NTNU
Norwegian University of
Science and Technology

Optimal PID settings for first and second-order processes

Comparison with different controller tuning approaches

Iosif Pappas

Master's Thesis
Submission date: July 2016
Supervisor: Sigurd Skogestad, IKP
Co-supervisor: Ioannis Kookos, IKJ

Norwegian University of Science and Technology
Department of Chemical Engineering

Optimal PID settings for first and second-order processes

Comparison with different tuning approaches

by

Iosif Pappas

NTNU
Norwegian University of Science and Technology
Department of Chemical Engineering
Process Systems Engineering Group

Trondheim, July 2016

Abstract

PID controllers are extensively used in industry. Although many tuning methodologies exist, finding good controller settings is not an easy task and frequently optimization-based design is preferred to satisfy more complex criteria. In this thesis, the focus was to find which tuning approaches, if any, present close to optimal behavior.

Pareto-optimal controllers were found for different first and second-order processes with time delay. Performance was quantified in terms of the integrated absolute error, while robustness was defined as the maximum peak of the sensitivity function. Ideal PID controllers were used to create SIMC, K-SIMC and optimal tuning curves. As for the FOPTD, the SIMC performed close to the optimal for the whole robustness region. On the other hand, the K-SIMC settings performed well and outperformed the SIMC at the less robust region. That is due to the fact, that the K-SIMC rules suggest a PI controller for small values of the sensitivity peak. When it comes to the SOPTD, the SIMC and the K-SIMC were almost identical performance-wise. However, when the time delay is relatively small the I-SIMC tunings should be chosen. If higher-order models are required to be approximated as a first or second-order process with time delay, the half rule approximations achieved superior results when compared to the K-SIMC approximations.

Furthermore, SIMC, K-SIMC and optimal controllers were found for a given process and for specified robustness. The K-SIMC rules suggested a PI controller, instead of the PID controllers which were suggested by the other two tuning approaches. The controllers were applied to models, which had slightly different characteristics than the original process. The PI controller handled a wider variety of processes. However, similar results would have been derived for a PI SIMC or for an optimal PI controller.

In addition, two other tuning approaches were investigated. Firstly, the Syrcos & Kookos settings were found and compared with the optimal and the SIMC. The former methodology showcased very satisfactory results, which could be proven to be more beneficial than the SIMC for processes with relatively large time delay. Secondly, the MATLAB™ Tuning Toolbox was used to find balanced controllers, in terms of performance and robustness. First and second-order processes with varying time delay values were examined. Although the ™ Tuning Toolbox presented acceptable results, it should not be used for small time delay values due to the low proportional gain and the absence of derivative action of the controller.

Finally, optimal, SIMC and K-SIMC controllers were tested on the thermal/optimal plant uDAQ28/LT, where only the temperature was the measured variable. The process was approximated as an integrating process and due to the high input usage and the small compensation from the use of derivative action, a PI controller is the most advantageous available choice.

Acknowledgments

I want to wholeheartedly thank my supervisor, Professor Sigurd Skogestad, for accepting as a member of the Process Systems Engineering Group and letting me work on this project. Without his motivation and contribution, this thesis would never have emerged. His feedback was always invaluable to me, especially when the results were not as expected.

Words can not describe how grateful i am for the constant guidance, patience and support of my supervisor, Associate Professor, Ioannis Kookos. He was there for me throughout this thesis and my studies in general. He always encouraged me, especially after disappointments, which is something that i will never forget. My education would not have been the same without him.

My thanks go to all the people from the Process Systems Engineering Group, here in Trondheim, for making me feel that the 2nd floor of K4 was my home. However, i want to especially thank Chriss Grimholt who provided me his unpublished work. Without Chriss this thesis would not have been the same. His comments and suggestions were always on point. My special thanks go also to Vladimiros Minasidis for everything he did for this work and for making me enjoy Trondheim even more. But, i mostly want to thank him for giving me incentives to learn interesting things. I really hope i will get to meet all of them again in the future.

Professor Petros Koutsoukos helped me so much with his advice and making this trip a reality. Thank you very much.

This thesis would have impossible to complete without the endless love and support of Antigoni even though sometimes i was thinking about Process Control when i shouldn't. Also, thanks to my good friends from the University of Patras, Konstantinos, Leonidas and Spyros and to my flatmates in Norway, Ji, Peng and Yi, who will graduate this year from Jiao Tong in Shanghai and to Heiko.

Finally, i want to dedicate this thesis to my family for all the sacrifices they do, so that i am here today writing these words.

The financial support of the Erasmus+ program is greatly acknowledged.

Iosif Pappas
July 2016
Trondheim, Norway

Declaration of Compliance

I hereby declare that this thesis is an independent work in agreement with the exam rules and regulations of the Norwegian University of Science and Technology.

Trondheim, July 30, 2016

Ιωσήφ Παππάς

Contents

List of Figures

List of Tables

Nomenclature

Greek Symbols

Symbol	Description	Unit
ω	Frequency	$\mathrm{rad\,s^{-1}}$
φ_{du}	Input disturbance normalization factor	
φ_{dy}	Output disturbance normalization factor	
τ	Time	s
τ_{D}	Controller derivative time	s
τ_{I}	Controller integral time	s
τ_0	Process time constant	
ϑ	Process time delay	
ϑ'	Normalized process time delay	
τ_{I}'	Controller integral time in series form	s
${\tau_{\mathrm{D}}}'$	Controller derivative time in series form	s

Roman Symbols

Symbol	Description	Unit
$\mathrm{d_u}$	Plant step input disturbance	
$\mathrm{d_y}$	Plant step output disturbance	
e	Process error	
f	PID series and parallel form connection factor	
G	Process model	

IAE	Integrated absolute error	
$\mathrm{IAE}_{\mathrm{du}}$	IAE for PID controller only with step input disturbance	
$\mathrm{IAE}^{\mathrm{o}}_{\mathrm{du}}$	Reference IAE for PID controller only with step input disturbance	
$\mathrm{IAE}_{\mathrm{dy}}$	IAE for PID controller only with step output disturbance	
$\mathrm{IAE}^{\mathrm{o}}_{\mathrm{dy}}$	Reference IAE for PID controller only with step output disturbance	
J	Cost function - Performance	
K	Process controller	
$\mathrm{k_c}$	Process gain	
$\mathrm{k_d}$	Derivative gain	s
$\mathrm{k_i}$	Integral gain	s^{-1}
$\mathrm{k_p}$	Process gain	
M^{ub}	Sensitivity peak upper bound	
$\mathrm{M_S}$	Peak of the Sensitivity function	
$\mathrm{M_T}$	Peak of the Complimentary Sensitivity function	
n	Process noise	
S	Sensitivity function	
s	Laplace variable	
T	Complimentary sensitivity function	
$\mathrm{T}^{\mathrm{inv}}_{\mathrm{j0}}$	Positive numerator time constant	
u	Controller output	
$\mathrm{y_s}$	Set-point value	
G'	Approximated process model	
$\mathrm{k_c}'$	Controller gain in series form	

List of Abbreviations

Abbreviation	Description
FOPTD	First-order process with time delay
I-SIMC	Improved Simple Internal Model Control
MIMO	Many Inputs Many Outputs
SIMC	Simple Internal Model Control
SOPTD	Second-order process with time delay
PO	Pareto - Optimal

Chapter 1

Introduction

1.1. Motivation

The proportional-integral-derivative (PID) controllers are the main controllers used in industry. Therefore, it is of great significance to efficiently tune them. Many tuning methods and approaches have been proposed for the determination of the tuning parameters. Some of the previous work includes the paper by Ziegler and Nichols, the IMC tunings by Rivera et al. and the direct synthesis tuning rules in the book by Smith and Corripio. In 2003, Skogestad proposed the famous SIMC tuning rules, which were lately revisited by Lee et al.

Grimholt and Skogestad [2012, 2013] have done extensive work on finding optimal controllers and they have tested both PI and PID controllers using numerical gradients. Moreover, in 2016, they have successfully applied analytical expressions for the gradients of the objective functions and the constraints, in order to find optimal controllers, which satisfy specific design criteria. The optimal controllers are derived through the minimization of the Integral Absolute Error (IAE), which is determined as the performance objective. Nonetheless, the controllers should have the same robustness in order to be compared. Thus, the sensitivity peak, $\mathrm{M_S}$, is selected as the robustness criterion.

Since optimal controllers could be found, they could compare them with SIMC controllers. The SIMC rules have been assessed in terms of optimality for PI and PID controllers for numerous processes and different parametrizations (Grimholt and Skogestad [2012, 2013]). PI, PID or even Smith Predictors have been thested by comparing them with the Pareto optimal and surprisingly, the SIMC rules gave controllers close to the optimal. The most recent case examined, which was presented here in Trondheim, was the double integrating process, which proved that SIMC rules give controllers with almost identical performance with the optimal.

However, there are numerous methodologies that have not been tested in terms of their optimality. Therefore, for various processes it was interesting to see, which approaches are the most advantageous to follow. In addition, their sensitivity in the presence of model uncertainty for a given robustness might lead the engineer to different choices.

1.2. Aim of this work

Grimholt and Skogestad, did not only give the opportunity and the capability to find optimal controllers but also to compare them only with SIMC controllers. Taking advantage of their approach, it is possible to compare controllers obtained using different methodologies and evaluate them in terms of their performance and their robustness.

This thesis is an effort to try to extend the results to even more processes. A variety of values of time constants and time delays is examined. For those processes, the SIMC, the I(improved)-SIMC and the K-SIMC rules are utilized to find sufficient controllers for different robustness constraints. Furthermore, the approach proposed by Syrcos & Kookos is investigated, since it gave promising results. Finally, PID tunings are compared against the tunings suggested by the MATLAB Tuning Toolbox. The main focus of this study was to evaluate how close to the optimal are the tunning parameters that each methodology gives

In addition to that, it is often very difficult to find or approximate the model of a plant. Thus, it is even more complicated to design sufficient controllers. For that reason, it is interesting to see how *sensitive* the controller tunings are for cases with model error. As a final part of this work, it was interesting to evaluate the different controllers in a practical example and demonstrate their performance and robustness on a real process. Therefore, in the last section of the results, a case study is presented and a try-out of the controllers for setpoint changes and disturbances is conducted.

1.3. Thesis Overview

This Thesis consists, besides the introductory and concluding remarks, of three main chapters. More specifically, **Chapter 2** covers the all the necessary theoretical aspects of this work. The tuning methodologies, which are used in this work are analytically presented. **Chapter 3** describes the Pareto optimization and the optimization methodology in general. **Chapter 4** contains all the results, which were derived using the different controller tuning methodologies. The tuning approaches are compared in terms of their performance and their robustness. Various processes are considered and the SIMC, the I-SIMC, the K-SIMC, the Syrcos & Kookos method and MATLAB Toolbox settings are applied. Moreover, the sensitivity of the PID settings to model error is investigated to evaluate the performance of the controllers in processes with model error. In the final part the tunings are tested in a case study, a thermal/optical plant. Furthermore, Chapter 4 captures all the analysis and the discussion of the results. Ultimately, future work is discussed.

1.4. Appendices

The appendices consist of two parts: one which contains MATLAB scripts, utilized for the results and a second one which has all the Simulink schemes used.

Chapter 2

Theoretical Background

2.1. Some history

Nowadays, process control plays one the most significant roles in the efficient, safe and smooth operation of an operation unit or even a plant. However, how was that point reached?

Ctesibius (Κτησίβιος, 285 ∼ 222 B.C.) was a Greek inventor and mathematician, who lived in Alexandria, Egypt. His device, which was called, the hydrion horoskopion (”ὺδρίον ὼροσκοπεῖον”), was a hydraulic clock, which was suitably controlled, so that the level of the liquid was a linear function of time. It is considered the first feedback control system in history. Ctesebius disseminated his knowledge to his student, Philo of Byzantium (Φίλων ὸ Βυζάντιος) (ca. 280 B.C. – ca. 220 B.C.), who also lived in Alexandria. He created an oil lamp which was automatically controlled with a feedback control loop.

After three centuries, Ὴρων ο Ὰλεξανδρεὺς (Hero of Alexandria), created many machines, which were mainly used for the automatic control of the level of liquids.

Later, Drebbel (∼ 1600) and Papin (∼ 1700) created temperature and pressure controllers respectively. In 1769, Watt created the Flyball governor, which is considered the first industrial controller and it controlled the shaft rounds of a steam engine. Maxwell, in 1868 created the first theory of automatic control using differential equations.

Contemporary, process control flourished in many fields and especially contributed in the development of telecommunications, space technology and generally in industry.

2.2. First and second-order processes with time delay

Processes that possess the capacity to store, mass or energy and act as a buffer between inflowing and outflowing streams are be modeled as a first-order system. First-order systems include the most common processes in chemical plants and generally in practice. For instance, the dynamic response of tanks that have the capacity to store liquids or gases can be modeled as a first-order process.

On the other hand, **second** or even higher-order dynamics can arise from

- Processes that consist of two or more first-order systems, which are connected in series.
- Fluid or mechanical solid components of a process that possess inertia or are subjected to acceleration.
- The controller, which is used, may exhibit second or higher-order dynamics. Therefore, the installed controller introduces additional dynamics to the process itself and it will result to second or higher-order behavior.

If a change takes place in one of the input variables, its effect is never instantaneously observed in the output variables. Thus, there is always a time interval during which no change is noticed on the outputs of the system. This time interval is called **time delay** or **dead time**.

It is clear that the most common processes in industry and especially in chemical plants can be approximated as first and second-order processes. That was the main drive that led to the examination of processes in this thesis.

2.3. PID Controllers

The commercial controllers are usually electronic devices, which process electric signals, based on simple mathematical relations. Based on the deviation between the set point and the measurement value, three actions can be applied by them:

- **P**roportional action: Its actuating output produces a value which is proportional to the current error value.
- **I**ntegral action: Its actuating signal eliminates the offset and accelerates the movement of the process towards the setpoint value.
- **D**erivative action: With the presence of the derivative action, the controller can anticipate what the error will be in the immediate future.

In this work, the PID controller is examined. The output of the controller is given by:

$$\mathrm{K}(\mathrm{t}) \;=\; \mathrm{k_c e}(\mathrm{t}) \;+\; \frac{\mathrm{k_c}}{\tau_\mathrm{I}}\int \mathrm{e}(\mathrm{t}) \;+\; \mathrm{k_c}\tau_\mathrm{D}\frac{\mathrm{de}}{\mathrm{dt}} \;+\; \mathrm{K_s} \tag{2.1}$$

or

$$\mathrm{K}(\mathrm{s}) = \mathrm{k_c}\left(1 + \frac{1}{\tau_\mathrm{I}\mathrm{s}} + \tau_\mathrm{D}\mathrm{s}\right) \tag{2.2}$$

which is the parallel form.

A PID controller can also be created from a serial junction of a PI and a PD controller

$$\mathrm{K}(\mathrm{s}) = \mathrm{k_c'}\left(1 + \frac{1}{\tau_\mathrm{I}'\mathrm{s}}\right)\left(1 + \tau_\mathrm{D}'\mathrm{s}\right) \tag{2.3}$$

and is called the serial form of the PID controller.

In the past, there were cases that people thought that no difference existed between the parallel and the serial form. After the years, the different PID parametrizations were well-known. Nowadays, the parallel form is the most dominant and the most widely used and controllers in that form are going to be examined here.

2.4. SIMC method

The two-step systematic procedure proposed by Skogestad [2003] is a method to find sufficient settings of a PID controller.

The procedure consists of two steps:

1. Approximate a high-order model into a first or second order model with time delay using the Half-rule.

2. Derive the controller settings using the SIMC rules.

2.4.1. Model reduction

It is often possible that a process might have more complicated process dynamics. Skogestad [2003] proposed an approximation method which can reduce a high-order model with multiple time constants to a first or second order process with time delay.

Assume a given transfer function model, which has the following form,

$$\mathrm{g_0(s)} = \frac{\prod_\mathrm{j}\left(-\mathrm{T^{inv}_{j0}s} + 1)\right)}{\prod_\mathrm{i}\left(\tau_{\mathrm{i}_0}\mathrm{s} + 1)\right)}\mathrm{e}^{\vartheta_0}\mathrm{s} \tag{2.4}$$

where the lags τ_{i_0} are sorted according to their magnitude and $-\mathrm{T^{inv}_{j0}} < 0$.

Half-rule

The largest neglected (denominator) time constant (lag) is distributed evenly to the effective delay (ϑ) and the smallest retained time constant (τ_1 or τ_2).

Following that rule, one can approximate a model of the form in 2.4 into a **first-order**

process with time delay as:

$$\tau_1 = \tau_{10} + \frac{\tau_{20}}{2} \tag{2.5}$$

$$\vartheta = \vartheta_0 + \frac{\tau_{20}}{2} + \sum_{i\geq 3} \tau_{i0} + \sum_{j} T^{\mathrm{inv}}_{j0} + \frac{h}{2} \tag{2.6}$$

or as a **second-order process with time delay** as:

$$\tau_1 = \tau_{10} \tag{2.7}$$

$$\tau_2 = \tau_{20} + \frac{\tau_{30}}{2} \tag{2.8}$$

$$\vartheta = \vartheta_0 + \frac{\tau_{30}}{2} + \sum_{i\geq 4} \tau_{i0} + \sum_{j} T^{\mathrm{inv}}_{j0} + \frac{h}{2} \tag{2.9}$$

When it comes to positive numerator time constants T_0, they can be canceled against a "neighboring" lag time constant τ_0 by the following rules:

$$\frac{T_0 s+1}{\tau_0 s+1} \approx \begin{cases} T_0/\tau_0 & \text{for } T_0 \geq \tau_0 \geq \tau_c & \text{(Rule T1)} \\ T_0/\tau_c & \text{for } T_0 \geq \tau_c \geq \tau_0 & \text{(Rule T1a)} \\ 1 & \text{for } \tau_c \geq T_0 \geq \tau_0 & \text{(Rule T1b)} \\ T_0/\tau_0 & \text{for } \tau_0 \geq T_0 \geq 5\tau_c & \text{(Rule T2)} \\ \frac{(\tilde{\tau}_0/\tau_0)}{(\tilde{\tau}_0 - T_0)s+1} & \text{for } \tilde{\tau} \overset{\mathrm{def}}{=} \min(\tau_0, 5\tau_c) \geq T_0 & \text{(Rule T3)} \end{cases}$$

Keep in mind, that only the desired closed-loop time constant, τ_c, is the only tuning parameter. Usually, τ_c is selected equal to the effective time delay.

2.4.2. SIMC rules

After the first- or second-order plus delay model is obtained the SIMC rules can be, the model-based controller parameters can be derived:

For a **first-order model**

$$g_1(s) = \frac{k}{(\tau_1 s + 1)} e^{-\vartheta s} \tag{2.10}$$

the SIMC method results in a PI controller settings

$$k_c' = \frac{1}{k}\frac{\tau_1}{\tau_c + \vartheta} = \frac{1}{k'}\frac{1}{\tau_c + \vartheta} \tag{2.11}$$

$$\tau_I' = \min\left\{\tau_1, 4\left(\tau_c + \vartheta\right)\right\} \tag{2.12}$$

In 2013, Grimholt and Skogestad, updated the SIMC rules so that a PID controller can result from a first-order with time delay process. More specifically, the suggested that:

$$\tau_D' = \vartheta/3 \tag{2.13}$$

For a **second-order model**

$$g_1(s) = \frac{1}{\left(\tau_1 s + 1\right)\left(\tau_2 s + 1\right)}e^{\vartheta s} \tag{2.14}$$

the SIMC method results in a PID controller settings in cascade form

$$k_c' = \frac{1}{k}\frac{\tau_1}{\tau_c + \vartheta} = \frac{1}{k'}\frac{1}{\tau_c + \vartheta} \tag{2.15}$$

$$\tau_I' = \min\left\{\tau_1, 4\left(\tau_c + \vartheta\right)\right\} \tag{2.16}$$

$$\tau_D' = \tau_2 \tag{2.17}$$

However, the *I-SIMC* was proposed, which stands for Improved SIMC and sets $\tau_D' = \vartheta/3$. In fact, it is mentioned that when the value of τ_2 is very small, the I-SIMC rule should be used.

The settings for the second order plus time delay process apply to a PID controller in the series form. The corresponding settings for a parallel PID controller are

$$k_c = k_c' f \quad \tau_I = \tau_I' f \quad \tau_D = \tau_D'/f \tag{2.18}$$

Normally, the tuning parameter (τ_c) is chosen after obtaining the effective time delay. Therefore, one might have to guess the value of τ_c and iterate in order to approximate the desired model.

2.5. K-SIMC method

Similarly to Skogestad, Lee et al. [2013] proposed an alternative method to find the PID controller settings. More specifically, they re-examined the SIMC rules and introduced some improvements compared to the SIMC. Nevertheless, it should be mentioned that the mindset behind the *K-SIMC* method is very close to Skogestad's.

Suppose a model of the following form, where τ_{i0} 's are in descending order

$$g_0(s) = \frac{\prod_j \left(-T_{j0}^{inv}s + 1)\right)}{\prod_i \left(\tau_{i_0}s + 1)\right)} e^{\vartheta_0 s} \tag{2.19}$$

According to the K-SIMC method the approximations positive numerator time constants are

$$\frac{T_0 s+1}{\tau_0 s+1} \approx \begin{cases} k = \frac{\sqrt{1+(T_0/\lambda)^2}}{\sqrt{1+(\tau_0/\lambda)^2}} & \text{for } T_0 \geq \tau_0 \text{ or } \tau_0 \geq T_0 \geq \lambda \quad \text{(Rule 3a)} \\ \frac{1+T_0{}^2/(2\lambda)^2}{1+\tau_0 T_0/(2\lambda)^2} \frac{1}{\frac{\tau_0 - T_0}{1+\tau_0 T_0/(2\lambda)^2}s+1} & \text{for } 5\lambda \geq \tau_0 \geq T_0 \quad \text{(Rule 3b)} \\ \text{apply Rules 3a and 3b to } \frac{5\lambda s+1}{\tau_0 s+1}\frac{T_0 s+1}{5\lambda s+1} & \text{for } \tau_0 \geq 5\lambda \geq T_0 \quad \text{(Rule 3c)} \end{cases}$$

After the elimination of positive numerator time constants the high-order model can be reduced to a *first-order process with time delay*

$$g_1(s) = \frac{k}{(\tau_1 s + 1)} e^{-\vartheta s} \tag{2.20}$$

$$\tau_1 = \tau_{10} + 0.5\tau_{20}^2/\tau_{10} \tag{2.21}$$

$$\vartheta = \vartheta_0 + \tau_{20}\left(1 - 0.5\tau_{20}/\tau_{10}\right) + \sum_{i\geq 3} \tau_{i0} + \sum_j T_{j0}^{inv} + \frac{h}{2} \tag{2.22}$$

or as a *second-order process with time delay* as:

$$\tau_1 = \tau_{10} \tag{2.23}$$

$$\tau_2 = \tau_{20} + 0.5\tau_{30}^2/\tau_{20} \tag{2.24}$$

$$\vartheta = \vartheta_0 + \tau_{20}\left(1 - 0.5\tau_{30}/\tau_{20}\right) + \sum_{i\geq 4}\tau_{i0} + \sum_{j} T_{j0}^{inv} + \frac{h}{2} \tag{2.25}$$

For a **first-order model**

$$g_1(s) = \frac{k}{(\tau s + 1)} e^{-\vartheta s} \tag{2.26}$$

The resulting parameters for a **PID** controller are

$$k_c' = \frac{\tau}{\lambda + \vartheta} \tag{2.27}$$

$$\tau_I' = \min\left\{\tau, 5\left(\lambda + \vartheta\right)\right\} \tag{2.28}$$

$$\tau_D' = \max\left\{\left(\vartheta - \lambda\right)/2, 0\right\} \tag{2.29}$$

For a **second-order model**

$$g_1(s) = \frac{k}{\left(\tau^2 s^2 + 2\zeta\tau s + 1\right)} e^{-\vartheta s} \tag{2.30}$$

the SIMC method results in a **PID** controller settings in parallel form

$$k_c = \frac{\tau^2/\tau_D}{k\left(\lambda + \vartheta\right)} \tag{2.31}$$

$$\tau_I = \min(2\zeta\tau,\ 7.07\lambda) \tag{2.32}$$

$$\tau_D = \min\left(\frac{\tau}{2\zeta},\ 3.54\lambda\right) \tag{2.33}$$

Please bear in mind, that equations 2.27-2.28 refer to the cascade form of the controller while equations 2.30-2.32 refer to the ideal form. The controller parameters for the FOPTD using the K-SIMC method will be suitably calculated for the ideal form of the PID controller by using equations 2.18.

2.6. Syrcos & Kookos tuning method

In 2005, Syrcos and Kookos [2005] proposed a general mathematical programming formulation in order to obtain customized PID controller settings. The solved a mixed integer nonlinear optimization problem by minimizing an objective cost function which is dependent on the output and the control variables. After the examination of several cases, they concluded that when $\vartheta \in [2, 5]$ a FOPTD can be tuned using the following rules:

$$\mathrm{k_c k_p} = 0.31 + 0.6\left(\frac{1}{\vartheta'}\right) \tag{2.34}$$

$$\frac{\tau_\mathrm{I}}{\tau_1} = 0.777 + 0.45\vartheta' \tag{2.35}$$

$$\frac{\tau_\mathrm{D}}{\tau_1} = 0.44 - 0.56\left(\frac{1}{\vartheta'}\right)^{2.2} \tag{2.36}$$

where

$$\vartheta' = \frac{\vartheta}{\tau_1} \tag{2.37}$$

The equations above refer to the ideal form of the PID controller. The results of that specific approach were very promising for the values of the time delay considered. It has to be noted, that the tuning rules are not universal, since the suggested settings are restricted for a specific region of time delays. Nevertheless, many processes exhibit such behavior. Therefore, a case study was considered and compared with the optimal and the SIMC.

Chapter 3

Pareto Optimization

The term "Controller Optimization" refers to the selection of the optimal parameters of the controller, which ensures that the operation of the controlled system is the optimal. Many methods have been proposed in order to find good PID controller settings. However, some of these approaches give insufficient results or might be very time consuming, such as the trial and error method. The necessity to satisfy more complex criteria leads to optimization-based design.

In the past, efforts have been made to find controllers through optimization. Namely, not only Hall [1943] but also Hazebroek and Van der Waerden [1950] proposed that the minimization of the integrated square error (ISE) can lead to optimal controller settings. Later, in 1958, Balchen, was the first to introduce the minimization of the integrated absolute error (IAE), which included the performance and robustness trade-off. In the last two decades, influential has been the work of Shinskey [1990], Panagopoulos et al. [2002], Skogestad (2003), Åström and Hägglund [2012], where IAE is adopted as the main evaluation criterion in terms of performance in PID design. This approach is used also in this work. Foss [2012] and Holene [2013] have investigated both first and second-order processes and compared the SIMC tunings with the optimal. However, the gradients were calculated numerically and convergence problems were stated.

Grimholt and Skogestad [2016] managed to overcome problems with the estimation of the gradients by developing analytical expressions for them. As a result, they achieved much better results in terms of convergence for a wider range of conditions and models. Their approach is going to be used in this work.

3.1. Feedback system

The linear feedback system depicted in Figure 2.1 is considered. The system consists of two components, the process (plant) G(s) and the controller K(s). Moreover, the system is influenced by three external signals, the reference $\mathrm{y_s}$, the plant input disturbance $\mathrm{d_u}$ and the plant output disturbance $\mathrm{d_y}$. The plant output disturbance can be also considered as a setpoint change. Control is based on the measured signal y, while the measurements are corrupted by the output step disturbance. The process is also influenced by the controller through the control variable u. The difference between the measured variable and the reference point is denoted by, e. Summing up:

- Inputs: control variable u, process input disturbance $\mathrm{d_u}$, process output disturbance $\mathrm{d_y}$.
- Output: Measured signal y.

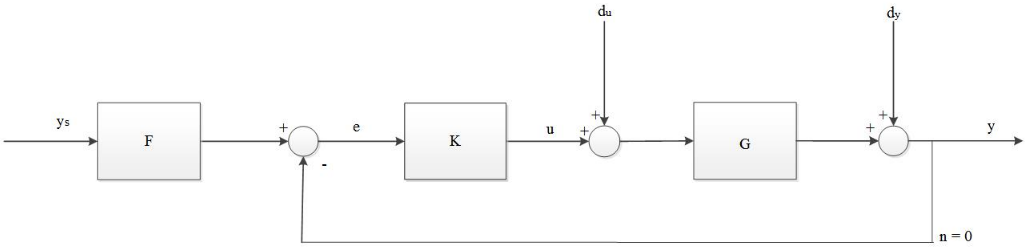


Figure 3.1: Block diagram of the closed loop system, with controller K(s) and plant G(s).

By determining the Laplace transforms, the following functions are obtained from the block diagram in Figure 3.1, which describe the relations between the different variables of the system

$$\mathrm{S(s)} = \frac{1}{1+\mathrm{G(s)K(s)}}, \qquad \mathrm{T(s)} = 1 - \mathrm{S(s)} \tag{3.1}$$

$$\mathrm{GS(s)} = \mathrm{G(s)S(s)}, \qquad \mathrm{KS(s)} = \mathrm{K(s)S(s)} \tag{3.2}$$

In this case, F is considered equal to 1 and the control actions are based from the feedback from the error only. The (pure) system can be described by those four transfer functions above, which are nicknamed as, *The Gang of Four.* In addition to that, S(s) can be specified as the sensitivity function and T(s) as the complementary sensitivity function.

The effect on the control error and plant input is,

$$-\mathrm{e} = \mathrm{y} - \mathrm{y_s} = \mathrm{S(s)d_y} + \mathrm{GS(s)d_u} - \mathrm{T(s)n} \tag{3.3}$$

$$-\mathrm{u} = \mathrm{KS(s)d_y} + \mathrm{T(s)d_u} + \mathrm{KS(s)n} \tag{3.4}$$

When it comes to the controller type, the PID controller in parallel form is used,

$$K_{PID}(s;p) = k_p + k_i/s + k_d s = k_c = \left(1 + \frac{1}{\tau_I s} + \tau_D s\right), \tag{3.5}$$

$$\text{where} \qquad p = \begin{pmatrix} k_p & k_i & k_d \end{pmatrix}^T \tag{3.6}$$

The proportional, integral and derivative gain are represented as $k_p = k_c, k_i = k_c/\tau_I$ and $k_d = k_c\tau_D$ respectively, while τ_I and τ_D are the integral and derivative times.

Grimholt and Skogestad [2016] proposed a method to optimal PID controllers for known process models. This approach, gives Pareto-Optimal results in terms of controller performance and robustness. Pareto-optimality can be applied in many different academic fields, such as economics, engineering and life sciences. It refers to multiobjective problems, and means that no further improvement can be achieved in objective 1 without sacrificing objective 2.

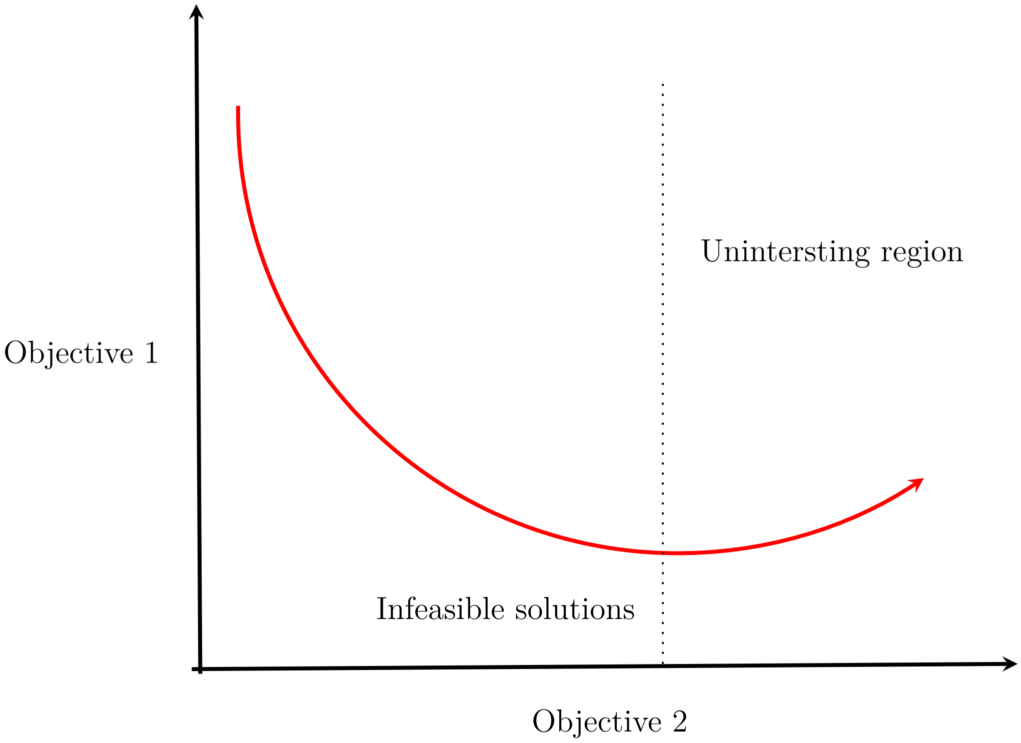


Figure 3.2: Two objectives Pareto optimization.

However, many issues have to be considered in the analysis and the design of control systems, which must satisfy some basic requirements

- Stability
- Ability to follow reference signals (performance)

- Reduction of effects of load disturbances (performance)
- Reduction of effects of measurement noise (performance)
- Reduction of effects of model uncertainties (robustness)

In this case, optimality is represented by:

- Output performance (Objective 1)
- Robustness and input usage (Objective 2)

3.1.1. Performance

The performance of the controller is specified by considering the integrated absolute error (IAE) when the system is subject to step disturbances.

$$\mathrm{IAE(p)} = \int_0^{t_f} |e(t;p)|\,dt \tag{3.7}$$

Both plant input and output step disturbances enter the system and a weighted cost function is chosen

$$J(p) = 0.5(\varphi_{dy}\,\mathrm{IAE}_{dy}(p) + \varphi_{du}\,\mathrm{IAE}_{du}(p)) \tag{3.8}$$

In equation (2) φ_{dy} and φ_{du} are the "weights" or the normalization factors. In general, the normalization method can be chosen on a subjective basis. Nevertheless, it is necessary to normalize the resulting IAE_{dy} and IAE_{du}, to to be able to to get a good balance of the two terms, compare them and evaluate which contributes more to the cost function.

In this case the normalization factors are selected, similar to Shinskey (1990), as the inverse of the optimal IAE values for reference controllers (e.g. PI, PID) tuned for a step change on the plant input (IAE^{o}_{dy}) and output (IAE^{o}_{du}) respectively.

It must be noted, that the IAE values calculated for the weighting are for optimal (e.g. PI, PID) controllers for plant input and output step disturbances respectively. Those controllers are required to have $M_S = M_T = 1.59$. This specific value $M_S = 1.59$ is the resulting M_S value for a Simple Internal Control (SIMC) tuned PI controller for $\tau_c = \vartheta$ of a first order process with time delay (FOPTD) with $\tau \leq 8\vartheta$. Therefore, two optimal controllers are used to obtain the two reference IAE values, whereas a single controller K is used to find the IAE_{du} and IAE_{dy} values for the desired controller.

$$\varphi_{du} = \frac{1}{\mathrm{IAE}^{o}_{du}} \quad \varphi_{dy} = \frac{1}{\mathrm{IAE}^{o}_{dy}} \tag{3.9}$$

3.1.2. Robustness

Robustness is quantified in terms of the largest sensitivity peak, $\mathrm{M_{ST}} = \max(\mathrm{M_S}, \mathrm{M_T})$ (Garpinger and Hägglund, 2008). Thus,

$$\mathrm{M_S} = \max_{\omega} |\mathrm{S}(j\omega)| = \|\mathrm{S}(j\omega)\|_{\infty} \tag{3.10}$$

$$\mathrm{M_T} = \max_{\omega} |\mathrm{T}(j\omega)| = \|\mathrm{T}(j\omega)\|_{\infty} \tag{3.11}$$

where $\|\cdot\|_{\infty}$ is the H_{∞} norm (maximum peak as a function of frequency).

The maximum sensitivity essentially gives the largest amplification of the disturbances. The maximum occurs at the frequency $\omega_{\mathrm{M_S}}$ or $\omega_{\mathrm{M_T}}$ respectively.

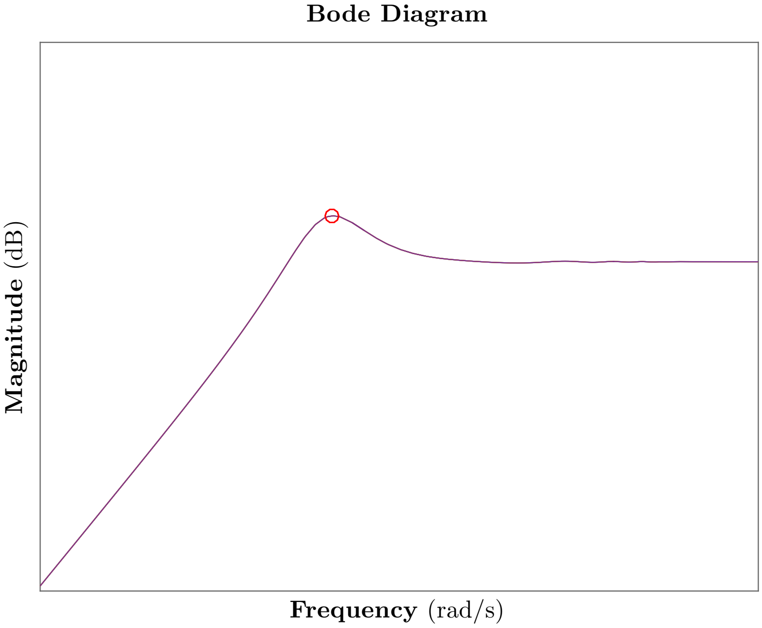


Figure 3.3: Peak of the sensitivity function, $\mathrm{M_S}$

The sensitivity peaks can also be related with the gain and phase margins by the following relations

$$\mathrm{GM} \geq \frac{\mathrm{M_S}}{\mathrm{M_S} - 1} \qquad \mathrm{GM} \geq 1 + \frac{1}{\mathrm{M_T}} \tag{3.12}$$

$$\mathrm{PM} \geq \frac{1}{\mathrm{M_S}} \qquad \mathrm{PM} \geq \frac{1}{\mathrm{M_T}} \tag{3.13}$$

In that thesis, as it was mentioned, the ideal form of the PID controller is used. The optimal controller can have several peaks or plateaux for the magnitude of the sensitivity function in the frequency domain $|\mathrm{S}(\mathrm{j}\omega)|$. Therefore, the optimizer may jump between peaks during iterations. To avoid this issue, which might lead to inaccuracies, multiple constraints are obtained by gridding the frequency response,

$$|\mathrm{S}\,(\mathrm{j}\omega)| \leq \mathrm{M}^{\mathrm{ub}} \qquad \text{for all } \omega \text{ in } \Omega, \tag{3.14}$$

where Ω is the set of selected frequency points. According to this approach, one inequality constraint is derived for each grid frequency.

3.2. Optimization problem

Summing up, the desired optimal controller can be found by solving the following optimization problem

$$\underbrace{\text{minimize}}_{\mathrm{p}} \quad \mathrm{J(p)} = 0.5(\varphi_{\mathrm{dy}}\,\mathrm{IAE}_{\mathrm{dy}}\,(\mathrm{p}) + \varphi_{\mathrm{du}}\,\mathrm{IAE}_{\mathrm{du}}\,(\mathrm{p})) \tag{3.15}$$

$$\text{subject to} \qquad \mathrm{c_S(p)} = |\mathrm{S}(\mathrm{j}\omega;\mathrm{p})| - \mathrm{M}_{\mathrm{S}}^{\mathrm{ub}} \leq 0 \quad \text{for all } \omega \text{ in } \Omega \tag{3.16}$$

$$\mathrm{c_T(p)} = |\mathrm{T}(\mathrm{j}\omega;\mathrm{p})| - \mathrm{M}_{\mathrm{T}}^{\mathrm{u\,b}} \leq 0 \quad \text{for all } \omega \text{ in } \Omega \tag{3.17}$$

$\mathrm{M}_{\mathrm{S}}^{\mathrm{ub}}$ and $\mathrm{M}_{\mathrm{T}}^{\mathrm{u\,b}}$ represent the upper bound on S(s) and T(s) respectively and in most cases it is selected as $\mathrm{M}_{\mathrm{S}}^{\mathrm{ub}} = \mathrm{M}_{\mathrm{T}}^{\mathrm{u\,b}} = \mathrm{M}_{\mathrm{ST}}$. If there is a trade-off between performance and robustness, at least one of the constraints will be active.

3.3. Gradients

Consider a function f(p) which is dependent of $\mathrm{n_p}$ parameters.

$$\nabla_{\mathrm{p}}\,\mathrm{f(p)} = \left(\frac{\partial \mathrm{f}}{\partial \mathrm{p}_1} \quad \frac{\partial \mathrm{f}}{\partial \mathrm{p}_2} \quad ... \quad \frac{\partial \mathrm{f}}{\partial \mathrm{p}_{\mathrm{n_p}}} \right)^{\mathrm{T}} \tag{3.18}$$

Each partial derivative or *sensitivity* of the function f can be approximated with the assistance of forward finite differences

$$\frac{\partial \mathrm{f}}{\partial \mathrm{p_i}} \approx \frac{\mathrm{f}(\mathrm{p_i} + \Delta \mathrm{p_i}) - \mathrm{f}(\mathrm{p_i}))}{\Delta \mathrm{p_i}} \tag{3.19}$$

Please keep in mind, that $\nabla \equiv \nabla_{\mathrm{p}}$.

3.3.1. Cost function gradient

The gradient of the cost function J(p) has the following expression

$$\nabla \mathrm{J(p)} = 0.5(\varphi_{\mathrm{dy}}\, \nabla \mathrm{IAE}_{\mathrm{dy}}(\mathrm{p}) + \varphi_{\mathrm{du}}\, \nabla \mathrm{IAE}_{\mathrm{du}}(\mathrm{p})) \tag{3.20}$$

Assuming that $|\mathrm{e(t);p)}|$ and sign $\{\mathrm{e(t)}\}\ \nabla \mathrm{e(t)}$ are continuous, the sensitivities of IAE can be expressed as

$$\nabla \mathrm{IAE}_{\mathrm{dy}} = \int_0^{\mathrm{t_f}} \mathrm{sign}\left\{\mathrm{e_{dy}}\right\} \nabla \mathrm{e_{dy}(t)dt} \tag{3.21}$$

$$\nabla \mathrm{IAE}_{\mathrm{du}} = \int_0^{\mathrm{t_f}} \mathrm{sign}\left\{\mathrm{e_{du}}\right\} \nabla \mathrm{e_{du}(t)dt} \tag{3.22}$$

In addition to that, $\mathrm{e_{dy}}$ and $\mathrm{e_{du}}$ are evaluated as

$$\mathrm{e_{dy}(s) = S(s)d_y} \qquad \text{for process output disturbances} \tag{3.23}$$

$$\mathrm{e_{du}(s) = GS(s)d_u} \qquad \text{for process input disturbances} \tag{3.24}$$

and the sensitivities of the errors are

$$\nabla \mathrm{e_{dy}} = -\mathrm{GS(s)S(s)}\nabla \mathrm{K(s)d_y} \tag{3.25}$$

$$\nabla \mathrm{e_{du}} = -\mathrm{GS(s)GS(s)}\nabla \mathrm{K(s)d_u} \tag{3.26}$$

As it is shown in equations 3.23 and 3.24 the sensitivities of the errors can be expressed as a function of the sensitivity of the controller.

Furthermore, the ideal PID controller parameter sensitivities have the following form

$$\nabla \mathrm{K_{PID}(s)} = (1 \quad 1/\mathrm{s} \quad \mathrm{s})^{\mathrm{T}} \tag{3.27}$$

Here, it should be noted, that $|\mathrm{e(t);p)}|$ and sign $\{\mathrm{e(t)}\}\ \nabla \mathrm{e(t)}$ might be not continuous in the whole time span. However, very small steps are used at the numerical integration method which is used for the simulation and therefore this inaccuracy can be considered negligible. Also, this method is only valid for processes with proper transfer functions. If the transfer functions are not proper, a filter is necessary to be added to make either the controller or the gradient transfer function proper.

3.3.2. Constraint gradients

The expressions for the robustness constraints are

$$\nabla c_S (j\omega; p) = \nabla |S (j\omega)| = \frac{1}{|S (j\omega)|} \{S^{*} (j\omega) \nabla S (j\omega)\} \quad \text{for all } \omega \text{ in } \Omega \tag{3.28}$$

$$\nabla c_T (j\omega; p) = \nabla |T (j\omega)| = \frac{1}{|T (j\omega)|} \{T^{*} (j\omega) \nabla T (j\omega)\} \quad \text{for all } \omega \text{ in } \Omega \tag{3.29}$$

Elaborating a little more and using the chain rule we obtain

$$\nabla S (j\omega) = -GS (j\omega) S (j\omega) \nabla K (j\omega) \tag{3.30}$$

$$\nabla T (j\omega) = \nabla (1 - S (j\omega)) = -\nabla S (j\omega) \tag{3.31}$$

Summing up, a Pareto optimal controller can be found by following the procedure described in Table 3.1.Of course, one can adjust the algorithm in terms of his present needs. For instance, it might be the case that the aim is to examine the performance for different values of the time delay. Then, what should be determined, are the values of the time delay while the robustness remains at the desired point.

Table 3.1: Creating Pareto-optimal curves.

Step	**Description**
1:	Set the process model whose optimal controller is to be found
2:	Decide the controller parametrization and determine its derivative
3:	Specify the time and frequency interval of the calculations
4:	for $\mathbf{M_S = 1.59}$ find the optimal PID which minimizes **J** for input disturbances
5:	for $\mathbf{M_S = 1.59}$ find the optimal PID which minimizes **J** for output disturbances
6:	Set the weights of the objective function, **J**
7:	Determine the desired values of of $\mathbf{M_S}$
8:	Find the optimal PID which minimizes **J** for input and output disturbances
9:	Create Performance vs. Robustness plots

[1] The asterisk (*) is used to indicate the complex conjugate.

3.4. Simulations

For all the calculations, the time was selected accordingly to each process. However, the simulations must be done until a stead-state is reached. Usually 10^4 steps are more than enough. Also, the frequency "scan" interval was selected from $0.01/\ \vartheta$ to $100/\ \vartheta$. As the interval of the sensitivity function peak, it was selected as $1.3 < \mathrm{M_S} < 2$, which corresponds to the most common case.

In the results chapter which follow, SIMC refers to the tuning method developed by Skogestad [2003], K-SIMC to the revisited SIMC rules which were developed by Lee et. al. [2013] and I-SIMC to the improved SIMC which uses $\vartheta/3$ as the derivative time. Furthermore, Syrcos & Kookos refers to the methodology developed by Syrcos & Kookos [2005] and finally Toolbox refers to the MATLAB Tuning Toolbox.

When it comes to the form of the PID controller used in the simulations, the **parallel** form was selected. It has to be mentioned that in order to find the optimal controller parameters the exact model was used. In addition, that is the case for the sensitivity peaks too. As initial conditions, the SIMC or the K-SIMC settings for each specific controller were used.

For all the results and the calculations MATLAB and SIMULINK were used. For the optimization an active set method was chosen. Not only the SIMULINK model is included in the Appendix B, but also all the codes utilized for the results. All the calculations were conducted on an Intel Core i7 @ 1.80GHz / 8.00GB RAM computer and the the computational time was of the order of seconds.

Chapter 4

Results and Discussion

4.1. First-order processes

Pareto optimal curves were created for first-order processes with time delay (FOPTD). Different cases were examined. The process time constant varied while the time delay remained constant and equal to 1. The optimal controllers were compared with those calculated using the SIMC and K-SIMC tunings. No approximations on the models were used. It has to be mentioned that, the K-SIMC method gives the PID controller parameters for FOPTD only in serial form. Therefore, the same rule as in 2.18 is used in order to find the controller parameters for the parallel form. The processes examined are summarized in Table 4.1, which also presents the tuning parameters for $M_S = 1.59$.

Table 4.1: Comparison of PID controller settings with different tuning methods for FOPTD with $M_S = 1.59$.

Case	Process	J	J_{SIMC}	$J_{K\text{–}SIMC}$	IAE^o_{du}	IAE^o_{dy}									
							Optimal			SIMC			K-SIMC		
							k_c	k_i	k_d	k_c	k_i	k_d	k_c	k_i	k_d
1a	$G(s) = \frac{e^{-s}}{(0.1s+1)}$	1.000	1.074	1.445	1.506	1.504	0.293	0.666	0.037	0.269	0.621	0.021	0.050	0.500	0.000
2a	$G(s) = \frac{e^{-s}}{(0.5s+1)}$	1.000	1.093	1.461	1.494	1.454	0.568	0.767	0.184	0.517	0.621	0.103	0.250	0.500	0.000
3a	$G(s) = \frac{e^{-s}}{(s+1)}$	1.013	1.085	1.416	1.557	1.419	0.856	0.756	0.294	0.829	0.622	0.207	0.500	0.500	0.000
4a	$G(s) = \frac{e^{-s}}{(1.5s+1)}$	1.035	1.115	1.441	1.583	1.327	1.157	0.776	0.413	1.139	0.621	0.311	0.75	0.500	0.000
5a	$G(s) = \frac{e^{-s}}{(2s+1)}$	1.021	1.123	1.445	1.726	1.227	1.461	0.824	0.540	1.450	0.622	0.415	1.000	0.500	0.000
6a	$G(s) = \frac{e^{-s}}{(10s+1)}$	1.304	1.717	2.188	1.608	0.494	6.411	2.040	2.599	6.411	0.931	2.032	4.649	0.808	0.000

First, it is of great interest to see the behavior of the complementary sensitivity function peak, M_T, when it is compared to the sensitivity function peak, M_S. A large value of either M_S or M_T are indications of poor performance as well as poor robustness. Usually,

$\mathrm{M_S}$ is required to be smaller than 2 and $\mathrm{M_T}$ to be smaller than 1.3.

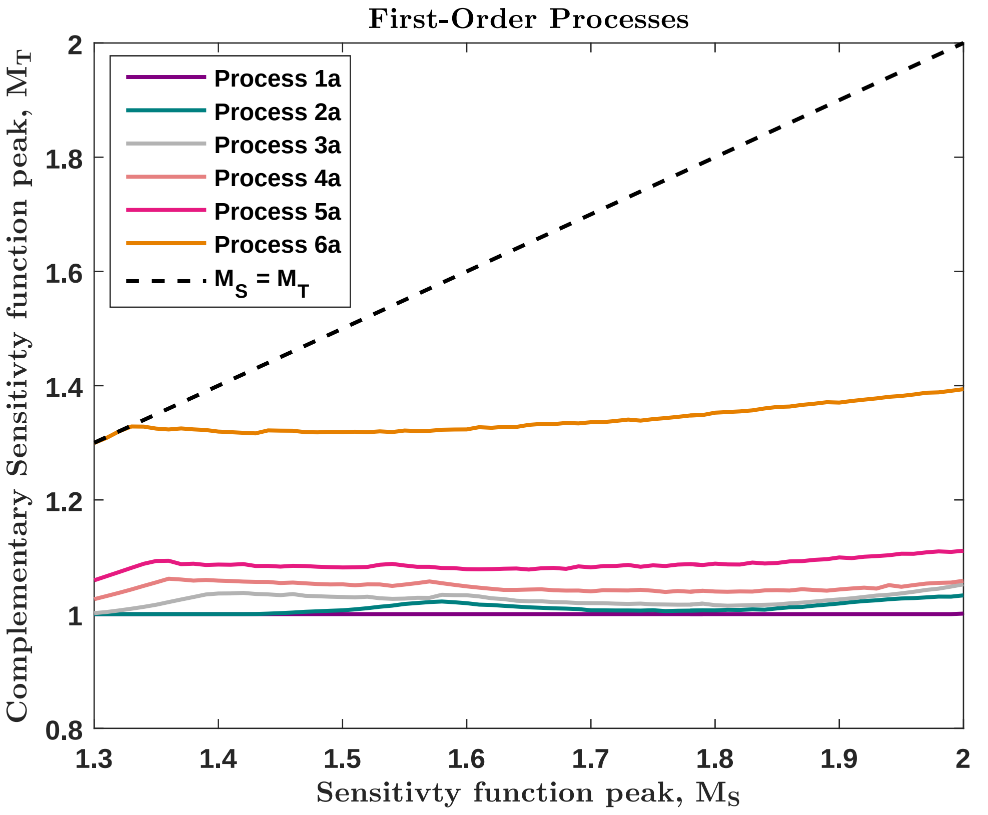


Figure 4.1: Complementary sensitivity function and sensitivity function peak comparison for the FOPTD examined.

It is known, that the $\mathrm{M_S}$ is bound

$$\mathrm{M_T} \leq \mathrm{M_S} + 1$$

and generally it is smaller than the $\mathrm{M_S}$. That is proven to be also the case for all the processes investigated here. However, for constant $\mathrm{M_S}$, as the process time constant increases, $\mathrm{M_T}$ also increases which is also a sign that as the time constant increases the sensitivity to noise is amplified.

The graphs which follow compare the performance of the controllers, derived from those 3 methodologies, for different $\mathrm{M_S}$ values. Clearly, there can be no tuning methodology or approach, which can achieve better performance than the optimal. That is apparent in Figures 4.2−4.7 for all the cases examined. The question here is how far from optimality the other two PID tunings are.

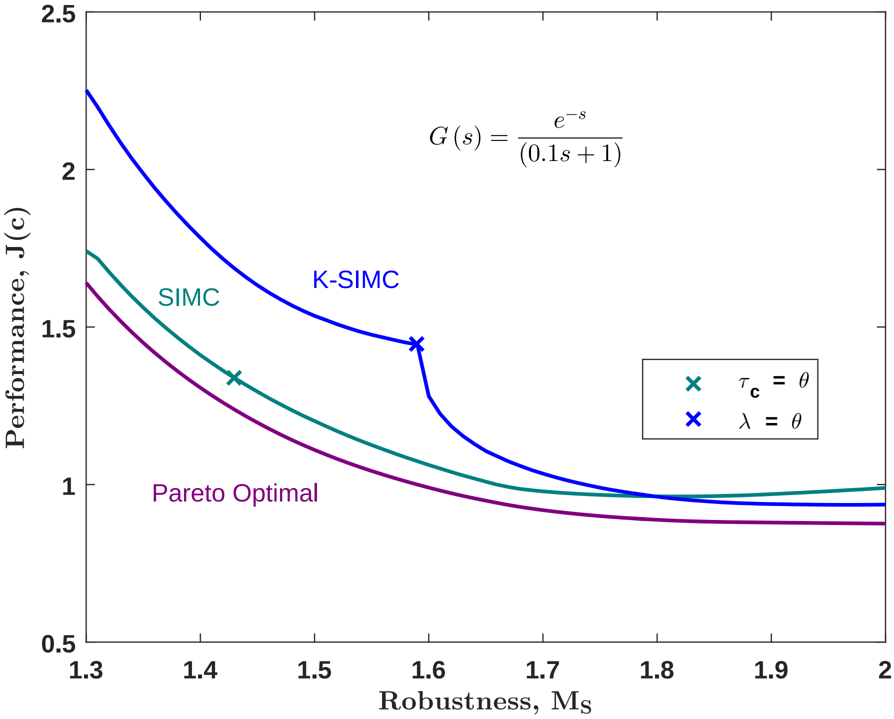


Figure 4.2: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-\mathrm{s}}}{(0.1\mathrm{s}+1)}$.

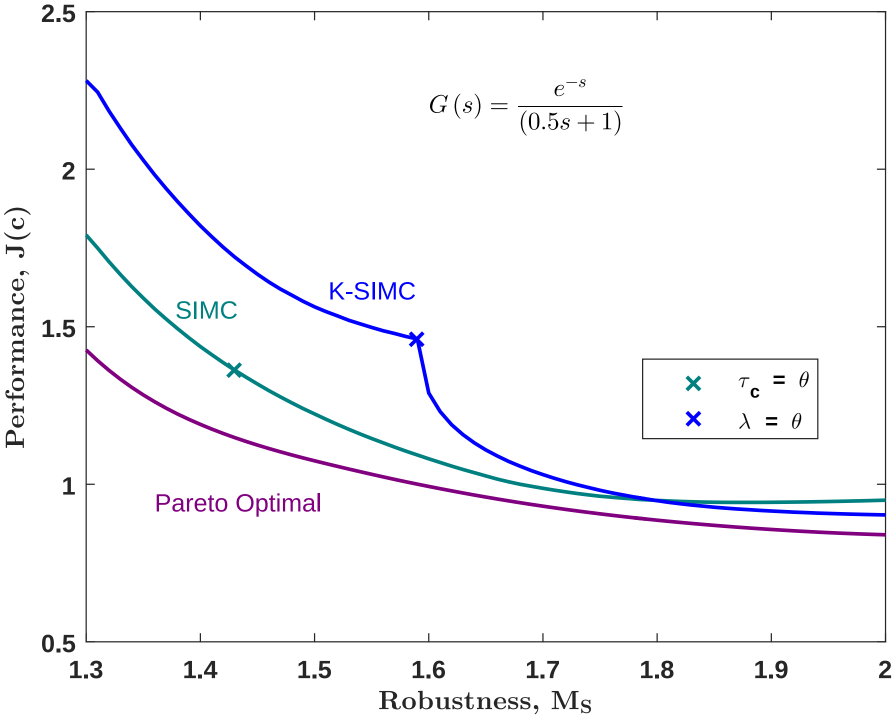


Figure 4.3: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-\mathrm{s}}}{(0.5\mathrm{s}+1)}$.

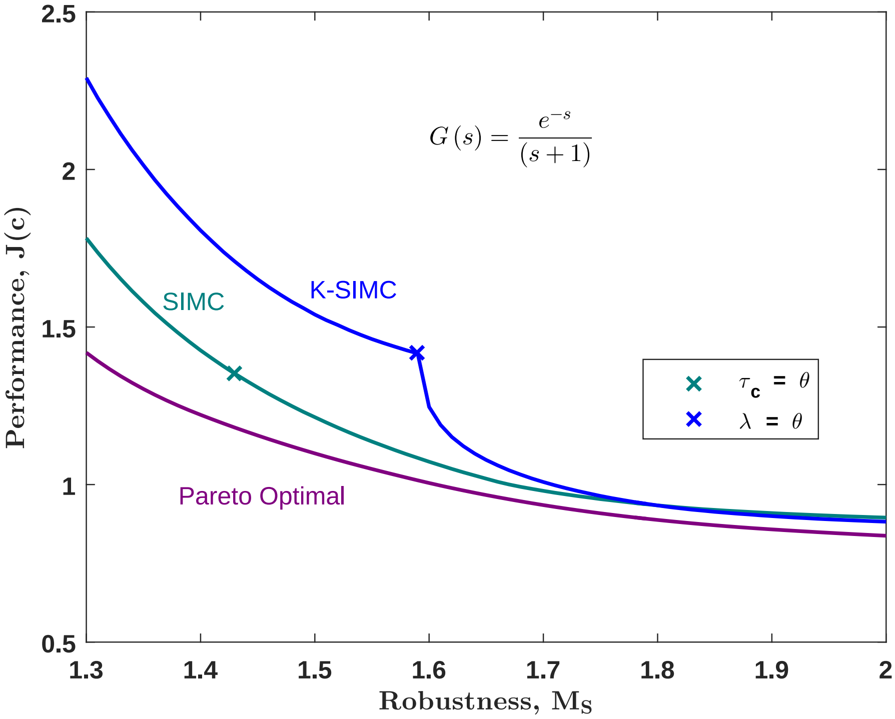


Figure 4.4: Cost function J for difference constraint values, using ideal PID controllers for the process $G(s) = \frac{e^{-s}}{(s+1)}$

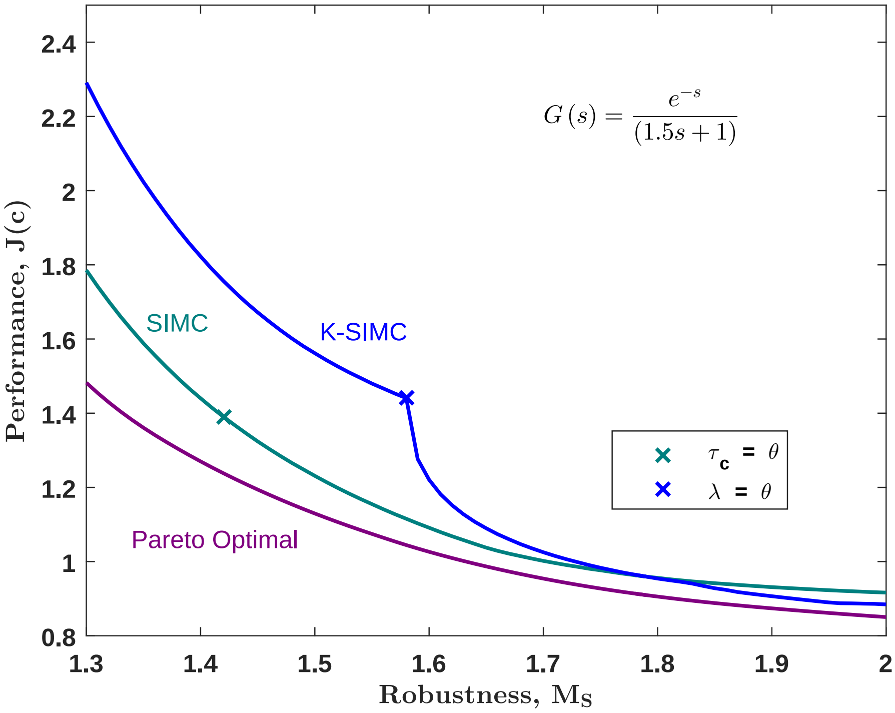


Figure 4.5: Cost function J for difference constraint values, using ideal PID controllers for the process $G(s) = \frac{e^{-s}}{(1.5s+1)}$

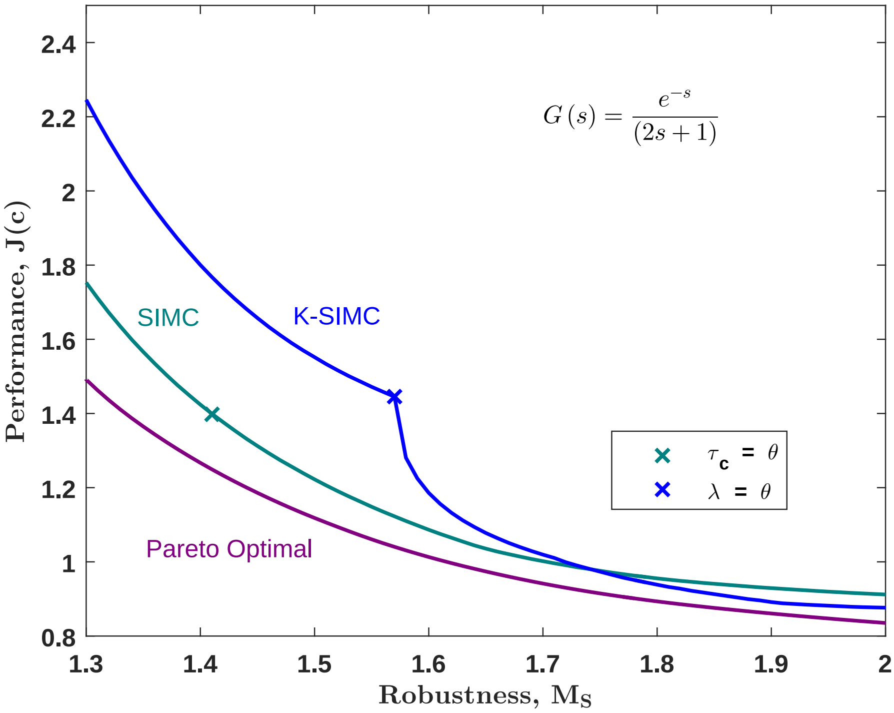


Figure 4.6: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-\mathrm{s}}}{(2\mathrm{s}+1)}$

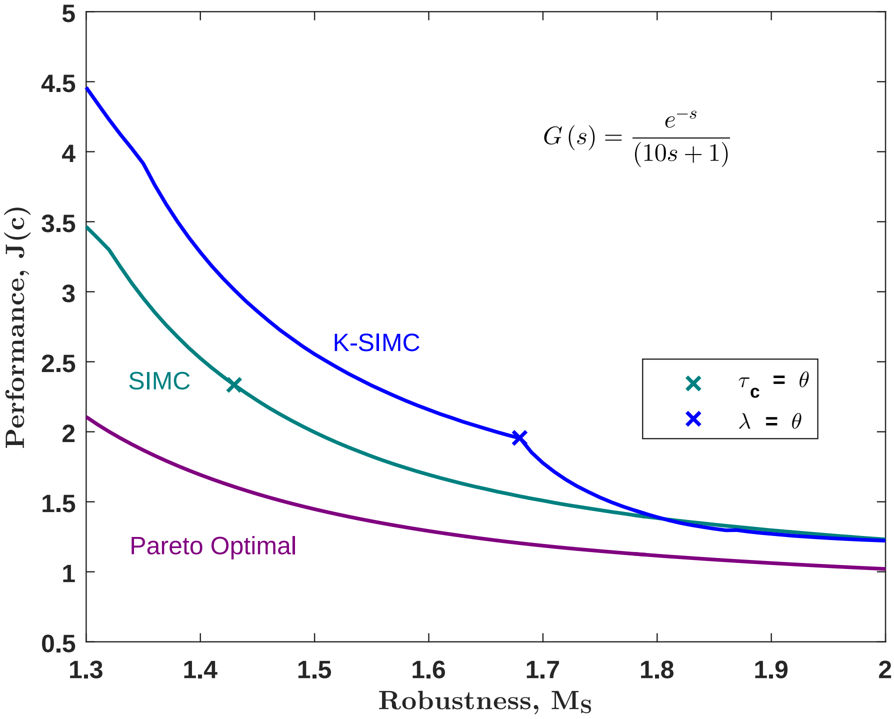


Figure 4.7: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-\mathrm{s}}}{(10\mathrm{s}+1)}$

The divergence of the other two tuning methods is greater for small values of the sensitivity peak. Nonetheless, as M_S increases the performance of the SIMC and K-SIMC controllers is greatly improved. A prime example is the case 3a, where $\tau_1/\vartheta = 1$, where the performance of both of the other tuning methodologies give close to optimal results. Nevertheless, for the last case examined in this section, both the methodologies fail to give similar to optimal behavior and especially the K-SIMC tuning rules give almost twice as worse results when compared to the optimal.

Furthermore, the SIMC provides much better performance than the K-SIMC tuning for the majority of the values of M_S which were investigated. More specifically, for the smaller values of M_S, or for the more *robust* controllers, the SIMC highly overpeforms the K-SIMC. From the performance vs. robustness graphs it is apparent that for approximately $M_S < 1.75$ the SIMC gives more satisfactory results. On the other hand, for less robust controllers the K-SIMC tunings can improve the performance by almost 5% in some cases, than the SIMC. However, this advantage of the K-SIMC controllers is almost nonexistent for some cases, such as the last case where $\tau_1/\vartheta = 10$, where no discrepancy is present.

Moreover, something that needs to be highlighted is the interesting behavior of the K-SIMC plots. For the more robust regions, the K-SIMC controllers completely underperform, even when they are compared to the SIMC controllers. That is a consequence, of the tuning rule of the derivative time which essentially states that **only** when λ is smaller than ϑ, the resulting controller is a PID. Even though the PI controllers are more robust to disturbances and process noise, there is a lack of performance mainly due to the lack of derivative action. That is plainly depicted in the graphs. The turning point, where the K-SIMC tunings give a PID instead of a PI controller is also evident. After that specific point, a dramatic increase of performance is displayed, and the K-SIMC controllers start to "catch up". Eventually, the K-SIMC controllers surpass the SIMC. That is also an indication that as robustness decreases the value of τ_c also declines. That should be expected, since the tuning rules suggest high τ_c values for more robust controllers in the trade-off of performance. Finally, it has to be mentioned that for the K-SIMC controllers used in case 6a, a filter was added for $M_S > 1.85$. The average deviation of the performance of the cases examined here summarized in Table 4.2

Table 4.2: Performance comparison of the different FOPTD examined.

Case	Process	PO-SIMC Variance (Avg %)	PO-K-SIMC Variance (Avg %)
1a	$G(s) = \frac{e^{-s}}{(0.1s+1)}$	8.7	25.7
2a	$G(s) = \frac{e^{-s}}{(0.5s+1)}$	13.2	30.5
3a	$G(s) = \frac{e^{-s}}{(s+1)}$	10.5	27.3
4a	$G(s) = \frac{e^{-s}}{(1.5s+1)}$	10.7	27.1
5a	$G(s) = \frac{e^{-s}}{(2s+1)}$	12.4	27.0
6a	$G(s) = \frac{e^{-s}}{(10s+1)}$	48.5	84.9

When someone looks at the overall performance, the SIMC tuning rules should be the way to go for most of robustness region. Table 4.1 shows that the controller proportional gain between the PO and the SIMC is almost identical while the integral time of the SIMC controllers is slightly larger. In contrast, the PO give higher values of the derivative time. The K-SIMC controllers are proven to be marginally more efficient for high values of the sensitivity peak, but a choice of a K-SIMC controller can not be justified when they also give more than twice worse performance in the whole interval examined. Please bear in mind, for for very small λ values, a setpoint filter might be added. On the contrary, the SIMC and the other tuning rules do not associate the controller settings with the filtering. They suggest that first the controller performance should be seen and then decide about the filtering.

4.2. Second-order processes

In this section, Pareto optimal controllers are found for second-order processes with time delay. Please note, that there is no need for approximations. The PO controllers are compared with the ones found using SIMC , and K-SIMC for different values of M_S. Seven different cases were examined, which present various combinations of, τ_1/τ_2 and τ_1/ϑ. More specifically, the cases examined here are summarized in Table 4.3.

Table 4.3: Comparison of PID controller settings with different tuning methods for SOPTD for $M_S = 1.59$.

Case	Process	J	J_{SIMC}	J_{K-SIMC}	J_{I-SIMC}	IAE^o_{du}	IAE^o_{dy}									
								Optimal			SIMC			K-SIMC		
								k_c	k_i	k_d	k_c	k_i	k_d	k_c	k_i	k_d
1b	$G(s) = \frac{e^{-0.7s}}{(s+1)(0.4s+1)}$	1.023	1.111	1.111	-	1.376	1.248	1.093	0.819	0.453	1.000	0.714	0.286	1.000	0.714	0.286
2b	$G(s) = \frac{e^{-0.7s}}{(s+1)(0.1s+1)}$	1.025	1.272	1.272	1.099	1.228	1.071	1.041	0.969	0.298	0.786	0.715	0.072	0.786	0.715	0.072
3b	$G(s) = \frac{e^{-0.7s}}{(s+1)(0.7s+1)}$	1.032	1.078	1.078	-	1.417	1.288	1.254	0.773	0.646	1.214	0.714	0.500	1.214	0.714	0.500
4b	$G(s) = \frac{e^{-0.7s}}{(1.3s+1)(0.4s+1)}$	1.041	1.131	1.131	-	1.385	1.199	1.297	0.833	0.570	1.214	0.714	0.371	1.214	0.714	0.371
5b	$G(s) = \frac{e^{-0.7s}}{(0.7s+1)(0.4s+1)}$	1.010	1.110	1.110	-	1.356	1.281	0.895	0.821	0.343	0.786	0.715	0.200	0.786	0.715	0.200
6b	$G(s) = \frac{e^{-2s}}{(s+1)(0.5s+1)}$	1.001	1.258	1.258	1.169	3.445	3.379	0.443	0.250	0.164	0.375	0.250	0.125	0.375	0.250	0.125
7b	$G(s) = \frac{e^{-0.9s}}{(1.5s+1)(1.2s+1)}$	1.049	1.090	1.090	-	1.849	1.626	1.508	0.595	1.184	1.499	0.555	1.000	1.499	0.555	1.000

Again, it is interesting to examine the behavior of the complementary sensitivity function. Once more, M_T does not surpass the value of M_S for all the cases examined here. M_T remains almost constant for the majority of the robustness region presented. Nevertheless, for larger values of M_S, there is an apparent rise of the M_T.

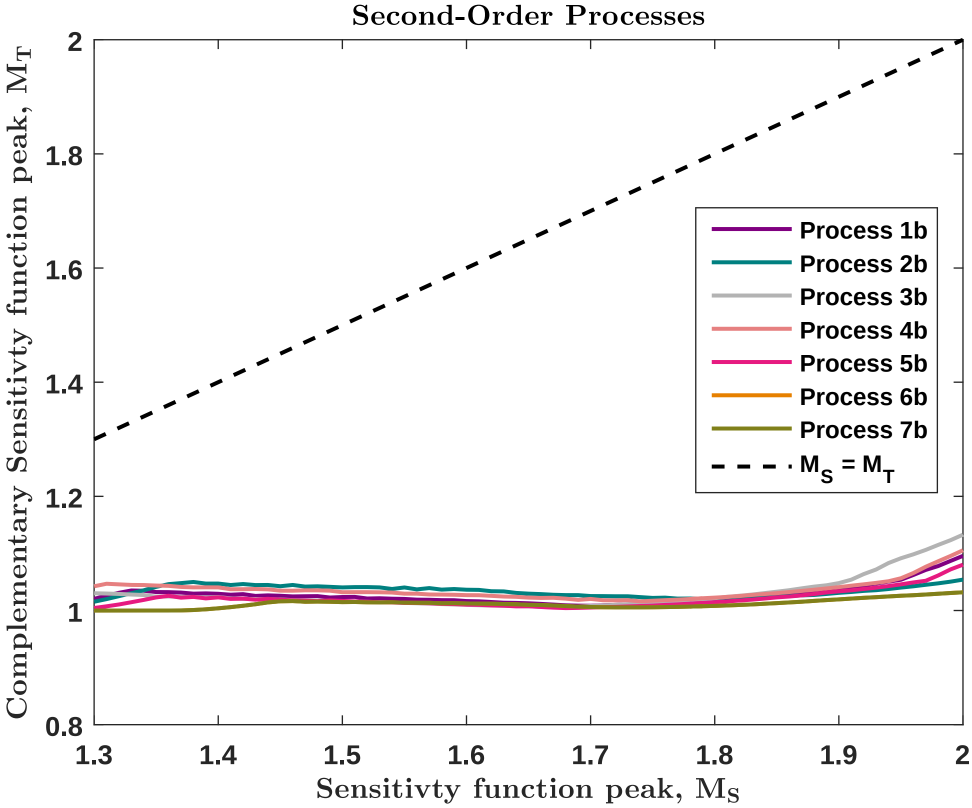


Figure 4.8: Complementary sensitivity function and sensitivity function peak comparison for the FOPTD examined.

The performance vs. robustness graphs follow. Figures 4.9 up to 4.15 show clearly that for larger values of the sensitivity peak the SIMC and the K-SIMC controllers provide a closer to optimal performance. Nevertheless, what is impressive, is that the tuning rules of the SIMC and the K-SIMC give almost the same controllers. That is clearly observed in Table 4.3, where the tuning parameters are the same up to the third decimal.

More specifically, this had to do more on the K-SIMC rules for the τ_D and τ_I. For almost every K-SIMC controller in the the second-order performance vs. robustness figures, the value of λ led to the use of:

$$\tau_I = 2\zeta\tau \quad \text{and} \quad \tau_D = \frac{\tau}{2\zeta}$$

Those rules give the same tunings as the SIMC rules when they are used in the ideal form. The proportional gain of the controller is dependent on the derivative time. Therefore, all three controller parameters are almost the same. In the M_S region of interest, λ is still large enough that the alternative rules for the calculation of the integral and derivative time for the K-SIMC method are not used.

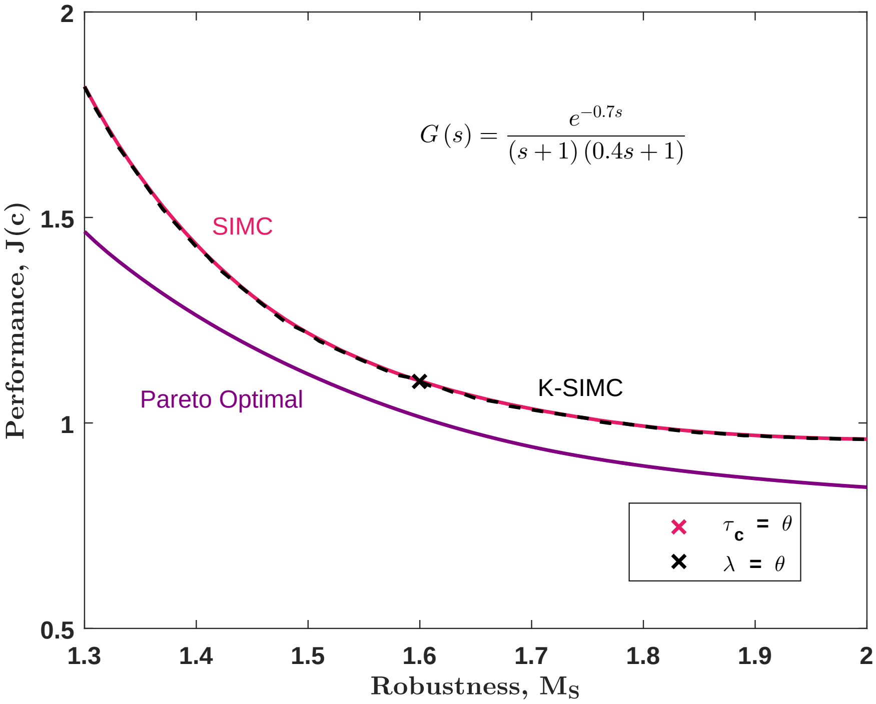


Figure 4.9: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.7\mathrm{s}}}{(\mathrm{s}+1)(0.4\mathrm{s}+1)}$.

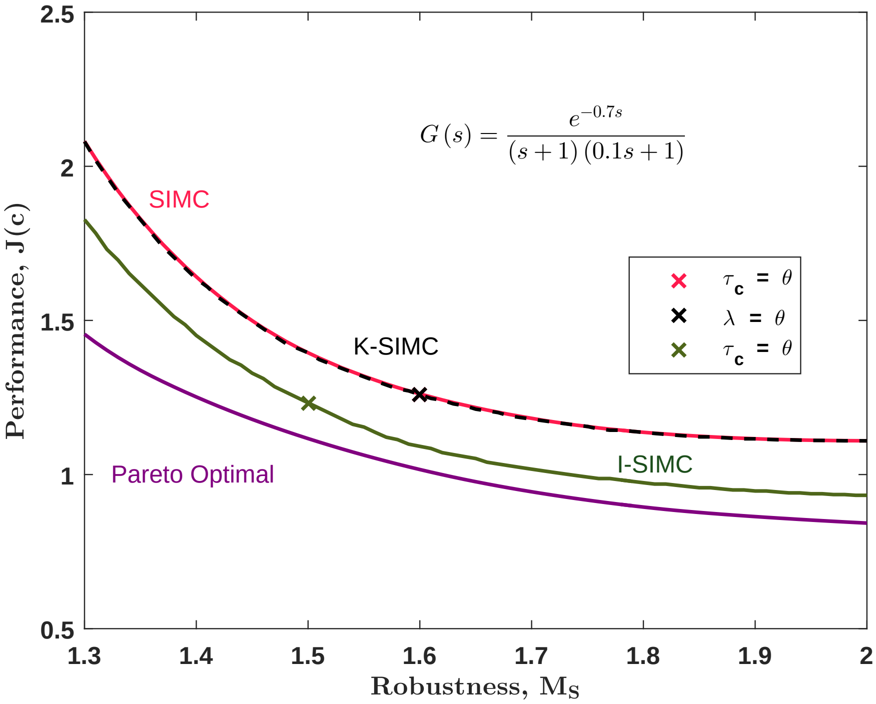


Figure 4.10: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.7\mathrm{s}}}{(\mathrm{s}+1)(0.1\mathrm{s}+1)}$.

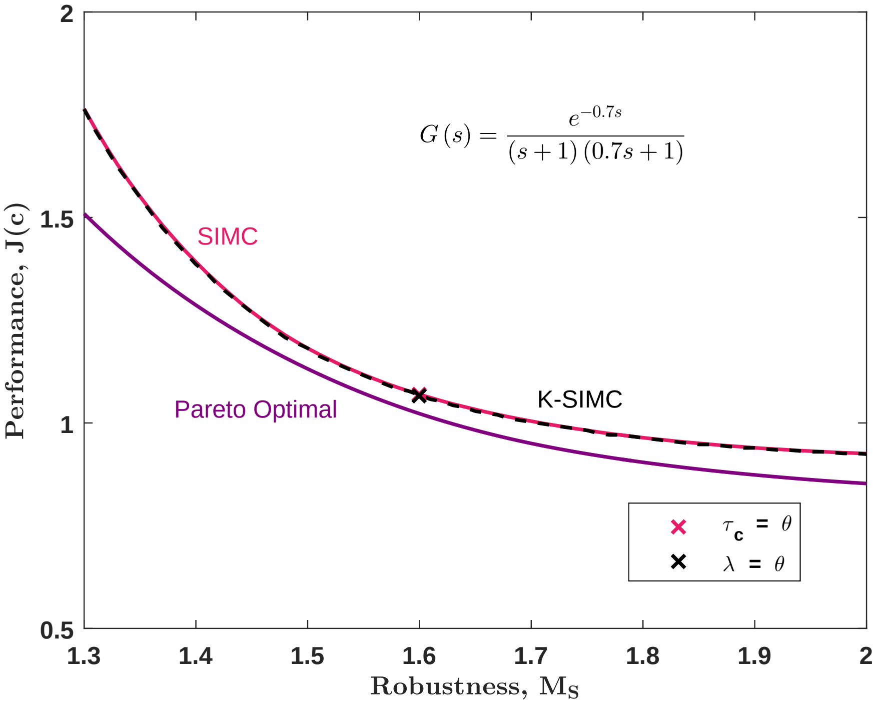


Figure 4.11: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.7\mathrm{s}}}{(\mathrm{s}+1)(0.7\mathrm{s}+1)}$.

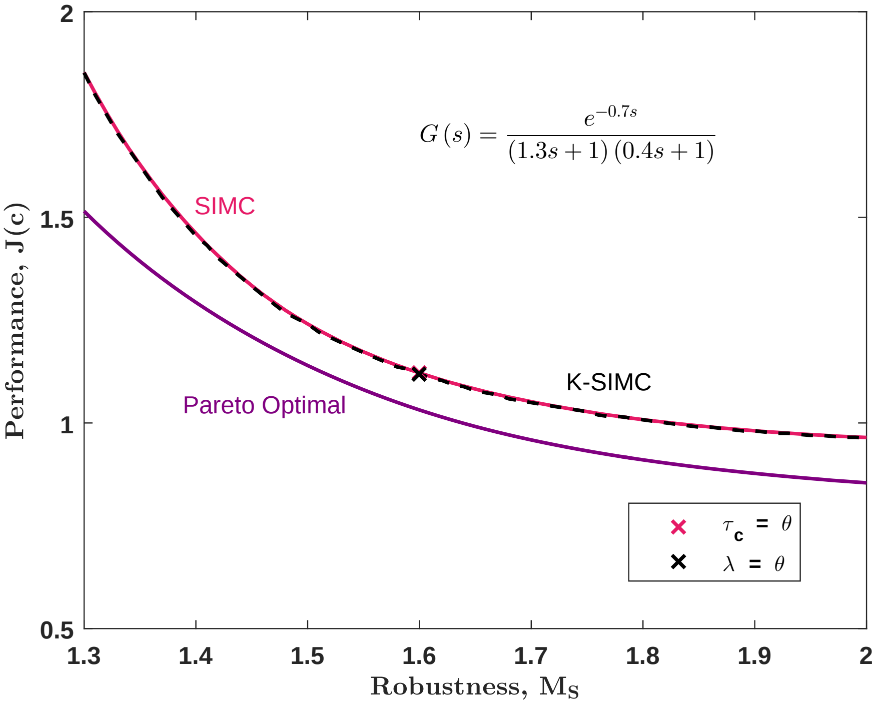


Figure 4.12: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.7\mathrm{s}}}{(1.3\mathrm{s}+1)(0.4\mathrm{s}+1)}$.

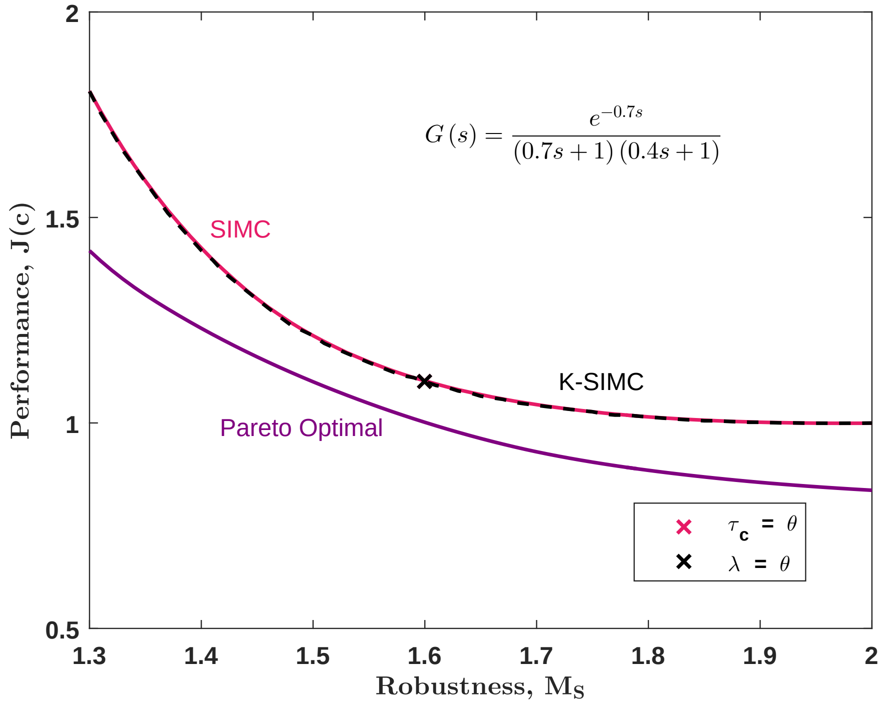


Figure 4.13: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.7\mathrm{s}}}{(0.7\mathrm{s}+1)(0.4\mathrm{s}+1)}$.

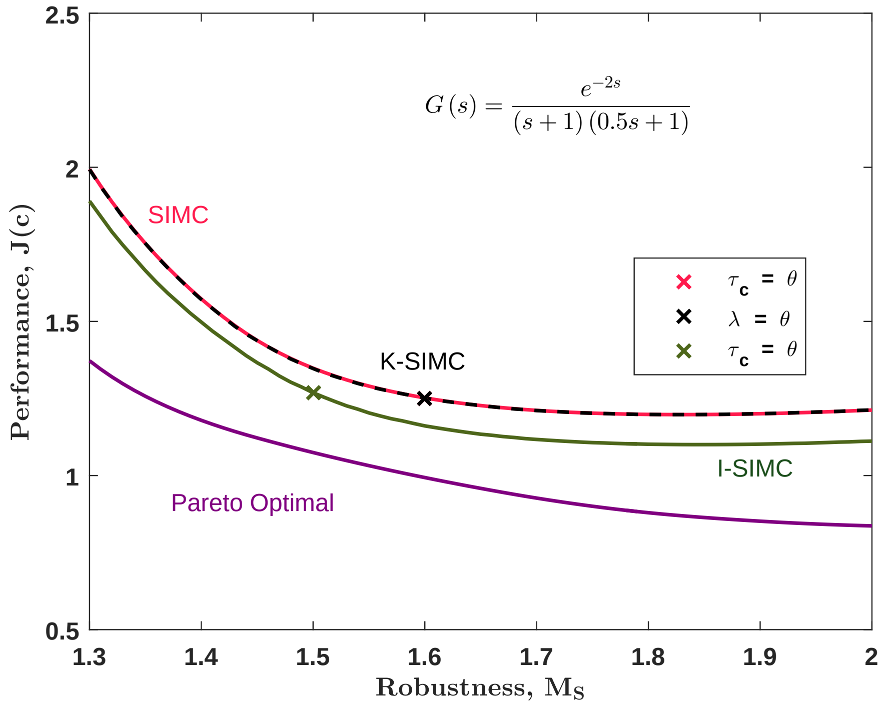


Figure 4.14: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-2\mathrm{s}}}{(\mathrm{s}+1)(0.5\mathrm{s}+1)}$.

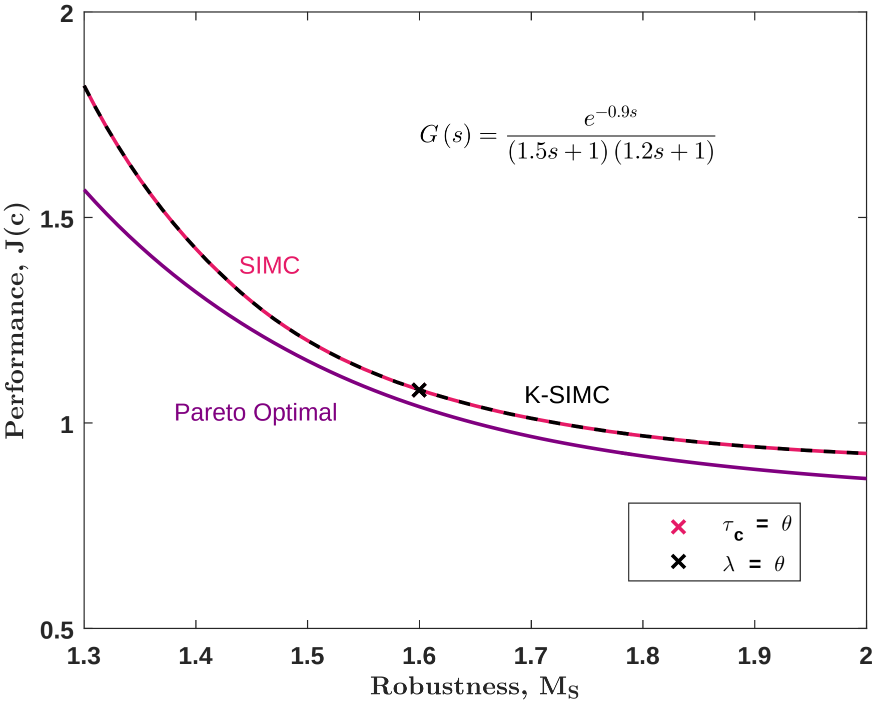


Figure 4.15: Cost function J for difference constraint values, using ideal PID controllers for the process $G(s) = \frac{e^{-0.9s}}{(1.5s+1)(1.2s+1)}$.

Looking more thoroughly in Table 4.4, the K-SIMC tuning settings cause a slim improvement in terms of performance for most of the cases, when they are compared to the SIMC. However, that difference could be considered negligible. The PO controllers exhibit almost similar k_c with the other two tuning approaches. Alike the FOPTD, the SIMC and the K-SIMC tunings give larger values of the integral time, while the opposite happens with the derivative time, where the optimal settings consist of larger τ_D.

As far as the *optimality* of the tunings investigated is concerned, both tuning methodologies present an inferior divergence from the optimal controllers, when compared to the FOPTD. Notable are the cases 3b and 7b, where less than 8% difference is observed from the performance of the PO controllers. These are the **only** cases, where τ_2 is smaller than ϑ. On the other hand, when time delay is significantly larger than the time constant τ_2. Then, the improved, the I-SIMC tuning comes into play. The I-SIMC settings propose an alternative for the derivative time. Instead of using, $\tau_D = \tau_2$, the derivative time is selected equal to $\vartheta/3$. A great example which showcases the betterment of the performance is case 2. There the I-SIMC controller provides almost 20 % superior results. Another instance, where the I-SIMC is extremely useful is case 6b, where almost 10 % improvement is observed. Table 4.4 presents the average deviation of the different methods, for the processes investigated in this section.

Table 4.4: Performance comparison of the different SOPTD examined.

Case	Process	PO-SIMC Variance (Avg %)	PO-K-SIMC Variance (Avg %)	PO-I-SIMC Variance (Avg %)
1b	$\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.7\mathrm{s}}}{(\mathrm{s}+1)(0.4\mathrm{s}+1)}$	12.6	12.3	-
2b	$\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.7\mathrm{s}}}{(\mathrm{s}+1)(0.1\mathrm{s}+1)}$	29.7	29.5	12.2
3b	$\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.7\mathrm{s}}}{(\mathrm{s}+1)(0.7\mathrm{s}+1)}$	7.7	7.4	-
4b	$\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.7\mathrm{s}}}{(1.3\mathrm{s}+1)(0.4\mathrm{s}+1)}$	12.4	12.0	-
5b	$\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.7\mathrm{s}}}{(0.7\mathrm{s}+1)(0.4\mathrm{s}+1)}$	15.2	15.0	-
6b	$\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-2\mathrm{s}}}{(\mathrm{s}+1)(0.5\mathrm{s}+1)}$	33.8	33.8	24.8
7b	$\mathrm{G}(\mathrm{s}) = \frac{\mathrm{e}^{-0.9\mathrm{s}}}{(1.5\mathrm{s}+1)(1.2\mathrm{s}+1)}$	6.5	6.5	-

4.3. Model reduction of higher-order models

Higher order models are introduced. The models are approximated both as a FOPTD and a SOPTD using the Half-rule and the K-SIMC model reduction rules. The objective of this section is to decide which approximations give controllers, which are closer to the optimal performance-wise.

First, the following process is examined

$$\mathrm{G}(\mathrm{s}) = \frac{(-0.5\mathrm{s}+1)(-0.1\mathrm{s}+1)}{(5\mathrm{s}+1)(3\mathrm{s}+1)(\mathrm{s}+1)(0.5\mathrm{s}+1)}\,\mathrm{e}^{-\mathrm{s}} \tag{4.1}$$

The model has only negative numerator time constants. Using the Half-Rule, the process can be approximated as

$$\mathrm{G}'(\mathrm{s}) = \frac{1}{6.5\mathrm{s}+1}\,\mathrm{e}^{-4.6\mathrm{s}} \tag{4.2}$$

as a FOPTD or as a SOPTD:

$$\mathrm{G}'(\mathrm{s}) = \frac{1}{(5\mathrm{s}+1)(3.5\mathrm{s}+1)}\,\mathrm{e}^{-2.6\mathrm{s}} \tag{4.3}$$

and by using the K-SIMC approximations the FOPTD process, which is derived is

$$\mathrm{G}'(\mathrm{s}) = \frac{1}{5.9\mathrm{s}+1}\,\mathrm{e}^{-5.2\mathrm{s}} \tag{4.4}$$

and the SOPTD is as follows

$$\mathrm{G}'(\mathrm{s}) = \frac{1}{(5\mathrm{s}+1)(3.167\mathrm{s}+1)}\, \mathrm{e}^{-2.933\mathrm{s}} \tag{4.5}$$

Figure 4.16 exhibits the performance of the PID settings derived from the different approximations for the desired values of $\mathrm{M_S}$. Please, bear in mind that the tunings were applied to the **exact** model.

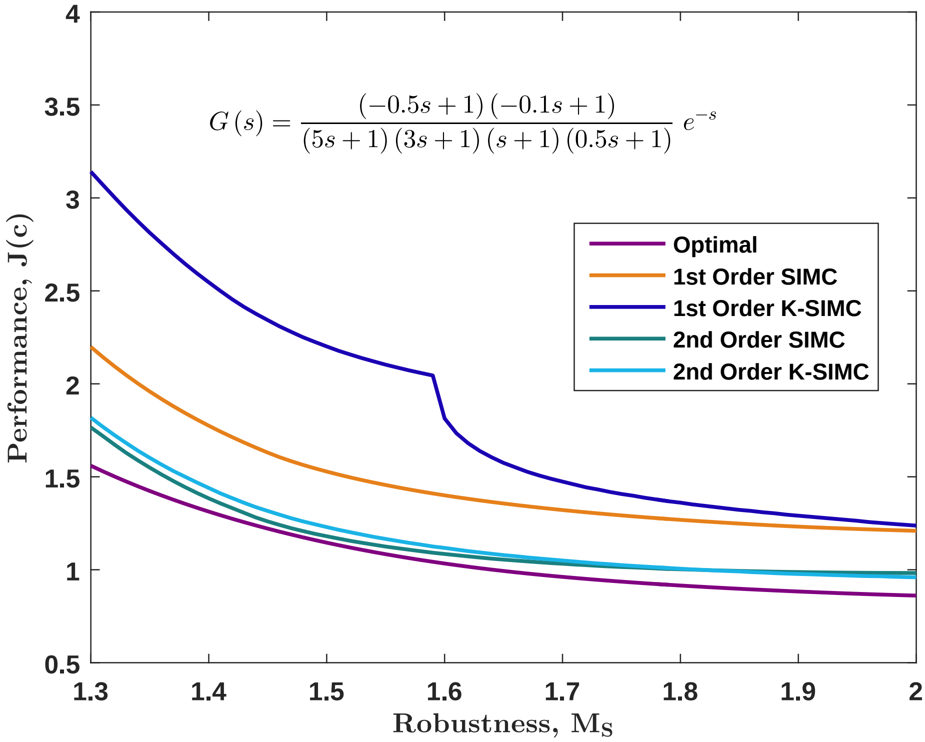


Figure 4.16: Cost function J for difference constraint values, using ideal PID controllers for the process $\mathrm{G}(\mathrm{s}) = \frac{(-0.5\mathrm{s}+1)(-0.1\mathrm{s}+1)}{(5\mathrm{s}+1)(3\mathrm{s}+1)(\mathrm{s}+1)(0.5\mathrm{s}+1)}\, \mathrm{e}^{-\mathrm{s}}$. First and second-order approximations are presented.

Looking more closely at the various approximations (Equations 4.2-4.5), two observations can be made. As far as the FOPTD are concerned, the mindset behind the approximation rules is similar. Nonetheless, equation shows that Skogestad preferred to add more on the time constant instead of the time delay, while Lee et al. preferred to add more on the time delay instead of the time constant. In addition to that, the Half-rule proposes a larger τ_2 than time delay while the K-SIMC approximations suggest the opposite.

The lack of the derivative term for the first-order K-SIMC approximation is obvious. Overall the K-SIMC showcases an almost 80 % deviation from the optimal performance, while the first-order "half-ruled" tunings achieve superior results performance-wise. Therefore, Skogestad's approach should be considered more efficient.

On the other hand, the second-order approximations behave significantly closer to optimality. Although, the "half-ruled" approximations perform better, none of them presents a deviation larger than 11 % from the optimal. All in all, the less approximations are made, of course, the more accurate the process model is. Hence, closer to optimal controller settings are obtained.

Next, a process model with only positive numerator time constants is investigated

$$\mathrm{G}(\mathrm{s}) = \frac{(20\mathrm{s}+1)(14\mathrm{s}+1)}{(42\mathrm{s}+1)(5\mathrm{s}+1)(2\mathrm{s}+1)(\mathrm{s}+1)}\,\mathrm{e}^{-\mathrm{s}} \tag{4.6}$$

Now, the approximation is a function of τ_c. Therefore the model can not be determined beforehand. In such circumstances the procedure which is to be done is:

For a given τ_c

1. Approximate the correct model
2. Find the PID controller
3. Find the corresponding M_S

This procedure is more computationally expensive since both the model and the M_S should be evaluated. Again, the performance is compared for the different tuning methods.

Similarly to the previous process, the exact process which was approximated as second-order processes give better controllers compared to the first-order ones. Due to the fact that the resulting τ_c values are relatively small, Rule T2 is used for the fraction $(42\mathrm{s}+1)/(20\mathrm{s}+1)$ while Rule T1 is used the fraction $(14\mathrm{s}+1)/(5\mathrm{s}+1)$. The K-SIMC approximations exhibit a non-smooth behavior towards the larger M_S values owing to an approximation rule change. Rule 3a is almost explicitly used for the approximations while for larger M_S Rule 3c is suitably selected.

Once again, a critical point for the first-order K-SIMC controller behavior is the point where $\lambda = \vartheta$ and instead of a PI controller, the rules suggest a PID controller. Moreover, while the first-order SIMC controllers result in more advantageous tunings compared to

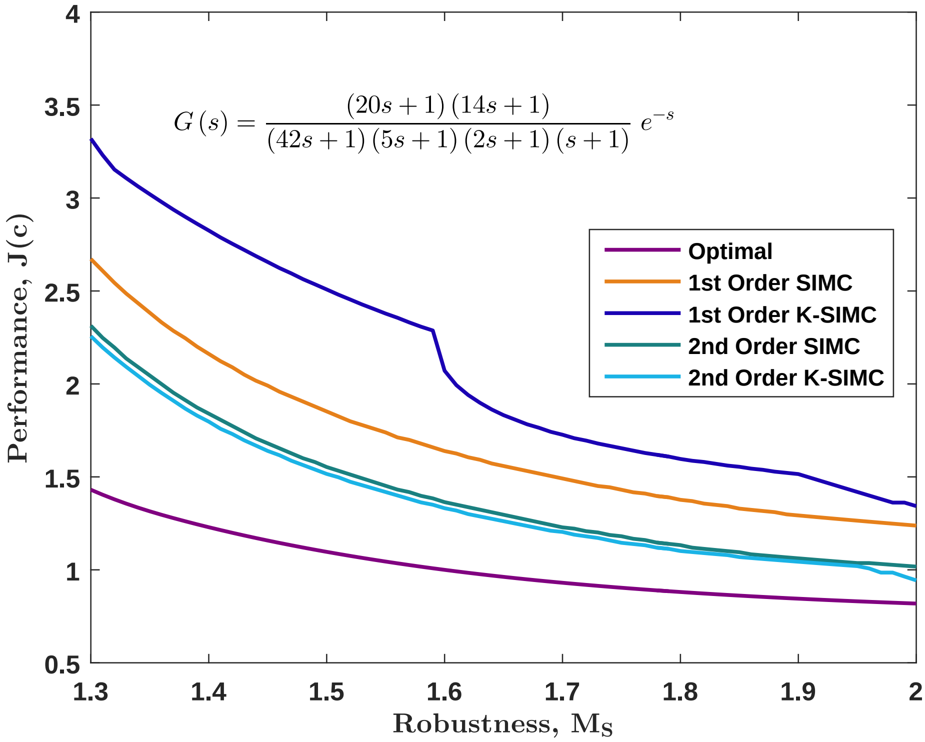


Figure 4.17: Cost function J for difference constraint values, using ideal PID controllers for the process $G(s) = \frac{(20s+1)(14s+1)}{(42s+1)(5s+1)(2s+1)(s+1)} e^{-s}$. First and second-order approximations are presented.

the equivalent K-SIMC, the settings derived from the approximated second-order K-SIMC model perform slightly better compared to the respective SIMC.

Overall, the divergence from optimality for all the approximations for the "positive-numerator-time- constant" process is substantial. More specifically, the average the second-order approximations for the process 4.1 is around 11 % while the same figure for the process 4.6 for the second-order approximations is almost 40 %. Like-wise, a similar conclusion can be drawn for the first-order processes. The performance of the FOPTD is almost twice as poor for the process 4.6 when compared to the respective ones for the process 4.1. More approximations add more uncertainty and hence more deviation from the optimal performance.

4.4. Process model uncertainty

So far in this work, the process model was assumed beforehand. Hence, all the results were based on a *perfect* model. Nonetheless, in practice it is impossible to have such a model. Usually in industry, bump tests are performed and there is always the presence of model uncertainty. Process gain or time constants can deviate up to 100 % from the actual value. For instance, if a time constant is assumed to be equal to 1, there is also a strong possibility that it might be either 0.5 or 2 too. Having into the mind and analyzing the results above, it seemed interesting to see how sensitive are the methods to model error. A process model was assumed:

$$G(s) = \frac{e^{-s}}{(8s + 1)} \tag{4.7}$$

and for $M_S = 1.59$ the Pareto-Optimal, SIMC and K-SIMC parameters were found and they are summarized in Table 4.5 :

Table 4.5: PID controller settings for the process 4.7 for $M_S = 1.59$.

Method	PID Controller Settings				
	k_c	k_i	k_d	GM	PM
Pareto-Optimal	5.170	1.713	2.084	2.80	49.7
SIMC	5.175	0.757	1.640	2.87	63.1
K-SIMC	3.792	0.684	0	3.23	56.1

While, the PO and the SIMC give PID controllers, the K-SIMC method gives a PI controller. The performance will benefit from the presence of the derivative action. However, on the other hand, the PID controllers will be more sensitive to disturbances and noises, which is a direct consequence of the direct changes happening in the measured process variable. The controllers above were tested to 7 other process models, which are listed in the Table 4.6.

Table 4.6: Cases considered to evaluate the performance of the controllers in Table 4.5.

Process Model
$G(s) = \frac{1}{(5s+1)} e^{-s}$
$G(s) = \frac{1}{(11s+1)} e^{-s}$
$G(s) = \frac{2}{(5s+1)} e^{-s}$
$G(s) = \frac{1}{(5s+1)} e^{-2s}$
$G(s) = \frac{1}{(5s+1)(s+1)} e^{-s}$
$G(s) = \frac{(s+1)}{(5s+1)} e^{-s}$
$G(s) = \frac{(-s+1)}{(5s+1)} e^{-s}$

As it is observed, the time constant, process gain and time delay is altered. Moreover, processes with zeros, both LHP and RHP are introduced. Although, the processes might not cover every case, they can give an indication on how the various methodologies could perform. An input and an output step disturbance was added at t=0 and t=20 respectively. It is expected that the PI controller can handle a wider variety of process, trading-off the worse performance when compared to the PID for the stable systems. The input and output responses of the systems are displayed in Figures 4.18-4.24.

$$\boxed{\mathrm{G(s)} = \frac{\mathrm{e}^{-\mathrm{s}}}{(8\mathrm{s}+1)}}$$

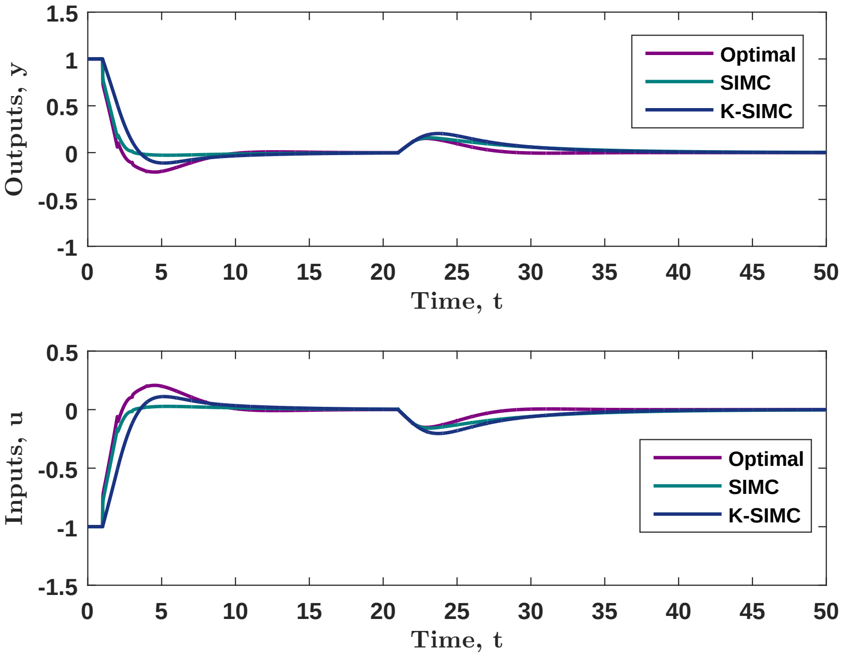
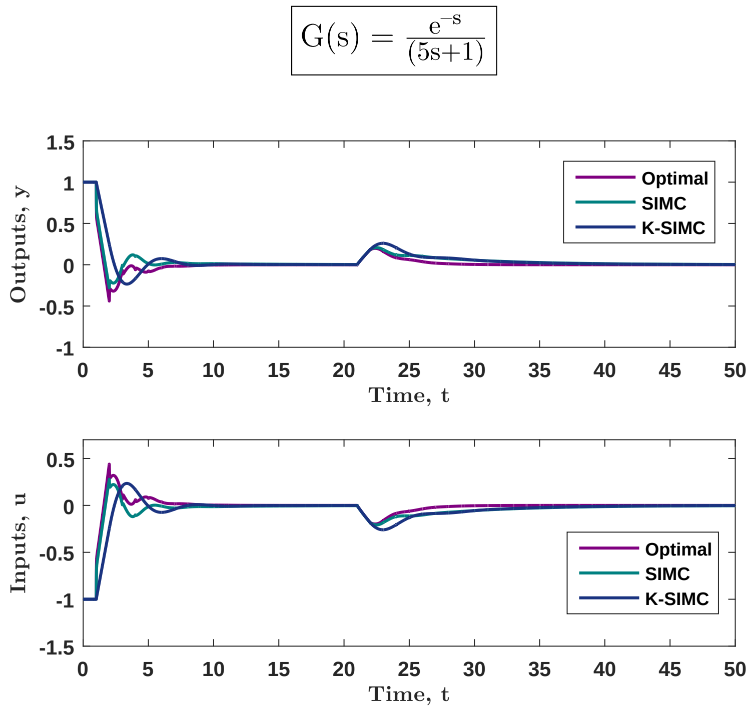


Figure 4.18: Input and Output responses for step input and output disturbances for the process $\mathrm{G(s)} = \frac{\mathrm{e}^{-\mathrm{s}}}{(8\mathrm{s}+1)}$.

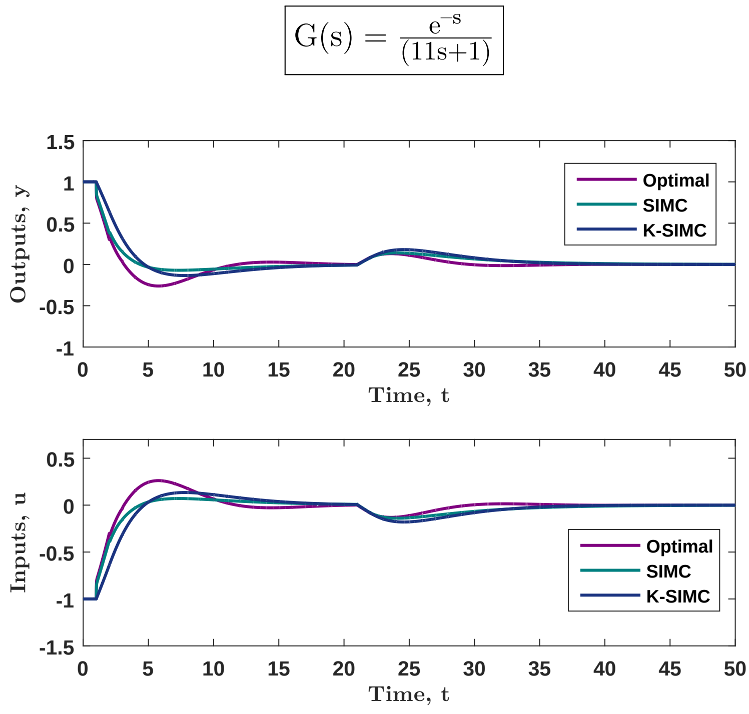
As it was anticipated, the PO controller outperforms the SIMC and the K-SIMC controllers. The K-SIMC settings result in slower response, which is due to the lack of derivative action of the controller. In Figure 4.19 and 4.20 the process time constant is changed, and once again the optimal and the SIMC give sufficient controllers performance-wise even though the response is more oscillatory. Although the K-SIMC tunings exhibit a slower responses, clearly seen in Figure 4.19, they can be characterized acceptable.

What might come to no surprise are the input and output responses when the process gain or the time delay is doubled while there is also a change in the time constant. Figures 4.21 and 4.22 clearly show that only the PI K-SIMC controller results in a stable system. The lack of the derivative term causes the system to be less sensitive in the trade-off of performance. Probably, similar results would have been derived if a SIMC PI controller was used.



$$G(s) = \frac{e^{-s}}{(5s+1)}$$

Figure 4.19: Input and Output responses for step input and output disturbances for the process $G(s) = \frac{e^{-s}}{(5s+1)}$.



$$G(s) = \frac{e^{-s}}{(11s+1)}$$

Figure 4.20: Input and Output responses for step input and output disturbances for the process $G(s) = \frac{e^{-s}}{(11s+1)}$.

$$\boxed{G(s) = \frac{2}{(5s+1)}\ e^{-s}}$$

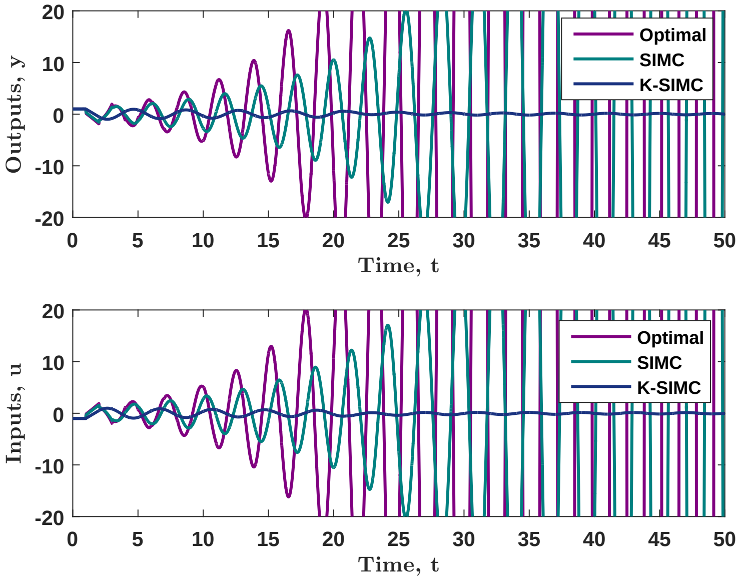


Figure 4.21: Input and Output responses for step input and output disturbances for the process $G(s) = \frac{2}{(5s+1)}\ e^{-s}$.

$$\boxed{G(s) = \frac{1}{(5s+1)}\ e^{-2s}}$$

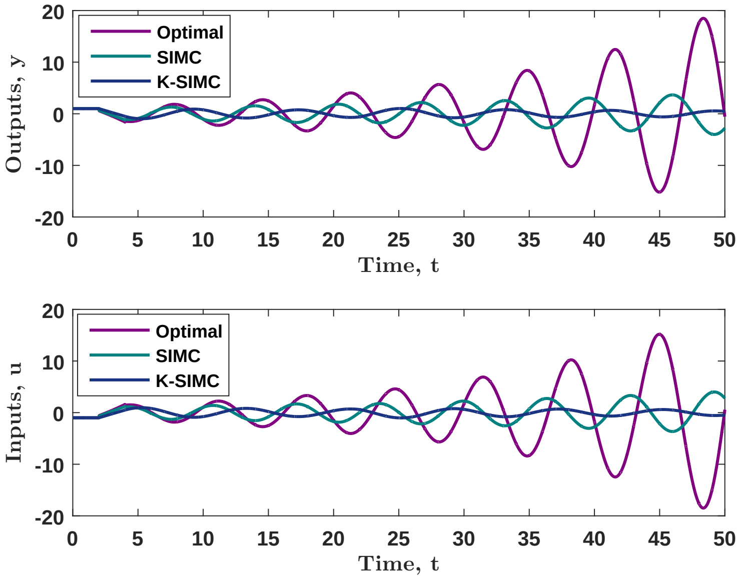
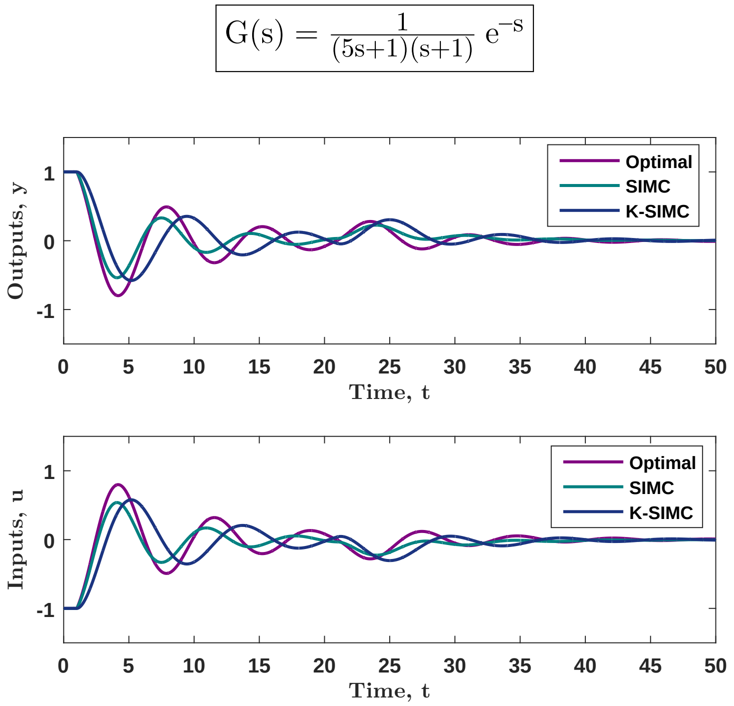
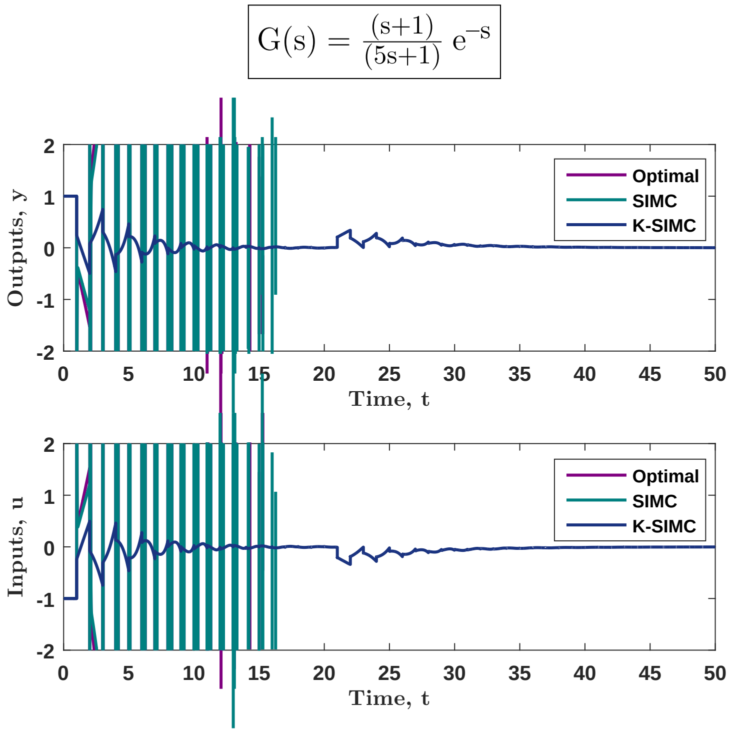


Figure 4.22: Input and Output responses for step input and output disturbances for the process $G(s) = \frac{1}{(5s+1)}\ e^{-2s}$.



$$\boxed{G(s) = \frac{1}{(5s+1)(s+1)}\, e^{-s}}$$

Figure 4.23: Input and Output responses for step input and output disturbances for the process $G(s) = \frac{1}{(5s+1)(s+1)}\, e^{-s}$.



$$\boxed{G(s) = \frac{(s+1)}{(5s+1)}\, e^{-s}}$$

Figure 4.24: Input and Output responses for step input and output disturbances for the process $G(s) = \frac{(s+1)}{(5s+1)}\, e^{-s}$.

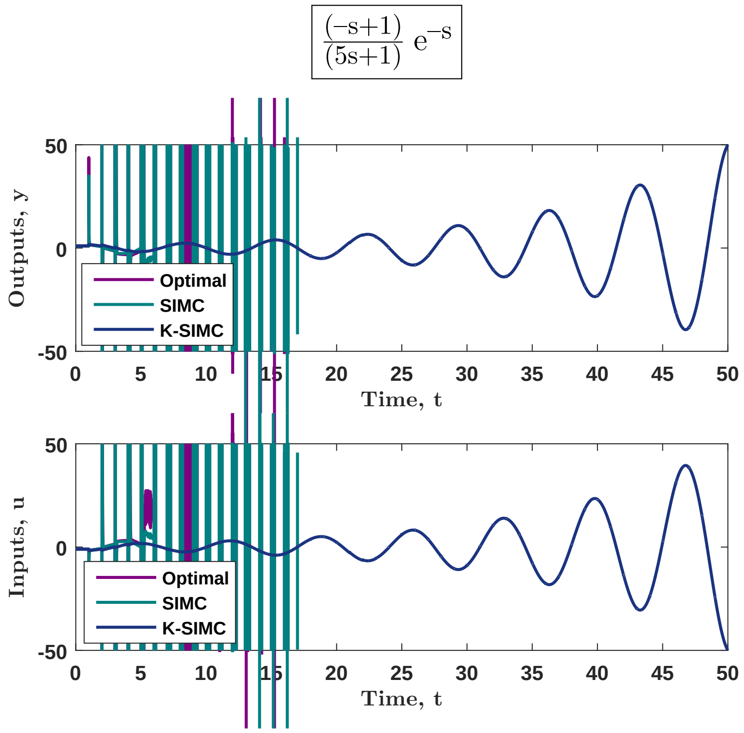


Figure 4.25: Input and Output responses for step input and output disturbances for the process $\frac{(-s+1)}{(5s+1)}\,e^{-s}$.

Furthermore, Table 4.5 also shows that the optimal and the SIMC controllers have significant higher gain too. So far in the analysis which has been done, the optimal and the SIMC tunings have almost identical proportional gain values. In addition to that, the SIMC rule suggests that the higher the process gain or the dead time of the process, the smaller the proportional term should be. Therefore, a suitable controller for those two cases here, should have had at least smaller proportional gain. All in all, the PO settings result in quicker oscillatory response compared to the SIMC. That can be attributed to the fact that, the PO controller has significantly larger integral gain or the integral time is less than half of the respective SIMC. To the existing knowledge, more integral actions tends to produce oscillatory responses and that is also observed here. That causes the system to be more sensitive, accompanied by sensitivity added by the presence of higher derivative action.

All of the tunings perform sufficiently in the SOPTD depicted in Figure 4.22 and the PO parameters settle the fastest. Nonetheless, this is not the case when zeros are added. It is known that negative values of the poles of the closed-loop characteristic equation can guarantee stability of a feedback system. Positive roots of the characteristic equations are present in the final two cases of this section. First, a LHP zero is introduced. The systems are unstable using the PO and the SIMC settings. But, on the other hand, the PI controller manages to make the system reach a steady-state, although there are steep changes in the measured variable. The presence of a RHP zero also makes the optimal and the SIMC systems to go unstable, which is also the case for the K-SIMC controller. Those results might be expected since the control of processes with zeros is a different story.

4.5. Other tuning methods

There is an abundance of PID tuning recipes which has been proposed in the literature. In this section, two of them are examined. The first one was proposed by Syrcos & Kookos, who solved an optimization problem to find customized PID settings and proposed rules to tune an ideal PID controller. Secondly, the Matlab Toolbox is able to propose the PID tuning parameters, which provide a balance between performance and robustness. For the former method a first-order process is investigated while for latter both first and second-order processes are tested.

4.5.1. Syrcos & Kookos tuning method

In this section, the PO the SIMC controllers will be compared to the ones derived form the approach, which was proposed by Syrcos & Kookos.

A first-order process with time delay is considered:

$$\mathrm{G(s)} = \frac{\mathrm{e}^{-\vartheta \mathrm{s}}}{(0.5\mathrm{s}+1)} \tag{4.8}$$

Now, there is no tuning parameter (τ_c or λ). Hence, first the time delay is scanned and the tuning parameters are obtained. Then, for the resulting PID settings the $\mathrm{M_S}$ values are calculated and for the same $\mathrm{M_S}$ values the SIMC controller tunings are found.

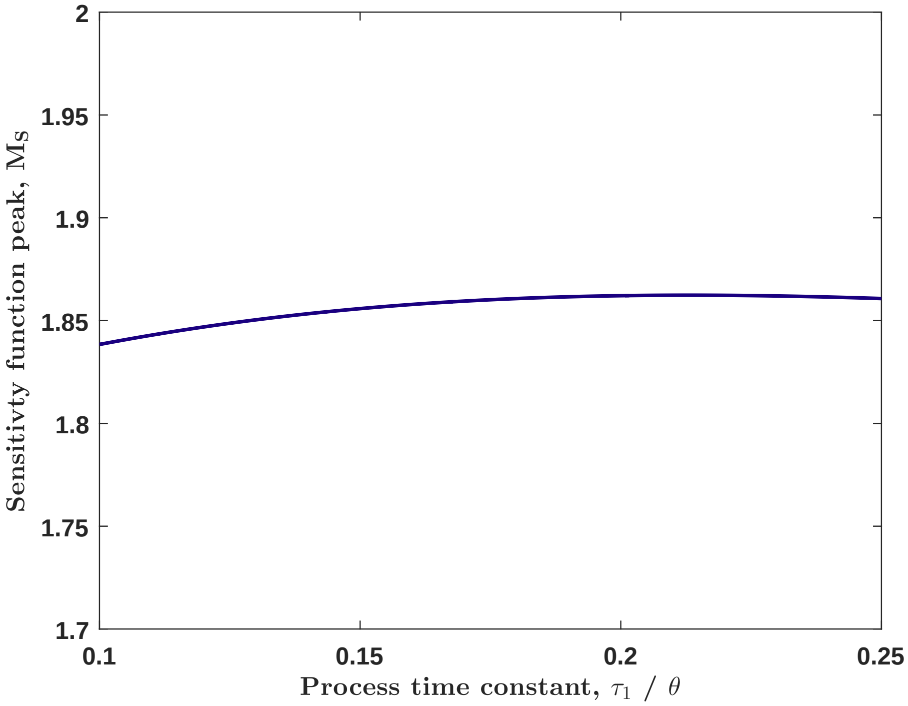


Figure 4.26: Sensitivity peak for different process time constants using Syrcos & Kookos method.

The results show that the obtained M_S ranges around 1.83 which is a reasonable robustness value. The PO optimal settings provide slightly higher gain and derivative time values to controller while the integral time is similar to the other two tuning methodologies. The performance of the different tunings is represented in Figure 4.27, and shows that for larger values of the time delay Syrcos & Kookos approach, actually performs better than the SIMC while SIMC gives superior results for smaller values of the time delay. Overall, both of them, have around 9 % deviation in terms of performance from the optimal. .

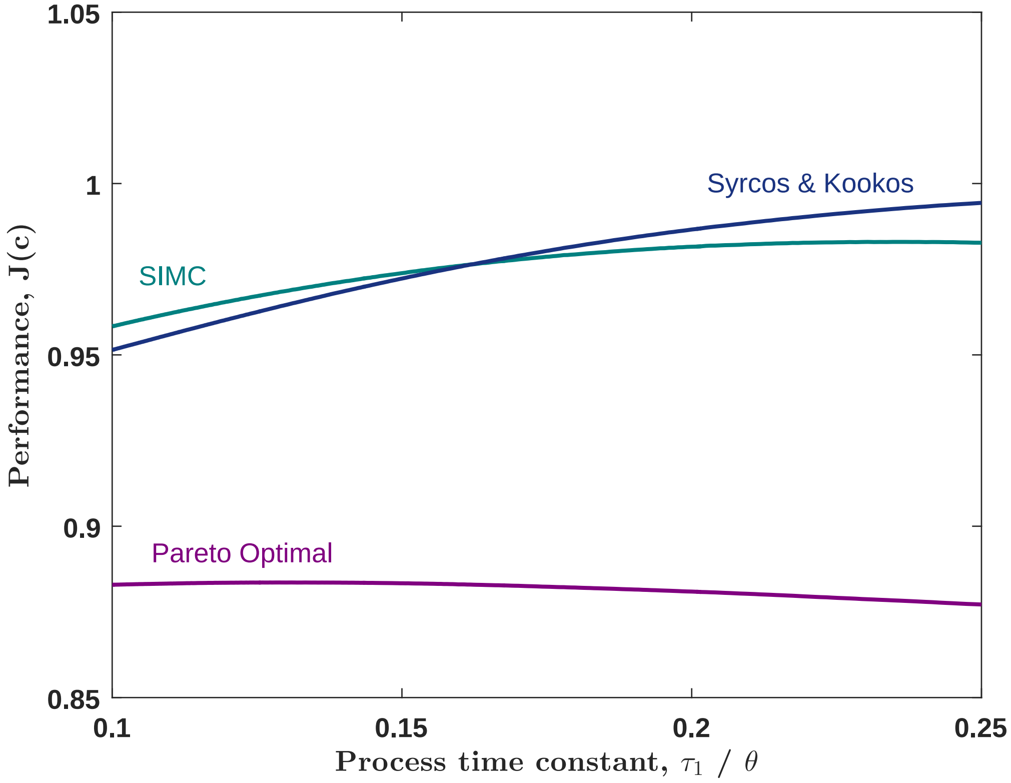


Figure 4.27: Cost function for different different robustness constraints using Syrcos & Kookos method.

Moreover, it is of great importance to calculate the frequency margins of the systems. Namely, the gain margin (GM), the phase margin (PM) and the delay margin (DM). All the derived settings give gain, phase and delay margins which are satisfactory. In general, a gain margin more than 2 is sufficient and in the case of phase margin, a value larger than 30 degrees is required. However, what is the most interesting in Figure 4.29 is the significantly, more than 20 %, larger delay margin that Syrcos & Kookos tunings provide, and that might be useful for cases where uncertainty is present.

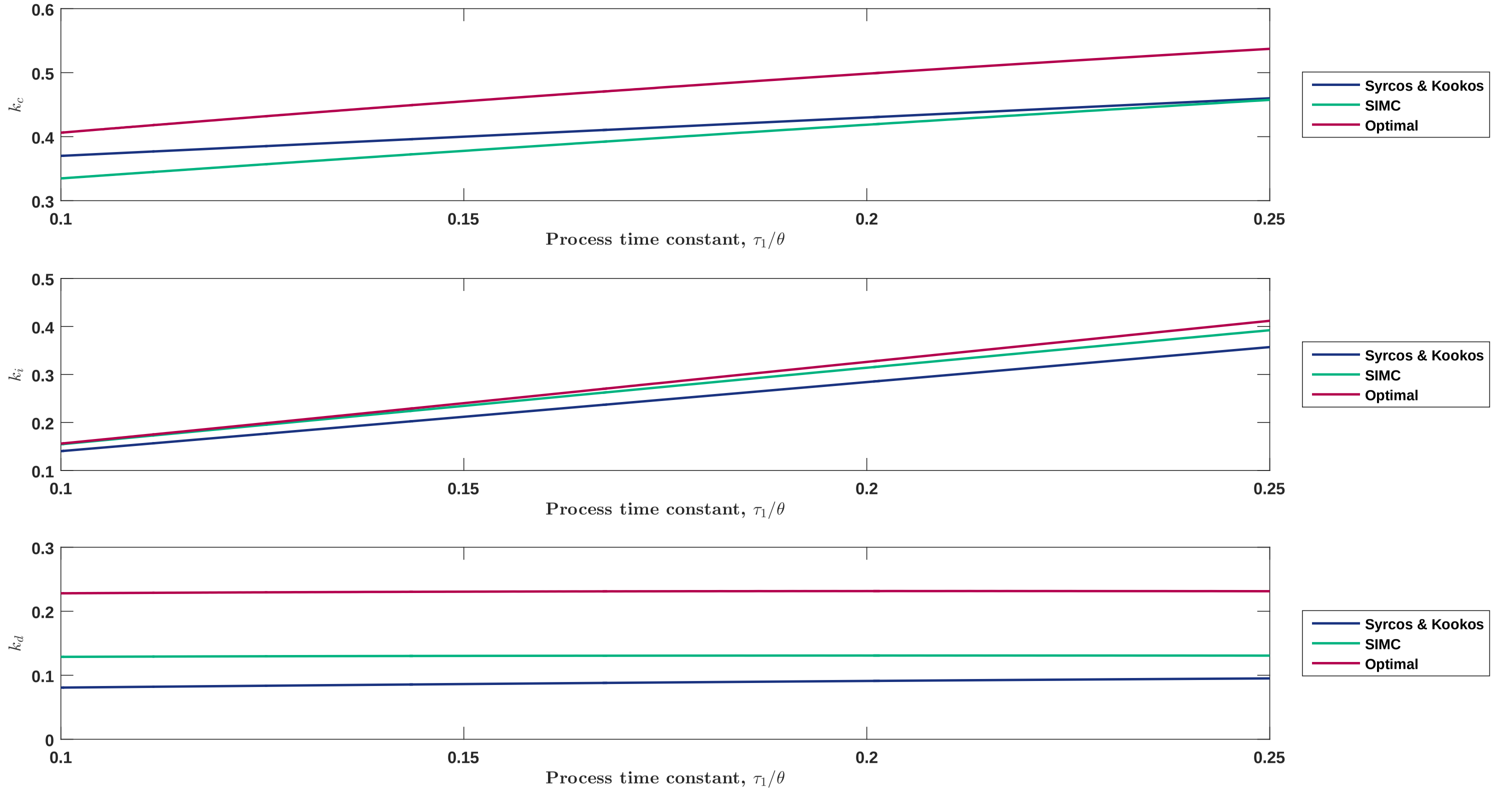


Figure 4.28: Resulting PID controller settings method for different process time constants using Syrcos & Kookos method. Comparison with ideal PID, SIMC and PO controllers.

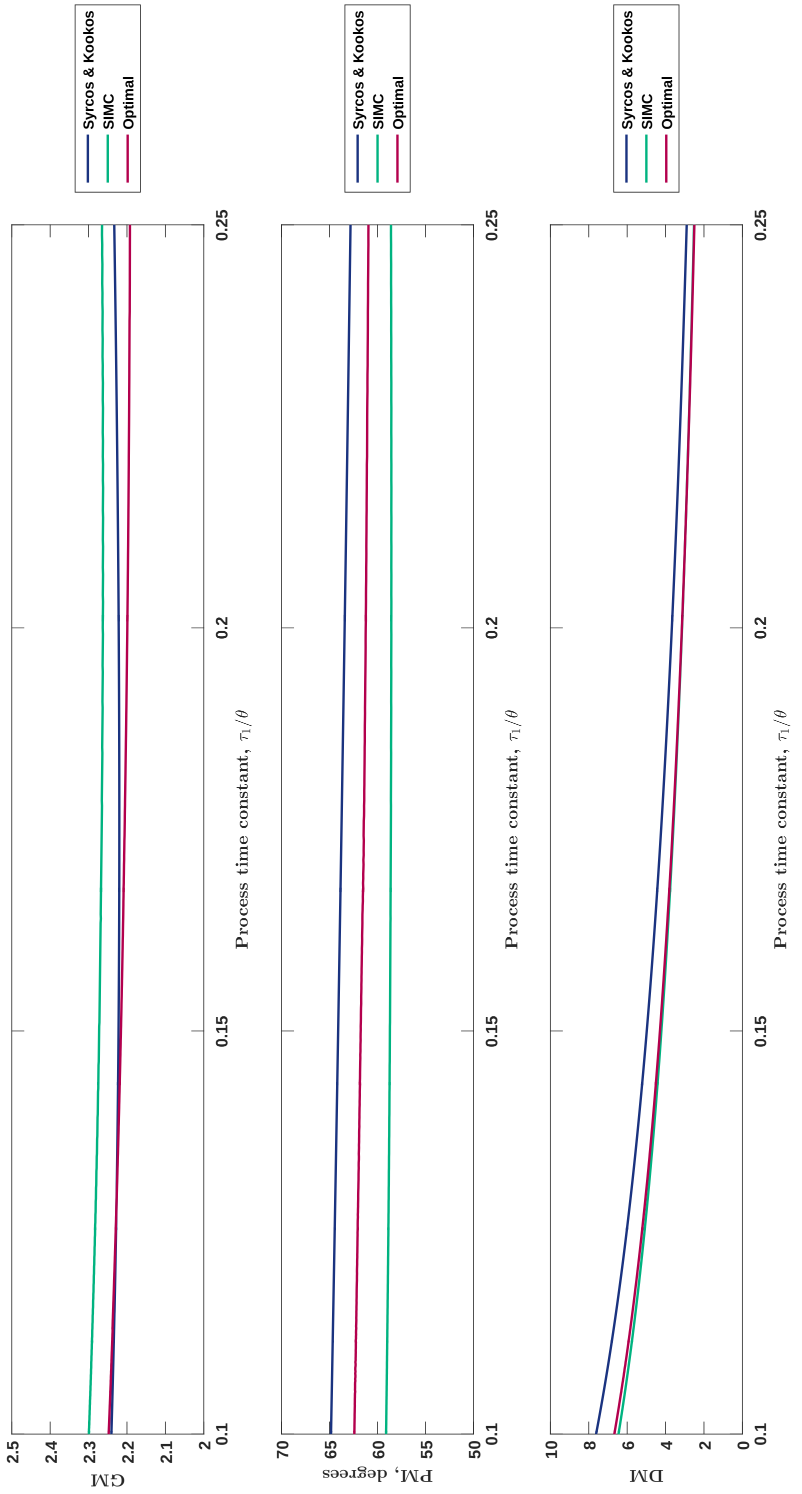


Figure 4.29: System Margins for different process time constants using Syrcos & Kookos method. Comparison with ideal PID, SIMC and PO controllers.

4.5.2. MATLAB Tuning Toolbox

MATLAB is able to automatically tune a controller for a given process. Taking advantage of the MATLAB Tuning Toolbox, balanced controllers in terms of performance and robustness are obtained. Here, ideal PID controllers are requested. The algorithm which is being used is the **pidtune** algorithm. Nevertheless, it is possible to *manually* determine the attitude of the controller (more aggressive or more robust).

First-order processes

Here, a first-order process with time delay is considered.

$$\mathrm{G(s)} = \frac{\mathrm{e}^{-\vartheta \mathrm{s}}}{(\mathrm{s}+1)} \qquad \text{for } \vartheta \ \in [0.1, 10] \tag{4.9}$$

where ϑ is the time delay.

The MATLAB Tuning Toolbox settings were found for $\vartheta \ \in [0.1, 10]$. Then, the resulting $\mathrm{M_S}$ values are calculated. For the same $\mathrm{M_S}$ values the SIMC and Pareto Optimal settings are found and the controllers are compared performance-wise.

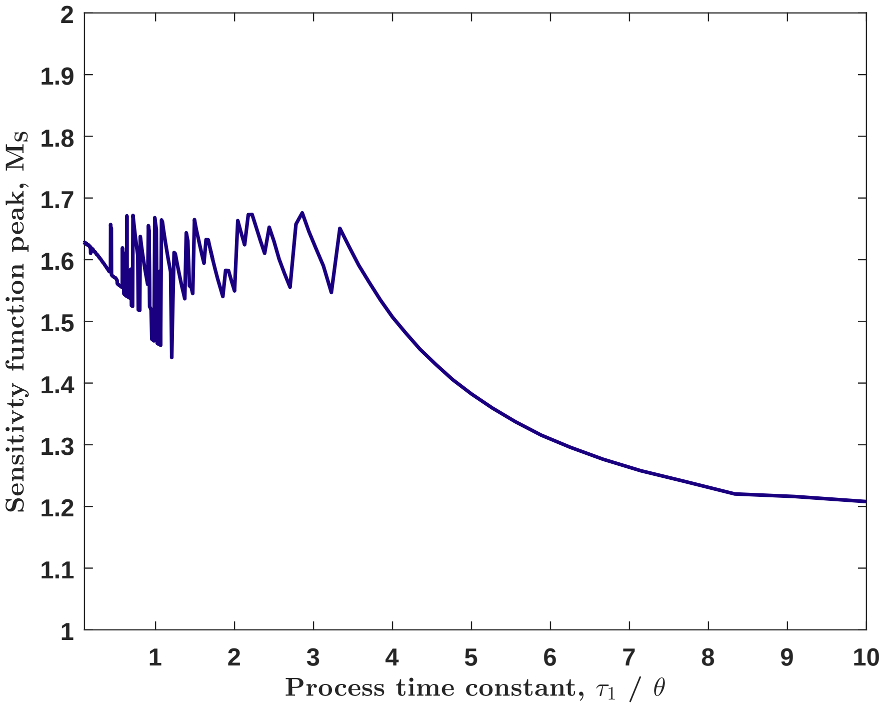


Figure 4.30: Sensitivity peak for different process time constants using the MATLAB Tuning Toolbox for a FOPTD.

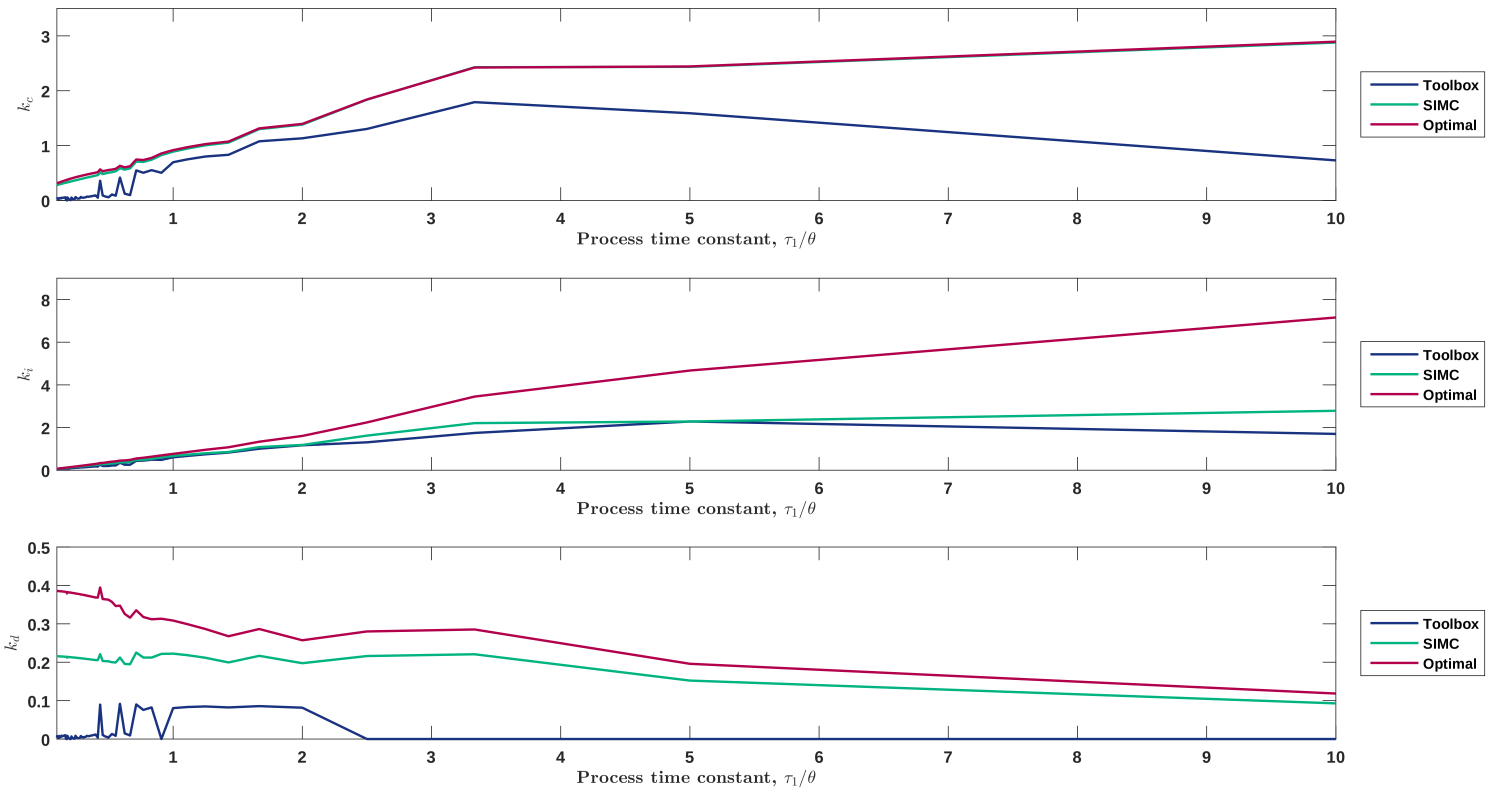


Figure 4.31: Resulting controller settings for different process time constants using the MATLAB Tuning Toolbox for a FOPTD.

From Figure 4.31 it is clear that the tuning parameters are not consistent or do not present a smooth behavior. Remarkable fluctuations in both k_c and k_d are apparent. Similar to the previous method discussed, k_c increases as ϑ decreases with an almost steady rate, which for the Syrcos and & Kookos method could be characterized as linear. Furthermore, even though for small values of the time delay a **PI** controller is obtained for ϑ for values close to $\vartheta = 1$, there is a considerable increase in the derivative gain. The reasons why that happens are not yet clear. However, it seems that for the intermediate values of the time delay the Toolbox suggests that the derivative term should exist. For very small or for very large values the derivative time is almost zero. Something that needs to be mentioned is the fact that the tuning obtained using the MATLAB Toolbox has also to do with the version of the MATLAB software. For instance, the controllers derived from the 2015 version of MATLAB give controllers with significantly higher M_S values. That probably has to do with the updates that are implemented to the software. Here, MATLAB 2014b was used.

The varying values of the PI or PID settings cause the sensitivity peak also to showcase a similar behavior. M_S fluctuates for high dead times again, but for the smaller values, the M_S diminishes and very robust controllers are obtained. That might be surprising because someone might have expected to see a more safe approach to higher dead times.

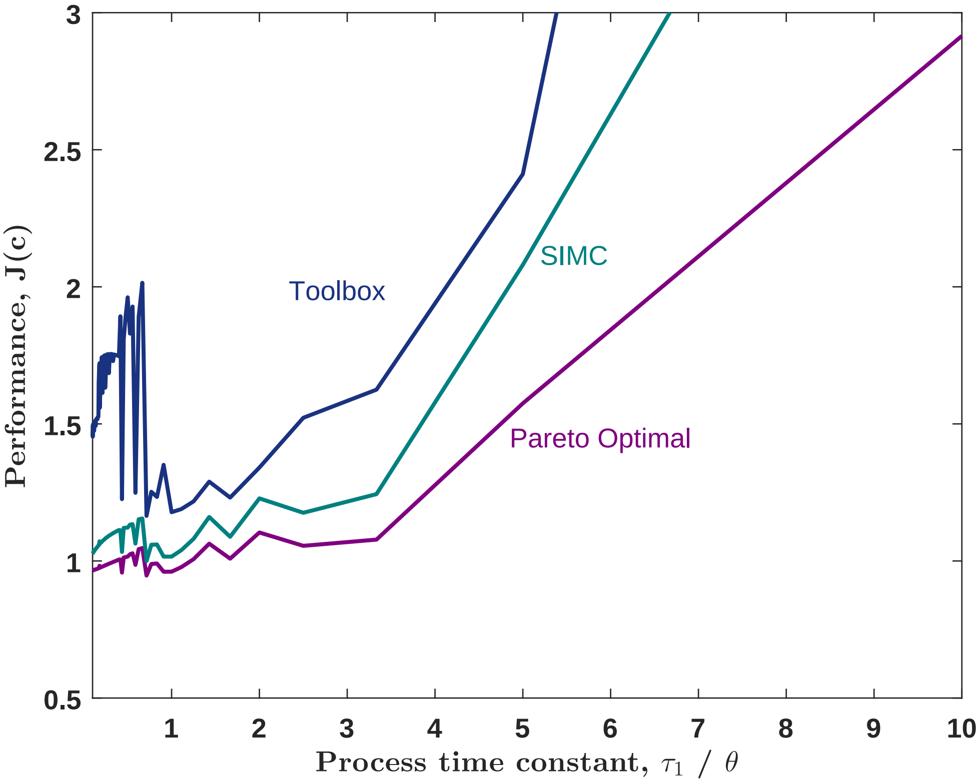


Figure 4.32: Cost function for different process time constants using the MATLAB Tuning Toolbox for a FOPTD.

The Performance vs. Robustness plots in the previous sections show that the difference of the cost function in the region of smaller M_S values is *larger* for a tuning method when compared to the optimal. The Matlab Tuning Toolbox gives controllers with low values of the M_S for small time delays. The varying M_S is translated also to the performance vs. robustness plots, in Figure 4.32. The SIMC tunings confirm the close to optimal behavior, which was demonstrated in the different cases which were investigated in the first section of the results. On the other hand, the Toolbox seems to noticeably give worse performance even when compared to the SIMC. That has to do, with the absence of derivative action of the controller and the substantially smaller proportional gain. When the value of the dead time decreases, that divergence from optimality is enhanced for both the other tuning approaches. That is not surprising at all, since for all the first and second-order processes examined so far, the deviation from the optimal performance was profound in the smaller values of the sensitivity peak. Overall, the PID Toolbox tunings presented almost 67 % inferior results when compared to the PO controllers, while the same percentage for the SIMC was considerably smaller.

All of the tunings, provide controllers with sufficient gain and phase margins while the delay margin dramatically diminishes as ϑ decreases. That essentially states that for the larger values of the time delay, even more time delay can be handled until the system goes unstable.

The most important part of the system margins graph is the Phase Margin plot. In the case of the MATLAB Toolbox, the phase margin does not change. It remains for the whole time delay interval at the same value, which is 60 degrees. Therefore, that is an indications which shows how the Toolbox chooses the controller parameters. MATLAB essentially states that **balance** means that the Phase Margin is 60 degrees! MATLAB is always trying to tune the controller from the "safe" side. It prefers to find a robust controller instead of specifying a degree of performance. That is probably the case for the SIMC tuning rules, in which sufficient margins values are also observed. Moreover, it should not be forgotten that Toolbox has two more options in terms of how the tuning of the controller is going to be done. First, the number of the open-loop unstable poles can be specified. Secondly, one can choose the type of the tuning, such as reference-tracking, disturbance rejection etc. Here, the default-balanced option is chosen, since the desired controller is one which has sufficient performance with acceptable robustness.

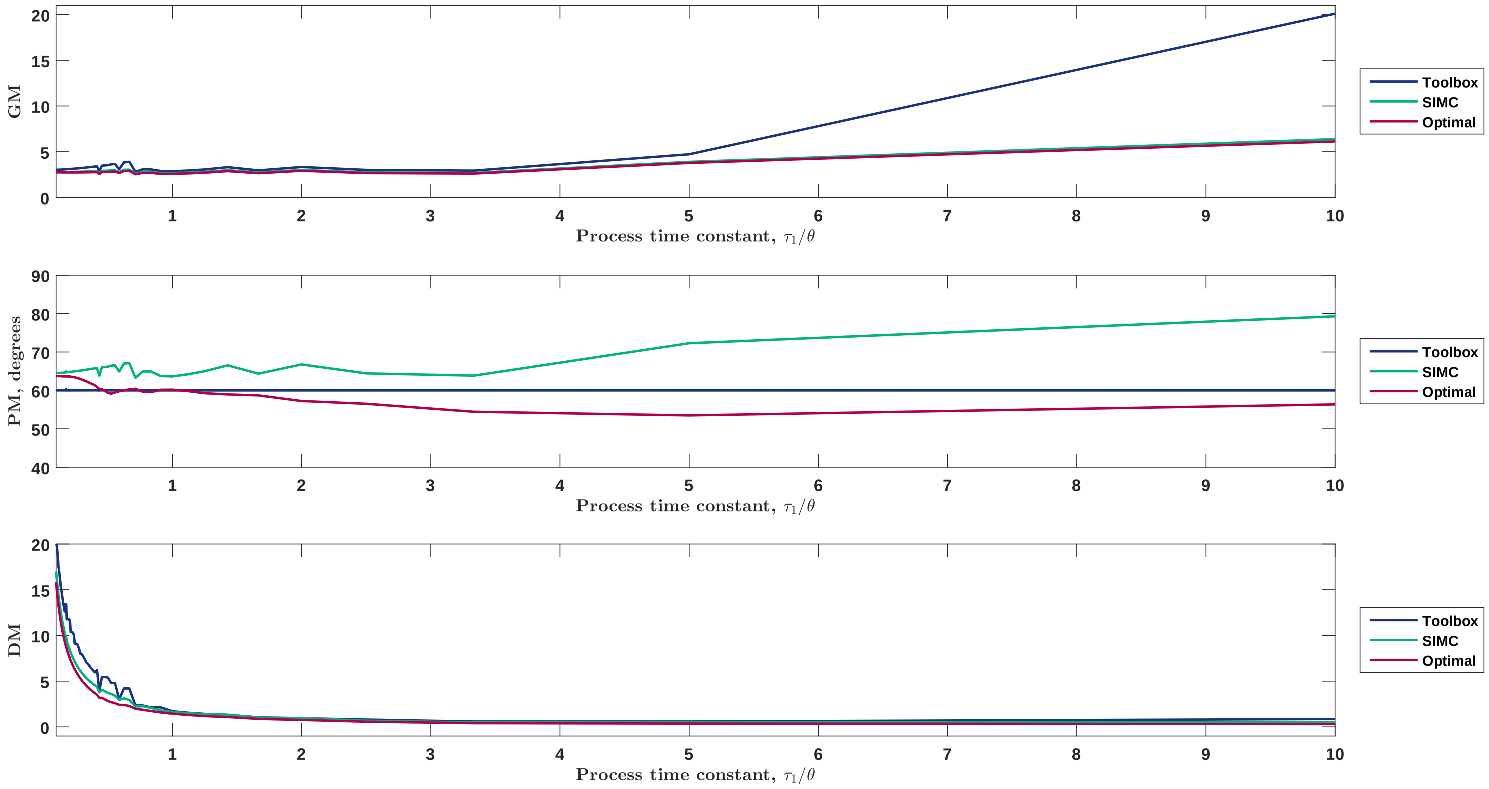


Figure 4.33: System Margins for different process time constants using the MATLAB Tuning Toolbox for a FOPTD. Comparison with ideal PID, SIMC and PO controllers.

Second-order processes

In addition to that, the a SOPTD is considered

$$G(s) = \frac{e^{-\vartheta s}}{(2s+1)\,(s+1)} \qquad \text{for } \vartheta \ \in [0.1, 10] \tag{4.10}$$

As it mentioned before, many processes can be described by second-order model. Therefore, it is of an equal interest to the FOPTD to observe the performance of the MATLAB Toolbox for the different time delays.

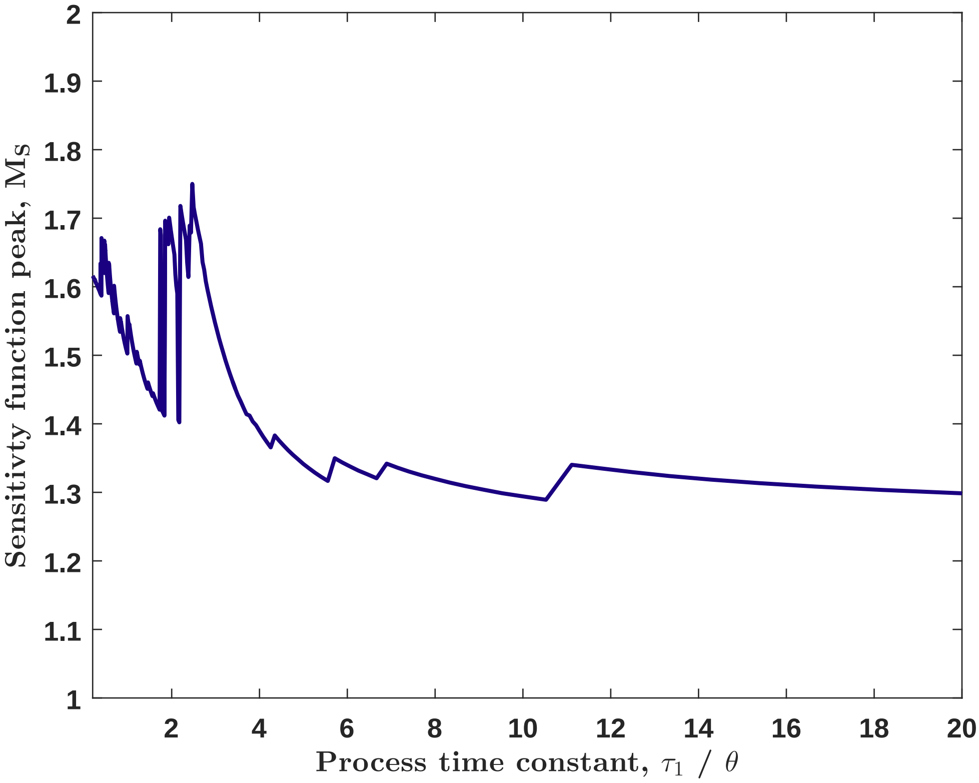


Figure 4.34: Sensitivity peak for different process time constants using the MATLAB Tuning Toolbox for a SOPTD.

As for the SOPTD, the results bear some resemblance to the ones derived from the FOPTD. The time constants of the process remained the same while ϑ varied. Figure 4.36 exhibits the PO, the SIMC and the MATLAB Toolbox tuning parameters. It is easily observable that although the the optimal and the SIMC tuning for k_c is almost identical, MATLAB essentially suggests that the gain of the controller should remain constant for small time delays. In addition to that, the divergence of both k_i and k_d is substantial. Something that needs to be highlighted tho, is that for a second-order process with time

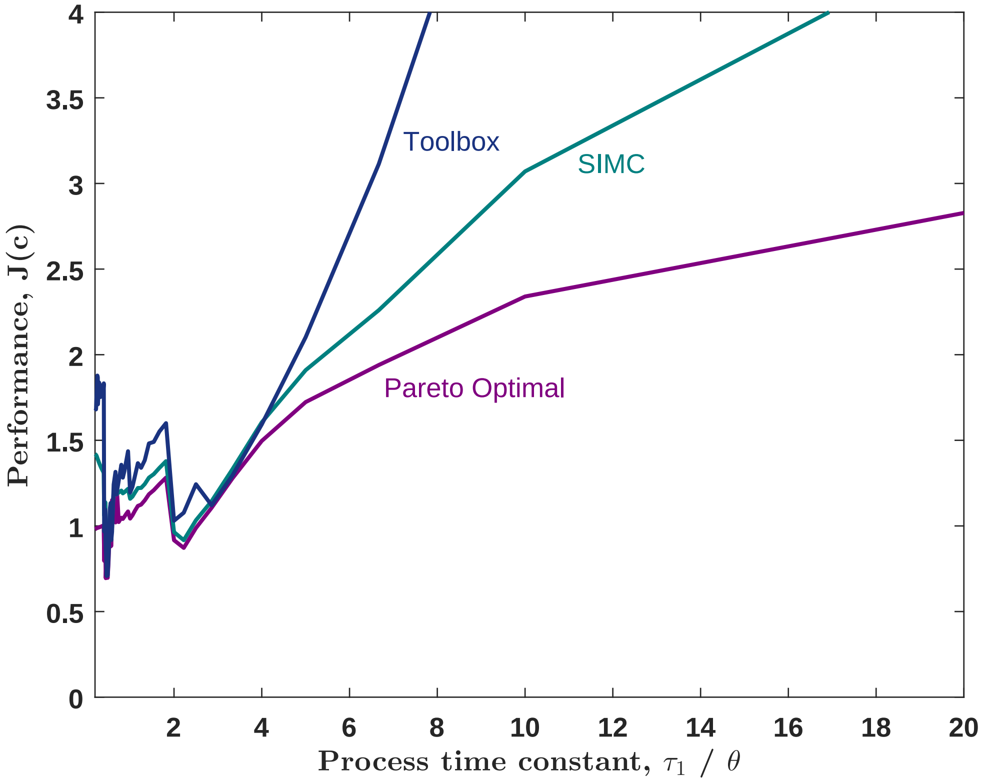


Figure 4.35: Cost function for different process time constants using the MATLAB Tuning Toolbox for a SOPTD. Comparison with ideal PID, SIMC and PO controllers.

delay in general, the MATLAB always adds the derivative action to the controller, which again limited compared to the PO and the SIMC.

Once more, small dead times result in tunings which give very robust controllers. The addition of the derivative action indeed improves the performance of the controller as it can clearly be seen from the performance graph. Although, the τ_1/ϑ range is larger again the overall performance deviation from the optimal remains almost the same. That is not the case for the SIMC, since the performance deteriorates. That is an direct consequence of the fact that the span, that M_S is at the very robust region, is larger.

All the margins guarantee sufficient stability. Similar to the FOPTD, the MATLAB Toolbox chooses again the phase margin as the tuning criterion. For all the cases, the phase margin is set to 60 degrees. Therefore, the Toolbox again prefers to design a robust controller instead of a more aggressive one. One interesting point, is the tendency of the Toolbox controllers have huge GM for small dead times. The time delay margin decreases as the time delay decreases and all three tunings present a similar behavior in terms of how much delay can be added to the system.

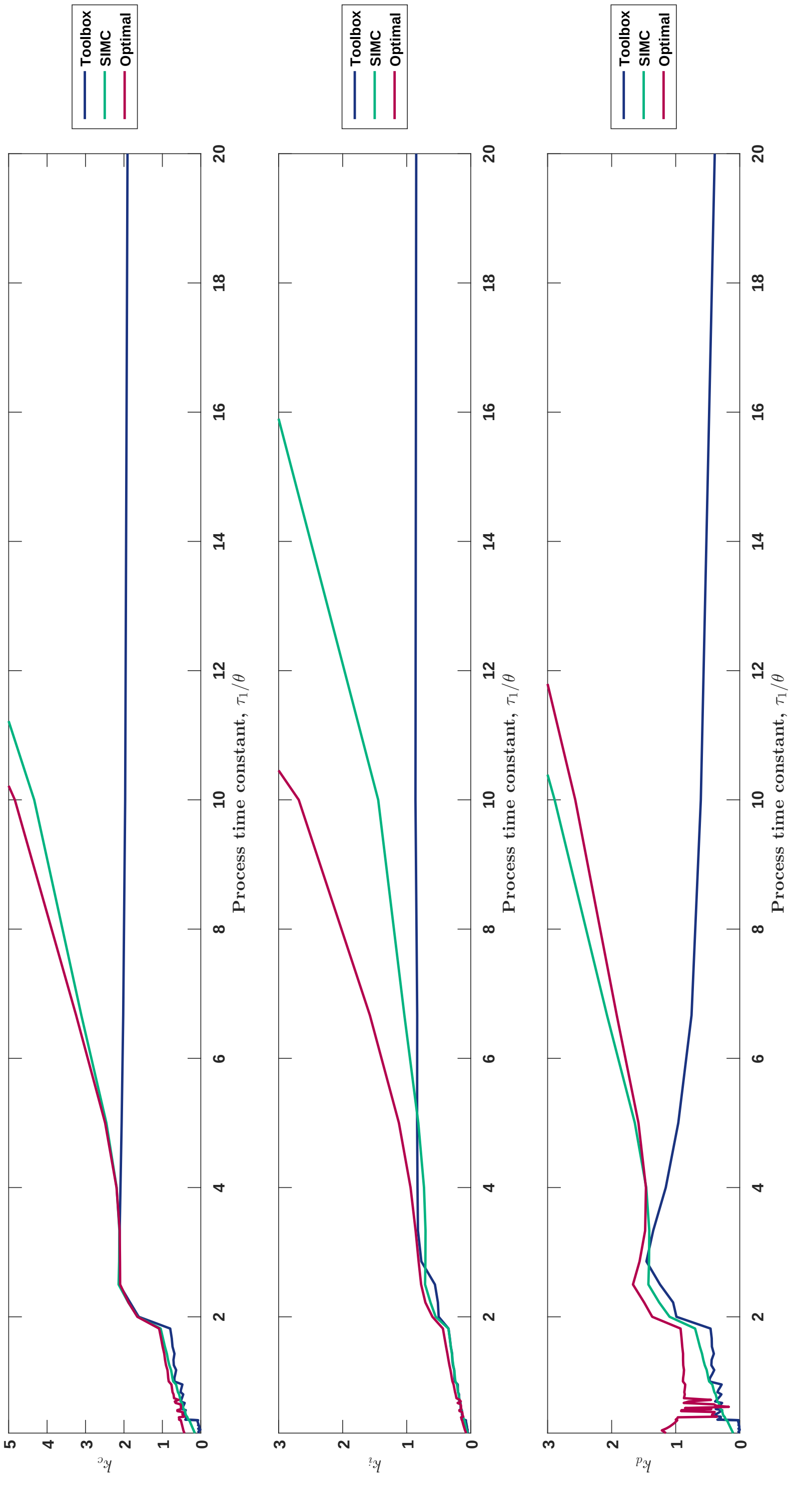


Figure 4.36: Resulting controller settings for different process time constants using the MATLAB Tuning Toolbox for a SOPTD. Comparison with ideal PID, SIMC and PO controllers.

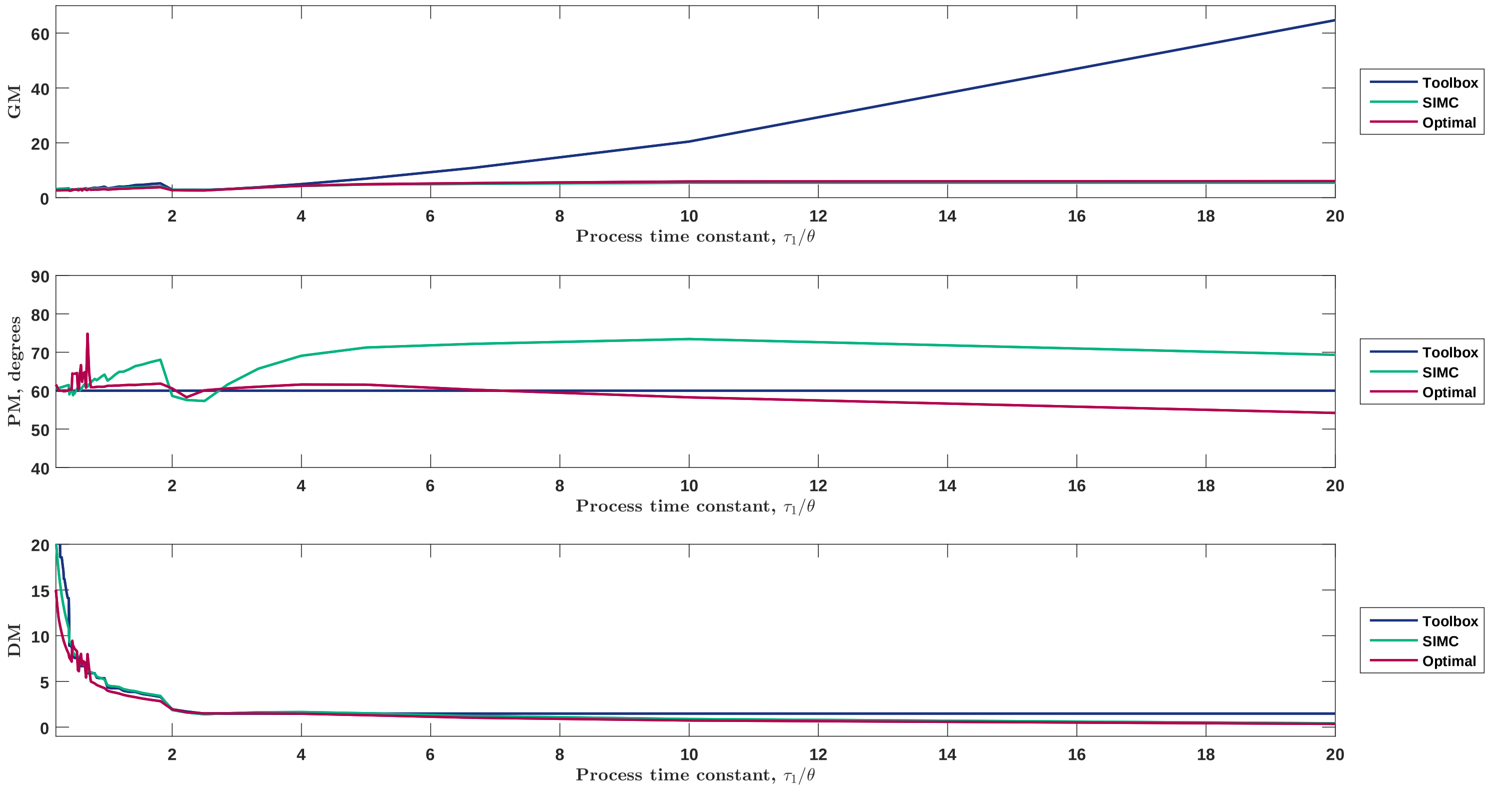


Figure 4.37: System Margins for different process time constants using the MATLAB Tuning Toolbox for a SOPTD. Comparison with ideal PID, SIMC and PO controllers.

4.6. Case study: The thermal/optical plant uDAQ28/LT

4.6.1. Introduction to the plant

So far in this work, many processes were examined. However, what might be missing, is a practical example to be able to concretely observe the performance of the different controller tunings. In such a way, a case study is investigated: The thermal/optical plant uDAQ28/LT. The plant is depicted in Figure 4.38

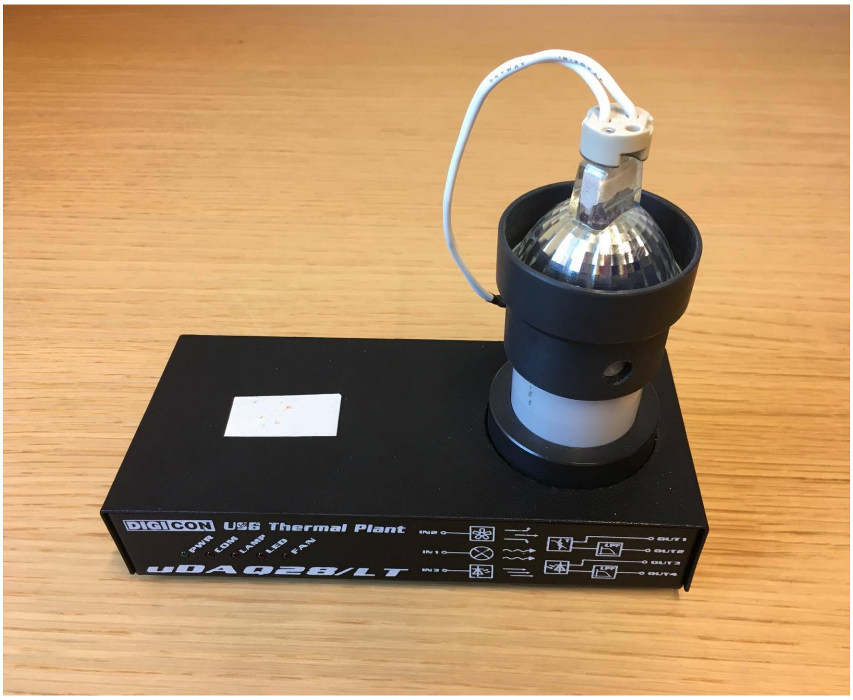


Figure 4.38: The thermal/optical plant uDAQ28/LT.

The thermal/optical plant is widely used for teaching mainly purposes in various universities around the world [2]. It has has a bulb and two LED's, which are sources of heat and light and a fan, which is used to decrease the temperature. A simplified scheme of the plant is shown in Figure 4.39.

[2] The uDAQ28/LT thermal/optical plant was developed at the Faculty of Electrical Engineering and Information Technology in Bratislava in cooperation with the company Digicon. Several universities use it at their laboratories for teaching purposes. Namely, some of them are the Fern Universität in Hagen, the University of Split, the University of Ancona and the NTNU.

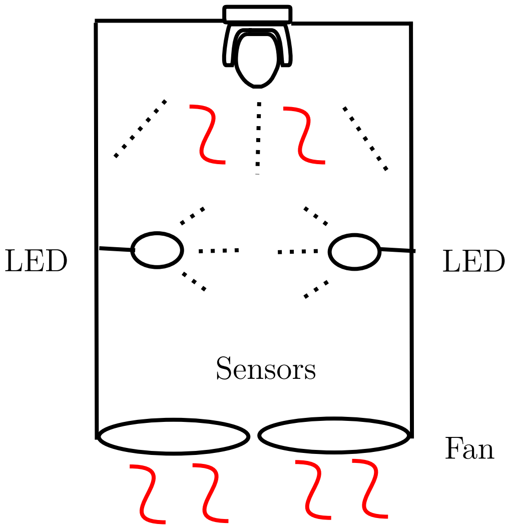


Figure 4.39: Sketch of the uDAQ28/LT thermal/optical plant.

The plant has 3 process inputs u and 2 process outputs y. Some of them have to do with the temperature manipulation and some of them with the control of the light intensity. The inputs and the outputs of the plant are summarized in 4.7 and 4.8

Table 4.7: Thermal/optical plant uDAQ28/LT inputs

Input	Description	Range
u_1	Bulb voltage (heat & light source)	0-5 V
u_2	Fan voltage (heat source)	0-5 V
u_2	LED (diode) (light source)	0-5 V

Table 4.8: Thermal/optical plant uDAQ28/LT outputs

Output	Description	Unit
y_1	Temperature measurement	°C
y_2	Light intensity measurement	Not known [3]

[3] The signal from the photodiode, which is used to measure the light intensity is scaled in the hardware.

As it is shown in Figure 4.39, the plant has:

- A bulb, which is used as a heat and light source.
- Two LED's which although give off little heat they are primarily a light source.
- A fan which is used to lower the temperature of the system.

Having that into consideration, there are two measurements: The temperature and the light intensity. Moreover, filtered measurements of the temperature and the light intensity are also supported and hence the user can take advantage of them to improve the control.

Since, the aim of this work is to test different tuning methodologies, a detailed analysis of the MIMO system is not required. Grimstad has already done such analysis [2009]. For the purposes of this thesis, only the bulb is going to be active during the experiment. The resulting output measurement will be the temperature.

There are two ways that heat is transferred in this case. First and foremost, through radiation, which is a direct result of the motion of charged particles. Besides radiation, heat is also transferred by convection. The temperature of the air between the bulb and the sensor starts to increase when the bulb turns on. After the while, hot air molecules reach the sensor and through that the amount of heat transferred increases. Having that in mind, the goal is to be able to efficiently control the plant using the different approaches. Nonetheless, first, the model of the plant must be found.

4.6.2. Model Identification

The model parameters for a process are usually obtained from a step response experiment. Although, this is probably one of the first things that someone learns in a dynamics course and it might also be not the most effective method, it is simple to use. For "lag" dominated processes, which essentially means that $\tau_1 > 8\vartheta$, it might take a really long time for a process to settle and reach an almost constant value. On that occasion, an option is to approximate it as an integrating process

$$\frac{ke^{-\vartheta s}}{\tau_1 s + 1} \approx \frac{k' e^{-\vartheta s}}{s} \tag{4.11}$$

where, $k' \overset{\text{def}}{=} k/\tau_1$.

The reasoning behind that, is that the individual values of the time constant τ_1 and the gain k are not very important for the controller design. Having that in mind, the process of the model identification follows.

Figure 4.40 shows how k' and = theta can be obtained from an open-loop experiment:
The reasoning behind that is that the individual values of the time constant τ_1 and the gain k are not very important for the controller design. On the other hand, what is extremely significant is their ratio k', which essentially decides the tuning parameters according to the SIMC and the K-SIMC rules.

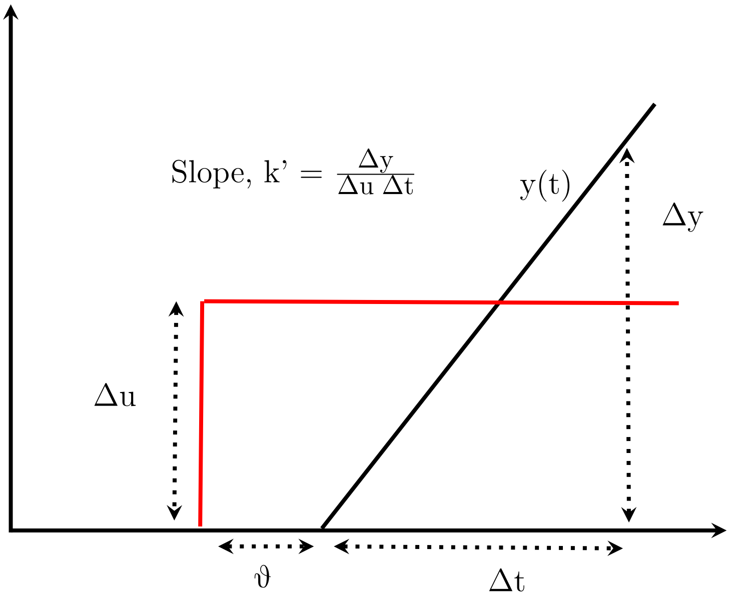


Figure 4.40: Open-loop step response to obtain the parameters k' and ϑ for an integrating process.

At t=0, the bulb is imposed to a step. As a result, light is emitted and the temperature starts to increase. The unfiltered temperature is measured and the sampling time was set to 0.11 seconds. After 300 seconds, the temperature continuous to increase and the contribution of convection is abundantly clear. The overall change in temperature is shown in Figure 4.41

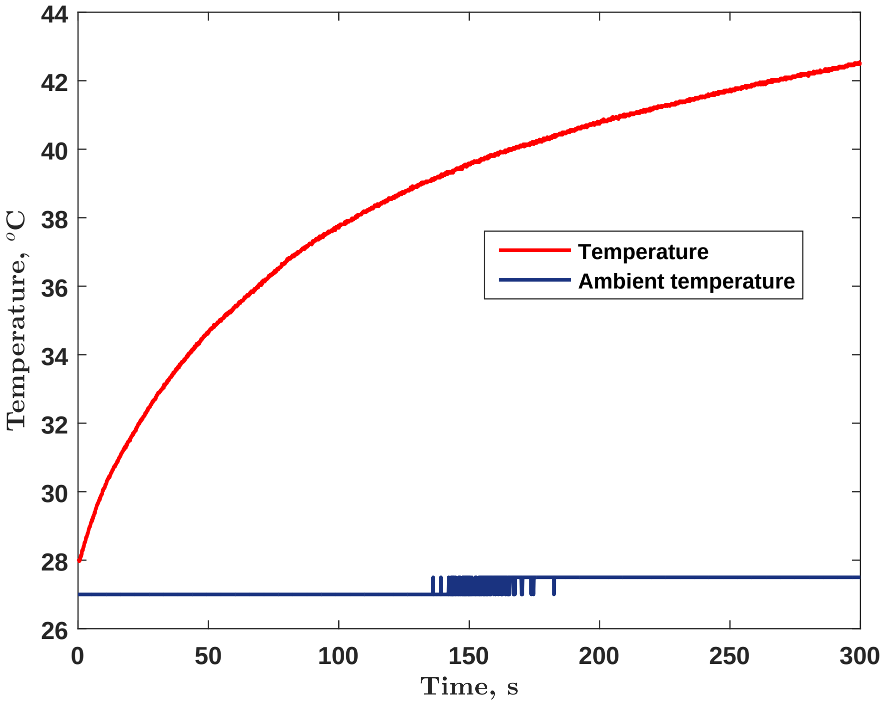


Figure 4.41: Step response of thermal/optical plant uDAQ28/LT.

It is evident that the process is very slow and it takes a long time to settle. That's why it is going to be approximated as an integrated process. Looking more closely to the first seconds of the process, it seems that it can be approximated by an integrating process, which has the following form:

$$\mathrm{G}\left(\mathrm{s}\right)=\frac{0.086\ \mathrm{e}^{-0.55\mathrm{s}}}{\mathrm{s}} \tag{4.12}$$

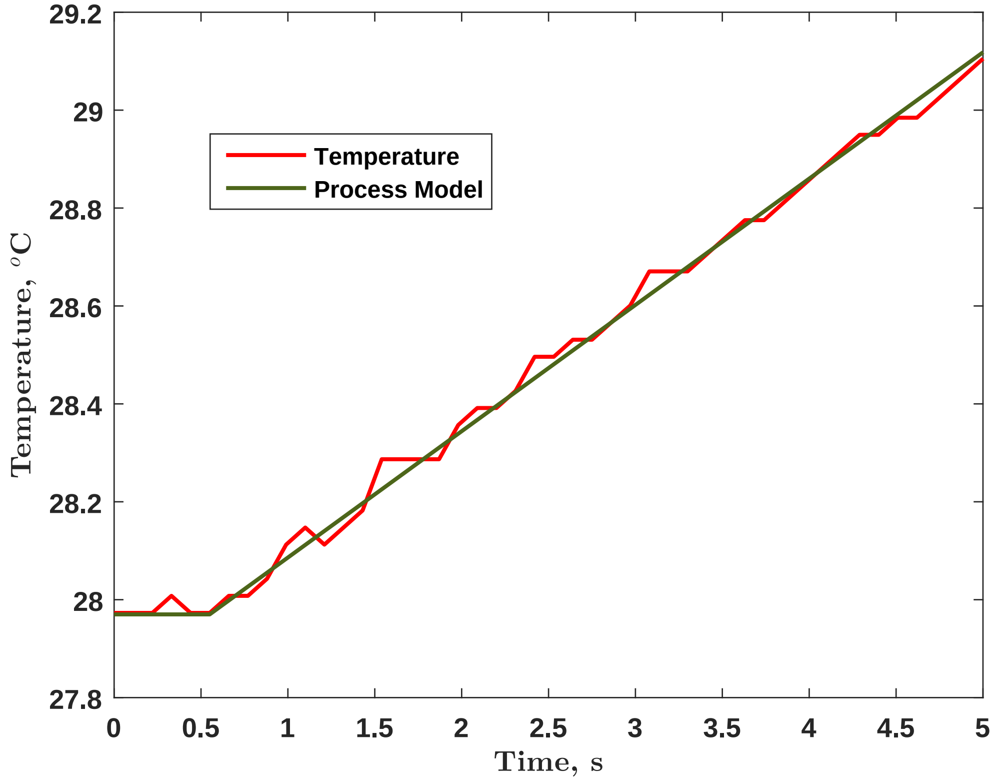


Figure 4.42: Step response of thermal/optical plant uDAQ28/LT.

The fitting is very close to the actual experimental data and the behavior of the temperature function can be approximated by the integrating process as it is depicted in Figure 4.42. That model is going to be used to find the controllers which are going to be implemented to control the temperature.

4.6.3. Controllers Design

Now that the process model is found, let's proceed to the design of the controllers. Besides the PID Pareto-Optimal controller, both the PI and PID using SIMC and the K-SIMC PI tunings are obtained using the rules presented in Table 4.9.

Table 4.9: SIMC and K-SIMC tunings for an integrating process.

Controller	k_c	τ_I	τ_D	$F_r(s)$
SIMC PI	$\frac{1}{k'}\frac{1}{(\tau_c+\vartheta)}$	$4(\tau_c+\vartheta)$	-	-
SIMC PID	$\frac{1}{k'}\frac{1}{(\tau_c+\vartheta)}$	$4(\tau_c+\vartheta)$	$k/3$	-
K-SIMC PI	$\frac{1}{k'}\frac{1}{(\lambda+\vartheta)}$	5λ	-	$\frac{2.5\lambda s+1}{5\lambda s+1}$

All the controllers, which are calculated should have the same robustness since they will be compared and they are summarized in 4.10

Table 4.10: PID controller settings for the process 4.12 for $M_S = 1.59$.

Method	PID	Controller	Settings		
	k_c	k_i	k_d	$F(r)$	M_S
Pareto-Optimal PID	13.143	6.977	3.015	-	1.59
SIMC PID	13.065	3.670	2.395	-	1.59
K-SIMC PI	8.917	2.365	0	$\frac{1.89s+1}{3.77s+1}$	1.59
SIMC PI	9.431	1.912	0	-	1.59

4.6.4. Experiments

A lot different experiments can be carried out using the plant. Here, the different controllers will be tested in terms of their performance and their robustness. Initially, the temperature is set to 35 °C. After the setpoint is reached, a 3 °C degrees step change is imposed to the plant. At this point, special attention should be paid to the rapidity that the setpoint is reached and if any oscillations are present. After 50s, a 3 unit plant output disturbance to the temperature causes the controller to act. The evaluation criterion will be the ability of the controller to efficiently suppress the disturbance. The resulting output and input responses are depicted in the following two figures:

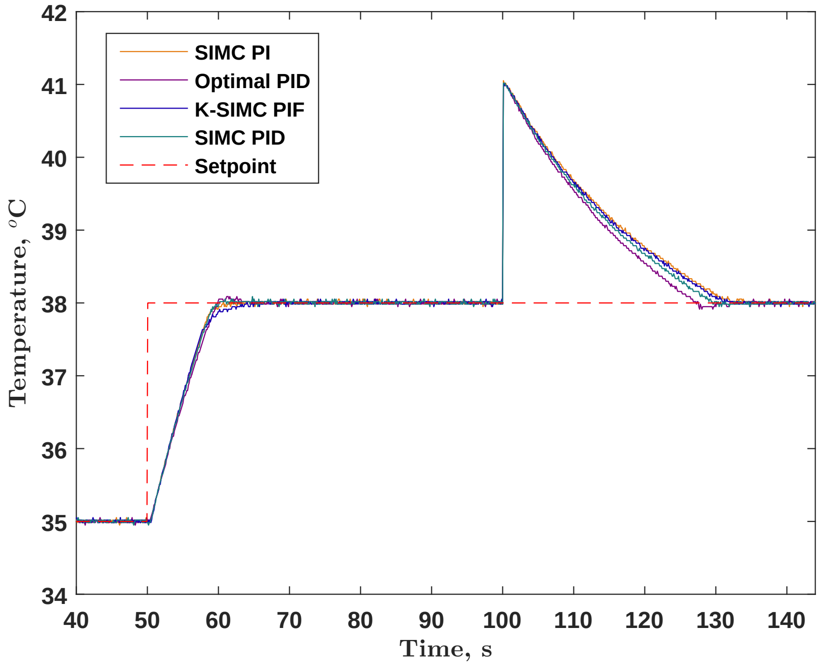


Figure 4.43: Output responses of the thermal/optimal plant uDAQ28/LT using using the controllers mentioned in Table 4.10.

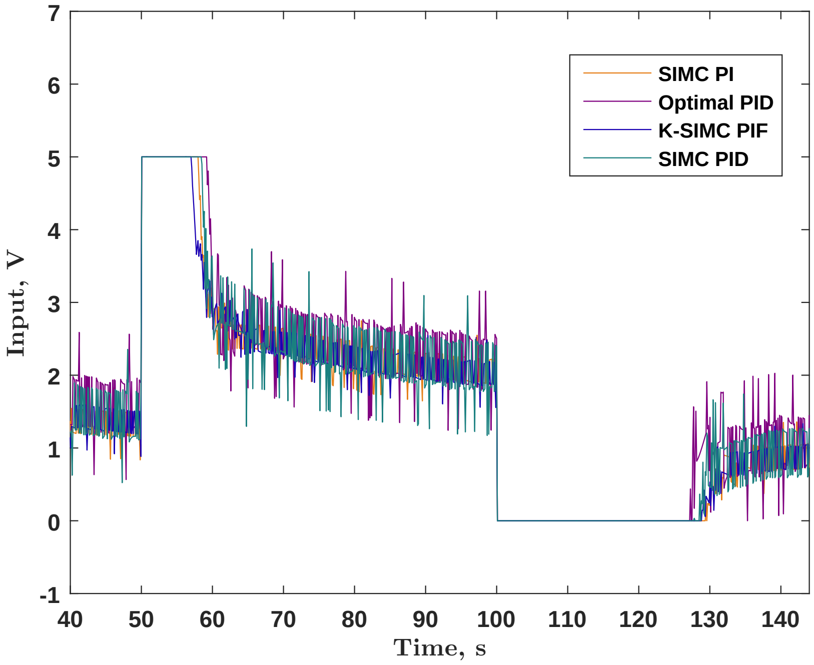


Figure 4.44: Input responses of the thermal/optimal plant uDAQ28/LT using using the controllers mentioned in Table 4.10.

It is apparent but also expected, that the derivative action causes higher input usage. That is more clearly depicted in Figure 4.44

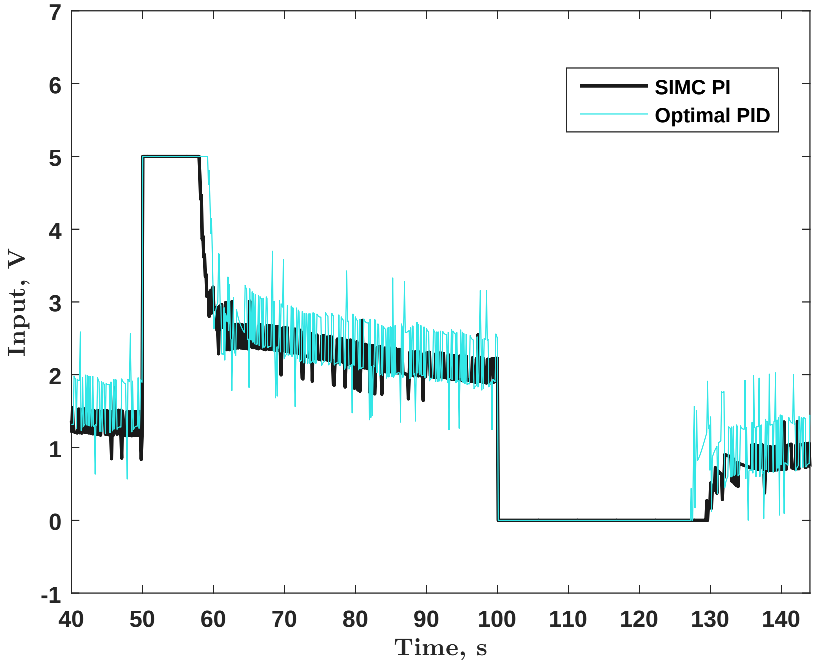


Figure 4.45: SIMC PI and Optimal PID input responses of the thermal/optimal plant uDAQ28/LT using using the controllers mentioned in Table 4.10.

The PI controllers have the advantage of using less input in the trade-off of performance. For instance, the SIMC PI controller requires less than half input compared to an optimal PID controller. Not only the average is higher but also peaks in Volt usage are observed. Those fluctuations in the electric potential could possibly cause problems such as attrition to the device.

The higher input usage should have been translated tho in better performance. That's not absolutely true for this experiment. Figure 4.44 clearly shows that the controller very quickly saturates. In fact, many experiments have been conducted, in which smaller setpoint changes or disturbances were applied and still saturation was observed. Therefore, one can conclude that the performance advantages which someone gets from the derivative action is negligible compared to massive difference is the input usage. Although in Figure 4.43 the capabilities of the PID controllers to suppress the disturbance are observable, it might be not worth to have such high input for that marginal performance increase.

For that reason, a measurement filter could be introduced and hopefully as a consequence, noise would be restrained. However, after many experiments the performance loss due to the filtering was so significant that the PI controller performed better than the PID. Comarping now, the two PI controllers, the K-SIMC with the filter and SIMC, the latter achieves the setpoints faster while both of them show the same performance on the disturbance rejection. Therefore, it can be concluded the PI SIMC controller is the most suitable for the control of the plant.

Chapter 5

Conclusions and future work

The aim of this work was to find ideal PID controllers using some of the most popular tuning methodologies and observe, how close to optimality the resulting controller parameters are. Therefore, Pareto-optimal ideal PID controllers were found, for several processes. The controllers presented a trade-off between performance and robustness. Performance was quantified in terms of the integrated absolute error, IAE, while robustness was measured in terms of the maximum peak of the sensitivity function, M_S. Due to the analytical expressions of the gradients both for the objective cost function and for the constraints, convergence to the true optimal was relatively easily achieved. Initially, first order-processes with time delay were considered. The time constant varied while the time delay remained constant. The SIMC rules presented a close-to-optimality behavior while the K-SIMC controllers perform sufficiently at larger M_S values. The K-SIMC controllers differ significantly performance-wise from the optimal, because they propose a PI controller instead of a PID for a big part of the robustness region, which results in larger IAE values. The lack of derivative action highly contributed to the deviation from optimality. Since, $1.3 < M_S < 2$, the SIMC tuning rules should be preferred. Nonetheless, it should be mentioned that when the ratio τ_1/ϑ is large, both tuning approaches show substantial divergence from the optimal.

In addition to that, second-order processes with time delay were investigated. The ratio between the time-constants and the time delay varied. The SIMC and the K-SIMC tunings give close to optimal controllers for most of the cases. Both approaches essentially find the same controller parameters. The desired closed-loop time constant, τ_c or λ, is relatively small. Hence, the tuning rule which the K-SIMC rules propose for the integral and the derivative time give almost the same tuning parameters to the SIMC for an ideal PID controller. When the second time constant is small compared to the time delay, the I-SIMC tuning rule should be used. It suggests that the derivative time should be chosen as the one third of the time delay. A dramatic improvement in terms of performance was observed when the I-SIMC tunings were implemented. For the examined robustness region, again the SIMC and the K-SIMC tunings give the same results, so the choice doesn't really matter. However, for processes, where the second time constant is very small the I-SIMC tunings are the way to go due to the advantages of the larger derivative action.

Furthermore, two more processes were introduced. The first one had negative numerator time constants while the second one had only positive numerator time constants. It is evident, that the more approximations are made the more deviation for the optimal performance will be. In both cases, the SIMC approximations outperformed the K-SIMC approximations. More specifically, as far the second-order approximation is concerned, the half-ruled SIMC controller surprisingly presents very close to optimal performance, although two approximations have already been done. The Half-rule suggests that when approximations are made it is preferred to add more on the time constant than the time delay, which is the opposite of what the K-SIMC approximations propose. All in all, the Half-rules should be the the most suitable choice, when an approximation on a higher-order model must be done. Therefore, a conclusion might be that time delay contributes more to performance behavior than the time constant.

Although, the SIMC rules seem to perform better than the K-SIMC, model uncertainty should also be considered. Almost always in practice, the exact model is not known. A relatively good M_S is chosen for a specific process and the resulting controller settings were applied to other processes with different characteristics, where process input and output step disturbances were added. Since, the K-SIMC rules propose a PI controller it was expected that the PI controller would behave better than the PO and SIMC ideal PID controllers when the model is significantly different. That was proven to be the case, and even when a LHP was added the K-SIMC controller was still be able to handle it. However, similar results would have arisen when a SIMC or PO PI controller would have been implemented. Therefore, no clear conclusions can be made.

Except for the SIMC and the K-SIMC, the tuning rules proposed by Syrcos & Kookos where investigated. Different time delay values were tested and the resulting controller were compared with the SIMC and the optimal. The two former tuning methodologies resulted in very efficient controllers with sufficient stability margins. It needs to be mentioned, that Syrcos & Kookos controllers present substantially larger delay margins. Hence, although their method is not valid for every time delay value, when that is the case, their settings might be the better choice. Moreover, ideal PID controllers were calculated through the MATLAB Tuning Toolbox. They were compared to the optimal and the SIMC. The Toolbox controllers perform very well for the larger time delay values. On the other hand, for small time delay values, MATLAB chooses to propose very robust controllers, which result in very poor performance. Very often, MATLAB suggested a PI controller instead of a PID which resulted in poorer performance. Thus, the MATLAB Toolbox should not be chosen for lag dominated processes.

Finally, a practical example was added. A thermal/optical plant was modeled and optimal, SIMC and K-SIMC controllers were implemented. Although, the PID controllers provide marginal performance advantages compared to the PI, their use is not recommended, since they require substantially higher input usage.

Future work

When it comes to PID controller tuning there are many things that could be investigated. Some of them follow

- Controllers with very large sensitivity peak values can be investigated. The tuning rules could more extensively be tested and one could be involved in set-point filter design which could be used. However, most of the times the controller tuning rules should not be associated with filter design.
- SIMC and K-SIMC controllers in serial-form might present interesting results, since here only PID controllers in parallel form were examined. Also, SIMC PI controllers could be added.
- Other approximation rules which were not used could be tested.
- One might try to expand the Syrcos & Kookos tunings for smaller and larger time delays and compare them to the optimal.

Bibliography

[1] Åström, K. J. ; Hägglund, T. (1994) Advanced PID Control. *ISA-The Instrumentation, Systems and Automation Society.*

[2] Åström, K. J. ; Panagopoulos, H. ; Hägglund, T. (1998) Design of PI-controllers based on non-convex optimization. *Automatica.* 34, 585-601.

[3] Balchen, J. G. (1958) A performance index for feedback control systems based on the Fourier transform of the control deviation. *Acta Polytechnica Scandinavica.*

[4] Boyd, S. ; Barratt, C. Linear Controller Design: Limits of Performance. *Prentice-Hall.* 1991.

[5] Foss, M. S. (2012) Validation of the SIMC PID Tuning Rules. *Technical Report.* Available at: http://www.nt.ntnu.no/users/skoge.

[6] Grimholt, C. ; Skogestad, S. Optimal PI-control and verification of the SIMC tuning rule. *IFAC conference on Advances in PID control (PID'12).* The International Federation of Automatic Control, March 2012.

[7] Grimholt, C. ; Skogestad, S.(2013) Optimal PID-control on first order plus time delays systems and verification of the SIMC rules. *10th IFAC International Symposium on Dynamics and Control of Process Systems.*

[8] Grimholt, C. ; Skogestad, S.(2016) Optimization of fixed-order controllers using exact gradients. *unpublished, submitted to Journal of Process Control.*

[9] Grimholt, C. ; Skogestad, S. (2016) Optimal PID control of double integrating processes. *11th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems.* Trondheim.

[10] Grimstad, B. (2009) Studies in static output feedback control.*Master Thesis.* Available at: http://www.nt.ntnu.no/users/skoge.

[11] Hägglund, T. Signal Filtering in PID Control. *IFAC conference on Advances in PID control (PID'12).* The International Federation of Automatic Control, March 2012.

[12] Hazebroek, P. ; Van der Waerden, B. L. (1950) The optimum tuning of regulators *Trans. ASME*, 72, 317-322.

[13] Holene A. L. (2013) Performance and Robustness of Smith Predictor Control and Comparison with PID Control. *Master Thesis.* Available at: http://www.nt.ntnu.no/users/skoge.

[14] Kookos, I. K.; Koutinas, A. A. Optimization of processes and systems with applications in MATLAB and GAMS. *Tziolas Publishing.* 2013.

[15] Kravaris, C. (2014) Notes on Process Dynamics and Control Course. *Department of Chemical Engineering. University of Patras.*

[16] Lee, J. ; Cho, W. ; Edgar T. F. (2014) Simple Analytic PID Controller Tuning Rules Revisited. *Ind. Eng. Chem. Res.*, 53, 5038-5047.

[17] O 'Dwyer, A. Handbook of PI and PID Controller Tuning Rules, 2nd edition. *Imperial College Press.* 2006.

[18] Panagopoulos, H. ; Hägglund, T. ; Åström, K. J. The Lambda Method for Tuning PI Controllers. *Internal Report. Department of Automatic Control. Lund Institute of Technology*, August 1997.

[19] Rivera, D. E.; Morari, M. ; Skogestad, S.(1986) Internal Model Control. 4. PID Controller Design. *Ind. Eng. Chem. Process Des. Dev.*, 25, 252-256.

[20] Seborg, D.E. ; Edgar, T.F. ; Mellichamp, D.A. ; Doyle, F.J. Process Dynamics and Control *3rd ed. ; John Wiley and Sons.* New York, 2010.

[21] Shinskey, F. G.(1990) How good are our controllers in absolute performance and robustness? *Measurement and Control*, 23, 114-121.

[22] Shinskey, F. G. ; Process-Control Systems, 2nd edition. *McGraw-Hill.* February, 1979.

[23] Skogestad, S.(2003) Simple analytic rules for model reduction and PID controller tuning. *Journal of Process Control*, 13, 291-309.

[24] Syrcos, G. ; Kookos, I. K.(2005) PID controller tuning using mathematical programming. *Chem. Eng. Res.*, 44, 41-49.

[25] Stephanopoulos, G. Chemical Process Control: An Introduction to Theory and Practice. *Prentice-Hall.* 1984.

[26] Skogestad, S. ; Postlethwaite, I. Multivariable Feedback Control. *John Wiley and Sons.* 1996.

[27] Ziegler, J. G. ; Nichols, N. B. (1942) *Optimum settings for automatic controllers.* trans. ASME, 64.

Appendix A

MATLAB code

Optimization

The following MATLAB script, is the script used to find the Pareto-Optimal controllers. This script calculates an optimal ideal PID controller for **input** and **output** disturbances. Two more optimal controllers should be found before. One optimal ideal PID only for input disturbances and an optimal ideal PID controller only for output disturbances. Since the full code is too long to be added, the most important point are selected and presented.

Listing A.1: Specifying the problem, for which the optimal controller should be found

```
%% Main script for generating trade-off curves
% Written by Chriss Grimholt in 2015 to calculate an optimal PI controller for input disturbances.

% Modified by Iosif Pappas in Spring 2016 to calculate the optimal ideal PID controller for input and output disturbances.

% Define the Laplace variable
s = tf('s');

%Set the process model, where k is the process gain, y the time delay and t
%the process time constant.

G = k*exp(-y*s)/(ts+1)

% PID controller, p = [kp + ki + kd]
K = @(p) (p(1) + p(2)/s + p(3)*s);

% The derivative of the ideal PID controller
dK = @(p) [tf(1); 1/s; s];

```

```
% Specify the robustness constraint
problem.ms_eqcon =  1.59;
problem.mt_eqcon = problem.ms_eqcon;

% Specify the initial guess
problem.x0 = [kc ki kd];

% Parameters for the step simulation and the gridding of the frequncy domain
problem.time_gridding = linspace(0,20, 1e4);
problem.freqency_gridding = logspace(-10,10,1e4);


% Create the transfer functions

% The Gang of Four
S = @(p) 1/(1+G*K(p));
GS = @(p) G*S(p);
KS = @(p) K(p)*S(p);

% The gradient of the error
de_dy = @(p) -GS(p)*S(p)*dK(p);
de_di = @(p) -GS(p)*GS(p)*dK(p);
```

Listing A.2: Solving the optimization problem

```

%In this case too, Chriss Grimholt in 2015 wrote the original script and was accordingly modified by Iosif Pappas in 2016 for the purposes of this thesis.

%This script solves the problem specified.


%Solving the problem
opt = optimset('Algorithm','active-set','GradConstr','on','GradObj','on','DerivativeCheck','off','Display','off');;

[res.tuning, res.cost, res.exitflag, res.output, res.lambda] = fmincon(...
    @(x) obj_fun_grad(x), ... %cost function
    x0, ... % initial guess
    [], ...
    [], ...
    [], ...
    [], ...
    [], ... % lower bounds
    [], ... % upper nounds
    @(x) nonlin_con(x), ... % constraints
    opt);

%Objective and gradient of the objective function

    function [J,g_J] = obj_fun_grad(x)

        S  = S_fun(x);
        e_dy = step(S,t);

        GS = GS_fun(x);
        e_di = step(GS, t);

        J_1 = trapz(t, abs(e_dy));
        J_2 = trapz(t, abs(e_di));
        J =  0.5∗((1/f_1)∗J_1 + (1/f_2)∗J_2) ;


        if nargout > 1;
            dSs = step(de_dy_fun(x), t);
            dGSs = step(de_di_fun(x), t);

            g_J_1 = trapz(t, bsxfun(@times, sign(e_dy), dSs));
            g_J_2 = trapz(t, bsxfun(@times, sign(e_di), dGSs));
            g_J =  0.5∗((1/f_1)∗g_J_1 + (1/f_2)∗g_J_2);

        end
    end

    %Constraints and gradients of the constraints

    function [c, ceq, g_c, g_ceq] = nonlin_con(x)

        ceq=[];
        nx = length(x);

        % Find corresponding frequency responce
        Sw = squeeze(freqresp(S_fun(x),w));
        Sm = abs(Sw);

        Tw = 1 − Sw;
        Tm = abs(Tw);

        cs = Sm - ms_eqcon;
        ct = Tm - mt_eqcon;

        c=[cs,ct];

        if nargout > 2;


            g_cs = zeros(nx,length(Sm));
            g_ct = zeros(nx,length(Tm));
            g_ceq = [];

            Gw = squeeze(freqresp(G, w));
            dKw = squeeze(freqresp(dK_fun(x), w)).';

            for i = 1:nx

              g_cs(i,:) = -Sm.*real(Sw.*Gw.*dKw(:,i));
              g_ct(i,:) = Tm.*real((Sw).^2./Tw.*Gw.*dKw(:,i));

            end

            g_c = [g_cs, g_ct];

        end
    end
```

Other tuning methods

The following script calculates the ideal PID controller parameters for the positive approximations process. Similar are the scripts for all the other tunings, where only the tuning rules must be changed.

Listing A.3: Half-rule approximations and SIMC ideal PID parameters calculations for a positive numerator time constant process.

```

%Written by Iosif Pappas in Spring 2016.
%The script approximates a high-order process model to a second-order
   process with time delay.
%In addition, it calculates the ideal PID SIMC parameters which satisfy
   the desired Ms values. Finally the resulting PID settings can be sent
   to a Simulink file to calculate the IAE.

```

```
tic
%Set the Laplace variable
s=tf('s');


%Original Model G(s) = ((20*s+1)*(14*s+1)*exp(-theta*s))/((42*s+1)*(5*s+1)*(2*s+1)*(s+1))

%process time constants
To_1  = 20;
To_2  = 14;
tau_1 =42;
tau_2 =5;
tau_3 =2;
tau_4 =1;

%counters
p=0
j=0

%specify the desired Ms value
for ms = 1.29:0.01:2
p=0
j=j+1

for tc = 0:0.01:5
p=p+1;

%Half-rule for (14*s+1)/(5*s+1)
if (To_2>=tau_2) && (tau_2>=tc)
g1(p,1) = To_2/tau_2;
elseif (To_2>=tc) && (tc>=tau_2)
g1(p,1) = To_2/tc;
elseif (tc>=To_2) && (To_2>=tau_2)
g1(p,1)= 1;
else
g1(p,1)=1
end

%Half-rule for (20*s+1)/((42*s+1)
if (tau_1>=To_1) && (To_1>=5*tc);
h = To_1/tau_1;
H(p,1) = (g1(p,1)*h*exp(-1*s))/((2*s+1)*(s+1))
q=0;
k=1;
tau1=2;
```

```
tau2=1;
kp=g1(p,1)*h;
else
tau = min(tau_1,5*tc);
q = tau-To_1;
v = tau/tau_1;

if (q>2)
w = q;
H(p,1) = (g1(p,1)*v*exp(-1.5*s))/((w*s+1)*(2.5*s+1));
k=1.5;
tau1=w;
tau2=2.5;
kp=g1(p,1)*v;
elseif (q<2)&&(q>1)
H(p,1) = (g1(p,1)*v*exp(-1.5*s))/((2*s+1)*((q+0.5)*s+1));
k=1.5;
tau1=2;
tau2=q+0.5;
kp=g1(p,1)*v;
elseif (q<1)
H(p,1) = (g1(p,1)*v*exp(-(1+q/2)*s))/((2*s+1)*((1+q/2)*s+1));
k=1+q/2;
tau1=2;
tau2=1+q/2;
kp=g1(p,1)*v;
end

end

% end
G(p,1) = H(p,1);

kc = ((tau1)/(kp*(tc+k)));
taud = tau2;
%set integral time
if tau1 < (4*(tc+k));
ti=tau1;
else
ti = 4*(tc+k);
end
f = 1 + (taud/ti);
kc = kc*f;
ti = ti*f;
taud = taud/f;

```

```
%Calculate the Ms
[G_n G_d] = tfdata(G(p,1),'v');
delay = totaldelay(G(p,1));
w = logspace(-10,10,10^4);
GG_no = conv([kc*ti*taud kc*ti kc],G_n);
GG_de = conv([ti 0],G_d);
GG_w1 = polyval(GG_no,w*1i);
GG_w2 = polyval(GG_de,w*1i);
delay_w = exp(-delay*w*1i);
G_G = GG_w1./GG_w2.*delay_w;
S = 1./abs(1+G_G);
MS_S = max(S);

%save the values which satisfy the requirements
%if Ms = 1.59 save the controller parameters
if (abs(ms-MS_S)<1e-4)

break;

%print the results
end
b(j,1) = kc;
b(j,2) = ti;
b(j,3) = taud;
b(j,4) = tc;
b(j,5) = k;
b(j,6) = MS_S;
b(j,7) = ms;
b(j,8) = p;

end
end

%If needed, calculate the IAE in Simulink
for p=1:1:71
SimOut = sim('Test');
h(p,1) = output1;
h(p,2) = output2;
end

toc
```

Plots

The Performance vs. Robustness plots are created using the MATLAB script which follows. Of course, suitable changes for each specific case have been made.

Listing A.4: Typical script used for plot creation.

```matlab

% Written by Iosif Pappas in Spring 2016, to create Performance vs Robustness graphs
p1=plot(num(31:1:101,1),num(31:1:101,2),'Color',[0.5 0 0.5])
hold on
p2=plot(num(31:1:101,1),num(31:1:101,3),'Color',[0 0.5 0.5])
hold on
p3=plot(num(31:1:101,1),num(31:1:101,4),'b')
hold on
p4=plot(num(44,1),num(44,3),'x','Color', [0 0.5 0.5])
hold on
p5=plot(num(69,1),num(69,4),'xb')
hold on

%Labels
ylabel('\textbf{Performance, J(c)}', 'Interpreter','LaTex')
xlabel('\textbf{Robustness,  $\bf{M_S}$}','Interpreter','LaTex')
%Axis limits
ylim([0.5 5])
xlim([1.3 2])
%Font
set(gca,'FontSize',10,'Fontweight','Bold');
legend('Optimal', 'SIMC', 'K-SIMC')
set(p1,'LineWidth',1.5)
set(p2,'LineWidth',1.5)
set(p3,'LineWidth',1.5)
set(p4,'LineWidth',1.5)
set(p5,'LineWidth',1.5)
legend([p4,p5],'\tau_c  =  \theta','\lambda  =  \theta', 'Location','SouthWest')
legend([p4,p5],'\tau_c  =  \theta','\lambda  =  \theta','Location',[0.68 0.4 0.15 0.05])

%Latex annotations
str = '$$G\left ( s \right ) = \frac{e^{-s}}{\left ( 10s+1 \right )} $$';
text(1.7,4.1,str,'Interpreter','LaTex')
gtext('Pareto Optimal','Color', [0.5 0 0.5])
gtext('SIMC','Color', [0 0.5 0.5])
gtext('K-SIMC','Color', 'b')
```

Appendix B

SIMULINK

The SIMULINK flow sheets are also added. The SIMULINK model belows calculates the integrated absolute error for input and output disturbances for an example process. Please, keep in mind that from a feedback point of view, output disturbances are equivalent to a step point change.

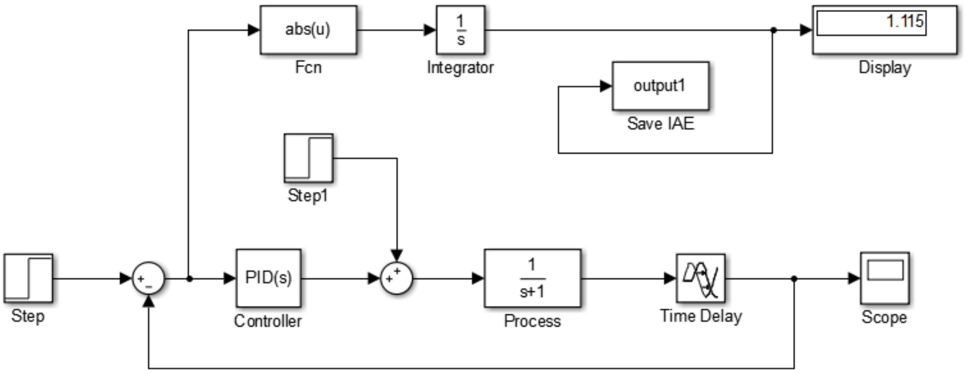


Figure B.1: SIMULINK model to calculate the integrated absolute error for input and output disturbances.

In addition to that, the main SIMULINK flow sheet of the udaq28/LT thermal/optical plant is presented it Figure B.2.

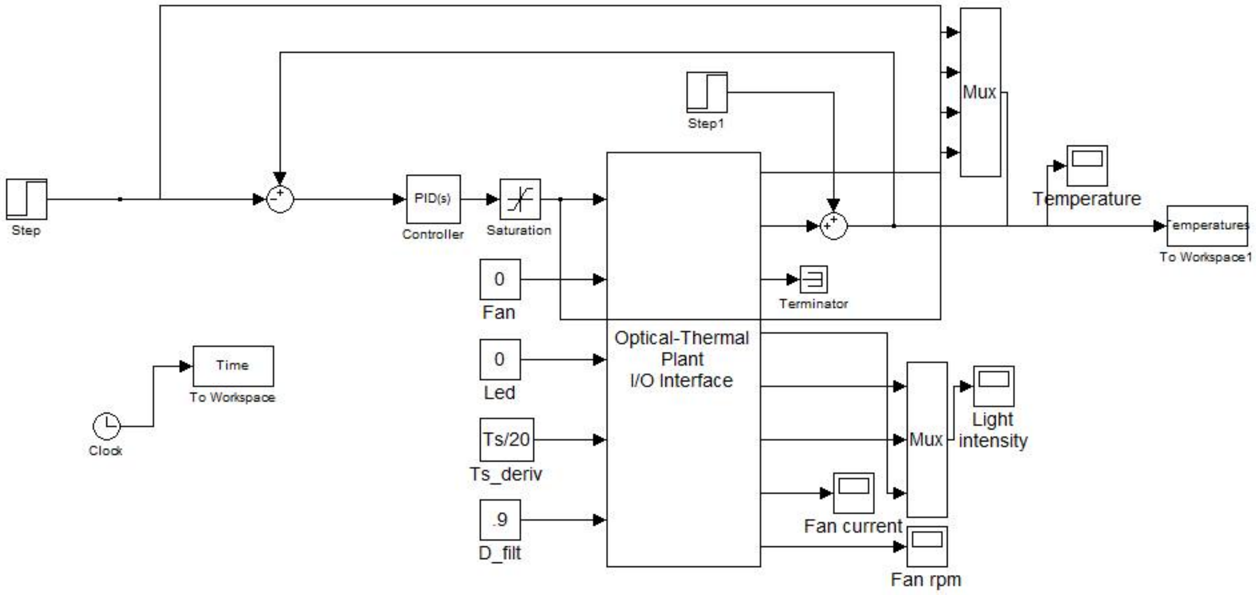


Figure B.2: SIMULINK model to calculate the integrated absolute error for input and output disturbances.