation technique

4.2. Benchmark Assessment

In this section the results obtained from the evaluation of the offline technique are proposed and analysed. Firstly we analyse the two parametrization procedures looking at the estimated normalized delay then we will propose the results of the technique for each benchmark class.

Analysing the two parametrization procedures from the point of view of θ_s we can say that the areas method is usually more aggressive than the percentage method $\theta_{s1} < \theta_{s2}$.

We plotted on the figures 4.4 the comparison of the θ_s of the two procedures. On the vertical axis there are the values of θ_s and on the horizontal axis the values of the class parameter α.
The only benchmark class that has a different behaviour is the third one, for $\alpha > 0.8$ the areas method is more conservative.

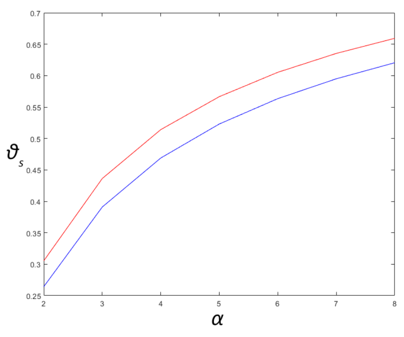
In figure 4.4c the two θ_s ranges are very different, θ_{s2} goes from 0.45 to 0.74 while θ_{s1} from 0.41 to 0.97 this means that M_2 has some problems with this benchmark class. The step response of P_3 (3.22) presents an initial undershoot due to the unstable zero, this undershoot is considered and computed in the areas method because it is an analytical procedure but percentage procedure considers only the positive part of the dynamics, the undershoot is treated as a pure dead time.

We can see that the two procedures have basically the same performance for the fourth class. This is reasonable because the goal of these two procedures is to compute the equivalent FOPDT parameters.

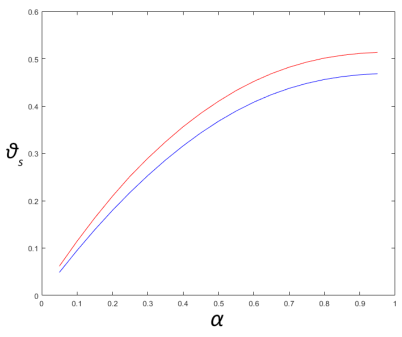
We can evaluate the quality of each procedure by computing the deviation between the estimated parameter and the benchmark class parameters that are $D = 1$ and $T = \alpha$. The deviation of M_1 for both parameters can be considered zero, the maximum absolute deviation is 0.01% for D and 0.001% for T. M_2 has a bigger absolute deviation, D ranges from 1.2% to 11.1% and T deviates in the range of 1.1% to 1.3%. For both procedures we can state that the estimation of the delay is the most critical part because a pure dead time can only be approximated and each approximation method introduces some errors. M_2 clearly struggles under this aspect with respect to M_1. In conclusion M_2 is worse than M_1 in terms of approximation but it is easier to implement because it does not need to compute any integration.

Looking at the other three classes we can see that the difference between M_1 and M_2 is almost constant in the first class, it slightly increases for higher α values. In the figure 4.4b the difference increases as θ_s increases, instead in class five the difference increases as θ_s decreases.

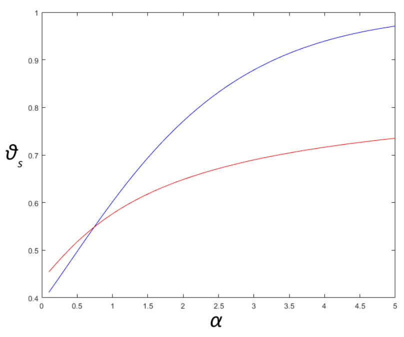
In the figures 4.4 there is a clear trend, θ_s grows as α increases for the first three benchmark classes while for the last two θ_s decreases as α grows. The only difference between these classes is the presence of a pure time delay otherwise classes four and five, for $T = 1$, would be equal to P_1, this behaviour can be further analysed.



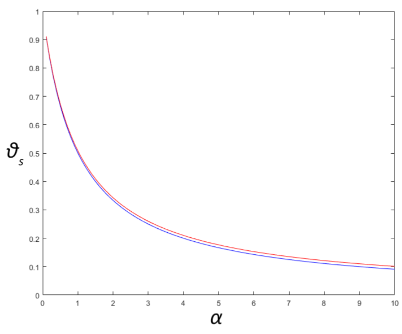
(a) class 1



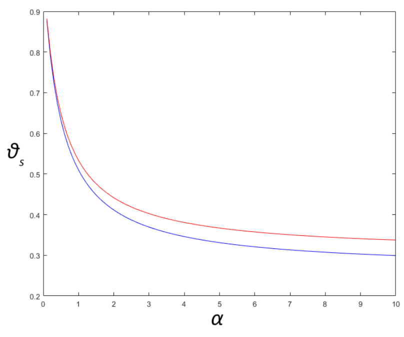
(b) class 2



(c) class 3



(d) class 4



(e) class 5

Figure 4.4: θ_s comparison with respect to benchmark class parameter, the colors are blue for M_1 and red for M_2

As said before the tuning rules that are too low damped or unstable are discarded from the pool of suitable results, we set as minimum acceptable phase margin to be $\varphi_m = 20°$. The indices comparison follows the rule that the smallest value is the best.

Given the problem noted in section 4.1.3 concerning θ_s we can not have a single table where we put together the two parametrization procedure, hence for each index there is one table for M_1 and one table for M_2.

The following tables coupled with the θ_s analysis form the rule $nI[A, C]$ to use in the online technique.

To improve the readability the tables are represented in figures where the best tuning rule is highlighted in red, the others are grayed. Each plot reports the value of parameter θ_s on the horizontal axis while the tuning rules are on the vertical axis. We reported both PI and PID rules in each table to save space, a bold line marks the separation between the two controllers.

The twelve PI tuning rules (hereinafter TR1 to TR12) are - in this order - IMC, Sko, Riv, Dsd, ABB, LSM, D&C, Mann, H&A, Lee, I&G and Smith. The nine PID tuning rules (TR13 to TR21) are - again, in this order - IMC, Connell, Moros, Liptàk, Sree, Wang, Frehauf, Riv and Lee.

All the performance indices are computed in response to a unit set-point step and to a unit load disturbance step. To have the results as uniform as possible all the tuning rules are simulated for 500 seconds, we chose this simulation time because the controllers that have a bigger settling time have worse performance.

We observe that all the controllers appear at least one time in the tables; an exception is Sko that has the same performance as IMC-PI.

These two rules have the same performances because the controller is tuned in the same way. The proportional part K_C is obtained through the same equation, the integral time T_i is computed differently. Sko computes T_i taking the minimum between T and $\frac{20k_aD+20T+4D}{5k_a}$ that is always bigger for our choice of λ (3.3) and the benchmark parameters, hence never taken in consideration.

In the figures there is not one controller that is the best for all benchmark processes. We have some tuning rules that overcome others for some indices e.g. maximum $|e|$ for P_2.

The figures of the maximum overshoot index have the regulator that changes frequently. This is caused by the computational power and the numerical approximation of the computer since some values are in the order of magnitude of 10^{-16}. This hardly happens with other indices because the values computed are bigger.

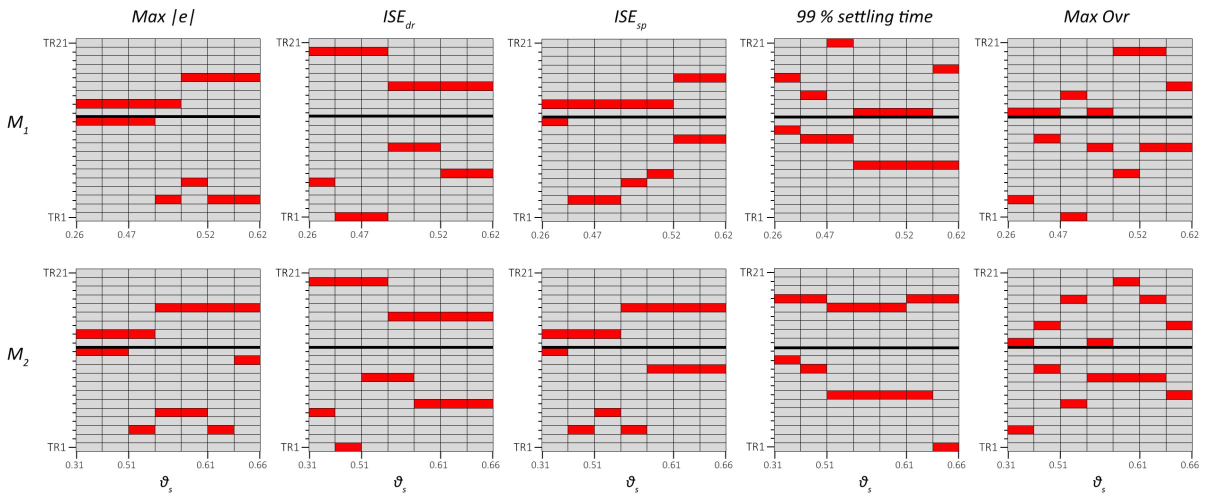


Figure 4.5: Best tuning rule per index, benchmark class 1. Red best tuning rule, Gray others

In the first benchmark class the indices of disturbance rejection ($Max|e|$ and ISE_{dr}) show different controllers for the two indices, ABB is the only rule that is present for both indices. This means that some tuning rules exploit better some indices than others, this observation is also true for all other classes and also for the indices of set-point tracking.

Comparing the two parametrization procedures the tuning rules are the same in the same range, only minimum $|e|$ in M_2 has an additional rule for the last value of $\theta_s = 0.66$ that is I&G.

For the set-point tracking scenario there is more variety of the rules. ISE_{sp} is similar to disturbance rejection indices, we have the same tuning rules in both methods but in different θ_s ranges.

In the settling time figures we can see that the rules for the two procedures are different in the PID case, for M_2 there are two controllers Sree and Wang but for M_1 there are also Lee, IMC and Moros. For the PI there are the same rules and M_2 has an additional rule for the θ_s value outside the M_1 range. This is an anomaly, since we would expect to find the same rules in a similar range as it is for the other indices, this means that settling time index is more susceptible to how the tuning parameters are computed.

The maximum overshoot shows this chess-pattern, it will be more evident for the other processes.

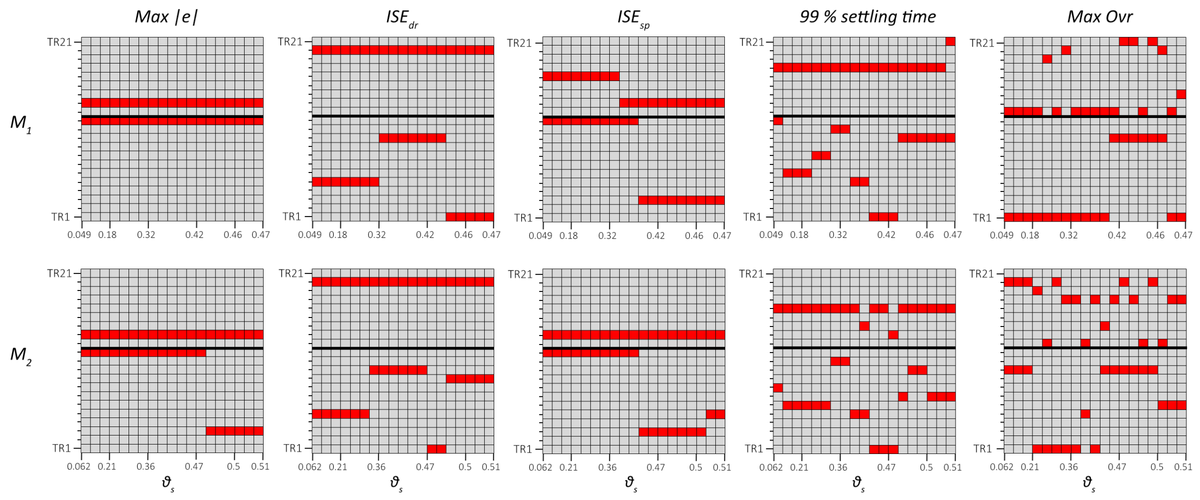


Figure 4.6: Best tuning rule per index, benchmark class 2. Red best tuning rule, Gray others

P_2 indices are more stable with respect to the previous ones. In most of them we find fewer rules, or even one rule.

We see that PID rules are more consistent, in $|e|$, ISE_{dr} and partially in ISE_{sp} one rule turns out to be optimal for both procedures. We have Connell in $|e|$, Riv in ISE_{dr}. For ISE_{sp} Connell in M_2 and Sree and Connell for M_1. On the PI side of this indices M_2 changes tuning rules more, this can be caused by the greater uncertainty in parameter estimation or simply one rule works better than another for the computed θ_s.

For the 99% settling time we observe that there is a chess-pattern for the PI controllers, we have almost the same tuning rules but for different θ_s. The PID on the other hand has different tuning rules, for M_1 we have only two rules Lee and Wang while for M_2 we have Sree, Connell and Moros.

The maximum overshoot for M_1 shows that the best rules, for the PI, are IMC and Lee. Since the rules are two we can say that this result is consistent with respect to the others where the rule changes often like for the second procedure or the M_1's PID.

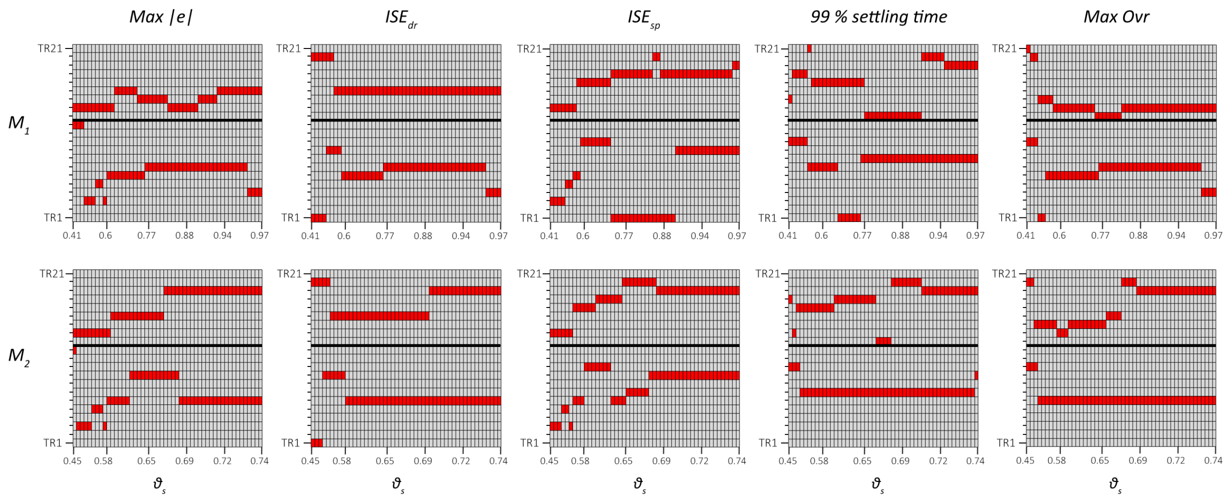


Figure 4.7: Best tuning rule per index, benchmark class 3. Red best tuning rule, Gray others

The third class is the most challenging for the controller due to the unstable zero; we have just analysed that the two procedures do not parametrize in the same way the process hence the indices results will be different. We can see that the unstable zero is challenging because the rule varies frequently.

Dsd appears only in this class. If we perform a step response of the closed-loop system in the set-point tracking framework we will see that it is a very conservative rule because the settling time is high therefore it can counteract the instability. This rule turns out to be unstable for most of the benchmark parameters but for the last values, $\alpha > 4.6$, it is stable with a huge settling time. Dsd is particularly suitable for disturbance rejection, in both indices it is the best in the last θ_s and in maximum overshoot has zero overshoot due to the long settling time. This tuning rule only show up in areas method because it parametrizes better the class achieving bigger values of θ_s.

As anticipated the results in M_1 figures are different from M_2 ones. We have different controllers, only few of them are the same; for example in ISE_{dr} Liptàk PID performs better in most of the results for both procedures.

The settling time index has a chess-pattern mostly for PID controllers; while for PI Mann seems to exploit better this index but this rule does not appear for the other indices.

The maximum overshoot is more regular for this class with respect to the others because fewer tuning rules are able to manage the instability of the zero.

We can see that in all M_2 indices in the second half of the samples one rule overcomes the others; LSM for $|e|$, ISE_{dr} and maximum overshoot, $D\&C$ for settling time and $H\&A$ for ISE_{sp} this is because the values of θ_s are very close to each others.

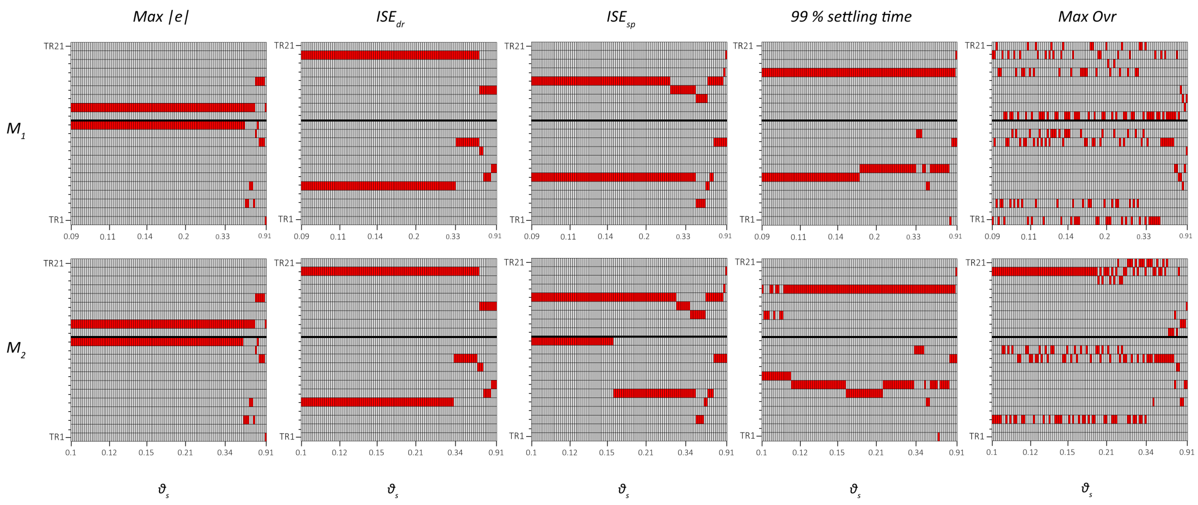


Figure 4.8: Best tuning rule per index, benchmark class 4. Red best tuning rule, Gray others

For this benchmark class we can see that the two procedures return the same results for $|e|$, ISE_{dr} and ISE_{sp} with small variations caused by the higher uncertainty of M_2.

The maximum overshoot instead has a completely different behaviour, as we see a large fragmentation. In M_1 figure few controllers achieve the best results but for fraction of θ_s, over 100 samples I&G has the longest sequence of 7 consecutive samples. M_2 on the other hand shows that Riv-PID overcomes the others for the first half of the samples then there is more variety, on the PI side we have a situation similar to M_1 where here I&G covers at most 10 consecutive samples.

Settling time is another index where the two parametrization procedures have different controllers, here the percentage procedure changes rule more frequently and more tuning rule appears. In PI controllers M_2 has Mann as additional controller in the first samples, for PID controllers there is Moros that does not appear in M_1. We can also see that the two PI controllers that have the majority of the samples are LSM and $D\&C$ but with switched positions for the two procedures. In PID we have mostly one rule that is Wang.

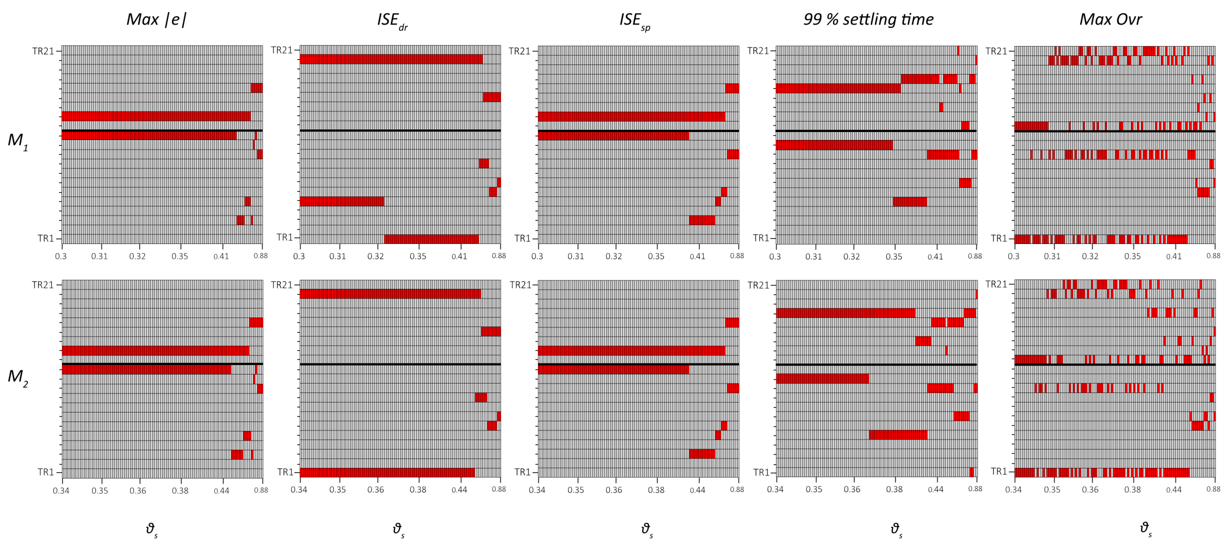


Figure 4.9: Best tuning rule per index, benchmark class 5. Red best tuning rule, Gray others

In this benchmark class we can make similar observations to P_4. For this class we have IMC-PI that exploits better ISE_{dr} for both procedures but for M_1 in the first part ABB has better performance. On the PID side we have the same controllers in the same range of the one of the previous benchmark class.

The maximum error has Smith for PI and Connell for PID that overcomes the others for most of θ_s , there is not a significant difference between the two MPPs.

Smith and Connell have the same behaviour also for ISE_{sp}, the two indices have very similar results in terms of tuning rule selected.

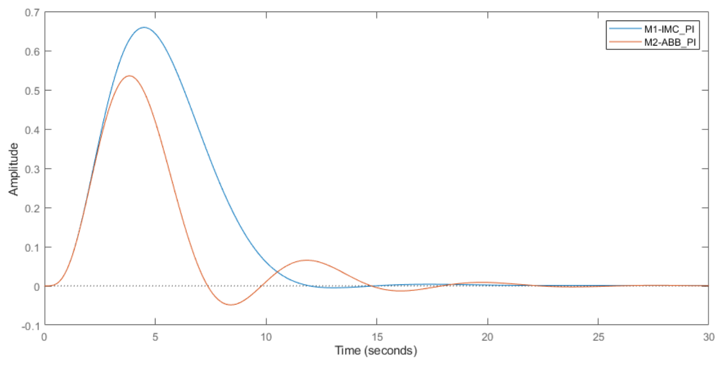
We have differences in settling time for the PID where for M_1 Sree exploits better the index while in M_2 we have Wang, this is similar to what observed for the same index for P_1 process. On the PI side I&G is the best in the first part of the results.

In overshoot we see that IMC-PI and PID develop the best results, another rule that works great is Lee for both control structures and Riv-PID. As observed for this index in the past classes there is a lot of fragmentation, we have the same rules that are the best for few instants then changes.

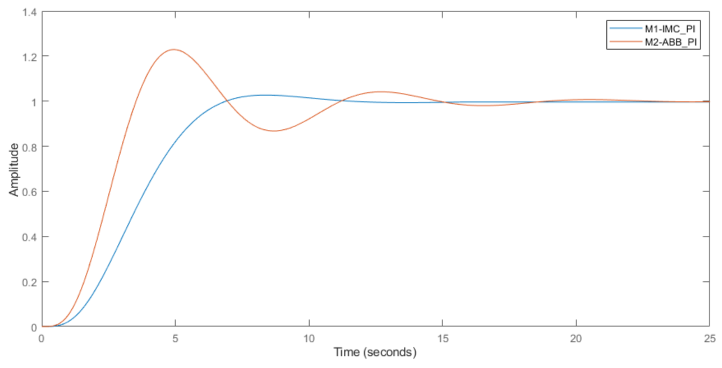
In conclusion we can say that the indices that present more uniformity from the point of view of the type of tuning rules selected are the two ISEs and $|e|$ with variations from benchmark to benchmark. The overshoot and settling time are the most variable because how the two indices are computed is highly dependent on the computational power and precision of the computer.

As reported we do not have a rule that is the best for all the indices and in the same index there can be a lot of variations of tuning rules. This analysis can be extended to other tuning rules to have a clearer view of which perform better and to other MPPs to have more approaches than aggressive and conservative.

Figures 4.10 finally show an example in which the selected MPP-TR compound is compared with another one in a load disturbance rejection case with the ISE as quality index and in a set-point tracking case with the maximum overshoot as quality index: as can be seen, a proper compound selection does help.



(a) ISE disturbance rejection



(b) Maximum overshoot set-point tracking

Figure 4.10: Comparison of the MPP-TR compound for disturbance rejection and set-point tracking. Process benchmark class 2, $\theta_{s1} = 0.46$, $\theta_{s2} = 0.51$

5 | Going Digital and then Event-Based

In this chapter we take the results of Chapter 4 and extend the technique so as to compute the event-related parameters of the selected realization paradigm. We will then test the obtained design in structurally nominal conditions.

For process in structurally nominal condition we mean that the process is not identified through a model parametrization procedure, hence the values of T, D and μ are considered "correct". The process in structurally nominal condition is the following FOPDT

$$P(s) = \mu \frac{e^{-sD}}{1+sT} \tag{5.1}$$

where $\mu = 1$, $T = 1$ and D comes from the normalized delay (3.2), such that $\theta \in (0, 1)$.

$$D = \frac{\theta T}{1-\theta} \tag{5.2}$$

We decided to keep the value of T fixed and make θ varying from 0 to 1 with a constant step because it was easier than combining different values for T and D to have θ varying with a constant step.

5.1. Event-Based Realization Technique

Starting from the realization [26] we used the technique proposed to compute Δ_R for all tuning rules previously tested.

The technique is the following:

1. Tune the continuous-time PI(D);
2. Form a continuous-time loop with the tuned controller and the normalized FOPDT, compute ω_c and the sampling time q as in (3.8);

3. Discretize the controller and an approximation of the process both at step q and form the process equivalent state-space representation (3.9) and the controller equivalent state-space matrices (3.14) in R-mode and H-mode;
4. compute the closed-loop switching dynamic matrix A_σ (3.15) for both R-mode and H-mode;
5. Compute the change of base T_H (3.17);
6. Evaluate the maximum multiplicative expansion $E_{H,\xi}$ (3.18);
7. Compute $\tilde{A_R} = T_H^{-1} A_R T_H$;
8. Repeatedly check the condition $E_{H,\xi} \|\tilde{A_R}^k\|_2 < 1 \quad \forall k \geq \Delta_R$ (3.19), until a suitable Δ_R is found.

The multiplicative expansion $E_{H,\xi}$, evaluated with the normalized FOPDT is $E_{H,\xi} = \sqrt{n_C}$. This means that the state of the system with a PI controller is non-expanding because the order of the controller n_C is 1 hence $E_{H,\xi} = 1$. For a PID the order of the controller is 2 hence the multiplying factor is $\sqrt{2}$.

To discretize the process and the controller we used the Backward Euler method

$$s = \frac{z-1}{q} z^{-1} \tag{5.3}$$

To approximate the process we decided to use $Padé_{1,1}$ approximation (5.4).

We keep the same parameters for all the computations setting $\Delta\varphi_m = 5°$ and $k_c = 0.5$, θ ranges from 0.1 to 0.9 with a step of 0.025. For the computation of Δ_R we decided to make k_a varying from 0.5 to 2 with a step of 0.25 to see the effect on the λ-dependent controllers.

5.2. Event-Based Realization Assessment

Now we present the results obtained from the technique described above.

In the figures 5.6 and 5.8 we can see that not all the tuning rules are able to satisfy the condition (3.19). During the iteration of $E_{H,\xi} \|\tilde{A_R}^k\|_2$ some rules fail to converge to zero with Δ_R that goes to infinity. We tested the step response of the closed-loop system of the failed configurations and the result is that some of them are unstable other have so many oscillations that the evaluation of the iterated condition can not be satisfied. Due to the fixed T this behaviour arises when there is a higher dead time D and, if present, a wrong acceleration factor for the dynamic of the closed-loop system.

Using the FOPDT process in structurally nominal conditions helped to diagnose that many of the tuning rules selected become unstable or too low damped when the ratio $\frac{D}{T}$ is too high. If the tuning rule depends on the parameter λ a too high accelerator factor coupled with an high $\frac{D}{T}$ can compromise the stability as well.

We observe a common behaviour for all the controllers, Δ_R decreases as θ grows and Δ_R increases as the accelerator factor k_a increases.

In the figures 5.6 and 5.8 it is possible to understand when a controller becomes unstable because Δ_R starts growing significantly; we did not plot the values of the unstable tuning rule, hence some lines are truncated.

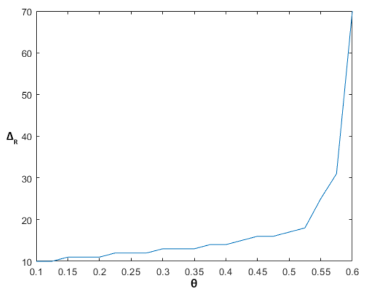
We observe that for $\theta \in [0.6, 0.7]$ all the tuning rules have a small spike, this is caused by the coupling of the discretization method and the Padé approximation of the process. We used $Pad\acute{e}_{1,1}$ approximation (5.4) and for $\theta = \frac{2}{3}$ the matrix b_P becomes singular hence the condition $E_{H,\xi}\|\tilde{A_R}^k\|_2 < 1$ cannot be satisfied. To prove this we repeated the technique using $Pad\acute{e}_{0,1}$ approximation (5.5), in this case Δ_R has a spike for $\theta = 0.5$. This has to be taken into account because the pure time delay has to be approximated, a solution could be to flatten the values around the singularity point considering it constant.

$$Pad\acute{e}_{1,1} = \frac{2 - sD}{2 + sD} \tag{5.4}$$

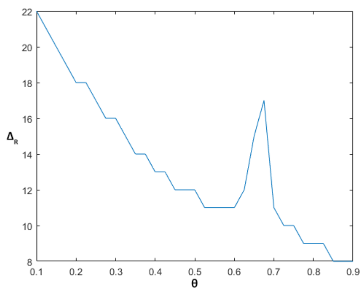
$$Pad\acute{e}_{0,1} = \frac{1}{1 + sD} \tag{5.5}$$

Analysing the PI tuning rule that are parameter-free we see that Δ_R decreases as θ increases for all the controllers except for ABB that has the opposite behaviour. This tuning rule has a growing Δ_R instead of a decreasing one and for $\theta > 0.6$ is unstable, for $\theta < 0.35$ it has results similar to the λ-dependent rules.

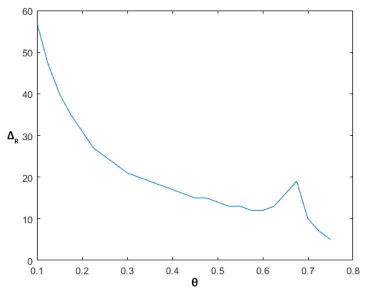
The other tuning rules show Δ_R usually higher with respect to the λ-dependent rules. The EB parameter decreases rapidly reaching the other rules for $\theta > 0.35$, an exception is H&A that has better performances than others from the beginning. LSM is another rule that is unstable for $\theta > 0.75$. Mann and D&C have very similar performances. Smith is the controller that starts with the worst performance, $\Delta_R > 90$, but improves quite rapidly then remains with average results. The 66% of the λ-independent rules are able to satisfy the condition (3.19).



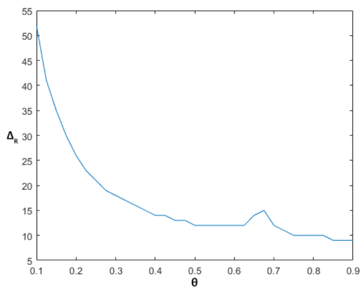
(a) ABB



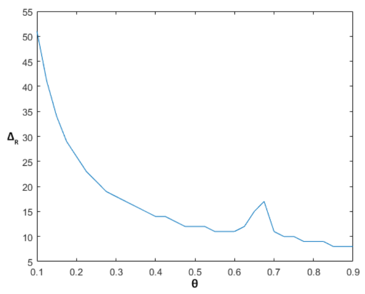
(b) H&A



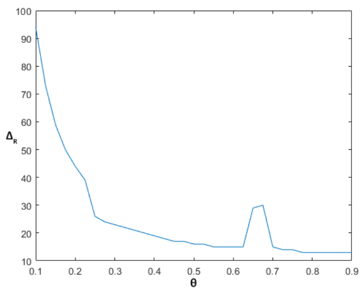
(c) LSM



(d) Mann



(e) D&C



(f) Smith

Figure 5.1: Δ_R plots of PI parameter-free tuning rules. Δ_R vertical axis, θ horizontal axis

Looking at the λ-dependent tuning rules we can see that the only controller that is stable for all θ and k_a is IMC-PI. Sko, as previously noted in section 4.2, tunes the controller identically to IMC-PI hence the EB parameters are the same. For $k_a = 0.5$ Δ_R is quite constant, as the accelerator factor increases the behaviour previously noted is more evident. Keeping θ fixed we can see clearly that Δ_R grows more for larger accelerator factor. As shown in the surface plot below for $\theta > 0.5$ Δ_R changes of few units in the k_a range with respect to the first half of the normalized time delay.

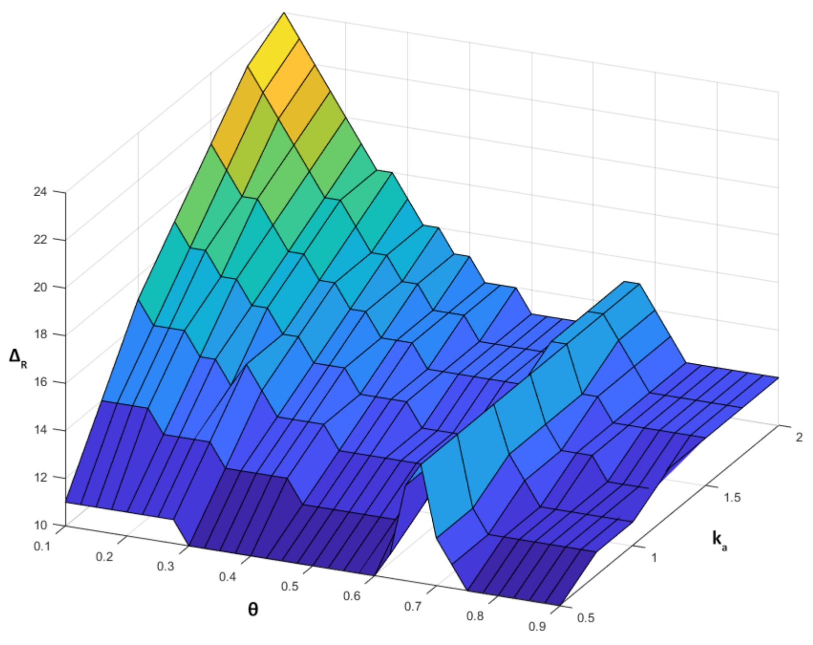


Figure 5.2: Surface plot of IMC-PI

Dsd is a tuning rule that fails to complete the technique for most of θ and the range of instability grows as k_a increases. For $k_a = 0.5$ Dsd is stable for $\theta \in [0.6, 0.9]$, this is the only time when this rule is stable in this range. For $k_a \geq 0.75$ the rule has usable results for $\theta < 0.4$. As noted for ABB Δ_R grows rapidly before becoming unstable.

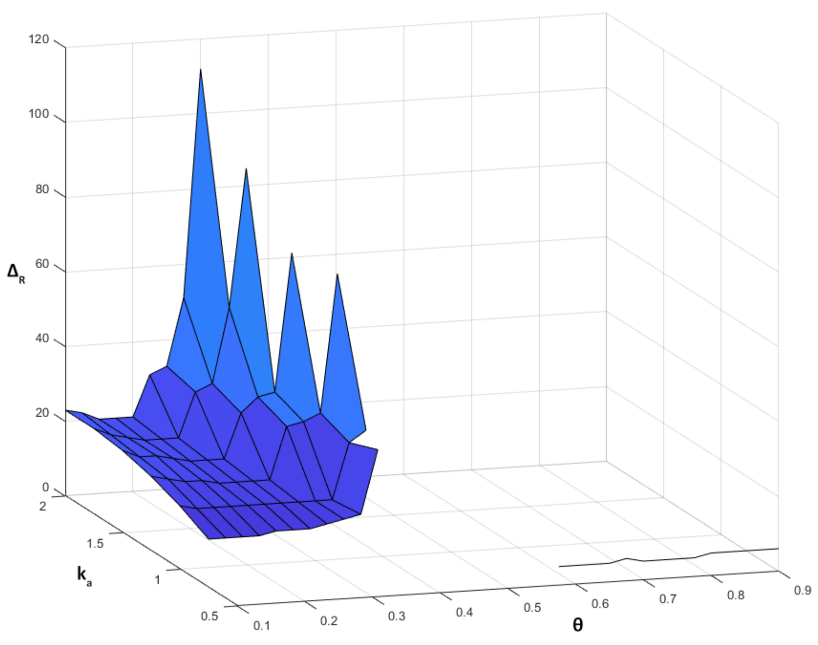


Figure 5.3: Surface plot of Dsd-PI

Riv is another rule that results unstable for larger values of k_a. We can see both behaviours, the decrement of Δ_R as θ increases followed by the quick growth of Δ_R due to the instability (e.g. for $k_a = 2$). As the accelerator factor is greater than 1.25 the range of stability decreases.

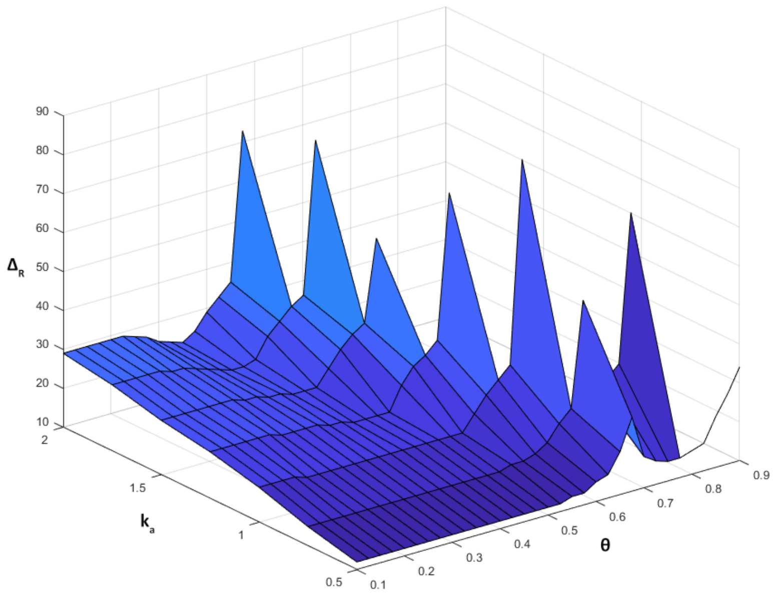
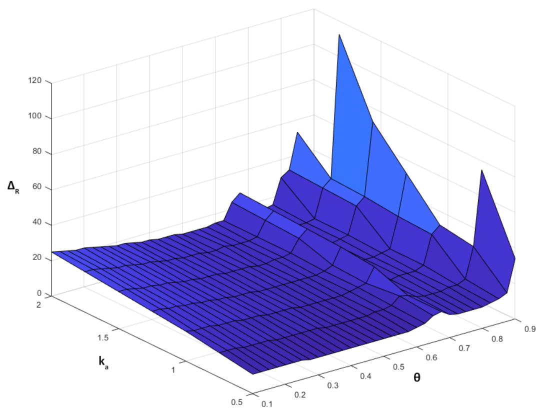
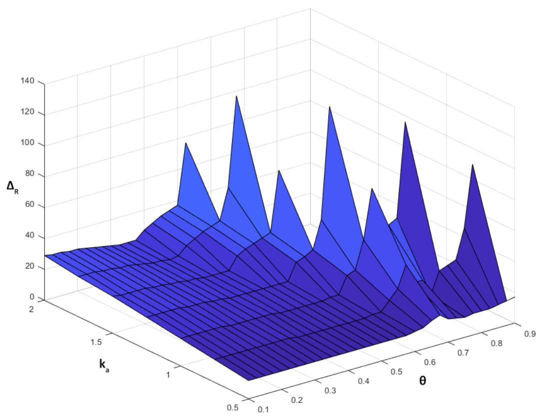


Figure 5.4: Surface plot of RIV-PI

Lee for most of θ values has good performances but for $\theta > 0.8$, it develops instability. We have a plateau for $\theta < 0.6$ followed by a spike caused by the approximation and the instability. I&G is similar to Lee but its interval of stability is smaller.



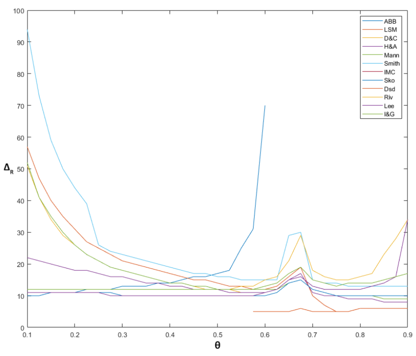
(a)



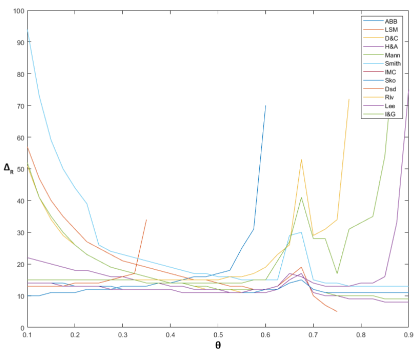
(b)

Figure 5.5: Surfaces for Lee-PI on the left and I&G-PI on the right

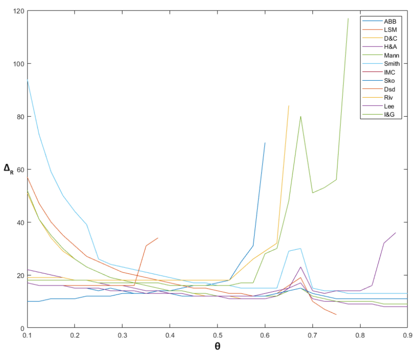
The following figures report all the PI tuning rules compared keeping the accelerator factor k_a fixed; on the horizontal axis there are the values of θ, on the vertical axis the values of the EB parameter Δ_R. As the accelerator factor increases many tuning rules become unstable. It can be seen that all tuning rules for $\theta \in [0.6, 0.7]$ present a spike. For small values of θ the λ-dependent controllers exploit the technique with similar values of Δ_R. For small k_a it is possible to recognize the parameter-free rules to the others because of the higher Δ_R value. As k_a grows, this distinction is less clear.



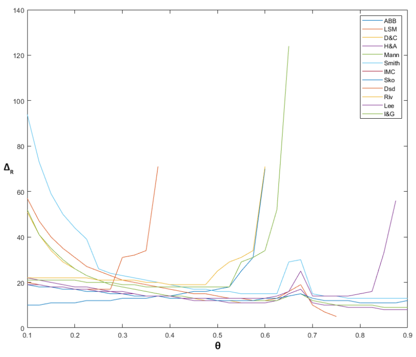
(a) $k_a = 0.5$



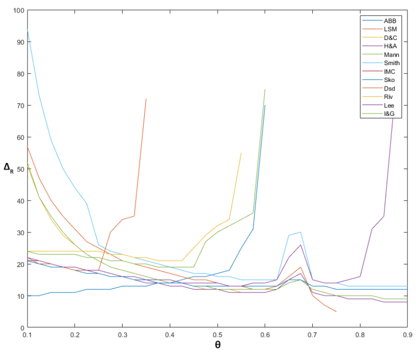
(b) $k_a = 0.75$



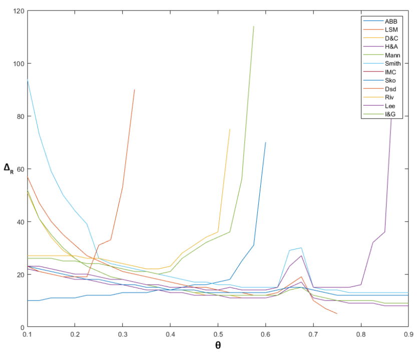
(c) $k_a = 1$



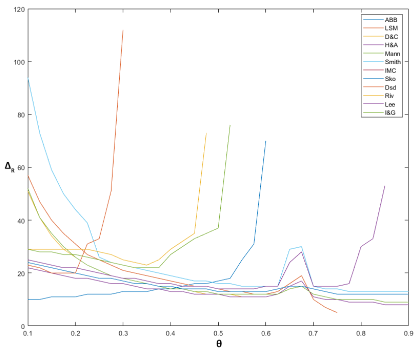
(d) $k_a = 1.25$



(e) $k_a = 1.5$



(f) $k_a = 1.75$



(g) $k_a = 2$

Figure 5.6: Δ_R comparison of PI tuning rules, k_a is fixed and θ is variable. Δ_R vertical axis, θ horizontal axis

Unlike PI controllers where the difference between the λ-dependent rules and the parameter-free ones is quite evident, the PID tuning rules present similar performances; we have a distinction between the parameter-dependent/independent rules but it is less clear as shown in figures 5.8. Most of the rules are stable in all the θ interval, the only controllers that are unstable for certain values are Sree and Lee.

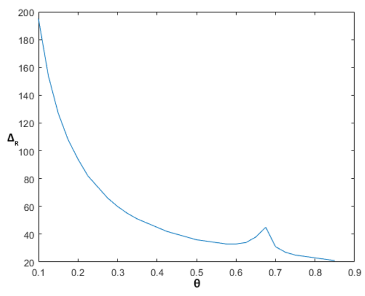
We can see that Connell and Moros show a similar behaviour, for $\theta < 0.6$ the curve becomes less steep where Δ_R decreases in the range of 5-10 unity. After the singularity region $\theta \in [0.6, 0.7]$ there is a quick decrement, we pass from $\Delta_R = 40$ to $\Delta_R = 10$.

Liptàk develops a similar behaviour to the previous two controllers but we can see an initial fast decrement for $\theta < 0.3$ then a plateau and for $\theta > 0.7$ we have another steep decrement.

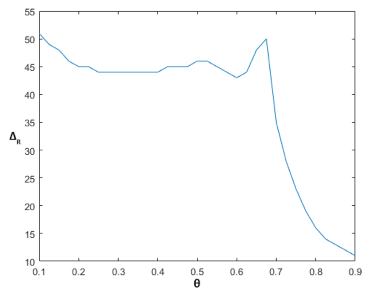
The other λ-independent rules have a similar behaviour to the one observed for PI controllers. Sree is a parameter-free rule and is able to satisfy the iterated condition for $\theta \in [0.1, 0.85]$, it behaves like Wang but with worse performances. Δ_R ranges from over 200 to 20 and for the first half of θs it has the worst performances.

Wang presents the Δ_R curve behaviour that is the most similar to the PI ones. It has the second worst performance and it is able to reach the other controllers for values close to $\theta = 0.6$.

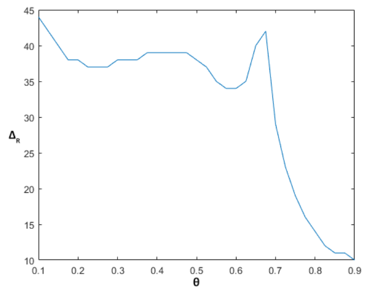
Frehauf is the only switching rule, we can see that for the first part ($\theta < 0.33$) we have an increment in Δ_R then when the control law changes it behaves like Wang.



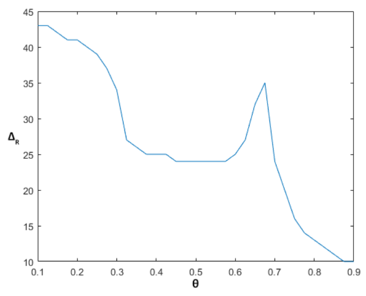
(a) Sree



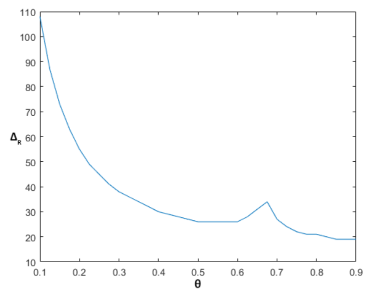
(b) Connell



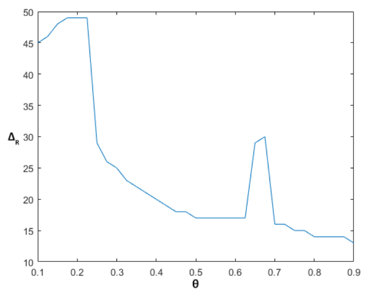
(c) Moros



(d) Liptàk

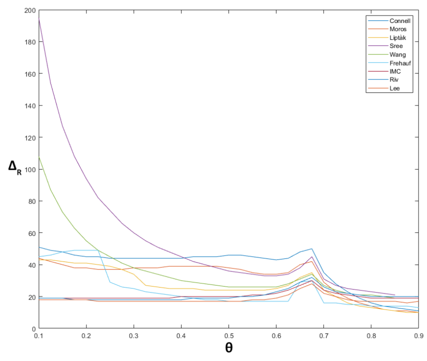


(e) Wang

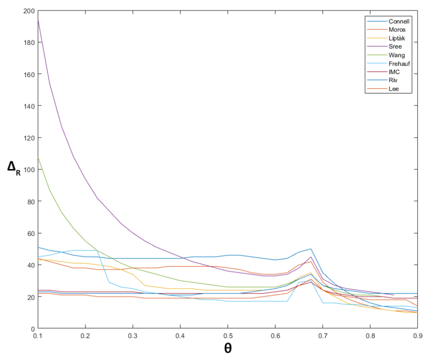


(f) Frehauf

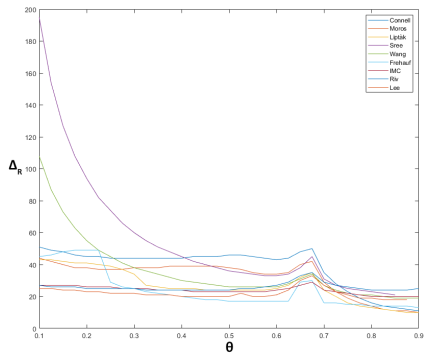
Figure 5.7: Δ_R plots of PID parameter-free tuning rules. Δ_R vertical axis, θ horizontal axis



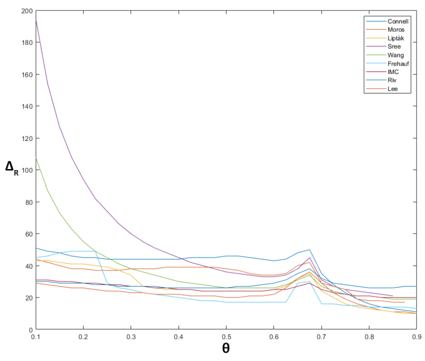
(a) $k_a = 0.5$



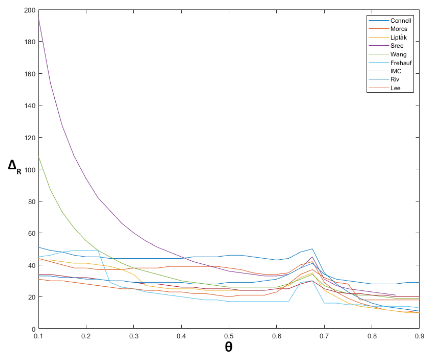
(b) $k_a = 0.75$



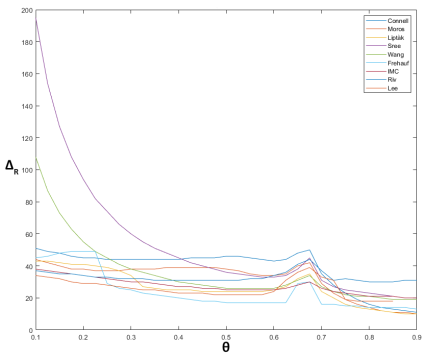
(c) $k_a = 1$



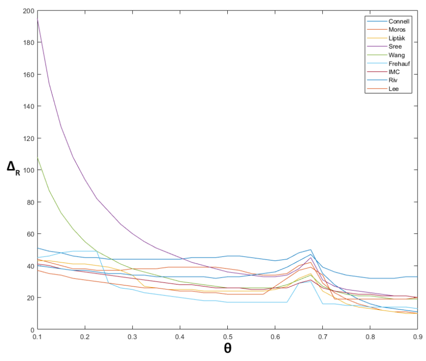
(d) $k_a = 1.25$



(e) $k_a = 1.5$



(f) $k_a = 1.75$

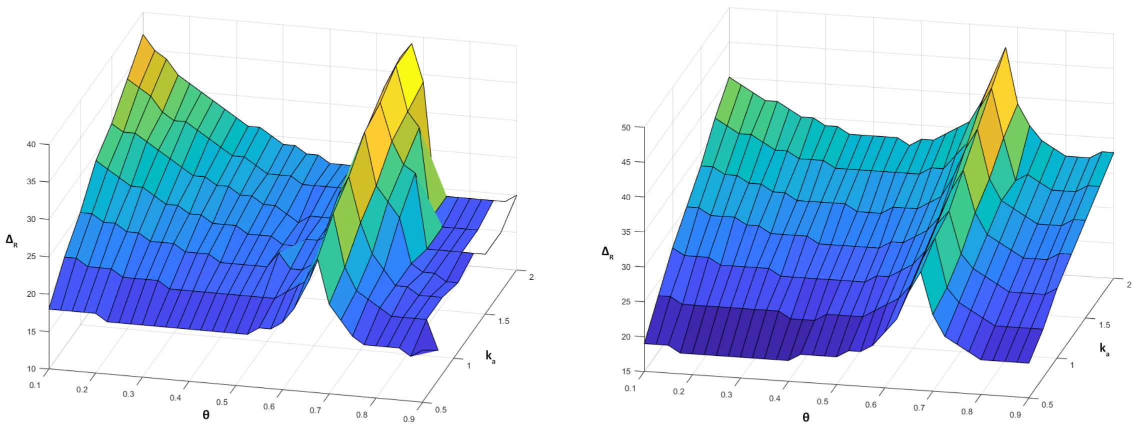


(g) $k_a = 2$

Figure 5.8: Δ_R comparison of PID tuning rules, k_a is fixed and θ is variable. Δ_R vertical axis, θ horizontal axis

The λ-dependent rules (IMC, Riv and Lee) have similar performances. The increment of Δ_R caused by an increasing k_a is more evident for small values of θ.

Lee has the best performances but becomes unstable for $\theta > 0.8$ and $k_a > 0.75$. It also has a significant spike in the interval of singularity. Riv, like Lee, has an evident spike in the same interval but it is stable for all θ and k_a. This tuning rule has higher Δ_R than the other λ-dependent tuning rules.



(a) (b)

Figure 5.9: Surfaces for Lee-PID on the left and Riv-PID on the right

IMC is in the middle. For lower θ IMC reaches the performances of Riv and for higher θ it has Δ_R similar to Lee. For $\theta \in [0.6, 0.7]$ it manages to keep Δ_R under control better than the other two controllers, this is similar to what happens for the PI implementation (figure 5.2).

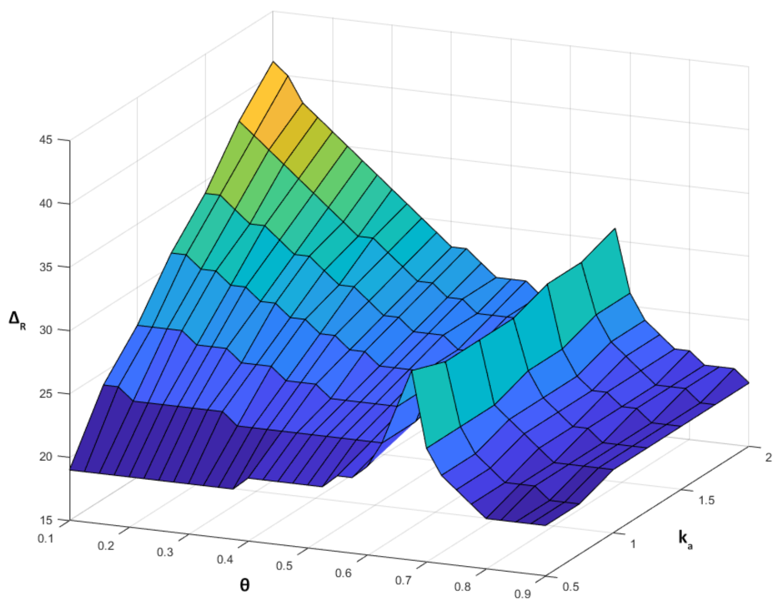


Figure 5.10: Surface plot of IMC-PID

From the analysis above, we can see that not all the tuning rules are suitable for this particular implementation. PI controllers have more problems, there are more tuning rules that are unstable and unable to satisfy the iterated condition. The results obtained in terms of Δ_R are similar, PID controllers return slightly higher results due to the higher complexity of the control structure for the presence of the derivative action. We can state that for lower θ and high k_a a higher number of iterations are necessary to satisfy the condition (3.19) therefore choosing an appropriate accelerator factor is very important if the goal is to reduce at a minimum Δ_R or to match the performances of the parameter-free tuning rules.

We can take the results above and interpolate some functions to compute the parameter Δ_R, in addition we can interpolate the sampling time q. For example the interpolating functions for ABB, $\theta \in [0.1, 0.6]$, are

$$\begin{aligned}\Delta_R = 8.855e^6\theta^8 - 2.311e^7\theta^7 + 2.549e^7\theta^6 - 1.547e^7\theta^5 + 5.624e^6\theta^4 \\ -1.249e^6\theta^3 + 1.646e^5\theta^2 - 1.717e^4\theta + 353.2\end{aligned} \tag{5.6}$$

$$q = -8.79\theta^5 + 23.49\theta^4 - 23.02\theta^3 + 10.41\theta^2 - 0.779\theta + 0.6482 \tag{5.7}$$

and for IMC-PID, $\theta \in [0.1, 0.9]$ and $k_a = 1$, are

$$\Delta_R = 3616\theta^6 - 1.01e^4\theta^5 + 1.079e^4\theta^4 - 5563\theta^3 1453\theta^2 - 195.7\theta + 26.55 \tag{5.8}$$

$$q = 149.2\theta^6 - 388.1\theta^5 + 400.2\theta^4 - 204.8\theta^3 + 54.46\theta^2 - 6.693\theta + 0.499 \tag{5.9}$$

These two parameters, related to an EB implementation, can be coupled with the results previously obtained in chapter 4.2 to choose the best model parametrization procedure-tuning rule compound for an event-based application.

This means that the rule previously set to find the best MPP-TR compound must be changed to return all the information previously returned and the event-related parameters. We have to take into account that not all the tuning rules are able to provide a value of Δ_R hence we should add some conditions to the comparative algorithm that produces the tables (e.g. figure 4.6), as a consequence the figures will probably change.

We could be satisfied with these results since the technique used to compute the event-related parameters starts from the design of the closed-loop system in continuous time but further research can be done. First of all we can test the technique with a campaign of benchmark directly in discrete-time to see if the stability is preserved and if the results for the best MPP-TR compound, obtained in continuous time, are valid for discrete time. Secondly we can implement an EB system to see how the number of transmission is managed by different tuning rules with respect to a fixed-rate implementation and how the tuning quality indices are affected by the EB implementation. This would be harder to carry out, but it can be object of a future research. The developer has to take into account, in addition to the problems that can arise in the realization of the controller, how events are created and managed by the event-triggering policy (e.g. send on delta, integrate and fire...) that depend on the specific implementation.

6 | Conclusions, Open Issues, Future Work

The goal of this thesis was to show that in the still flourishing field of autotuning new problems are emerging that deserve attention and to address two of these, namely

- the influence of the model parametrization procedure (MPP) on the results achievable by the controller,
- the need for extending tuning rules (TRs) to novel control computation paradigms such as the event-based one.

Concerning the first item, we proposed a technique that allows to choose the best MPP-TR compound given an index (discussed as well) to quantify the tuning quality for the particular problem at hand; the obtained results can be implemented and applied in the industry to ease the process of providing the optimal tuning for a given application.

Concerning the second item, we extended previous research results on an event-based paradigm based on multitransmission, using various tuning rules. We highlighted some problems about discretization and approximation of the process and the controller. We proved that the multitransmission realization is applicable with all the controllers, while the stability degree and more in general the results obtained depend on the tuning rule.

This treatise – as expected – leaves several open issues for future research, as outlined below.

- As said in Section 4.1.4 in order to apply correctly the online technique we have to develop a procedure to clearly determine to which benchmark class an unknown process belongs.
- It is necessary to implement and extend the technique for non model-based controllers.
- With reference to Section 5.1 we also need to perform, in the event-based framework, a complete benchmark campaign to have a clear idea of which controllers

save more transmissions maintaining good performances through the computation of performance indices.

- A study is in order on how to integrate the selection of the event-based parameters – in the addressed case, q and Δ_R – in the rule $nI[A, C]$. This is not an easy task because the indices are computed in the continuous time, hence we have to find a way to express them in discrete time effectively. This introduces more problems related to for example the discretization method or the choice of the sampling time.

As future work the rule $nI[A, C]$ can be modified to accept composite indices as input instead of simple indices. This could be done for example by assigning a weight to each index needed, e.g. someone could ask for the optimal tuning rule for 30% maximum $|e|$ and 70% ISE in disturbance rejection.

As for the identification of the benchmark class, we could develop a method that looks for similarities or could compute the difference between the real process data and the model; if the difference is smaller than a threshold then the process is identified. We could provide these information to a voting system composed by different elements. Each element will provide its identification of the benchmark class then all the results are compared and the most voted is selected.

The indices evaluation procedure can be extended to other model parametrization procedures and other tuning rules. We could also evaluate different tuning quality indices, in order to have a more comprehensive overview.

Bibliography

[1] A. Afram and F. Janabi-Sharifi. Theory and applications of hvac control systems–a review of model predictive control (mpc). *Building and Environment*, 72:343–355, 2014.

[2] A. P. Antony and E. Varghese. Comparison of performance indices of pid controller with different tuning methods. In *2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–6. IEEE, 2016.

[3] E. Aranda-Escolastico, M. Guinaldo, R. Heradio, J. Chacon, H. Vargas, J. Sánchez, and S. Dormido. Event-based control: A bibliometric analysis of twenty years of research. *IEEE Access*, 8:47188–47208, 2020.

[4] K. J. Åström and T. Hägglund. Benchmark systems for pid control. *IFAC Proceedings Volumes*, 33(4):165–166, 2000.

[5] K. J. Åström and T. Hägglund. Revisiting the ziegler–nichols step response method for pid control. *Journal of process control*, 14(6):635–650, 2004.

[6] B. Biswas, S. Chatterjee, S. Mukherjee, and S. Pal. A discussion on euler method: A review. *Electronic Journal of Mathematical Analysis and Applications*, 1(2):2090–2792, 2013.

[7] T. Blevins, M. Nixon, and W. Wojsznis. Event based control applied to wireless throttling valves. In *2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pages 1–6. IEEE, 2015.

[8] R. Braatz. Internal model control. In S. Levine, editor, *The control handbook*, pages 215–224. CRC Press, Boca Raton, FL, 1996.

[9] J. C. Butcher. A history of runge-kutta methods. *Applied numerical mathematics*, 20(3):247–260, 1996.

[10] D. Chen and D. Seborg. PI/PID controller design based on direct synthesis and disturbance rejections. *Industrial & Engineering Chemistry Research*, 41(19):4807–4822, 2002.

[11] B. Connell. *Process instrumentation applications manual.* McGraw-Hill Professional Publishing, 1996.

[12] C. Cox, P. Daniel, and A. Lowdon. QUICKTUNE: a reliable automatic strategy for determining PI and pPI controller parameters using a FOPDT model. *Control Engineering Practice*, 5(10):1463–1472, 1997.

[13] S. Durand and N. Marchand. Further results on event-based pid controller. In *2009 European control conference (ECC)*, pages 1979–1984. IEEE, 2009.

[14] P. S. Fruehauf, I.-L. Chien, and M. D. Lauritsen. Simplified imc-pid tuning rules. *ISA transactions*, 33(1):43–59, 1994.

[15] T. Hägglund and K. J. Åström. Revisiting the ziegler-nichols tuning rules for pi control. *Asian Journal of Control*, 4(4):364–380, 2002.

[16] A. J. Isaksson and S. F. Graebe. Analytical pid parameter expressions for higher order systems. *Automatica*, 35(6):1121–1130, 1999.

[17] N. Kazantzis and C. Kravaris. Time-discretization of nonlinear control systems via taylor methods. *Computers & chemical engineering*, 23(6):763–784, 1999.

[18] Y. Lee, S. Park, and M. Lee. Pid controller tuning to obtain desired closed-loop responses for cascade control systems. *IFAC Proceedings Volumes*, 31(11):613–618, 1998.

[19] A. Leva and A. Colombo. On the IMC-based synthesis of the feedback block of ISA-PID regulators. *Transactions of the Institute of Measurement and Control*, 26 (5):417–440, 2004.

[20] A. Leva and A. M. Colombo. On the imc-based synthesis of the feedback block of isa pid regulators. *Transactions of the Institute of Measurement and Control*, 26(5): 417–440, 2004.

[21] A. Leva and F. Donida. Normalised expression and evaluation of pi tuning rules. *IFAC Proceedings Volumes*, 41(2):12260–12265, 2008.

[22] A. Leva and F. Donida. Quality indices for the autotuning of industrial regulators. *IET Control Theory & Applications*, 3(2):170–180, 2009.

[23] A. Leva and M. Maggio. On the use of models with delay in pi (d) autotuning. In *49th IEEE Conference on Decision and Control (CDC)*, pages 3319–3324. IEEE, 2010.

[24] A. Leva and L. Piroddi. Model-specific autotuning of classical regulators: a neural approach to structural identification. *Control Engineering Practice*, 4(10):1381–1391, 1996.

[25] A. Leva, S. Negro, and A. V. Papadopoulos. Pi/pid autotuning with contextual model parametrisation. *Journal of Process Control*, 20(4):452–463, 2010.

[26] A. Leva, F. Terraneo, and S. Seva. A multitransmission event-based architecture for energy-efficient autotuning wireless controls. *IEEE Transactions on Control Systems Technology*, 30(4):1510–1524, 2021.

[27] Y. Li, K. H. Ang, and G. C. Chong. Patents, software, and hardware for pid control: an overview and analysis of the current art. *IEEE Control Systems Magazine*, 26(1): 42–54, 2006.

[28] Y. Li, K. Ang, and C. Chong. Patents, software, and hardware for PID control—an overview and analysis of the current art. *IEEE Control Systrems Magazine*, pages 42–54, february 2006.

[29] B. Lipták. Controller tuning ii: Problems and methods. *Control Engineering On Li ne*, 2001.

[30] X. Litrico, P.-O. Malaterre, J.-P. Baume, P.-Y. Vion, and J. Ribot-Bruno. Automatic tuning of pi controllers for an irrigation canal pool. *Journal of irrigation and drainage engineering*, 133(1):27–37, 2007.

[31] A. Lopez, C. Smith, and P. Murril. Controller tuning relationships based on integral criteria. *Instrument Technology*, 14(11):57, 1967.

[32] G. Mann, B.-G. Hu, and R. Gosine. Time-domain based design and analysis of new pid tuning rules. *IEE Proceedings-Control Theory and Applications*, 148(3):251–261, 2001.

[33] M. Morari and E. Zafiriou. *Robust Process Control.* Prentice-Hall, Upper Saddle River, NJ, 1989.

[34] R. Moros. Strecke mit ausgleich hoherer, 1999. URL `http://techni.tachemie.uni-leipzig.de/reg/regeintn.html`.

[35] A. O'dwyer. *Handbook of PI and PID controller tuning rules.* World Scientific Publishing Company, 2006.

[36] T. Ohta, N. Sannomiya, Y. Nishikawa, H. Tanaka, and K. Tanaka. A new optimization method of PID control parameters for automatic tuning by process computer. In

Prepr. IFAC Symposium on CAD of control systems, pages 133–138, Zürich, Switzerland, 1979.

[37] R. Padma Sree and M. Chidambaram. Control of unstable reactor with an unstable zero. *Indian Chemical Engineering*, 46:21–26, 2004.

[38] M. A. Rahimian and M. S. Tavazoei. Improving integral square error performance with implementable fractional-order pi controllers. *Optimal Control Applications and Methods*, 35(3):303–323, 2014.

[39] D. Rivera, M. Morari, and S. Skogestad. Internal model control 4 - PID controller design. *Industry and Engineering Chemical Process Design and Device*, 25(1):252–265, 1986.

[40] D. E. Rivera, M. Morari, and S. Skogestad. Internal model control: Pid controller design. *Industrial & engineering chemistry process design and development*, 25(1): 252–265, 1986.

[41] J. Sánchez, A. Visioli, and S. Dormido. An event-based pi controller based on feedback and feedforward actions. In *2009 35th Annual Conference of IEEE Industrial Electronics*, pages 1462–1467. IEEE, 2009.

[42] S. Seva, C. Cimino, and A. Leva. On the criticality of the model parametrisation method in industrial autotuning controllers. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 1137–1142. IEEE, 2021.

[43] F. G. Shinskey. Process control: as taught vs as practiced. *Industrial & engineering chemistry research*, 41(16):3745–3750, 2002.

[44] S. Skogestad. A method for improving the robustness of PID control. *IEEE Transactions on control systems technology*, 52(6):1669–1676, 2005.

[45] S. Skogestad. Tuning for smooth PID control with acceptable disturbance rejection. *Industrial & Engineering Chemistry Research*, 45(23):7817–7822, 2006.

[46] C. L. Smith. Intelligently tune pi controllers: automatic tuning offers only dubious advantages.(instrumentation & control). *Chemical Engineering*, 109(8):169–178, 2002.

[47] R. Socas, S. Dormido, and R. Dormido. Event-based control strategy for the guidance of the aerosonde uav. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2015.

[48] K. Sundaresan and P. Krishnaswamy. Estimation of time delay time constant param-

eters in time, frequency, and laplace domains. *The Canadian Journal of Chemical Engineering*, 56(2):257–262, 1978.

[49] U. Tiberi, J. Araújo, and K. H. Johansson. On event-based pi control of first-order processes. *IFAC Proceedings Volumes*, 45(3):448–453, 2012.

[50] D. Vrančić, Y. Peng, S. Strmčnik, and R. Hanus. A new tuning method for pi controllers based on a process step response. In *Proc. CESA'96*, pages 790–794, Lille, France, 1996.

[51] F.-S. Wang, W.-S. Juang, and C.-T. Chan. Optimal tuning of pid controllers for single and cascade control loops. *Chemical Engineering Communications*, 132(1): 15–34, 1995.

[52] X.-M. Zhang, Q.-L. Han, and B.-L. Zhang. An overview and deep investigation on sampled-data-based event-triggered control and filtering for networked systems. *IEEE Transactions on industrial informatics*, 13(1):4–16, 2016.

[53] J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Transactions of the American society of mechanical engineers*, 64(8):759–765, 1942.

A | Appendix A

In figures A.1, A.2 and A.3 are reported the Matlab script used to evaluate the indices and compute the tables; the additional Matlab packages necessary to run the script are ‘Symbolic Math Toolbox’ and ‘Control System Toolbox’.

We report only the script for the first benchmark class, in order to evaluate the other classes we have to change only the benchmark class parameter ‘a’ and the benchmark process ‘P’.

The indices are grouped in matrices of dimension rw by bb. Where rw is the total number of the controllers and bb is the length of the vector of the benchmark class parameter. Each index matrix is named as “Cn_I_A/P” where Cn is the class number, I is the index and $\_A/P$ is the parametrization procedure used, A for areas method and P for percentage procedure. For I we have “$errDR$” for maximum error in disturbance rejection, “$iseDR$” for ISE in disturbance rejection, “$iseSP$” for ISE in set-point tracking, “$ovrSP$” for maximum overshoot in set-point tracking and “$t99setSP$” for 99% settling time in set-point tracking.

```
clear all
close all

%% parameters definitions
a=2:1:8; % benchamark class parameter
ka=1;
min_phase=20;

bb=size(a);
C1_errDR_A=ones(21,bb(2));
C1_errDR_P=ones(21,bb(2));
C1_iseDR_A=ones(21,bb(2));
C1_iseDR_P=ones(21,bb(2));
C1_iseSP_A=ones(21,bb(2));
C1_iseSP_P=ones(21,bb(2));
C1_ovrSP_A=ones(21,bb(2));
C1_ovrSP_P=ones(21,bb(2));
C1_t99setSP_A=ones(21,bb(2));
C1_t99setSP_P=ones(21,bb(2));

%% indices computation
for u=1:length(a)
    % areas method
    syms s;
    syms x;

    P=1/(1+s)^a(u); %benchmark class process

    Ps=P/s;
    ys=ilaplace(Ps,x);

    fun1=matlabFunction(1-(ys));
    fun2=matlabFunction(ys);

    A0=integral(fun1,0,Inf);
    A1=integral(fun2,0,A0);

    % equivalent FOPDT parameters
    mu=1;
    T=A1/mu*exp(1);
    D=A0/mu-T;
    theta_S_A(u)=D/(T+D);
    clear s;
    s=tf('s');

    P=1/(1+s)^a(u); %benchmark class process
    % evaluation of indices for all tuning rules
    for rw=1:1:21
        C=tuning_Rules(T,D, ka, mu, rw, s);
        L=C*P;

        [Gm,Pm,Wcg,Wcp] = margin(L);
        Pm_A(rw,u)=Pm;   % phase margin

        yd=P/(1+L);     % y/d Complemetnary sensitivity function T(s)
```

Figure A.1: Matlab script 1 of 3

```
    yy0=L/(1+L);    % y/y0 Control sensitivity function K(s)

    % step responses
    t=0:0.001:500;  % simulation time
    [yy0Step, yy0Tout]=step(yy0,t);
    [ydStep, ydTout]=step(yd,t);

    % indices evaluation
        %  Max overshoot set-point tracking
    m=max(yy0Step);
    yInf=yy0Step(end);
    C1_ovrSP_A(rw,u)=(m-yInf)/yInf;
        % 99% settling time set-point tracking
    SI= stepinfo(L/(1+L));
    C1_t99setSP_A(rw,u)=SI.SettlingTime;
        % ISE set-point tracking
    C1_iseSP_A(rw,u)=trapz(0.001,(1-yy0Step).^2);
        % ISE disturbance rejection
    C1_iseDR_A(rw,u)=trapz(0.001,(1-ydStep).^2);
        % max |e| disturbance rejection
    errP=abs(max(ydStep));
    errM=abs(min(ydStep));
    if errP>errM
        C1_errDR_A(rw,u)=errP;
    end
    if errM>errP
        C1_errDR_A(rw,u)=errM;
    end
end
% percentage procedure, Sundaresan and Krishnaswamy method
clear s
s=tf('s');

P=1/(1+s)^a(u); %benchmark class process

t=0:0.001:100;
[ys,ysTout]=step(P,t);
for i=1:length(ysTout)
    if ys(i)>=0.353
        t1=ysTout(i);
        break
    end
end
for i=1:length(ysTout)
    if ys(i)>=0.853
        t2=ysTout(i);
        break
    end
end

%equivalent FOPDT parameters
mu=1;
T=2/3*(t2-t1);
D=1.3*t1-0.29*t2;
theta_S_P(u)=D/(T+D);
```

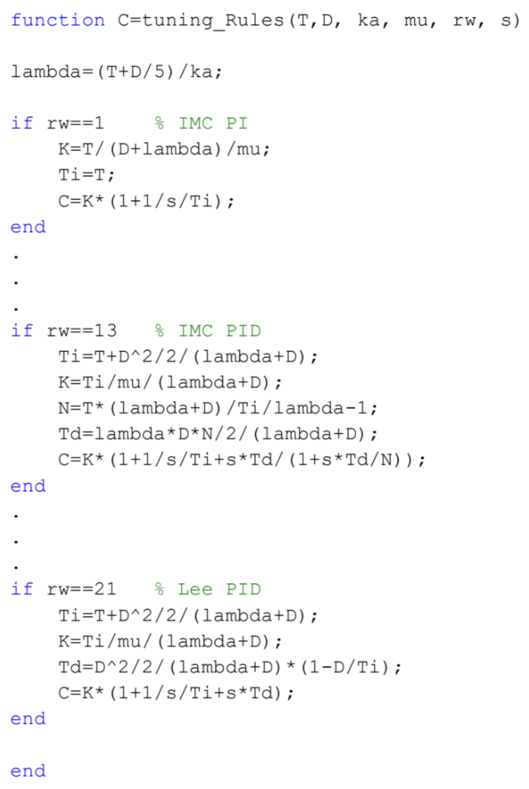
Figure A.2: Matlab script 2 of 3

```
        for rw=1:1:21
            C=tuning_Rules(T,D, ka, mu, rw, s);
            L=C*P;

            [Gm,Pm,Wcg,Wcp] = margin(L);
            Pm_P(rw,u)=Pm;  %phase margin

            yd=P/(1+L);      % y/d Complemetnary sensitivity function T(s)
            yy0=L/(1+L);     % y/y0 Control sensitivity function K(s)

            % step responses
            t=0:0.001:500;  % simulation time
            [yy0Step, yy0Tout]=step(yy0,t);
            [ydStep, ydTout]=step(yd,t);

            % indices evaluation
                %  Max overshoot Set-point tracking
            m=max(yy0Step);
            yInf=yy0Step(end);
            C1_ovrSP_P(rw,u)=(m-yInf)/yInf;
                % 99% settling time Set-point tracking
            SI= stepinfo(L/(1+L));
            C1_t99setSP_P(rw,u)=SI.SettlingTime;
                % ISE Set-point tracking
            C1_iseSP_P(rw,u)=trapz(0.001,(1-yy0Step).^2);
                % ISE Disturbance rejection
            C1_iseDR_P(rw,u)=trapz(0.001,(1-ydStep).^2);
                % max |e| Disturbance rejection
            errP=abs(max(ydStep));
            errM=abs(min(ydStep));
            if errP>errM
                C1_errDR_P(rw,u)=errP;
            end
            if errM>errP
                C1_errDR_P(rw,u)=errM;
            end
        end
    end
```

Figure A.3: Matlab script 3 of 3

The function $tuning\_Rules(T, D, ka, mu, rw, s)$ in figure A.4 returns the controller C tuned.

It is a simple switching function where the switching parameter is rw, the other input parameters are the process parameters computed by the model parametrization procedure, the accelerator factor ka and the continuous time transfer function s. We decided to show one case for each type of controller because completing with the other rules is straightforward. The PI tuning rule cases are in order from $rw = 1$ to $rw = 12$: IMC, Sko, Riv, Dsd, Abb, LSM, D&C, Mann, H&A, Lee, I&G and Smith. The PID tuning rules cases are in order from $rw = 13$ to $rw = 21$: IMC, Connell, Moros, Liptàk, Sree, Wang, Frehauf, Riv and Lee.



```
function C=tuning_Rules(T,D, ka, mu, rw, s)

lambda=(T+D/5)/ka;

if rw==1    % IMC PI
    K=T/(D+lambda)/mu;
    Ti=T;
    C=K*(1+1/s/Ti);
end
.
.
.
if rw==13   % IMC PID
    Ti=T+D^2/2/(lambda+D);
    K=Ti/mu/(lambda+D);
    N=T*(lambda+D)/Ti/lambda-1;
    Td=lambda*D*N/2/(lambda+D);
    C=K*(1+1/s/Ti+s*Td/(1+s*Td/N));
end
.
.
.
if rw==21   % Lee PID
    Ti=T+D^2/2/(lambda+D);
    K=Ti/mu/(lambda+D);
    Td=D^2/2/(lambda+D)*(1-D/Ti);
    C=K*(1+1/s/Ti+s*Td);
end

end
```

Figure A.4: Function tuning_Rules

The figure A.5 show a portion of code used for comparing the tuning rules. The first part of the code defines the matrices that will be displayed, they have dimensions $rw + 1$, $bb + 1$ otherwise the last tuning rule and the last value of θ_s will not be plotted by the function *pcolor*(...). This script compares the indices results keeping the smaller one and highlighting the corresponding tuning rule in the matrix displayed. In the comparison we maintained separated PI from PID.

We show only the comparison for the maximum error in disturbance rejection for the first parametrization procedure, the others are straightforward.

```
%% comparison and plot
min_phase=20;
C1_errDR_A_best=zeros(21+1,bb(2)+1);
C1_errDR_P_best=zeros(21+1,bb(2)+1);
C1_iseDR_A_best=zeros(21+1,bb(2)+1);
C1_iseDR_P_best=zeros(21+1,bb(2)+1);
C1_iseSP_A_best=zeros(21+1,bb(2)+1);
C1_iseSP_P_best=zeros(21+1,bb(2)+1);
C1_ovrSP_A_best=zeros(21+1,bb(2)+1);
C1_ovrSP_P_best=zeros(21+1,bb(2)+1);
C1_t99setSP_A_best=zeros(21+1,bb(2)+1);
C1_t99setSP_P_best=zeros(21+1,bb(2)+1);
for u=1:length(a)
    % areas method
    % |e|
    best=9999999999999999999999999;
    rwL=1;
    for rw=1:1:12     % PI
        if C1_errDR_A(rw,u)<best && (Pm_A(rw,u)>min_phase || ~isnan(C1_t99setSP_A(rw,↙
u)))
            if rwL~=rw
                C1_errDR_A_best(rwL,u)=0;
            end
            C1_errDR_A_best(rw,u)=1;
            best=C1_errDR_A(rw,u);
            rwL=rw;
        end
    end
    rwL=13;
    best=9999999999999999999999999;
    for rw=13:1:21     % PID
        if C1_errDR_A(rw,u)<best && (Pm_A(rw,u)>min_phase || ~isnan(C1_t99setSP_A(rw,↙
u)))
            if rwL~=rw
                C1_errDR_A_best(rwL,u)=0;
            end
            C1_errDR_A_best(rw,u)=1;
            best=C1_errDR_A(rw,u);
            rwL=rw;
        end
    end
end
```

Figure A.5: Indices comparison

List of Figures

3.1 Event-based loop 13
3.2 Switching representation of the controller 14

4.1 Block diagram of the considered closed-loop system 21
4.2 Flowchart of the offline part of the evaluation technique 23
4.3 Flowchart of the online part of the evaluation technique 26
4.4 θ_s comparison with respect to benchmark class parameter, the colors are blue for M_1 and red for M_2 28
4.5 Best tuning rule per index, benchmark class 1. Red best tuning rule, Gray others 30
4.6 Best tuning rule per index, benchmark class 2. Red best tuning rule, Gray others 31
4.7 Best tuning rule per index, benchmark class 3. Red best tuning rule, Gray others 32
4.8 Best tuning rule per index, benchmark class 4. Red best tuning rule, Gray others 33
4.9 Best tuning rule per index, benchmark class 5. Red best tuning rule, Gray others 34
4.10 Comparison of the MPP-TR compound for disturbance rejection and set-point tracking. Process benchmark class 2, $\theta_{s1} = 0.46$, $\theta_{s2} = 0.51$ 35

5.1 Δ_R plots of PI parameter-free tuning rules. Δ_R vertical axis, θ horizontal axis 40
5.2 Surface plot of IMC-PI 41
5.3 Surface plot of Dsd-PI 41
5.4 Surface plot of RIV-PI 42
5.5 Surfaces for Lee-PI on the left and I&G-PI on the right 42
5.6 Δ_R comparison of PI tuning rules, k_a is fixed and θ is variable. Δ_R vertical axis, θ horizontal axis 44

5.7 Δ_R plots of PID parameter-free tuning rules. Δ_R vertical axis, θ horizontal axis 46
5.8 Δ_R comparison of PID tuning rules, k_a is fixed and θ is variable. Δ_R vertical axis, θ horizontal axis 47
5.9 Surfaces for Lee-PID on the left and Riv-PID on the right 48
5.10 Surface plot of IMC-PID 49

A.1 Matlab script 1 of 3 60
A.2 Matlab script 2 of 3 61
A.3 Matlab script 3 of 3 62
A.4 Function tuning_Rules 63
A.5 Indices comparison 64

List of Tables

3.1 PI tuning rules 9
3.2 PID tuning rules 11

Acknowledgements

Il primo ringraziamento va ai miei genitori per avermi permesso di intraprendere questo percorso di studi.

Un enorme grazie va a mia sorella Lara e a tutta la mia famiglia per aver sempre creduto in me sostenendomi e aiutandomi durante tutti gli alti e bassi di questi anni.

Ringrazio il Professore Albero Leva per avermi dato l'opportunità di lavorare a questa tesi e per la disponibilità per chiarimenti e spiegazioni. Grazie a lui ho capito cosa significa svolgere un lavoro di ricerca e come bisogna destreggiarsi per fare in modo che il proprio lavoro sia il più chiaro e completo possibile.

Ringrazio i miei amici storici per essere una valvola di sfogo e un punto fisso per stemperare la tensione e farsi quattro risate davanti a una birra. Grazie ai nuovi amici trovati sui banchi dell'università per aver alleggerito le tante ore di lezione e per essere un punto di confronto e di scambio costante.