NTNU – Trondheim
Norwegian University of
Science and Technology

Performance and Robustness of Smith Predictor Control and Comparison with PID Control

Axel Lødemel Holene

Chemical Engineering and Biotechnology
Submission date: June 2013
Supervisor: Sigurd Skogestad, IKP
Co-supervisor: Chriss Grimholt, IKP

Norwegian University of Science and Technology
Department of Chemical Engineering

Axel Lødemel Holene

Performance and Robustness of Smith Predictor Control and Comparison with PID Control

Master's thesis
for the degree of Master of Technology

Trondheim, June 2013

Norwegian University of Science and Technology
Faculty of Natural Sciences and Technology
Department of Chemical Engineering

NTNU

Norwegian University of Science and Technology

Master's thesis
for the degree of Master of Technology

Faculty of Natural Sciences and Technology
Department of Chemical Engineering

ISBN N/A (printed version)
ISBN N/A (electronic version)
ISSN 1503-8181

Master's theses at NTNU, 2013:N/A

Printed by NTNU-trykk

ABSTRACT

The performance and robustness of the Smith predictor controller has been tested for first and second order processes with time-delay by comparing with PI and PID control. It was assumed that the performance would obviously be better when the time-delay is known.

Pareto optimal PI and PID controller tuning curves has been found. The processes studied is divided between five first-order-plus-time-delay and nine second-order-plus-time-delay models. Performance have been defined as a weighted average of the integrated absolute error for a step load change in input and output disturbances. Robustness has been defined in terms of the maximum peak of the sensitivity function (M_S).

The Pareto optimal Smith predictor tuning curves have been found. Deterioration in performance and robustness have been evaluated when the process time-delay deviate from the nominal time-delay for which the controllers are optimal.

SIMC tuning curves have been compared to the Pareto optimal PI and PID tuning curves, and a method for applying SIMC tuning to Smith predictor controllers have been suggested.

The Smith predictor have proven to yield small performance enhancements compared to optimal PI and PID control on second-order processes. For first-order processes the optimal PI and PID controllers have performance superior to the Smith predictor. When the process time-delay varies, even Smith predictor controllers with modest M_S values tend to destabilise. No reason are found for utilising a Smith predictor PI or PID controller over a Pareto optimal PI or PID controller. One will have to tune the Smith predictor to yield very low M_S values to avoid the instability issues when modelling error in the time-delay parameter occurs. The potential increase in perfor-

mance achieved is easy to compensate with a regular PI or PID controller by tuning it a little tighter.

PREFACE

"A sleeping student commits no sin. A sleeping student doesn't learn a darn thing about thermodynamics either."

— Tore Haug Warberg
Associate professor, NTNU

How do you explain to your mother the concept of optimisation, and why it is difficult to find global solutions in a non-convex world?

If you drop a small rubber ball in the bathtub, you know exactly where it will end up —— in the drain. Your bathtub is convex; all descent paths lead to the drain, which is the optimal point. Otherwise, it's a really incompetent bathtub. Now, imagine you are on a plane, flying over some mountain range. You close your eyes, and you throw out the rubber ball somewhere on the way. What do you think are the chances of that ball finding its way to the absolute deepest valley?

This thesis is the result of fabulous teamwork. It started out as a mission to quantify how much performance one would loose by *not* applying a Smith predictor to processes with time-delay. It ended up being somewhat of a slaughter of the Smith predictor, based on its instability when modelling error in the time-delay parameters occur.

To the reader: Beware! The amount of graphical results in this thesis is too large. Have patience when reading.

On my way through this process, I have fought battles against myself and my motivation, but mainly against MATLAB. The program does indeed carry a lot of convenient tools for simulation and analysis of control problems. But it has its limitations. This became very evident when I asked my everyday

supervisor what one would name a stable process that oscillate to infinity. His answer was “a contradiction”, and it was discovered that MATLAB didn’t return the correct gain and phase margins for the obviously unstable system.

I am forever grateful for the superb guidance and utmost impressive level of patience exercised by my everyday supervisor, Chriss Grimholt. This work had not been possible to carry out without him. I would also thank my main supervisor Sigurd Skogestad for his impressive level of knowledge. Whenever questions arose that were difficult to answer, an explanation were always to be found in his office. Vinicius de Oliveira has helped me out several times, even though I’m not his responsibility.

Hanne Sørgård, being a “sivilingeniør” in biotechnology, and still helping me out proofreading this thesis in process system theory; thank you for being strict on my language!

I would never have been at this point in life, completing a Master of Science, without my guardian angels in high school: Kristoffer Hellton, a fellow student and now PhD candidate at the University in Oslo, and Cathrine Tellefsen, my maths and physics teacher. You encouraged me to pursue my curiosity towards the natural phenomena, and you demanded that I worked with the science subjects. Thank you!

My social life and personal development have been covered by the chemistry union at Gløshaugen, Høiskolens Chemikerforening. I will always return.

Last, but not least: my partner, Kaja Sørgård Eriksen, deserve a prize. I have spent a lot of hours in the office, and a lot of hours having MATLAB or LaTeX on my mind. Even so, she has listened to my complaints, thoughts, frustration and delights, and also proofread a lot of my material. You were right: I couldn’t have done this without you. I will be forever thankful.

Declaration of Compliance

I declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology (NTNU).

Axel Lødemel Holene
Trondheim, June 2, 2013

Contents

List of Symbols

Alternative Controller Parameterisation Symbols

D	Derivative time	[s]
I	Integral time	[s^{-1}]
K	Controller gain	
P	Proportional action	

Mathematical Symbols

$\triangleq$	Equal by definition	
$\mathscr{H}_\infty$	Maximum matrix norm	
$\mathrm{Im}(z)$	Imaginary part of a complex number z	
$\mathscr{L}$	Laplace transform	
$\mathbb{C}$	Set of complex numbers	
$\mathbb{R}$	Set of real numbers	
$\mathscr{M}_S$	Set of M_S values	
$\mathscr{M}_T$	Set of M_T values	
ω	Phase	[rad s^{-1},deg s^{-1}]
ω_c	Gain crossover frequency	[rad s^{-1},deg s^{-1}]
$\mathscr{Q}$	Set of time-delay errors	
$\mathrm{Re}(z)$	Real part of a complex number z	
$f(\dot{})$	General function.	
i	Imaginary unit, $i^2 = -1$	
s	Laplace transform variable	
z	Complex number $z = a + ib$	

Parallel Parameterisation Symbols

$\tau_D^\star$	Derivative time	[s]
$\tau_I^\star$	Integral time	[s]
$K_c^\star$	Controller gain	

Robust Tuning Rules Symbols

β	Reference filter factor	
$K_c^\diamond$	Controller gain	
T	Lag time constant	[s]
T_i	Integral time	[s]
$T_\circ$	Filter numerator time constant	[s]
T_1	Denominator filter time	[s]

Cascade/SIMC Tuning Symbols

τ_D	Controller derivative time	[s]
τ_I	Controller integral time	[s]
K_c	Controller gain	
τ_c	Closed-loop time constant	

Control Theory Symbols

α	Derivative filter constant
$\bar{p}$	Steady-state controller output
K	Controller transfer function
$\widetilde{K}$	Smith predictor transfer function
d_i	Input signal disturbance
d_o	Output signal disturbance
$\Delta\theta$	Process time-delay deviation from model
$\delta\theta^+$	Upper time-delay error boundary
$\delta\theta^-$	Lower time-delay error boundary
e_i	Controller input
e	Output-reference error
F	Reference signal filter
G	Approximated process transfer function
G	Process transfer function
G_{d_i}	Input disturbance transfer function
GM	Gain margin
$G_\circ$	Process transfer function with no delay
$\widetilde{G}$	Smith predictor process model without time-delay
$\widetilde{G}$	Smith predictor process model
G_{d_o}	Output disturbance transfer function

IAE	Integrated absolute error	
IAE_{d_i}	IAE for a step output load disturbance	
$\text{IAE}^{\circ}_{d_i}$	Reference value for IAE_{d_i}	
IAE_{d_o}	IAE for a step input load disturbance	
$\text{IAE}^{\circ}_{d_o}$	Reference value for IAE_{d_o}	
L	Closed loop system transfer function	
M_S	Maximum sensitivity peak value	
M_T	Maximum complementary sensitivity peak value	
n	Measurement noise	
p	Controller output	
PM	Phase margin	[rad,deg]
$\widetilde{p}$	Smith predictor controller output	
r	Reference/set-point	
S	Closed-loop sensitivity function	
τ_1	Dominant lag time constant	[s]
τ_2	Second-order lag time constant	[s]
τ_F	Derivative filter time constant	
τ^{inv}	Inverse response time constant	
$\theta_{\circ}$	Smith predictor time-delay model	
T	Closed-loop complementary sensitivity function	
θ_{max}	Maximum additional time delay prior to instability	
$\theta_{\circ}$	Nominal time delay	[s]
$\tilde{u}$	Process input from Smith predictor	
u	Process input	
y	System output	
y_m	Measured output	
r	Reference/set-point	
k	Process/plant gain	
CL	Closed-loop transfer function	

Miscellaneous Symbols

t_{SimTime}	Simulation time	[s]

List of Figures

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Outset

The Russian American engineer Nicolas Minorsky published in 1922 an article on automatic steering of ships. His analysis was based on observations of a helmsman, where Minorsky noted that the helmsman controlled the ship not only based on the current error, but also on previous error and the ship's current rate of change. His goal was stability, and while proportional control provides stability against small disturbances, dealing with steady-state disturbance require an integral term. A derivative term was later added to enhance the control. Minorsky was with Maxwell, Routh, and Hurwitz one of the pioneers in the discussion of control theory — he theoretically showed the value of what today is known as the proportional-integral-derivative (PID) controller structure (Bennet, 1984, 1993).

When a driver drives his car, he is aware of the oncoming curve of the road. He then performs the necessary control actions, and safely maneuvers the car through the curve. A Smith predictor uses a similar concept for the prediction of future behaviour of a process with time-delay. If the prediction of the behaviour of the car is misleading, the result can be quite disastrous for both the driver and the car. Predicting the process behaviour should improve the overall performance, but could this lead to robustness issues? What will happen if the velocity of the car differs from what the driver anticipates?

Thesis Scope

Grimholt and Skogestad (2012) have assessed the optimality of the Simple Internal Model Control (SIMC)[a] tuning rules (Skogestad, 2003) for proportional-integral (PI) control of a set of first-order-plus-time-delay (FOPTD) processes. By evaluating the trade-off between performance and robustness for both optimal PI tuning the PI controller achieved by applying the SIMC tuning rules, the optimality of the SIMC tuning rules have been quantified. The utilised system was a one-degree-of-freedom negative feedback control scheme. Performance was evaluated by considering a weighted average of the absolute integrated error (IAE), while the robustness was measured in terms of the maximum peak value of the sensitivity function for the system (M_S). The feedback control system was then tested for disturbance rejection of a step load change in input and output. Grimholt and Skogestad (2012) report the SIMC tuning rule to yield good settings for disturbance rejection, except for pure time-delay processes.

This thesis is a natural extension of the work of Grimholt and Skogestad, expanding the scope to second-order-plus-time-delay (SOPTD) processes. Optimal PI and PID controller solutions are found for a set of first and second order process models with time-delay. Corresponding PI and PID controller tunings are found from the SIMC tuning rule. A previous project in the field have been performed by Foss (2012).

In industry, the cascade controller parameterisation is extensively used. Different parameterisations are inspected to quantify potential deviation from optimality when using a cascade parameterisation over a parallel.

The performance and robustness of a Smith predictor is tested for processes with time-delay, by comparing it with Pareto optimal PI and PID control. The performance is assumed better if the time-delay is known. Optimal PI and PID controllers are determined for the Smith predictor, and the closed-loop stability when modelling error in the time-delay parameters occurs is investigated. The illustrative example used by Adam, Latchman, and Crisalle (2000) to illustrate discontinuities in the Smith predictor stability domain is discussed, and possible instability is examined for a range of time-delay errors. Previous work on the topic has been carried out by Paulsen (2012). This work is perceived as unaccomplished, and will not be further addressed.

The possibility of using the SIMC tuning rules for Smith predictor tuning is studied, and the SIMC tuning rules is compared with a set of robust tuning rules given by Normey-Rico and Camacho (2007).

a) The “S” could also denote “Skogestad”.

The performance and robustness definitions in (Grimholt and Skogestad, 2012) is pursued. The phase margin is used to quantify the maximum allowed extra dead-time before instability occurs for the Smith predictor tuning cases.

The objective of this work is to compare the performance of optimal PI and PID control with the performance of optimal Smith predictive control. How optimal is the Smith predictor compared to the regular PI and PID control? What are the benefits and disadvantages of utilising the Smith predictor control structure over the simple and safe PID control structure? Will the SIMC tuning rules yield good results when tuning a Smith predictor?

Stay tuned!

Process Control and Optimisation

The phenomena of process control is ancient. Actually, it is as old as life itself. Regulatory mechanisms are essential in biological systems, and one could ask how human interactions would be possible without the concept of feedback?

From an industrial point of view, most processes need to be controlled. Without some sort of regulatory mechanism one cannot guarantee neither product quality, safety for humans, equipment or environment, nor good economic outcome for the process. With regards to economy, one would always like to operate a process at its optimum, whether that may be the optimal pressure and temperature for a reactor, or the optimal airplane speed with respect to fuel consumption and the fact that "time is money". The airplane example introduces an optimisation concept presented in Chapter 3. The overall optimality of a problem is here governed by two (or more) conflicting objectives. This is called multi-objective optimisation, with the resulting solutions called Pareto[b] optimal solutions. Such a problem arises when tuning PID controllers: a robust controller will not have the response necessary to yield high performance. Simultaneously, a high performance controller have low robustness to disturbances. Thus one will always have to make a trade-off between performance and robustness when tuning PID controllers.

b) After Vilfredo Pareto (1848-1923), an Italian economist who applied the concept in economic efficiency and income distribution studies.

PID Control

The PID controller is a structure with *Proportional*, *Integral* and *Derivative* action, and is almost universally used in industrial control (Goodwin, Graebe, and Salgado, 2001; Normey-Rico and Camacho, 2007). The controller has three parameters associated with it: gain (K_c), integral time (τ_I) and derivative time (τ_D). The PID controller design is built for simplicity, and its effect on the signal are concepts which are easy to understand.

Proportional action Often referred to as controller gain[c], is proportional to the error signal. Sensitivity to deviation between reference and output is adjusted with the gain.

Integral action The controller output depends on the integral of the error over time, and is necessary for eliminating steady-state offset.

Derivative action Anticipating the future behaviour of the error signal by evaluating the time derivative of the error signal.

The mentioned parameters impose a distinct effect on the controller input signal, while the overall output signal is also dependent on the internal signal structure of the controller. Common parameterisations are parallel and series signal processing. The series controller is also known as a cascade controller.

The different parameterisations of the PID controller can yield different tuning solutions. While the zeros[d] of a parallel PID controller can have both real and complex solutions, the zeros of the cascade controller can only be located in the set of real numbers. This is proven in Appendix J. If the optimal controller for a given process problem is located in the complex subdomain of a parallel controller configuration, and a cascade controller structure is used for the optimisation procedure, the resulting controller cannot be truly optimal.

The Smith Predictor

The main problem with controlling dead-time systems is associated with the time lapse before the effect of the disturbances, or the control actions, are experienced. The controller will be attempting to correct a situation that happened backwards in time. The performance of closed-loop systems with

c) This depends on the controller parameterisation, further studied in Chapter 2.3.

d) Zeros: values for s such that $K(s) \to 0$. Poles: values for s such that $K(s) \to \infty$

significant time-delay can be improved by applying a predictor correction structure (Normey-Rico and Camacho, 2007).

The Smith predictor consists of a regular controller and a model of the dead time process that is to be controlled. The purpose of the Smith predictor is to allow the controller to observe the expected process response before it occurs, thus cancelling the effect of the delay on the closed loop dynamics.

Adam et al. (2000) has examined the robustness of the Smith predictor with respect to uncertainty in the time-delay parameter. While the Smith predictor can achieve improved control of processes with delay, some unintuitive behaviour is observed when there is modelling mismatch between the time-delay in the process and the process model used by the Smith predictor. Instability is reported in some cases when the real time-delay is less than the modelled time-delay. As Smith predictors often are designed using an overestimated delay, discontinuities in the stability domain can pose serious service problems.

SIMC Tuning Rules

Although the PID controller has only the three mentioned parameters, finding good tuning values proves challenging; Skogestad (2003) claims that a large number of PID controllers in the industry are poorly tuned. In his article "Simple Analytic Rules for Model Reduction and PID Controller Tuning" he proposes a set of analytically derived tuning rules which yield good responses for setpoint changes and disturbance rejection in both integrating and pure time-delay processes. The only tuning parameter in the SIMC tuning rules, the closed-loop time constant denoted τ_c, is by Skogestad recommended to be set equal to the time-delay to yield a controller with fast response and good robustness. This assertion is examined and put in context with the proposed measures of performance and robustness in this thesis. The SIMC rules are also modified with the aim of tuning the Smith predictor controller structure.

Thesis Structure

Feedback Control

The necessary theory for this study is introduced in this chapter. The feedback scheme used throughout this thesis is declared, and the corresponding loop functions are defined. Definitions for controller parameterisations are

proclaimed, and the Smith predictor controller structure is given. The SIMC tuning rules are finally accounted for.

Pareto Optimisation

This chapter describes the concept of multi-objective optimisation. The definitions of controller performance and robustness is introduced and put in context with frequency domain controller analysis. The complete optimisation problem is formulated, and simulations and algorithms are summarised. The objective function is graphically represented for several cases to give an impression of the nature of the optimisation problem.

First Order Processes

In this chapter, the Pareto optimal tuning curves for the first-order processes are found. Deviation from optimality when using a cascade PID controller over a parallel PID controller is discussed. An analysis of the Pareto optimal controller stability when variations in the process time-delay are introduced, is performed. The SIMC tuning parameters for the processes are found, and compared to the Pareto optimal PI controller solutions.

Second Order Processes

This chapter resembles the “First Order Processes” chapter. The scope is expanded to yield second-order processes.

Optimality of the Smith Predictor

In this chapter, the Pareto optimal Smith predictor PI and PID tuning curves are found. The stability of the Smith predictor controllers are examined when there is time-delay modelling error, and the true time delay of the processes differs from the nominal time-delay.

Smith Predictor Tuning Rules

In this chapter, the SIMC tuning rules are evaluated in terms of being used for tuning the Smith predictor. A set of robust tuning rules given in Normey-Rico and Camacho (2007) have been compared to the suggested SIMC rules.

Conclusion

Conclusion are drawn, and suggestions for further work are given.

Appendices

The appendices are the main source of plots in this thesis. There are a lot of them, so have patience.

CHAPTER 2

FEEDBACK CONTROL

"Learn from the past, set vivid, detailed goals for the future, and live in the only moment of time over which you have any control: now."

— Denis Waitly
Author and motivator

2.1 Outset

The water level in the cistern of the traditional water flush toilet is controlled by a floating device which controls the inflow of water through a valve. In ancient Greece, The Greek Ktesibios started using this type of feedback structure for water-level control as early as 250 B.C. (Mayr, 1970). The purpose of the regulator was to keep the water level in a tank constant, such that the outflow of the tank would fill a second tank at a constant rate. The water level in the second tank then depend only on the time elapsed. In this way an accurate method for time measurement was achieved. (Lewis, 1992).

2.2 Principles of Feedback Control

In control theory, one distinguishes between open-loop (feed-forward) and closed-loop (feedback) arrangements. The open-loop scheme has the advantage of the question of system stability being trivial: the system is stable when both the controller and the process are stable. The fundamental reasons for using feedback control, as described by, amongst others, Skogestad and Postlethwaite (2005), are summarised to be the presence of signal uncer-

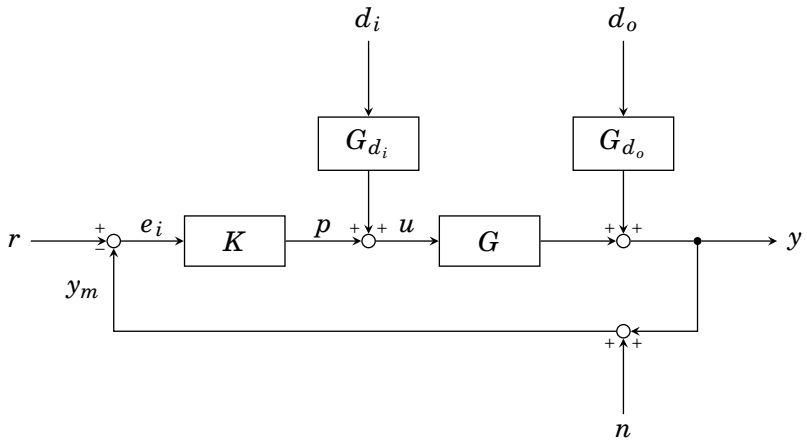


Figure 2.1 – Block diagram of the feedback control system described in (Skogestad and Postlethwaite, 2005)

tainty in the form of unknown disturbances, process model uncertainty and a potentially unstable plant.

In this work, a one-degree-of-freedom negative feedback control scheme, illustrated in Figure 2.1, is applied as described by Skogestad and Postlethwaite (2005). The controller input is $e_i = r - y_m$, where $y_m = y + n$ is the measured input, r is the reference or setpoint, y is the system output and n is the measurement noise. In the figure, K denotes the controller transfer function, while G, G_{d_i} and G_{d_o} denotes the transfer functions for the process, the input disturbance and the output disturbance, respectively. The Laplace transform variable s is omitted from the transfer functions for convenience. The controller output is p, the process input is u, and the process input and output disturbances are d_i and d_o, respectively. The error is regarded as the true output offset from the reference, $e = y - r$.

The output disturbance can be regarded as a type of setpoint change, and is often treated as such. Often, a reference filter is applied to enhance setpoint performance. The focus of this work is disturbance rejection, and as a setpoint filter removes the disturbance influence in the feedback loop, reference filter is disregarded and the notation of the output disturbance is retained.

It is assumed that the disturbance transfer functions, G_{d_i} and G_{d_o} equal unity and do not affect the disturbance signals, d_i and d_o. Thus the closed-loop response of the plant model with a one-degree-of-freedom controller can be written as Equation (2.1). The corresponding block diagram is illustrated

in Figure 2.3, and represents the system scheme used throughout this thesis.

$$y = \underbrace{\frac{1}{1+GK}}_{S} d_o + \underbrace{\frac{G}{1+GK}}_{GS} d_i + \underbrace{\frac{GK}{1+GK}}_{T} r - \underbrace{\frac{GK}{1+GK}}_{T} n. \tag{2.1}$$

Defining the functions

$$S \triangleq \frac{1}{1+GK} \quad \text{and} \quad T \triangleq \frac{GK}{1+GK}, \tag{2.2}$$

where S is the sensitivity transfer function and T is the complementary sensitivity transfer function. The influence of the disturbance, the reference and the noise signals on the output is identified as

$$u = Sd_i + KS(r - n - d_o). \tag{2.3}$$

The equivalent expression for the error is

$$e = Sd_o + GSd_i - Sr - Tn \tag{2.4}$$

The reference influences the plant input through the product KS and the error off-set through S. The complementary sensitivity indicate the noise amplification in the feedback loop. Consequently, if $T > 1$, the noise will eventually dominate the error. A schematic illustration of the frequency domain behaviour of S and T, along with a controller and the product KG, is given in Figure 2.2. In the figure, the controller is not ideal, as the reference tracking is poor and the noise is amplified ($T > S$). With regards to controller action, the KS line indicate that the controller is working heavily in the high-frequency region.

2.3 PID Controllers

The differential equation for the parallel form PID control algorithm in the time domain is

$$K(t) = \bar{p} + K_c^\star \left(e_i(t) + \frac{1}{\tau_I^\star} \int_0^t e_i(t)\mathrm{d}t + \tau_D^\star \frac{\mathrm{d}e_i(t)}{\mathrm{d}t} \right), \tag{2.5}$$

where $\bar{p}$ is the steady-state value, $K_c^\star$ is the controller gain, $\tau_I^\star$ is the controller integral time and $\tau_D^\star$ is the controller derivative time. The steady-state value is often set to zero as it only scales the values, and is further omitted.

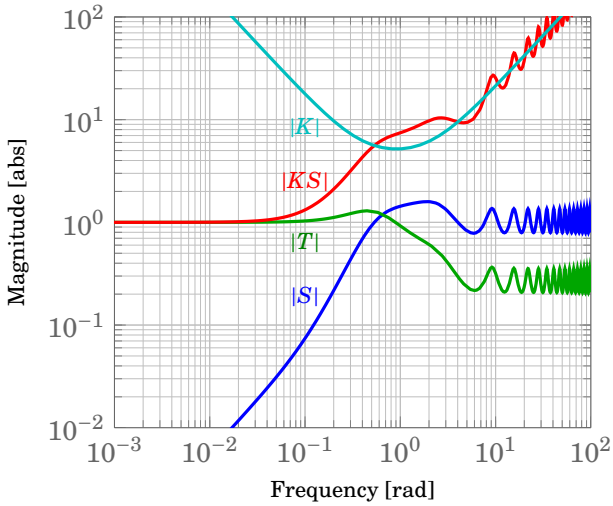


Figure 2.2 – Typical frequency plot of S, T, KS, and K. The example is for the process $G(s) = \frac{e^{-s}}{8s+1}$ with PID controller $K(s) = 4.35\left(\frac{2.52s+1}{2.52s}\right)(0.48s+1)$.

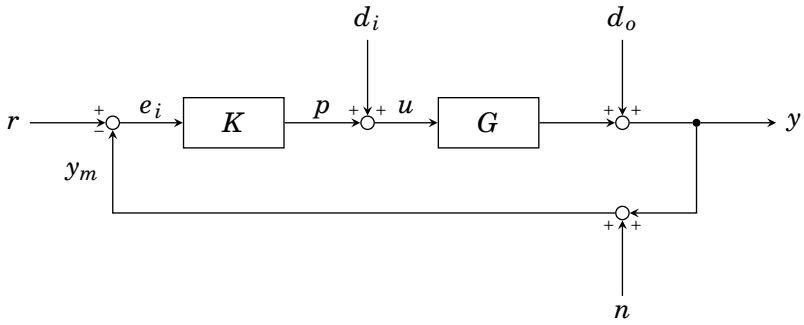


Figure 2.3 – Block diagram of the feedback control system utilised in this thesis.

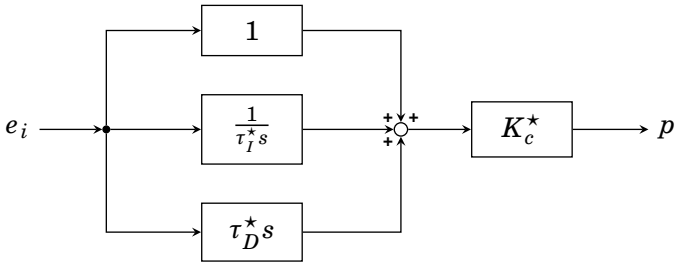


Figure 2.4 – Block diagram of the parallel form of PID control without derivative filter.

Controllers are usually displayed as a transfer function in the Laplace domain. For this thesis, the controller definition is

$$\mathscr{L}\{K(t)\}(s) = K(s) \triangleq \frac{p(s)}{e_i(s)}, \tag{2.6}$$

where $\mathscr{L}\{\cdot\}$ is the Laplace transform. The Laplace transform variable, s, is further omitted for convenience when denoting a controller in the Laplace domain.

Standard Parallel Controller

The block diagram representation of the standard parallel controller in the Laplace domain is given in Figure 2.4, with the corresponding transfer function given in Equation (2.7)

$$\frac{p}{e_i} = K_c^\star \left(1 + \frac{1}{\tau_I^\star s} + \tau_D^\star s\right). \tag{2.7}$$

Alternative Parallel Controller

An alternative representation of the parallel controller with a different set of tuning parameters is illustrated in Figure 2.5. The corresponding transfer function is

$$\frac{p}{e_i} = \mathsf{K}\left(\mathsf{P} + \frac{\mathsf{I}}{s} + \mathsf{D}s\right), \tag{2.8a}$$

$$\mathsf{P} = 1 - \mathsf{I} - \mathsf{D} \geq 0, \tag{2.8b}$$

$$\mathsf{I} \geq 0, \tag{2.8c}$$

$$\mathsf{D} \geq 0. \tag{2.8d}$$

Here, K, P, I and D represent the controller gain, proportional action, integral time and derivative time, respectively. The alternative configuration is motivated by the ability to produce pure I, D or ID-controllers without having the gain approach zero to remove the proportional term. Another advantage is that the integral term never approach infinity, which can be the case with the standard or cascade parameterisation. The parameterisation also carries the advantage of avoiding having the integral term approach infinity. The proportional term of the controller action is set by the integral and derivative time parameters, and the overall gain is used to adjust the sensitivity of the controller. The alternative parallel controller parameterisation can be transformed to the parallel controller parameterisation according to Appendix K.

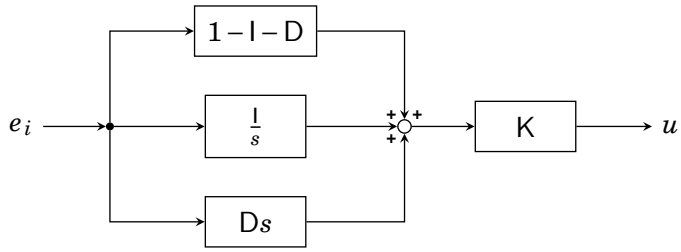


Figure 2.5 – Block diagram of an alternative parallel form of PID control without derivative filter.

Cascade Controller

In times past, when all controllers were analogue, it was convenient to have the PI and the PD element in series. This is the case for the series, or cascade, controller. This parameterisation is very common in industry. A block diagram representation of the controller is given in Figure 2.6, and the corresponding transfer function is

$$\frac{u}{e_i} = K_c \left(\frac{\tau_I s + 1}{\tau_I s} \right) (\tau_D s + 1). \tag{2.9}$$

Here, K_c, τ_I and τ_D are controller gain, integral time and derivative time, respectively.

Derivative filter

The aim of having derivative action in a controller, is to improve the dynamic response of the controlled variable and decrease the time necessary to reach

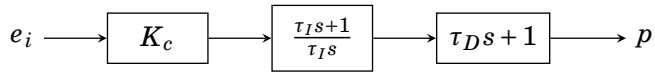


Figure 2.6 – Block diagram of a cascade form of PID control without derivative filter.

steady state. The derivative action is increasing the phase margin, and thus increasing robustness of the controller. This increase in robustness can be traded with a more aggressive tuning, which improves performance. The controller derivative action adds a zero to the system, which can increase the system sensitivity at high frequency. To avoid having the derivative of the measured variable alter excessively when experiencing the high frequent, random fluctuations of measurement noise, a filter can be added to the derivative action in the controller. The filter introduces an extra pole to the system, and "brings down" the high frequency sensitivity.

The cascade controller as presented in Equation (2.9) is physically unrealisable without a derivative filter. When using cascade PI control on a first-order process or cascade PID control on a second order process, the loop function will have more zeros than poles. A derivative filter is used to add one more pole, realizing the controller. The transfer function of the cascade form PID controller with derivative filter is given in Equation (2.10),

$$\frac{u}{e_i} = K_c \left(\frac{\tau_I s + 1}{\tau_I s} \right) \left(\frac{\tau_D s + 1}{\tau_F s + 1} \right), \tag{2.10}$$

where τ_F is the derivative filter time constant. τ_F is often written as a product of some factor and the derivative time constant, $\tau_F = \alpha \tau_D$, where α is called the derivative filter constant.

Solution Domain

As presented in the earlier sections, controllers are usually handled in the Laplace domain, and are often a ratio between two Laplace variable polynomials, that is, polynomials in s. When the numerator or the denominator of the controller transfer function approach zero, the value of the function approach either zero or infinity. The values for s where $K \to 0$ are denoted "zeros" while the values for s where $K \to \infty$ are named "poles". Therefore, by collecting the terms in Equations (2.7), (2.8) and (2.9), respectively, and factorizing, the controller solution domain in s for each parameterisation can be described. This is shown extensively in Appendix J. The cascade controller is limited to the real subdomain of the controller solutions for standard par-

allel and the alternative parallel PID controller parameterisation. For PI controllers, the solution domains are equal.

2.4 Smith Predictor

"Time-delay is control's worst enemy!"[a]. Time-delay adds phase lag, which affects the the closed-loop stability. Advanced techniques for the compensation of significant time-delay exists. The Smith predictor is a method where the time-delay theoretically is eliminated from the closed-loop transfer function. The Smith predictor utilises a model of the process to predict the system behaviour, and thus improve controller performance. The basis for the improved performance is that the controller doesn't have to "wait" to experience the system response, assuming a perfect prediction model. A block diagram of the Smith Predictor is given in Figure 2.7.

The derivation of the Smith predictor is based on the complementary sensitivity function T, with a preferred first order closed-loop response to setpoint changes. The Smith predictor is derived in Appendix H.

The model can for a FOPDT or SOPTD process be defined as

$$G \triangleq G_\circ e^{-\theta s}, \tag{2.11}$$

where $G_\circ$ contains the non-delayed dynamics of the model. For the Smith predictor, the model is defined as

$$\widetilde{G} \triangleq \widetilde{G}_\circ e^{-\theta_\circ s}, \tag{2.12}$$

where $\widetilde{G}$ is the nominal Smith predictor model with the underlying plant behavior given by $\widetilde{G}_\circ$, and $\theta_\circ$ is the nominal time-delay model. The Smith predictor controller, denoted $\widetilde{K}$, is a two-degree-of-freedom-controller with transfer function

$$\widetilde{K} = \frac{K}{1 + K\widetilde{G}_\circ\left(1 + e^{-\theta_\circ s}\right)}. \tag{2.13}$$

K is some controller configuration, also called the primary controller. By assuming a perfect plant model $\widetilde{G}_\circ = G_\circ$ and $\theta_\circ = \theta$ (i.e. no modelling error), no disturbance and no measurement noise, the error between the process output and the model output is zero, and the primary controller can be tuned as if there is no time-delay in the process (Normey-Rico and Camacho, 2007).

a) Quote Sigurd Skogestad, Process Control Course Notes, and many others.

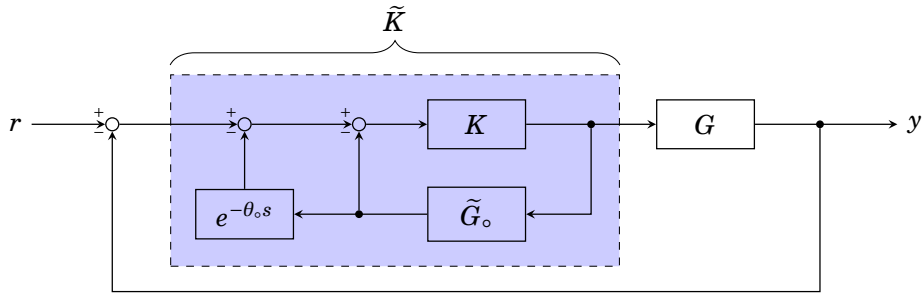


Figure 2.7 – Block diagram of the Smith Predictor

2.5 Simple Analytic Tuning Rules

Skogestad (2003) presented analytic rules for PID controller tuning based on the Internal Model Control approach (Rivera, Morari, and Skogestad, 1986). His aim was for the rules to be simple and yield good closed-loop behaviour. Simple analytic rules for model reduction to a first-order-plus-time-delay (FOPTD) or second-order-plus-time-delay (SOPTD) model was introduced, using the "half rule" for obtaining the effective time-delay. The tuning rules are in (Skogestad, 2001) claimed to be "probably the best tuning rules in the world", and are in short denoted SIMC for "Skogestad" or "simple" internal model control.

The SIMC method is a two-step procedure, as described in (Skogestad, 2003):

Step 1. Obtain a FOPTD or SOPTD model. The effective delay in the model may be obtained using the "half rule".

Step 2. Derive model-based controller settings. PI settings are derived from the FOPTD model and PID settings are derived from the SOPTD model.

A general experssion for of a FOPTD and a SOPTD model are given in Equations (2.14) and (2.15), respectively.

$$G(s) = \frac{k}{\tau s + 1} e^{-\theta s} \tag{2.14}$$

$$G(s) = \frac{k}{(\tau_1 s + 1)(\tau_2 s + 1)} e^{-\theta s} \tag{2.15}$$

Here, G is the process transfer function, k is the process gain, τ, τ_1 and τ_2 are process lag time constants, θ is the effective time-delay and s is the Laplace variable.

Step 1: Model Approximation To obtain models on the form of Equations (2.14) and (2.15), Skogestad and Grimholt (2012) states that the following model information needs to be estimated :

- The plant gain, k
- Dominant lag-time constant, τ_1
- Effective time-delay, θ
- Second-order lag time constant, τ_2

The parameters may be obtained in several ways. Examples are the Ziegler-Nichols open-loop step response and closed-loop setpoint response with P-controller methods (Ziegler and Nichols, 1942), an open loop step response experiment (Skogestad, 2003; Skogestad and Grimholt, 2012), the setpoint overshoot method (Shamsuzzoha and Skogestad, 2010) or approximation of the effective delay from a detailed higher order model using the "half rule" (Skogestad, 2003).

The "half rule" is based on a first order Taylor approximation of $e^{-\theta s} \approx 1-\theta s$, which is used to approximate negative numerator time constants and small time constants as time-delay. Since these time-delay approximations are conservative in terms of control, the rule is to distribute the largest neglected denominator time constant evenly to the effective delay and the smallest retained time constant. For a model in its original form

$$\frac{\prod_j \left(-\tau_{j_\circ}^{\text{inv}} s+1\right)}{\prod_i \left(\tau_{i_\circ} s+1\right)}, \tag{2.16}$$

the lag constants ($\tau_{i_\circ}$) are sorted in descending order, and $\tau_{j_\circ}^{\text{inv}} > 0$ denote the inverse response time constants. The model reduction of a higher order process is

$$\tau_1 = \tau_{1_\circ} + \frac{\tau_{2_\circ}}{2}; \quad \theta = \theta_\circ + \frac{\tau_{2_\circ}}{2} + \sum_{i\geq 3} \tau_{i_\circ} + \sum_j \tau_{j_\circ}^{\text{inv}}, \tag{2.17a}$$

$$\tau_1 = \tau_{1_\circ}; \quad \tau_2 = \tau_{2_\circ} + \frac{\tau_{3_\circ}}{2}; \quad \theta = \theta_\circ + \frac{\theta_3}{2} + \sum_{i\geq 4} \tau_{i_\circ} + \sum_j \tau_{j_\circ}^{\text{inv}}, \tag{2.17b}$$

for a first order and second order model reduction target, respectively.

Positive numerator time constants ($T_\circ s + 1$) are proposed cancelled by a "neighboring" denominator term ($\tau_\circ s + 1$), where the approximations

$$\frac{T_\circ s+1}{\tau_\circ s+1} \approx \begin{cases} T_\circ/\tau_\circ & \text{for} \quad T_\circ \geq \tau_\circ \geq \theta, \\ T_\circ/\theta & \text{for} \quad T_\circ \geq \theta \geq \tau_\circ, \\ 1 & \text{for} \quad \theta \geq T_\circ \geq \tau_\circ, \\ T_\circ/\tau_\circ & \text{for} \quad \tau_\circ \geq T_\circ \geq 5\theta, \\ \frac{\widetilde{\tau}_\circ/\tau_\circ}{(\widetilde{\tau}_\circ-\tau_\circ)s+1} & \text{for} \quad \widetilde{\tau}_\circ \triangleq \min(\tau_\circ, 5\theta) \geq T_\circ, \end{cases} \tag{2.18}$$

apply. Here, θ is the effective delay.

Step 2: Model Based Controller Settings The PID tuning derivation makes use of the Internal Model Control (IMC) approach for setpoints (Rivera et al., 1986) and the first order Taylor approximation of $e^{-\theta s}$ to yield a cascade parameterisation PID controller. The integral time was modified to give better disturbance rejection, and the resulting SIMC PID tuning rules for a SOPTD model is given in Equation (2.19). For FOPTD models, the derivative time term is set to zero, $\tau_D = 0$.

$$K_c = \frac{1}{k}\frac{\tau_1}{(\tau_c+\theta)}, \tag{2.19a}$$

$$\tau_I = \min[\tau_1, 4(\tau_c+\theta)], \tag{2.19b}$$

$$\tau_D = \tau_2. \tag{2.19c}$$

The closed-loop time constant, τ_c, is in Skogestad's original article recommended set equal to the time-delay to yield a robust controller with fast response, $\tau_c = \theta$.

CHAPTER 3

PARETO OPTIMISATION

"Premature optimization is the root of all evil"

— Donald E. Knuth
Computer scientist,
creator of TEX

3.1 Basic Principles

When performing optimisation of a nontrivial problem with multiple conflicting objectives, there doesn't exist a single solution that can optimise each of the conflicting objectives simultaneously. If none of the objective functions can be improved without a reduction of optimality in the other, the solution is said to be Pareto optimal. This trade-off is illustrated in Figure 3.1, where two conflicting objectives generate a Pareto optimal curve. The region above the curve contains the feasible set of solutions, while the region below the curve contains the set of infeasible solutions for the optimisation problem. As the optimisation problems are defined as minimising some objective, the optimality is increasing towards the origin for both objectives. Optimisation problems with multiple conflicting are called multi-objective or Pareto optimisation problems.

The tuning parameters for PID controllers can be adjusted as to yield high performance or high degree of robustness towards disturbances. However, combining these two objectives proves difficult as high performance is achieved at the expense of high robustness. One would prefer to find controller tunings that exercise good trade-off between the two competing objectives, thus the Pareto optimal controller solutions are essential. The goal of

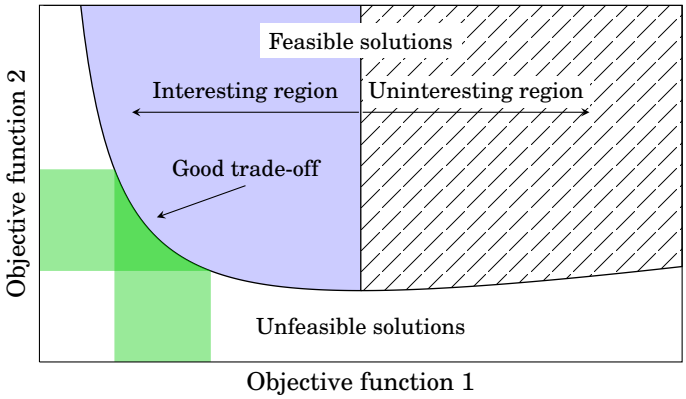


Figure 3.1 – An illustration of a Pareto-optimal curve with two conflicting objective functions. The "Uninteresting region" denotes a subset of solutions where a decrease in optimality in Objective function 1 does not result in an increase in optimality in Objective function 2.

this procedure is to quantify the trade-off between performance and robustness.

3.2 Performance

The output performance of the controller tuning is quantified as done by Grimholt and Skogestad (2012). The purpose of the objective function is to give a scalar, well balanced appearance of the trade-off between disturbance rejection and setpoint tracking. The controller tunings are evaluated by performing a positive step load in the input and output disturbance. The response behaviour may be oscillatory, and the magnitude of the integrated absolute error (IAE), defined in Equation (3.1), is set as a basis for the performance objective function. The IAE integrates the offset form reference for a given timespan. Zero steady-state error is assumed, that is, the controller is required to have integral action unless a pure integrating process is simulated.

$$\text{IAE} \triangleq \int_0^\infty |y(t) - r(t)| \mathrm{d}t. \tag{3.1}$$

Considering a weighted average of IAE for a step input load disturbance, d_i, and IAE for a step output load disturbance, d_o, the cost function is defined

as

$$J(K) \triangleq \frac{1}{2}\left[\frac{\mathrm{IAE}_{d_o}(K)}{\mathrm{IAE}^{\circ}_{d_o}} + \frac{\mathrm{IAE}_{d_i}(K)}{\mathrm{IAE}^{\circ}_{d_i}}\right], \tag{3.2}$$

where the reference weighting factors $\mathrm{IAE}^{\circ}_{d_o}$ and $\mathrm{IAE}^{\circ}_{d_i}$ are for given reference PID controllers. The reference weighting factors are found separately by performing a step load change in input and output for a given process, and finding the corresponding optimal controller. Thus, the reference values $\mathrm{IAE}^{\circ}_{d_o}$ and $\mathrm{IAE}^{\circ}_{d_i}$ are obtained from different optimal controllers, while the values IAE_{d_o} and IAE_{d_i} are given from the single controller yielding optimal trade-off between performance and robustness.

The reference PID controller is set to have a maximum sensitivity peak (M_S) value[a] of $M_S = 1.59$ originating from Grimholt and Skogestad (2012) reporting it to be the resulting M_S value for a SIMC PI controller with $\tau_c = \theta = \tau = 1$ and for the process $G = \frac{1}{\tau s+1}e^{-\theta s}$.

The reason for weighting the objective function is to provide relative values for the IAE resulting from the input and output step load change, respectively. Disturbance rejection performance is often rather poor compared to setpoint tracking performance, which may result in the output disturbance rejection completely dominating the cost function.

The weighting factors also causes the objective function to be independent of the size of the step load change and the process gain.

To be able to compare the Pareto optimal solutions for different controller structures and parameterisations, the IAE weights found for the Pareto optimal PID controller with the alternative parallel parameterisation was set as a fixed basis for all cost calculations.

As the weights in the cost function are found at $M_S = 1.59$, the optimal solution for $M_S = 1.59$ equals unity. By observing the deviation from $J = 1$ for $M_S = 1.59$, the optimality loss caused by the tuning compromise between performance and robustness is quantified. Further, the distance from the Pareto optimal curve to the tuning curve is denoted non-optimality loss. The loss definitions are illustrated in Figure 3.2.

3.3 Robustness

The robustness of a controller can be defined from its stability margins, that is, how close a stable closed-loop system is to instability (Skogestad and

a) Premature notation. M_S is defined as the maximum sensitivity peak; the topic is covered in Chapter 3.3.

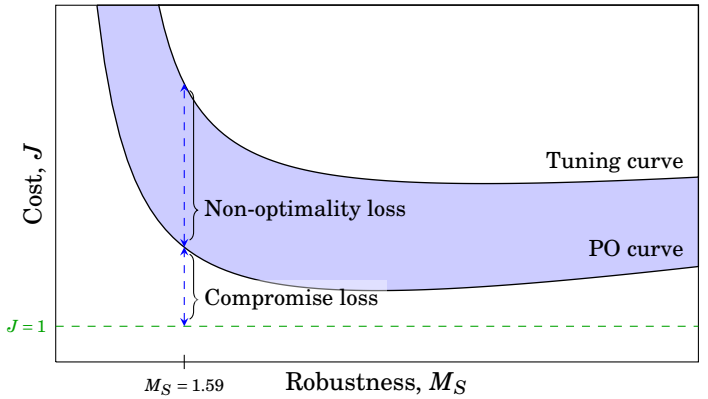


Figure 3.2 – An illustration of the performance-robustness compromise loss and the loss due to non-optimality of controller tuning.

Postlethwaite, 2005). To evaluate the controller robustness, gain margin (GM) and phase margin (PM) are introduced.

The gain margin is defined as

$$\mathrm{GM} \triangleq \frac{1}{|L(i\omega_{180})|}, \tag{3.3}$$

where $L(s) = L(i\omega)$ is the closed loop transfer function of a stable system with negative feedback, and ω_{180} is the phase crossover frequency where the Nyquist plot of $L(i\omega)$ crosses the negative real axis between $\mathrm{Re}(i\omega) = -1$ and $\mathrm{Re}(i\omega) = 0$.

The phase margin is defined as

$$\mathrm{PM} \triangleq \angle L(i\omega_c) + 180°, \tag{3.4}$$

where ω_c is the gain crossover frequency defined such that $|L(i\omega_c)| = 1$. The requirements for gain and phase margins are typically of magnitude $\mathrm{GM} > 2$ and $\mathrm{PM} > 30°$, respectively. A schematic illustration of a Nyquist plot of a possible L with corresponding gain and phase margins, and gain and phase crossover frequencies, are given in Figure 3.3.

The gain and phase margins are related to the stability transfer function S and complementary stability transfer function T through the maximum peak criteria. The maximum peaks of S and T are defined as

$$M_S \triangleq \max_\omega |S(i\omega)| = \|S\|_\infty, \tag{3.5a}$$

$$M_T \triangleq \max_\omega |T(i\omega)| = \|T\|_\infty, \tag{3.5b}$$

respectively, where $\|\cdot\|_\infty$ is the $\mathscr{H}_\infty$-norm. From the definition of S and T, the relation $S+T=1$ is true, and it follows that M_T is bounded by the selected M_S value by

$$M_T \leq M_S + 1. \tag{3.6}$$

The boundary relation in Equation (3.6) is not absolute. For stable systems, M_T is often less than M_S (Skogestad and Postlethwaite, 2005).

In the Nyquist plot, $1/M_S$ is the closest distance between $L(i\omega_c)$ and the critical point $\mathrm{Re}(L(i\omega)) = -1$. The relationship between M_S and M_T, and the gain and phase margins are

$$\mathrm{GM} \geq \frac{M_S}{M_S - 1} \qquad \text{and} \qquad \mathrm{GM} \geq 1 + \frac{1}{M_T}, \tag{3.7a}$$

$$\mathrm{PM} \geq \frac{1}{M_S} \qquad \text{and} \qquad \mathrm{PM} \geq \frac{1}{M_T}. \tag{3.7b}$$

Large values for M_S and M_T indicates poor robustness. M_S is used as the main criteria for the evaluation of robustness in terms of gain margin, with the complementary sensitivity being a measurement of the degree of oscillatory behaviour in the system. The recommended lower stability bound, and the upper maximum sensitivity peak value, is $M_S = 2$, which guarantees $\mathrm{GM} \geq 2$ and $\mathrm{PM} \geq 29°$ (Skogestad and Postlethwaite, 2005).

Closed loop instability occurs at the phase frequency $\omega_c = -180°$. As the phase margin quantifies how much phase lag that may be tolerated before this point is reached, the maximum additional time-delay before instability, $\theta_{\max}$, is reached is given by

$$\theta_{\max} = \frac{\mathrm{PM}}{\omega_c}. \tag{3.8}$$

While M_S is a quantification of the distance to the instability limit, the complementary sensitivity peak, M_T, represents the sensitivity to modelling errors of zeros and time-delay, and the worst-case noise amplification (Grimholt and Skogestad, 2013).

3.4 Optimisation Problem Formulation

Based on the previous chapters, the overall problem formulation for solving the Pareto optimal solutions is

$$\min_K \; J(K) = \frac{1}{2}\left[\frac{\mathrm{IAE}_{d_o}(K)}{\mathrm{IAE}^\circ_{d_o}} + \frac{\mathrm{IAE}_{d_i}(K)}{\mathrm{IAE}^\circ_{d_i}}\right], \tag{3.9a}$$

$$\text{s.t.} \quad M_S = m \qquad m \in \mathscr{M}_S, \tag{3.9b}$$

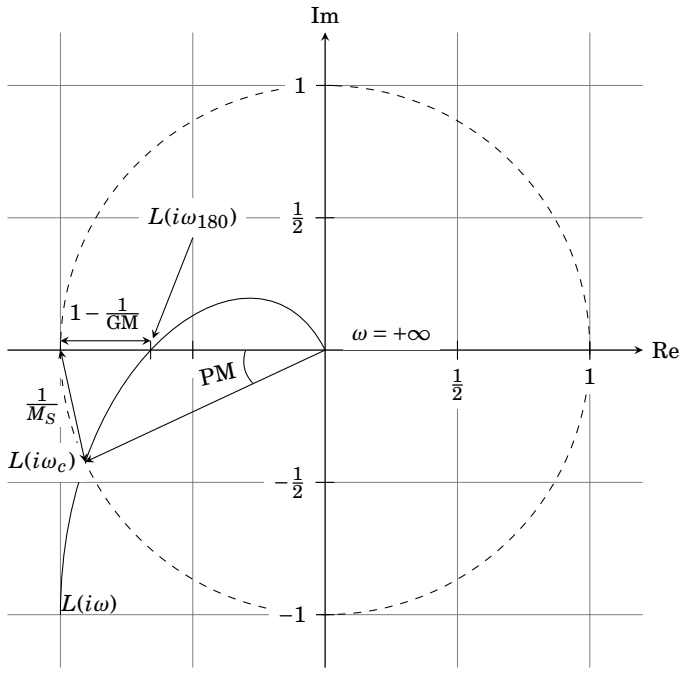


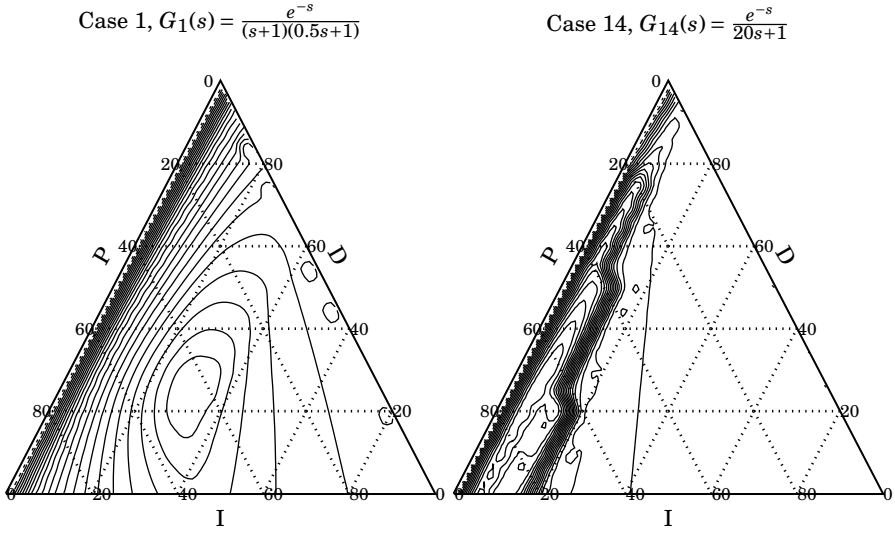
Figure 3.3 – A schematic Nyquist plot of a closed loop system with feedback transfer function ($L(s)$) with gain and phase margins (GM, PM) and the maximum sensitivity peak (M_S) of the system sensitivity transfer function (S). Re and Im denote the real and imaginary axes, respectively.

where K is a P, PI, PID or Smith predictor controller and $\mathscr{M}_S$ is a set of M_S values where m is the M_S value currently of interest. The process gain used for all models are positive ($k > 0$), thus providing the inequality constraint of the controller parameters being greater than zero,

$$K_c,\, \tau_I\, \tau_D \geq 0. \tag{3.10}$$

3.5 Optimal Plot Examples

The shape of the cost function described in Chapter 3.2 can be graphically presented by “brute force” evaluation. As good initial points for the optimisation routine are essential to obtain preferably global minima, knowledge about the curvature of the objective function proves helpful. Figure 3.4 illustrate the contour plot of the cost function surface for case 1 and case 14, respectively. The cost function for the SOPTD process is less non-convex com-



(a) Cost function J as a function of controller parameter values for $G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$

(b) Cost function J as a function of controller parameter values for $G_{14}(s) = \frac{e^{-s}}{20s+1}$

Figure 3.4 – "Brute force" evaluation of the objective function for a SOPTD and FOPTD model.

pared to the FOPTD process cost function. The latter will be more prone to erroneous optimisation, as the optimal point probably is located in a non-convex valley bottom in the area south west in the plot. The south eastern part of the plot is almost completely flat, which makes it very difficult for gradient based optimisation routines to find a descent direction.

3.6 Simulations

MATLAB was used to find the data necessary to perform the analysis. Appendix L extensively documents the approach and code used. It should be noted that the requirements for the programming structure changed throughout this work, and the documented structure is not optimal for continued research.

To simulate the step responses, two approaches is used.

1. Using MATLAB's native `step` function to return error and time values, which then is integrated by the MATLAB native `trapz` function.

2. A SIMULINK flow sheet, returning the integrated signal (Appendix N).

The simulation time for both approaches is set to a fixed value of $t_{\text{SimTime}} = 50\,\text{s}$ to ensure production of comparable results for all simulations. For cascade PID controllers, the derivative filter constant was set to $\tau_F = 0.001\tau_D$. The set of robustness values for the Pareto optimal PI and PID controllers is set to $\mathscr{M}_S = [1.25,\ldots,3.00]$ with a resolution of $M_{Si+1} - M_{Si} = 0.025$. For the Smith predictor the corresponding values are $\mathscr{M}_S = [1.25,\ldots,2]$, with a resolution of $M_{Si+1} - M_{Si} = 0.01$. When investigating the sensitivity to time-delay modelling error, the modelling error resolution was set to $\theta_{i+1} - \theta_i = 0.01\,\text{s}$.

3.7 Algorithm

The workflow for finding the Pareto optimal solutions for the problem formulated in Chapter 3.4 is outlined in Algorithm 1. The main MATLAB script used in the procedure is given in Appendix L, with appropriate support functions given in Appendix M. The optimisation routine consists of a primary and a secondary solver, called through the functions `fmincon` and `fminsearch` . The basic idea is that `fmincon` will perform a search for a solution controller K. If this solution is non-optimal based on the convergence criteria returned by the function, `fminsearch` will be invoked. If `fminsearch` returns a non-optimal solution, the complete run is terminated with an error message.

If `fmincon` or `fminsearch` consider the solution optimal, `fmincon` is finally invoked with "hot start" in the presumed optimal solution for confirmation.

`fmincon` is a native MATLAB-function which makes use of gradient based optimisation routines to solve a nonlinear constrained optimisation problem (Mathworks). The solver chosen is sequential quadratic programming (SQP). This optimisation method is more efficient than Nelder-Mead, used by `fminsearch`, with respect to computation time, but requires good initial points to converge to a global solution.

`fminsearch` is a native MATLAB-function which can be used for solving nonlinear unconstrained optimisation problems. It makes use of the `Nelder-Mead simplex direct search` method (Mathworks). As `fminsearch` is unconstrained, an `fzero` function is added within the cost calculations (`noGrad.m` in Appendix M.10) to find the appropriate controller gain yielding the desired M_S value. `fminsearch` performs a search for a minima in the plane spanned by the integral and derivative time.

Details on the mathematical aspect of the SQP and Nelder-Mead optimisation routines is extensively documented in the literature on numerical optimisation. An example is "Numerical Optimization" by Nocedal and Wright (2006).

Algorithm 1 Find Pareto optimal tuning curve.

for `model` in set of models **do**
  if Controller configuration equal `po:pid:alternative` **then**
    for $M_S = 1.59$ **do**
      Find K that minimizes J for a step input load disturbance (d_i) $\rightarrow \text{IAE}_{d_i}$
      Find K that minimizes J for a step output load disturbance (d_o) $\rightarrow \text{IAE}_{d_o}$
    end for
  else
    Load IAE reference weighting factors for `model`
  end if
  for all M_S in $\mathcal{M}_S$ **do**
    Find K that (globally) minimizes J for a combined step in d_i and d_o.
  end for
  `Plot` $J = f(M_S)$
end for

CHAPTER 4

FIRST ORDER PROCESSES

4.1 Introduction

In this chapter, the results of the study of Pareto optimal PI controllers found by Grimholt and Skogestad (2012) for a set of FOPTD process models are reproduced. Pareto optimal PI controller solutions are found and compared with the SIMC tuning rules. Reference points for different choices for the closed-loop time constant (τ_c) are demonstrated, and the behaviour in these points are discussed. The results found by Grimholt and Skogestad (2013) in their study of Pareto optimal PID controllers on the same processes are also confirmed. The question of optimality and controller parameterisation is addressed and discussed.

One should always consider the occurrence of uncertainty. Numerical uncertainty is always present, and the optimisation routines may find solutions that are close to optimality, but not truly the global optimum. The integration step size for the SIMULINK block diagram was set to a constant value of 10^{-3}, which yield a 1 % loss in the cost function compared to the `trapz` integration of the values returned by the `step` function. Thus, when discussing the performance, all figures are rounded downwards to an integer value. The lower bound integer values have been chosen to provide conservative results.

The complete set of Pareto optimal tuning curves, PI and PID tuning examples, and reference SIMC tuning parameters can be found in Appendix A.

Table 4.1 – Process descriptions for the first-order-plus-delay processes investigated.

Case	Process description	Transfer function
10	Integrating process	$G_{10}(s) = \frac{e^{-s}}{s}$
11	Time-delay dominated process	$G_{11}(s) = \frac{e^{-s}}{(s+1)}$
12	Lag dominated process	$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$
13	Pure time-delay process	$G_{13}(s) = e^{-s}$
14	High lag-time process	$G_{14}(s) = \frac{e^{-s}}{20s+1}$

4.1.1 Cases

Five FOPTD models have been investigated. The cases are presented in Table 4.1. The cases are divided between a pure integrating process, a time-delay dominated and a lag-time dominated process, a pure time-delay process, and a process with high lag-time. The lag dominated process with time constant $\tau = 8$ is interesting as this is the "breaking point" for the SIMC tuning rule for the integral time, while the last process with $\tau = 20$ holds a highly integrating behaviour. The reason for the somewhat unintuitive numbering and internal ordering of the cases is that the FOPTD cases were the last to be investigated. In addition, the number of cases was originally four; the fifth case was added to confirm a trend in the Pareto optimal Smith predictor tuning observed. As a consequence, the SOPTD cases are numbered 1–9 while the FOPTD cases are numbered 10–14.

For a pure time-delay process, the SIMC tuning yields $K_c = 0$ and $\tau_I = 0$. To provide non-zero SIMC tuning, the pure time-delay process has been approximated to yield a FOPTD model with a minor lag-time constant,

$$G_{13} \approx \frac{1}{0.005s+1} e^{-s} = G_{13}^{\star}, \tag{4.1}$$

denoted $G_{13}^{\star}$.

4.2 Pareto Optimal PI and PID Controllers

The Pareto optimal PI and PID controller solutions were found according to Algorithm 1, described in Chapter 3.7. Table 4.2 summarise the controller tuning at $M_S = 1.59$, with the corresponding IAE trade-off values and the cost function values. The complete set of Pareto optimal plots with complementary sensitivity are listed in Appendix A. The Pareto optimal PI tuning

curves are listed in Figure A.1(a) through A.5(a). The corresponding plots for the Pareto optimal PID controllers are given in Figure A.1(b) through A.5(b).

4.2.1 Optimality of Controller Parameterisation

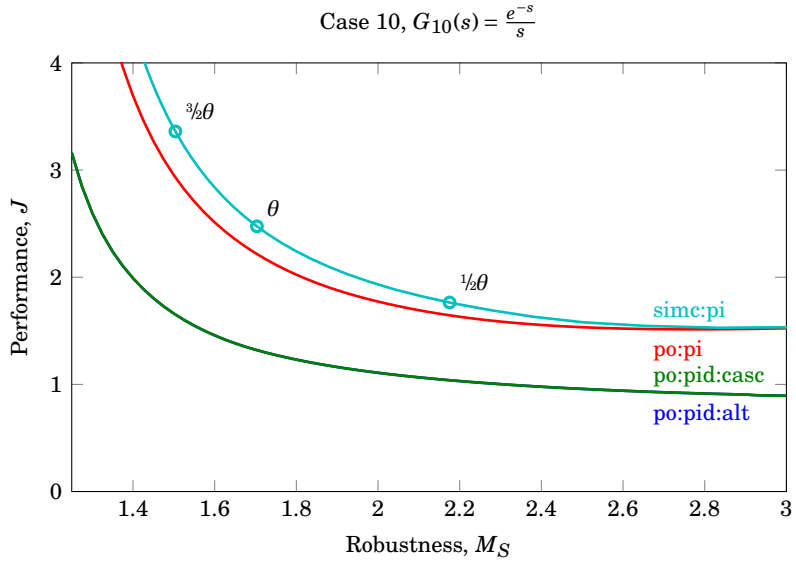
The Pareto optimal PI and PID tuning curves are given in Figure 4.1(a) for case 10, and in Figures 4.3 through 4.6, for case 11–14. The alternative parallel and cascade controller parameterisations are included. Only in case 11 the alternative and cascade controller parameterisation have unequal Pareto optimal PID tuning — the curves don't overlap completely in the M_S region investigated. The non-optimality of the cascade controller is confirmed by the alternative controller having complex zeros in the given M_S domain. For all the other cases, the controller zeros are real, indicating that a cascade controller can achieve Pareto optimal trade-off. This is also evident from the IAE values. Equal values yield identical controller.

In case 11, the deviation observed is at its maximum at very high or very low robustness. At low robustness the deviation becomes significant outside the Pareto optimal region. At high robustness the performance can be improved by 12 % by switching from optimal cascade to optimal parallel controller parameterisation. In the trade-off region where $2 \geq M_S \geq 1.4$, the performance deviation is practically nil.

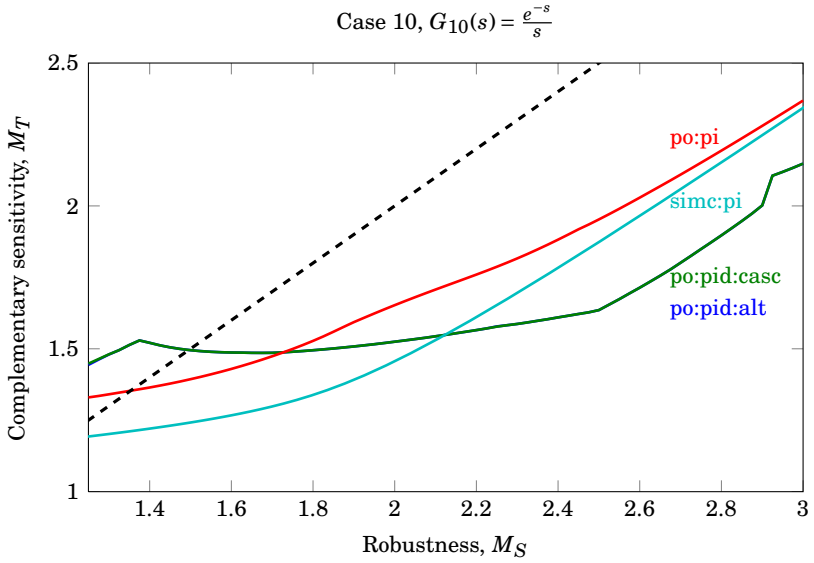
4.2.2 Sensitivity of the Pareto Optimal Controllers

Systems experience disturbances not only in the form of input and output disturbances, but also in the process itself. The controller optimality will deteriorate if the system dynamics are shifted. Consequently, the optimality of the tuning should be exposed to system disturbances to investigate their behaviour. This work focus on time-delay modelling error, and the Pareto optimal solutions have been exposed to time-delay error to provide a basis of comparison for the stability analysis of the Smith predictor structure in Chapter 6.4. The time-delay error tested is ±90 %, such that the time delay of the process is $0.1\theta_\circ \leq \theta \leq 0.9\theta_\circ$.

The Pareto optimal controller tuning sensitivity to variations in time-delay has been evaluated in terms of change in performance and robustness, respectively. The results for the FOPTD models are given in Figures C.10 through C.14 in Appendix C.1. The plots have been produced by extracting a Pareto optimal controller for a given robustness target, and then evaluate the performance of the controller when the time-delay parameter for the process model is changed.

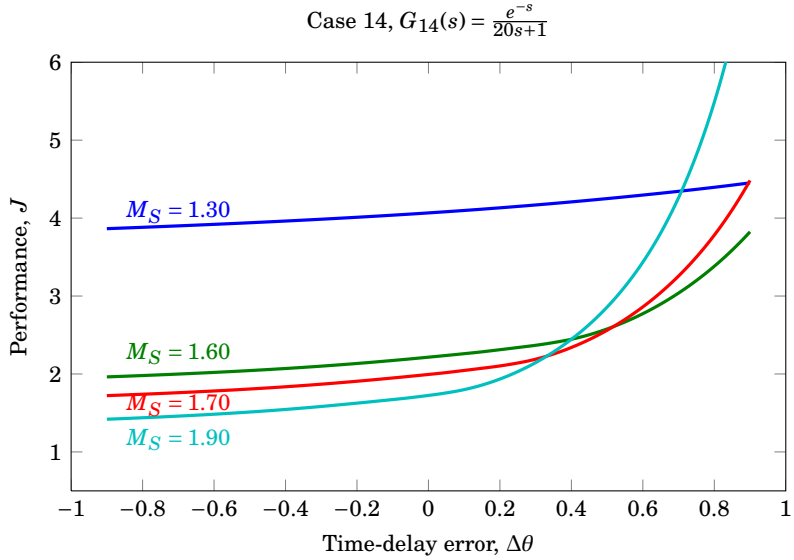


(a) Cost value as a function of robustness, $J(M_S)$.

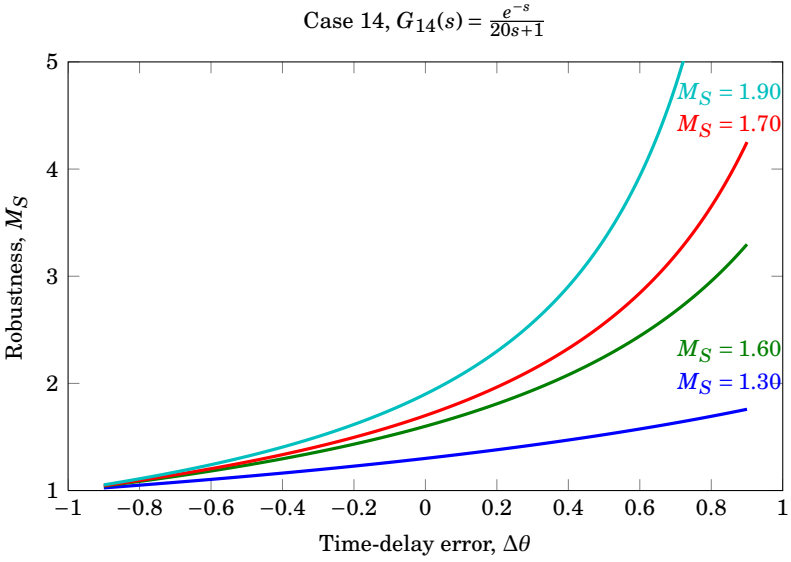


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure 4.1 – SIMC tuning controllers compared to Pareto optimal PI and PID controllers for Case 10, integrating process: $G_{10}(s) = \frac{e^{-s}}{s}$.



(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.



(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure 4.2 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_{14}(s) = \frac{e^{-s}}{20s+1}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

Table 4.2 – Optimal PI and PID controllers for $M_S = 1.59$.

	Process	$\min_K J(K)$					
		Cascade parameterisation					
		K_c	τ_I	τ_D	J	IAE_{d_o}	IAE_{d_i}
PI	$G_{10}(s) = \frac{e^{-s}}{s}$	0.41	6.13	-	2.55	4.35	15.24
	$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	0.54	1.10	-	1.39	2.08	2.04
	$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	3.44	3.96	-	1.96	3.12	1.15
	$G_{13}(s) = e^{-s}$	0.20	0.32	-	1.03	1.62	1.62
	$G_{14}(s) = \frac{e^{-s}}{20s+1}$	8.34	5.04	-	2.24	3.70	0.61
PID	$G_{10}(s) = \frac{e^{-s}}{s}$	0.54	3.27	0.48	1.47	3.02	6.83
	$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	0.42	0.61	0.61	1.02	1.56	1.47
	$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	4.35	2.54	0.48	1.27	2.35	0.63
	$G_{13}(s) = e^{-s}$	0.19	0.31	0.02	1.00	1.59	1.59
	$G_{14}(s) = \frac{e^{-s}}{20s+1}$	10.87	2.95	0.48	1.38	2.70	0.30
		Alternative parallel parameterisation					
		K	I	D	J	IAE_{d_o}	IAE_{d_i}
PID	$G_{10}(s) = \frac{e^{-s}}{s}$	1.05	0.16	0.25	1.47	3.02	6.83
	$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	1.91	0.40	0.15	1.01	1.58	1.43
	$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	8.97	0.19	0.23	1.27	2.35	0.63
	$G_{13}(s) = e^{-s}$	0.84	0.75	0.00	1.00	1.59	1.59
	$G_{14}(s) = \frac{e^{-s}}{20s+1}$	21.48	0.17	0.24	1.38	2.70	0.30

The sensitivity to time-delay error is as expected for the Pareto optimal PI controllers. Case 14 is used as an illustrative example, with performance variations in Figure 4.2(a) and robustness efficiency in 4.2(b). High M_S value yields better performance compared to low M_S values. The controller performances are decreasing at approximately equivalent rates when the true time-delay (θ) is minor the nominal ($\theta_\circ$). When $\theta > \theta_\circ$, the controllers at high nominal M_S value deteriorates before the controllers at low nominal M_S value, and the performance reduction rate is increasing more rapidly for the high M_S controllers. The same holds for the robustness efficiency for all cases.

4.2.3 PI vs. PID Control

The quantitative evaluation of potential performance improvements when switching from PI to PID control is performed at the robustness reference point $M_S = 1.59$. The data is collected from Table 4.2. For the PID controller, the cascade parameterisation is used as basis.

It can be observed that the PID controller outperform the PI controller by 42 % for the integrating process in case 10. For the time-delay dominant process in case 11, the improvement is 27 %. In the lag-dominated process in case 12, the performance improvement is 35 %, while for the highly lag dominated process in case 14 the improvement is 38 %.

For the pure time-delay process in case 13, the performance can be improved by 3 % according to the parameter values in Table 4.2. The IAE values for G_{13} are equal, which implies that there are no trade-off between input and output disturbance rejection. According to the IMC rules, optimal control is then achieved by zero-pole cancellation in G_{13} (Rivera et al., 1986), which for G_{13} implies that $\tau_I = \tau_1 = 0$. Consider the controller parameterisation

$$K = \frac{K_I}{s}(\tau_I s + 1)(\tau_D s + 1), \tag{4.2}$$

where the integral gain term K_I is defined by

$$K_I \triangleq \frac{K_c}{\tau_I} = \frac{\frac{1}{k}\frac{\tau_1}{\tau_c+\theta}}{\tau_I} \xrightarrow{\tau_I=\tau_1} \frac{1}{k}\frac{1}{\tau_c+\theta} \neq 0 \text{ for } \tau_I = 0. \tag{4.3}$$

If $\tau_I = 0$, the controller expression is reduced to

$$K = \underbrace{\frac{K_I}{s}(\tau_D s + 1)}_{ID} = \underbrace{K_I\tau_D + \frac{K_I}{s}}_{PI}, \tag{4.4}$$

which indicates that the parameterisation of a PI controller is equal to that of an ID controller.

Usually the integral term cancels a pole in the model, whereas it in G_{13} is cancelling the dead-time expression ($e^{-\theta s} \approx \frac{1}{1+\theta s}$). Thus a two-term controller is optimal for the pure time-delay. It should then be expected that the PID controller wouldn't obtain a better performance than the PI controller. However, as $G_{13}^{\star}$ has dynamics in form of the added pole, the derivative action will improve the system response. Figure 4.5 also indicates that the offset between the Pareto optimal PI and PID curve is constant throughout the trade-off region.

In the non-robust region ($M_S \geq 2$), both the PID cascade controllers and PID parallel controllers yield almost constant performance. This differs from

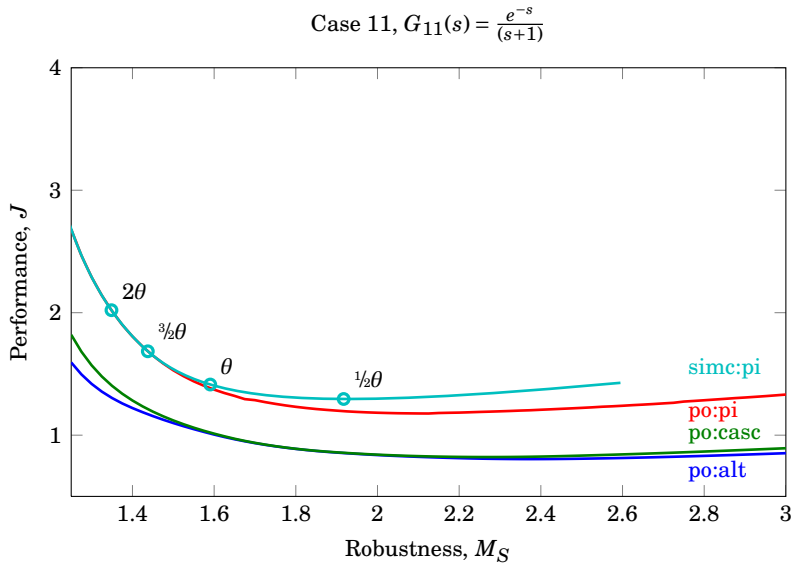


Figure 4.3 – Pareto optimal tuning curve for PI and PID controllers for $G_{11}(s) = \frac{e^{-s}}{(s+1)}$.

the results obtained by Grimholt and Skogestad (2013), where the Pareto optimal PID tuning curve experiences a decrease in performance in the non-robust region. Evaluation of the sensitivity function show that the M_S target value is indeed met; $M_S = 1.925$ at about $\omega = 1.3 \cdot 10^3 \,\mathrm{rad\,s^{-1}}$. For $M_S = 1.9$ the corresponding frequency is $\omega = 2.2 \,\mathrm{rad\,s^{-1}}$. The sensitivity function is experiencing severe oscillation in the high frequency domain caused by the time-delay of the process, which yield heavy gain oscillations at high frequency. Preferably one would have MATLAB pick one of the first peaks of the sensitivity function to compute the M_S value. A possible explanation of the difference with Grimholt's work can be if he defined his M_S domain on a smaller range of frequencies. A search for the maximum peak value limited to, for instance, $\omega \in [10^{-2}, 10^2]$ will obviously record the time-delay oscillatory response of the sensitivity function at $\omega > 10^2$.

The complementary sensitivity, M_T, is equal for the cascade and parallel parameterisation in cases 10, 12 and 13, while the parallel PID controller yield a slightly smaller M_T value in case 11. This indicates that noise amplification at high frequencies may pose a problem in this robustness region.

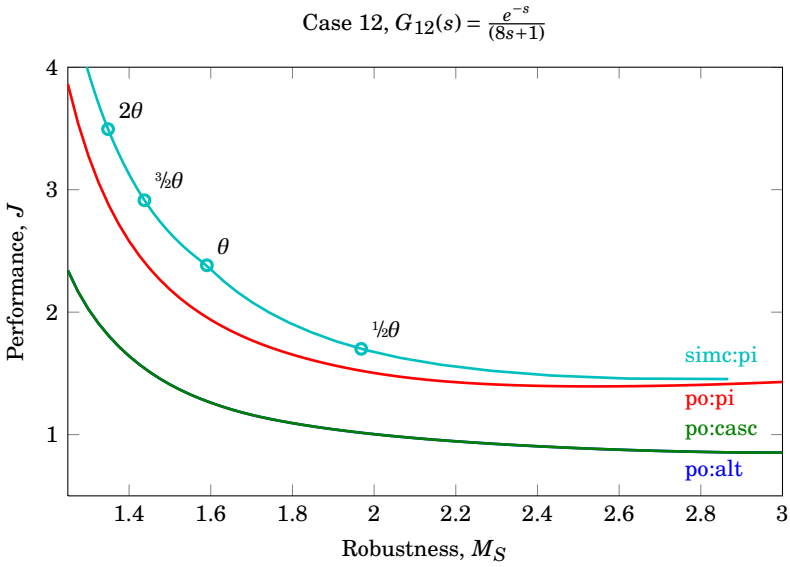


Figure 4.4 – Pareto optimal tuning curve for PI and PID controllers for $G_{12}(s) = \frac{e^{-s}}{(8s+1)}$.

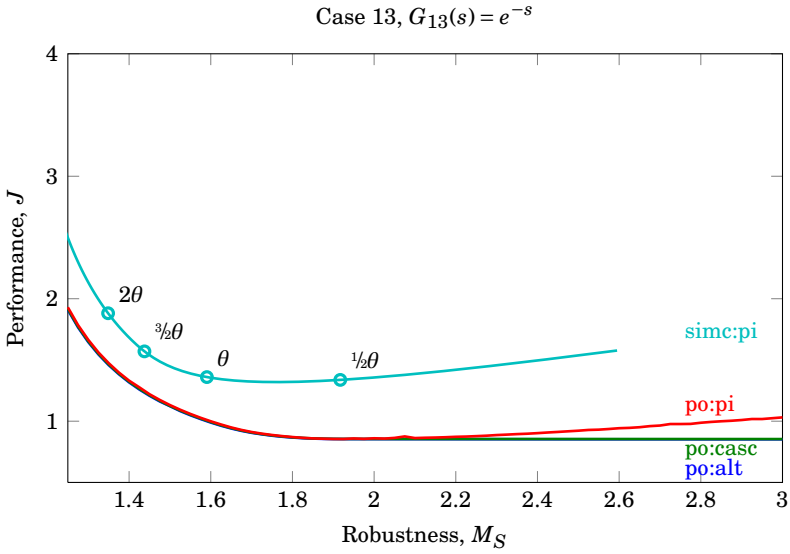


Figure 4.5 – Pareto optimal tuning curve for PI and PID controllers for $G_{13}(s) = e^{-s}$.

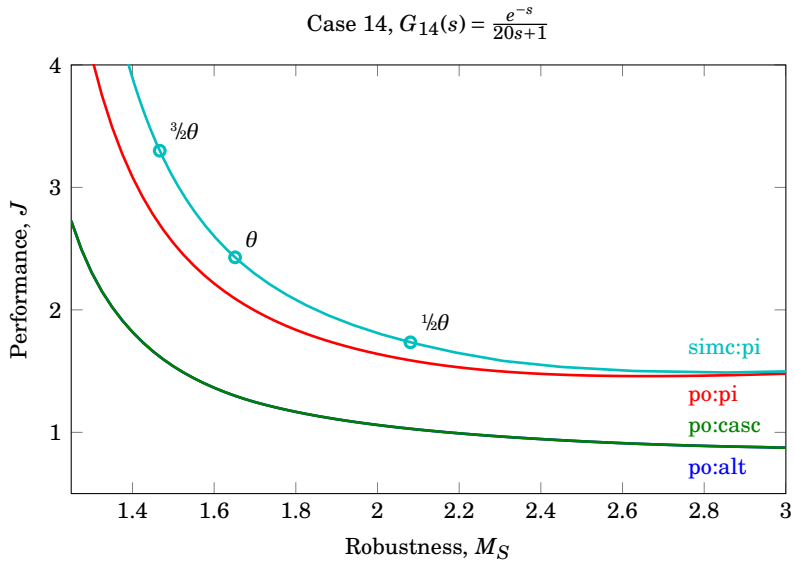


Figure 4.6 – Pareto optimal tuning curve for PI and PID controllers for $G_{14}(s) = \frac{e^{-s}}{20s+1}$.

4.3 SIMC Control

The SIMC tuning rules are given in Equation (2.19) in Chapter 2.5. The closed loop time constant, τ_c, is the only degree of freedom. The SIMC tuning curves in Figures 4.1, and Figure 4.3 through 4.5 was generated by evaluating the system robustness and performance for a wide range of τ_c values. The points $\tau_c = 2\theta$ (excessively robust tuning), $\tau_c = \frac{3}{4}\theta$ (smooth tuning), $\tau_c = \theta$ (tight tuning), and $\tau_c = \frac{1}{2}\theta$ (aggressive tuning) have been marked for reference. The recommended setting is $\tau_c = \theta$ for good trade-off between robustness and performance. The algorithm followed to retrieve the SIMC controller tuning is outlined in Algorithm 2.

For an integrating process, τ can be approximated to be close to infinity. Thus, when applying the SIMC tuning rules for an integrating process, the integral time will always be given by the second term in Equation (2.19b). For case 10 ($G = \frac{e^{-s}}{s}$), to achieve nonzero SIMC tuning parameters, the pure time-delay process was estimated as described in Chapter 5.1.1.

Algorithm 2 Determine SIMC tuning curve.

```
for model in set of models do
    Load IAE reference weighting factors for model.
    if PI control and SOPTD model then
        Apply half rule
    end if
    for all τc in span of τc do
        Generate SIMC controller
        Calculate J for a combined step in di and do.
        Calculate MS value
    end for
    plot J = f(MS)
end for
```

4.3.1 Optimality of SIMC Tuning

For case 10, 11 and 12, the SIMC tuning rules achieve tuning close to optimal. As reported by Grimholt and Skogestad (2012), the SIMC tuning rules are observed to be non-optimal for pure time-delay processes. The achieved cost at $M_S = 1.59$ is about 36 % less optimal compared to the Pareto optimal PI or PID performance. Grimholt and Skogestad (2012) report that the optimal PI controller for a pure time-delay process has an integral time at approximately $\tau_I = \theta/3$. This value concur with the results obtained in this study, where the optimal integral time is $\tau_I = 0.32$. They propose a simple change to the SIMC tuning rule to achieve better performance for pure time-delay processes. This suggestion has not been evaluated further.

4.3.2 The Closed Loop Constant

The reference points for values of $\tau_c/\theta \in [½\ 1\ ¾\ 2]$ for the FOPTD models are listed in Table 4.3. The M_S value for a given τ_c does not vary excessively, but is almost constant regardless of process. The internal model control principle of designing the controller to cancel out dynamics in the system stabilises the M_S value. The exception is the integrating process in case 10, where the process gain and lag-time constant can be assumed to be infinitely large, and the SIMC tuning rules yield an integral time given by $4(\tau_c+\theta)$. Consequently, the τ_c values are shifted towards higher M_S values.

Choosing tight tuning ($\tau_c = \theta$) for the SIMC rules yields robustness values in the range $1.7 \geq M_S \geq 1.59$, where the high robustness levels are for the processes where $\tau_1 < 4(\tau_c + \theta)$. Increasing the closed-loop constant to smooth

Table 4.3 – SIMC PI tuning reference points for $\tau_c/\theta \in [½\ 1\ ⅜\ 2]$.

Process	τ_c/θ	τ_c	K_c	τ_I	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_{10}(s) = \frac{e^{-s}}{s}$	½	0.50	0.67	6.00	1.76	2.18	3.40	9.00
	1	1.00	0.50	8.00	2.47	1.70	3.92	15.99
	⅜	1.50	0.40	10.00	3.36	1.50	4.51	24.97
	2	2.00	0.33	12.00	4.41	1.39	5.14	35.87
$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	½	0.50	0.67	1.00	1.30	1.92	2.13	1.73
	1	1.00	0.50	1.00	1.41	1.59	2.17	2.03
	⅜	1.50	0.40	1.00	1.68	1.44	2.50	2.50
	2	2.00	0.33	1.00	2.02	1.35	3.00	3.00
$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	½	0.50	5.33	6.00	1.70	1.97	2.37	1.12
	1	1.00	4.00	8.00	2.38	1.59	2.17	2.00
	⅜	1.50	3.20	8.00	2.91	1.44	2.50	2.49
	2	2.00	2.67	8.00	3.49	1.35	3.00	2.99
$G_{13}(s) = e^{-s}$	½	0.50	0.00	0.01	1.34	1.92	2.13	2.13
	1	1.00	0.00	0.01	1.36	1.59	2.17	2.17
	⅜	1.50	0.00	0.01	1.57	1.44	2.50	2.50
	2	2.00	0.00	0.01	1.88	1.35	3.00	3.00
$G_{14}(s) = \frac{e^{-s}}{20s+1}$	½	0.50	13.33	6.00	1.74	2.08	2.98	0.45
	1	1.00	10.00	8.00	2.43	1.65	3.20	0.80
	⅜	1.50	8.00	10.00	3.30	1.47	3.43	1.25
	2	2.00	6.67	12.00	4.33	1.36	3.68	1.78

tuning ($\tau_c = ⅜\theta$) yields $1.5 \geq M_S \geq 1.44$. The recommended settings for tight tuning give robustness well within the recommended limit of $M_S \leq 2$, and there should be no reason for reducing the performance by applying the closed-loop constant value for smooth tuning.

4.4 Summary: First Order Processes

The difference between the alternative parallel controller parameterisation and the cascade parameterisation is zero for all but one process. As the potential performance improvement for this process is very small, it is concluded that there are no reasons for switching from cascade to parallel controller parameterisation for the PID controller.

Considering the difference observed between the Pareto optimal PID controller curve in Figure 4.5 and the result Grimholt and Skogestad (2013) obtained for PID control of a pure time-delay process, where the PI and PID tuning curves are superimposed in the entire M_S region. As explained in Chapter 3.6, the frequency region chosen for this study was set to $\omega \in$

$[10^{-4}, 10^{4}]$. A plausible explanation of the deviation is that Grimholt had a smaller frequency region defining the M_S value. For further studies, the significant frequency region for each case should be examined prior to performing optimisation to ensure sensible M_S values being obtained.

For the FOPTD with dynamics, that is, all cases studied except the pure time-delay process in case 13, there are substantial potential for performance improvement by switching from PI to PID control. Several cases displays improvements in performance of more than 30 %.

The SIMC tuning rules are displaying very good results for the FOPTD processes, except for the pure dead-time process. The improvement suggested by Grimholt and Skogestad (2012) was not further examined. Tight tuning of the SIMC tuning rules does indeed give good trade-off between robustness and performance, as claimed by Skogestad (2003).

CHAPTER 5

SECOND ORDER PROCESSES

5.1 Introduction

Foss (2012) found in his follow up project of Grimholt and Skogestad (2012) Pareto optimal PID controllers for a set of SOPTD processes. In this chapter, the results found by Foss (2012) are reproduced. The Pareto optimal PI and PID controller tunings are found for nine SOPTD models. Pareto optimal parallel and cascade controller parameterisations are compared. The sensitivity of the Pareto optimal tuning solutions towards variations in the process time-delay is investigated and discussed. The SIMC PI and PID tuning are retrieved for a wide range of τ_c values, with the corresponding reference tuning values for $\tau_c/\theta \in [½\ 1\ 1½\ 2]$ being listed. Derivative filter was not applied to the SIMC PID controllers.

The potential loss of optimality when using a cascade controller parameterisation over a parallel controller is discussed in Section 5.2.1. The optimality of the SIMC tuning rules are compared to the Pareto optimal solutions. The choice of τ_c for time-delay dominated and lag-time dominated processes are treated in Section 5.3.

The complete set of Pareto optimal tuning curves, PI and PID tuning examples, and reference SIMC tuning parameters can be found in Appendix B.

Table 5.1 – Model descriptions for the second-order-plus-delay processes investigated.

Case	k	τ_1	τ_2	θ	Transfer function
1	1	1	$0.5\tau_1$	1	$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$
2	1	1	$0.8\tau_1$	1	$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$
3	1	1	$0.3\tau_1$	1	$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$
4	1	1	$0.5\tau_1$	$\tau_2/1.5$	$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$
5	1	1	$0.8\tau_1$	$\tau_2/1.5$	$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$
6	1	1	$0.3\tau_1$	$\tau_2/1.5$	$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$
7	1	1	$0.5\tau_1$	$\tau_2/2$	$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$
8	1	1	$0.8\tau_1$	$\tau_2/2$	$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$
9	1	1	$0.3\tau_1$	$\tau_2/2$	$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$

5.1.1 Cases

Nine SOPTD models have been investigated. The three first cases are time-delay dominated processes with $\theta > \tau_2$, while the six latter are lag dominated with $\tau_2 > \theta$. The model data and process transfer functions are given in Table 5.1.

5.2 Pareto Optimal PI and PID Control

The Pareto optimal PI and PID controllers were found by applying the procedure presented in Chapter 3.7. A review of PI controllers at a given robustness, $M_S = 1.59$, with corresponding IAE trade-off values and cost function values is given in Table 5.2. A similar review of the PID controllers is given in Table 5.3, including the alternative parallel parameterisation and the cascade parameterisation. The obtained cost values differs slightly from the values reported by Foss (2012), with a general deviation of less than 5 %. Foss (2012) never clarifies whether he apply a derivative filter to his controllers. It is assumed that no derivative filter was used by (Foss, 2012), which can explain the deviation.

The Pareto optimal PI and PID tuning curve for case 3 is given in Figure 5.1(a). The alternative parallel parameterisation and the cascade parameterisation is included for the PID controller. The SIMC PI and PID is also included, with reference points marked as circles. Figure 5.1(b) show

Table 5.2 – Optimal PI cascade controllers for $M_S = 1.59$

	$\min_K J(K)$				
Process	K_c	τ_I	J	IAE_{d_o}	IAE_{d_i}
G_1	0.48	1.34	1.50	2.82	2.77
G_2	0.52	1.56	1.60	3.09	3.02
G_3	0.48	1.21	1.44	2.58	2.54
G_4	0.98	1.29	2.29	1.48	1.31
G_5	0.78	1.55	2.04	2.15	1.99
G_6	1.38	1.07	2.63	0.98	0.77
G_7	1.17	1.28	2.76	1.31	1.09
G_8	0.93	1.56	2.39	1.87	1.67
G_9	1.62	1.04	3.25	0.88	0.64

how the M_T value varies as a function of M_S for case 3.

5.2.1 Optimality of Controller Parameterisation

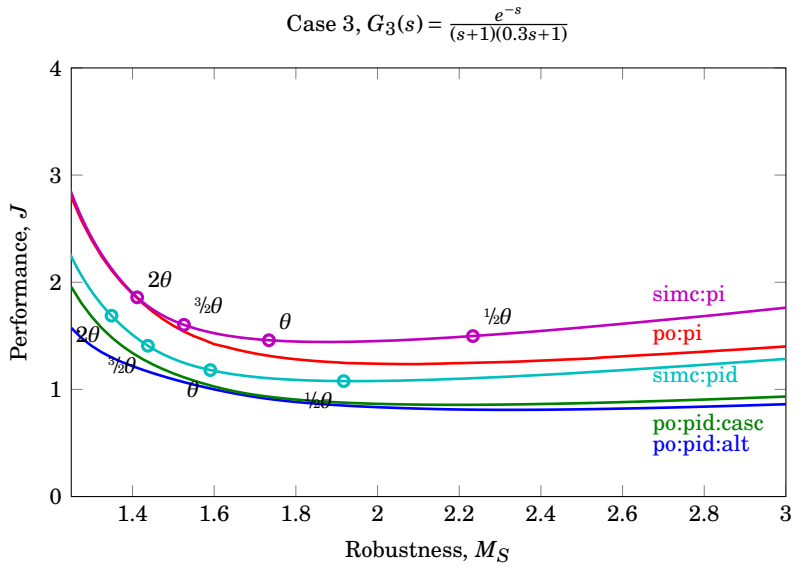
Considering a robustness level of $M_S = 1.59$, the deviation from optimality when using a cascade controller parameterisation instead of a parallel controller parameterisation is virtually nil. Table 5.3 indicates that the performance improvement varies from 4 % to zero at this robustness. In the robust solution domain, $M_S \leq 1.59$, the cascade controller is notably less optimal. In case 1, a performance improvement of approximately 25 % can be achieved by replacing an optimal cascade controller with an optimal parallel controller at $M_S = 1.25$. This holds for the time-delay dominated cases, where the improvement at high robustness ranges from 20 % to 25 %. At low robustness ($M_S \geq 2$), the difference is notable. Importantly, the difference between the parameterisations are close to minimal in the region where $1.4 \leq M_S \leq 1.9$.

For the lag-time dominated processes, the optimality loss is negligible for values of M_S less than 2.

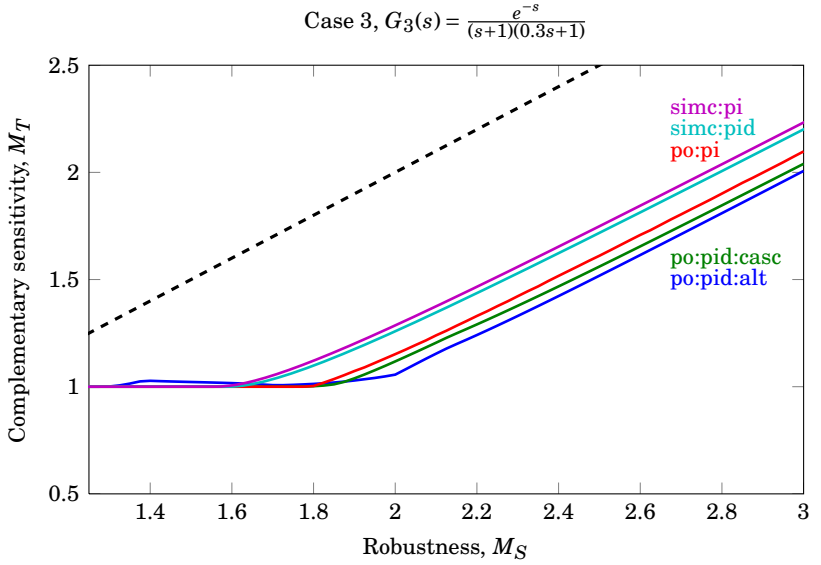
The complementary sensitivity peak value plots are smooth for all cases, with $M_T < M_S$. There is in general no loss in M_T due to parameterisation, though the alternative parallel parameterisation yields a slightly smaller M_T value in the non-robust region.

5.2.2 Sensitivity of the Pareto Optimal Controller

As in Chapter 4.2.2, the sensitivity of the Pareto optimal controllers have been evaluated with respect to variations in the process time-delay. The pro-



(a) Cost value as a function of robustness, $J = f(M_S)$.



(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

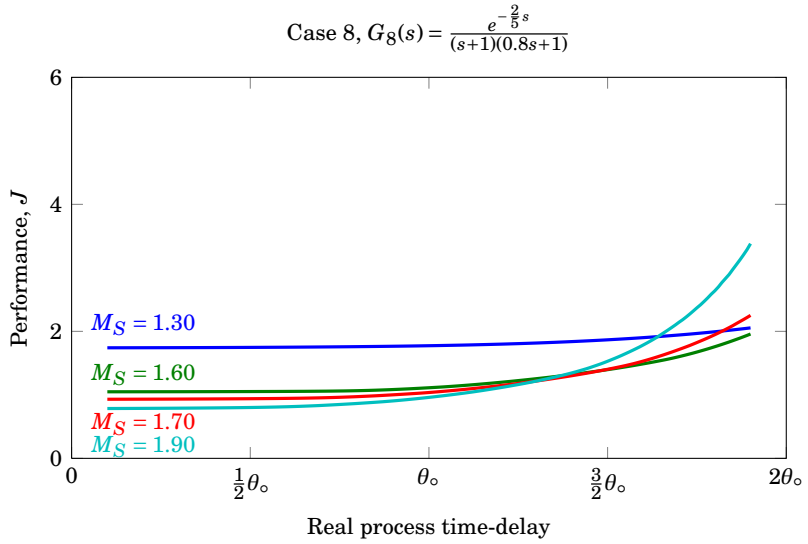
Figure 5.1 – SIMC tuning controllers compared to Pareto optimal PI and PID controllers for $G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$.

Table 5.3 – Optimal PID controllers for $M_S = 1.59$

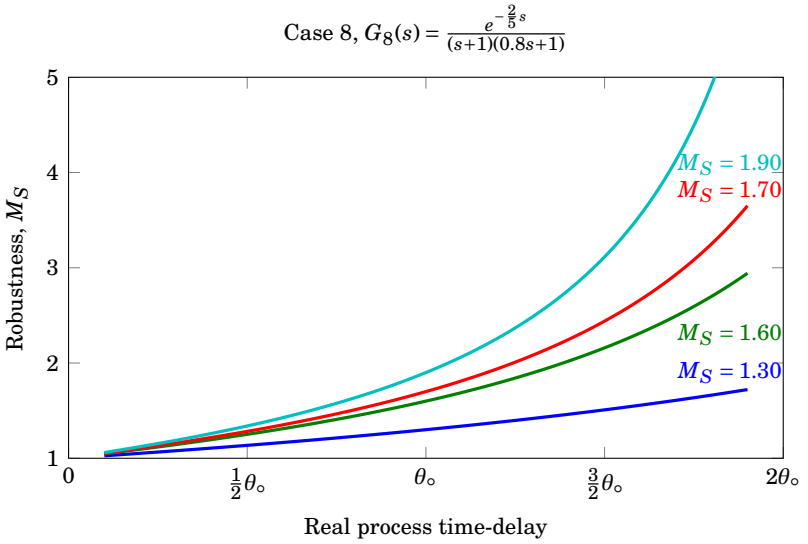
Process	$\min_K J(K)$					
	Cascade parameterisation					
	K_c	τ_I	τ_D	J	IAE_{d_o}	IAE_{d_i}
G_1	0.42	0.82	0.82	1.05	1.96	1.95
G_2	0.47	0.96	0.96	1.06	2.05	2.03
G_3	0.40	0.73	0.73	1.04	1.86	1.83
G_4	1.12	0.68	0.68	1.12	0.76	0.61
G_5	0.84	0.90	0.90	1.09	1.15	1.07
G_6	1.63	0.47	0.47	1.18	0.53	0.29
G_7	1.52	0.63	0.63	1.17	0.63	0.41
G_8	1.13	0.88	0.88	1.12	0.89	0.78
G_9	2.24	0.42	0.42	1.24	0.44	0.19
	Alternative parameterisation					
	K	I	D	J	IAE_{d_o}	IAE_{d_i}
G_1	1.91	0.30	0.24	1.01	1.93	1.83
G_2	2.13	0.26	0.28	1.02	2.00	1.90
G_3	1.81	0.34	0.21	1.01	1.85	1.74
G_4	4.79	0.36	0.17	1.11	0.78	0.59
G_5	3.55	0.28	0.24	1.06	1.16	1.02
G_6	7.56	0.47	0.10	1.18	0.53	0.29
G_7	6.60	0.39	0.15	1.16	0.64	0.40
G_8	4.70	0.30	0.23	1.10	0.92	0.72
G_9	10.89	0.50	0.09	1.24	0.44	0.19

cess time-delay is $0.1\theta_\circ \leq \theta \leq 0.9\theta_\circ$, while the controllers are Pareto optimal for $\theta = \theta_\circ$. Figure 5.2 and 5.3 illustrate how the performance and robustness efficiency varies with changes in the process time-delay for a SOPTD and FOPTD, respectively. The behaviour is as expected, with less time-delay the performance and robustness of the controller increase, while they deteriorate for increasing time-delay. The system does not reach instability in the simulated time-delay error domain.

The complete set of plots for the Pareto optimal PID controller sensitivity to time-delay variations are listed in Figures C.15 through C.28 in Appendix C.1.

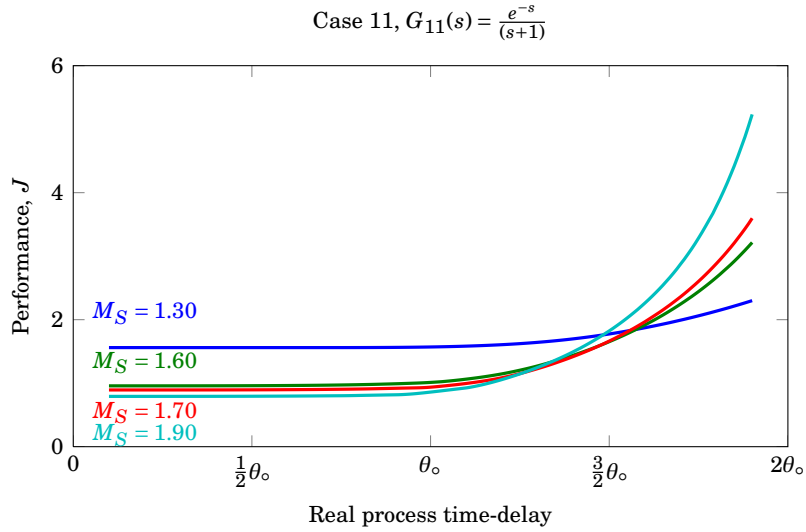


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

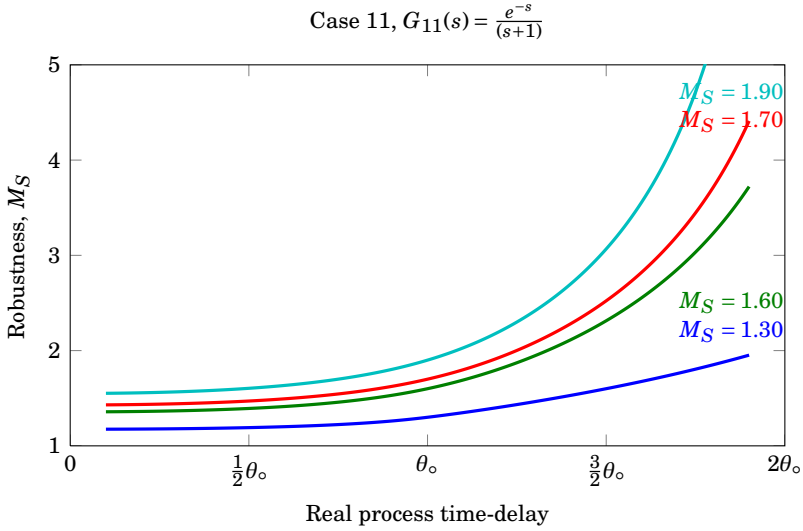


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure 5.2 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.



(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.



(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure 5.3 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_{11}(s) = \frac{e^{-s}}{(s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

5.2.3 PI vs. PID Control

Time-delay dominated processes For the time-delay dominated processes in case 1, 2 and 3, the improvement in performance when using a PID controller over a PI controller is 30 %, 34 % and 27 %, respectively, for the cascade controller at $M_S = 1.59$.

Lag-time dominated processes The performance can be vastly improved by switching from PI to PID controller for the lag dominated processes. When the ratio between the time-delay and lag-time constant decrease, the improvement seem to increase. On average, an improvement in performance of 53 % with cascade PID controller is observed for the lag-time dominated processes. For the alternative parallel parameterisation, the improvement is 54 %, which confirms that the cascade controller is not underperforming mentionably compared to the alternative parallel controller parameterisation.

The complementary sensitivity function peak value, M_T, is less than M_S for all cases. The parallel parameterisation yield a slightly better M_T value, though without dramatic variations.

5.3 SIMC Control

An illustrative example of the SIMC PI and PID tuning curves is given in Figure 5.1(a). The complete set of SIMC tuning curves for the SOPTD cases are given in Figure B.1(a) through B.9(a) in Appendix B. The tuning curves was found by computing the SIMC tuning for a given τ_c and evaluating performance and robustness for the system. Both PI and PID SIMC tuning rules was found, where the "half rule" method was used for model reduction to yield the FOPTD processes necessary to find SIMC PI tuning. The algorithm is outlined in Algorithm 2 in Chapter 4.3. Reference points was calculated for $\tau_c/\theta \in [½\ 1\ ⅔\ 2]$. Tuning, cost and IAE trade-off values for the reference points are listed in Table B.3 and B.4. A summary for the SIMC PI reference points are given in Table 5.4, while a summary for the SIMC PID reference points are given in Table 5.5.

5.3.1 SIMC PI Control

The SIMC PI controller tuning achieves in general close to Pareto optimal performance in the extremely robust region ($M_S \leq 1.4$). Apart from this, the performance can be divided between the time-delay dominated processes and the lag-time dominated processes. Larger time-delay generally deteriorates

Table 5.4 – SIMC PI tuning summary for SOPTD models.

Process	τ_c/θ	τ_c	K_c	τ_I	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	1	1.25	0.44	1	1.59	1.79	3.14	2.8
	⅔	1.88	0.35	1	1.7	1.56	3.24	3.1
$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	1	1.4	0.42	1	1.83	1.84	3.74	3.3
	⅔	2.1	0.32	1	1.91	1.59	3.78	3.57
$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	1	1.15	0.47	1	1.46	1.73	2.73	2.46
	⅔	1.72	0.37	1	1.6	1.53	2.89	2.81
$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	1	0.58	1.09	1	2.25	1.89	1.69	1.11
	⅔	0.87	0.83	1	2.45	1.62	1.68	1.32
$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	1	0.93	0.68	1	2.32	1.89	2.7	2.07
	⅔	1.4	0.52	1	2.45	1.62	2.69	2.33
$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	1	0.35	1.82	1	2.25	1.89	1.01	0.55
	⅔	0.52	1.38	1	2.57	1.62	1.01	0.72
$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	1	0.5	1.33	1	2.64	1.9	1.49	0.89
	⅔	0.75	1	1	2.91	1.63	1.48	1.09
$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	1	0.8	0.83	1	2.7	1.9	2.38	1.69
	⅔	1.2	0.62	1	2.89	1.63	2.37	1.94
$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	1	0.3	2.22	1	2.71	1.9	0.89	0.45
	⅔	0.45	1.67	1	3.14	1.63	0.89	0.6

the performance of the SIMC PI tuning, with a maximal loss at approximately 30 % for case 8 compared to Pareto optimal PI control.

Time-delay dominated processes The SIMC PI tuning achieves close to Pareto optimal tuning at high robustness ($M_S \leq 1.4$). The optimality deteriorates with increasing magnitude of the minor lag-time constant.

The difference in performance when choosing $\tau_c = ⅔\theta$ (smooth tuning) over the recommended setting of $\tau_c = \theta$ (robust tuning) is small for the time-delay dominated processes. The increase in performance when $\tau_c = \theta$ is 6 %, 4 % and 8 %, respectively for case 1, 2 and 3, when compared to $\tau_c = ⅔\theta$. The deterioration in robustness is −14 %, −15 %, and −13 %. However, all M_S values for $\tau_c = \theta$ are within the recommended limit of $M_S \leq 2$.

Lag-time dominated processes The decrease of optimality in the lag-time dominated processes resembles that of the time-delay dominated processes, except that the M_S values are shifted to a slightly higher value for τ_c.

Table 5.5 – SIMC PID tuning summary for SOPTD models.

Process	τ_c/θ	τ_c	K_c	τ_I	τ_D	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	½	0.5	0.67	1	0.5	1.02	1.92	2.13	1.67
	1	1	0.5	1	0.5	1.12	1.59	2.17	2.02
$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	½	0.5	0.67	1	0.8	0.97	1.92	2.13	1.59
	1	1	0.5	1	0.8	1.09	1.59	2.17	2.01
$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	½	0.5	0.67	1	0.3	1.08	1.92	2.13	1.71
	1	1	0.5	1	0.3	1.18	1.59	2.17	2.03
$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	½	0.17	2	1	0.5	0.98	1.92	0.71	0.5
	1	0.33	1.5	1	0.5	1.14	1.59	0.72	0.67
$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	½	0.27	1.25	1	0.8	0.94	1.92	1.14	0.8
	1	0.53	0.94	1	0.8	1.09	1.59	1.16	1.07
$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	½	0.1	3.33	1	0.3	1.07	1.92	0.43	0.3
	1	0.2	2.5	1	0.3	1.27	1.59	0.43	0.4
$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	½	0.12	2.67	1	0.5	1.03	1.92	0.53	0.38
	1	0.25	2	1	0.5	1.21	1.59	0.54	0.5
$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	½	0.2	1.67	1	0.8	0.96	1.92	0.85	0.6
	1	0.4	1.25	1	0.8	1.13	1.59	0.87	0.8
$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	½	0.07	4.44	0.9	0.3	1.12	1.94	0.34	0.2
	1	0.15	3.33	1	0.3	1.39	1.59	0.33	0.3

Almost optimal tuning is achieved in case 6 and 9, that is, when the minor lag-time constant is small. Small minor lag-time constants produces less additional time-delay added through the "half rule", and the nominal time-delay of these processes are in general small.

The SIMC PI tuning achieves practically identical robustness levels for the different tuning parameters $\tau_c/\theta \in [½\ 1\ ⅔\ 2]$, all with the recommended tight tuning ($\tau_c = \theta$) attaining robustness levels of $M_S < 1.9$. The more aggressive tuning with $\tau_c = ½$ are outside the recommended robustness region. The loss when applying $\tau_c = ⅔\theta$ over $\tau_c = \theta$ is increasing with increasing τ_2/θ ratio. A potential trend of increasing loss with decreasing time-delay is observed, indicating a more aggressive tuning is favorable.

5.3.2 SIMC PID Control

For all values of $\tau_c/\theta \in [½\ 1\ ⅔\ 2]$, the reported robustness is within the recommended limits of $M_S \leq 2$. As this is not the case for the PI tuning, the increased robustness can be explained by the added derivative action. The general trend is that the SIMC PID tuning is close to the Pareto optimal

tuning for a cascade controller.

Time-delay dominated processes For the lag-time dominated processes, the SIMC PID tuning optimality is increasing with greater values of the minor lag-time constant. For case 2, the SIMC tuning with a closed-loop time constant of $\tau_c = \theta$ is practically coinciding with the Pareto optimal solution. For case 3, where the minor lag-time constant is small compared to the major, and the ratio τ_2/θ is small, the deviation from optimality for the SIMC rule is at its highest.

The more aggressive tuning with $\tau_c = ½\theta$ is within the stability limit of $M_S \leq 2$. The potential improvement in performance compared to $\tau_c = \theta$ is 9 %, 11 % and 8 % for case 1, 2 and 3, respectively, with a corresponding deterioration in robustness of −20 % for all cases.

Lag-time dominated processes The SIMC tuning achieves high degree of optimality compared to the Pareto optimal controllers for the lag-time dominated processes. The trend of discrepancy between SIMC and Pareto optimal tuning for processes with small minor lag-time constants and low τ_2/θ ratio is visible. The point with least fortunate performance for the SIMC rule is for case 9, where the SIMC tuning underperform by 10 % compared to the Pareto optimal tuning.

By switching to more aggressive tuning, the performance can be improved by between 13 % (case 5) and 19 % (case 9).

5.3.3 SIMC PI vs. SIMC PID Control

The general trend is that the SIMC PI tuning is close to the Pareto optimal PI tuning. The performance of the tuning rule deteriorates with increasing second-order dynamics of the system, independent of dominance of time-delay in the process. This is expected, as greater second-order dynamics implies a higher degree of process approximation.

For the time-delay dominated processes, there is less improvement when switching from PI to PID control compared to the lag-time dominated processes. This is according to the improvement observed when switching from Pareto optimal PI to PID control.

The robustness achieved with SIMC PI control and $\tau_c = ⅜\theta$ equals the robustness of $\tau_c = \theta$ for SIMC PID control. By comparing the performance of the SIMC PI and PID rules it is observed that the increase in performance for the time-delay dominated processes in cases 1, 2 and 3 is 34 %, 42 % and 26 %, respectively. For the lag-dominated processes the increase in performance is

56 % on average, with the minimal improvement of 50 % in case 6, and the highest improvement of 64 % in case 9.

5.3.4 Complementary Sensitivity

The complementary robustness is generally reduced by a value of 0.2 when using SIMC PID rules over PI tuning. Both the SIMC PI and PID have slightly less optimal M_T values compared to the Pareto optimal values. In cases 6 through 9 the SIMC PID tuning achieve better M_T values in certain ranges for M_S, compared to the Pareto optimal controllers.

5.4 Summary: Second Order Processes

In the performance-robustness trade-off region, the difference between the cascade and parallel PID controller is negligible. At extremely high robustness, the potential increase in performance is approximately 10−20 %.

For the time-delay dominated processes, the improvement when switching from PI to PID control was found to be approximately 30 %. For the lag-time dominated processes, the increase was at approximately 50 %. For second-order processes it may seem as PID control is the proper way to provide good control.

The SIMC PI tuning are close to optimal for small second-order lag-time constants (τ_2), while the performance deteriorate with higher τ_2. The worst-case scenario is observed in case 8, where the SIMC PI underperform by 30 % compared to Pareto optimal PI. Choosing $\tau_c = \theta$ for SIMC PI provide somewhat aggressive tuning with $M_S \approx 1.9$, though the robustness is within the recommended limit of $M_S \leq 2$.

The SIMC PID tuning rules are displaying practically optimal performance. The closed-loop constant for tight tuning provide an almost constant level of robustness with $M_S \approx 1.6$ for $\tau_c = \theta$. This is true as long as $\tau_1 < 4(\tau_c + \theta)$, as this for the SIMC rules yields zero-pole cancellation, thus stabilising the robustness.

CHAPTER 6

OPTIMALITY OF THE SMITH PREDICTOR

6.1 Introduction

In this chapter, the Smith predictor controller structure described in Chapter 2.4 is evaluated. Pareto optimal tuning curves for PI and PID Smith predictor controllers are found and compared to the Pareto optimal PI and PID tuning curves for the FOPTD and SOPTD models. Both the primary controller in the Smith predictor structure and the Pareto optimal PI and PID controllers are of cascade parameterisation. A general robustness boundary for the Smith predictor is found. The controller gain, integral time and derivative time found from the optimisation is put in context with the robustness.

During the optimisation routine, the question of stability was not directly assessed. It was assumed that the IAE calculations in combination with the robustness target would provide closed-loop stable Pareto optimal solutions. This proved true as all of the Pareto optimal controllers produced closed-loop systems with non-negative gain and phase margins. However, Adam et al. (2000) reports in his article that the Smith predictor can display some unintuitive stability behaviour when modelling errors in the time-delay parameter occur. He claim that a closed-loop system with a Smith predictor controller structure can become unstable if the true time-delay of the process is less of what the Smith predictor model time-delay is. From an heuristic point of view controllers would be tuned for a worst-case modelling error

scenario. The allegation of having reduced time-delay being associated with instability could pose some serious challenges if there is a high degree of uncertainty or expected variances in the time-delay parameter.

The stability issues concerning time-delay modelling errors discussed by Adam et al. (2000) are investigated. The sensitivity to time-delay modelling errors in the Smith predictor structure are compared to the behaviour of the Pareto optimal PI and PID controllers when they undergo an equivalent time-delay modelling error. The analysis is based on modelling errors assumed to be within $\pm 90\%$ of the nominal time-delay, and is performed for all FOPTD and SOPTD processes. Variations in performance and robustness when modelling errors occur are investigated and instability issues are discussed.

To avoid referring to "the robustness of the robustness" or "the robustness performance" when the M_S values are changing due to time-delay error, the term "robustness efficiency" is introduced. High robustness efficiency equals low M_S value, and vica versa.

6.2 Pareto Optimal Smith Predictor

The Pareto optimal Smith predictor PI and PID controller tunings for case 1–14 was found by following the principle outlined in Algorithm 1 in Chapter 3.7. The cascade controller parameterisation was used. For the Smith predictor Pareto optimal PID controller optimisation, the `SQP` routine used by `fmincon` proved inefficient. It repeatedly found local solutions which usually displayed the expected curvature of a Pareto optimal curve, but which proved to be non-optimal. The `fminsearch` routine tended to be more robust, but less efficient with respect to computation time. Supplying good initial values for the optimisation routine was crucial regardless of routine. The complete set of initial values used can be found in Appendix M.6.

A summary of the Pareto optimal controllers with corresponding IAE weights and cost function values for a given robustness target is presented in Table 6.1. The complete set of Pareto optimal Smith predictor plots are listed in Appendix D. Figure D.1(a) through D.14(a) illustrate the performance as a function of robustness. Figure D.1(b) through D.14(b) show how the maximum additional time-delay before instability occur ($\theta_{\max}$) varies with changing robustness. $\theta_{\max}$ is defined according to Equation (3.8) in Chapter 3.3.

In Appendix E the sensitivity to modelling error in the time-delay parameter is presented. In Section E.1 the sensitivity of the Pareto optimal Smith predictor PI controller is illustrated, while the Pareto optimal Smith predictor PID controller sensitivity is presented in Section E.2. Figures E.1(a)

through E.14(a) and Figures E.15(a) through E.28(a) show how the performance of the controller changes when modelling error is introduced. In Figure E.1(b) through E.14(b) and Figure E.15(b) through E.28(b) the robustness' variations to time-delay modelling errors are illustrated. It should be noted that the M_S domain investigated is not completely consistent; for some of the investigated cases the point $M_S = 2.00$ is included, while for some the last robustness frontier is $M_S = 1.99$ [a]. Whenever instability occurs, $\theta_{\max}$ is set to zero. In this way, discontinuities in stability domain are clearly visible. The alternative of letting $\theta_{\max} =$ `NaN` provided single points in the plot, which is graphically less visible.

Pareto optimal Smith predictor PI and PID controllers are compared to Pareto optimal PI and PID controllers in Figure D.1(a) through D.13(a). Figure D.1(b) through D.13(b) illustrates the maximum additional time-delay that can be added before the system reaches instability according to Equation (3.8) in Chapter 3.3.

The robustness of the Smith predictor is bounded by $M_S \leq 2$. This limit originates from the definition of the Smith predictor, and the existence and value of the boundary is proven in Appendix H. Consequently, the Smith predictor controllers have not been attempted evaluated at M_S values higher than 2, as this is an unfeasible domain.

6.2.1 Smith Predictor PI Control

Consider the Pareto optimal Smith predictor tuning curves for the FOPTD processes, given in Figure D.10(a) through D.14(a) in Appendix D. The Smith predictor PI controller are performing lesser for the integrating process in case 10 than for the regular PI controller. The behaviour of case 14 is given in Figure 6.1, where the Smith predictor is displaying a small performance improvement near $M_S \approx 1.25$. Decreasing the lag-time constant is improving Smith predictor performance, and performs slightly better compared to the Pareto optimal PI controller for case 12. In case 11, the increase in performance compared to Pareto optimal PI is about 18 %. Case 11, 12 and 14 have increasing integrating behaviour, while case 10 is a pure integrator. It may look as though the Smith predictor is not optimal for integrating processes, or processes that have a high degree of integrating behaviour. From the $\theta_{\max}$ plots, the systems are in general phase margin stable ($\theta_{\max} > 0$) in the examined robustness region, where $M_S = 1.99$ is the highest value evaluated.

a) The optimisation routines for the Smith predictor was extremely time consuming. The last M_S value was disregarded as it proved cumbersome to achieve convergence for some cases.

Table 6.1 – Optimal Smith predictor PI and PID cascade controllers for $M_S = 1.59$.

	Process	$\min_K J(K)$					
		K_c	τ_I	τ_D	J	IAE_{d_o}	IAE_{d_i}
PI-control	$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	0.85	1.15	-	1.27	2.37	2.37
	$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	0.86	1.35	-	1.35	2.58	2.58
	$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	0.89	1.02	-	1.21	2.15	2.15
	$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	1.64	1.28	-	1.90	1.20	1.12
	$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	1.31	1.52	-	1.70	1.75	1.70
	$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	2.29	1.06	-	2.24	0.80	0.69
	$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	1.87	1.29	-	2.34	1.08	0.95
	$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	1.52	1.56	-	2.01	1.53	1.44
	$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	2.59	1.06	-	2.84	0.73	0.58
	$G_{10}(s) = \frac{e^{-s}}{s}$	1.10	3.57	-	4.92	2.40	53.20
	$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	1.37	0.93	-	1.14	1.70	1.69
	$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	9.43	2.90	-	1.78	2.03	1.33
	$G_{13}(s) = e^{-s}$	0.72	0.32	-	0.92	1.47	1.47
	$G_{14}(s) = \frac{e^{-s}}{20s+1}$	22.28	3.15	-	2.63	2.23	1.08
PID-control	$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	1.16	0.75	0.74	0.90	1.68	1.67
	$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	1.29	0.90	0.88	0.89	1.72	1.71
	$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	1.14	0.65	0.65	0.90	1.61	1.59
	$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	3.88	0.79	0.51	0.94	0.59	0.56
	$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	2.56	0.90	0.75	0.89	0.92	0.89
	$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	5.10	0.44	0.36	1.01	0.53	0.41
	$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	4.75	0.60	0.50	0.98	0.58	0.46
	$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	3.08	0.77	0.81	0.99	0.77	0.69
	$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	6.97	0.39	0.33	1.10	0.50	0.39
	$G_{10}(s) = \frac{e^{-s}}{s}$	1.36	1.07	0.72	4.38	1.95	55.84
	$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	1.48	0.52	0.50	0.92	1.43	1.38
	$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	11.54	0.94	0.58	1.41	1.66	1.17
	$G_{13}(s) = e^{-s}$	0.69	0.30	0.01	0.90	1.43	1.43
	$G_{14}(s) = \frac{e^{-s}}{20s+1}$	7.39	0.47	2.37	2.39	1.70	1.04

Several of the performance curves have a small "bend" in the upper end of the M_S domain, which also is depicted in the corresponding $\theta_{\max}$ plot.

For the pure time-delay process, case 13, the Smith predictor show a small increase in performance compared to the Pareto optimal PI controller. An interesting behaviour is observed around $M_S \approx 1.92$ where the phase margin indicate that the system is unstable, while the performance starts deteriorating. By solely looking to the performance, one could assume the controller is leaving the Pareto optimal region. At around $M_S \approx 1.95$ the performance of the controller literally crashes (MATLAB returned $J > 10^{10}$), while the system appears stable again at $M_S = 2.00$.

A similar stability behaviour is also noticed for case 11, $\frac{e^{-s}}{(s+1)}$, where the phase margin is negative for $M_S = 1.99$ and positive for $M_S = 2.00$. The performance, however, is displaying an increase in optimality relative to the Pareto optimal PI controller towards the point where the phase margin turns negative. Except from the relative increase, the performance never indicate the instability of the system. The stability of the last point has not been further verified.

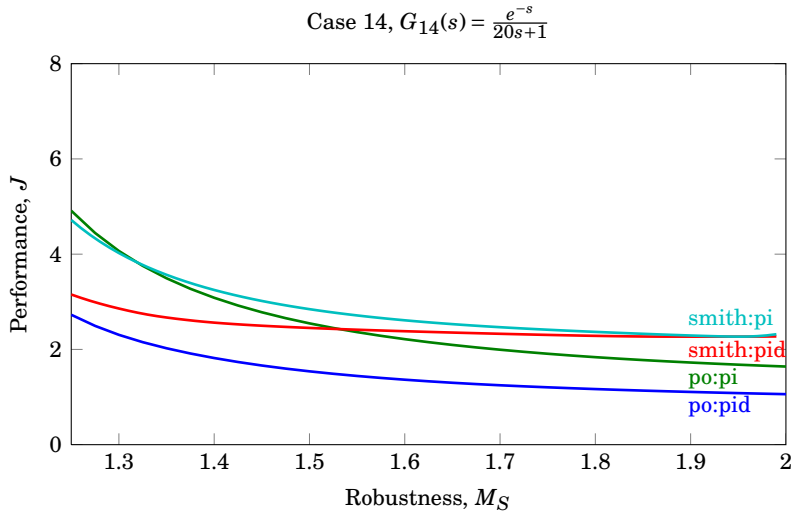
Time-delay dominated processes The Pareto optimal Smith predictor PI controller display a 15 % gain in performance for cases 1 and 2, and 16 % for case 3, for a given robustness of $M_S = 1.59$. At high M_S values, the Smith predictor PI controller is approaching the Pareto optimal PID controller in performance, and outperforming it in case 3. Its performance at low robustness seems dependent on the amount of second-order dynamics, as it outperforms the Pareto optimal PID in cases 1 and 3. In the robust region, the improvement of performance compared to the Pareto optimal PI controller is fairly constant, which is also confirmed by Table 6.1.

For the first-order time-delay dominated process, case 11, the increase in performance is 18 %. The controller renders an increase in performance when $M_S \to 2$, where it performs better than the Pareto optimal PID controller.

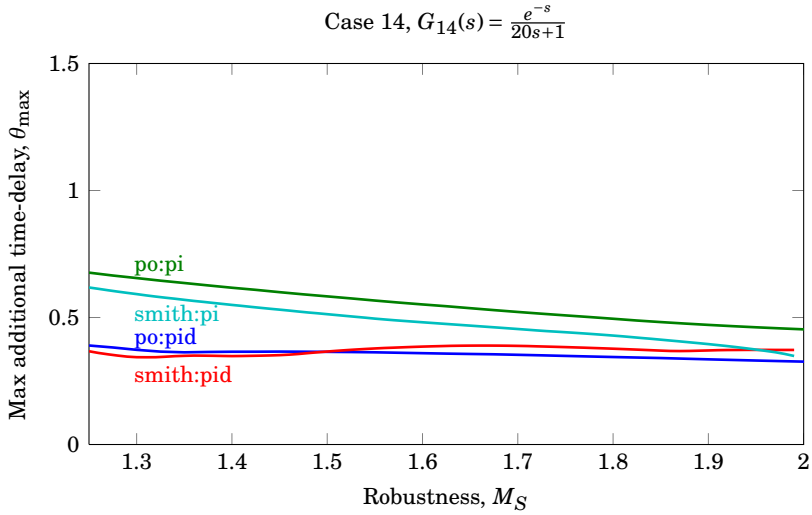
Lag-time dominated processes The lag-time dominated processes displayed an increase in optimality of between 12 % and 17 % for the SOPTD cases. The performance improvement tend to grow with increasing M_S values.

6.2.2 Smith Predictor PID Control

The Pareto optimal Smith predictor PID controller display behaviour similar to that of the Smith predictor PI controller. For the integrating cases, the



(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$



(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure 6.1 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_{14}(s) = \frac{e^{-s}}{20s+1}$.

Smith predictor PID controller is performing worse than the Pareto optimal PID controller. In the SOPTD models, the Smith predictor is in general displaying a small performance improvement compared to the Pareto optimal PID controllers. Case 4 is used as an illustrative example, given in Figure 6.2.

Time-delay dominated processes For the time-delay dominated processes, the increase in performance for the Pareto optimal Smith predictor PID controllers compared to the Pareto optimal PID controllers is 15 % for cases 1 and 2, and 9 % for case 3 for $M_S = 1.59$.

Lag-time dominated processes The increase in performance varies between 11 % and 18 % for the lag-time dominated cases. The increase in performance with respect to the Pareto optimal PID tuning is not as evident as for the Smith predictor PI controller.

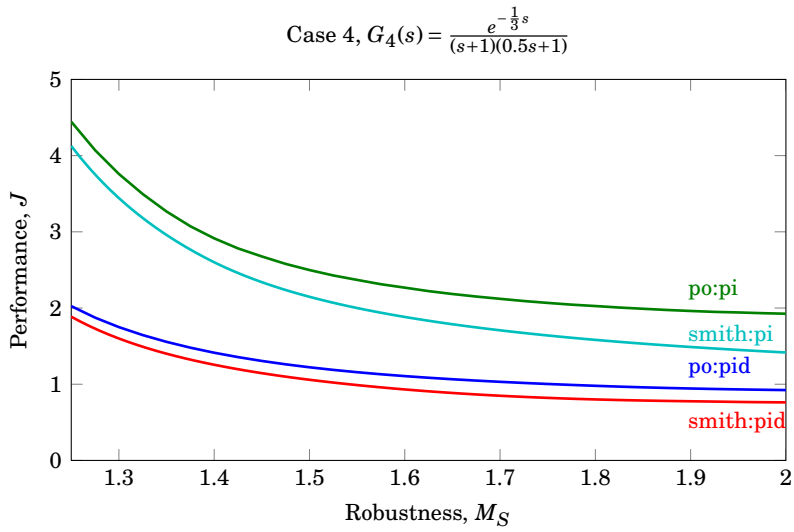
6.3 Controller Action

The controller parameters for the Pareto optimal PI and PID controllers have been compared to those of the Pareto optimal Smith predictor controller. Through graphical representation it is possible to qualitatively evaluate convergence of the optimisation routines, and easier assess the optimal controller structure.

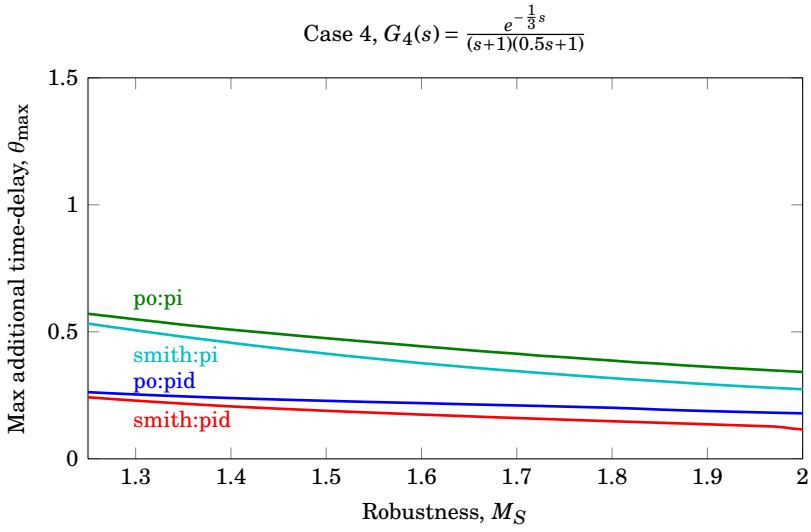
For the time-delay dominated second-order processes the gain, integral and derivative time of the Pareto optimal PID controller are displaying a homogeneous behaviour throughout the studied robustness region. The integral and derivative time are identical, which is what Ziegler and Nichols (1942) recommends, and concur with the results of Grimholt and Skogestad (2013). For the Pareto optimal Smith predictor the behaviour is different, with gain increasing with increasing M_S, and approaching infinity when M_S is approaching it's upper boundary for the system. This trend is consistent for the FOPTD processes. The integral time and derivative time are almost equal and fairly constant.

For the SOPTD dominated processes the Pareto optimal PI controller gain and integral time behaviour can be divided between the region of high robustness, and the remaining M_S region. A non-smooth buckling point is observed at $M_S \approx 1.4$. For the Pareto optimal Smith predictors, the buckling point is shifted slightly to the right in the robustness region.

For the Pareto optimal PID controllers, the gain, integral time and derivative action display smooth behaviour. The Smith predictor Pareto optimal so-



(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$



(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure 6.2 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$.

lutions are displaying two-parted behaviour. The gain increases to a certain M_S value, where it suddenly decreases. The integral time is usually quite constant and experience an increase which start at about $M_S = 1.8$, while the derivative time decrease in the same region.

By using a Smith predictor, the time-delay is eliminated from the closed-loop behaviour, which should increase robustness. The Smith predictor should be able to have a higher controller gain compared to a normal PI or PID controller at the same M_S value. Higher gain will yield better performance. From the gain plots, it seems as this is the case. The difference in gain between the Pareto optimal PI and PID controllers and the Smith predictor PI and PID controllers are distinct, while the integral and derivative time mainly are proportional.

An important difference between the Smith predictor plots and the regular controller plots, are the lack of smooth behaviour in Smith controller parameters. The trend in the plots are clear, but non-smooth variations are present. This illustrates that the optimisation routines may not have converged to the true optimum for all M_S values, but have found some local optimum close to the global optimum. When the M_S values are calculated, the entire M_S region of interest is gridded and the maximum peak value in the grid is found. This gives a good indication of where the M_S value is, but not necessarily a value with high numerical precision. For the Smith predictor, small fluctuations in M_S result in huge variations in controller gain. Thus the Smith predictor curves may not be the true Pareto optimal curves, but they give a good representation of the Pareto optimal behaviour.

6.4 Stability of the Smith Predictor

The stability of the Smith predictor has been investigated based on modelling error in the time-delay parameter. It has been assumed that the non-delay dynamics of the process model is completely known, such that $\widetilde{G}_\circ = G_\circ$. The uncertainty in the time-delay parameter is given by

$$\mathcal{Q} \triangleq \left\{\theta \in \mathbb{R} : \theta \geq 0,\, \theta_\circ + \delta\theta^- \leq \theta \leq \theta_\circ + \delta\theta^+\right\}, \tag{6.1}$$

where $\theta_\circ$ is the nominal modelled time-delay for the Smith predictor process model ($\widetilde{G}$), $\delta\theta^-$ and $\delta\theta^+$ are lower and upper boundary limits, respectively, for the model error, and θ is the true time-delay of the process (G). It has been assumed that the true time-delay may differ from the modelled time-delay with $\pm 90\,\%$, such that $\mathcal{Q} = [0.1\theta_\circ, 1.9\theta_\circ]$.

Stability was quantified by positive gain margin and phase margin, and was evaluated at different levels of closed-loop robustness. The set of M_S

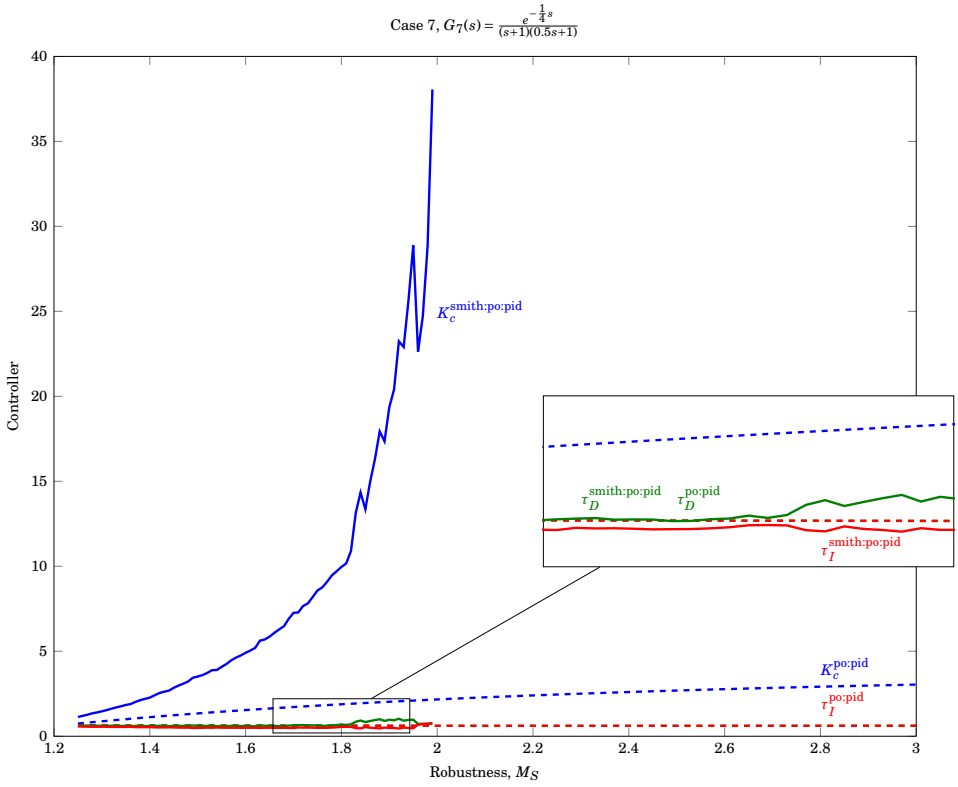


Figure 6.3 – Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line, τ_I and τ_D are superimposed) and the Pareto optimal Smith predictor PID controller, as functions of robustness, for $G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$.)

values evaluated is given by

$$\mathscr{M}_S = \begin{bmatrix} 1.3 & 1.6 & 1.7 & 1.9 \end{bmatrix}. \tag{6.2}$$

The Pareto optimal Smith predictor controllers were assumed for the study, while the time-delay for the process models (case 1 through 13) was varied according to $\mathscr{Q}$ given in Equation (6.1). The results were visualised as $J = f(\theta)$ and $M_S = f(\theta)$. For PI control, the results are given in Figure E.1(a) through E.14(a) and Figure E.1(b) through E.14(b) in Appendix E.1. The equivalent results for PID control are given in Figure E.15(a) through E.28(a) for performance, and Figure E.15(b) throught E.28(b) for robustness.

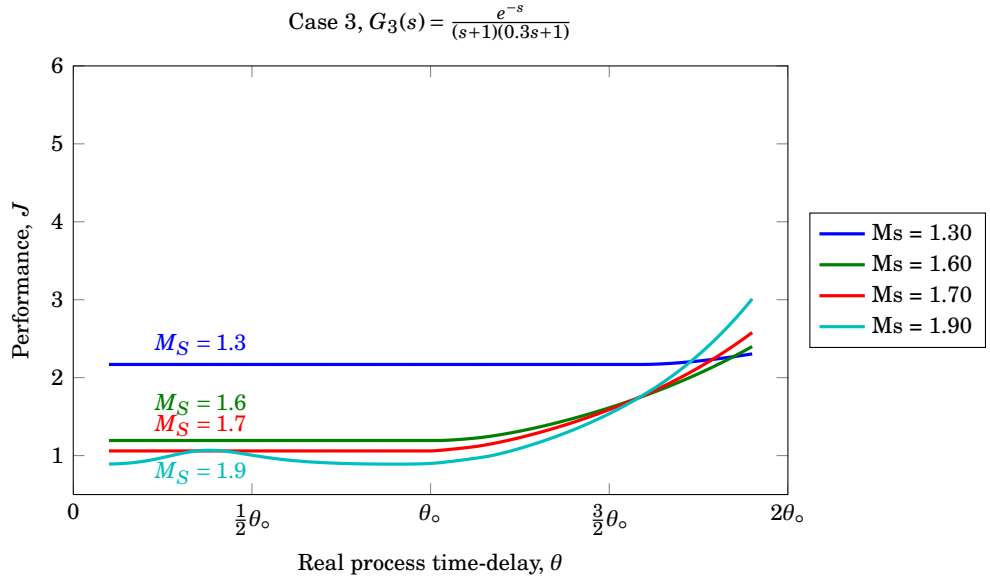
When finding J, M_S and $\theta_{\max}$ for a given θ, the close-loop system was tested for stability by examining the phase and gain margins, which should be greater than zero for a stable system. Whenever the system proved unstable, the values for M_S returned by MATLAB was surprisingly good. Typically, the M_S value could approach infinity for a given θ, and stabilise at $M_S \leq 1.5$ for $\theta + \epsilon$, where $\epsilon \ll \theta$. The definition of M_S depends on closed-loop stability, as steady-state behaviour is necessary to achieve a bounded peak value. For some reason, MATLAB returns some value for M_S even though the peak is at infinity. The exact reason for this behaviour has not been pursued, though it is returning issue that MATLAB doesn't always handle unstable systems in an easy and intuitive way.

6.4.1 Smith Predictor PI Controller Sensitivity

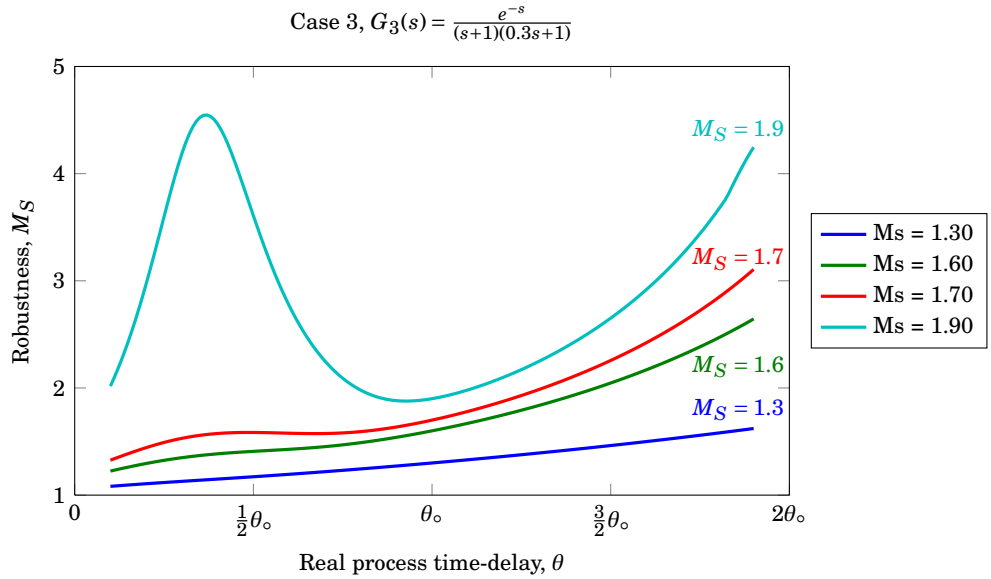
The Smith predictor performance when time-delay modelling error is introduced are graphically presented in Appendix E.1 for the PI controller, and in Appendix E.2 for the PID controller. The corresponding plots for the Pareto optimal PI and PID controllers are illustrated in Appendix C, Sections C.1 and C.2. The robustness efficiency plots are presented along with the associated performance plots.

The performance of the Smith predictor PI controller and the Pareto optimal PI controller are showing similar behaviour for most of the SOPTD processes. Both the performance and the robustness are increasing in optimality when the time delay is reduced, and the values increase when the time-delay modelling error yields an increase in process time-delay. This behaviour matches the intuitive understanding of robustness — a controller will not perform worse than some worst-case situation. This is the case for the regular PI and PID controllers, illustrated in Figure 6.4.

One exception is case 3 (Figure 6.4), where the $M_S = 1.9$ target curve is displaying a tendency to increase when the true time-delay is about half of



(a) Performance sensitivity, $J = f(\theta)$.



(b) Robustness sensitivity, $M_S = f(\theta)$.

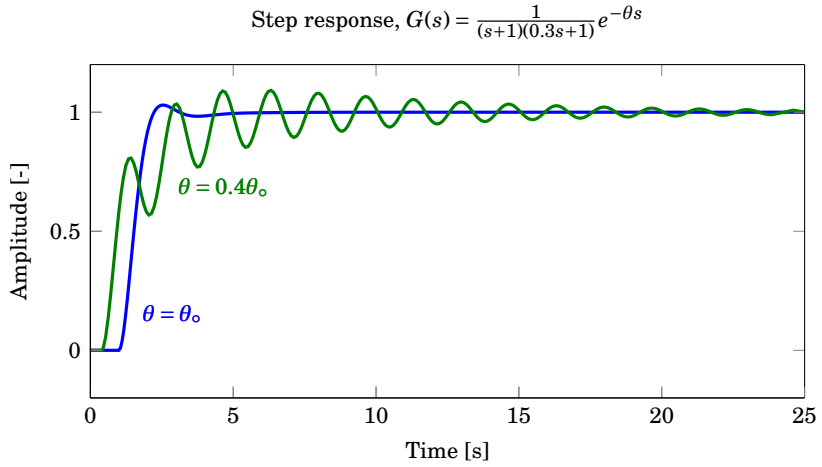
Figure 6.4 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$.

the nominal time-delay. The corresponding robustness plot shows the M_S value is violently increasing at this point. The closed-loop behaviour is illustrated in Figure 6.5(a), where a step in both the process with nominal delay and with reduced delay are plotted. Notice how the optimised system yields a smooth response for the nominal time-delay, $\theta = \theta_\circ$, while the reduction in time delay to $\theta = 0.4\theta_\circ$ give a highly oscillatory response. The response for the disturbed system is actually very good until the first oscillation occurs. This causes the overall performance in terms of IAE to be surprisingly good, but with a robustness of $M_S = 4.46$. The Bode plot of the disturbed system is given in Figure 6.5(b), which show that even though the phase margin is excellent, the gain margin of the disturbed system is small. This also reflect the high M_S value.

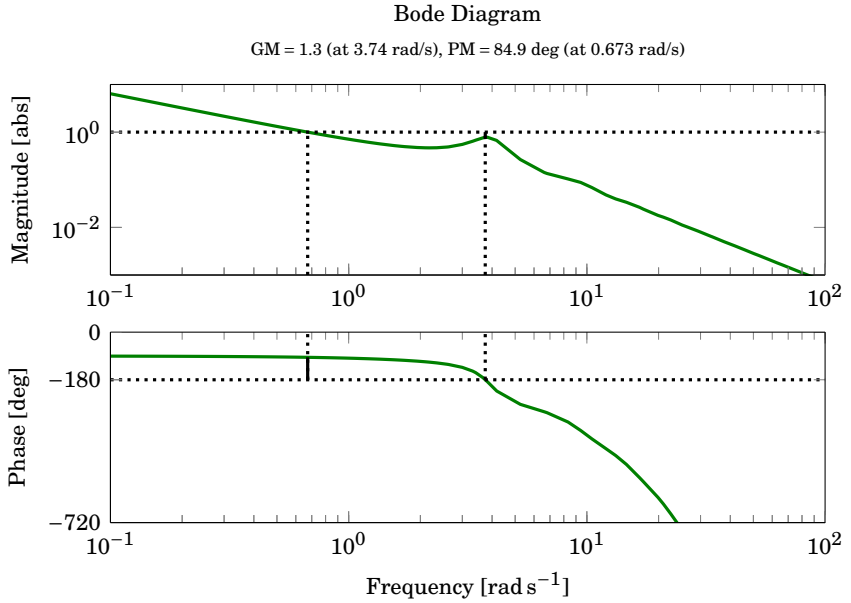
For the FOPTD processes, the sensitivity of the Smith predictor PI controller is illustrated at a more extreme level compared to the SOPTD process in case 3. For the pure integrating process in case 10 ($\frac{e^{-s}}{s}$), the robustness efficiency is starting to oscillate around $\theta = 0.4\theta_\circ$ at $M_S = 1.6$. At $M_S = 1.7$ the oscillation is obvious, and at $M_S = 1.9$ the system is unstable. However, at $\theta = 0.5\theta_\circ$, MATLAB report the system to be stable in terms of gain and phase margins. However, The system response amplitude is increasing with time, and thus seems unstable. A closer look at the Bode plot of the system reveals that MATLAB is not picking up the first crossing of the absolute magnitude unity line, and thus report the unstable system to be stable. As the step response never reach zero off-set, the cost function value is dependent on the simulation time, and is achieving very high values for the apparently stable regions. As of this, one need to evaluate the performance plot together with the robustness efficiency plot — if the robustness seems too good to be true, it probably isn't.

A similar robustness-performance pattern as described above is observed for all the FOPTD processes.

For the pure time-delay process, the only M_S target displaying a predictable (stable) behaviour is found for $M_S = 1.3$. Even though one could use $M_S = 1.6$, the robustness deteriorates almost instantly to $M_S \approx 4$, yielding small robustness margins. The IAE does, as previously described, not really depict the closeness to instability because of a good initial response to the step load change, but for $\theta = ½\theta_\circ$ and $\theta > ⅜\theta_\circ$ there is an obvious deterioration of performance. At $M_S = 1.7$ the system is stable within $\theta = [-0.01\theta_\circ, 0.06\theta_\circ]$, which in practice means it is almost marginally to time-delay disturbances.



(a) Step response for a closed-loop system, where $G = \frac{1}{(s+1)(0.3s+1)}e^{-\theta}$. The controller is a Pareto optimal Smith predictor PI controller, optimised for $G(\theta = \theta_\circ)$ and $M_S = 1.9$. The blue line illustrate the response with $G(\theta = \theta_\circ)$. For the green response, $G(\theta = 0.4\theta_\circ)$.



(b) The Bode plot of the system where $G = \frac{1}{(s+1)(0.3s+1)}e^{-0.6\theta_\circ}$, and the Smith predictor PI controller is optimised for $\theta = \theta_\circ$.

Figure 6.5 – Step response for both the nominal and the disturbed process, along with the Bode plot for the disturbed process.

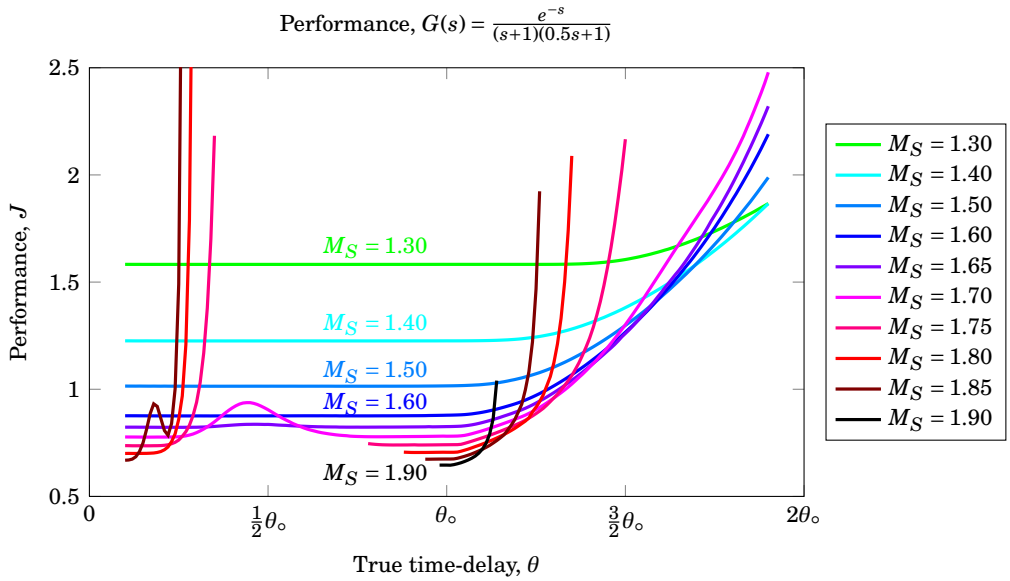


Figure 6.6 – Example of performance deterioration and instability when exposing a Pareto optimal Smith predictor PID controller to time-delay modelling error. Discontinuities in the plot are caused by instability. The process plotted is $G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$

6.4.2 Smith Predictor PID Controller Sensitivity

Figure 6.6 illustrates how the performance of a Pareto optimal Smith predictor PID controller can deteriorate when time-delay modelling error is introduced. The corresponding robustness plot is given in Figure 6.7. MATLAB's issues with providing the correct gain and phase margins for unstable systems are affecting the plots as discussed in Section 6.4.1. In Figure 6.6 and 6.7 this behaviour have manually been removed, but are still evident in the complete set of plots in Appendix E. One must always consider both the performance plot and the robustness efficiency plot in order to achieve the correct stability picture. If a surprisingly good value for M_S occur while the performance are not visible in the plotted scope, the system is unstable. A small value for M_S in a stable system have proven to always yield decent performance.

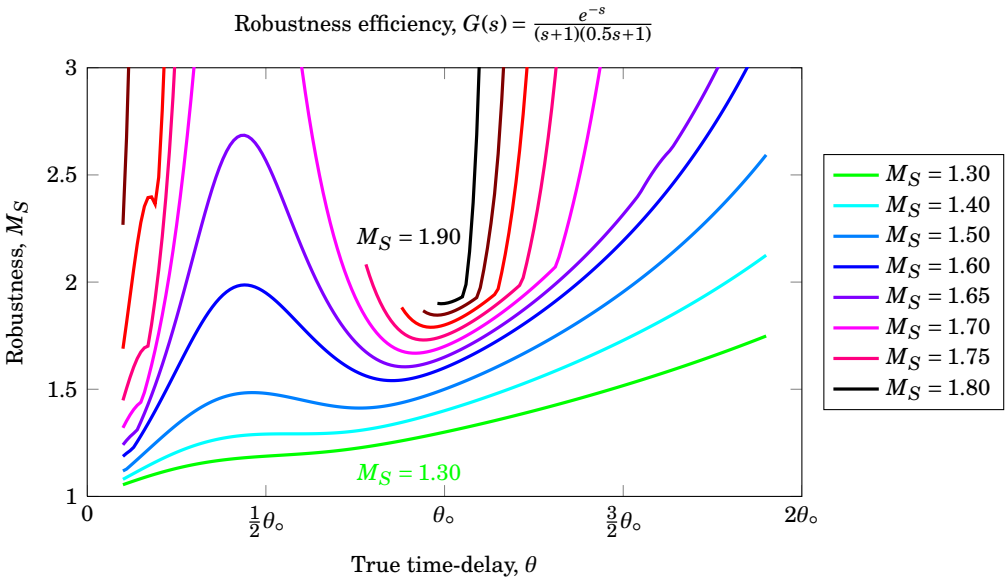


Figure 6.7 – Example of robustness deterioration and instability when exposing a Pareto optimal Smith predictor PID controller to time-delay modelling error. Discontinuities in the plot are caused by instability.

6.5 Summary: Smith Predictor

The Pareto optimal Smith predictor PI and PID tuning plots illustrate a relatively limited performance improvement compared to the Pareto optimal PI and PID controller tuning curves. The maximum additional dead-time that can be added before instability occur is steadily decreasing with increasing M_S value, and has a higher deterioration rate compared to regular PI and PID controllers.

For the majority of the FOPTD processes, the Smith predictor is less optimal compared to regular PI and PID controllers. This probably originates from the fact that the Smith predictor is derived for a first-order set-point response. With increasing integrating behaviour, the input disturbances are conducting the system response, and the Smith predictor fails the task of disturbance rejection compared to a regular PI or PID controller.

For the robust region, $M_S \leq 1.4$ the Smith predictor proves stable with respect to disturbances in the time-delay parameter. The sensitivity behaviour resembles that of the regular PI and PID controller. Thus, if one are to operate a SOPTD process in this very robust region, the potential performance

improvement compared to regular PI or PID controller can be realised. However, the performance improvement is not very high and can be realised by a regular PI or PID controller by tuning it slightly more aggressive. Typically, a robustness decrease from $M_S = 1.4$ to $M_S = 1.5$ for the Pareto optimal PID controller will yield performance equal to the Smith predictor.

For the less robust region, $M_S \geq 1.6$, or even as low as $M_S \geq 1.4$ for the FOPTD processes, the Smith predictor is highly sensitive to variations in the time-delay parameter. The sensitivity function (S) are starting to oscillate prior to observable deterioration in performance. When the oscillating behaviour is initiated, the M_S value increase rapidly, and the gain margin of the Smith predictor controller is decreasing. Importantly, reduction in the time-delay of the process tend to yield unstable controllers. Even for Smith predictor PID controllers tuned to a modest robustness level of $M_S = 1.7$, the M_S value quickly are pushed to values higher than 2 for small changes in the time-delay parameter.

It is also evident from the stability analysis that $\theta_{\max}$ is valid for a strictly positive direction. Several of the Smith predictor sensitivity plots show that the Smith predictor appear stable when the time-delay is increased more than $\theta_{\max}$.

The question of using M_S as a quantification of robustness for the Smith predictor naturally occurs when studying the behaviour of $M_S = f(\theta_{\max})$. When disregarding MATLAB's inability of providing proper M_S values for unstable systems[b],it appears as if M_S is monotonically decreasing with increasing robustness in the same way as for regular PI and PID controllers. Where the regular PI and PID controller have only an upper robustness boundary of $M_S = 1$ when designing the controller, the Smith predictor has a well defined upper M_S limit. This limit only applies for optimal tuning; it has been verified multiple times that the M_S value for systems operating with non-optimal Smith predictors easily exceed $M_S = 2$.

In the illustrative example used by Adam et al. (2000), the process is equivalent to case 11 in this thesis. Adam's controller is a P controller with $K_c = 4$. The proportional controllers are always optimal as they have only one degree of freedom, thus Adam is using an optimal controller. A proportional controller will never achieve zero steady-state offset, except for integrating processes, and the value of the cost function is dependent on the simulation time used. The M_S value for Adam's example system is $M_S \approx 1.7$. In light of the previous discussion on sensitivity to time-delay modelling error, the discontinuous stability domain described by Adam is not surprising. In ad-

b) Skogestad denoted the low M_S value for unstable systems as the system being "robustly unstable".

dition, Adam is applying a primary controller structure which should not be used in the Smith predictor. Neither the Smith predictor PI nor PID controller perform very well on the simple FOPTD in case 11, possibly due to the design basis of the Smith predictor. As a proportional controller is optimal for integrating processes only, an extensive analysis of the optimality and sensitivity of a Smith predictor P controller on purely integrating processes should be performed.

CHAPTER 7

SMITH PREDICTOR TUNING

7.1 Introduction

In this chapter, two approaches for the tuning of a Smith predictor controller structure is investigated:

i) "Robust Tuning" of Smith predictors (Normey-Rico and Camacho, 2007),

ii) SIMC tuning rules (Skogestad, 2003).

The rules given in the "Robust Tuning" will further be referred to as robust tuning rules.

7.1.1 Robust Tuning

The robust tuning rules given by Normey-Rico and Camacho (2007) is achieved by considering only time-delay error, assuming that the maximum dead-time-estimation error and underlaying process behaviour is known. The rules are based on the assumption that the process can be described by an FOPTD model. There are four parameters to be tuned, namely $K_c^\diamond$, T_i, $T_\circ$ and T_1. There is only one degree of freedom if the process model is set.

The controller used is a two-degree-of-freedom dead-time-compensator with a PI controller. The structure is illustrated in Figure 7.1. This scheme

uses a reference signal filter, F, defined as

$$F(s) \triangleq \frac{1+s\beta T_i}{1+sT_i}, \tag{7.1}$$

where β is a factor which in the tuning rules is chosen such that $\beta T_i = T_\circ$. From a condition of robust performance, $T_\circ = 1.7\Delta\theta$, where $\Delta\theta = \theta - \theta_\circ$ is the maximum time-delay error. As this study investigates disturbance loads in input and output, the reference filter for the robust tuning rules won't influence the results, and is disregarded from further study. The robust tuning rules then has one degree of freedom, namely $\Delta\theta$, and the rules are summarised in Table 7.1.

Table 7.1 – Robust tuning rules for the Smith predictor structure, using $T_\circ = 1.7\Delta\theta$

Predictor model	Controller	Filter	$K_c^\diamond$	T_i	T_1
$\frac{k}{1+sT}e^{-\theta s}$	$\frac{K_c^\diamond(1+T_i s)}{T_i s}$	$\frac{1+T_\circ s}{1+T_1 s}$	$\frac{1}{k}\frac{T}{T_\circ}$	T	$T_1 \in [T, T_\circ]$

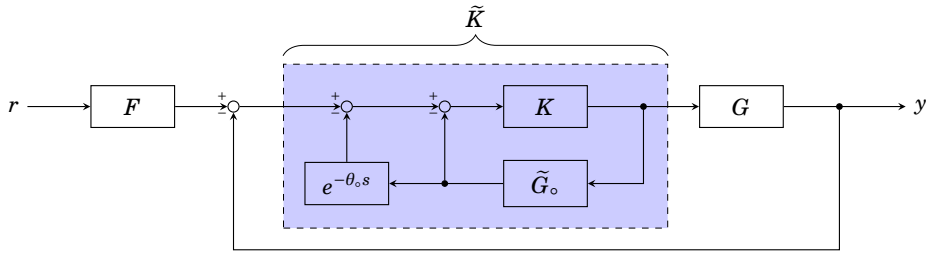


Figure 7.1 – Block diagram of the two-degree-of-freedom Smith predictor used by Normey-Rico and Camacho (2007).

7.1.2 SIMC Tuning

In Grimholt (2013), appliance of the SIMC tuning rules on a Smith predictor is analysed. The analysis is confirmed in Appendix I.

When applying the SIMC rules to a PID controller in the Smith predictor structure, the process governing the tuning is the SP model, $\widetilde{G}$, which have

no time-delay. Thus, the rules may be redefined to yield

$$K_c \triangleq \frac{1}{k}\frac{\tau_1}{\tau_c}, \tag{7.2a}$$

$$\tau_I \triangleq \min[\tau_1, 4\tau_c], \tag{7.2b}$$

$$\tau_D \triangleq \tau_2, \tag{7.2c}$$

where $\Delta\theta$ is the maximum time-delay error one wish to consider and it is required that $\tau_c > \Delta\theta > 0$.

7.1.3 Analysis

Comparing the robust tuning rules with the SIMC tuning rules yields the relations given in Equation (7.3).

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c} = \frac{1}{k}\frac{T}{T_\circ}, \tag{7.3a}$$

$$\tau_I = \min[\tau_1, 4(\tau_c)] = T, \tag{7.3b}$$

$$\tau_D = \tau_2 = 0. \tag{7.3c}$$

As the robust tuning rules are only given for PI control, the derivative term is compared with zero. From the comparison of the rules, $T_\circ = \tau_c$ and $T = \tau_1$ which relate the robust tuning rules and the SIMC tuning rules. The robust tuning rule differs from SIMC rules only in the minimum term in the integral time. Otherwise, the rules are equal in terms of only having different parameterisations, but the same solutions as long as $\tau_1 \leq 4(\tau_c)$.

7.2 Smith Predictor Tuning Results

The maximum dead-time-estimation error used is given by $\delta\theta^+ = 0.9$ from $\mathcal{Q}$, defined in Chapter 6.4. When finding PI tuning, the SOPTD models were reduced to FOPTD models by using the "half-rule" defined in Chapter 2.5.

For the study of the potential for using the SIMC tuning rules as tuning rules for the Smith predictor structure, the tuning parameters were calculated as for normal PI and PID tuning procedures (Chapter 8). The complete plot results are given in Appendix F. An illustrative case is given in Figure 7.2. The SIMC tuning rules provides tuning curves close to Pareto optimality. The reference points are

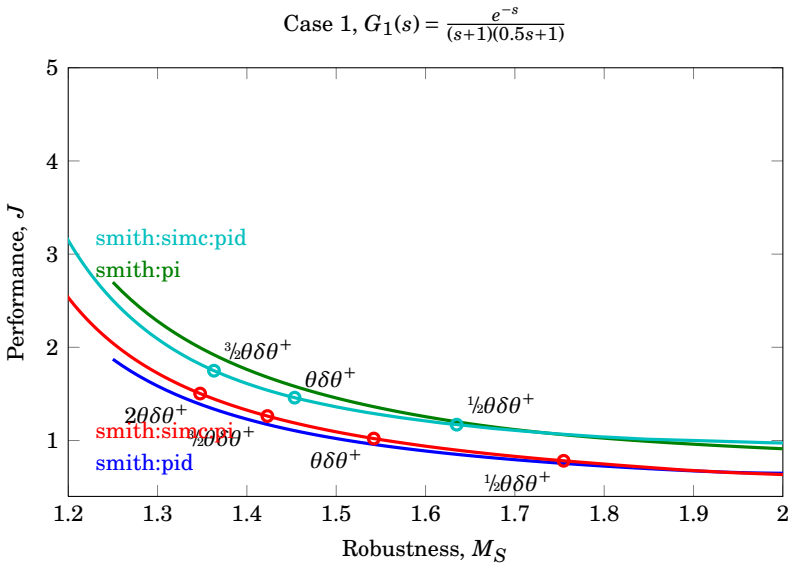


Figure 7.2 – Pareto optimal Smith PI and PID tuning compared with Smith SIMC PI and PID tuning for $G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$.

7.3 Summary: SIMC Tuning Rules for Smith Predictor

Analytically, the SIMC tuning rules are suited for tuning a Smith predictor. The tuning curves prove to be very close to the Pareto optimal Smith predictor tuning curves. The question of modelling error has not been addressed beyond the discussion in Chapter 6.

CHAPTER 8

CONCLUSION

Pareto Optimal Controllers

The cascade controller parameterisation for PID controllers are slightly less optimal compared to parallel PID controllers. The reduction in optimality is not severe in the M_S region which yield good trade-off between performance and robustness.

SIMC Tuning Rules

The SIMC tuning rules are achieving surprisingly good results for both FOPTD and SOPTD processes, considering the simplicity of the tuning rules.

The Smith Predictor

Assuming one operate the process in a highly robust region ($M_S \leq 1.5$), the Smith predictor display an increase in performance for SOPTD processes. The performance improvement is limited, and can be realised with regular PI or PID controllers by tuning the controller slightly less robust. For the FOPTD processes, the Smith predictor is mainly less optimal compared to regular PI and PID controller.

At lower robustness, equivalent with higher M_S values, the Smith predictor is extremely sensitive to time-delay modelling error. Importantly, for a Smith predictor aggressively tuned towards a robustness target of $M_S \geq 1.6$, the system is at risk of rendering unstable when the real time-delay of the

process is smaller than the nominal time-delay used in the Smith predictor model.

There are absolutely no reason for using a Smith predictor PI or PID controller over a Pareto optimal PI or PID controller. One will have to tune the Smith predictor extremely robust to avoid the instability issues when modelling error in the time-delay parameter occurs. The potential increase in performance achieved is easy to compensate with a regular PI or PID controller by tuning it a little tighter. If one insist on using a Smith predictor, the SIMC tuning rules can be applied when the maximum time-delay error is known.

Recommendations for Further Work

The stability of the Smith predictor controller when modelling errors in the process lag-time constants should be analysed. For modelling errors in the gain, time-delay parameter and lag-time constants, a wide range of process models should be investigated.

Alternative structures for the primary controller should be investigated to find more optimal stability behaviour.

The step response in the time domain has not been evaluated beyond the integrated absolute error in the evaluation of performance. The time domain response should be evaluated further.

BIBLIOGRAPHY

E.J. Adam, H.A. Latchman, and O.D. Crisalle. Robustness of the smith predictor with respect to uncertainty in the time-delay parameter. *Procedings of the American Control Conference*, pages 1452–1457, 2000.

S. Bennet. Nicolas minorsky and the automatic steering of ships. *Control Systems Magazine, IEEE*, pages 10–15, 1984.

S. Bennet. *A History of Control Engineering: 1930–1955*. Peter Peregrinus Ltd. on behalf of the Institution of Electrical Engineers, London, United Kingdom, 1993.

Martin S. Foss. Validation of the SIMC tuning rules. Technical report, Norwegian University of Science and Technology, 2012. Available at `http://www.nt.ntnu.no/users/skoge/diplom/prosjekt12/foss/SIMC%20PID-validation.pdf`.

Graham C. Goodwin, Stefan F. Graebe, and Mario E. Salgado. *Control System Design*. Prentice Hall, Upper Saddle River, New Jersey, 2001.

Chriss Grimholt. Applying SIMC tuning rules to smith-predictor. Internal note, 2013.

Chriss Grimholt and Sigurd Skogestad. Optimal PI-control and verification of the SIMC tuning rules. In *IFAC Conference on Advances in PID Control*. IFAC, March 28.–30. 2012.

Chriss Grimholt and Sigurd Skogestad. Optimal PID-control on first order plus time delay systems & verification of the SIMC rules. Submitted to DYCOPS, 2013.

F. L. Lewis. *Applied Optimal Control and Estimation*. Prentice Hall, 1992.

Mathworks. `http://www.mathworks.se`, Cited: June 2, 2013.

O. Mayr. *The Origins of Feedback Control*. MIT Press, Cambridge, 1970.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, second edition, 2006.

J.E. Normey-Rico and E.F. Camacho. *Control of Dead-Time Processes*. Springer, London Limited, 2007.

Helene Paulsen. Stability and robustness of the smith predictor controller. Technical report, Norwegian University of Science and Technology, 2012. Available at `http://www.nt.ntnu.no/users/skoge/diplom/prosjekt12/paulsen/Prosjektrapport%20PDF%20final%20HP.pdf`.

Daniel E. Rivera, Manfred Morari, and Sigurd Skogestad. Internal model control. 4. PID controller design. *Ind. Eng. Chem. Process Des. Dev.*, 25: 252–265, 1986.

Dale E. Seborg, Thomas F. Edgar, and Duncan A. Mellichamp. *Process Dynamics and Control*. Wiley & Sons, second edition, 2004.

Mohammad Shamsuzzoha and Sigurd Skogestad. The setpoint overshoot method: A simple and fast closed-loop approach for PID tuning. *Journal of Process Control*, 20:1220–1234, 2010.

Sigurd Skogestad. Probably the best simple PID tuning rules in the world. AIChE Annual Meeting, Reno, USA. Available at `http://www.nt.ntnu.no/users/skoge/publications/2001/tuningpaper_reno/tuningpaper_06nov01.pdf`, 2001. Cited: June 2, 2013.

Sigurd Skogestad. Simple analytic tuning rules for model reduction and PID controller tuning. *Journal of Process Control*, 13:291–309, 2003.

Sigurd Skogestad and Chriss Grimholt. The SIMC method for smooth PID controller tuning. *Advances in Industrial Control*, pages 147–175, 2012.

Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control Analysis and Design*. Wiley & Sons, second edition, 2005.

J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Trans. ASME*, 64:759–768, 1942.

APPENDIX A

FIRST ORDER PLOT RESULTS

Tuning example values for the Pareto optimal controllers are given in Chapter A.1. SIMC tuning reference points are listed in Chapter A.2. The Pareto optimal PI and PID tuning curves for the FOPTD process models are graphically represented by $J = f(M_S)$, with the complementary sensitivity plotted as $M_T = f(M_S)$. The Pareto optimal tuning curves are given in Chapter A.3.

A.1 Pareto Optimal Tuning Example Values

Examples of tuning values for the Pareto optimal PI and PID cascade tuning is given in Table A.1 and A.2.

Table A.1 – Pareto optimal PI cascade tuning examples for FOPTD models.

Process	K_c	τ_I	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_{10}(s) = \frac{e^{-s}}{s}$	0.41	6.07	2.51	1.60	4.31	14.88
$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	0.55	1.10	1.37	1.60	2.07	2.01
$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	3.48	3.93	1.94	1.60	3.11	1.13
$G_{13}(s) = e^{-s}$	0.20	0.32	1.00	1.60	1.60	1.59
$G_{14}(s) = \frac{e^{-s}}{20s+1}$	8.43	5.00	2.22	1.60	3.69	0.60

Table A.2 – Pareto optimal PID cascade tuning examples for FOPTD models.

Process	K_c	τ_I	τ_D	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_{10}(s) = \frac{e^{-s}}{s}$	0.55	3.25	0.48	1.46	1.60	2.99	6.71
$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	0.43	0.61	0.61	1.01	1.60	1.55	1.46
$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	4.39	2.53	0.48	1.26	1.60	2.34	0.62
$G_{13}(s) = e^{-s}$	0.20	0.31	0.01	0.99	1.60	1.58	1.58
$G_{14}(s) = \frac{e^{-s}}{20s+1}$	10.98	2.94	0.48	1.36	1.60	2.68	0.30

Table A.3 – SIMC tuning parameters for the τ_c reference points for models 10 to 14. The closed-loop time constant values are $\tau_c/\theta \in [½\ 1\ ⅜\ 2]$.

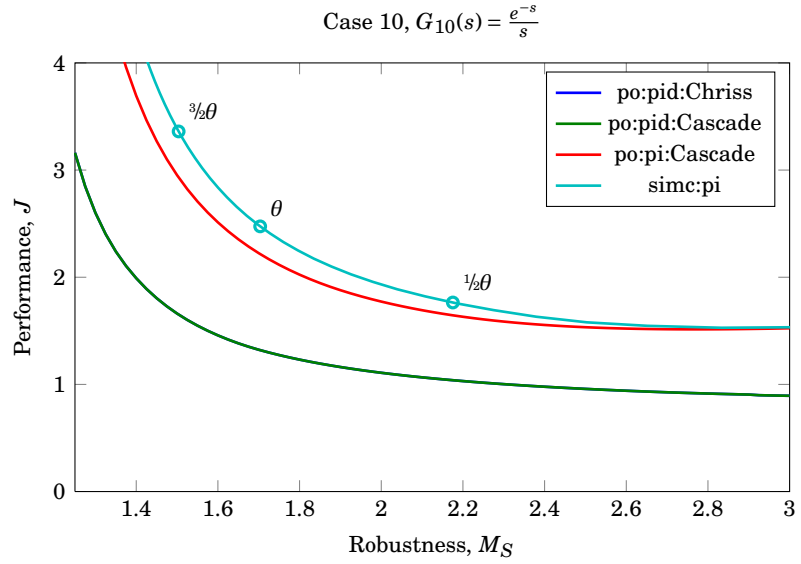
Process	τ_c/θ	τ_c	K_c	τ_I	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_{10}(s) = \frac{e^{-s}}{s}$	½	0.50	0.67	6.00	1.76	2.18	3.40	9.00
	1	1.00	0.50	8.00	2.47	1.70	3.92	15.99
	⅜	1.50	0.40	10.00	3.36	1.50	4.51	24.97
	1	2.00	0.33	12.00	4.41	1.39	5.14	35.87
$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	½	0.50	0.67	1.00	1.30	1.92	2.13	1.73
	1	1.00	0.50	1.00	1.41	1.59	2.17	2.03
	⅜	1.50	0.40	1.00	1.68	1.44	2.50	2.50
	2	2.00	0.33	1.00	2.02	1.35	3.00	3.00
$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	½	0.50	5.33	6.00	1.70	1.97	2.37	1.12
	1	1.00	4.00	8.00	2.38	1.59	2.17	2.00
	⅜	1.50	3.20	8.00	2.91	1.44	2.50	2.49
	2	2.00	2.67	8.00	3.49	1.35	3.00	2.99
$G_{13}(s) = e^{-s}$	½	0.50	0.00	0.01	1.34	1.92	2.13	2.13
	1	1.00	0.00	0.01	1.36	1.59	2.17	2.17
	⅜	1.50	0.00	0.01	1.57	1.44	2.50	2.50
	2	2.00	0.00	0.01	1.88	1.35	3.00	3.00
$G_{14}(s) = \frac{e^{-s}}{20s+1}$	½	0.50	13.33	6.00	1.74	2.08	2.98	0.45
	1	1.00	10.00	8.00	2.43	1.65	3.20	0.80
	⅜	1.50	8.00	10.00	3.30	1.47	3.43	1.25
	2	2.00	6.67	12.00	4.33	1.36	3.68	1.78

A.2 SIMC Tuning Reference Points

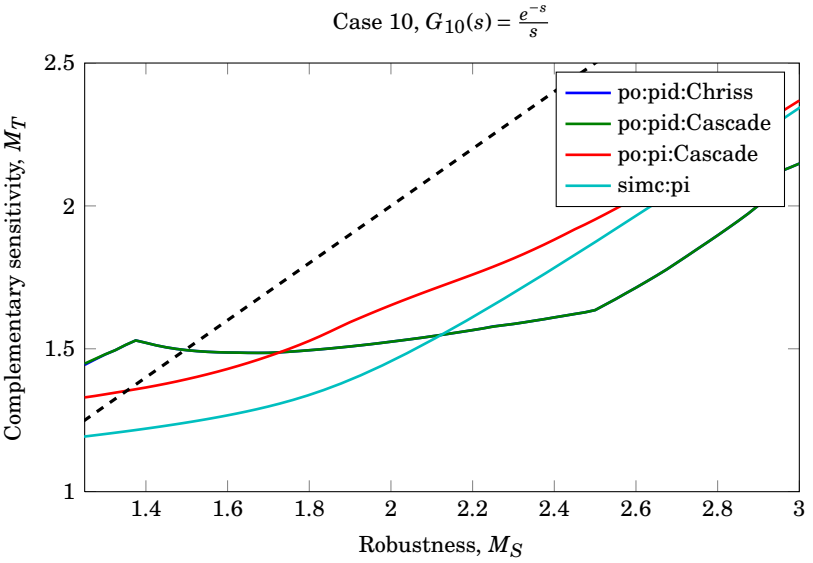
The tuning reference points for the SIMC tuning rules are given for SIMC PI controllers in Table A.3. The closed-loop time constant values for the reference points are given by $\tau_c/\theta \in [½\ 1\ ⅜\ 2]$.

A.3 Pareto Optimal PI Performance and SIMC Tuning

The Pareto optimal PI controller tuning are represented in Figure A.1(a) through A.5(a), where the tuning performance (J) is plotted as a function of robustness (M_S). The SIMC tuning curve is included, with the τ_c reference points marked as circles. Figure A.1(b) through A.5(b) illustrates the correpsonding M_S-M_T relation.

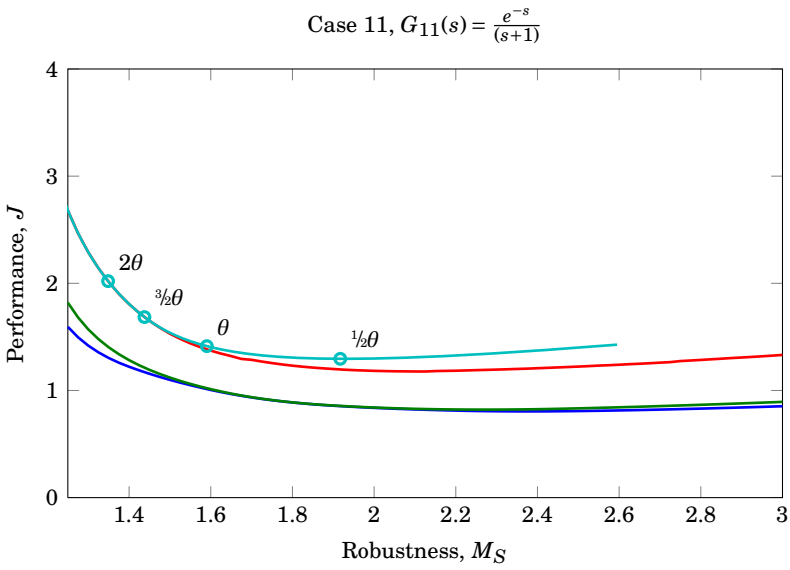


(a) Cost value as a function of robustness, $J(M_S)$.

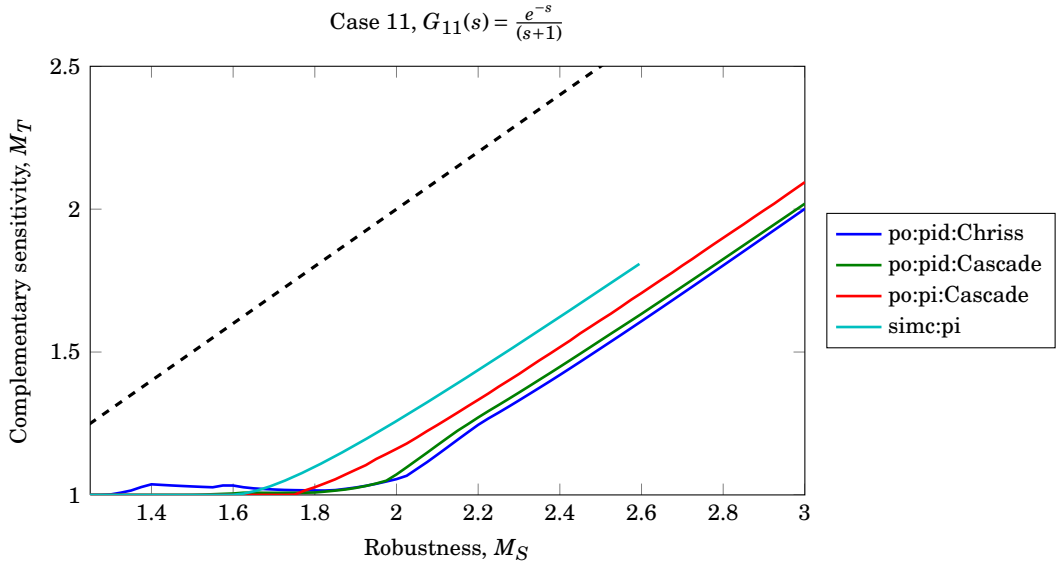


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure A.1 – SIMC controller tuning compared to Pareto optimal PI and PID controller tuning curve for Case 10, integrating process: $G_{10}(s) = \frac{e^{-s}}{s}$.

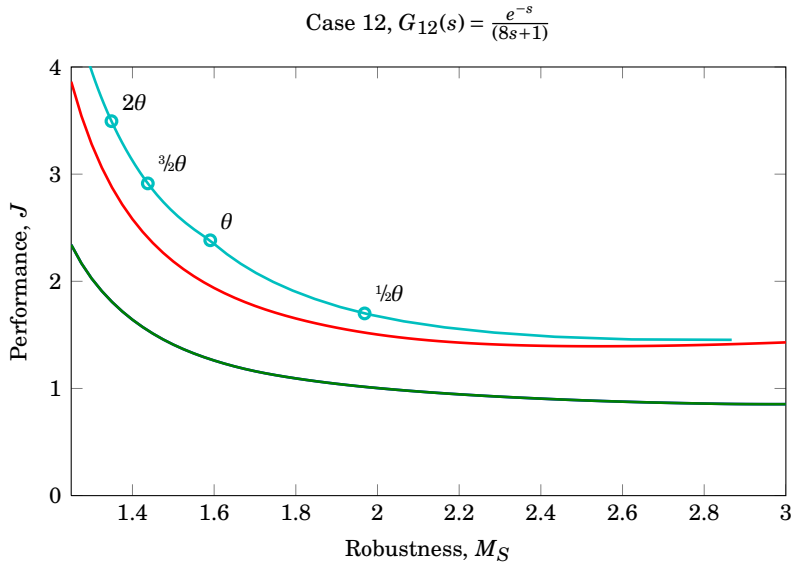


(a) Cost value as a function of robustness, $J(M_S)$.

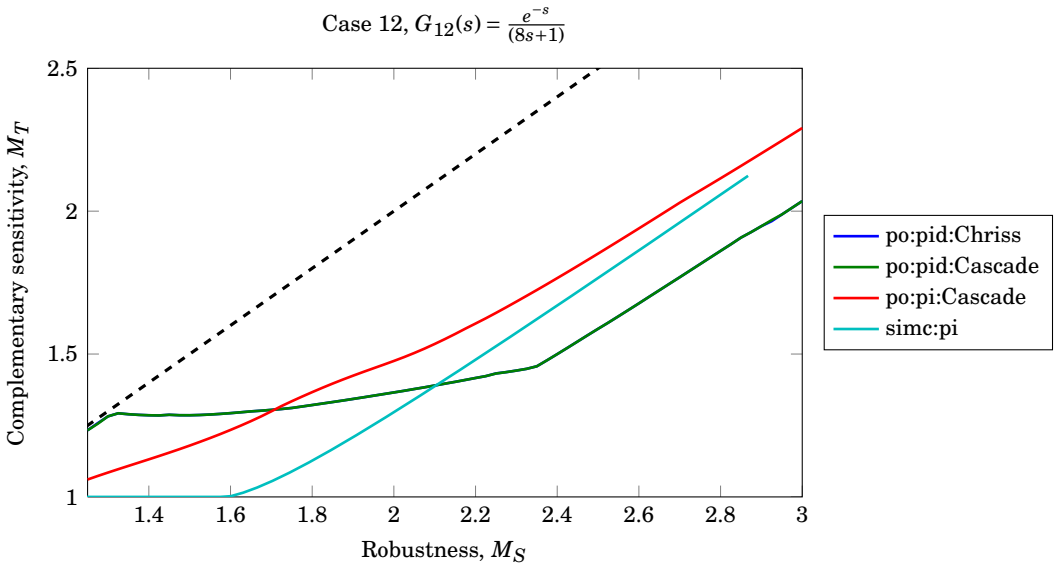


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure A.2 – SIMC controller tuning compared to Pareto optimal PI and PID controller tuning curve for Case 11, time delay dominated process: $G_{11}(s) = \frac{e^{-s}}{(s+1)}$.

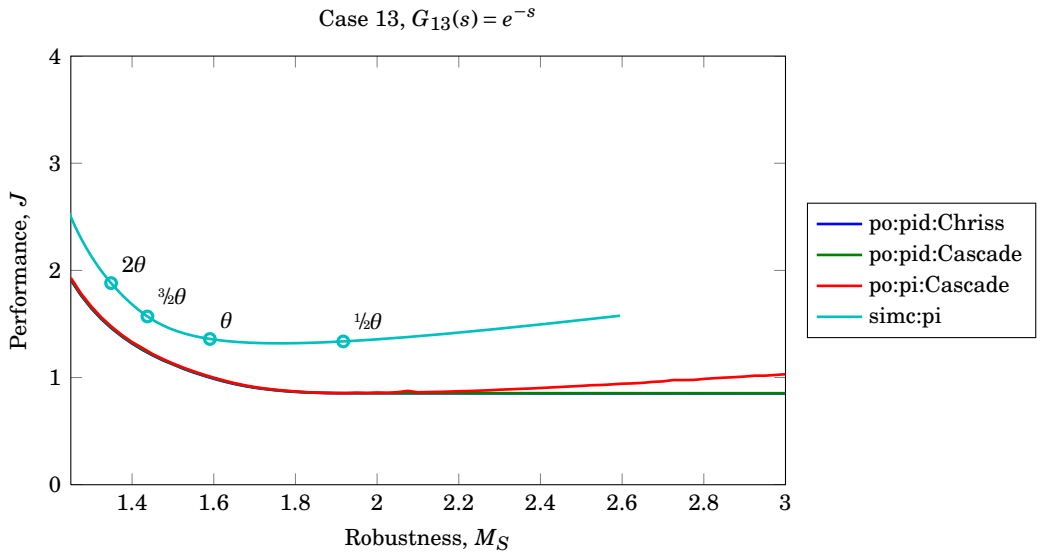


(a) Cost value as a function of robustness, $J(M_S)$.

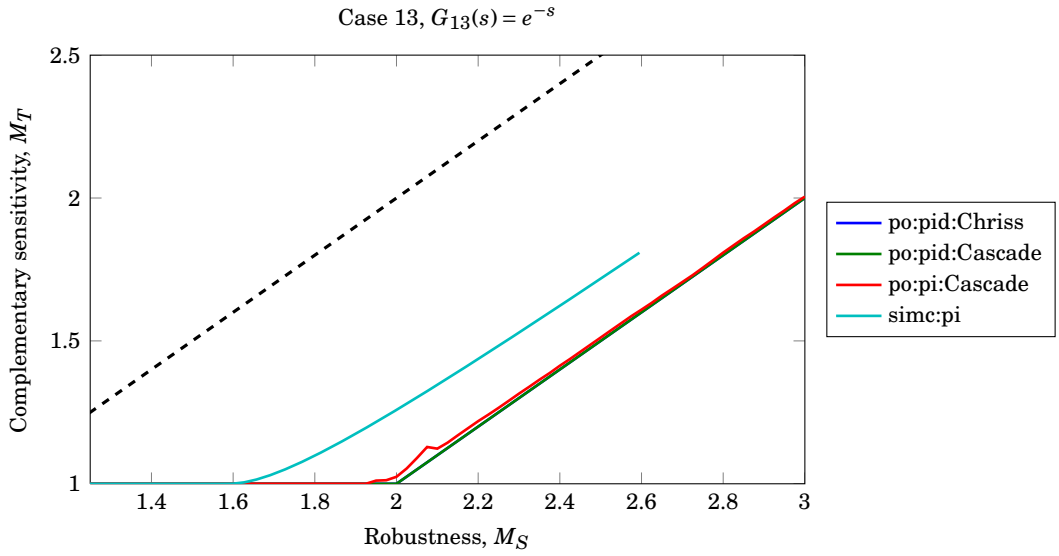


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure A.3 – SIMC controller tuning compared to Pareto optimal PI and PID controller tuning curve for Case 12, lag dominated process: $G_{12}(s) = \frac{e^{-s}}{(8s+1)}$.

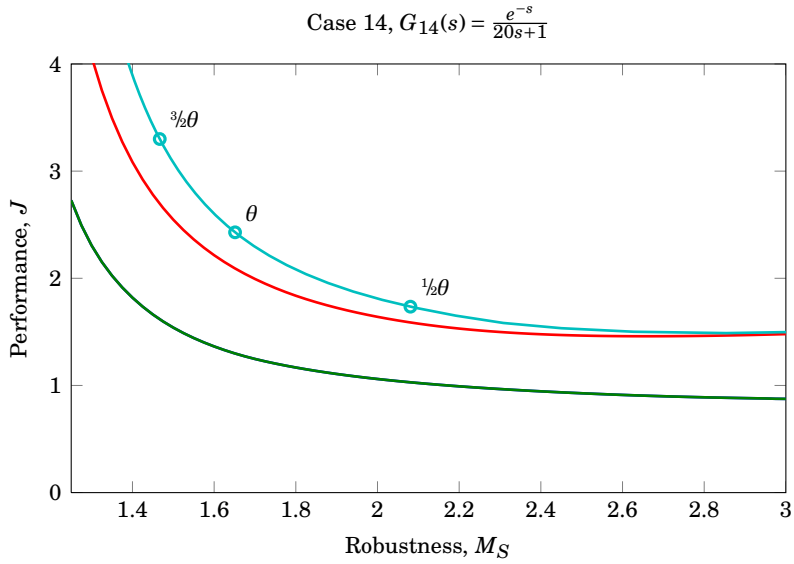


(a) Cost value as a function of robustness, $J(M_S)$.

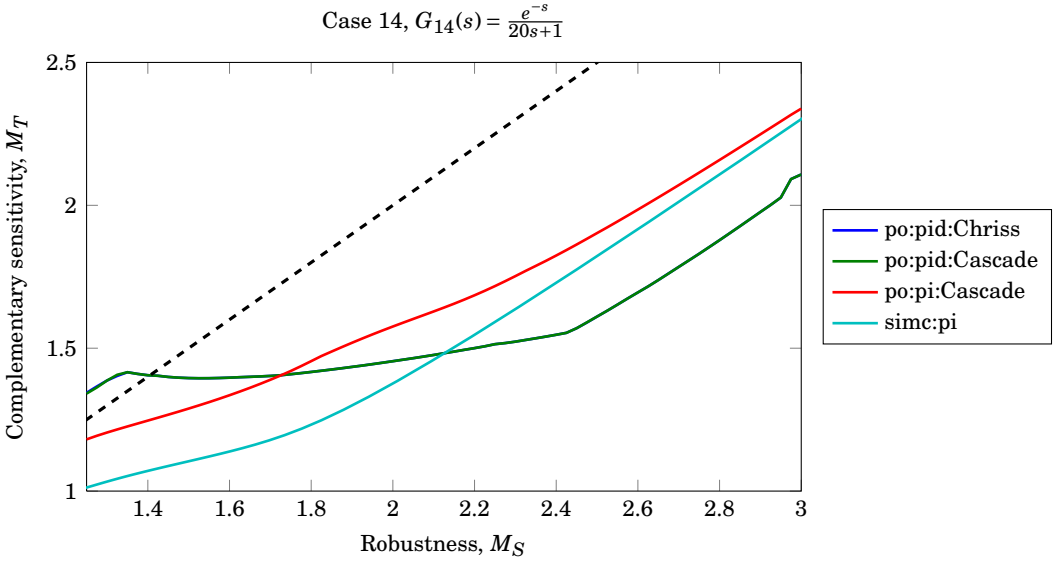


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure A.4 – SIMC controller tuning compared to Pareto optimal PI and PID controller tuning curve for Case 13, pure time delay process: $G_{13}(s) = e^{-s}$.



(a) Cost value as a function of robustness, $J(M_S)$.



(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure A.5 – SIMC controller tuning compared to Pareto optimal PI and PID controller tuning curve for Case 14, pure time delay process: $G_{14}(s) = \frac{e^{-s}}{20s+1}$.

APPENDIX B

SECOND ORDER PLOT RESULTS

Tuning example values for the Pareto optimal controllers are given in Chapter B.1. SIMC tuning reference points are listed in Chapter B.2. The Pareto optimal PI and PID tuning curves for the FOPTD process models are graphically represented by $J = f(M_S)$, with the complementary sensitivity plotted as $M_T = f(M_S)$. The Pareto optimal tuning curves are given in Chapter B.3.

B.1 Pareto Optimal Tuning Example Values

Example tuning values are given for the Pareto optimal PI and PID controllers in Table B.1 and B.2, respectively. The examples are given for the cascade parameterisation of the PI and PID controllers.

B.2 SIMC Tuning Reference Points

The tuning reference points for the SIMC tuning rules are given for SIMC PI controllers in Table B.3, while the reference points for the SIMC PID controllers are given in Table B.4. The closed-loop time constant values for the reference points are given by $\tau_c/\theta \in [½\ 1\ ⅔\ 2]$.

Table B.1 – Pareto optimal PI cascade tuning examples for SOPTD models.

Process	K_c	τ_I	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	0.49	1.35	1.49	1.6	2.8	2.75
$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	0.53	1.57	1.58	1.6	3.07	2.99
$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	0.48	1.2	1.42	1.6	2.56	2.49
$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	1	1.29	2.27	1.6	1.48	1.29
$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	0.79	1.56	2.02	1.6	2.14	1.97
$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	1.4	1.07	2.61	1.6	0.98	0.76
$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	1.19	1.29	2.73	1.6	1.3	1.08
$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	0.93	1.54	2.36	1.6	1.87	1.64
$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	1.65	1.05	3.22	1.6	0.87	0.64

Table B.2 – Pareto optimal PID cascade tuning examples for SOPTD models.

Process	K_c	τ_I	τ_D	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	0.42	0.82	0.82	1.04	1.6	1.94	1.93
$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	0.48	0.96	0.96	1.05	1.6	2.04	2.01
$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	0.4	0.73	0.73	1.03	1.6	1.84	1.81
$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	1.14	0.69	0.69	1.11	1.6	0.76	0.6
$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	0.85	0.9	0.9	1.08	1.6	1.14	1.06
$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	1.65	0.47	0.47	1.17	1.6	0.52	0.29
$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	1.54	0.63	0.63	1.16	1.6	0.63	0.41
$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	1.14	0.88	0.88	1.11	1.6	0.88	0.77
$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	2.27	0.42	0.42	1.23	1.6	0.43	0.19

Table B.3 – SIMC PI tuning for SOPTD models. The closed-loop time constant values are $\tau_c/\theta \in [½\ 1\ ⅜\ 2]$.

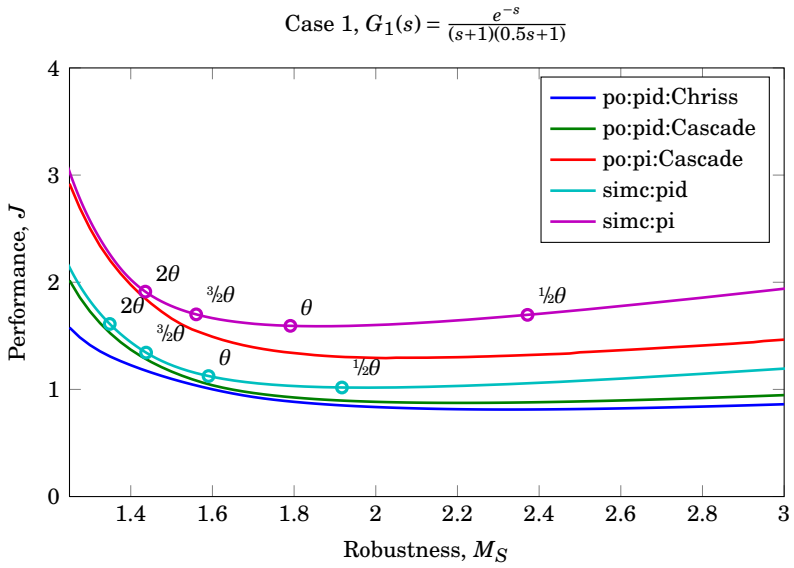
Process	τ_c/θ	τ_c	K_c	τ_I	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	½	0.62	0.62	1	1.7	2.37	3.53	2.81
	1	1.25	0.44	1	1.59	1.79	3.14	2.8
	⅜	1.88	0.35	1	1.7	1.56	3.24	3.1
	2	2.5	0.29	1	1.91	1.44	3.58	3.55
$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	½	0.7	0.59	1	2	2.5	4.32	3.37
	1	1.4	0.42	1	1.83	1.84	3.74	3.3
	⅜	2.1	0.32	1	1.91	1.59	3.78	3.57
	2	2.8	0.26	1	2.1	1.46	4.07	3.99
$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	½	0.57	0.63	1	1.5	2.23	2.97	2.38
	1	1.15	0.47	1	1.46	1.73	2.73	2.46
	⅜	1.72	0.37	1	1.6	1.53	2.89	2.81
	2	2.3	0.3	1	1.86	1.41	3.31	3.3
$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	½	0.29	1.6	1	2.34	2.64	1.98	0.99
	1	0.58	1.09	1	2.25	1.89	1.69	1.11
	⅜	0.87	0.83	1	2.45	1.62	1.68	1.32
	2	1.17	0.67	1	2.74	1.48	1.77	1.56
$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	½	0.47	1	1	2.48	2.64	3.16	1.97
	1	0.93	0.68	1	2.32	1.89	2.7	2.07
	⅜	1.4	0.52	1	2.45	1.62	2.69	2.33
	2	1.87	0.42	1	2.68	1.48	2.84	2.64
$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	½	0.17	2.67	1	2.24	2.64	1.19	0.44
	1	0.35	1.82	1	2.25	1.89	1.01	0.55
	⅜	0.52	1.38	1	2.57	1.62	1.01	0.72
	2	0.7	1.11	1	2.96	1.48	1.06	0.9
$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	½	0.25	2	1	2.68	2.67	1.74	0.75
	1	0.5	1.33	1	2.64	1.9	1.49	0.89
	⅜	0.75	1	1	2.91	1.63	1.48	1.09
	2	1	0.8	1	3.28	1.49	1.56	1.3
$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	½	0.4	1.25	1	2.82	2.67	2.78	1.54
	1	0.8	0.83	1	2.7	1.9	2.38	1.69
	⅜	1.2	0.62	1	2.89	1.63	2.37	1.94
	2	1.6	0.5	1	3.18	1.49	2.49	2.23
$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	½	0.15	3.33	1	2.6	2.67	1.04	0.33
	1	0.3	2.22	1	2.71	1.9	0.89	0.45
	⅜	0.45	1.67	1	3.14	1.63	0.89	0.6
	2	0.6	1.33	1	3.65	1.49	0.93	0.75

Table B.4 – SIMC PID tuning for SOPTD models. The closed-loop time constant values are $\tau_c/\theta \in [½\ 1\ ⅔\ 2]$.

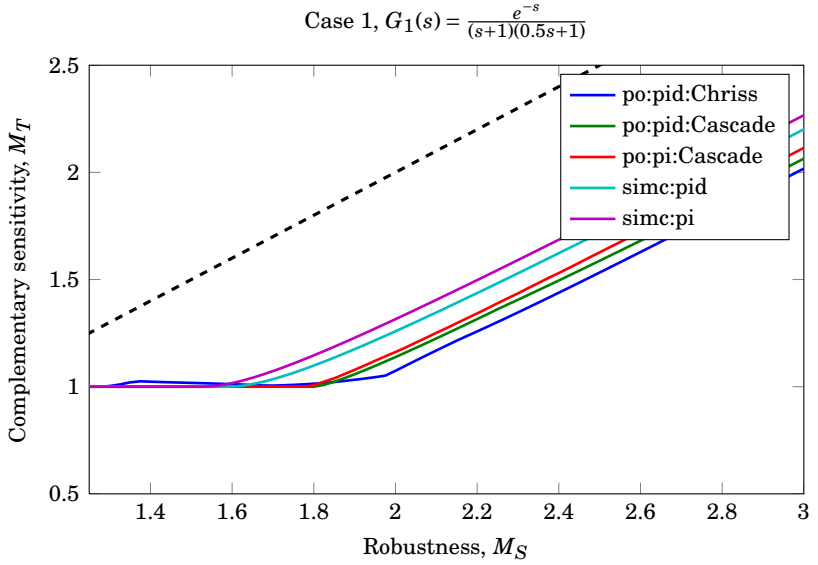
Process	τ_c/θ	τ_c	K_c	τ_I	τ_D	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	½	0.5	0.67	1	0.5	1.02	1.92	2.13	1.67
	1	1	0.5	1	0.5	1.12	1.59	2.17	2.02
	⅔	1.5	0.4	1	0.5	1.34	1.44	2.5	2.5
	2	2	0.33	1	0.5	1.61	1.35	3	3
$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	½	0.5	0.67	1	0.8	0.97	1.92	2.13	1.59
	1	1	0.5	1	0.8	1.09	1.59	2.17	2.01
	⅔	1.5	0.4	1	0.8	1.3	1.44	2.5	2.5
	2	2	0.33	1	0.8	1.56	1.35	3	3
$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	½	0.5	0.67	1	0.3	1.08	1.92	2.13	1.71
	1	1	0.5	1	0.3	1.18	1.59	2.17	2.03
	⅔	1.5	0.4	1	0.3	1.41	1.44	2.5	2.5
	2	2	0.33	1	0.3	1.69	1.35	3	3
$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	½	0.17	2	1	0.5	0.98	1.92	0.71	0.5
	1	0.33	1.5	1	0.5	1.14	1.59	0.72	0.67
	⅔	0.5	1.2	1	0.5	1.38	1.44	0.84	0.83
	2	0.67	1	1	0.5	1.65	1.35	1	1
$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	½	0.27	1.25	1	0.8	0.94	1.92	1.14	0.8
	1	0.53	0.94	1	0.8	1.09	1.59	1.16	1.07
	⅔	0.8	0.75	1	0.8	1.31	1.44	1.34	1.33
	2	1.07	0.62	1	0.8	1.57	1.35	1.6	1.6
$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	½	0.1	3.33	1	0.3	1.07	1.92	0.43	0.3
	1	0.2	2.5	1	0.3	1.27	1.59	0.43	0.4
	⅔	0.3	2	1	0.3	1.54	1.44	0.5	0.5
	2	0.4	1.67	1	0.3	1.84	1.35	0.6	0.6
$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	½	0.12	2.67	1	0.5	1.03	1.92	0.53	0.38
	1	0.25	2	1	0.5	1.21	1.59	0.54	0.5
	⅔	0.38	1.6	1	0.5	1.46	1.44	0.63	0.63
	2	0.5	1.33	1	0.5	1.75	1.35	0.75	0.75
$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	½	0.2	1.67	1	0.8	0.96	1.92	0.85	0.6
	1	0.4	1.25	1	0.8	1.13	1.59	0.87	0.8
	⅔	0.6	1	1	0.8	1.36	1.44	1	1
	2	0.8	0.83	1	0.8	1.63	1.35	1.2	1.2
$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	½	0.07	4.44	0.9	0.3	1.12	1.94	0.34	0.2
	1	0.15	3.33	1	0.3	1.39	1.59	0.33	0.3
	⅔	0.22	2.67	1	0.3	1.68	1.44	0.38	0.37
	2	0.3	2.22	1	0.3	2.02	1.35	0.45	0.45

B.3 Pareto Optimal PI and PID Tuning Curves and SIMC Tuning

The Pareto optimal PI and PID controllers have been found as described in Chapter 3.7, while the SIMC tuning curves have been found as given in Chapter 4.3. The cases studied are the second-order-plus-time-delay models in case 1 to case 9. Figures B.1(a) through B.9(a) depict the performance (J) as a function of robustness (M_S) while Figures B.1(b) through B.9(b). The SIMC reference values are denoted as circles for $\tau_c/\theta \in [½\ 1\ ½\ 2]$. Figures B.1(b) through B.9(b) depict the M_S-M_T relation.

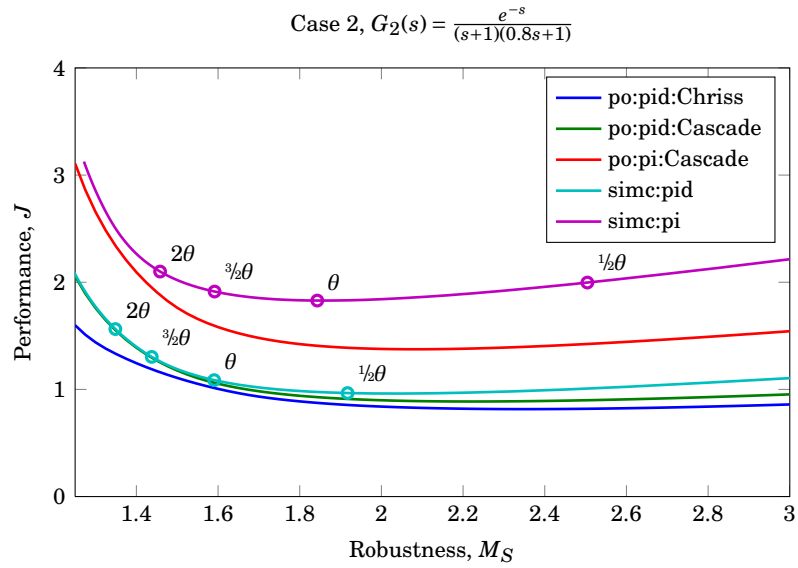


(a) Cost value as a function of robustness, $J = f(M_S)$.

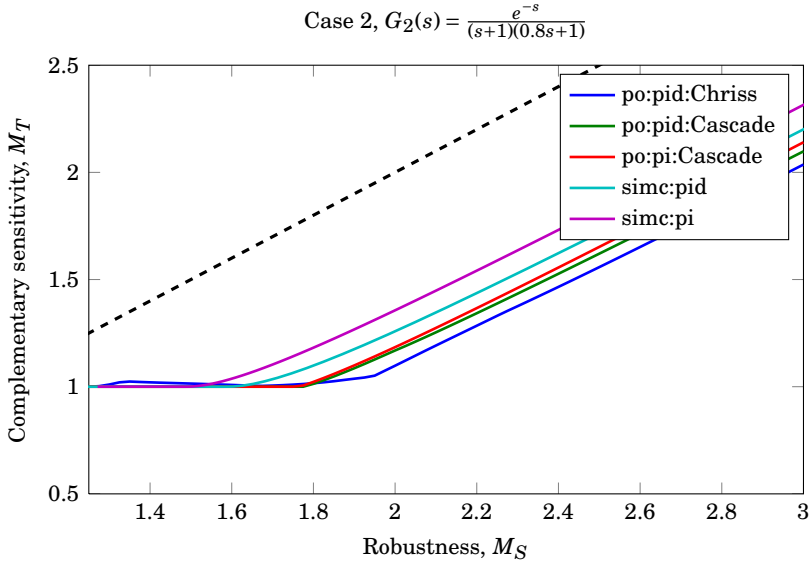


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure B.1 – SIMC tuning controllers compared to Pareto optimal PI and PID controllers for $G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$.

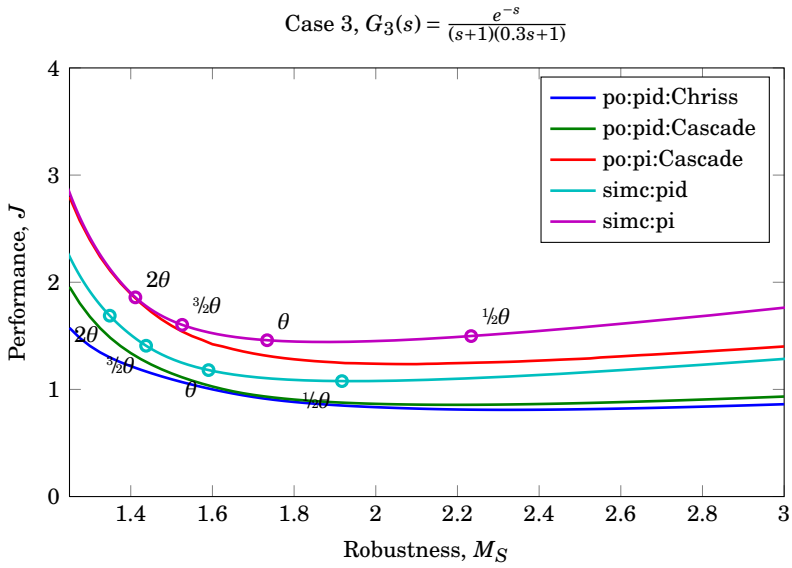


(a) Cost value as a function of robustness, $J = f(M_S)$.

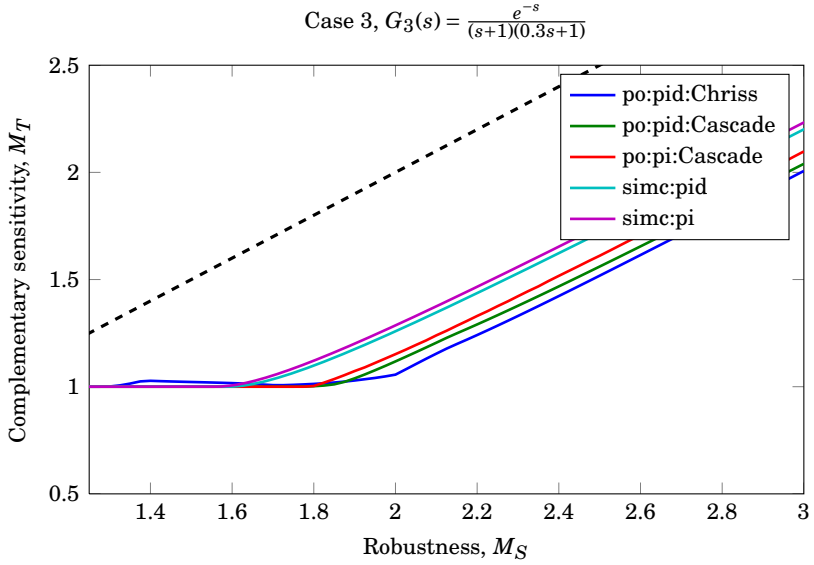


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure B.2 – SIMC tuning controllers compared to Pareto optimal PI and PID controllers for $G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$.

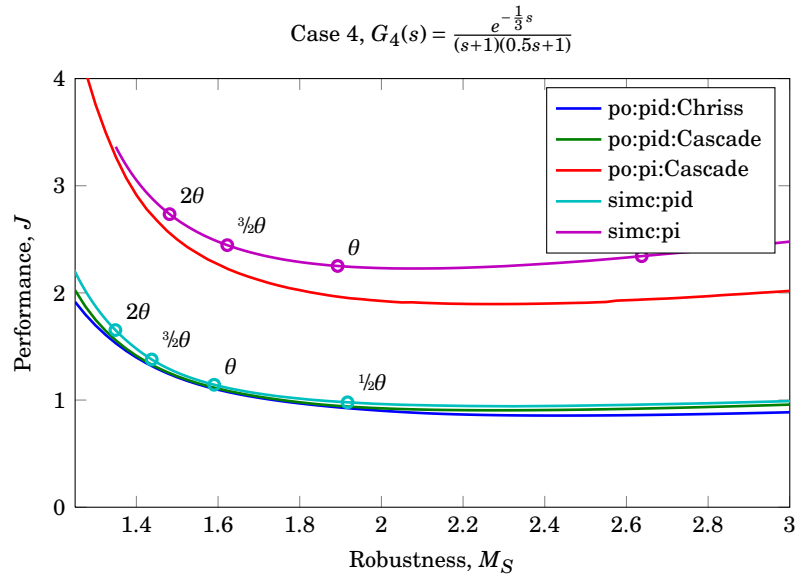


(a) Cost value as a function of robustness, $J = f(M_S)$.

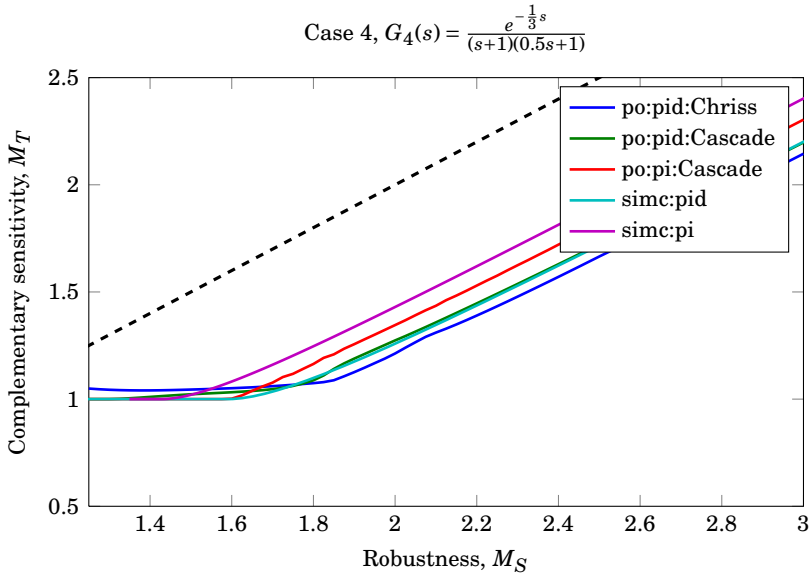


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure B.3 – SIMC tuning controllers compared to Pareto optimal PI and PID controllers for $G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$.

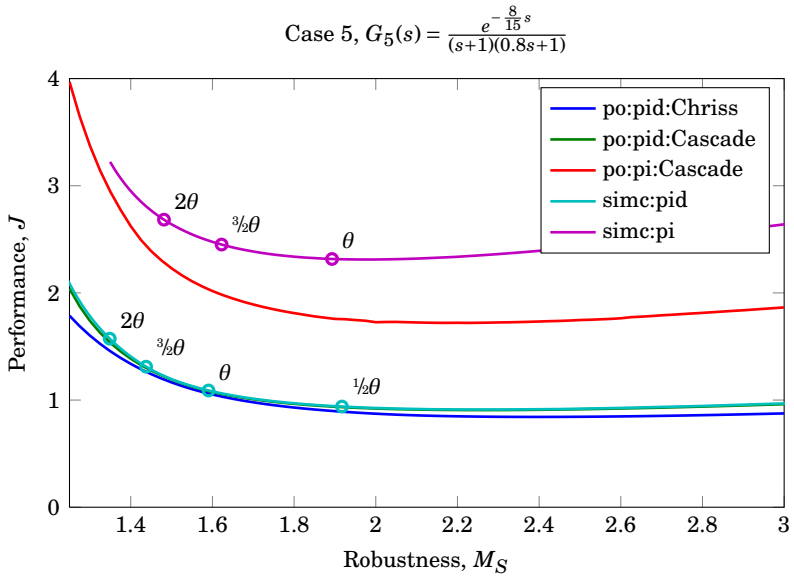


(a) Cost value as a function of robustness, $J = f(M_S)$.

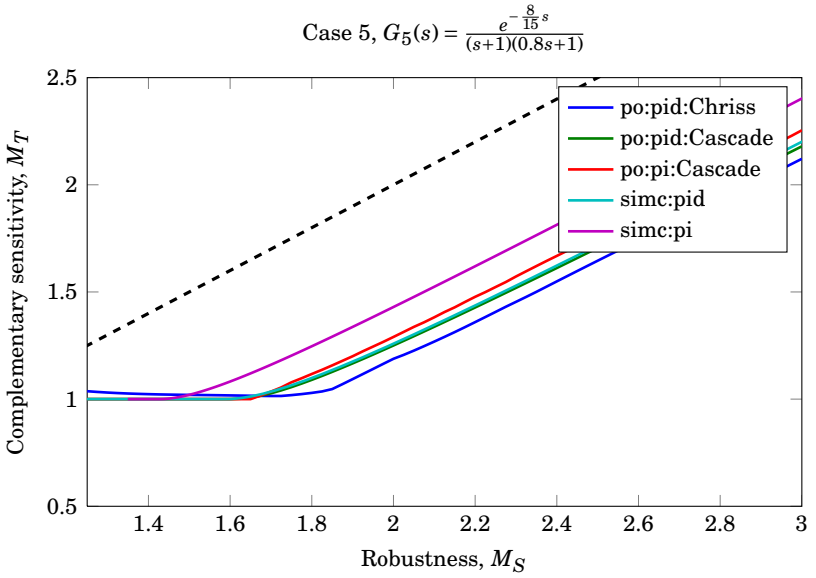


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure B.4 – SIMC tuning controllers compared to Pareto optimal PI and PID controllers for $G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$.

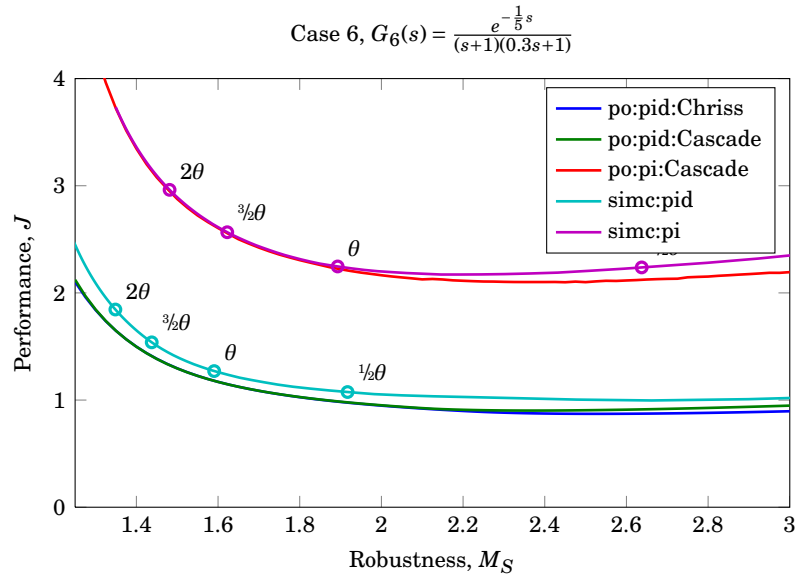


(a) Cost value as a function of robustness, $J = f(M_S)$.

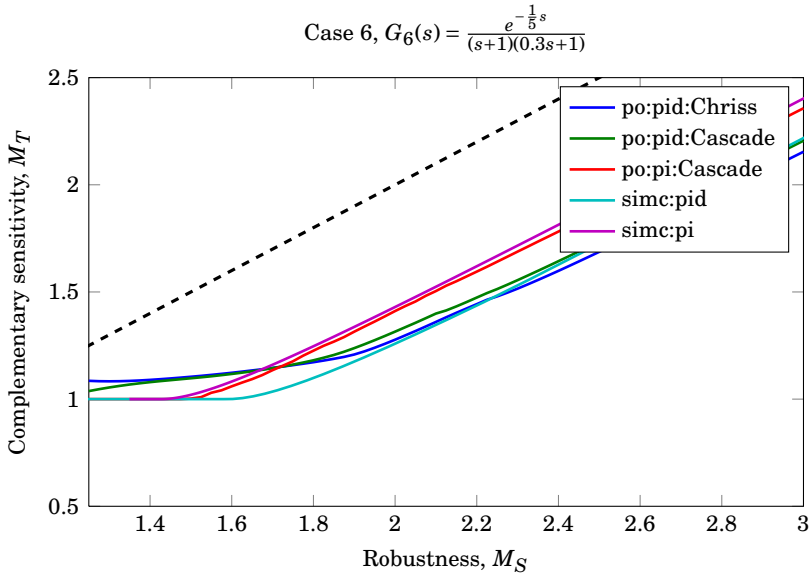


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure B.5 – SIMC tuning controllers compared to Pareto optimal PI and PID controllers for $G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$.

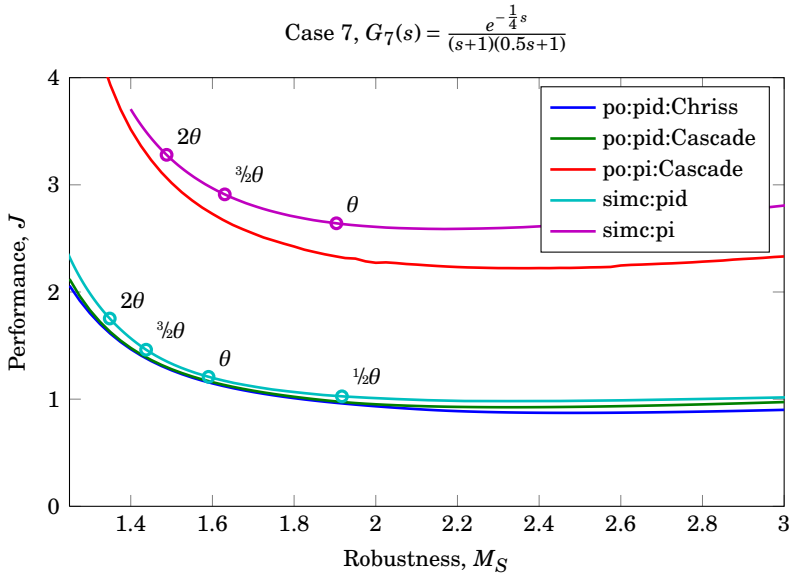


(a) Cost value as a function of robustness, $J = f(M_S)$.

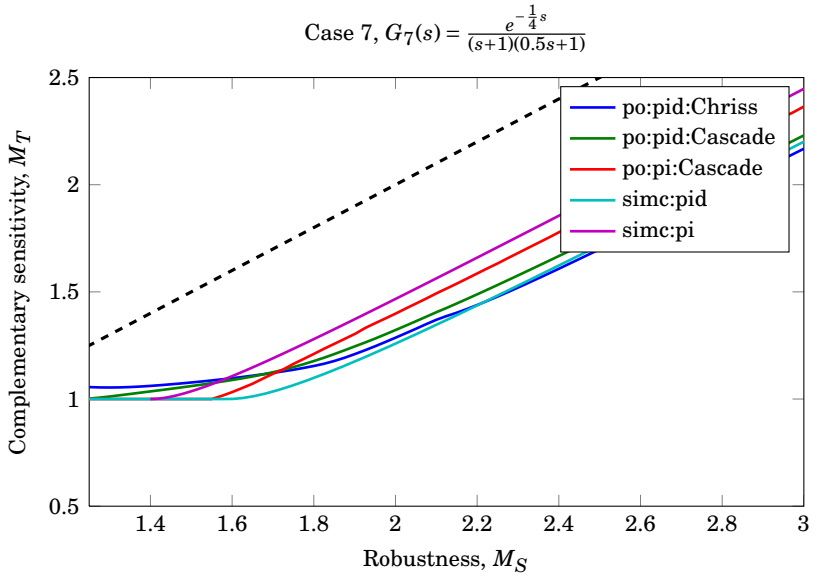


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure B.6 – SIMC tuning controllers compared to Pareto optimal PI and PID controllers for $G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$.

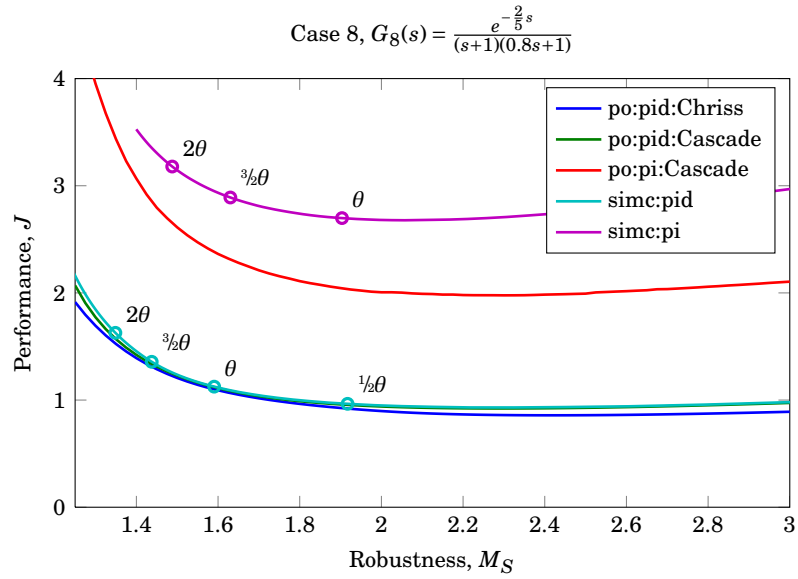


(a) Cost value as a function of robustness, $J = f(M_S)$.

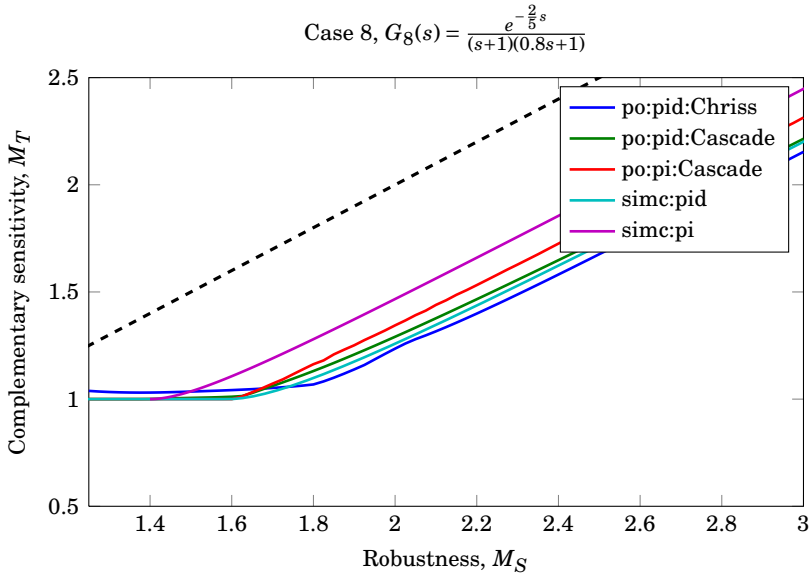


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure B.7 – SIMC tuning controllers compared to Pareto optimal PI and PID controllers for $G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$.

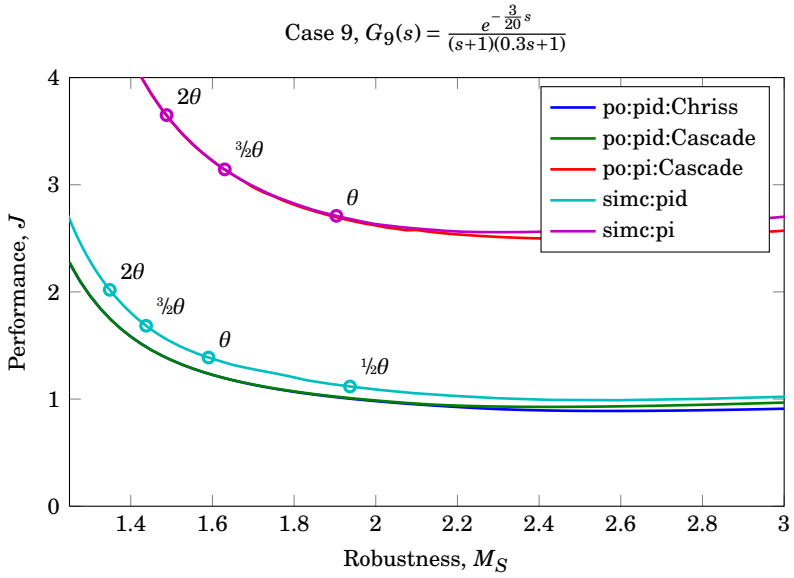


(a) Cost value as a function of robustness, $J = f(M_S)$.

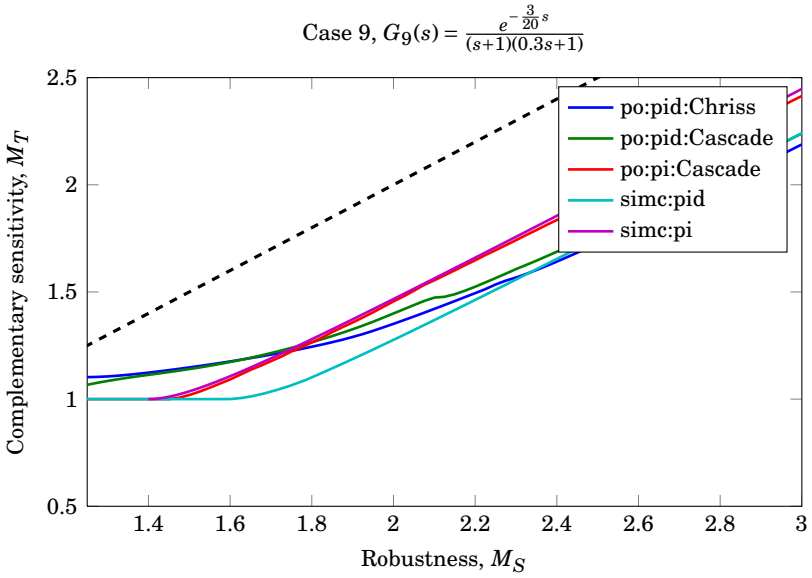


(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure B.8 – SIMC tuning controllers compared to Pareto optimal PI and PID controllers for $G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$.



(a) Cost value as a function of robustness, $J = f(M_S)$.



(b) Complementary sensitivity peak value as a function of the sensitivity peak value, $M_T = f(M_S)$. $M_T = M_S$ is represented by the dashed line.

Figure B.9 – SIMC tuning comcontrollers compared to Pareto optimal PI and PID controllers for $G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$.

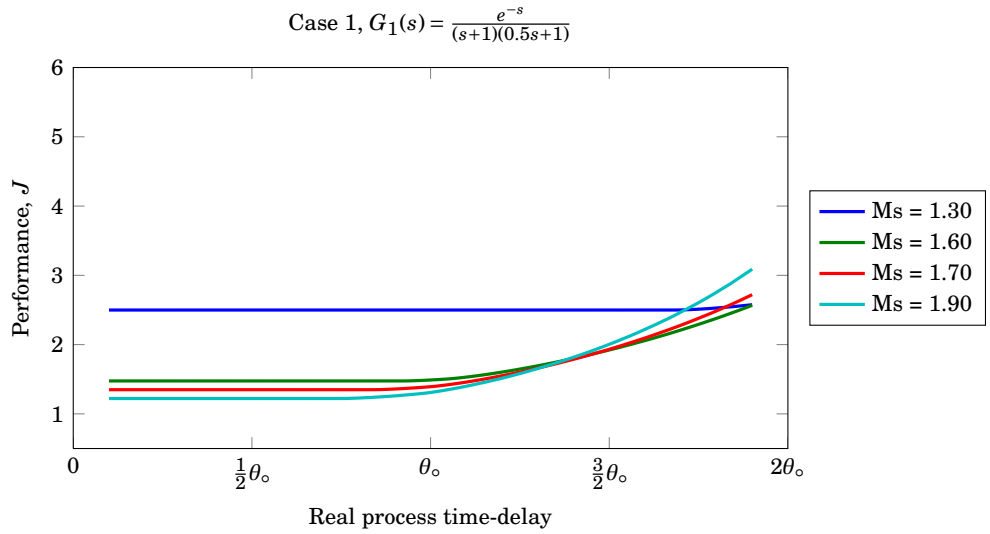
APPENDIX C

PARETO OPTIMAL SENSITIVITY TO TIME-DELAY MODELLING ERROR

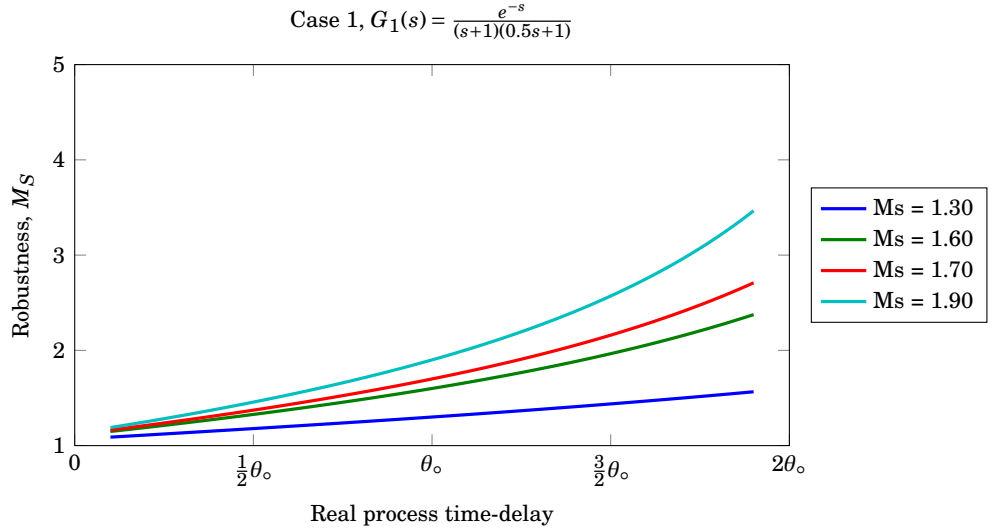
The Pareto optimal PI and PID controllers for the process models in cases 1–14 have been evaluated when the real process time-delay (θ) is different from the nominal modelled time-delay ($\theta_\circ$). The real process time-delay have been changed within $\pm 90\,\%$ of the nominal time-delay. Performance and robustness efficiency have been plotted as functions of the real time-delay.

C.1 PI Controller Sensitivity

The sensitivity of the Pareto optimal PI controller for in terms of performance are given in Figure C.1(a) through C.14(a). The robustness efficiency is given in Figure C.1(b) through C.14(b). Cases 1–14 have been investigated.

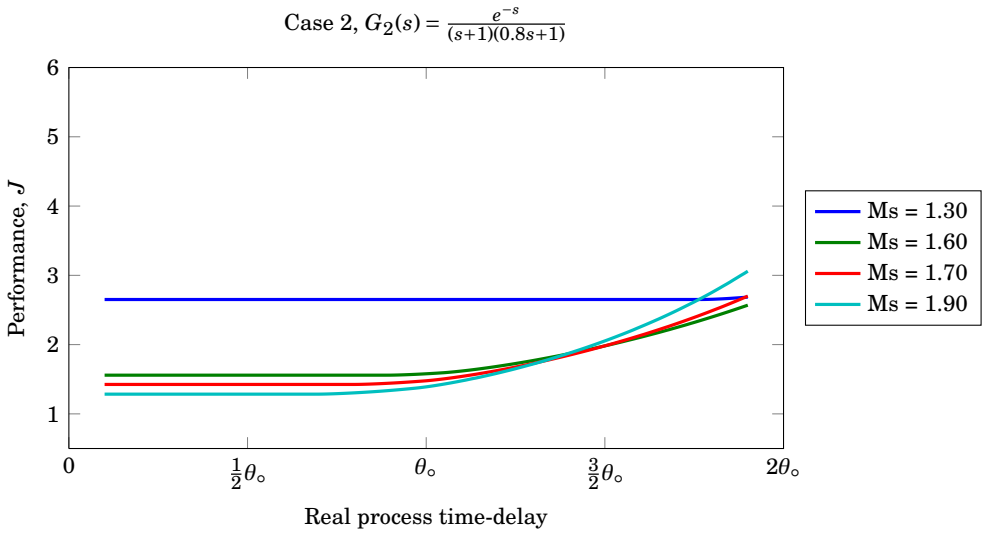


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

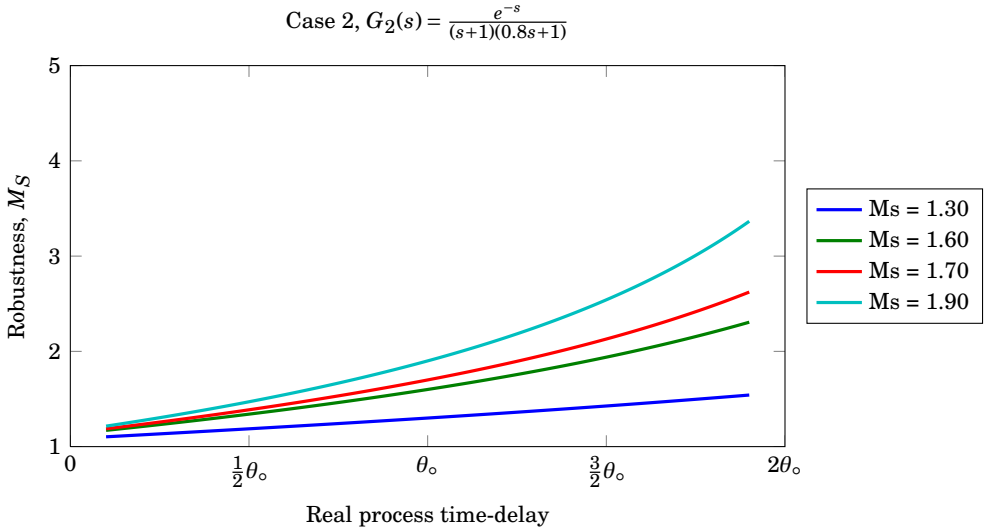


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.1 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

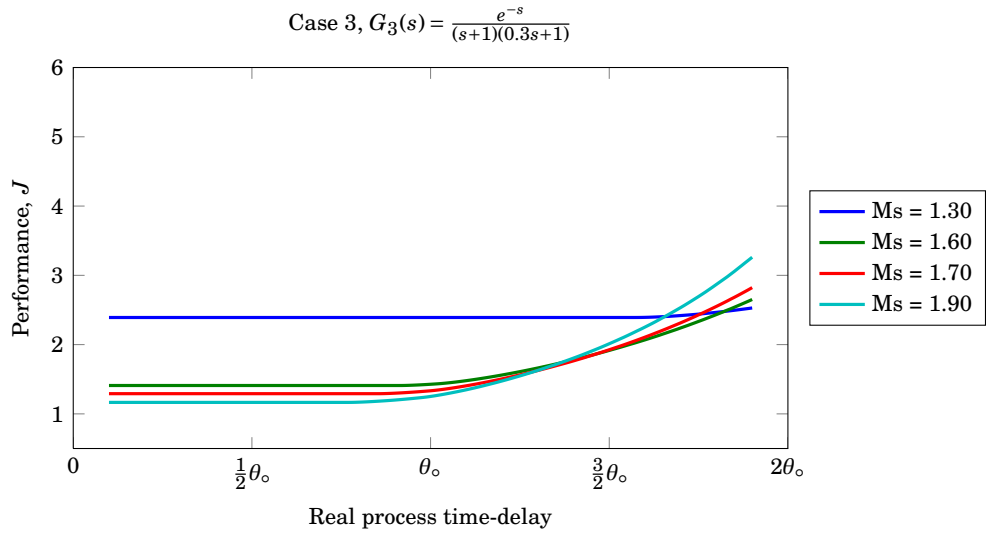


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

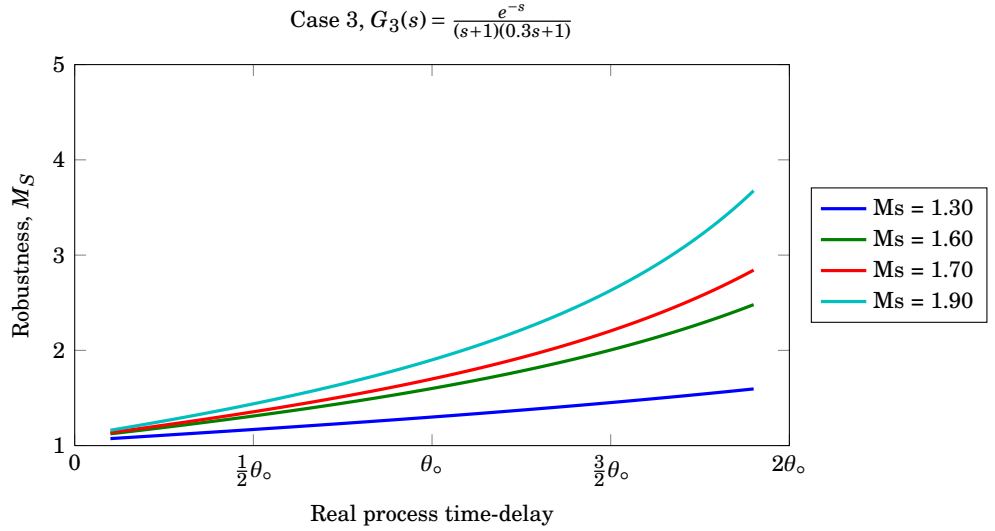


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.2 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

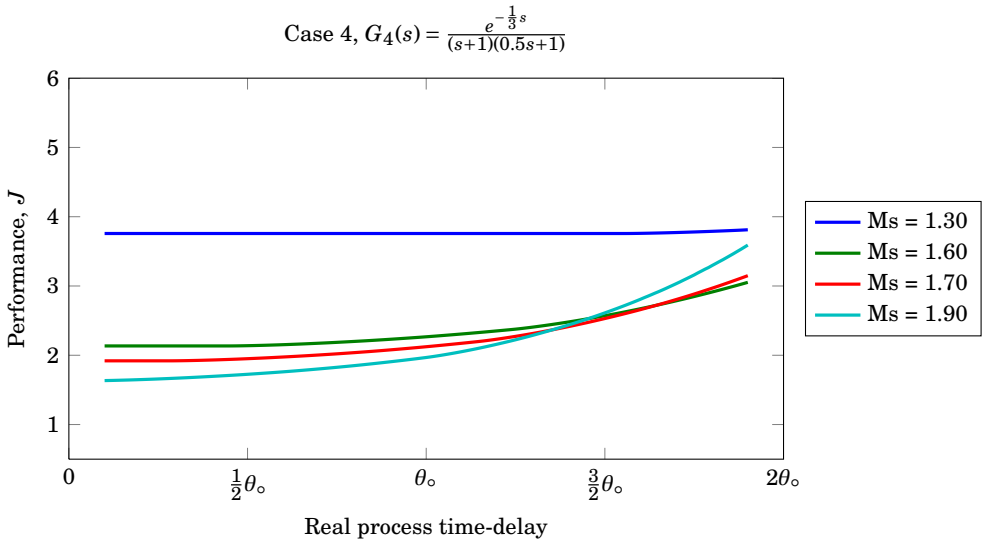


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

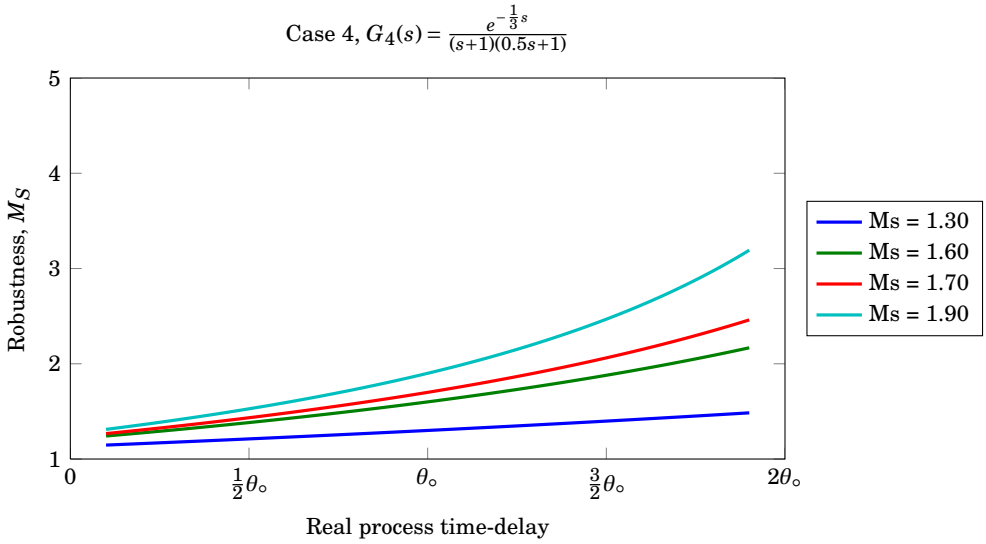


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.3 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

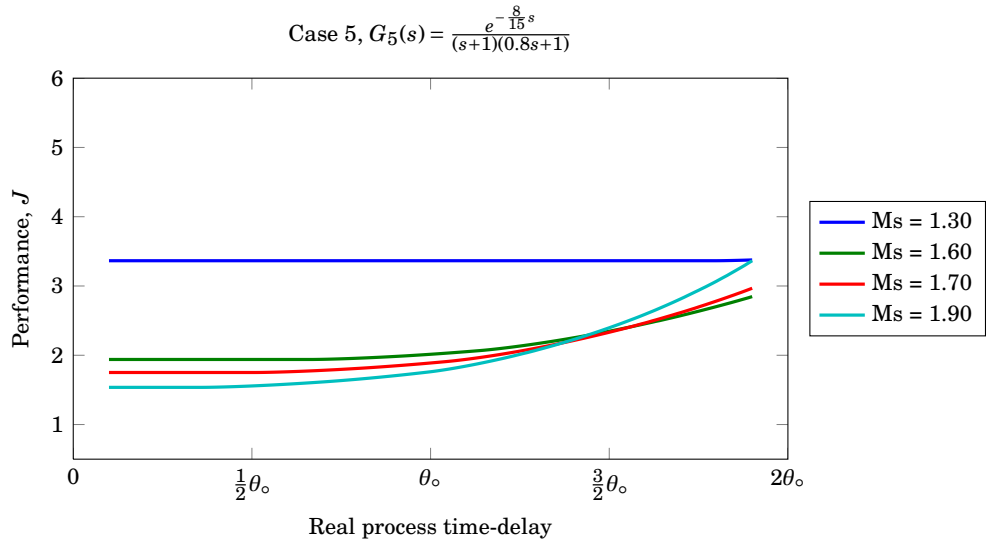


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

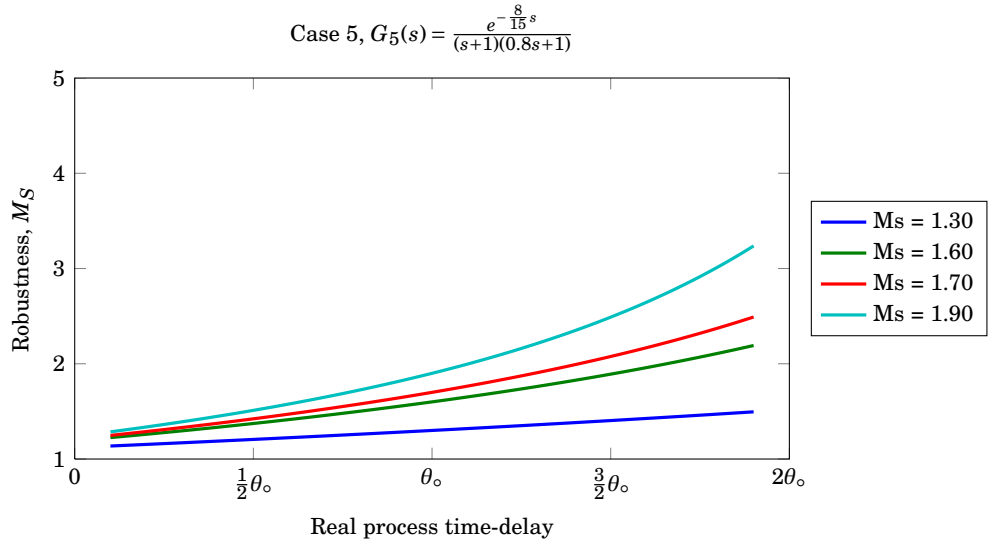


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.4 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

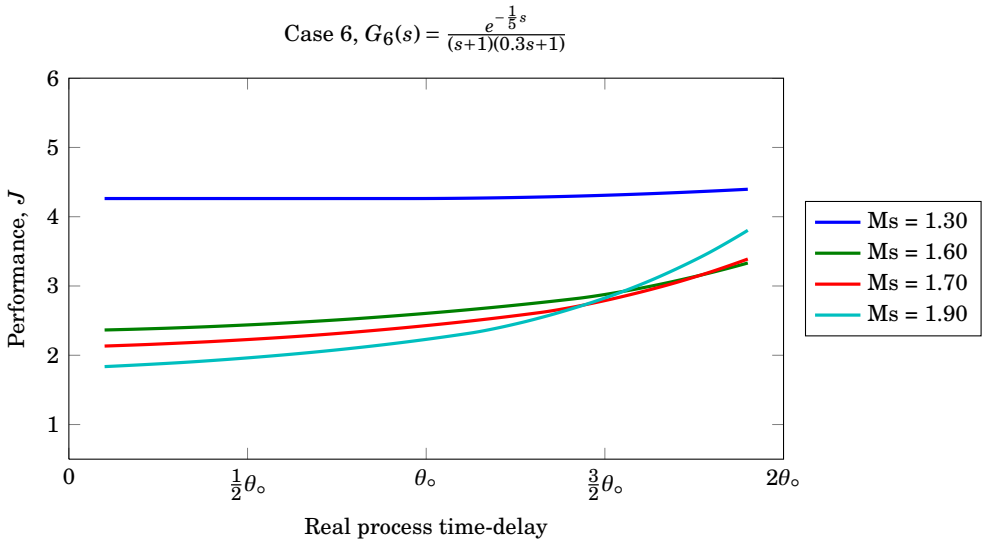


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

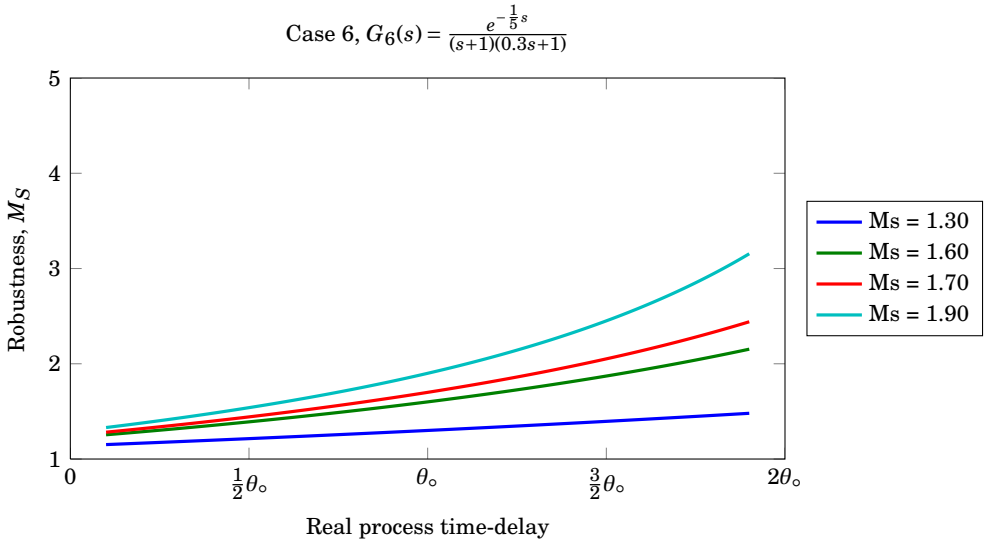


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.5 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

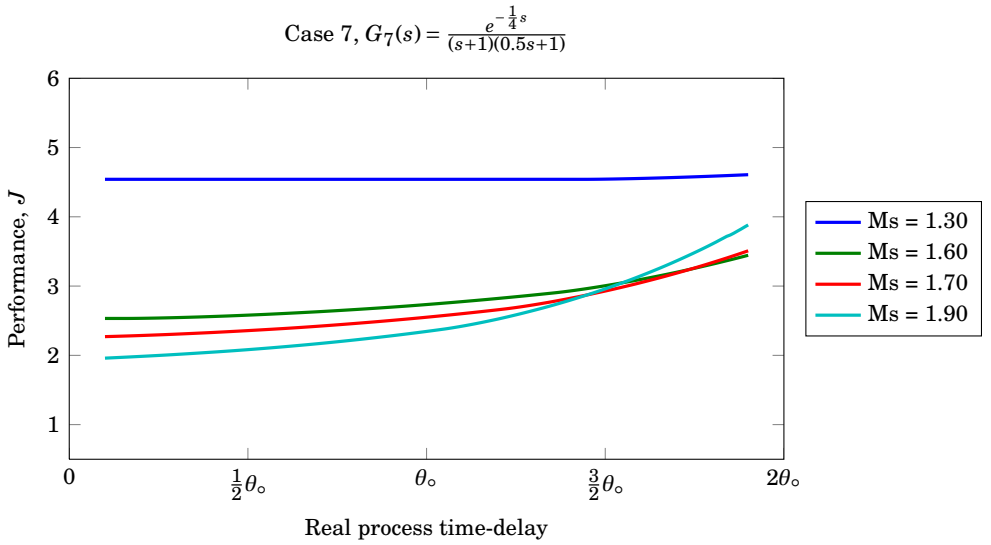


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

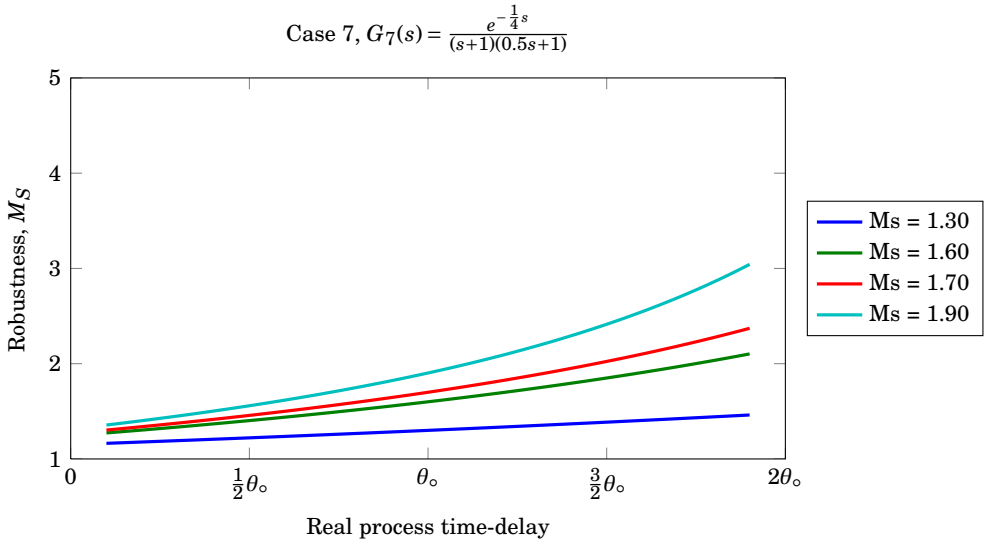


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.6 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

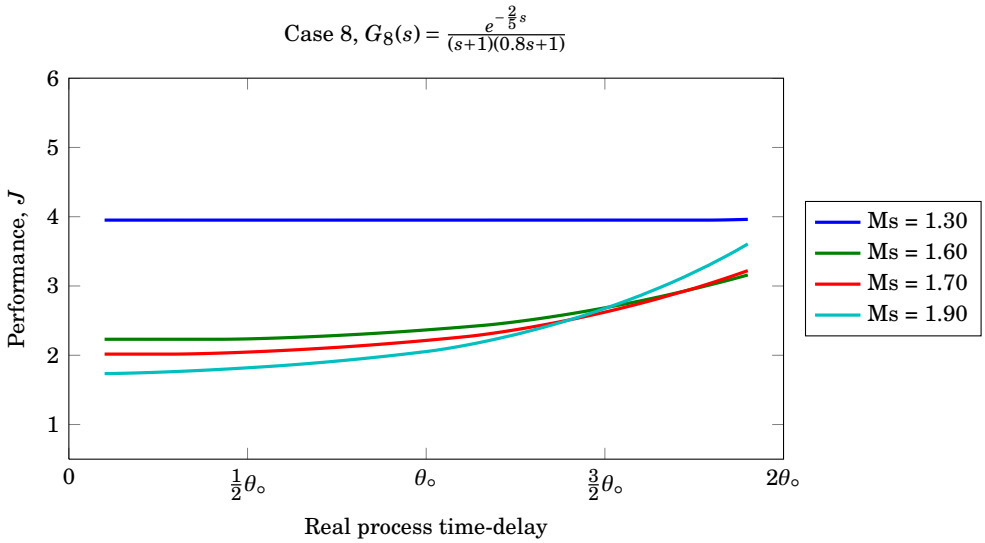


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

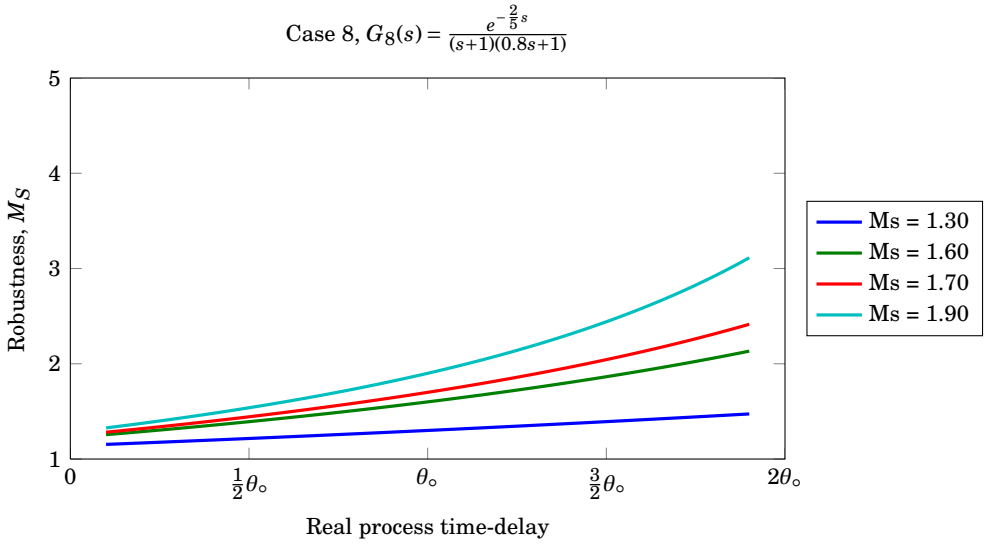


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.7 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

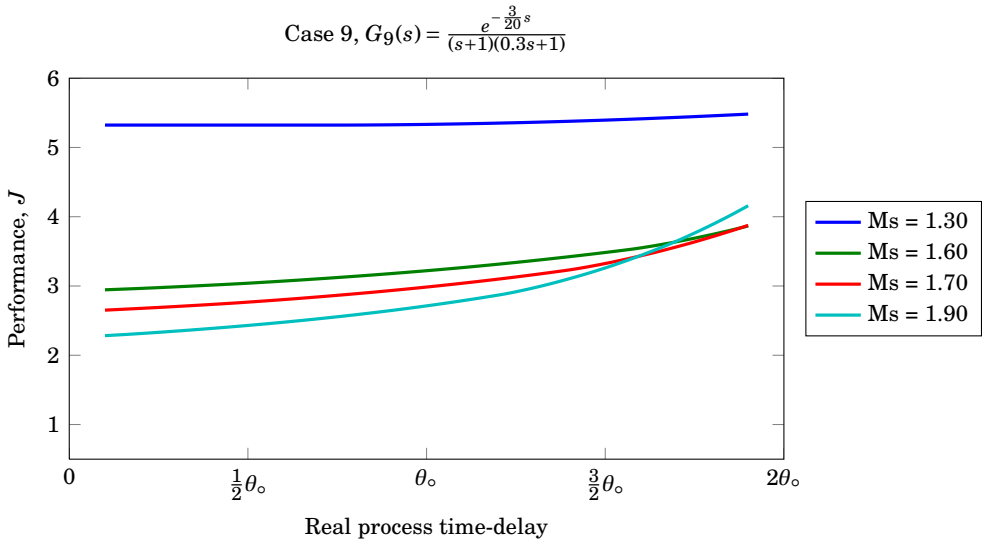


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

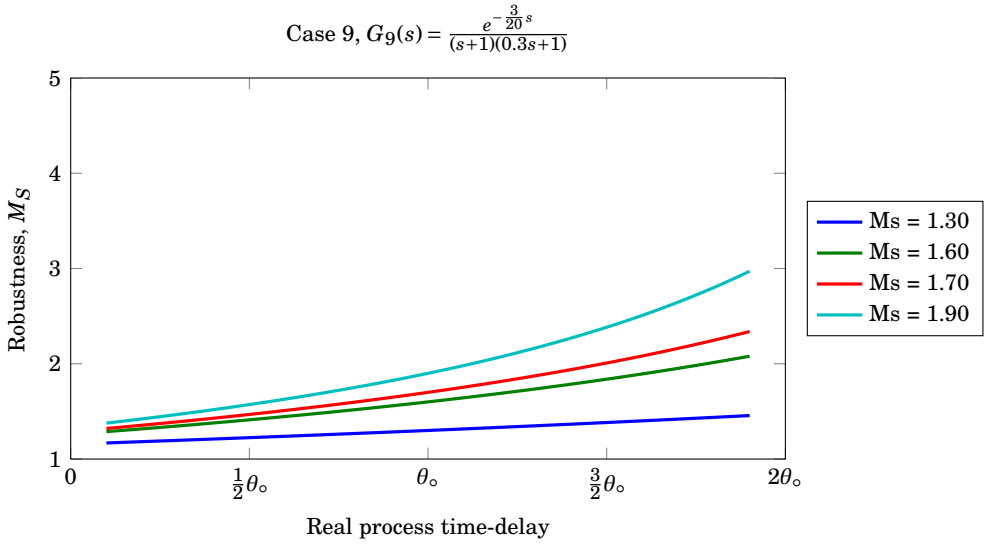


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.8 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

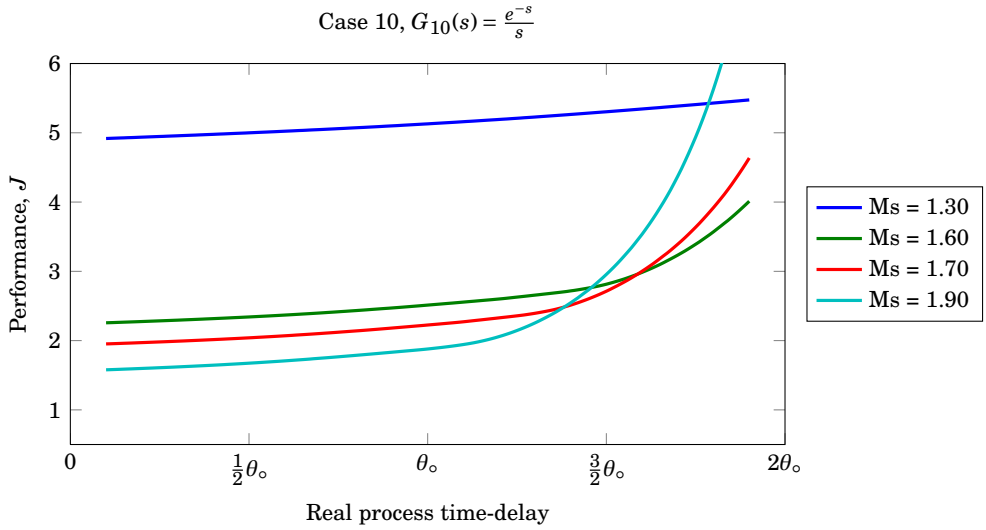


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

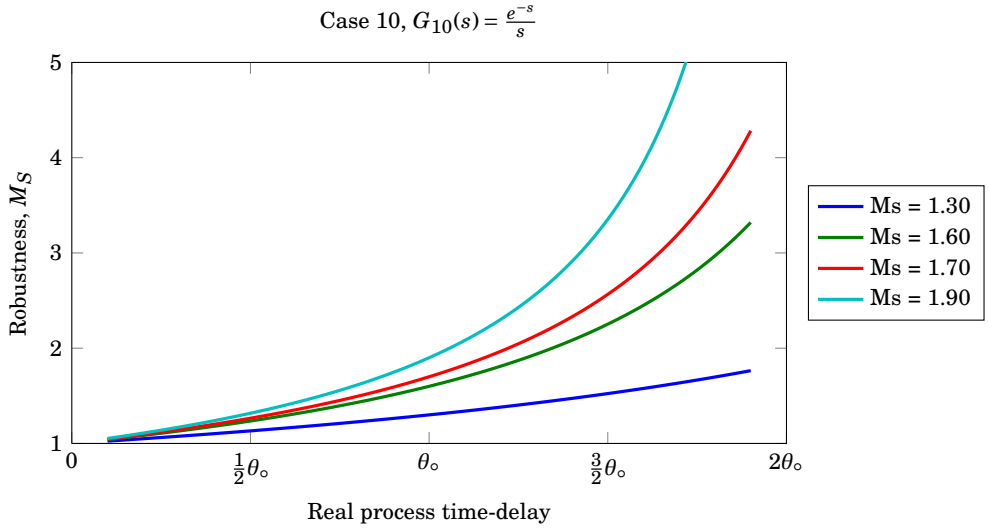


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.9 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

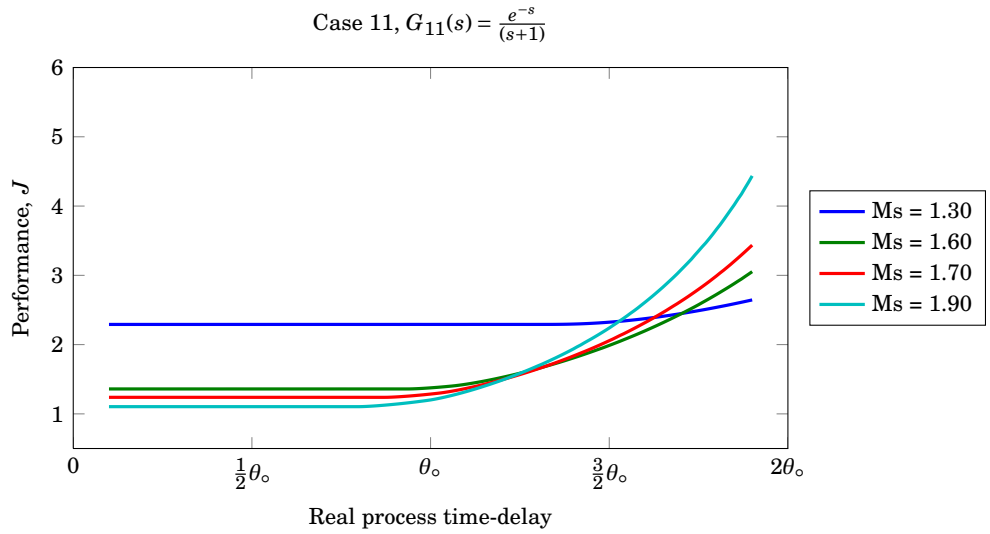


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

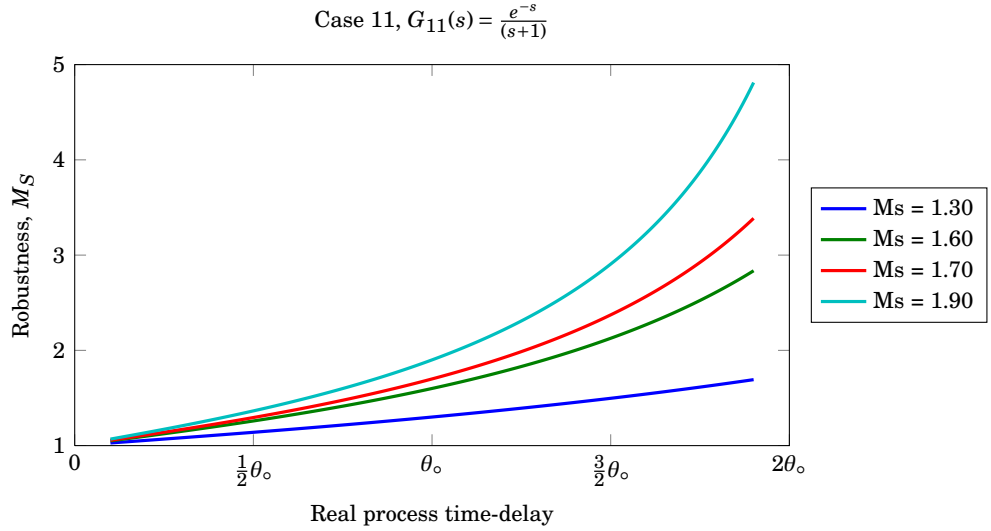


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.10 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_{10}(s) = \frac{e^{-s}}{s}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

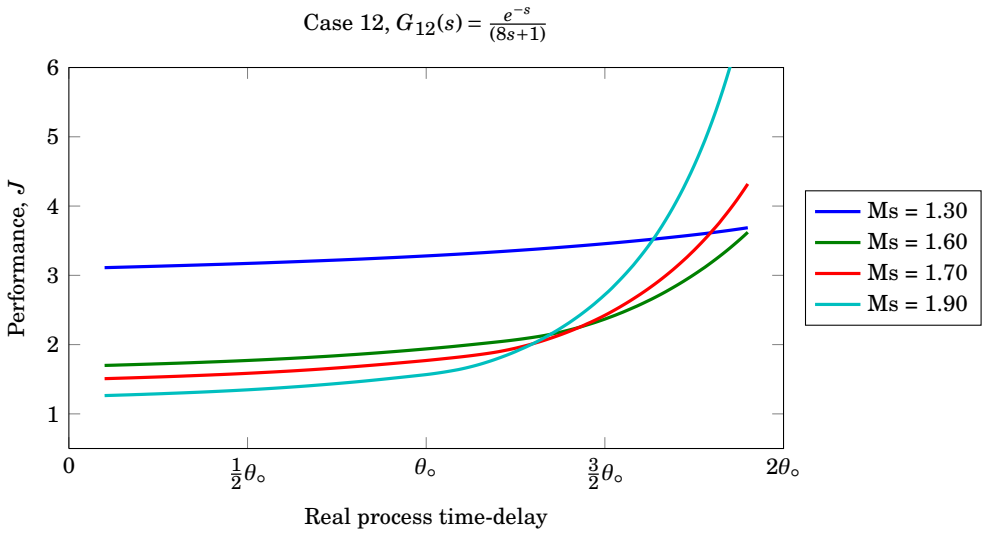


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

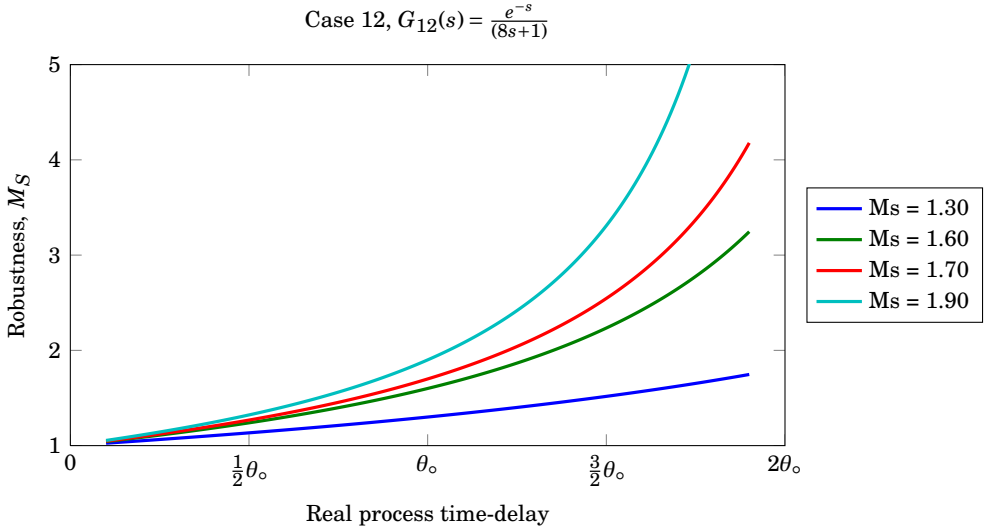


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.11 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_{11}(s) = \frac{e^{-s}}{(s+1)}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

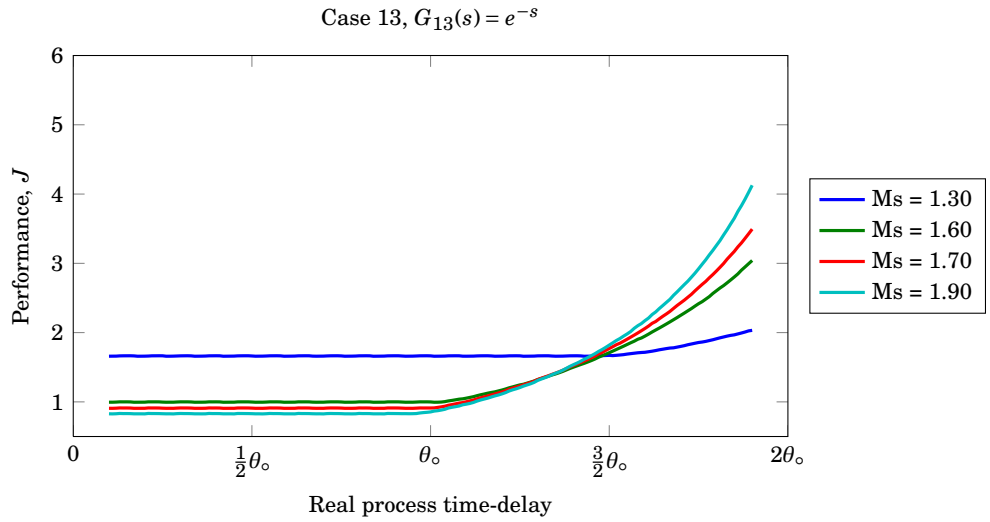


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

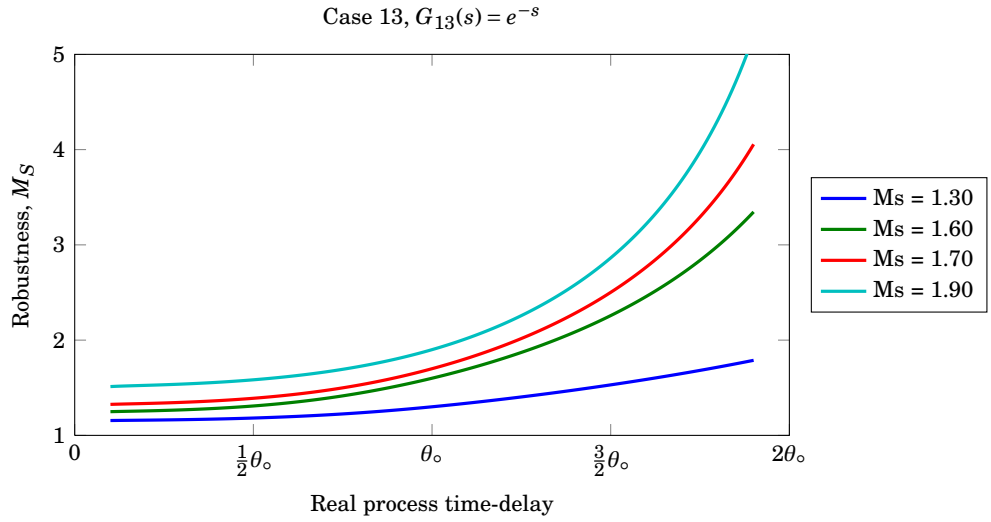


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.12 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_{12}(s) = e^{-s}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

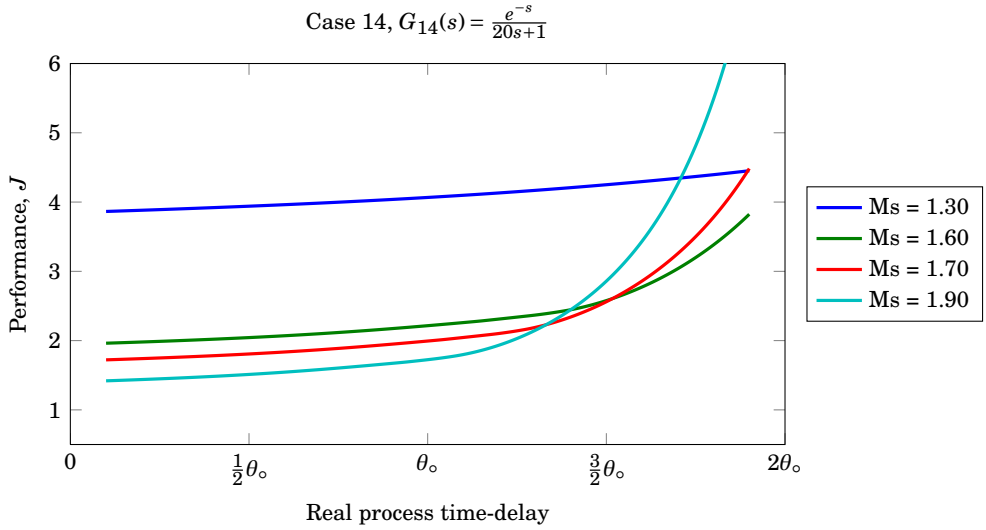


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

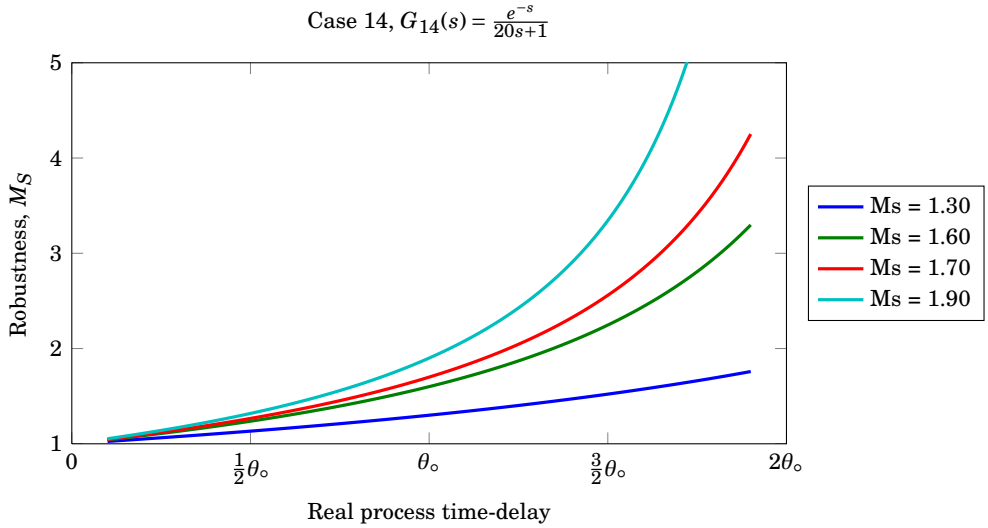


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.13 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_{13}(s) = e^{-s}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.



(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

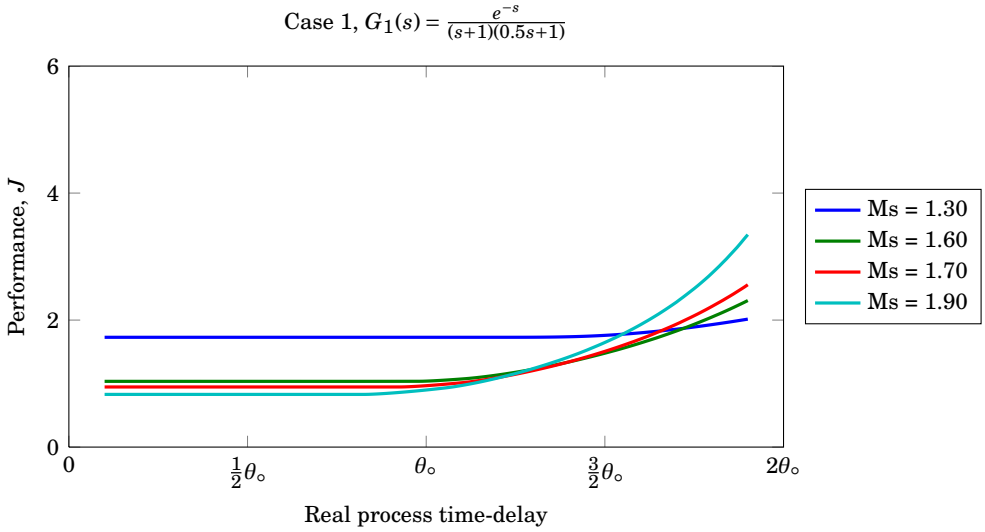


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

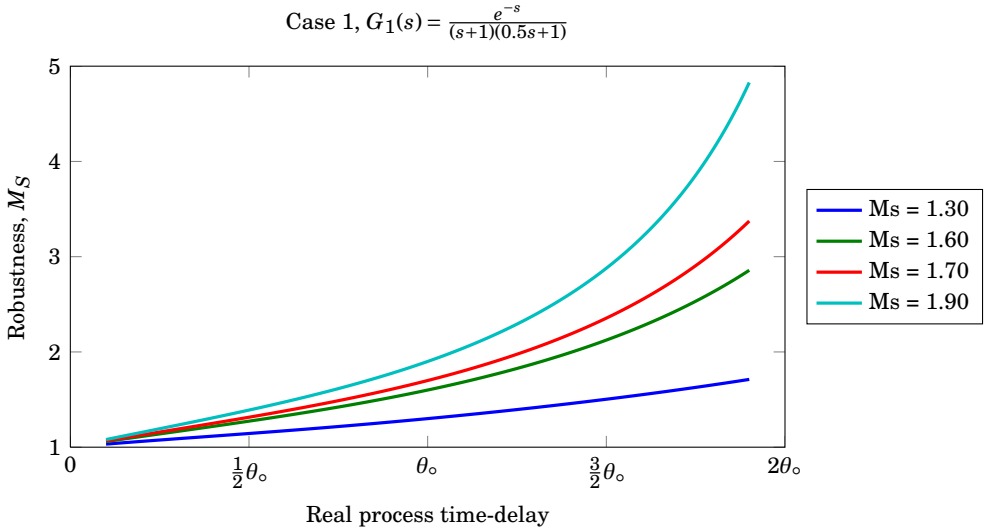
Figure C.14 – Pareto optimal PI controller sensitivity to time-delay modelling error for $G_{14}(s) = \frac{e^{-s}}{20s+1}$. $\theta_\circ$ is the nominal modelled time-delay for the controller design.

C.2 PID Controller Sensitivity

The sensitivity of the Pareto optimal PID controllers for cases 1–14, in terms of performance, are given in Figure C.15(a) through C.28(a). The corresponding robustness efficiency is given in Figure C.15(b) through C.28(b).

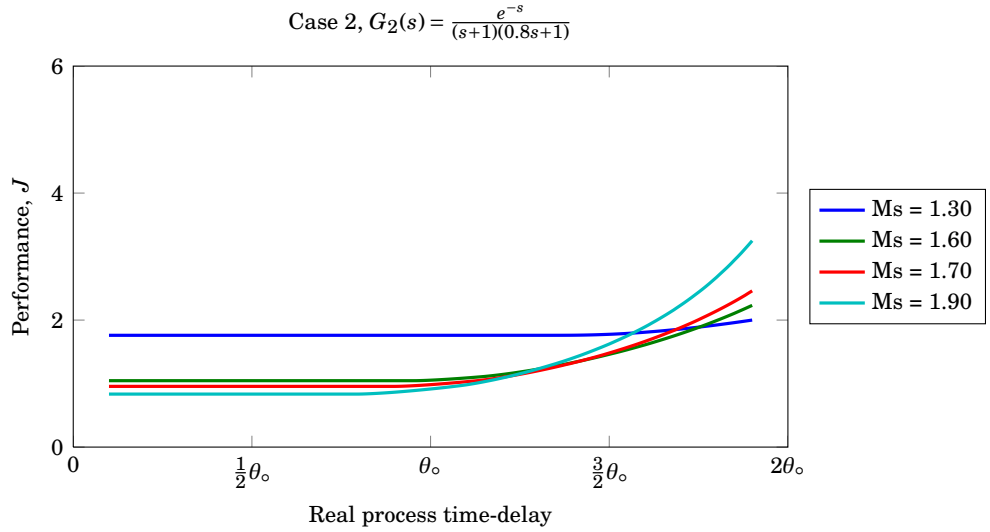


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

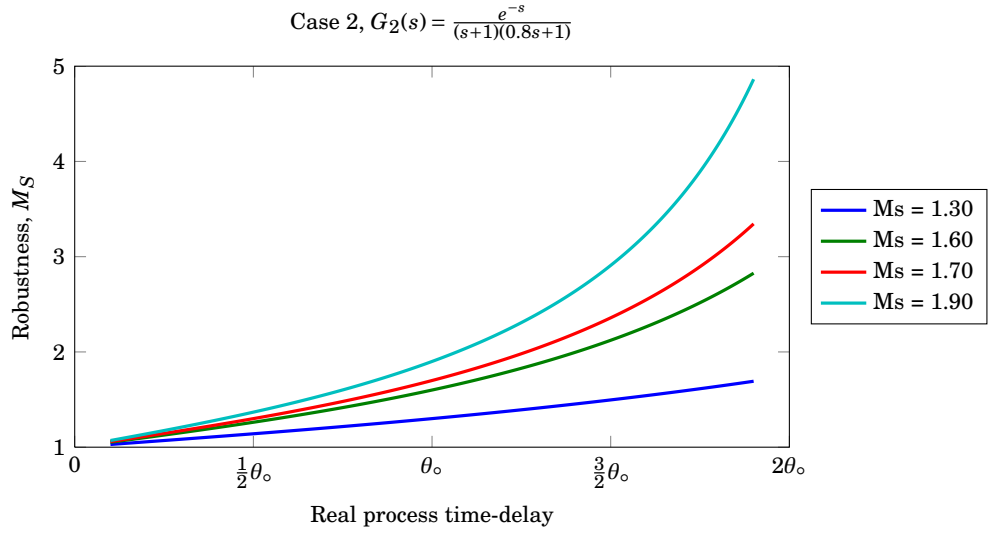


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.15 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

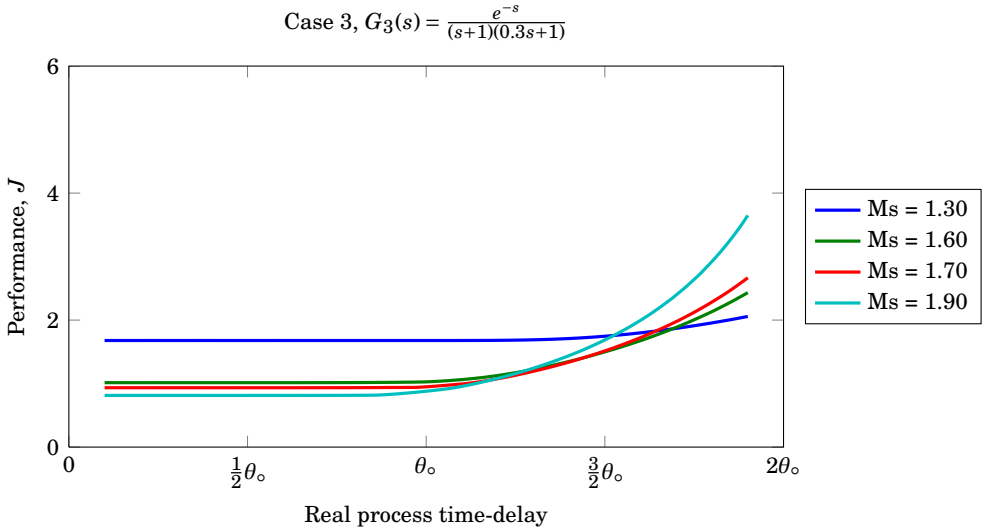


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

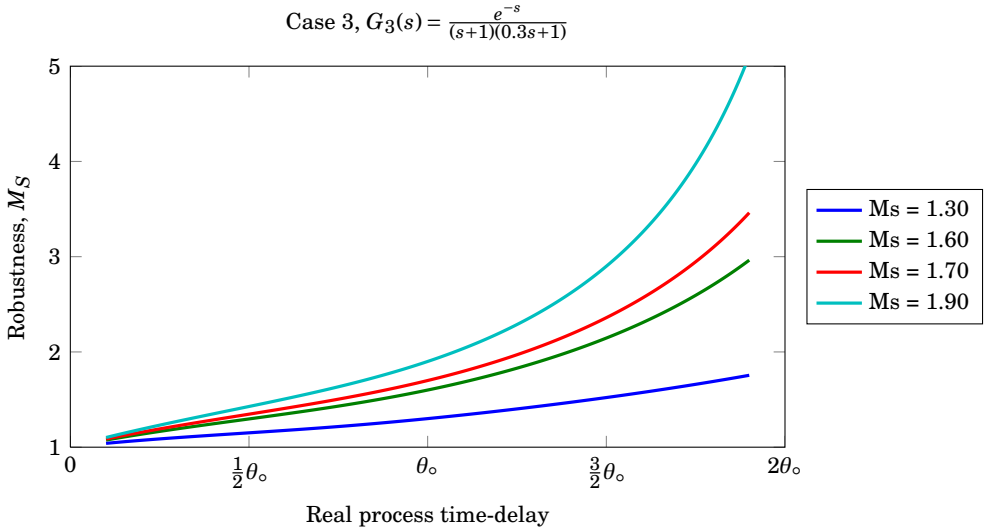


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.16 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

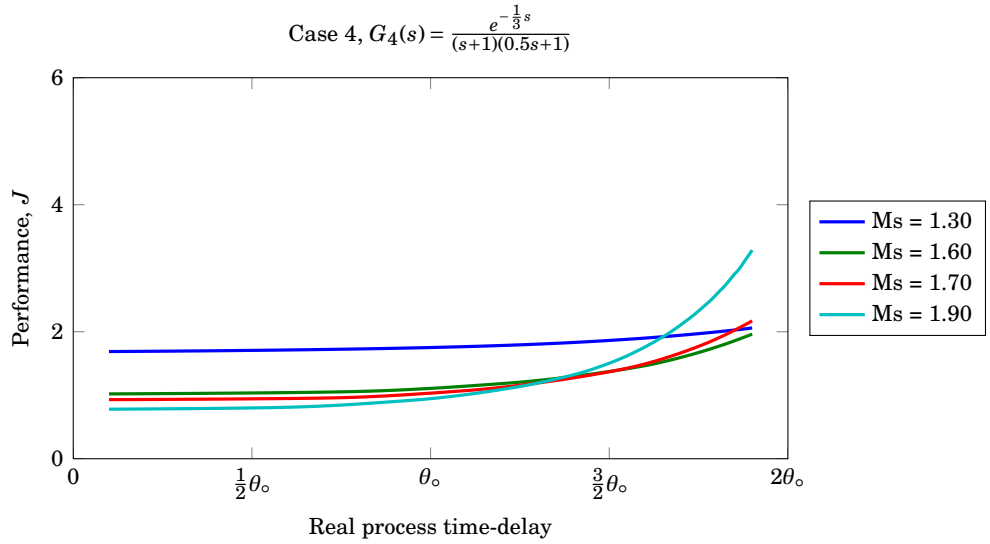


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

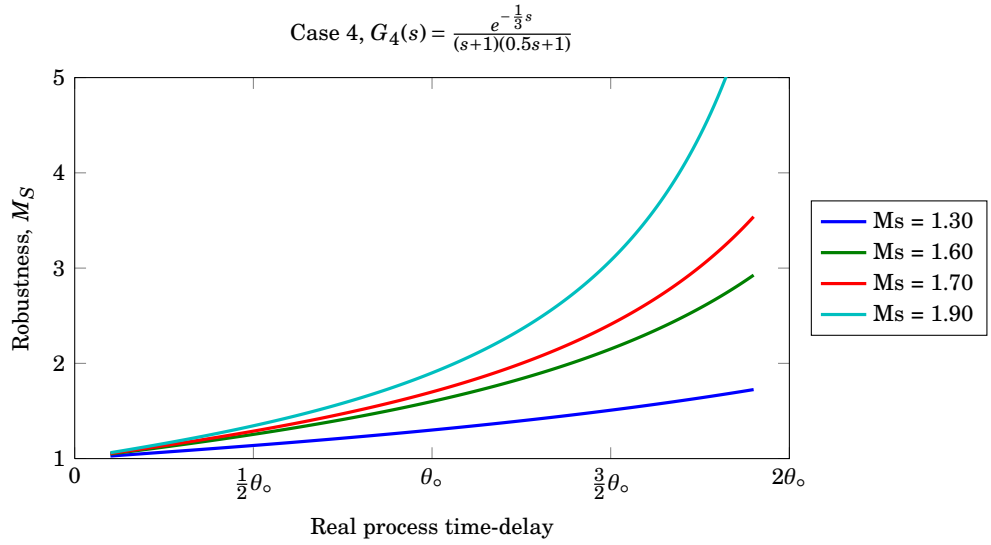


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.17 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

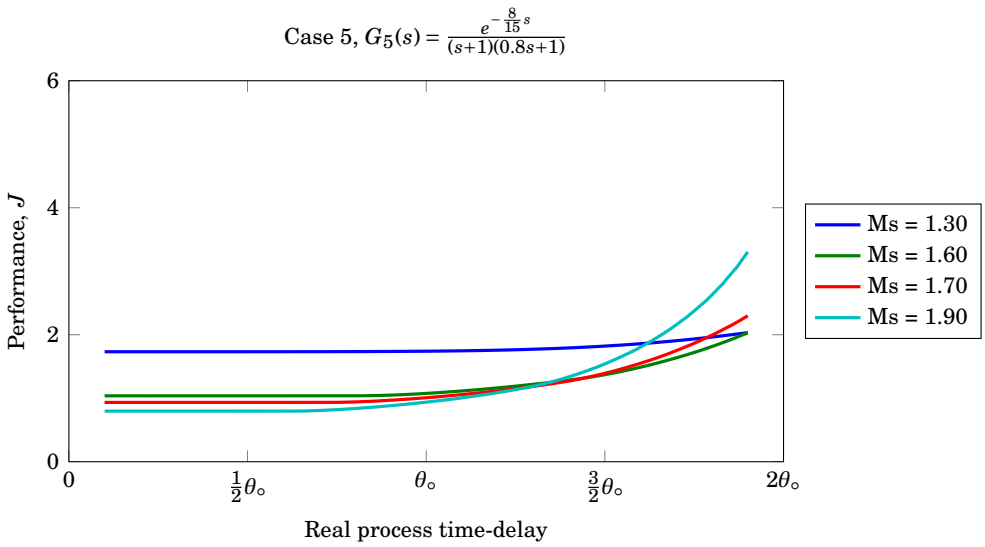


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

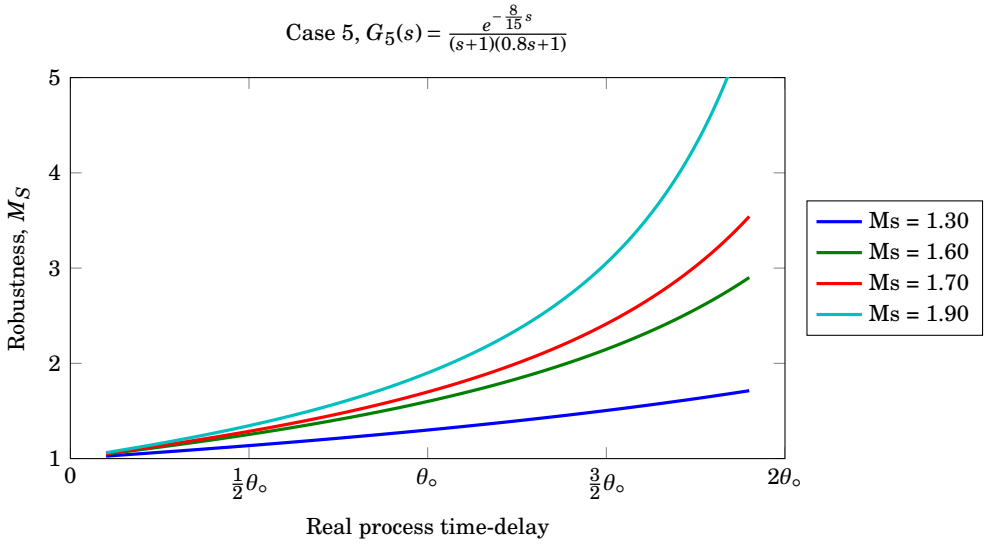


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.18 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

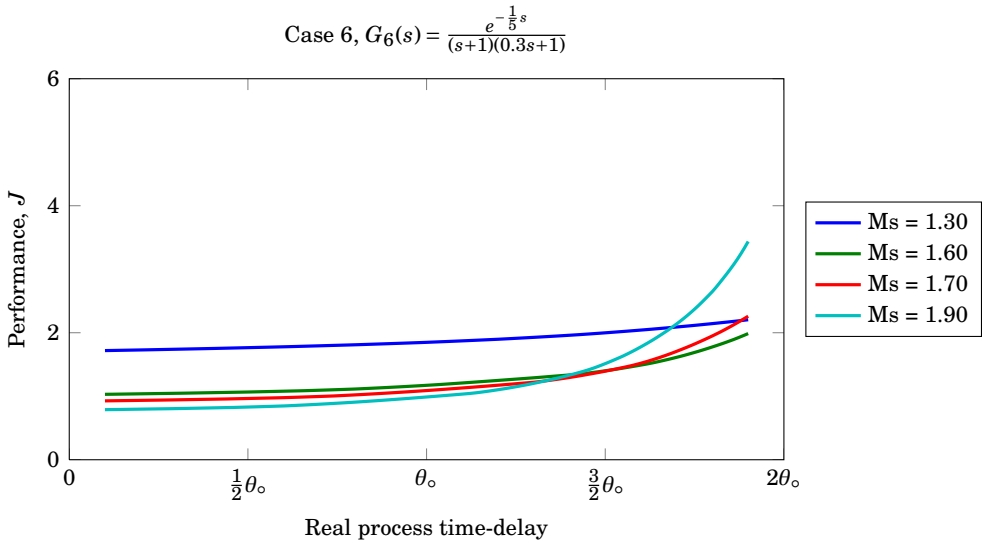


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

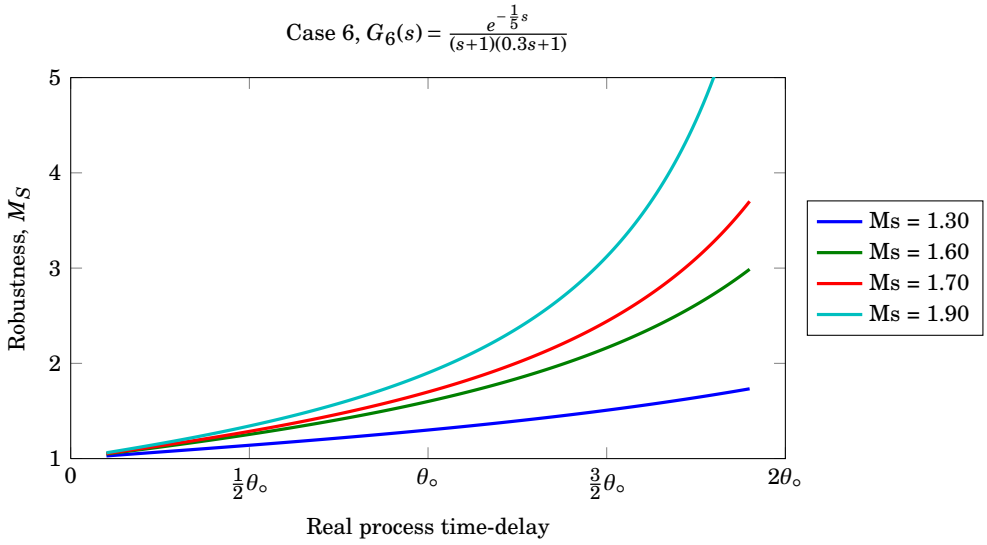


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.19 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

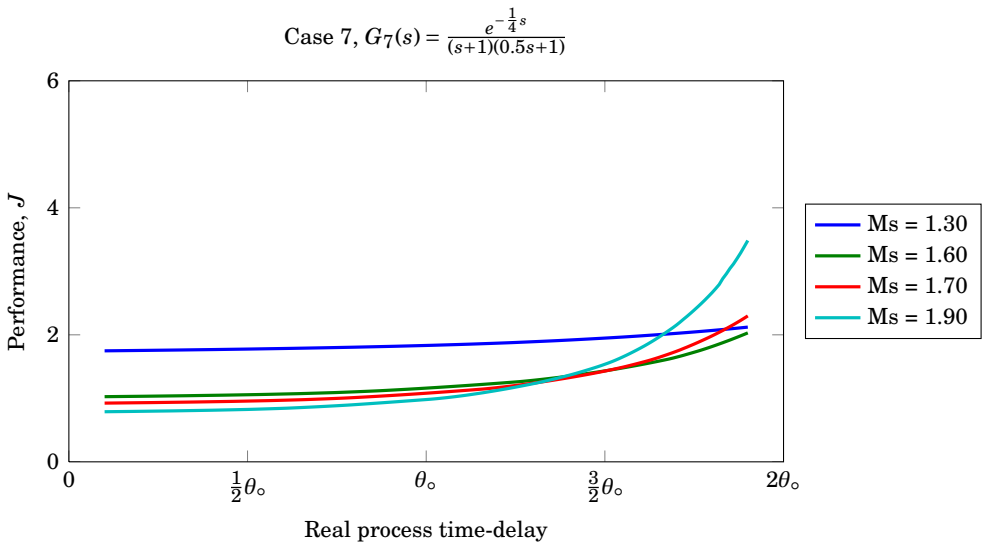


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

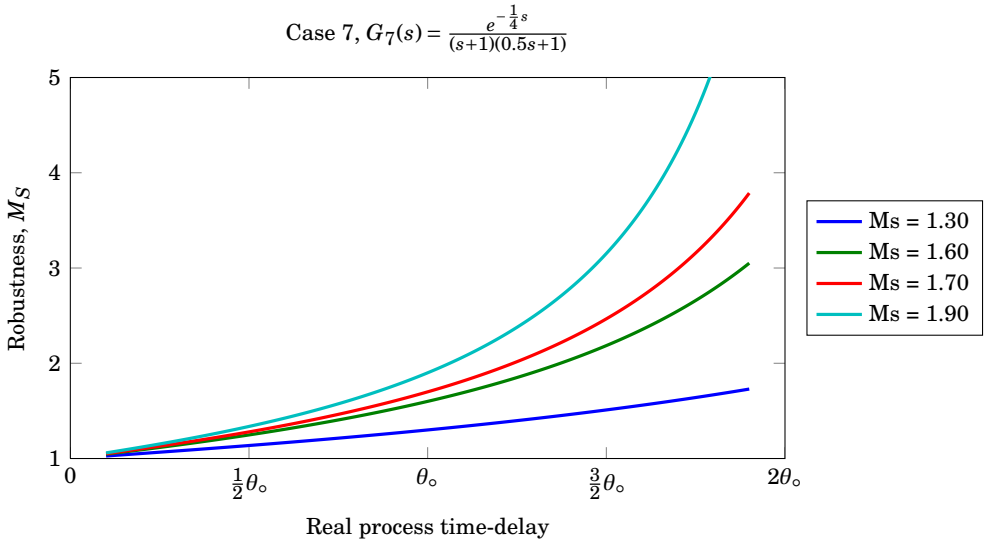


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.20 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

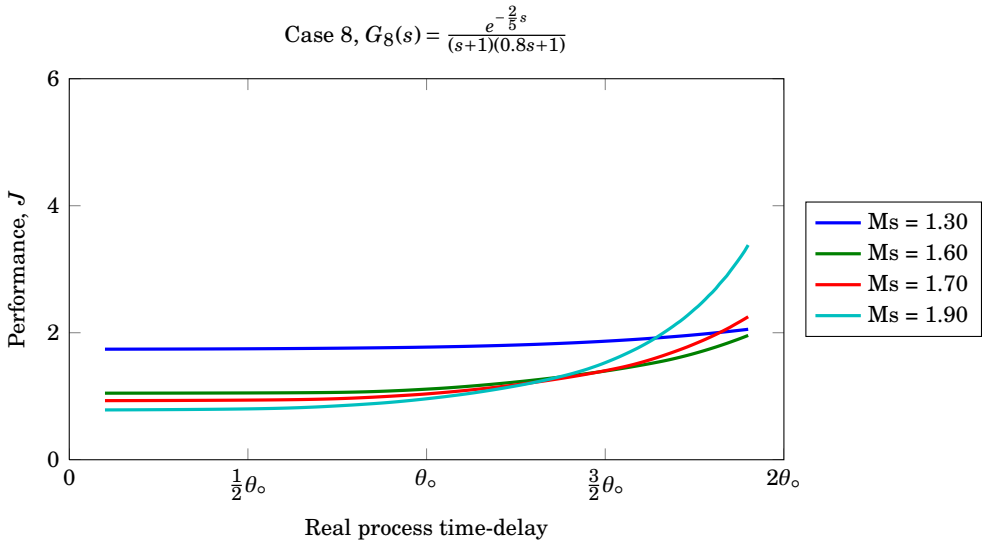


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

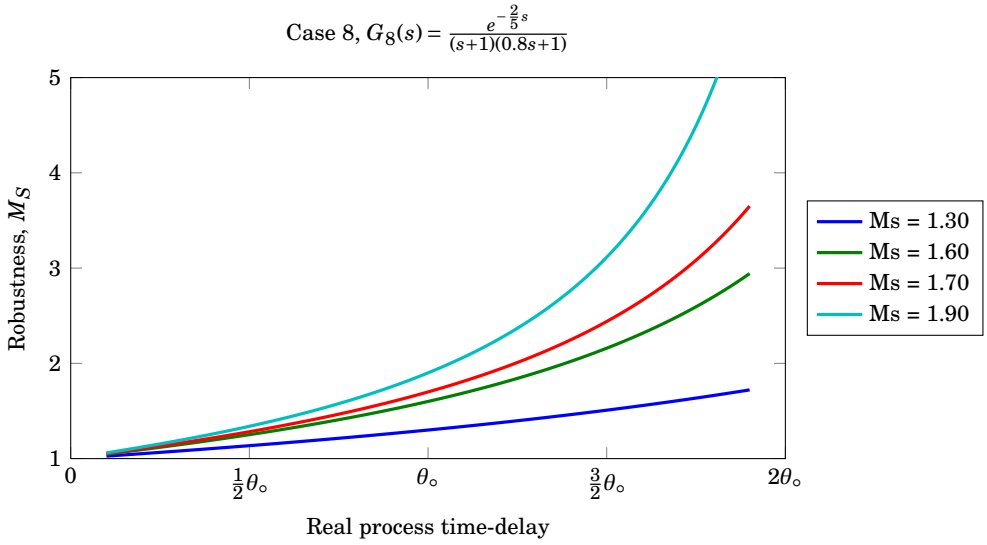


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.21 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

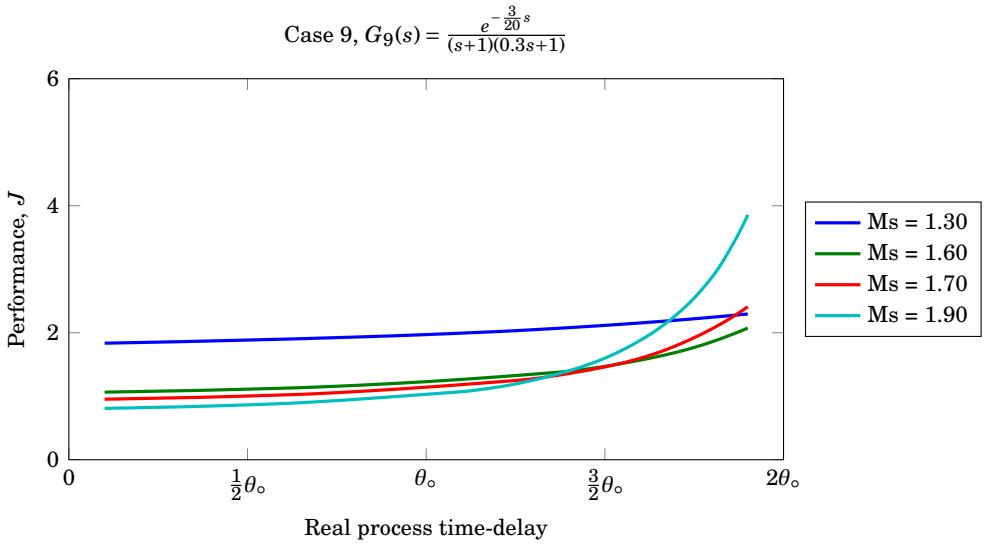


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

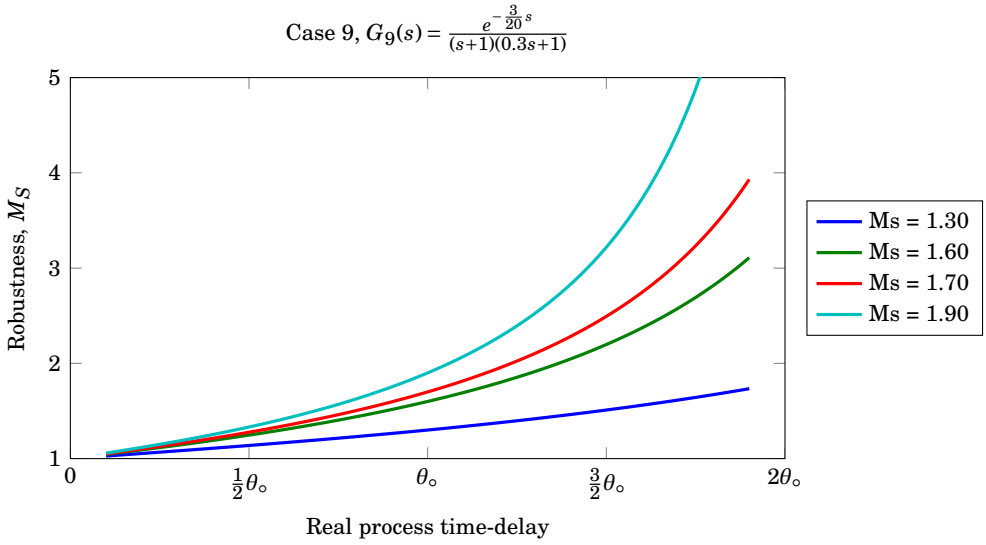


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.22 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

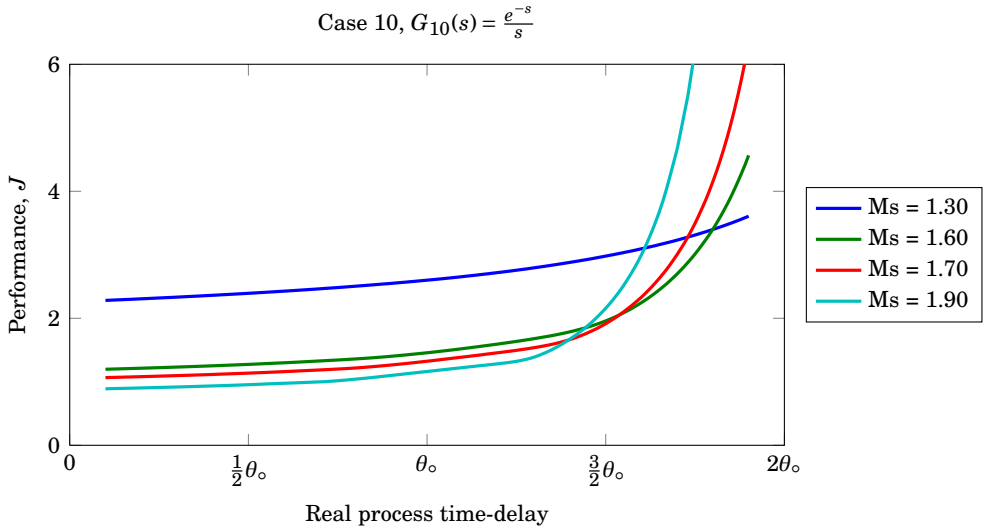


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

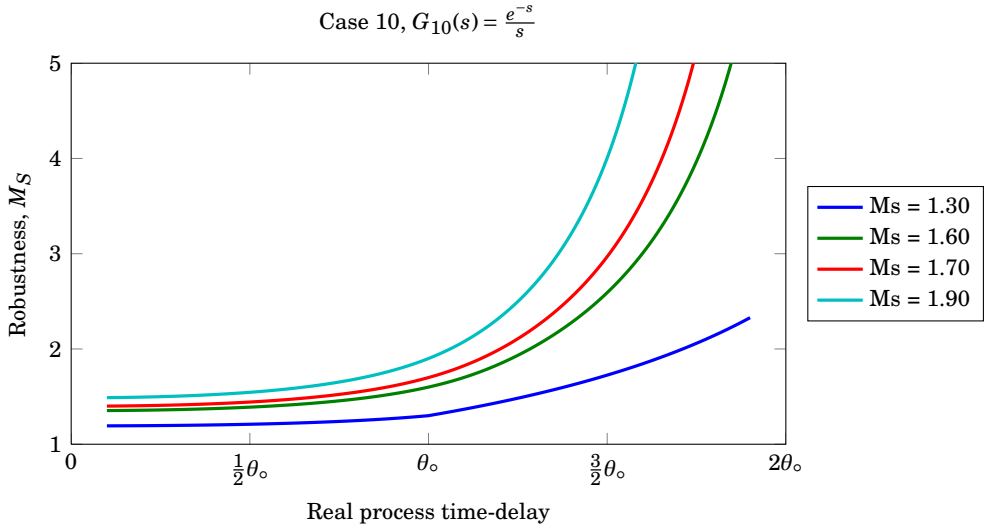


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.23 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

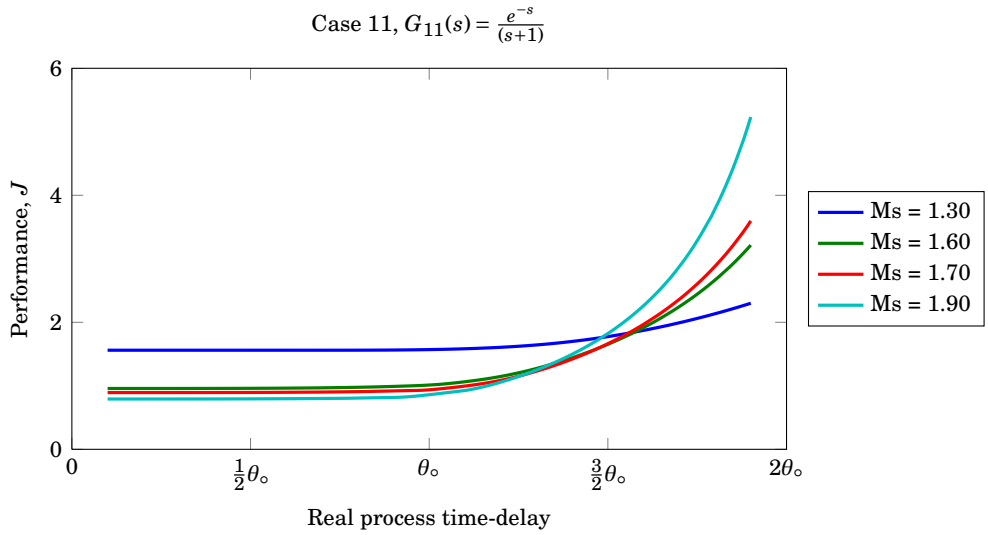


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

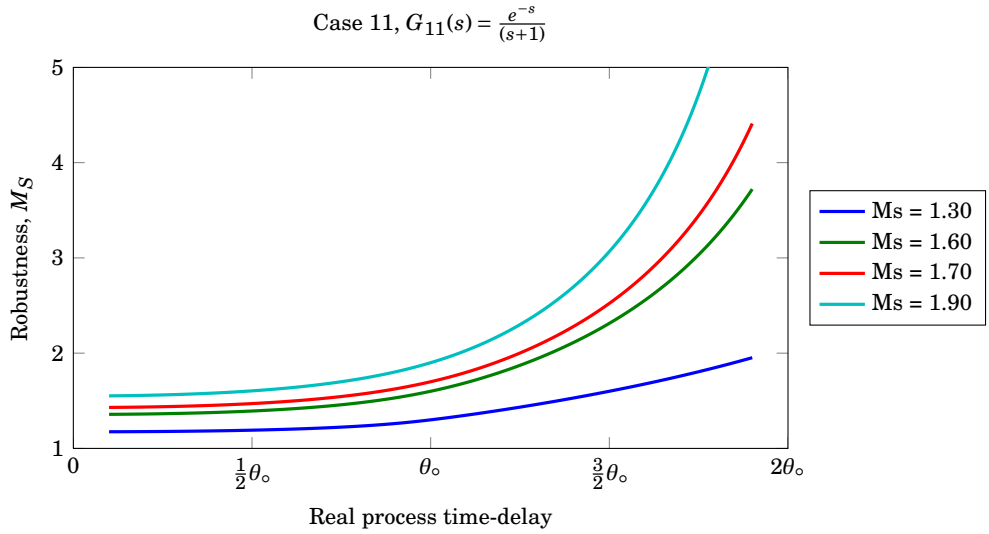


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.24 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_{10}(s) = \frac{e^{-s}}{s}$. $\theta_\circ$ is the nominal modelled time-delay.

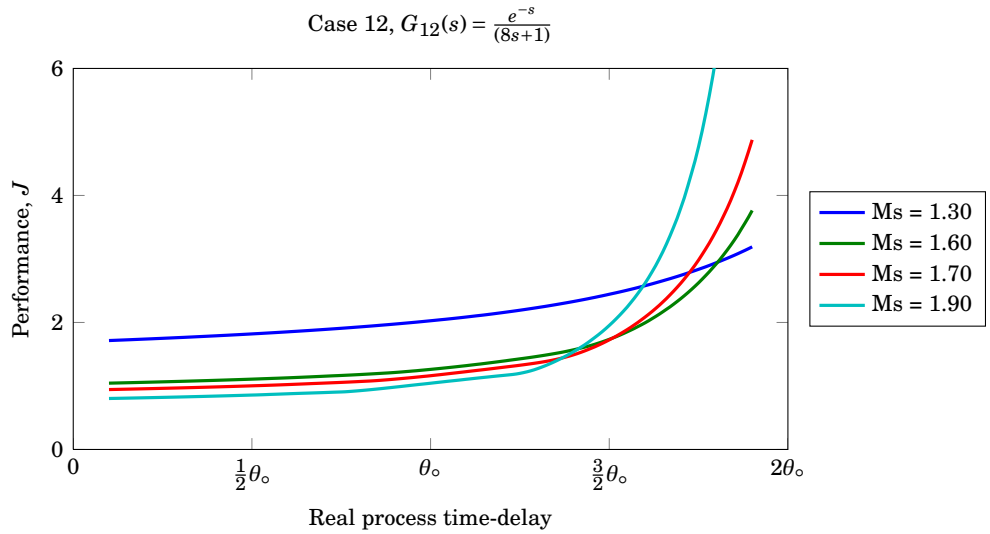


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

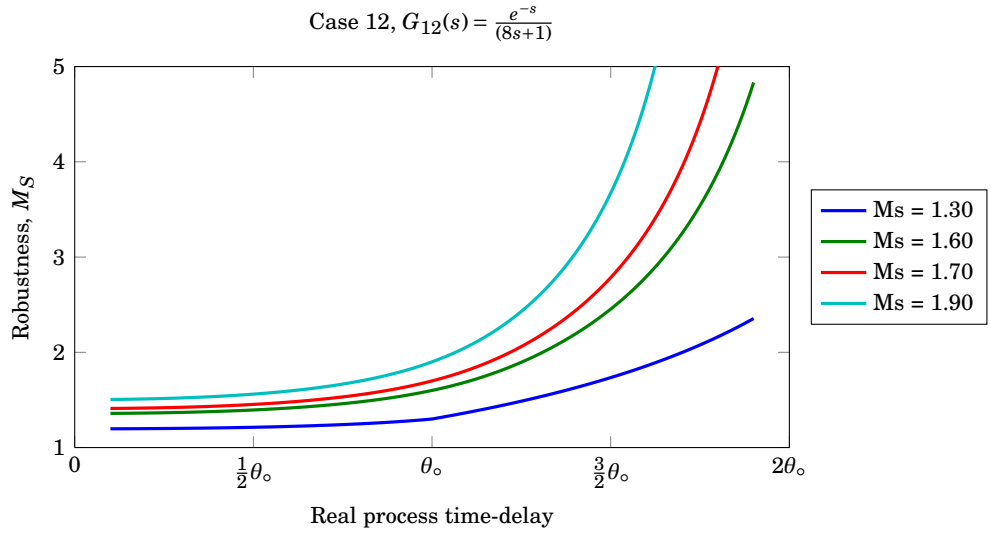


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.25 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_{11}(s) = \frac{e^{-s}}{(s+1)}$. $\theta_\circ$ is the nominal modelled time-delay.

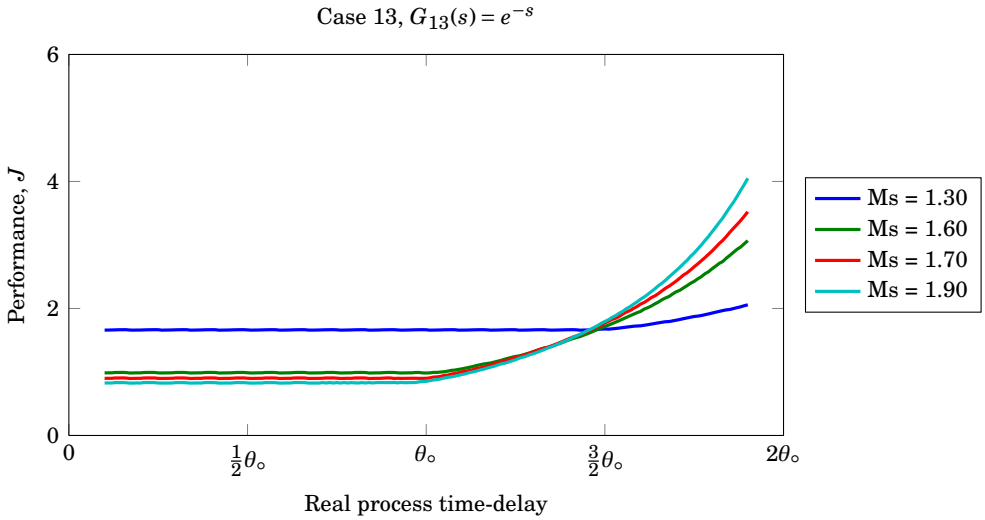


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

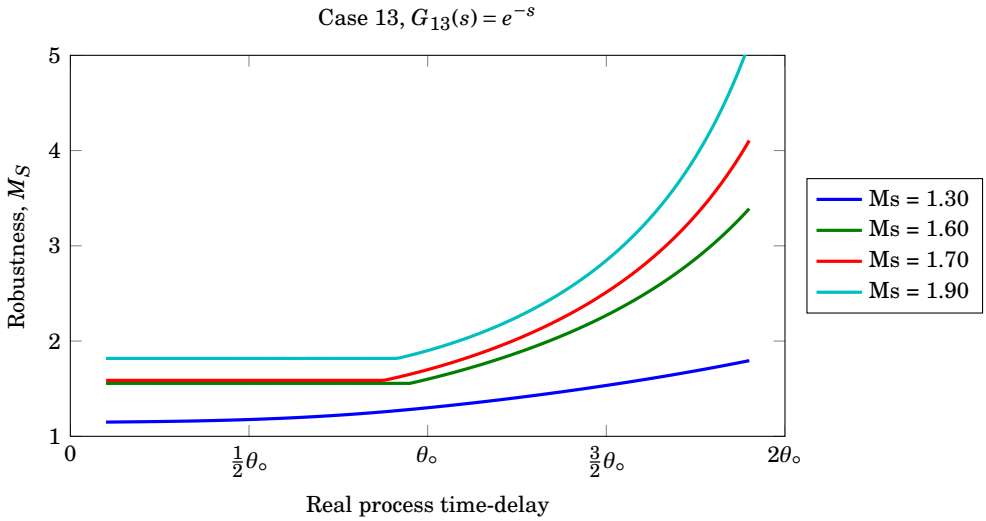


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.26 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_{12}(s) = e^{-s}$. $\theta_\circ$ is the nominal modelled time-delay.

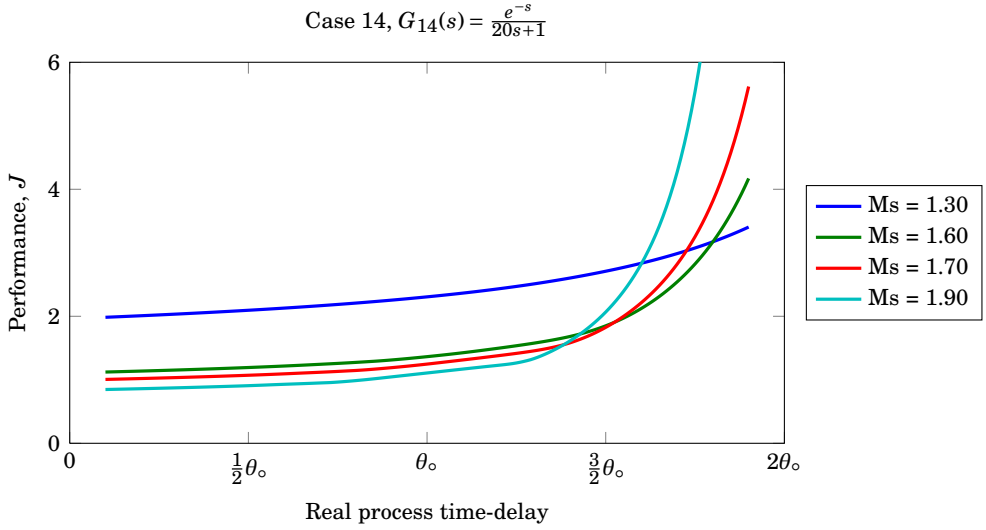


(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.

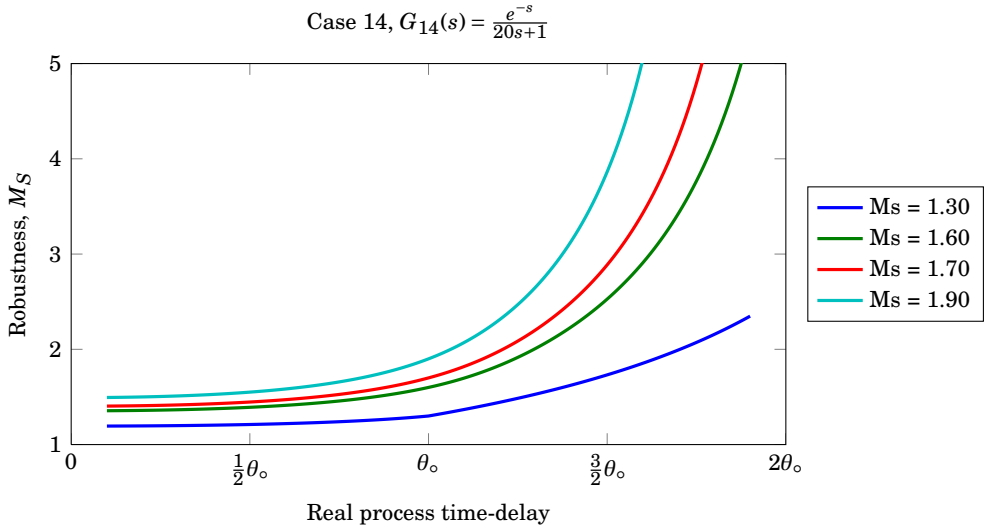


(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.27 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_{13}(s) = e^{-s}$. $\theta_\circ$ is the nominal modelled time-delay.



(a) Cost function value as a function of time-delay modeling error for a set of target M_S values, $J = f(\theta)$.



(b) Robustness as function of the real process time-delay when modelling error occur, plotted for a set of target M_S values, $M_S = f(\theta)$

Figure C.28 – Pareto optimal PID controller sensitivity to time-delay modelling error for $G_{14}(s) = \frac{e^{-s}}{20s+1}$. $\theta_\circ$ is the nominal modelled time-delay.

APPENDIX D

SMITH PREDICTOR PARETO OPTIMAL SOLUTIONS

The Pareto optimal Smith predictor tuning curves have been found according to Algorithm 1 as described in Chapter 3.7. Case 1–14 have been investigate. The results are given in Chapter D.2. The controller tuning performance is represented as a function of robustness, along with the maximum additional time-delay before instability occur ($\theta_{\max}$). Tuning examples are given in Chapter D.1.

D.1 Smith Predictor Pareto Optimal Tuning Examples

Example tuning values are given for the Pareto optimal Smith Predictor PI and PID controllers in Table D.1 and D.2, respectively. The examples are given for the cascade controller parameterisation.

D.2 Smith Predictor Pareto Optimal Performance

Where they come from.

Figure D.10(a), D.12(a) and D.14(a), that is, the plots for case 10, 12 and 14, have another ordinate scale compared to the rest of the plots. This is be-

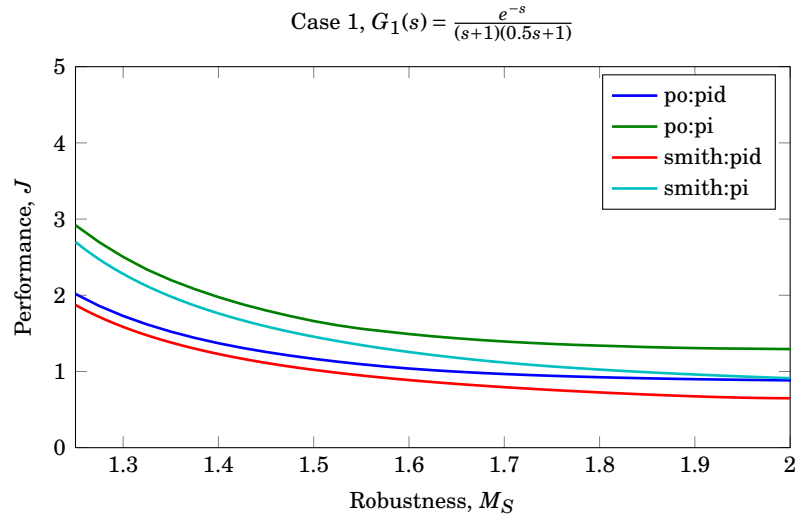
Table D.1 – Pareto optimal Smith PI cascade tuning examples.

Process	K_c	τ_I	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	0.86	1.15	1.26	1.60	2.33	2.33
$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	0.88	1.35	1.33	1.60	2.54	2.54
$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	0.91	1.02	1.20	1.60	2.11	2.11
$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	1.68	1.28	1.88	1.60	1.18	1.10
$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	1.34	1.53	1.68	1.60	1.73	1.67
$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	2.34	1.06	2.22	1.60	0.77	0.65
$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	1.92	1.29	2.31	1.60	1.05	0.92
$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	1.55	1.57	1.98	1.60	1.51	1.40
$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	2.65	1.06	2.81	1.60	0.70	0.55
$G_{10}(s) = \frac{e^{-s}}{s}$	1.14	3.58	4.90	1.60	2.36	53.13
$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	1.42	0.93	1.12	1.60	1.67	1.67
$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	9.44	2.67	1.76	1.60	2.04	1.32
$G_{13}(s) = e^{-s}$	0.75	0.32	0.91	1.60	1.45	1.45
$G_{14}(s) = \frac{e^{-s}}{20s+1}$	23.10	3.14	2.61	1.60	2.19	1.08

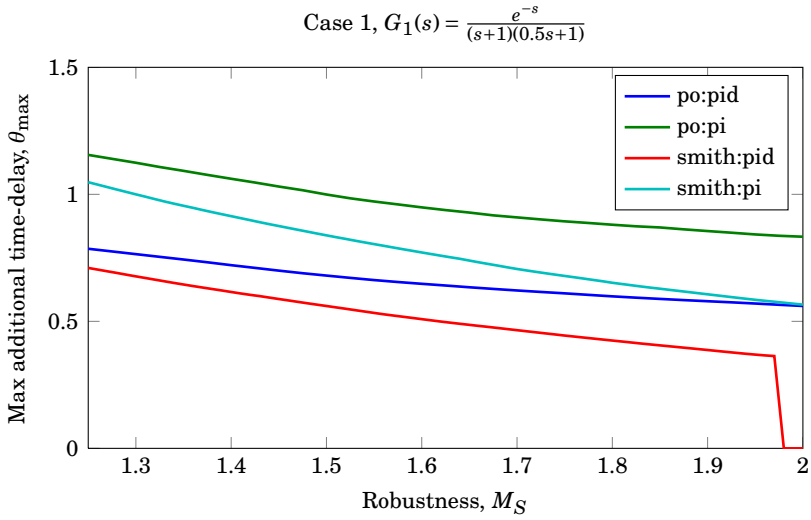
Table D.2 – Pareto optimal Smith PID cascade tuning examples.

Process	K_c	τ_I	τ_D	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	1.20	0.76	0.75	0.89	1.60	1.66	1.65
$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	1.34	0.90	0.88	0.88	1.60	1.70	1.69
$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	1.18	0.65	0.65	0.89	1.60	1.60	1.57
$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	3.98	0.79	0.51	0.93	1.60	0.58	0.55
$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	2.65	0.90	0.75	0.88	1.60	0.91	0.88
$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	5.26	0.43	0.36	1.00	1.60	0.39	0.29
$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	4.91	0.60	0.50	0.97	1.60	0.47	0.37
$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	3.16	0.76	0.82	0.98	1.60	0.77	0.68
$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	7.77	0.42	0.30	1.09	1.60	0.31	0.21
$G_{10}(s) = \frac{e^{-s}}{s}$	1.40	1.07	0.74	4.38	1.60	1.59	49.47
$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	1.51	0.50	0.49	0.91	1.60	1.38	1.33
$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	12.19	0.98	0.59	1.40	1.60	1.52	1.08
$G_{13}(s) = e^{-s}$	0.77	0.32	0.01	0.88	1.60	1.42	1.41
$G_{14}(s) = \frac{e^{-s}}{20s+1}$	7.24	0.46	2.50	2.38	1.60	1.68	1.04

cause the performance for these cases are significantly poorer than for cases 1-9, 11 and 13. To sustain readability of the latter cases, and make the plots for case 10, 12 and 14 easily comparable, the maximum ordinate value is set to 8.

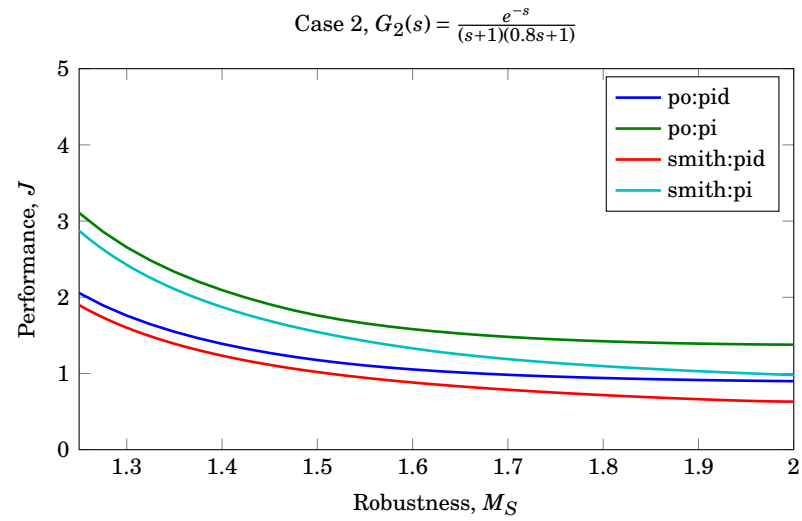


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

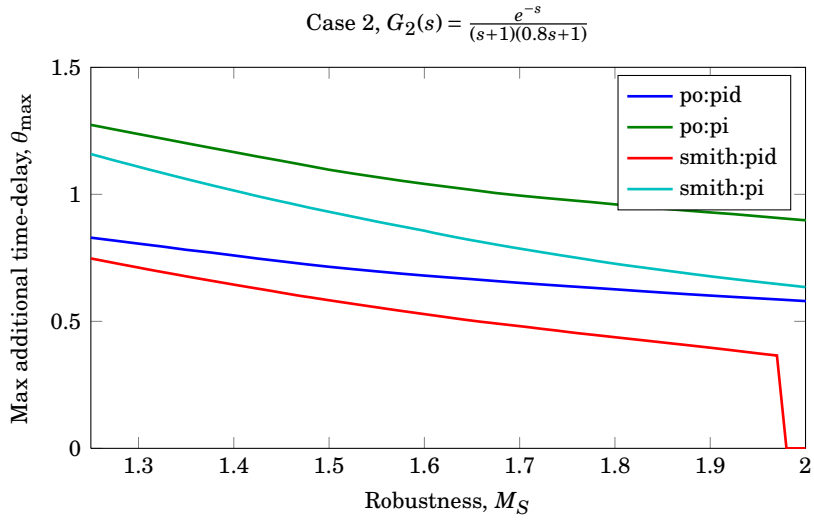


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.1 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$.

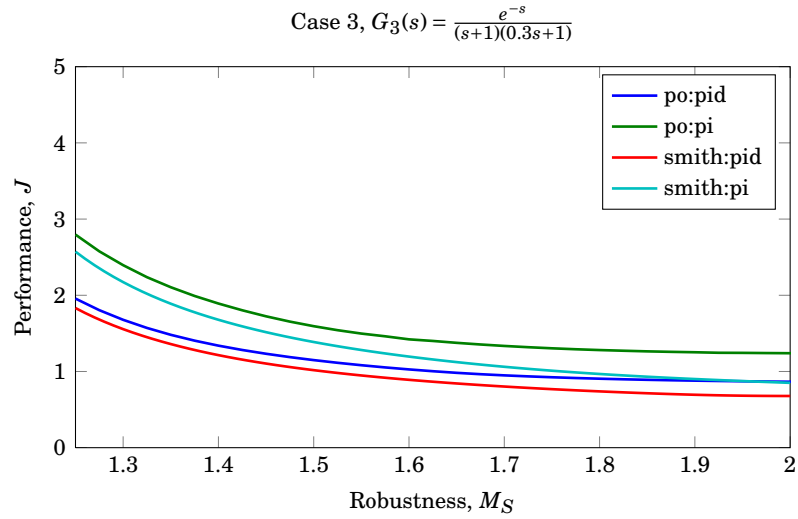


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

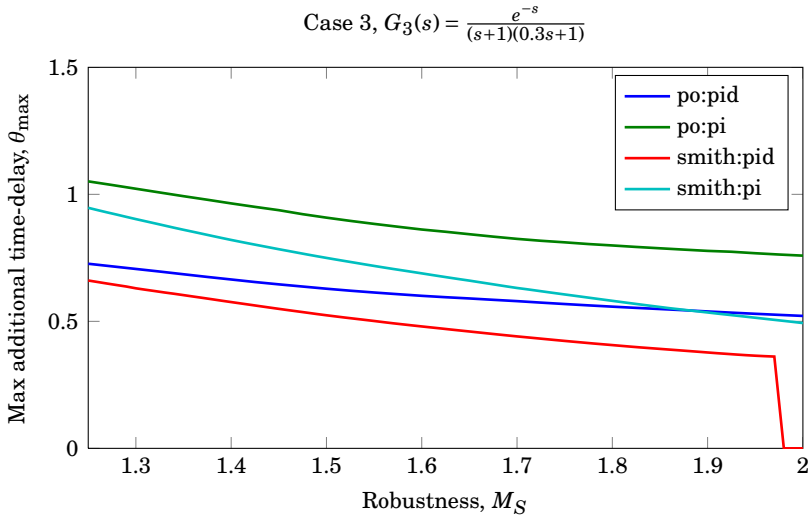


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.2 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_2(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$.

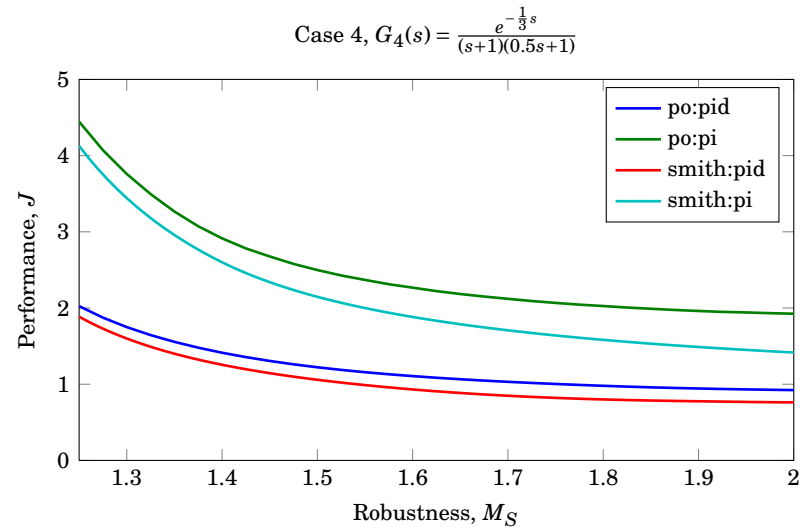


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

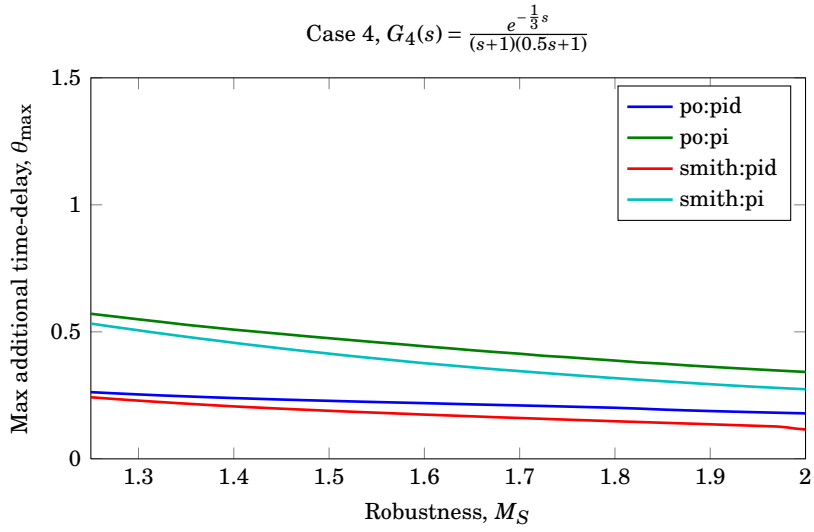


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.3 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_3(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$.

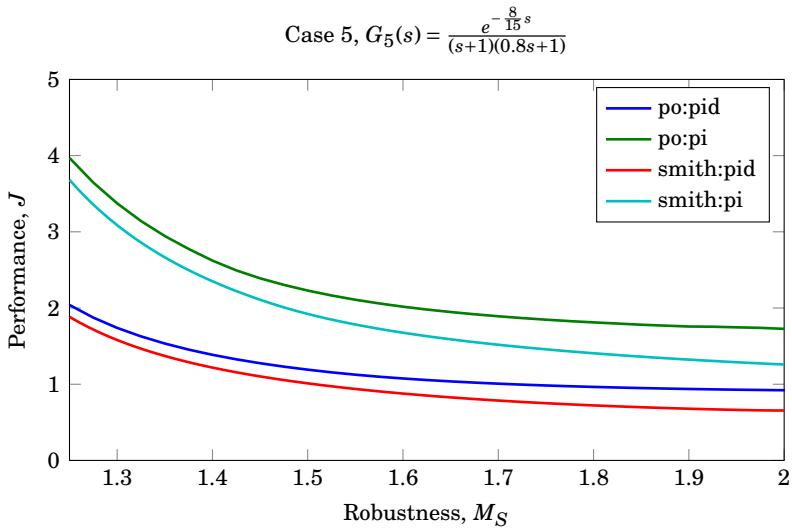


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

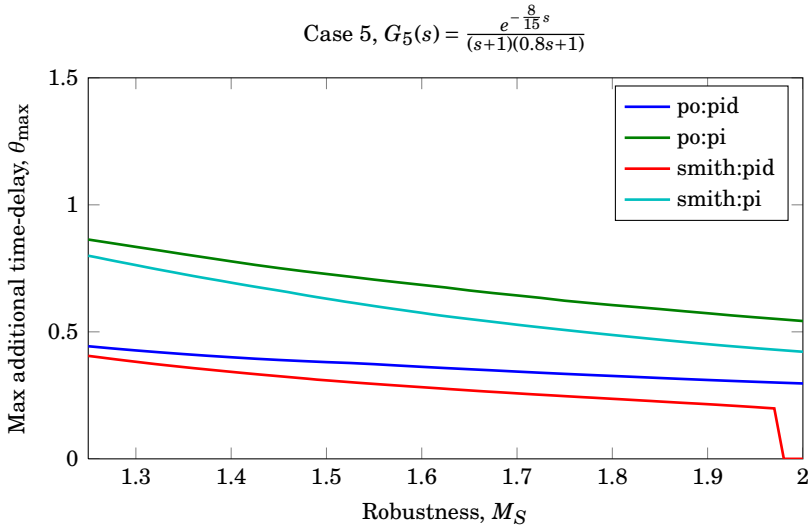


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.4 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$.

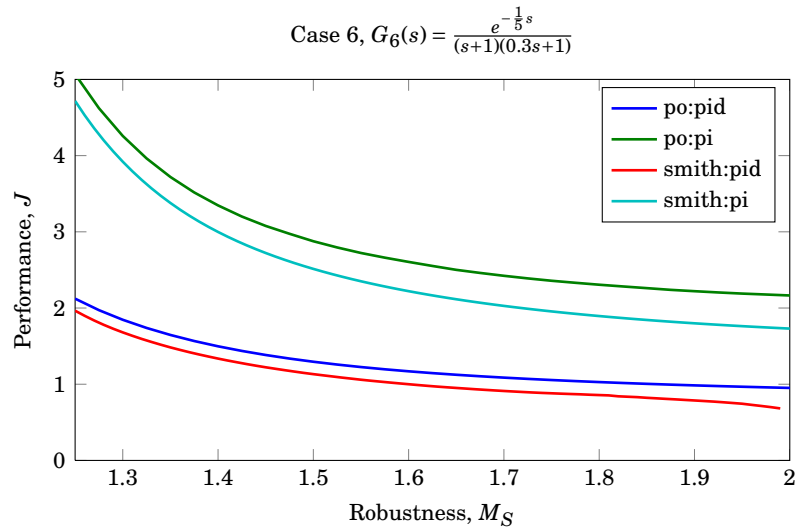


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

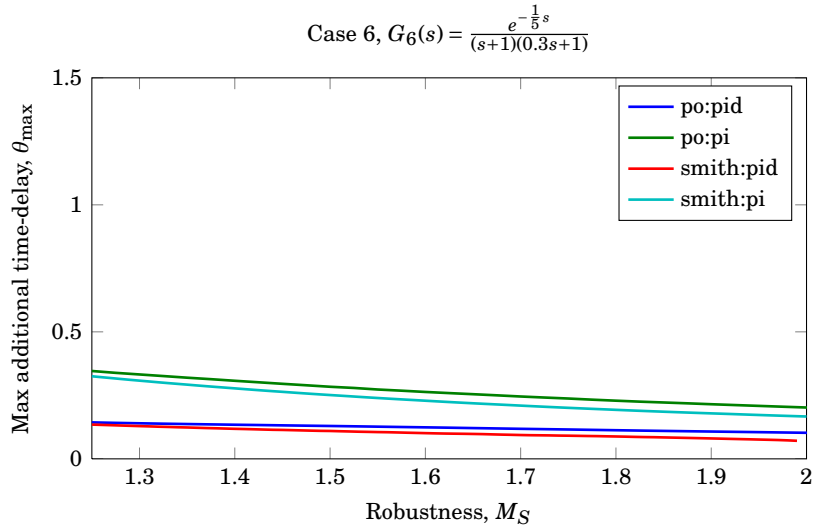


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.5 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$.

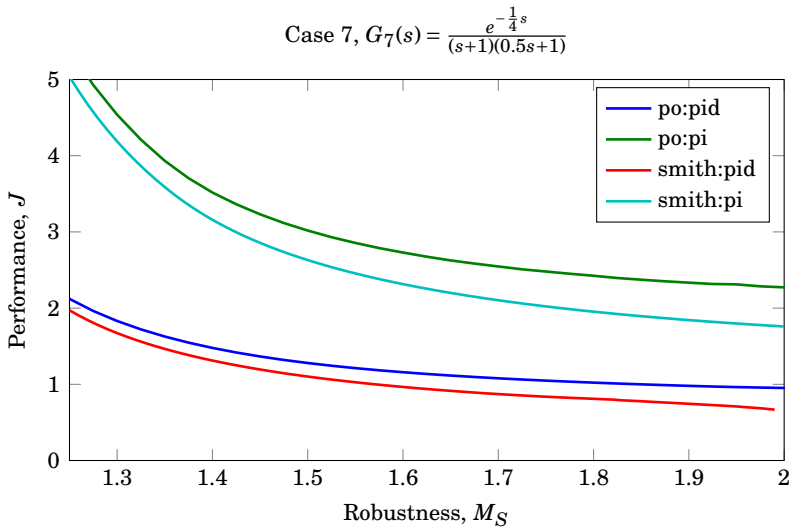


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

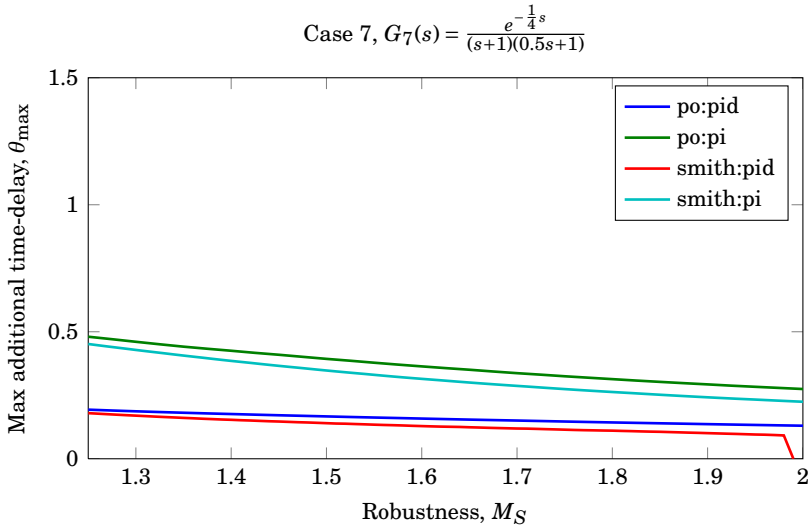


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.6 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$.

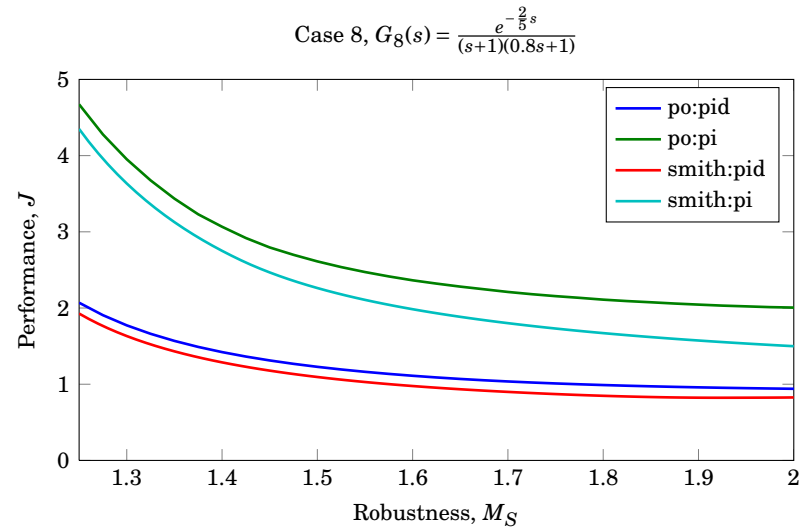


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

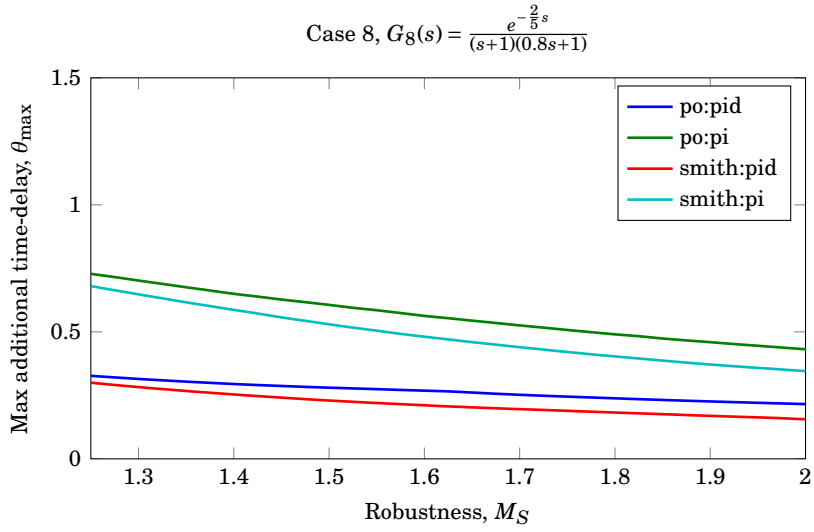


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.7 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$.

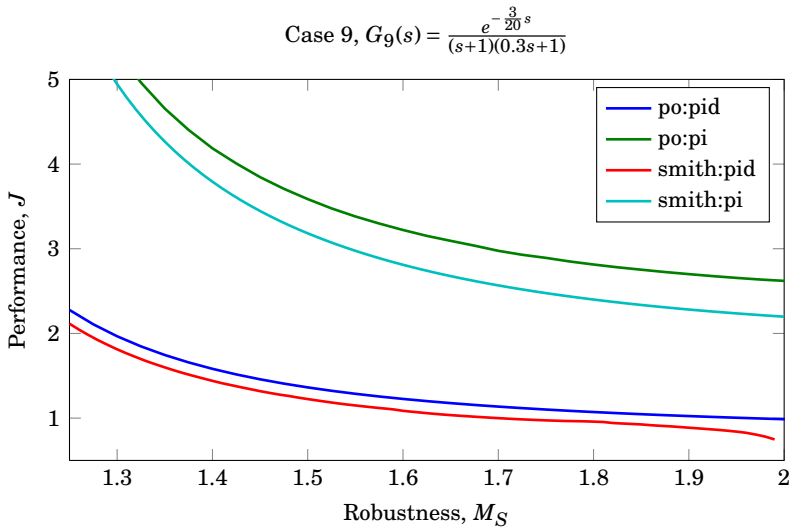


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

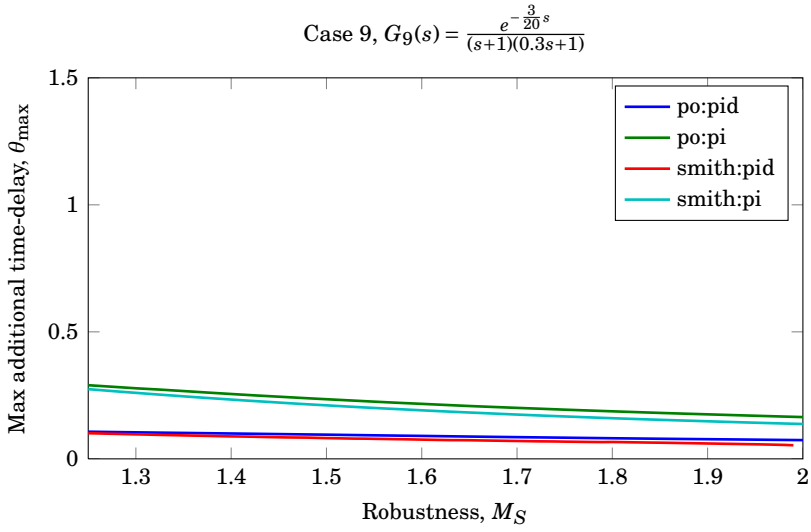


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.8 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$.

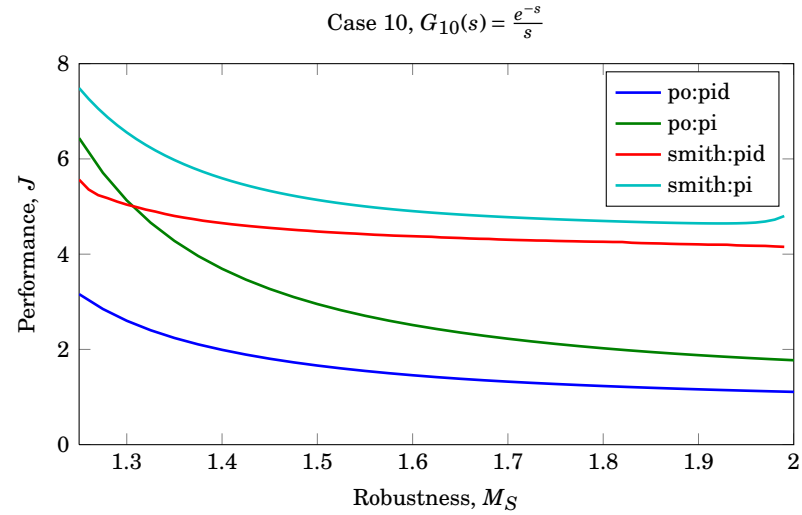


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

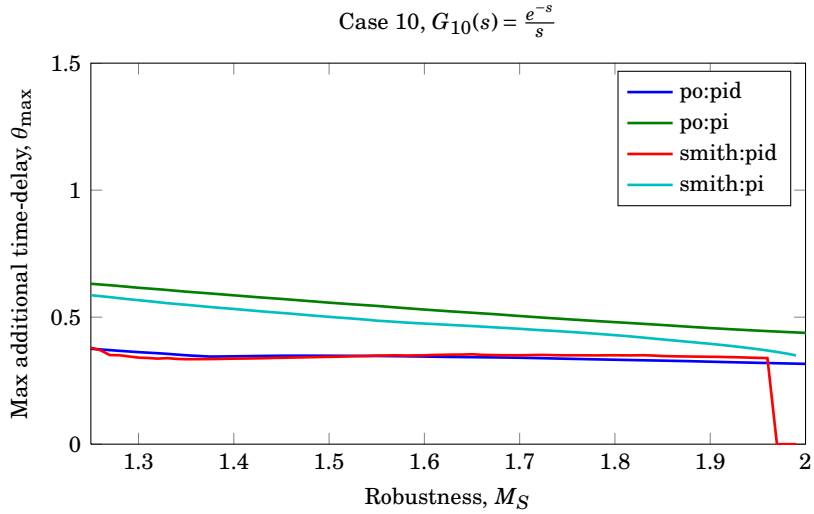


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.9 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$.

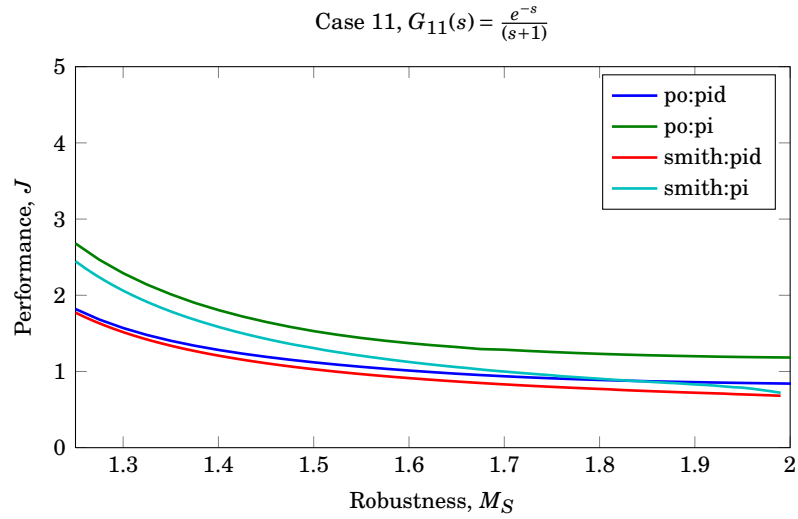


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

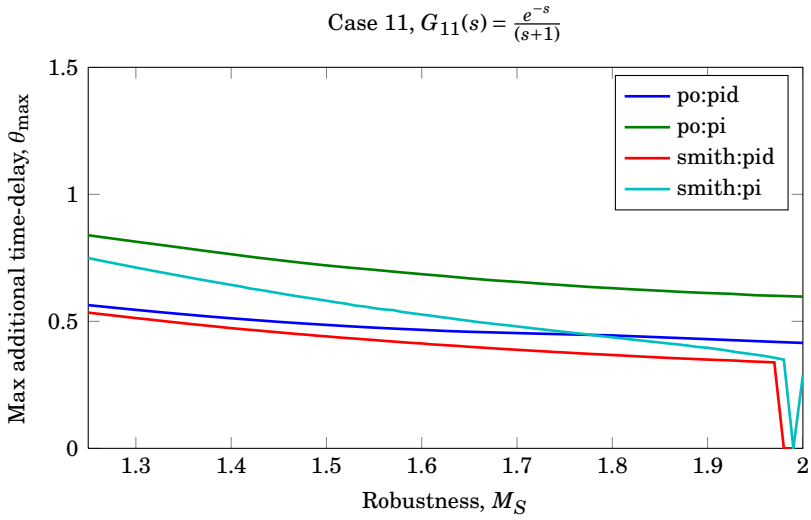


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.10 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_{10}(s) = \frac{e^{-s}}{s}$.

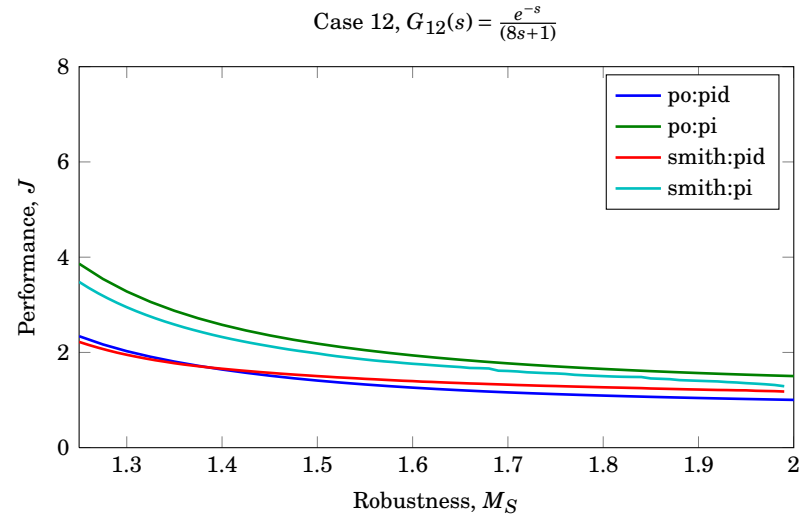


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

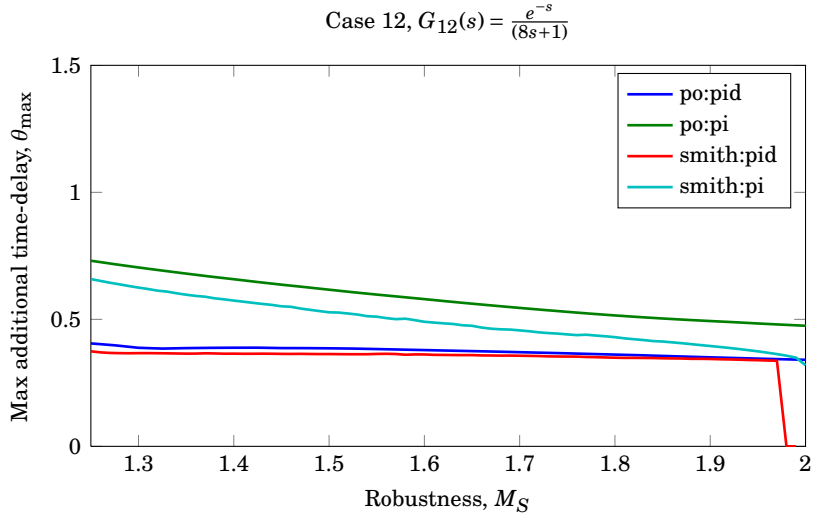


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.11 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_{11}(s) = \frac{e^{-s}}{(s+1)}$.

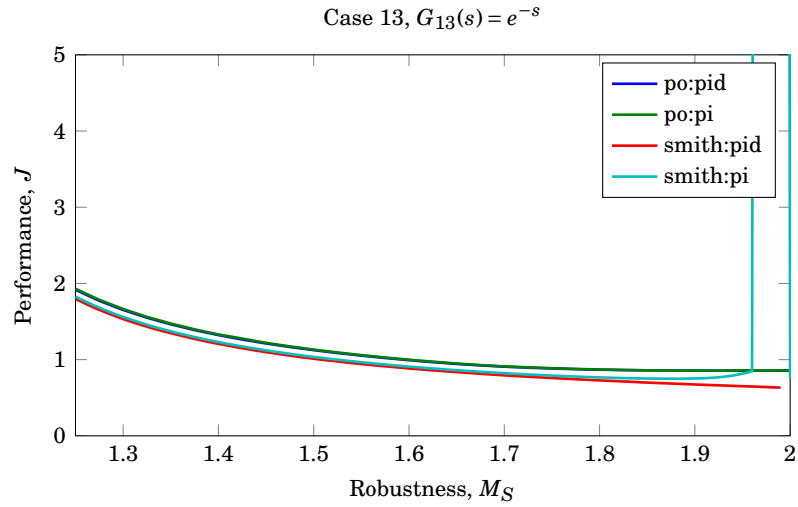


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

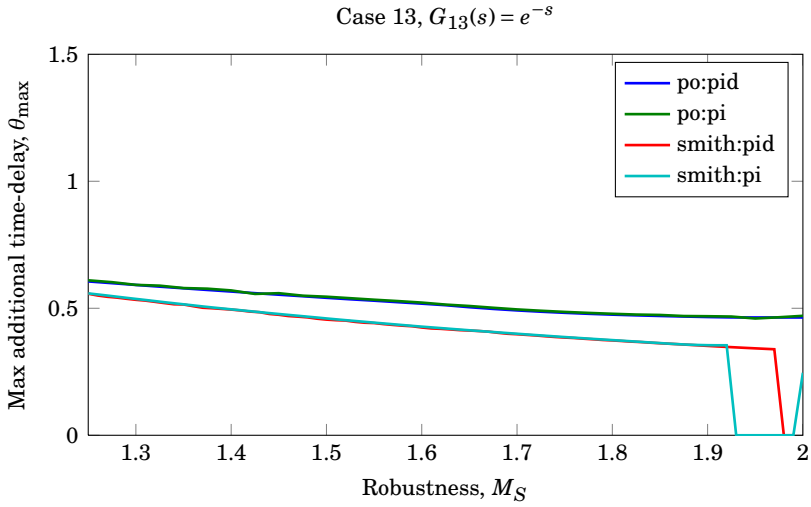


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.12 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_{12}(s) = \frac{e^{-s}}{(8s+1)}$.

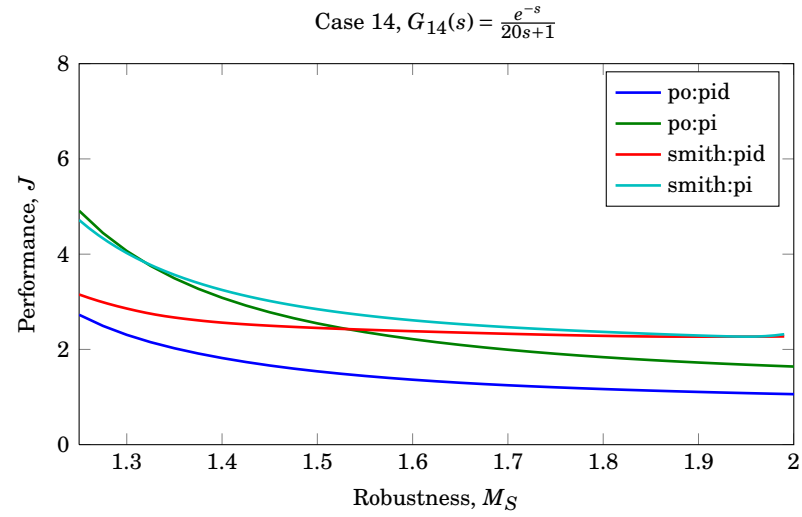


(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$

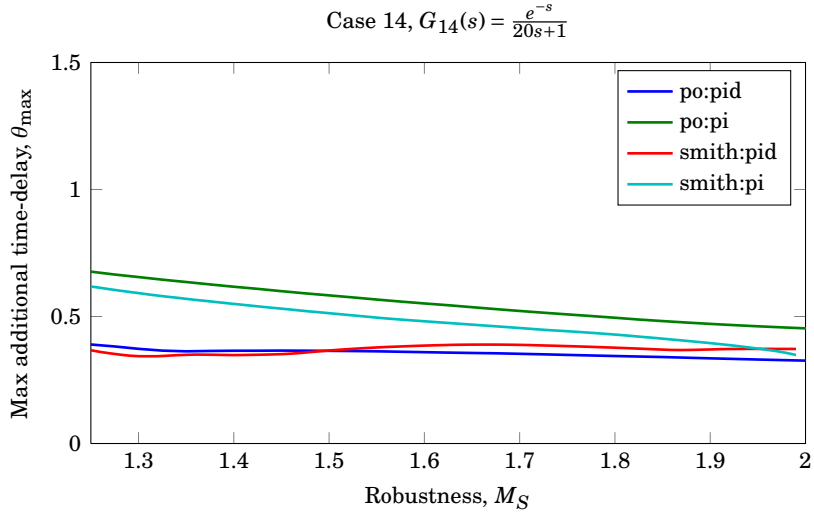


(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.13 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_{13}(s) = e^{-s}$.



(a) Pareto optimal Smith predictor solutions, $J = f(M_S)$



(b) Maximum additional time-delay for Pareto optimal Smith predictor controllers, $\theta_{\max} = f(M_S)$

Figure D.14 – Pareto optimal Smith predictor solutions for PI and PID controllers compared to Pareto optimal PI and PID controller. Also, the maximum additional time-delay ($\theta_{\max}$) before instability occur. Both are plots for the process $G_{14}(s) = \frac{e^{-s}}{20s+1}$.

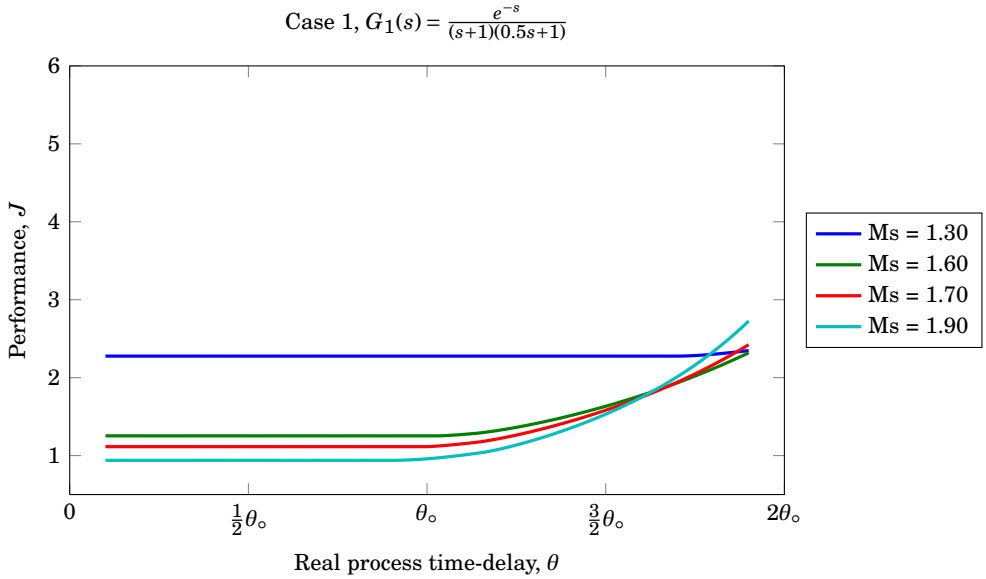
APPENDIX E

SMITH PREDICTOR SENSITIVITY TO TIME-DELAY MODELLING ERROR

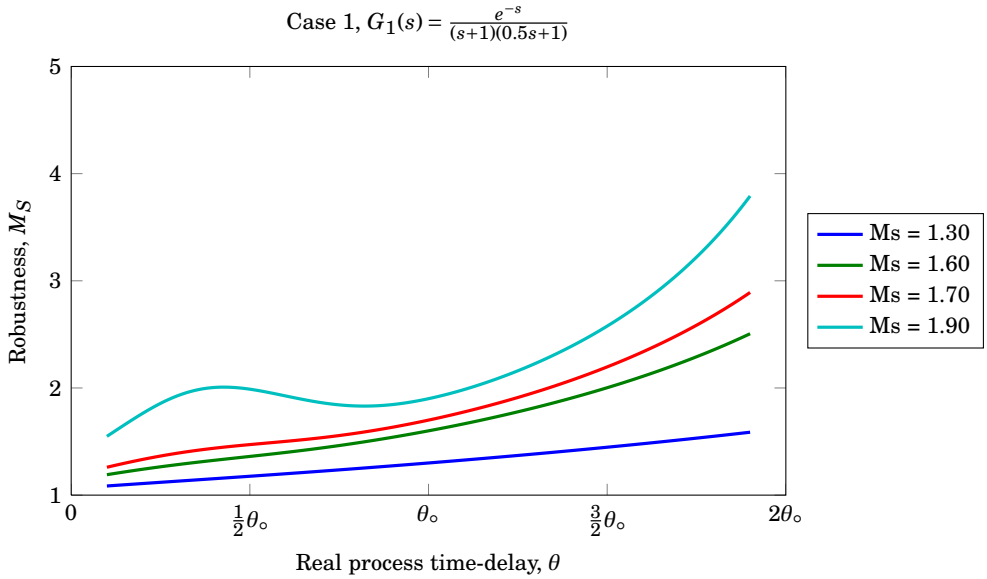
The Pareto optimal Smith predictor PI and PID controllers for the process models in cases 1–14 have been evaluated when the real process time-delay (θ) is different from the nominal modelled time-delay ($\theta_\circ$). The real process time-delay have been changed within $\pm 90\%$ of the nominal time-delay. Performance and robustness efficiency have been plotted as functions of the real time-delay.

E.1 Smith Predictor PI Control Sensitivity

The sensitivity plots for the Pareto optimal Smith predictor PI controller tunings are given in terms of performance in Figure E.1(a) through E.14(a). The corresponding robustness efficiency plots are displayed in Figure E.1(b) through E.1(b).

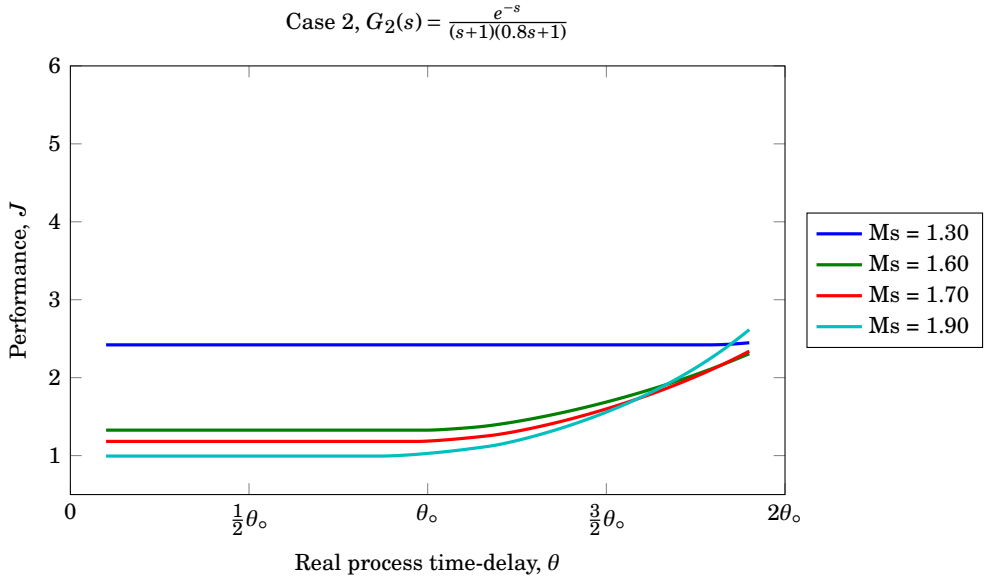


(a) Performance sensitivity, $J = f(\theta)$.

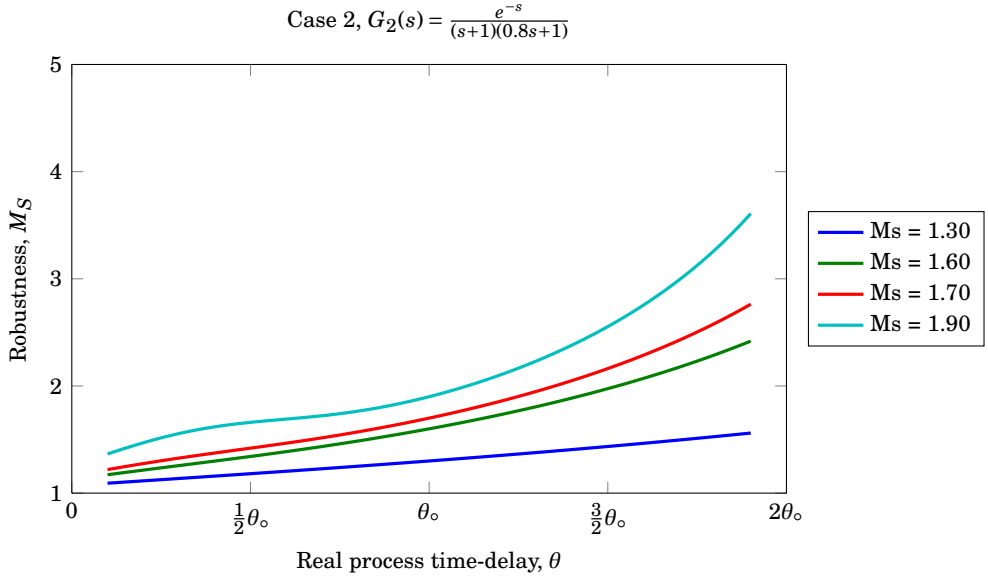


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.1 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$.

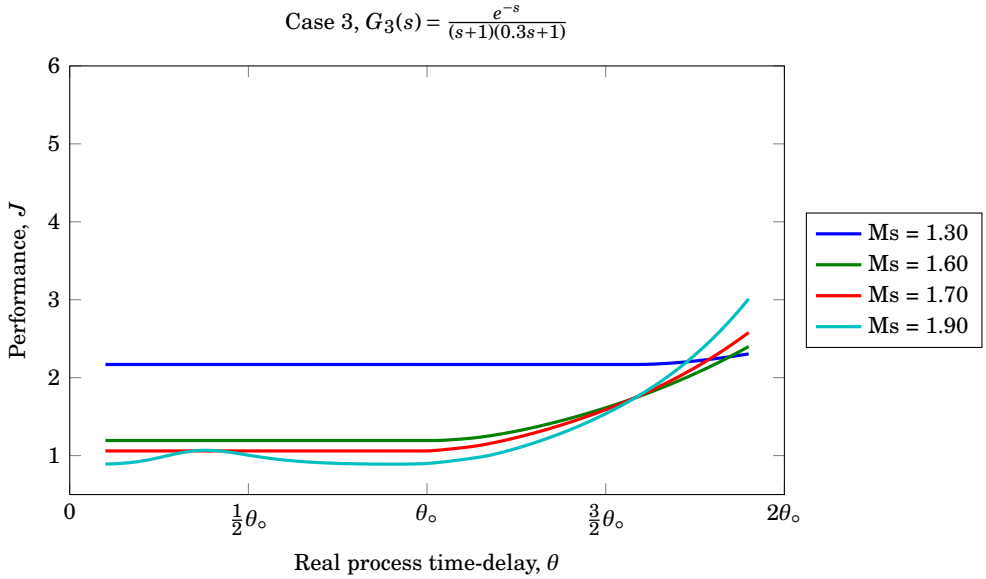


(a) Performance sensitivity, $J = f(\theta)$.

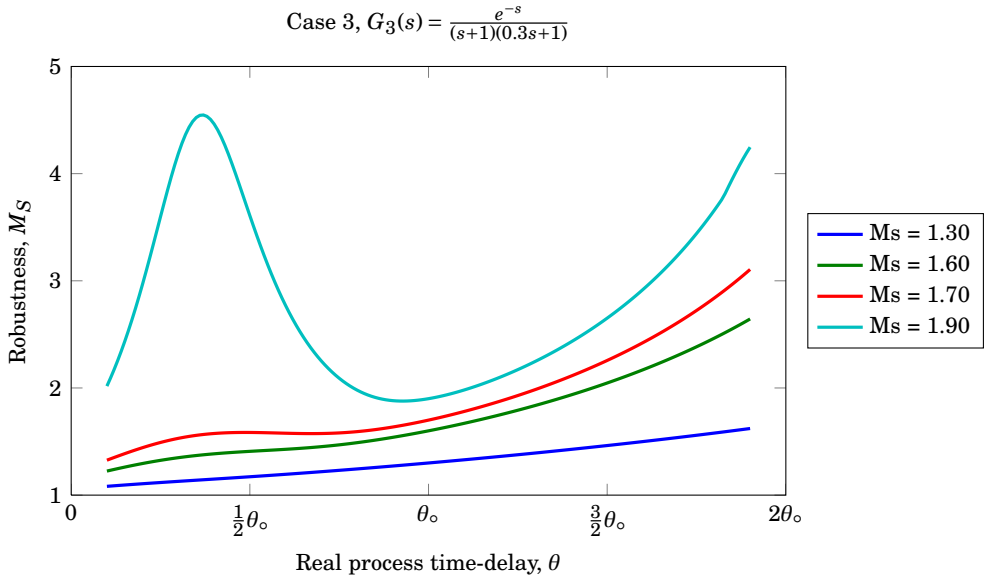


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.2 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$.

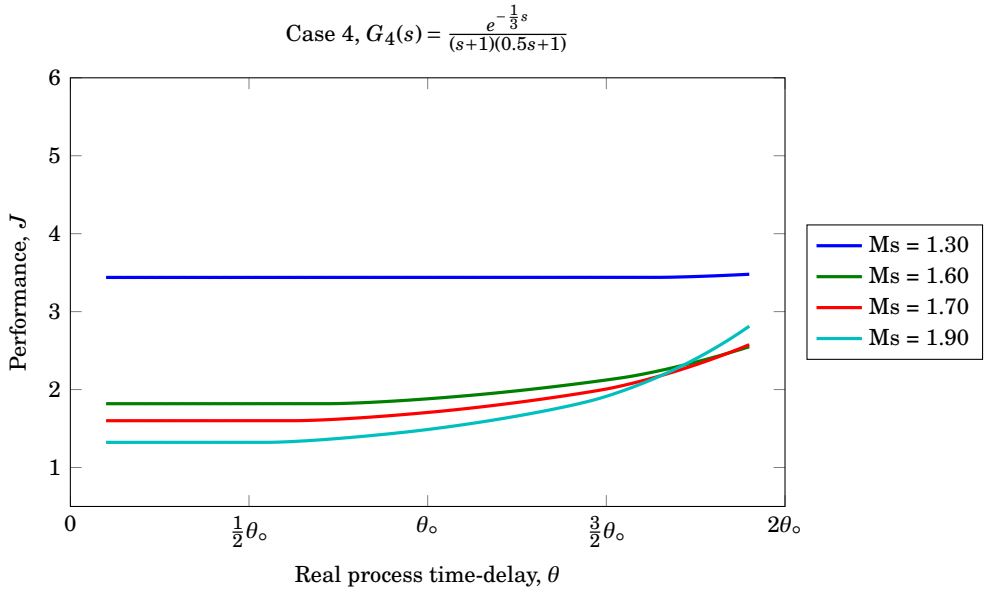


(a) Performance sensitivity, $J = f(\theta)$.

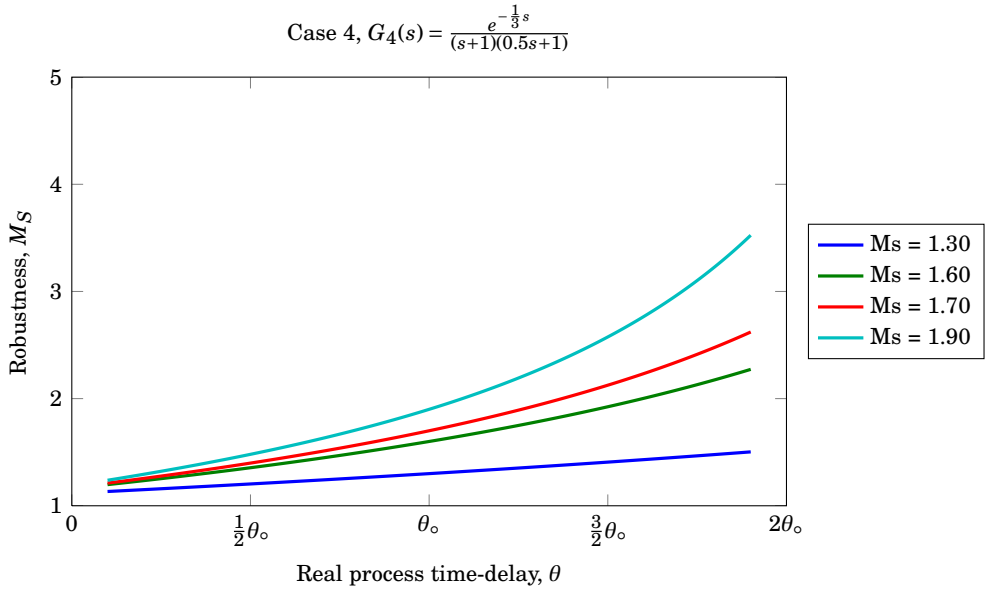


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.3 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$.

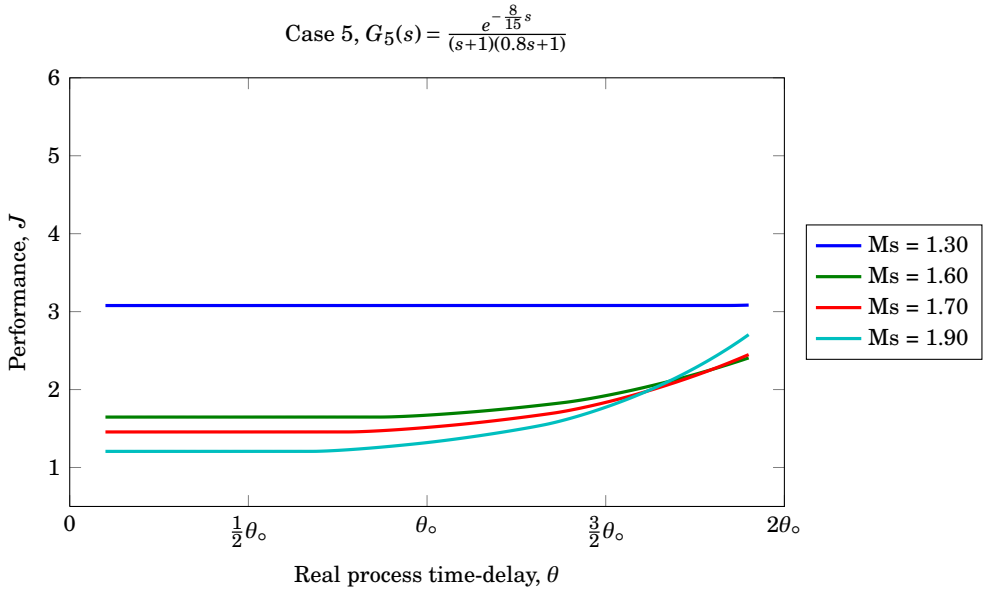


(a) Performance sensitivity, $J = f(\theta)$.

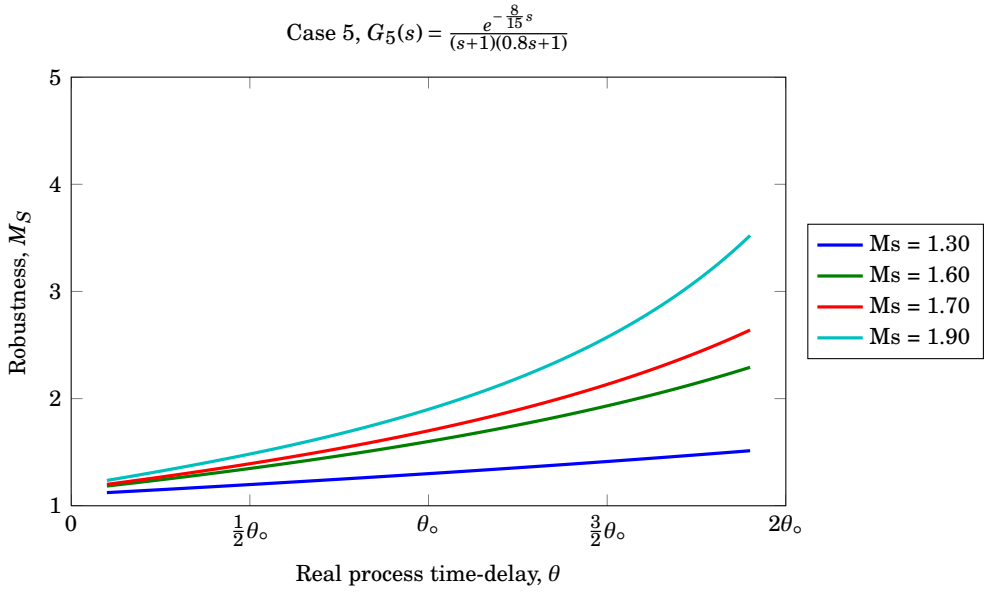


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.4 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$.

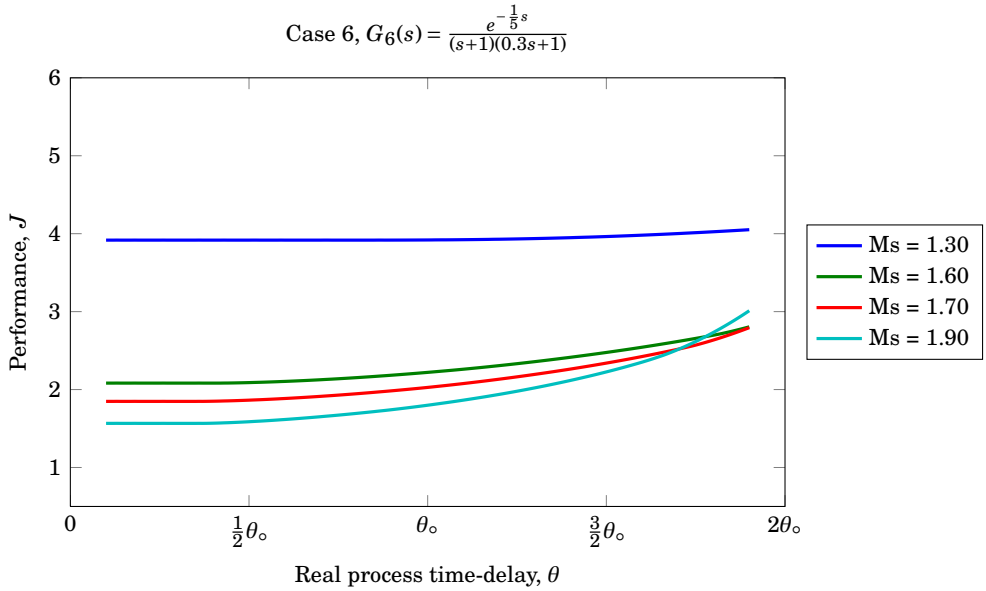


(a) Performance sensitivity, $J = f(\theta)$.

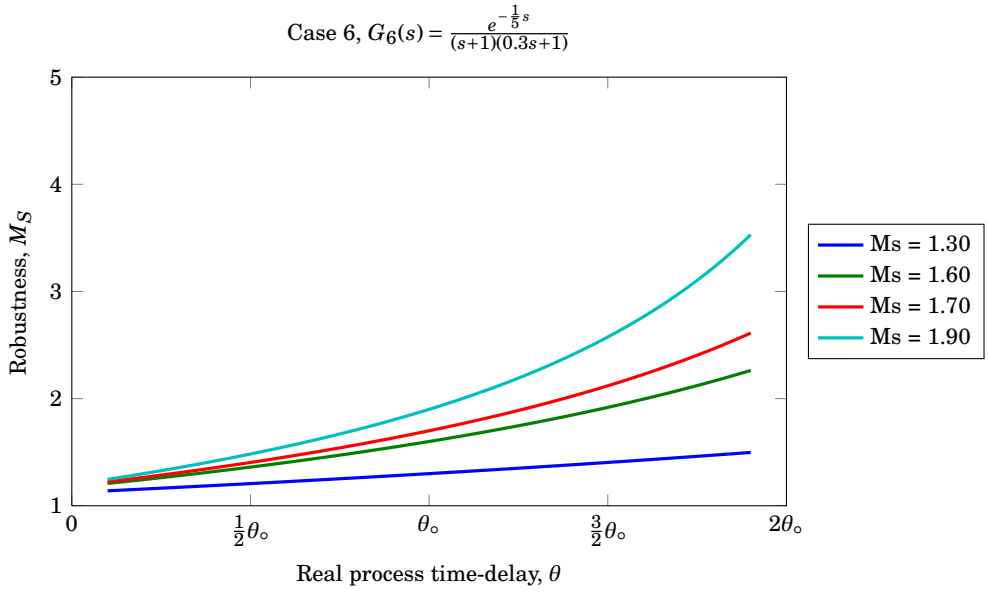


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.5 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$.

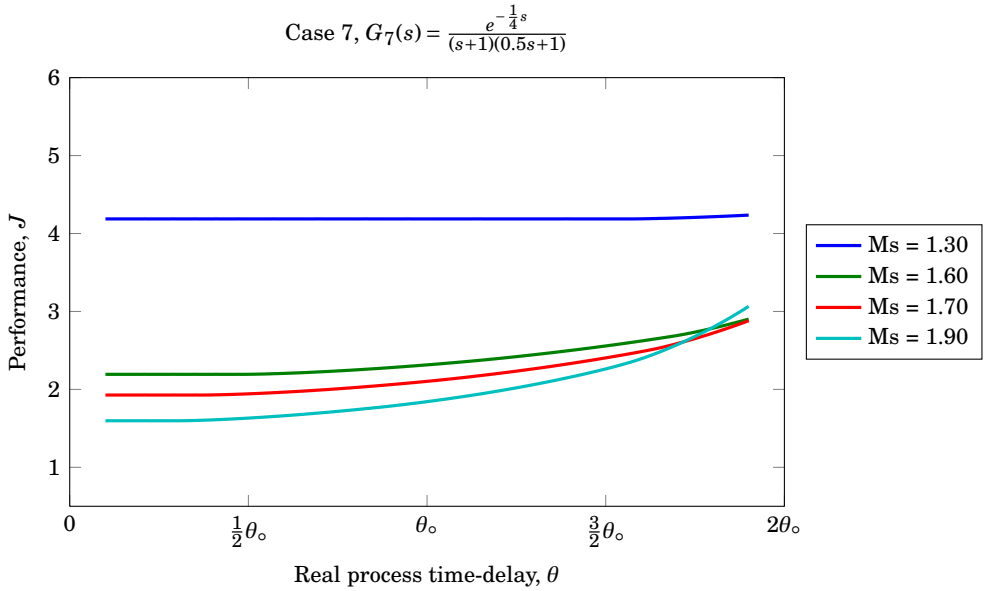


(a) Performance sensitivity, $J = f(\theta)$.

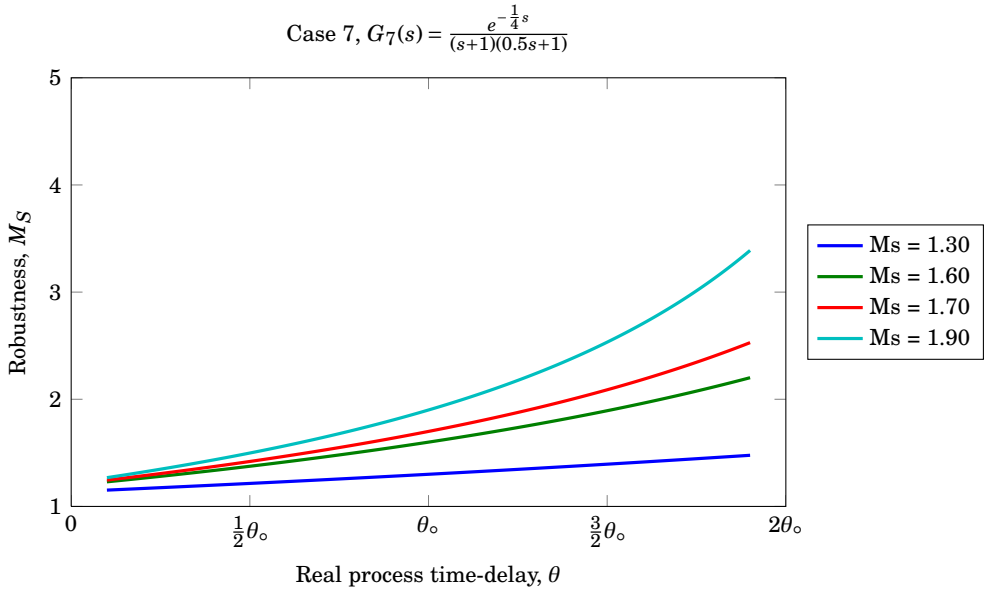


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.6 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$.

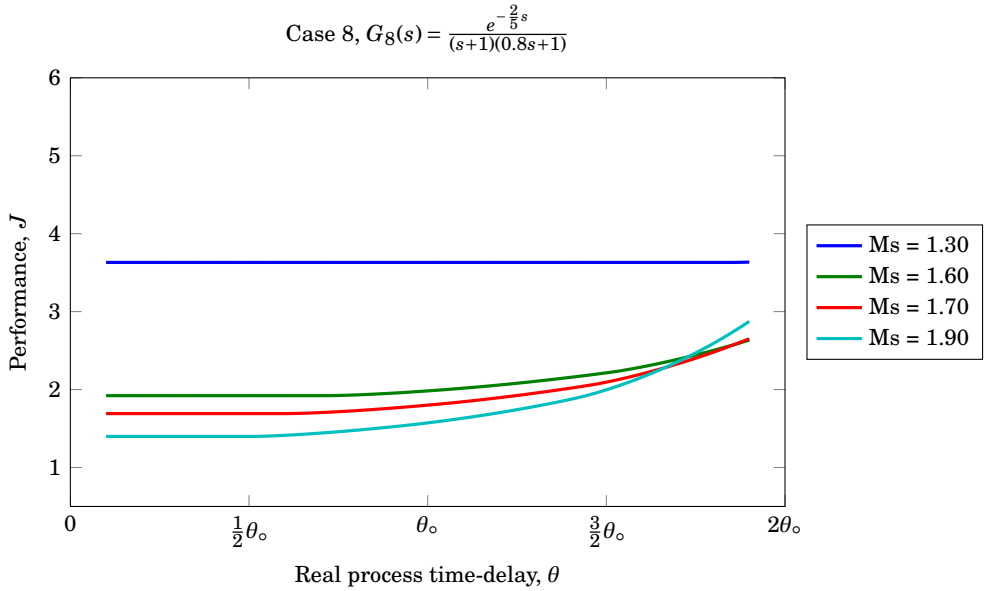


(a) Performance sensitivity, $J = f(\theta)$.

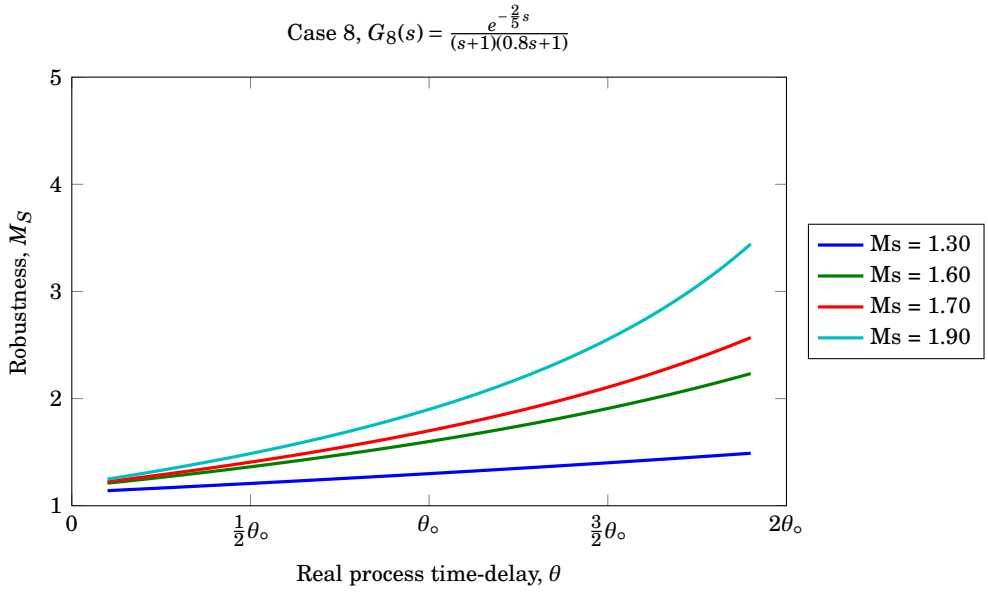


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.7 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$.

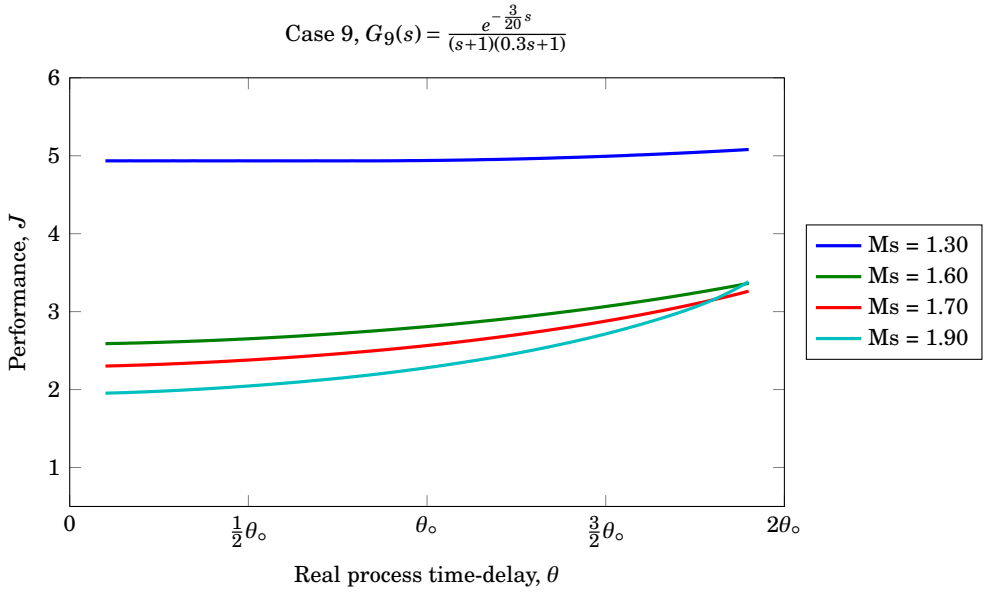


(a) Performance sensitivity, $J = f(\theta)$.

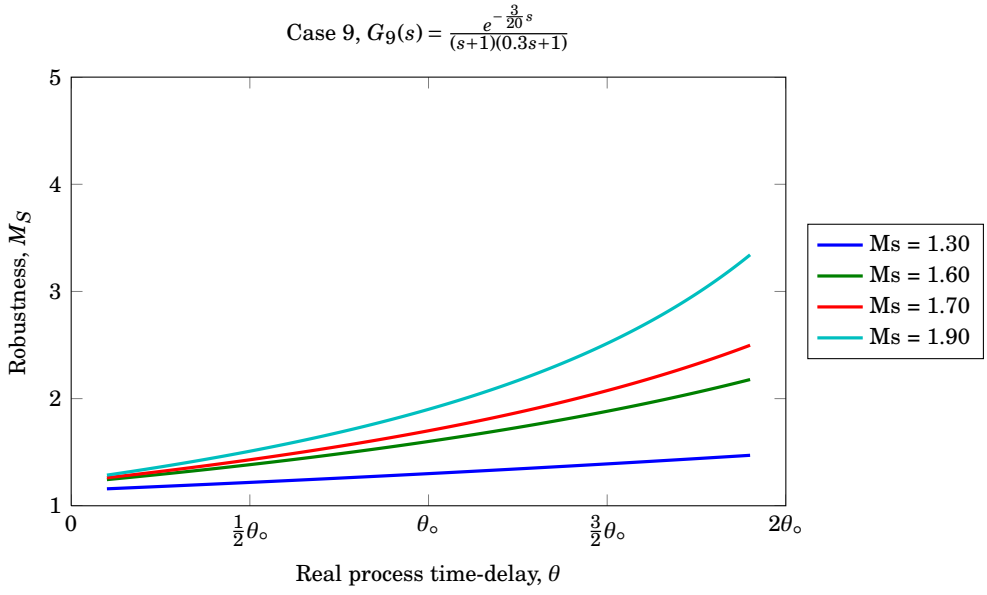


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.8 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$.

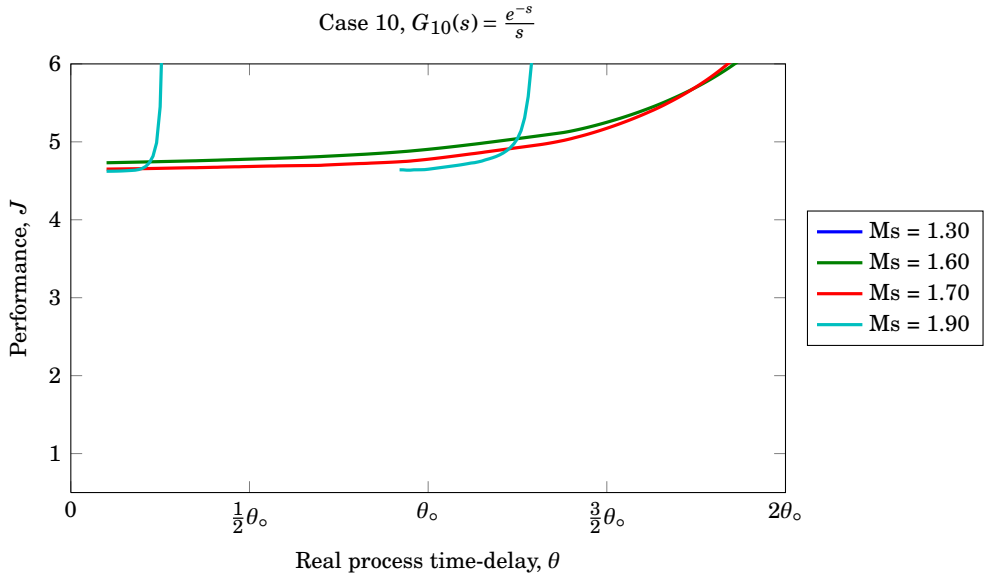


(a) Performance sensitivity, $J = f(\theta)$.

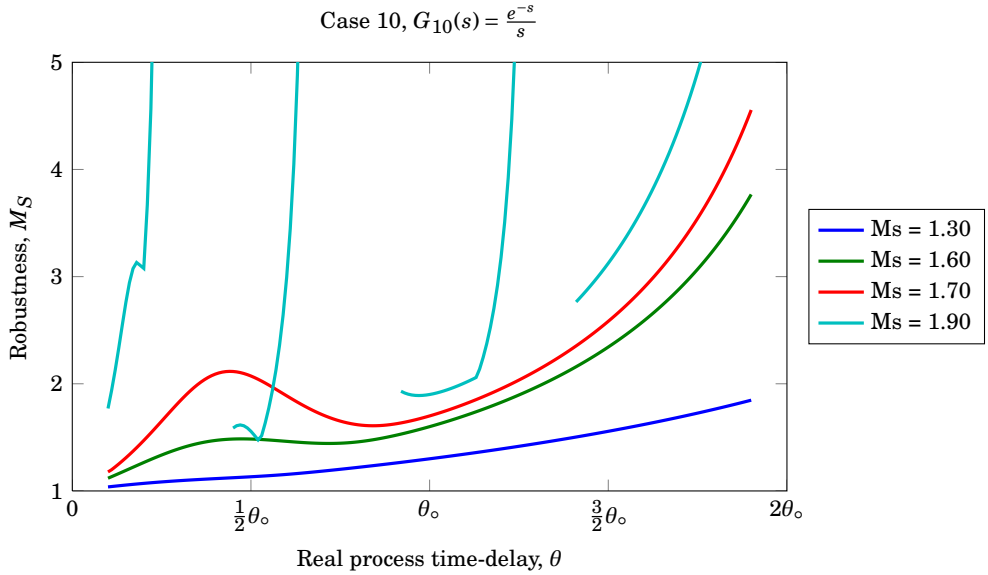


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.9 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$.

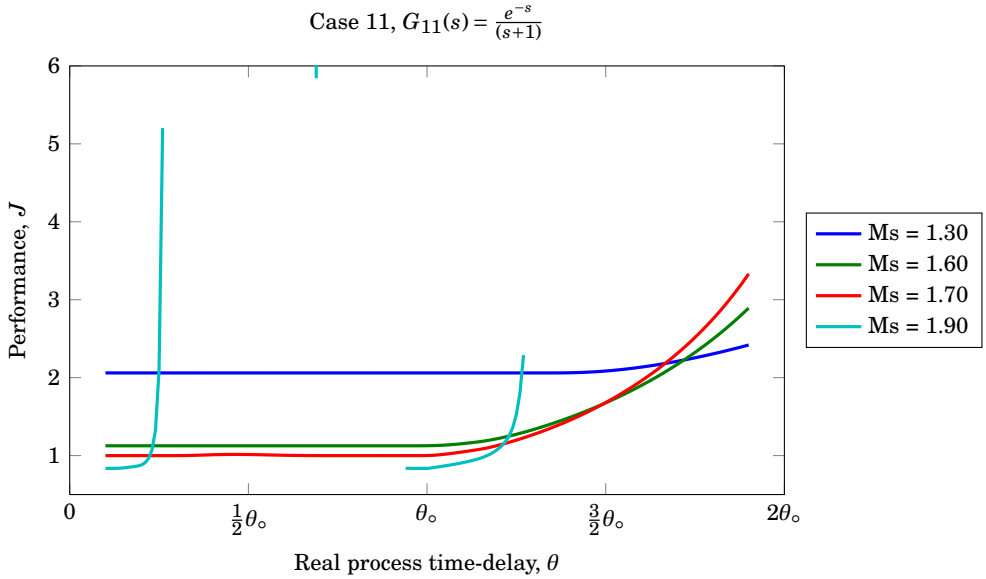


(a) Performance sensitivity, $J = f(\theta)$.

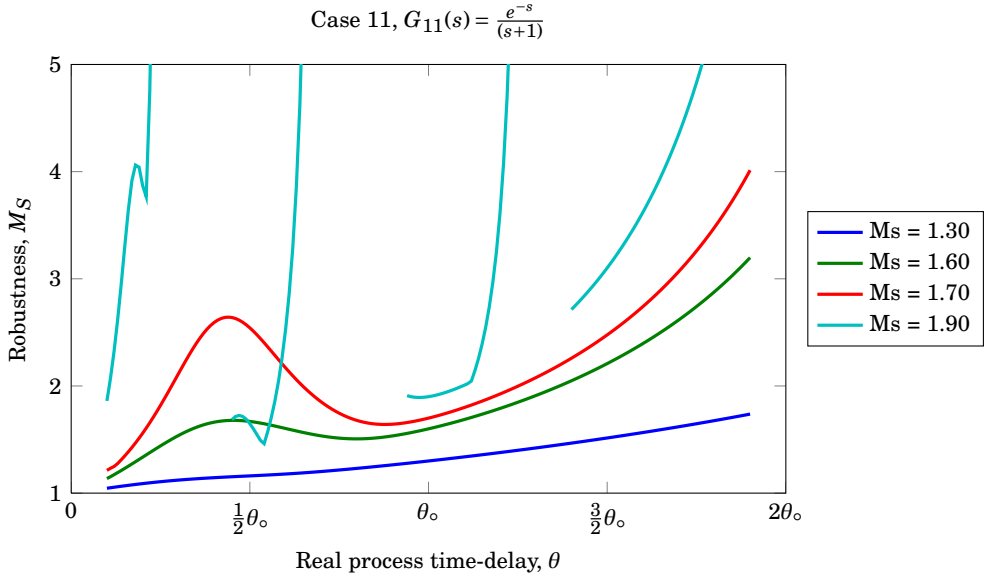


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.10 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_{10}(s) = \frac{e^{-s}}{s}$.

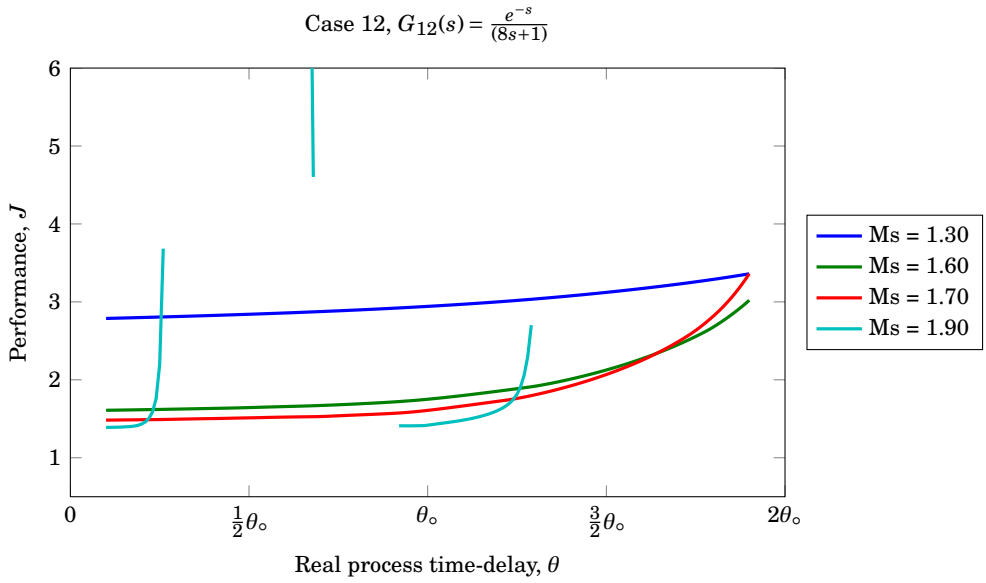


(a) Performance sensitivity, $J = f(\theta)$.

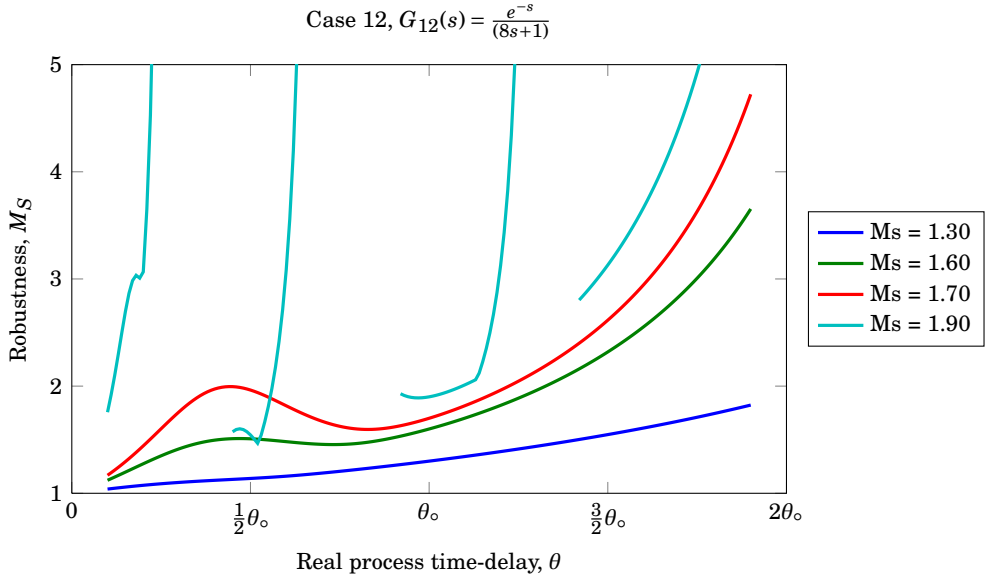


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.11 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_{11}(s) = \frac{e^{-s}}{(s+1)}$.

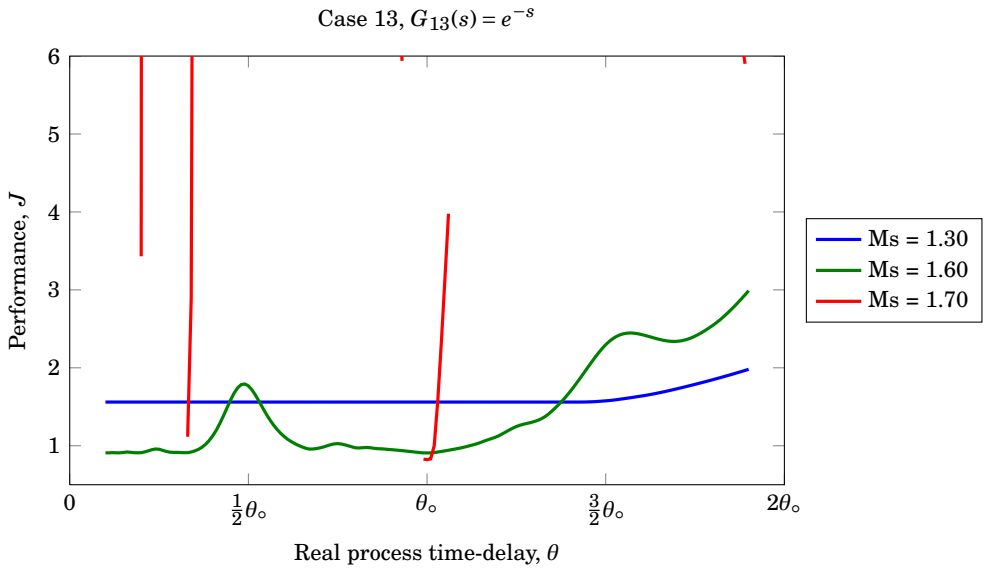


(a) Performance sensitivity, $J = f(\theta)$.

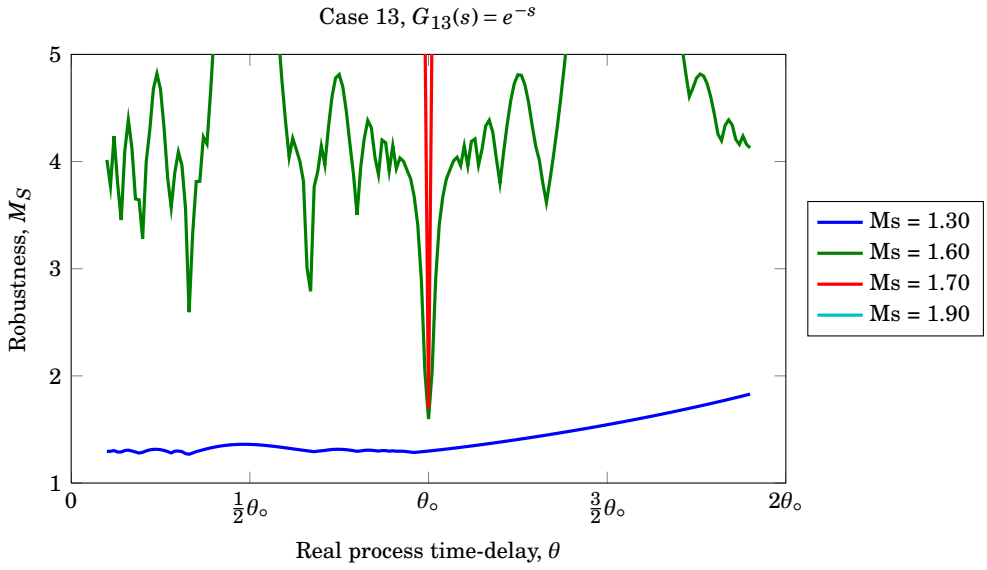


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.12 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_{12}(s) = e^{-s}$.

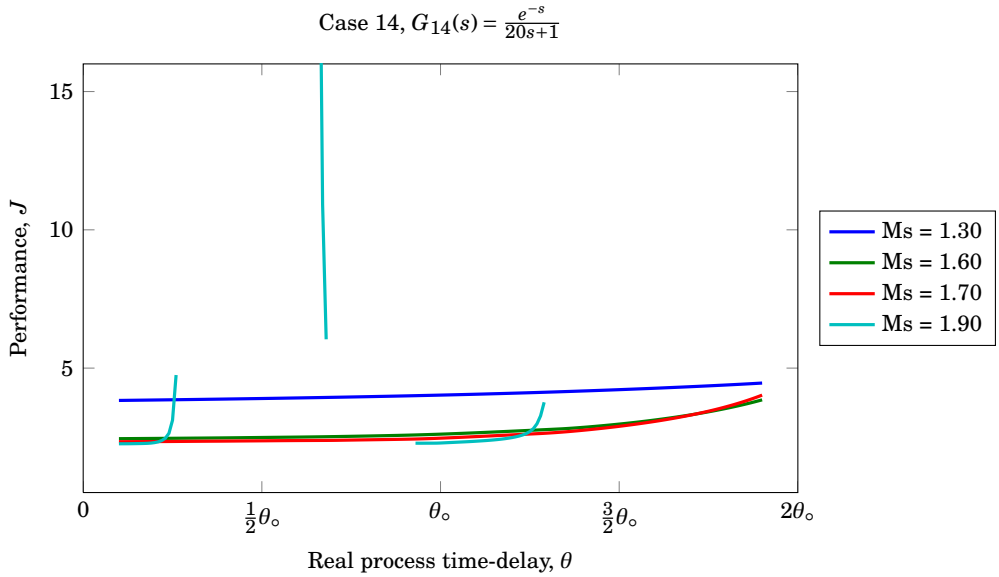


(a) Performance sensitivity, $J = f(\theta)$.

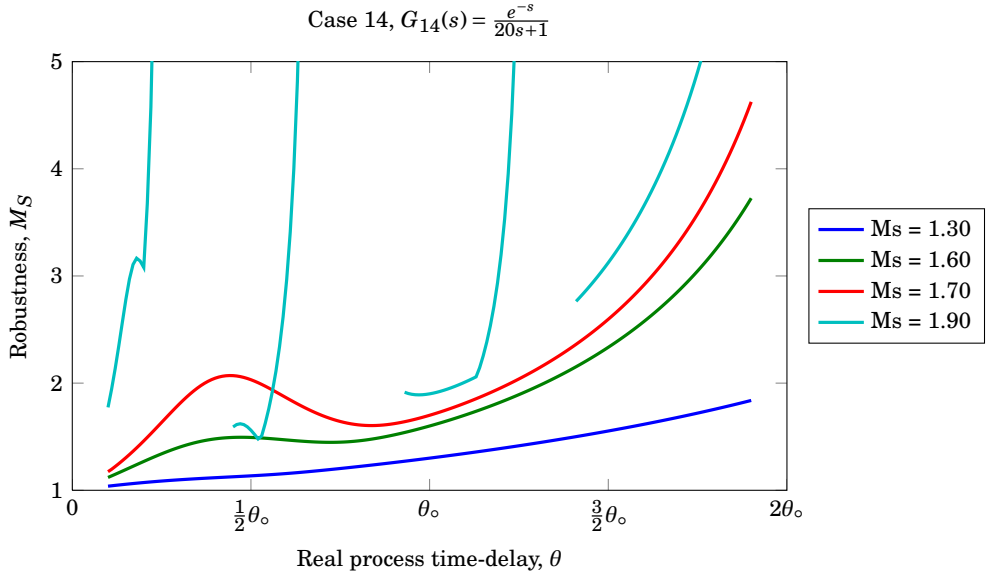


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.13 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_{13}(s) = e^{-s}$.



(a) Performance sensitivity, $J = f(\theta)$.

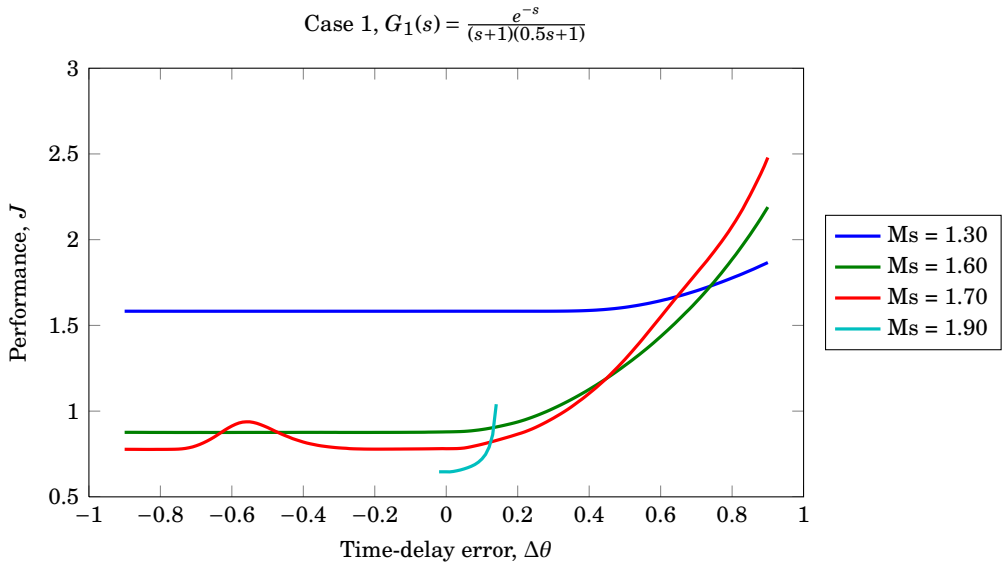


(b) Robustness sensitivity, $M_S = f(\theta)$.

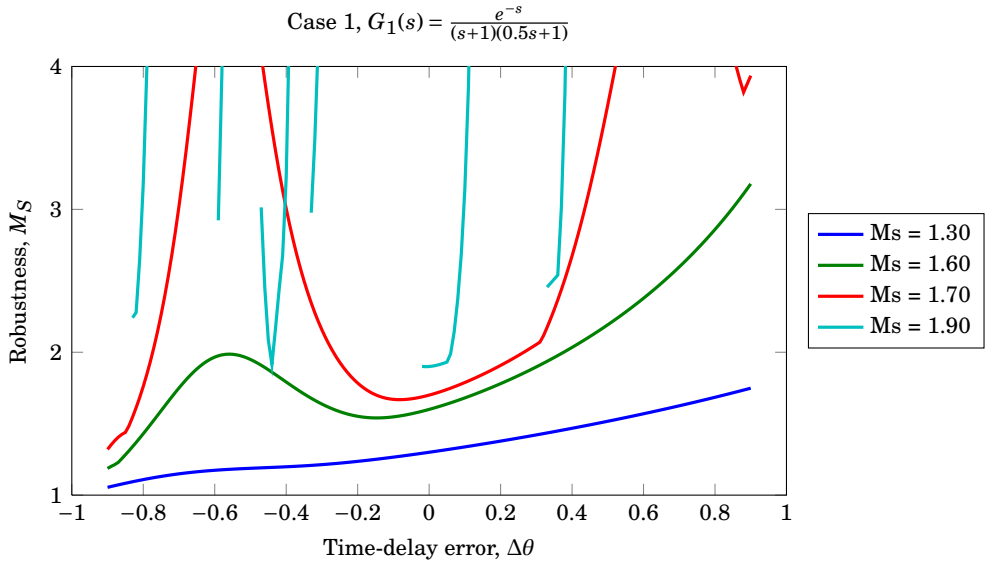
Figure E.14 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PI controller for a set of target M_S values for model $G_{14}(s) = \frac{e^{-s}}{20s+1}$.

E.2 Smith Predictor PID Control Sensitivity

The sensitivity plots for the Pareto optimal Smith predictor PI controller tunings are given in terms of performance in Figure E.1(a) through E.14(a). The corresponding robustness efficiency plots are displayed in Figure E.1(b) through E.1(b).

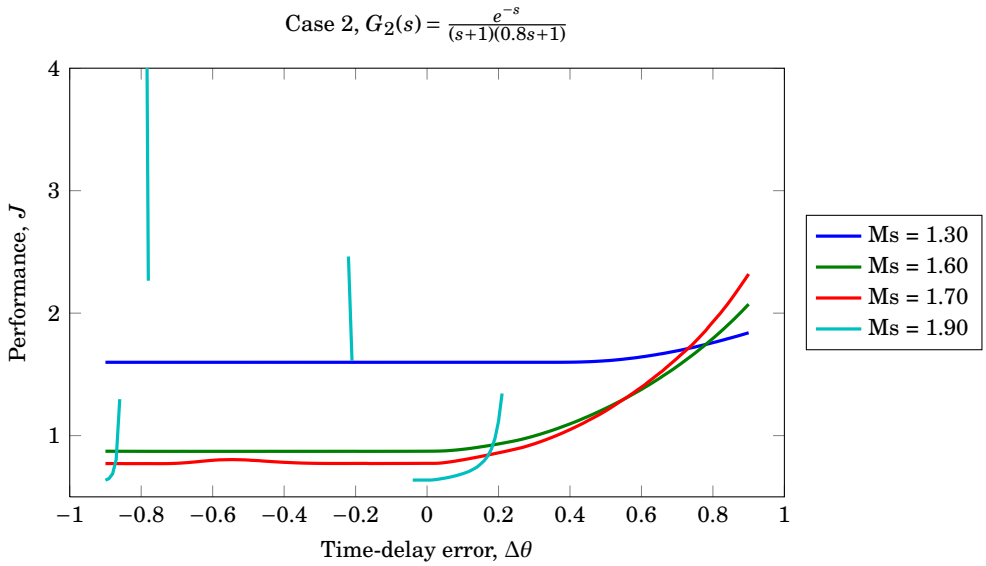


(a) Performance sensitivity, $J = f(\theta)$.

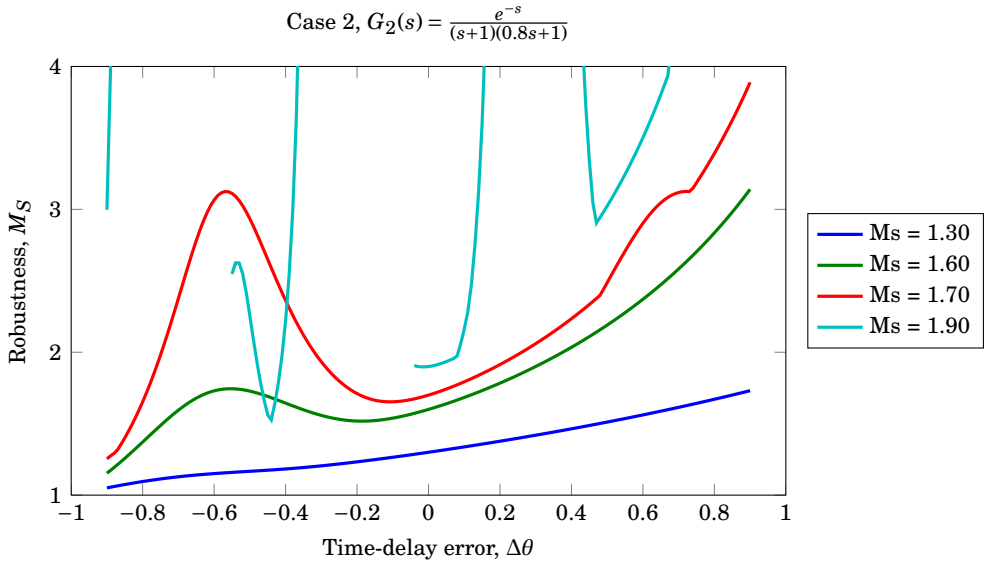


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.15 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$.

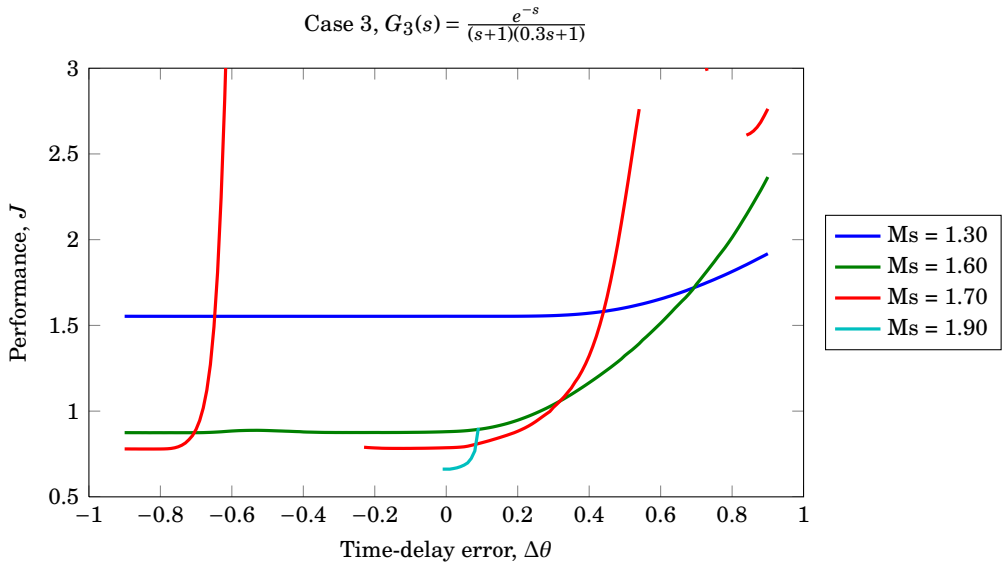


(a) Performance sensitivity, $J = f(\theta)$.

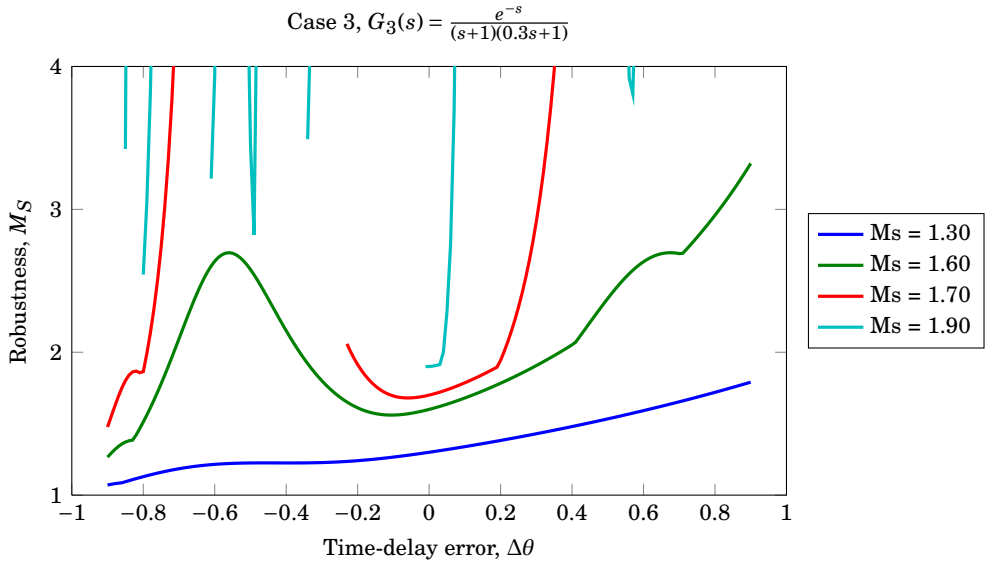


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.16 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$.

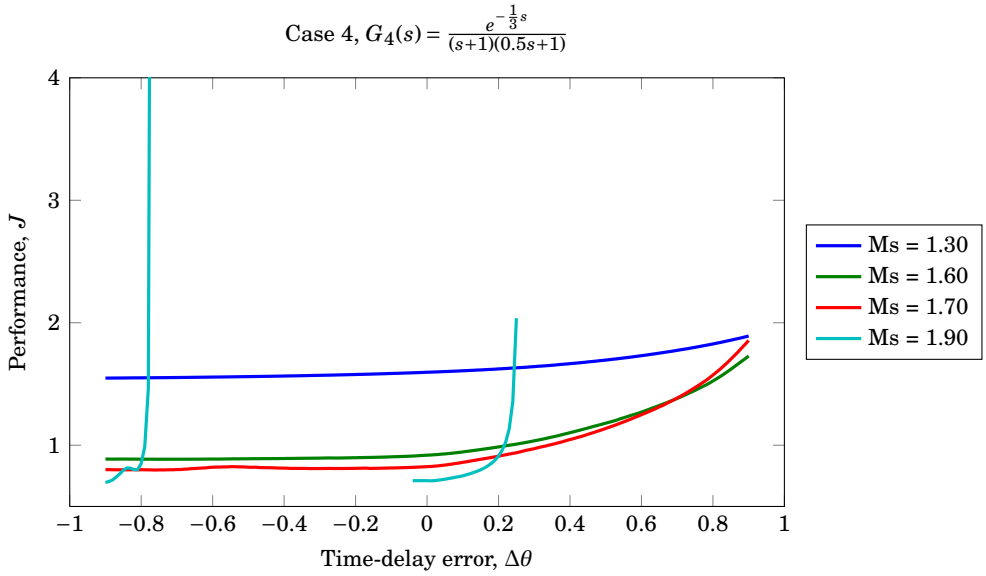


(a) Performance sensitivity, $J = f(\theta)$.

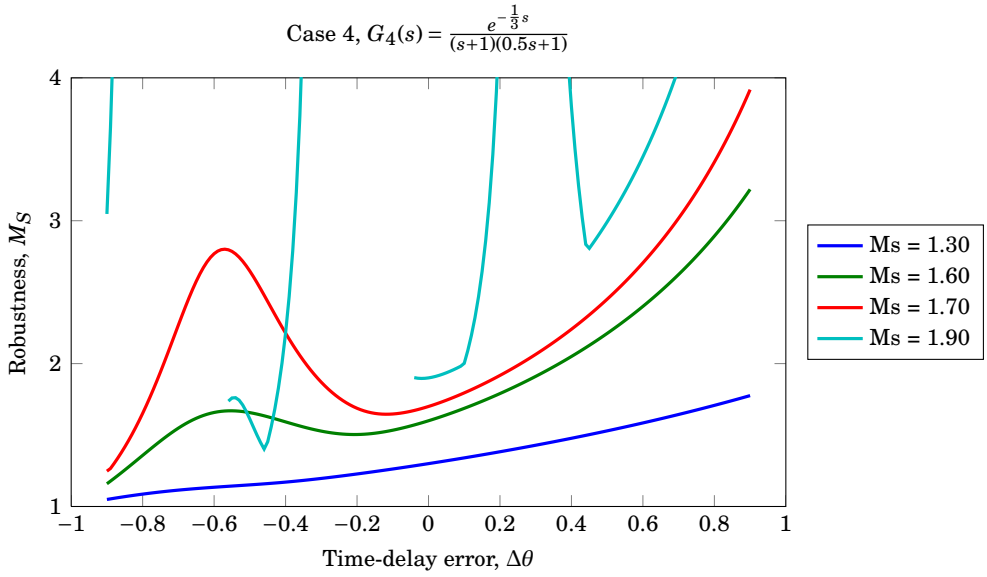


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.17 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$.

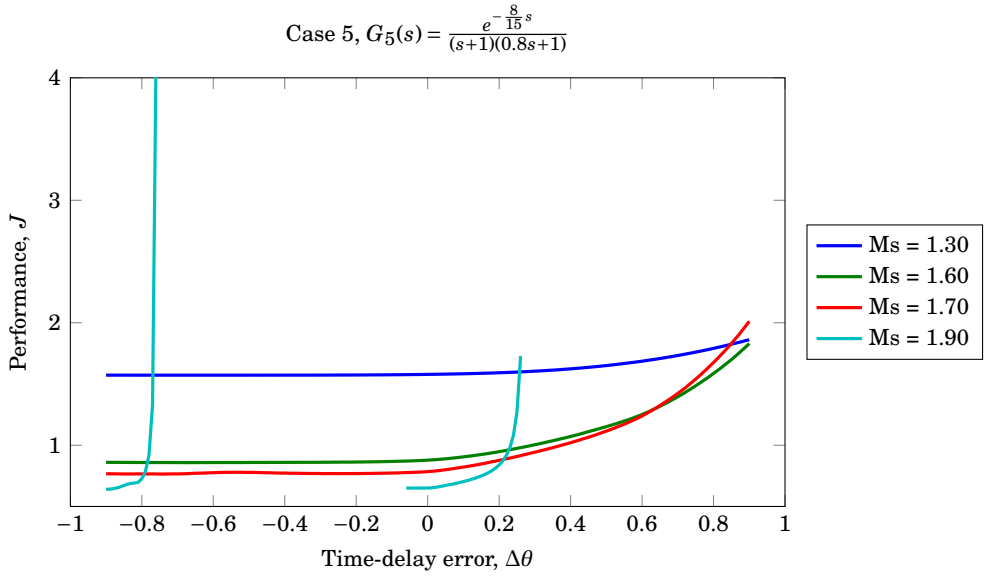


(a) Performance sensitivity, $J = f(\theta)$.

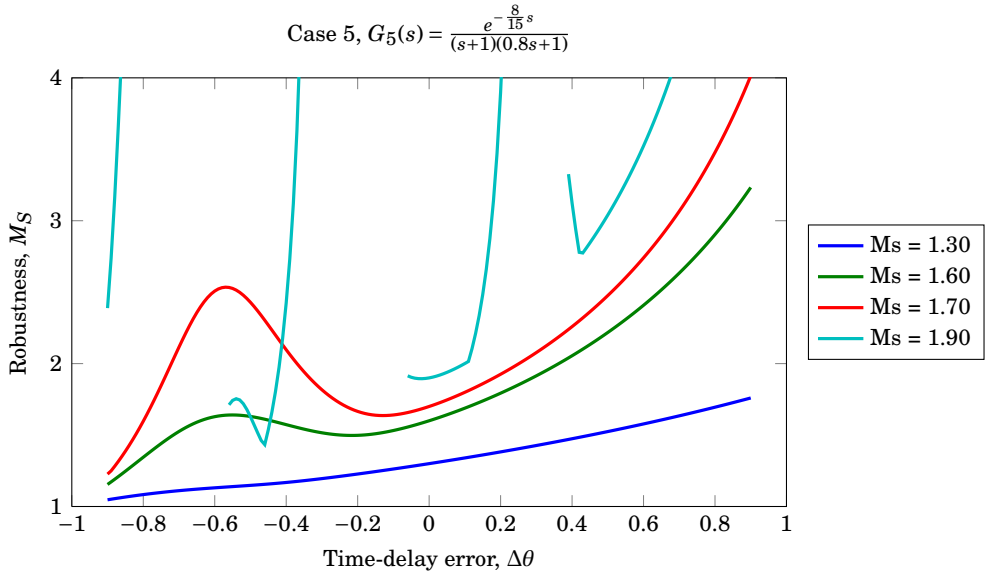


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.18 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$.

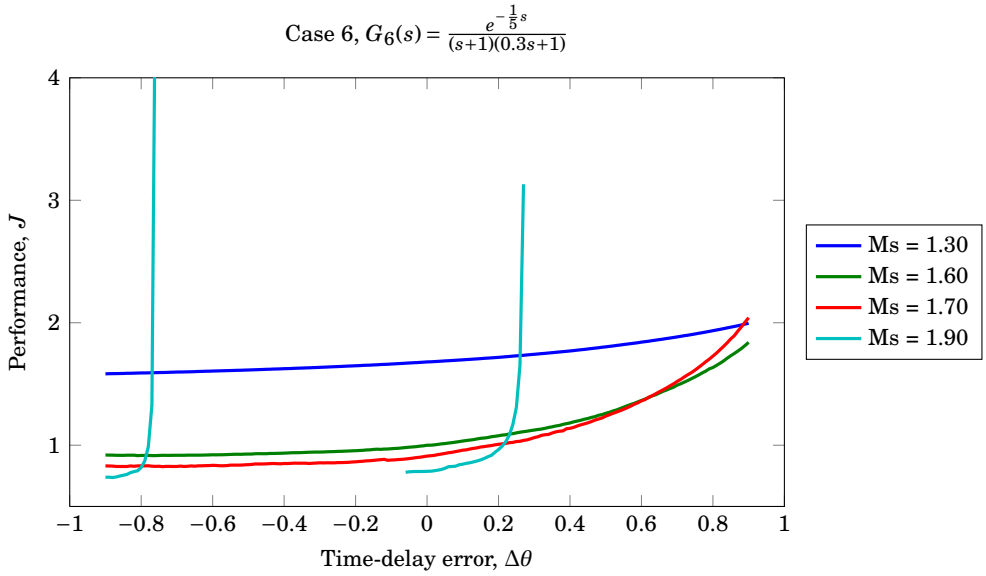


(a) Performance sensitivity, $J = f(\theta)$.

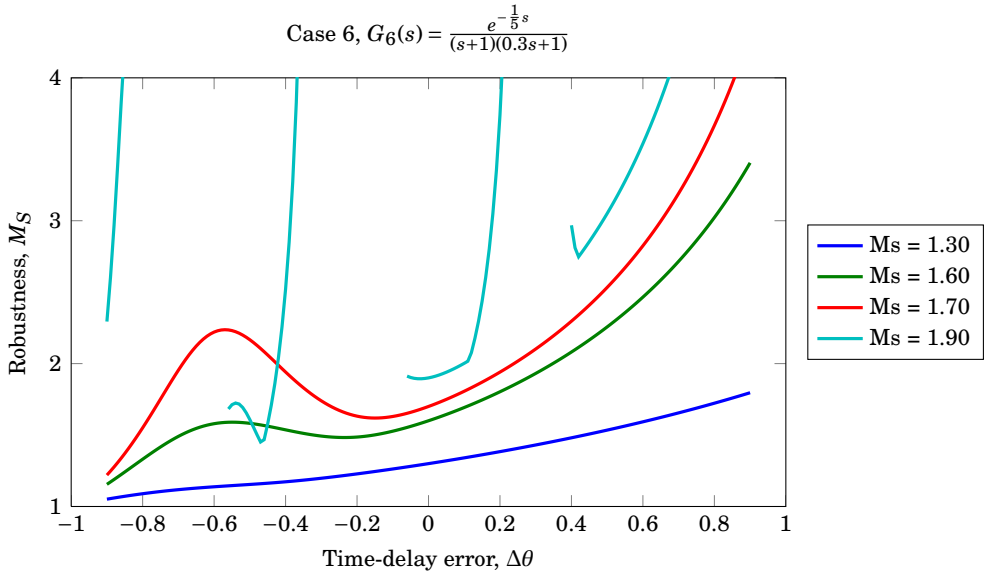


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.19 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$.

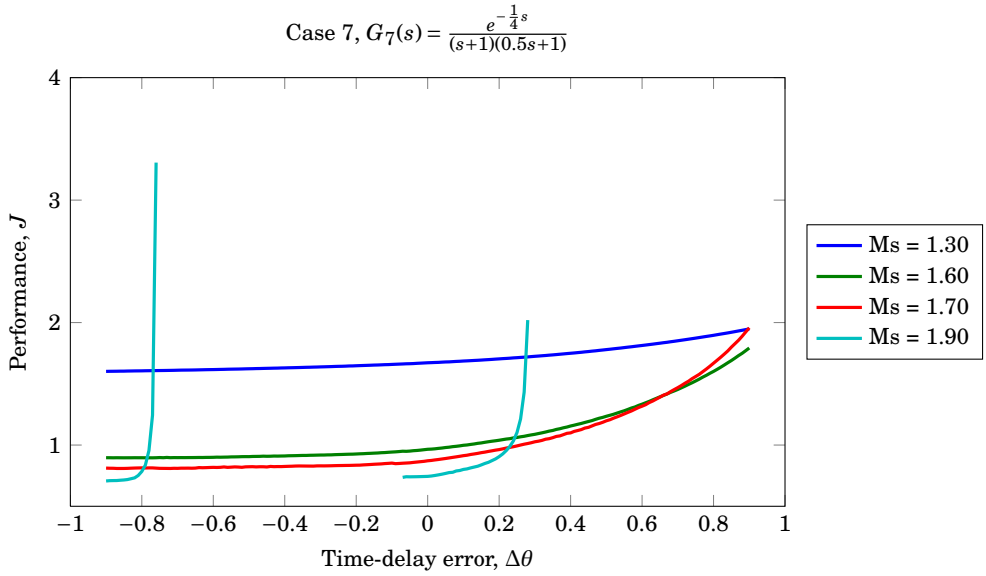


(a) Performance sensitivity, $J = f(\theta)$.

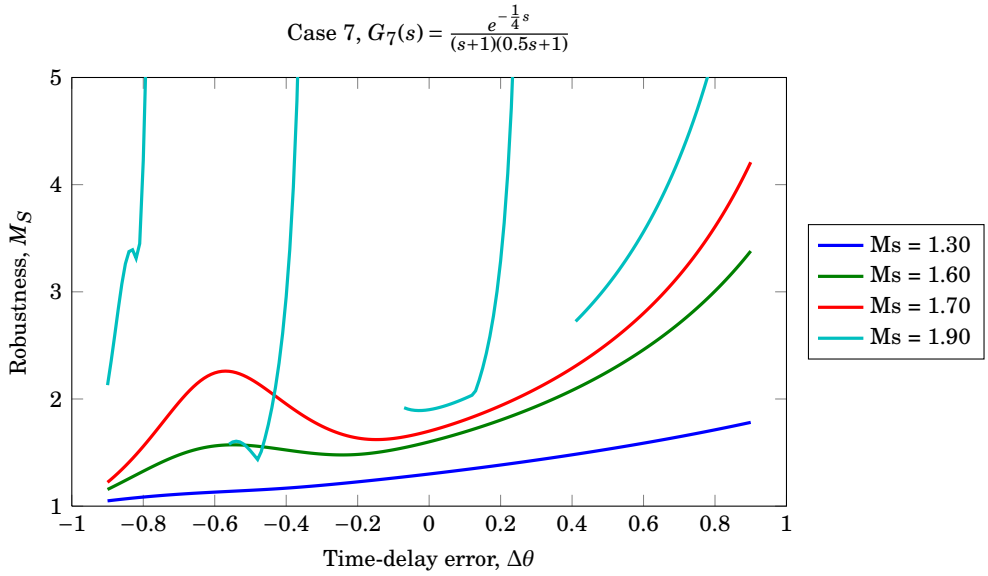


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.20 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$.

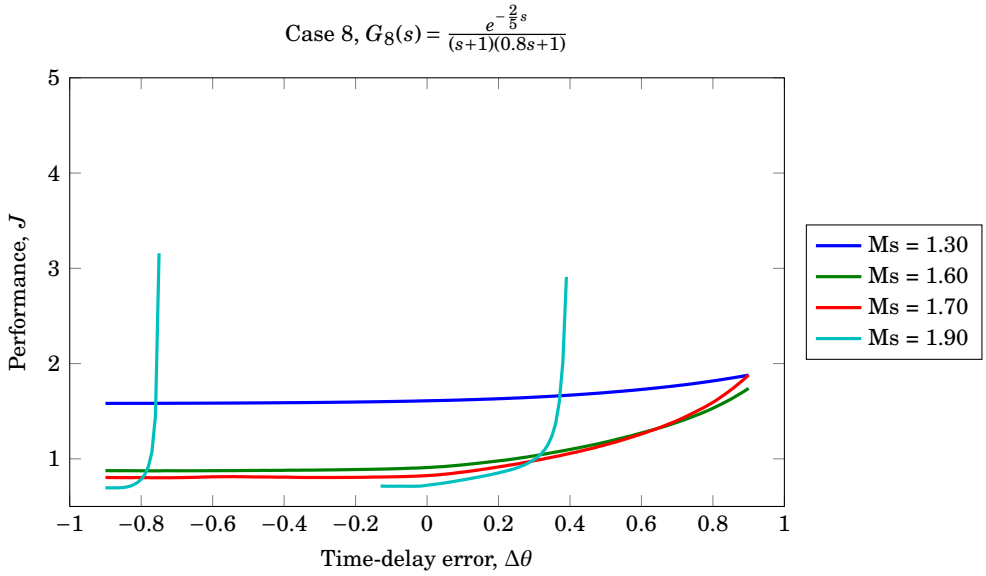


(a) Performance sensitivity, $J = f(\theta)$.

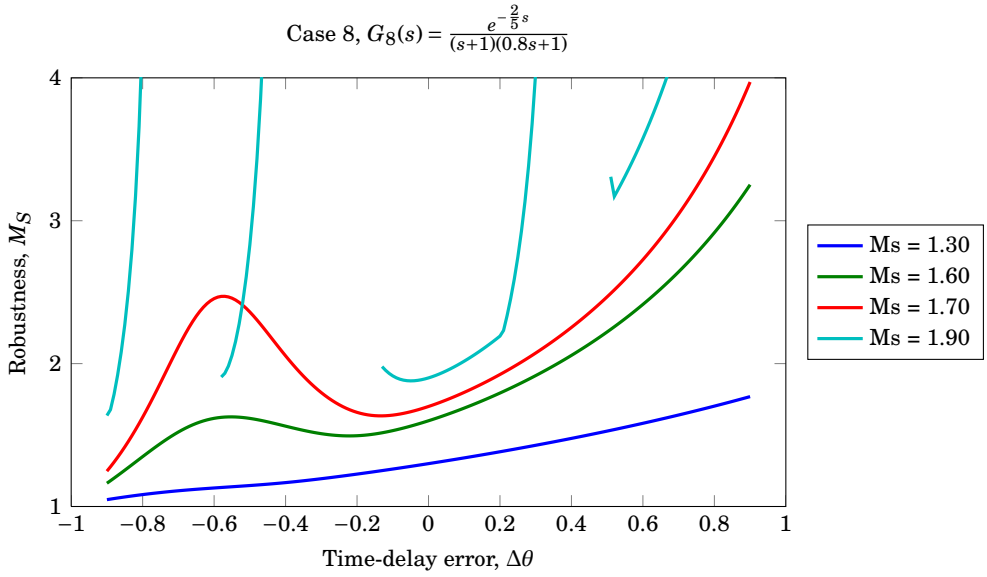


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.21 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$.

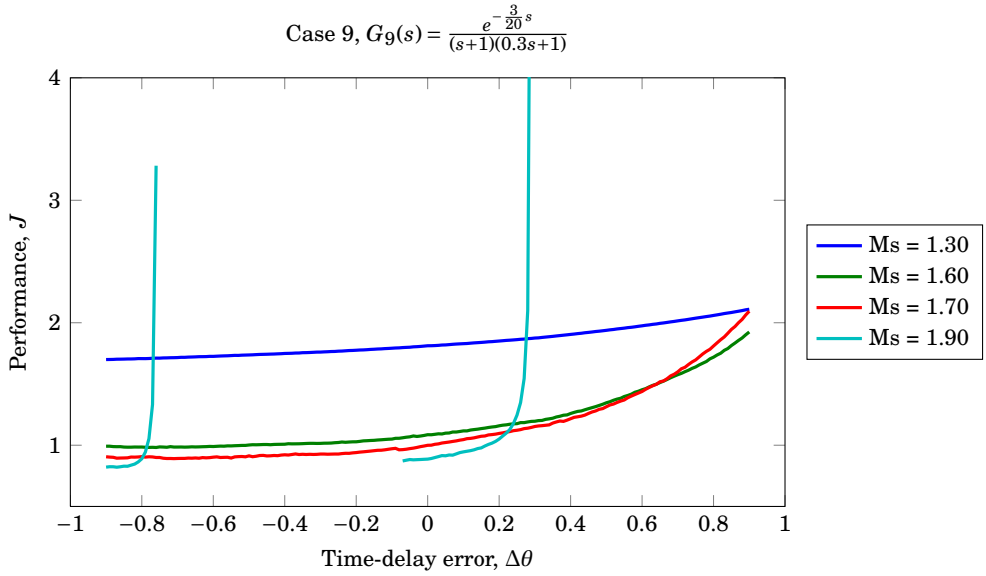


(a) Performance sensitivity, $J = f(\theta)$.

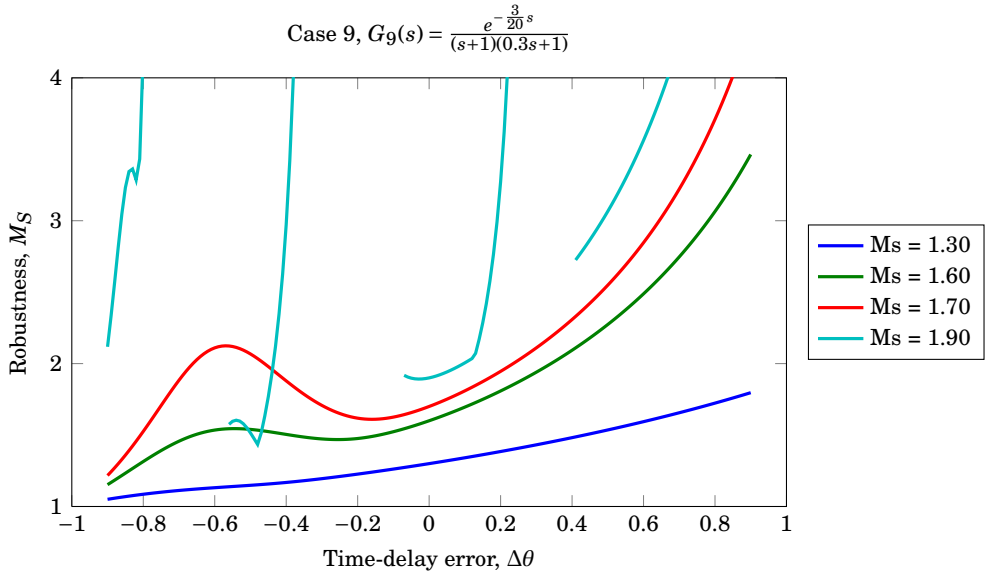


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.22 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$.

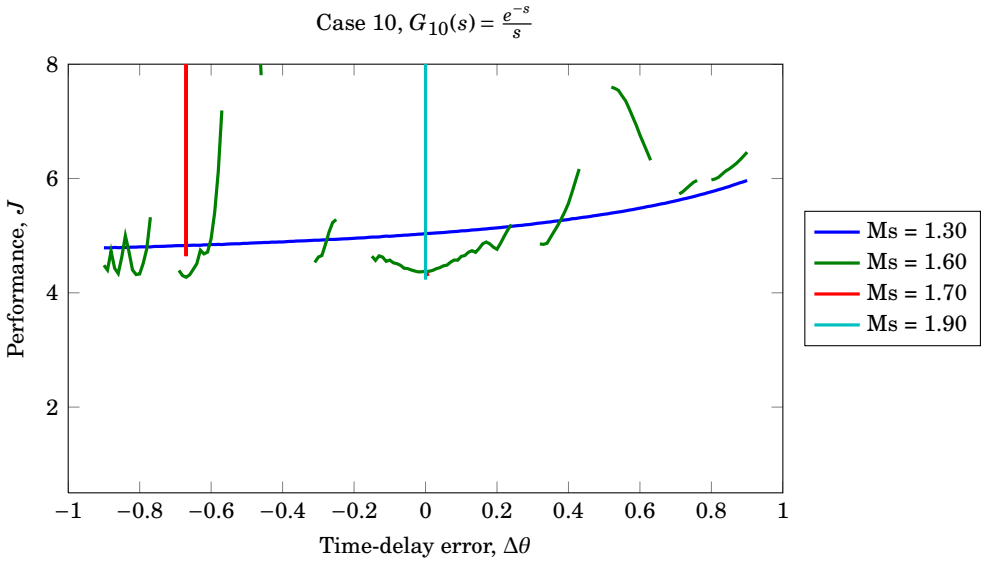


(a) Performance sensitivity, $J = f(\theta)$.

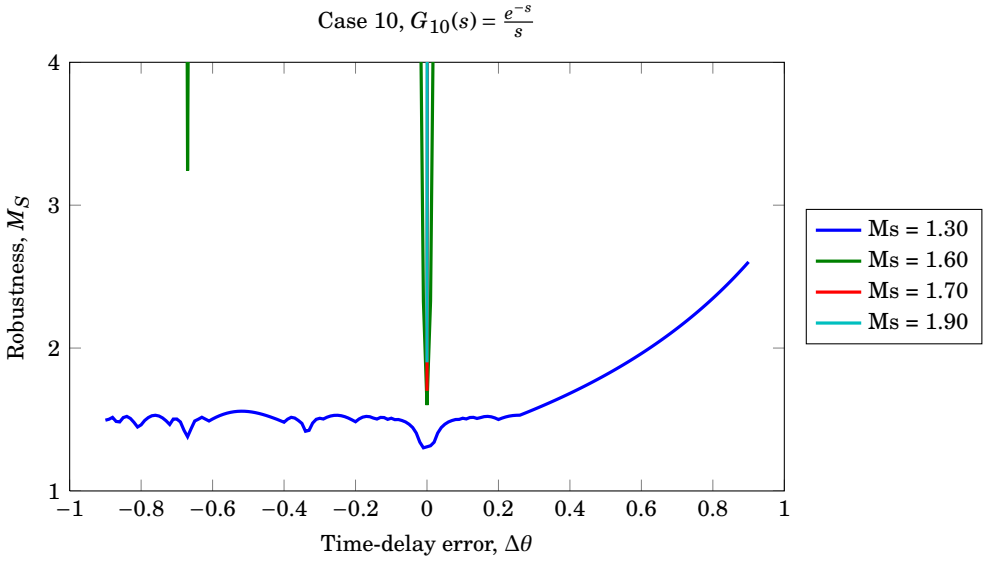


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.23 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$.

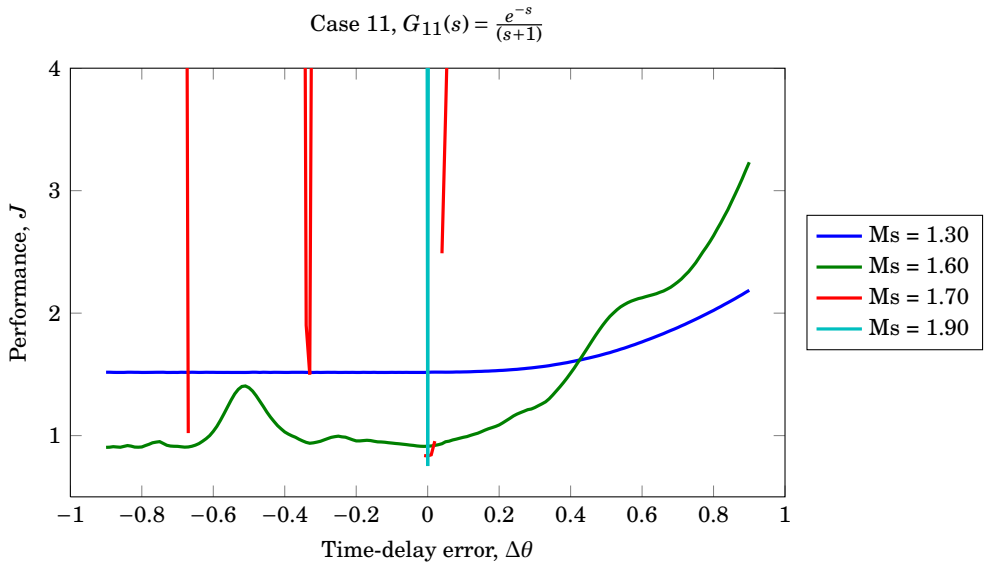


(a) Performance sensitivity, $J = f(\theta)$.

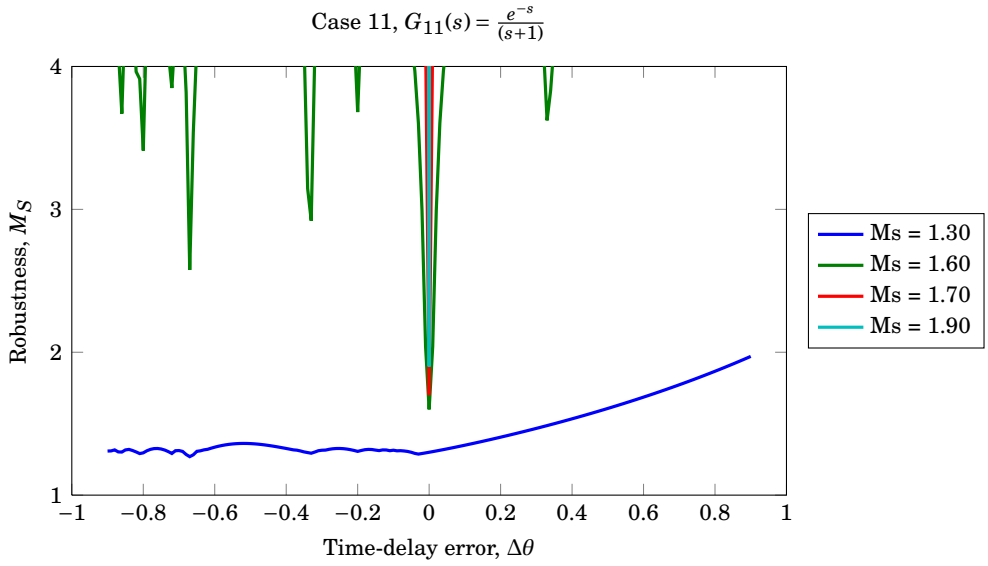


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.24 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_{10}(s) = \frac{e^{-s}}{s}$.

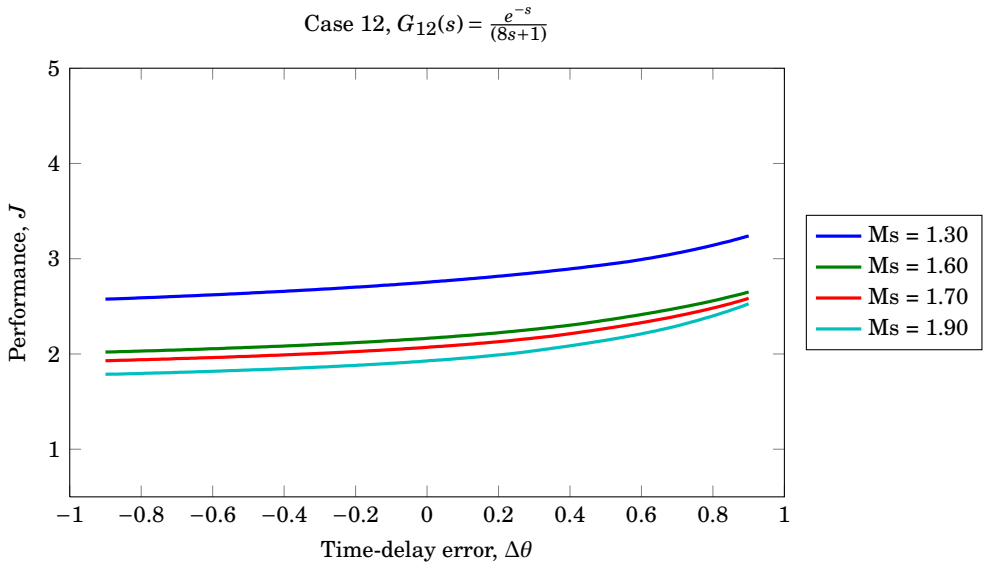


(a) Performance sensitivity, $J = f(\theta)$.

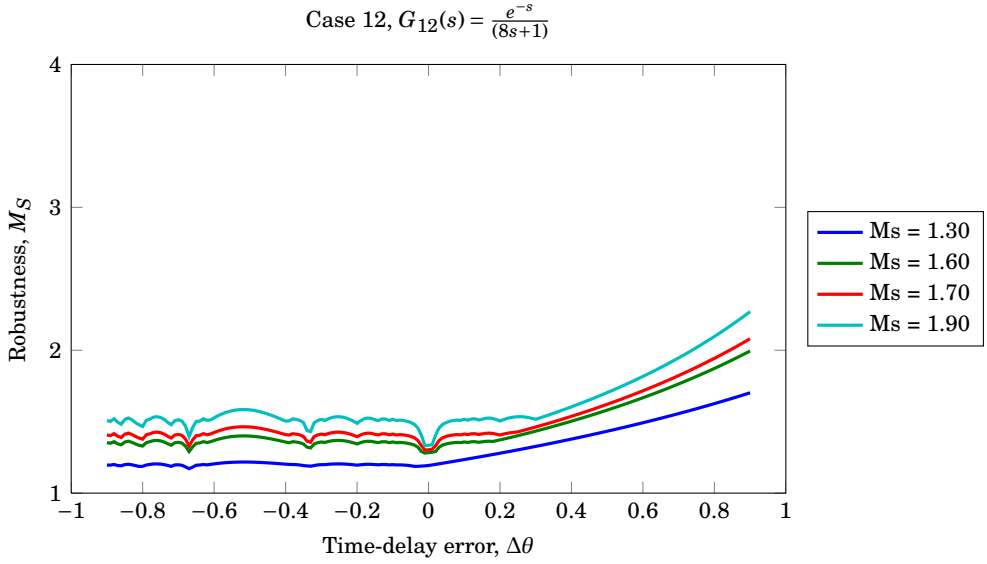


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.25 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_{11}(s) = \frac{e^{-s}}{(s+1)}$.

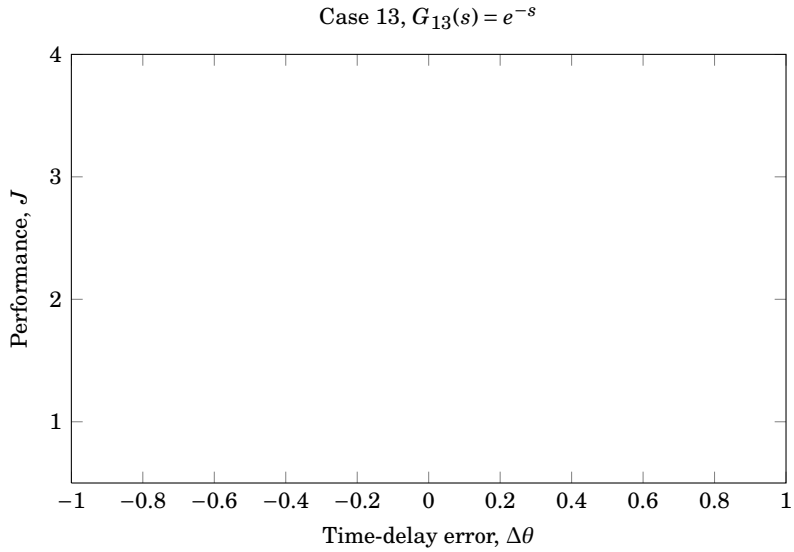


(a) Performance sensitivity, $J = f(\theta)$.

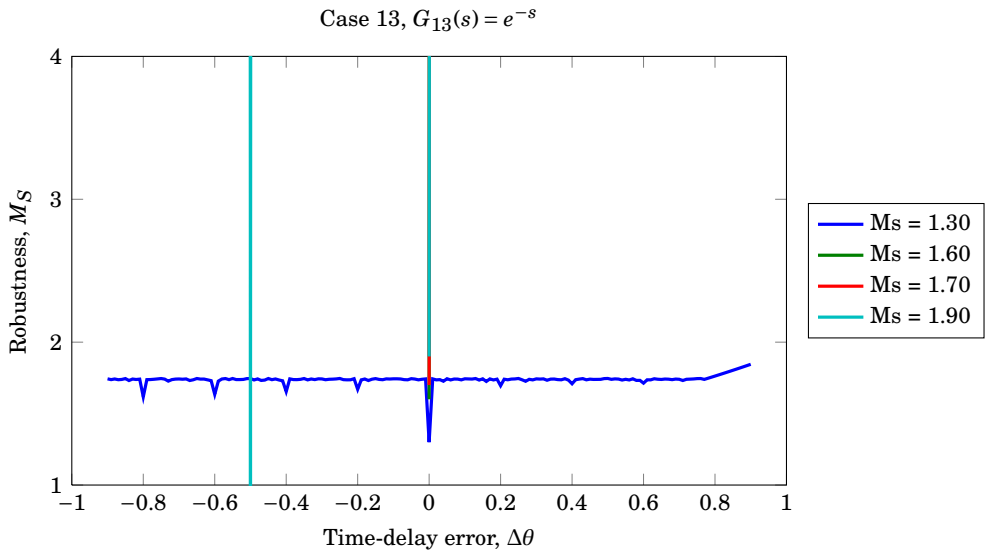


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.26 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_{12}(s) = \frac{e^{-s}}{(8s+1)}$.

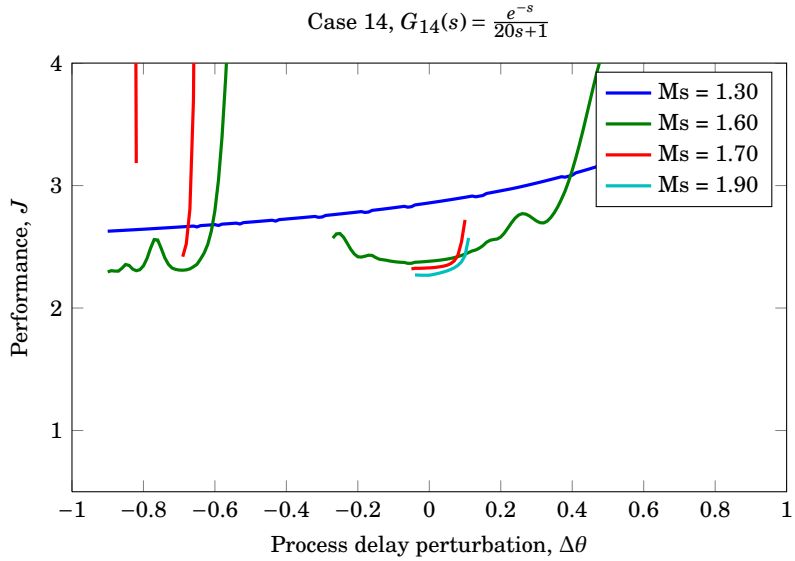


(a) Performance sensitivity, $J = f(\theta)$.

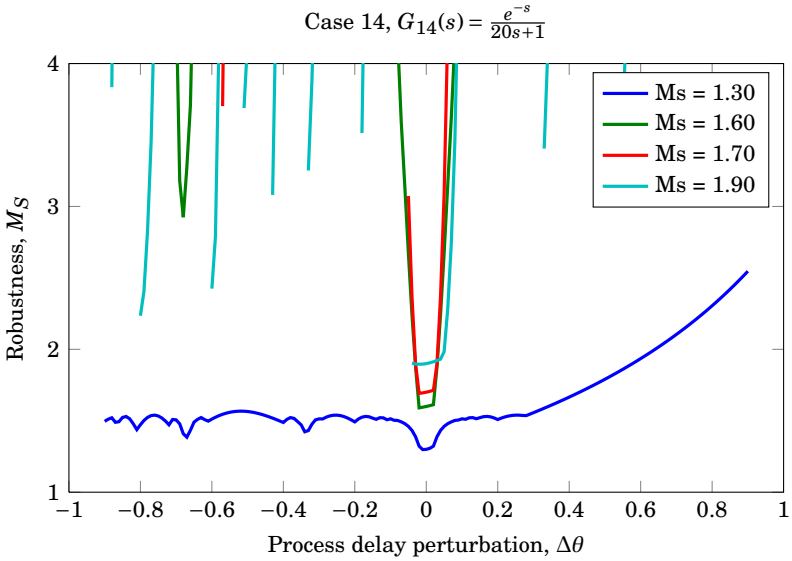


(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.27 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_{13}(s) = e^{-s}$.



(a) Performance sensitivity, $J = f(\theta)$.



(b) Robustness sensitivity, $M_S = f(\theta)$.

Figure E.28 – Performance and robustness sensitivity to modelling error in the time-delay parameter for a Pareto optimal Smith predictor PID controller for a set of target M_S values for model $G_{14}(s) = \frac{e^{-s}}{20s+1}$.

Appendix F

Smith SIMC Tuning Plots

The SIMC tuning rules for the Smith predictor have been derived from the standard SIMC tuning rules in Appendix I, and are

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c}, \tag{F.1a}$$

$$\tau_I = \min[\tau_1, 4\tau_c], \tag{F.1b}$$

$$\tau_D = \tau_2. \tag{F.1c}$$

The rules have been appended to models 1–14 by defining some span of τ_c values based on the set of expected time-delay disturbances defined in Chapter 6.4, $\mathcal{Q}$, and then computing the corresponding controller tuning. The response of the system defined by the model and the controller was computed, and the system robustness was calculated. The results were plotted and are displayed in Figure F.1–F.13. For the PI tuning of second-order models the "half rule" was used for model reduction. Reference points for the closed-loop constant are given from the set $\tau_c/\theta = [½\delta\theta^+ \ \delta\theta^+ \ ¾\delta\theta^+ \ 2\delta\theta^+]$, where $\delta\theta^+$ is the maximum time-delay error factor. Tables F.1, F.2 and F.3 show the reference points with the corresponding cost function and M_S value, along with the IAE values for the step response.

Table F.1 – Smith SIMC PI tuning for model 1, 2 and 3. The closed-loop time constant values are $\tau_c/\theta = [½\delta\theta^+ \ \delta\theta^+ \ ³⁄₂\delta\theta^+ \ 2\delta\theta^+]$.

Process	τ_c/θ	τ_c	K_c	τ_I	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	½	0.62	0.62	1.00	1.70	2.37	3.53	2.81
	1	1.25	0.44	1.00	1.59	1.79	3.14	2.80
	³⁄₂	1.88	0.35	1.00	1.70	1.56	3.24	3.10
	2	2.50	0.29	1.00	1.91	1.44	3.58	3.55
$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	½	0.70	0.59	1.00	2.00	2.50	4.32	3.37
	1	1.40	0.42	1.00	1.83	1.84	3.74	3.30
	³⁄₂	2.10	0.32	1.00	1.91	1.59	3.78	3.57
	2	2.80	0.26	1.00	2.10	1.46	4.07	3.99
$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	½	0.57	0.63	1.00	1.50	2.23	2.97	2.38
	1	1.15	0.47	1.00	1.46	1.73	2.73	2.46
	³⁄₂	1.72	0.37	1.00	1.60	1.53	2.89	2.81
	2	2.30	0.30	1.00	1.86	1.41	3.31	3.30

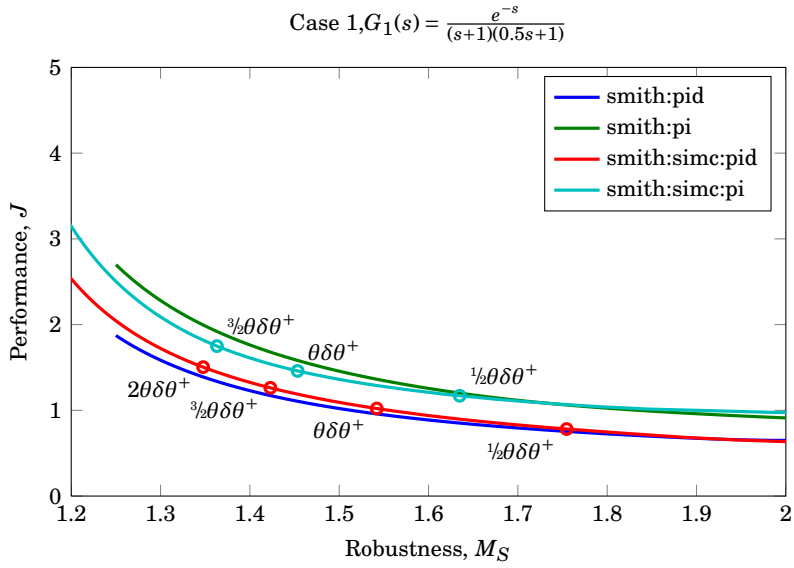


Figure F.1 – Pareto optimal Smith PI and PID tuning compared with Smith SIMC PI and PID tuning for $G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$.

Table F.2 – Smith SIMC PI tuning for model 4 through 13. The closed-loop time constant values are $\tau_c/\theta = [½\delta\theta^+ \ \delta\theta^+ \ ¾\delta\theta^+ \ 2\delta\theta^+]$.

Process	τ_c/θ	τ_c	K_c	τ_I	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	½	0.29	1.60	1.00	2.34	2.64	1.98	0.99
	1	0.58	1.09	1.00	2.25	1.89	1.69	1.11
	¾	0.87	0.83	1.00	2.45	1.62	1.68	1.32
	2	1.17	0.67	1.00	2.74	1.48	1.77	1.56
$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	½	0.47	1.00	1.00	2.48	2.64	3.16	1.97
	1	0.93	0.68	1.00	2.32	1.89	2.70	2.07
	¾	1.40	0.52	1.00	2.45	1.62	2.69	2.33
	2	1.87	0.42	1.00	2.68	1.48	2.84	2.64
$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	½	0.17	2.67	1.00	2.24	2.64	1.19	0.44
	1	0.35	1.82	1.00	2.25	1.89	1.01	0.55
	¾	0.52	1.38	1.00	2.57	1.62	1.01	0.72
	2	0.70	1.11	1.00	2.96	1.48	1.06	0.90
$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	½	0.25	2.00	1.00	2.68	2.67	1.74	0.75
	1	0.50	1.33	1.00	2.64	1.90	1.49	0.89
	¾	0.75	1.00	1.00	2.91	1.63	1.48	1.09
	2	1.00	0.80	1.00	3.28	1.49	1.56	1.30
$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	½	0.40	1.25	1.00	2.82	2.67	2.78	1.54
	1	0.80	0.83	1.00	2.70	1.90	2.38	1.69
	¾	1.20	0.62	1.00	2.89	1.63	2.37	1.94
	2	1.60	0.50	1.00	3.18	1.49	2.49	2.23
$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	½	0.15	3.33	1.00	2.60	2.67	1.04	0.33
	1	0.30	2.22	1.00	2.71	1.90	0.89	0.45
	¾	0.45	1.67	1.00	3.14	1.63	0.89	0.60
	2	0.60	1.33	1.00	3.65	1.49	0.93	0.75
$G_{10}(s) = \frac{e^{-s}}{s}$	½	0.50	0.67	6.00	1.76	2.18	3.40	9.00
	1	1.00	0.50	8.00	2.47	1.70	3.92	15.99
	¾	1.50	0.40	10.00	3.36	1.50	4.51	24.97
	2	2.00	0.33	12.00	4.41	1.39	5.14	35.87
$G_{11}(s) = \frac{e^{-s}}{(s+1)}$	½	0.50	0.67	1.00	1.30	1.92	2.13	1.73
	1	1.00	0.50	1.00	1.41	1.59	2.17	2.03
	¾	1.50	0.40	1.00	1.68	1.44	2.50	2.50
	2	2.00	0.33	1.00	2.02	1.35	3.00	3.00
$G_{12}(s) = \frac{e^{-s}}{(8s+1)}$	½	0.50	5.33	6.00	1.70	1.97	2.37	1.12
	1	1.00	4.00	8.00	2.38	1.59	2.17	1.99
	¾	1.50	3.20	8.00	2.91	1.44	2.51	2.49
	2	2.00	2.67	8.00	3.50	1.35	3.00	2.99
$G_{13}(s) = e^{-s}$	½	0.50	0.00	0.01	1.33	1.92	2.13	2.12
	1	1.00	0.00	0.01	1.36	1.59	2.17	2.16
	¾	1.50	0.00	0.01	1.57	1.44	2.50	2.49
	2	2.00	0.00	0.01	1.88	1.35	3.00	2.99
$G_{14}(s) = e^{-s}$	½	0.50	13.33	6.00	1.74	2.08	2.98	0.45
	1	1.00	10.00	8.00	2.43	1.65	3.20	0.80
	¾	1.50	8.00	10.00	3.30	1.47	3.43	1.25
	2	2.00	6.67	12.00	4.33	1.36	3.68	1.78

Table F.3 – Smith SIMC PID tuning for all models. The closed-loop time constant values are $\tau_c/\theta = [½\delta\theta^+ \ \delta\theta^+ \ ¾\delta\theta^+ \ 2\delta\theta^+]$.

Process	τ_c/θ	τ_c	K_c	τ_I	τ_D	J	M_S	IAE_{d_o}	IAE_{d_i}
$G_1(s) = \frac{e^{-s}}{(s+1)(0.5s+1)}$	½	0.50	0.67	1.00	0.50	1.02	1.92	2.13	1.67
	1	1.00	0.50	1.00	0.50	1.12	1.59	2.17	2.02
	¾	1.50	0.40	1.00	0.50	1.34	1.44	2.50	2.50
	2	2.00	0.33	1.00	0.50	1.61	1.35	3.00	3.00
$G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$	½	0.50	0.67	1.00	0.80	0.97	1.92	2.13	1.59
	1	1.00	0.50	1.00	0.80	1.09	1.59	2.17	2.01
	¾	1.50	0.40	1.00	0.80	1.30	1.44	2.50	2.50
	2	2.00	0.33	1.00	0.80	1.56	1.35	3.00	3.00
$G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$	½	0.50	0.67	1.00	0.30	1.08	1.92	2.13	1.71
	1	1.00	0.50	1.00	0.30	1.18	1.59	2.17	2.03
	¾	1.50	0.40	1.00	0.30	1.41	1.44	2.50	2.50
	2	2.00	0.33	1.00	0.30	1.69	1.35	3.00	3.00
$G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$	½	0.17	2.00	1.00	0.50	0.98	1.92	0.71	0.50
	1	0.33	1.50	1.00	0.50	1.14	1.59	0.72	0.67
	¾	0.50	1.20	1.00	0.50	1.38	1.44	0.84	0.83
	2	0.67	1.00	1.00	0.50	1.65	1.35	1.00	1.00
$G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$	½	0.27	1.25	1.00	0.80	0.94	1.92	1.14	0.80
	1	0.53	0.94	1.00	0.80	1.09	1.59	1.16	1.07
	¾	0.80	0.75	1.00	0.80	1.31	1.44	1.34	1.33
	2	1.07	0.62	1.00	0.80	1.57	1.35	1.60	1.60
$G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$	½	0.10	3.33	1.00	0.30	1.07	1.92	0.43	0.30
	1	0.20	2.50	1.00	0.30	1.27	1.59	0.43	0.40
	¾	0.30	2.00	1.00	0.30	1.54	1.44	0.50	0.50
	2	0.40	1.67	1.00	0.30	1.84	1.35	0.60	0.60
$G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$	½	0.12	2.67	1.00	0.50	1.03	1.92	0.53	0.38
	1	0.25	2.00	1.00	0.50	1.21	1.59	0.54	0.50
	¾	0.38	1.60	1.00	0.50	1.46	1.44	0.63	0.63
	2	0.50	1.33	1.00	0.50	1.75	1.35	0.75	0.75
$G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$	½	0.20	1.67	1.00	0.80	0.96	1.92	0.85	0.60
	1	0.40	1.25	1.00	0.80	1.13	1.59	0.87	0.80
	¾	0.60	1.00	1.00	0.80	1.36	1.44	1.00	1.00
	2	0.80	0.83	1.00	0.80	1.63	1.35	1.20	1.20
$G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$	½	0.07	4.44	0.90	0.30	1.12	1.94	0.34	0.20
	1	0.15	3.33	1.00	0.30	1.39	1.59	0.33	0.30
	¾	0.22	2.67	1.00	0.30	1.68	1.44	0.38	0.37
	2	0.30	2.22	1.00	0.30	2.02	1.35	0.45	0.45

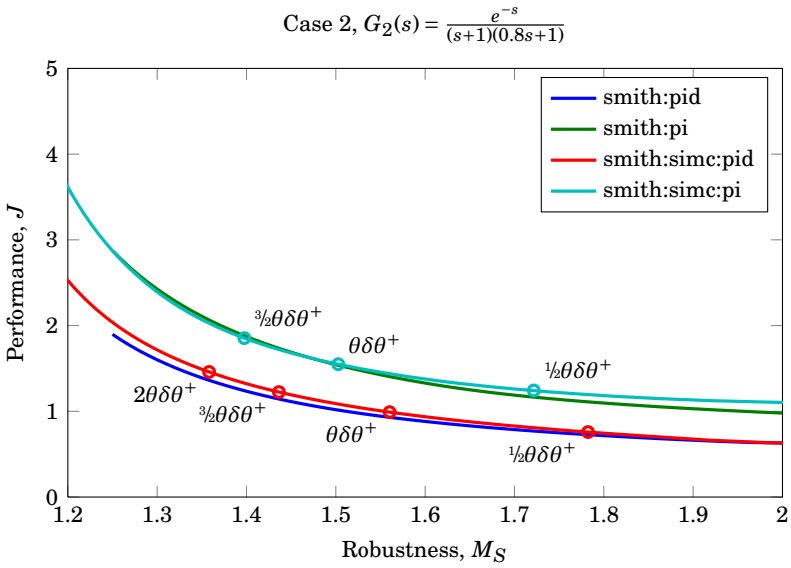


Figure F.2 – Pareto optimal Smith PI and PID tuning compared with Smith SIMC PI and PID tuning for $G_2(s) = \frac{e^{-s}}{(s+1)(0.8s+1)}$.

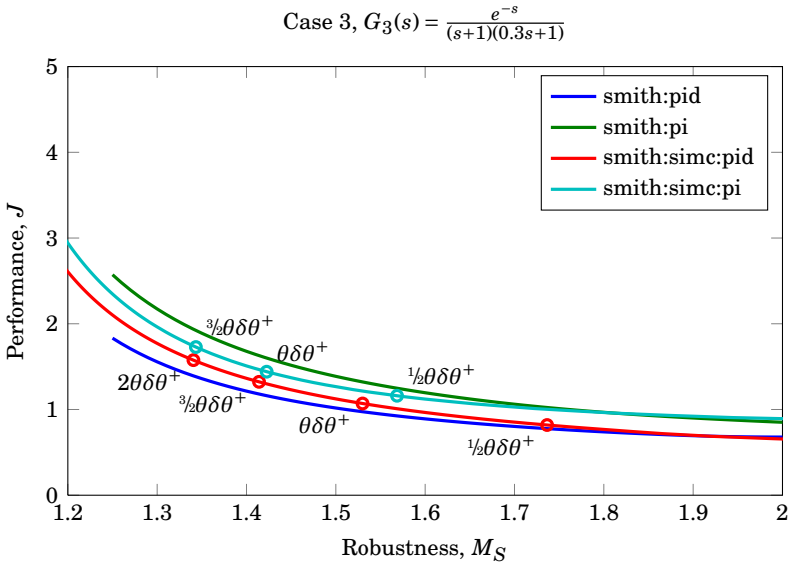


Figure F.3 – Pareto optimal Smith PI and PID tuning compared with Smith SIMC PI and PID tuning for $G_3(s) = \frac{e^{-s}}{(s+1)(0.3s+1)}$.

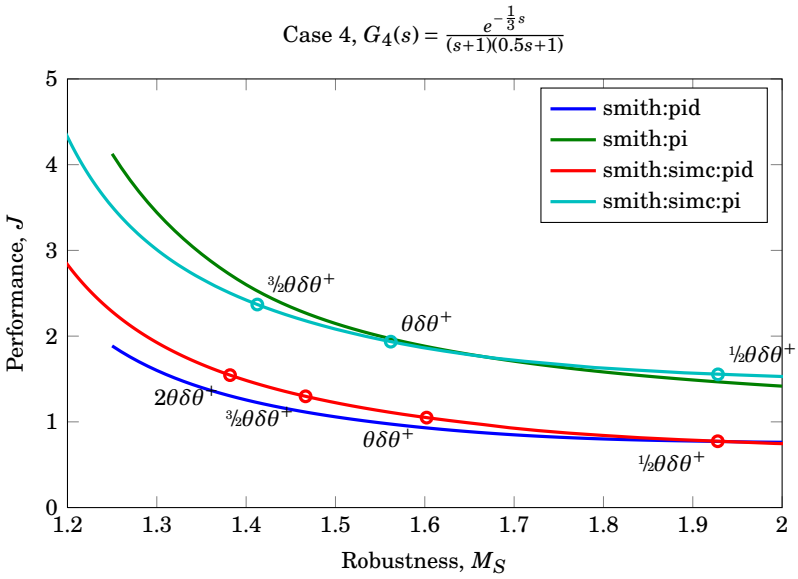


Figure F.4 – Pareto optimal Smith PI and PID tuning compared with Smith SIMC PI and PID tuning for $G_4(s) = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$.

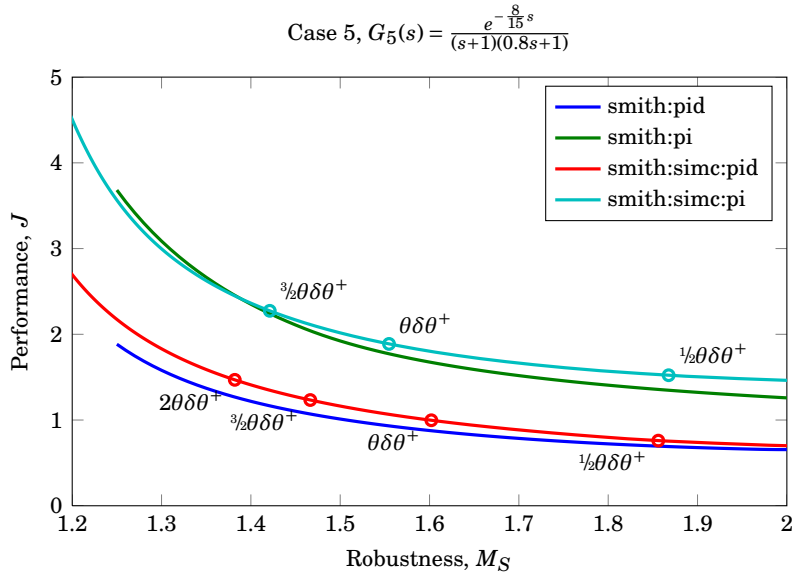


Figure F.5 – Pareto optimal Smith PI and PID tuning compared with Smith SIMC PI and PID tuning for $G_5(s) = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$.

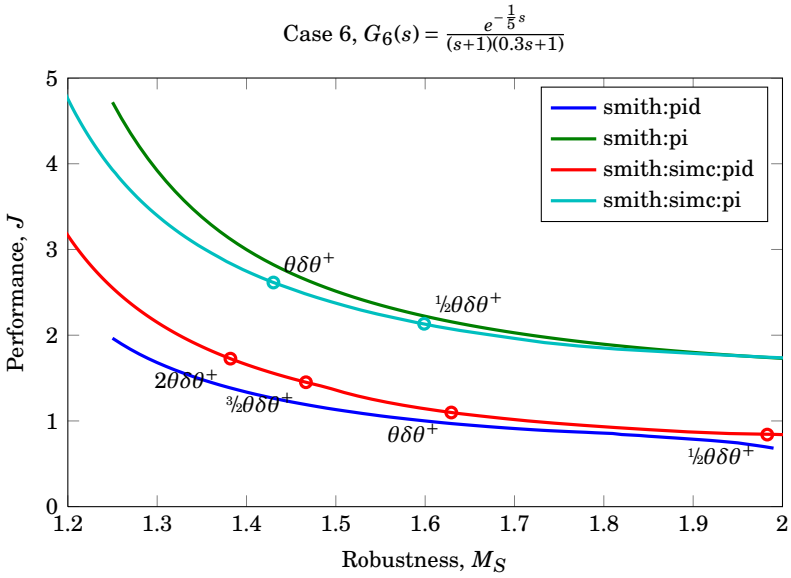


Figure F.6 – Pareto optimal Smith PI and PID tuning compared with Smith SIMC PI and PID tuning for $G_6(s) = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$.

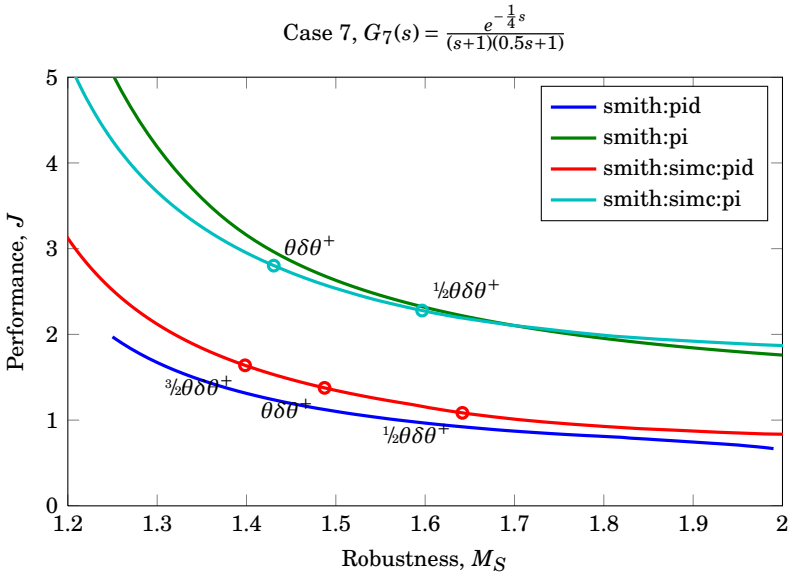


Figure F.7 – Pareto optimal Smith PI and PID tuning compared with Smith SIMC PI and PID tuning for $G_7(s) = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$.

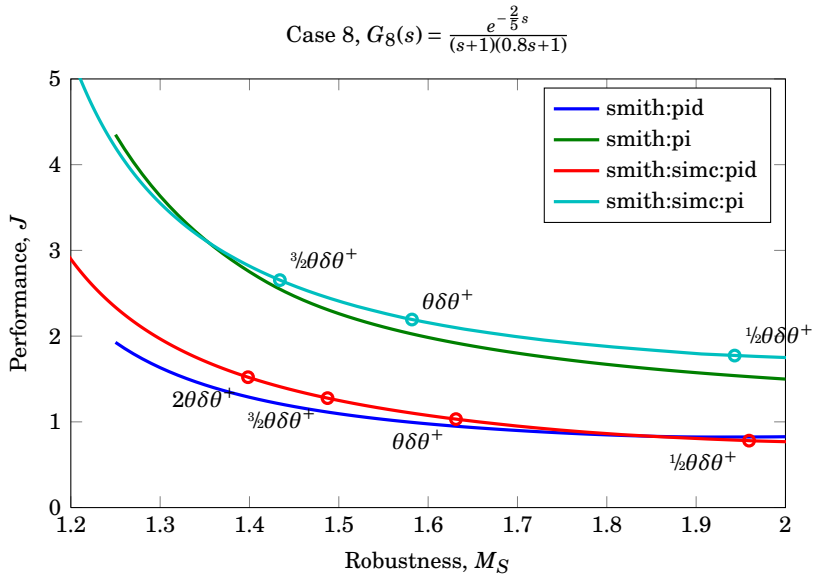


Figure F.8 – Pareto optimal Smith PI and PID tuning compared with Smith SIMC PI and PID tuning for $G_8(s) = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$.

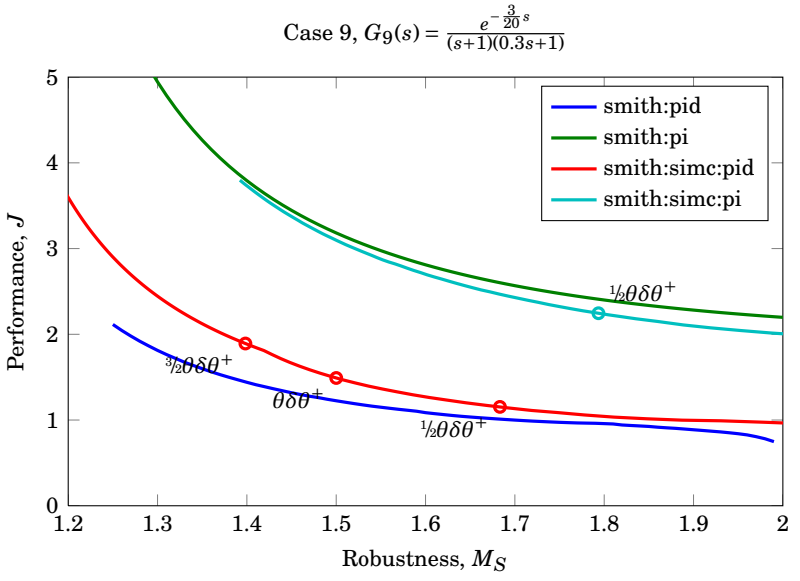


Figure F.9 – Pareto optimal Smith PI and PID tuning compared with Smith SIMC PI and PID tuning for $G_9(s) = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$.

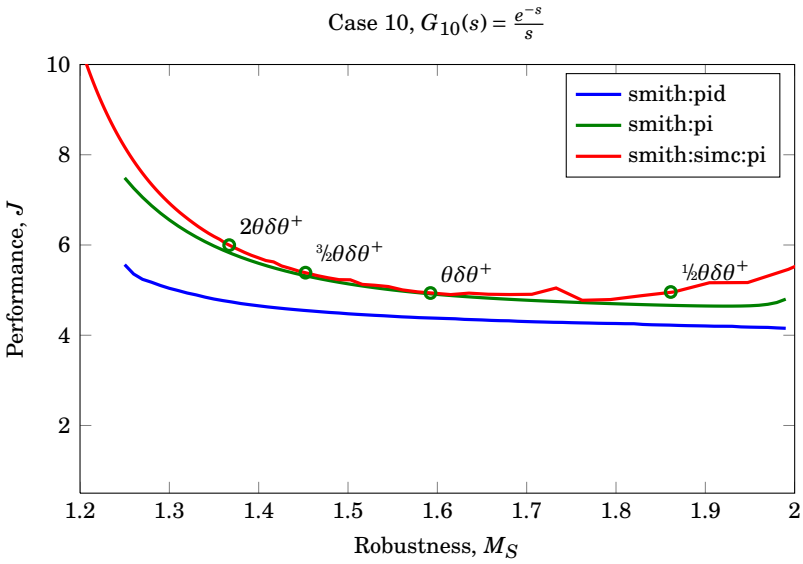


Figure F.10 – Pareto optimal Smith PI tuning compared with Smith SIMC PI tuning for $G_{10}(s) = \frac{e^{-s}}{s}$.

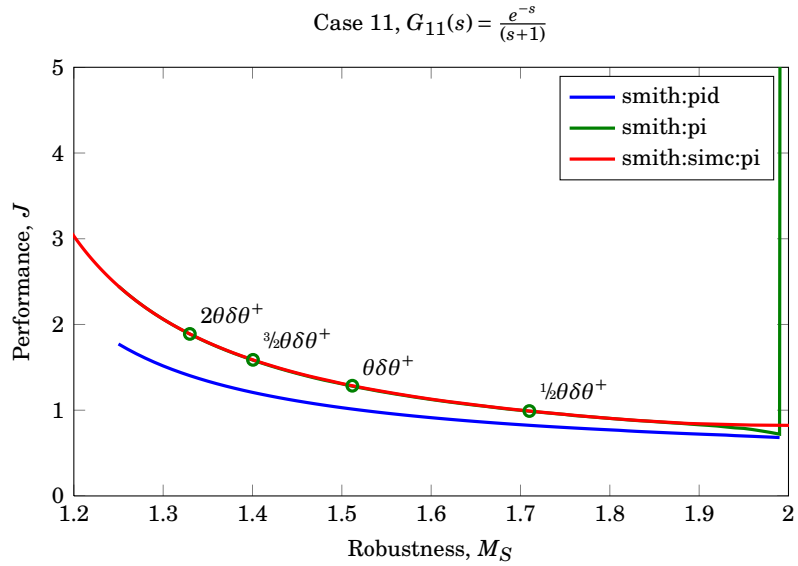


Figure F.11 – Pareto optimal Smith PI tuning compared with Smith SIMC PI tuning for $G_{11}(s) = \frac{e^{-s}}{(s+1)}$.

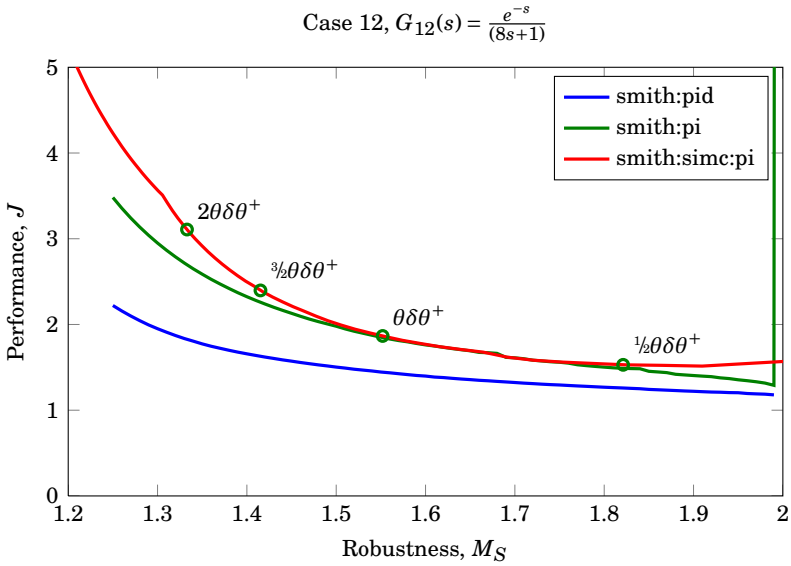


Figure F.12 – Pareto optimal Smith PI tuning compared with Smith SIMC PI tuning for $G_{12}(s) = \frac{e^{-s}}{(8s+1)}$.

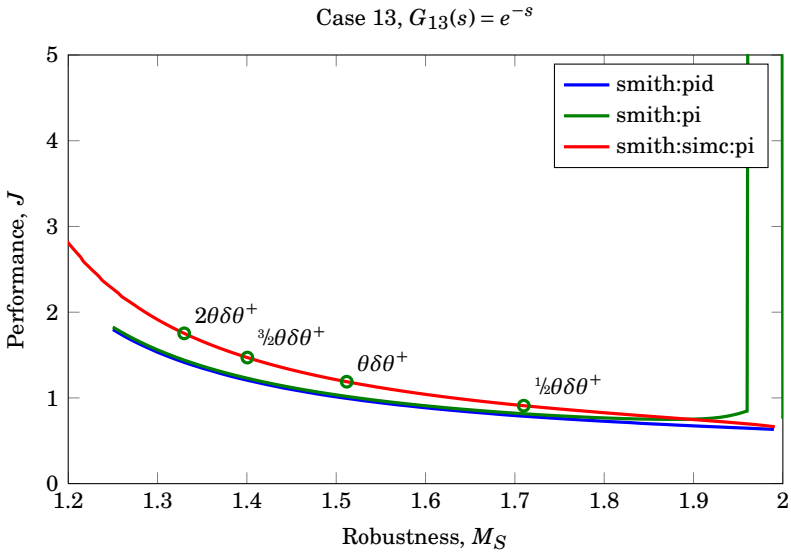


Figure F.13 – Pareto optimal Smith PI tuning compared with Smith SIMC PI tuning for $G_{13}(s) = e^{-s}$.

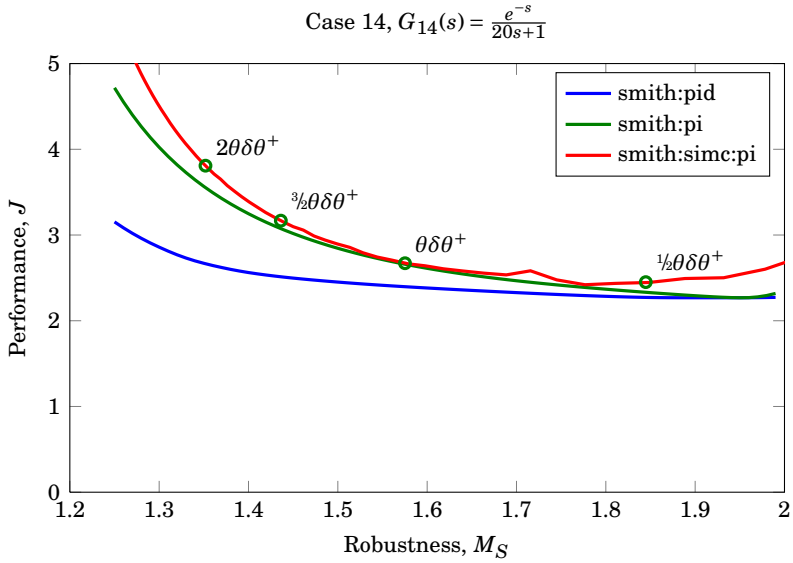
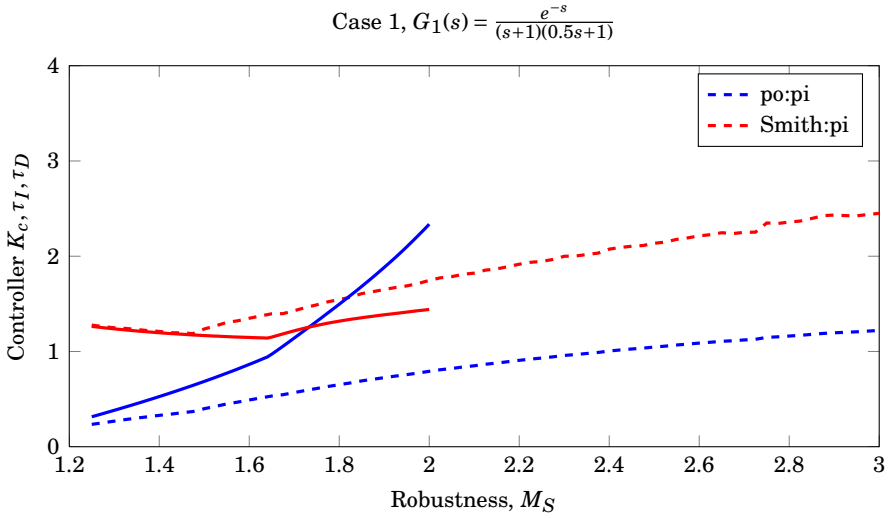


Figure F.14 – Pareto optimal Smith PI tuning compared with Smith SIMC PI tuning for $G_{14}(s) = \frac{e^{-s}}{20s+1}$.

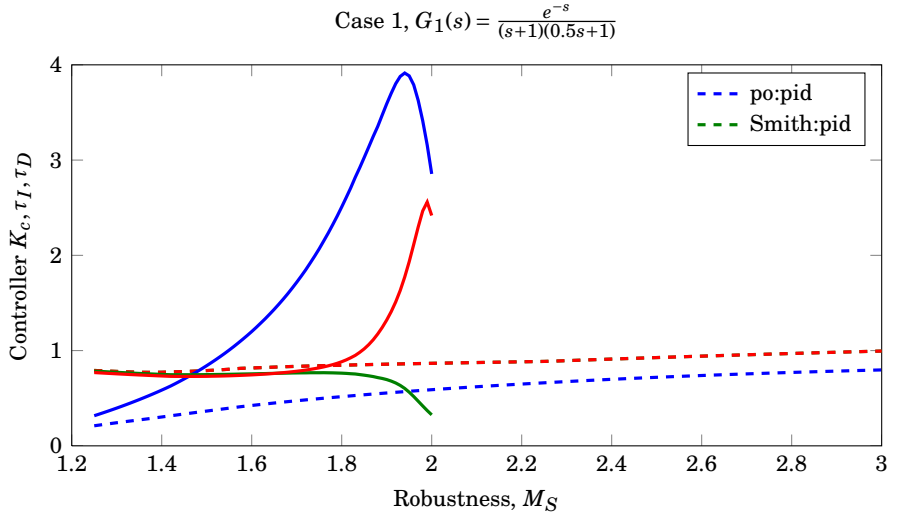
APPENDIX G

CONTROLLER ACTION

The controller parameters gain (K_c), integral time (τ_I) and derivative time (τ_D) is graphically represented as functions of robustness (M_S) for the Pareto optimal PI and PID controllers, together with the Pareto optimal Smith predictor PI and PID controllers. The purpose is to evaluate convergence of the optimisation routines and easily compare Pareto optimal PI and PID controller tuning with the Pareto optimal Smith predictor controller tuning. Case 1 through 14 has been evaluated. The plots are given in Figures G.1 through G.13. It should be noted that the ordinate are of unequal scale for the different cases to increase readability of the plots.

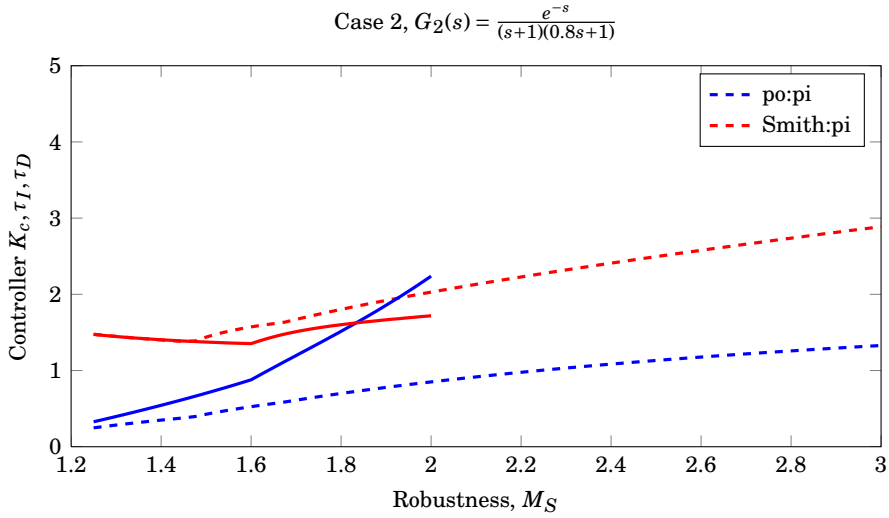


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

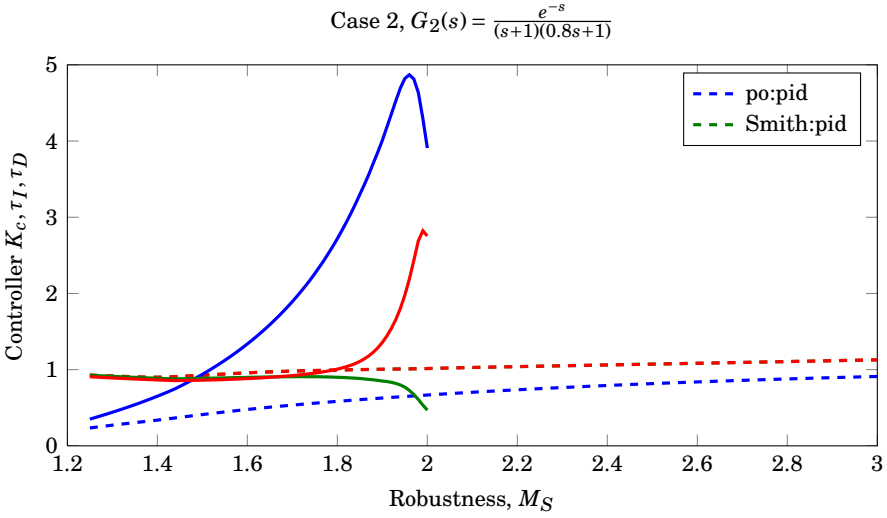


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.1 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_1 = \frac{e^{-s}}{(s+1)(0.5s+1)}$.

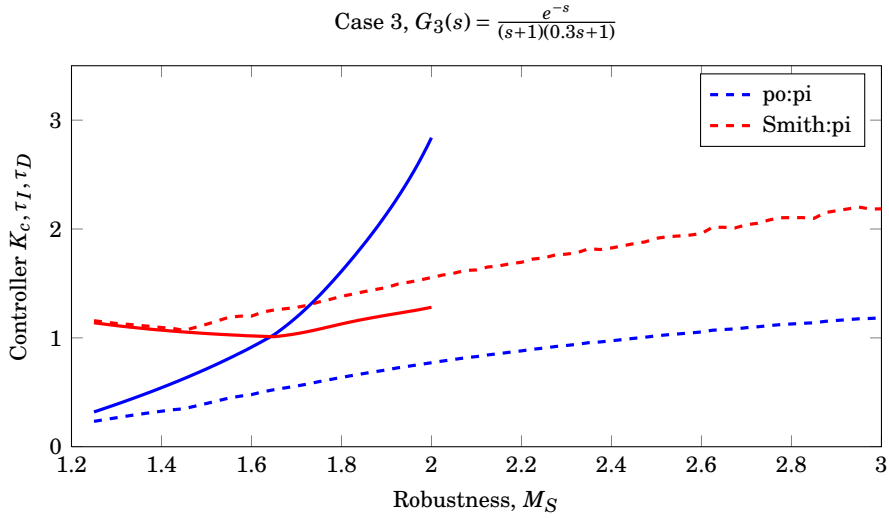


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

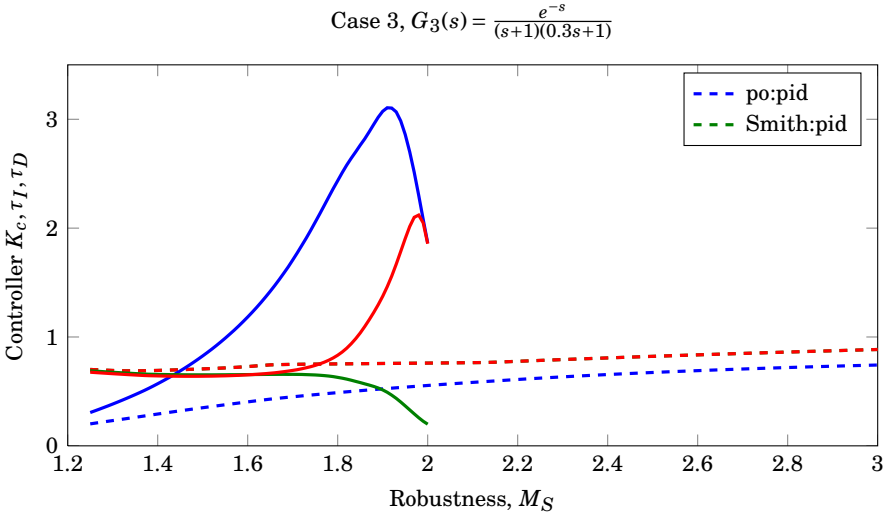


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.2 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_2 = \frac{e^{-s}}{(s+1)(0.8s+1)}$.

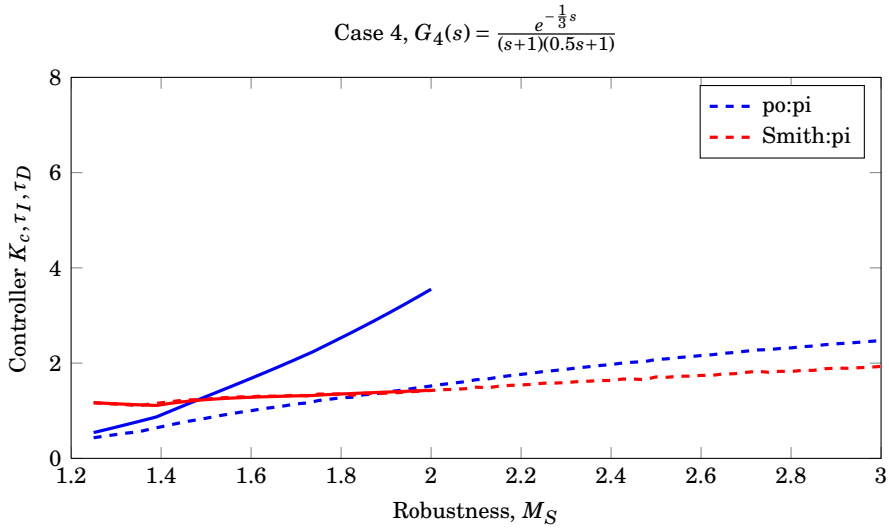


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

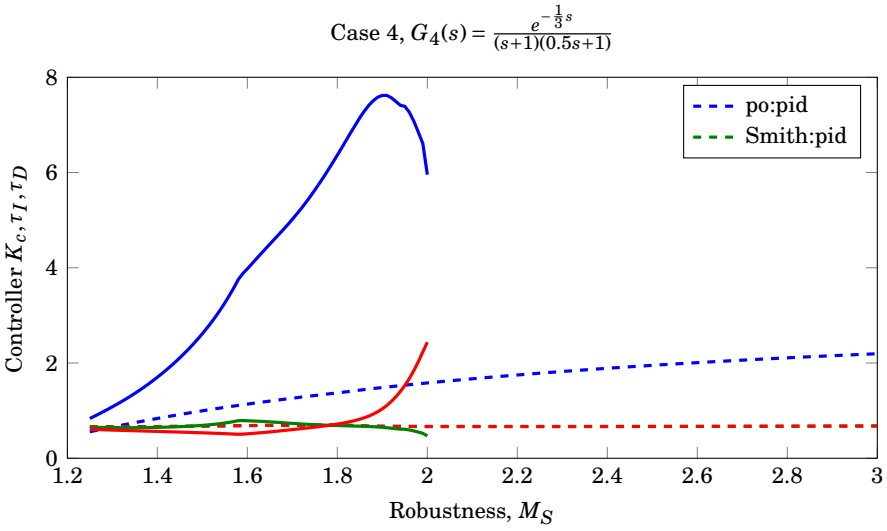


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.3 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_3 = \frac{e^{-s}}{(s+1)(0.3s+1)}$.

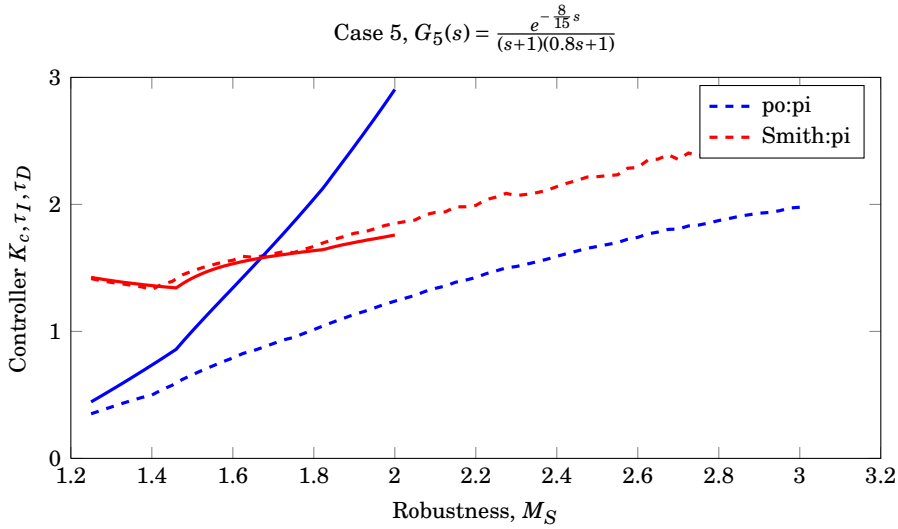


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

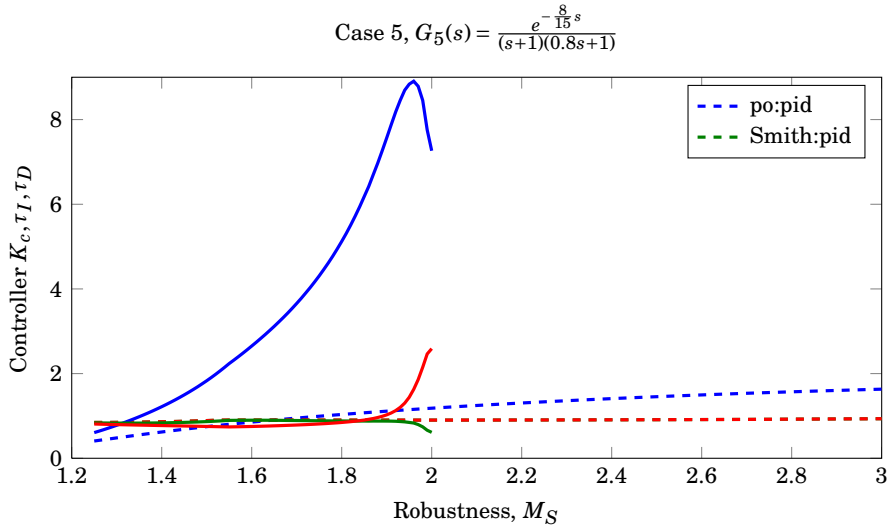


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.4 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_4 = \frac{e^{-\frac{1}{3}s}}{(s+1)(0.5s+1)}$.

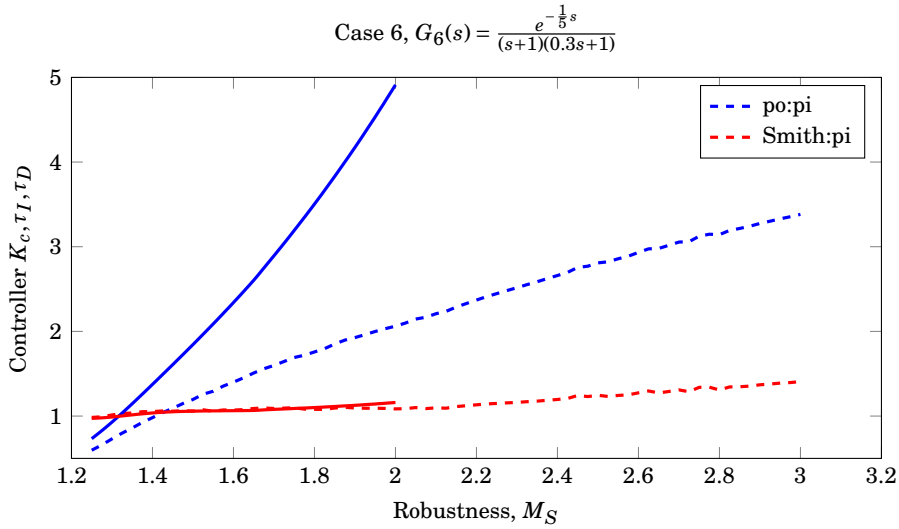


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

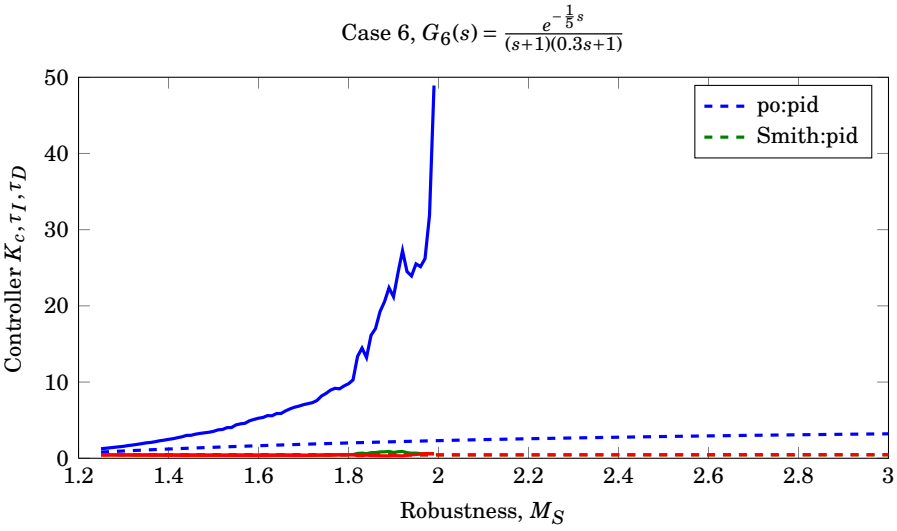


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.5 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_5 = \frac{e^{-\frac{8}{15}s}}{(s+1)(0.8s+1)}$.

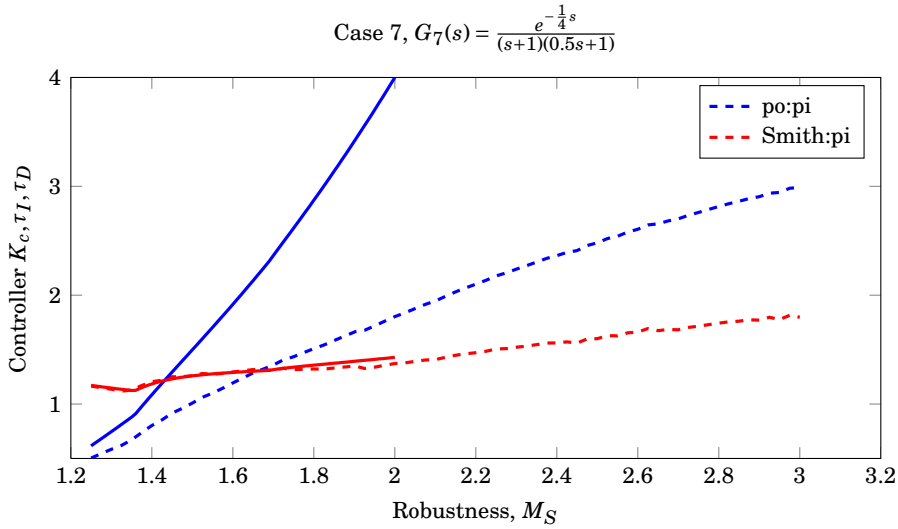


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

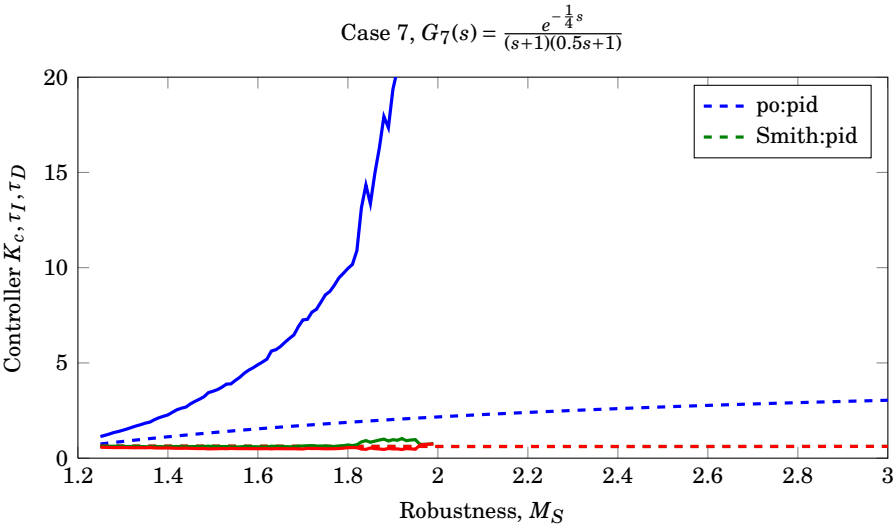


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.6 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_6 = \frac{e^{-\frac{1}{5}s}}{(s+1)(0.3s+1)}$.

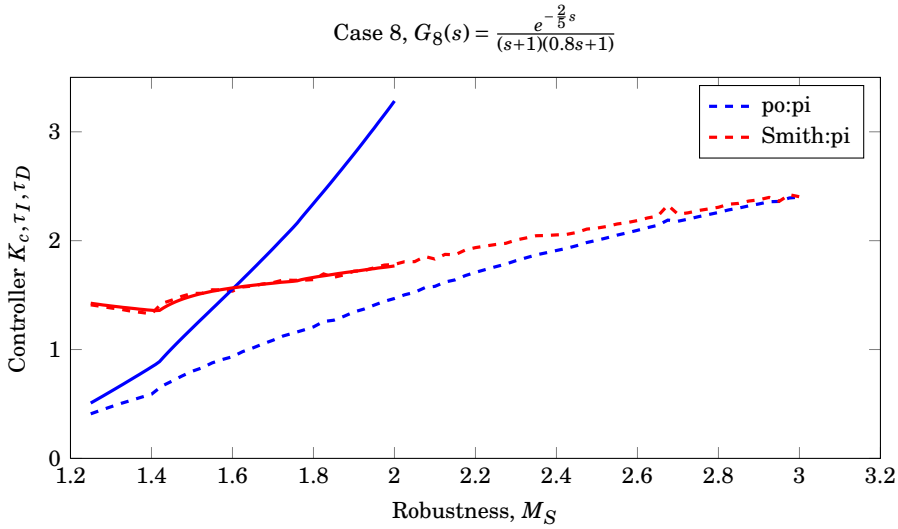


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

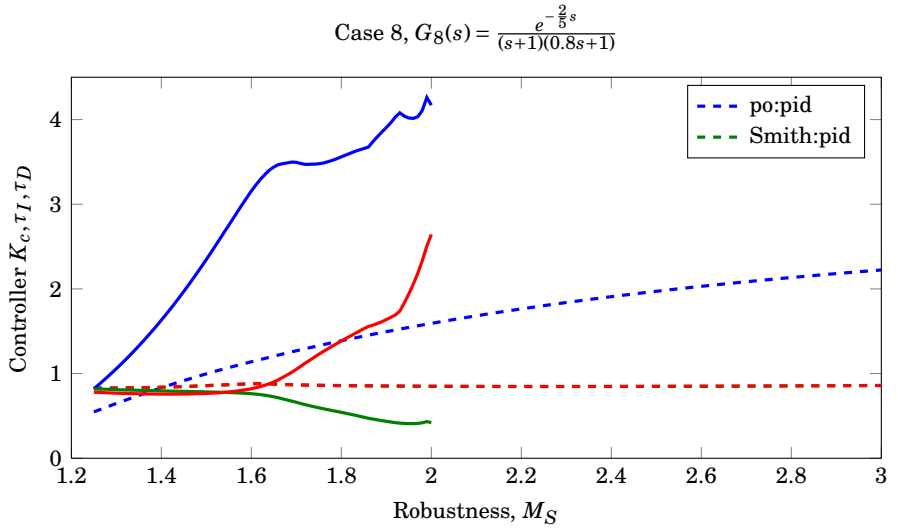


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.7 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_7 = \frac{e^{-\frac{1}{4}s}}{(s+1)(0.5s+1)}$.

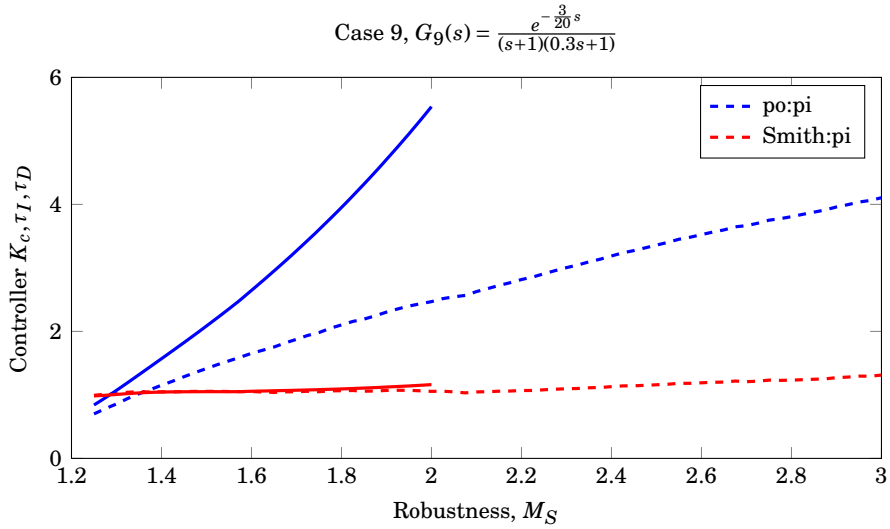


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

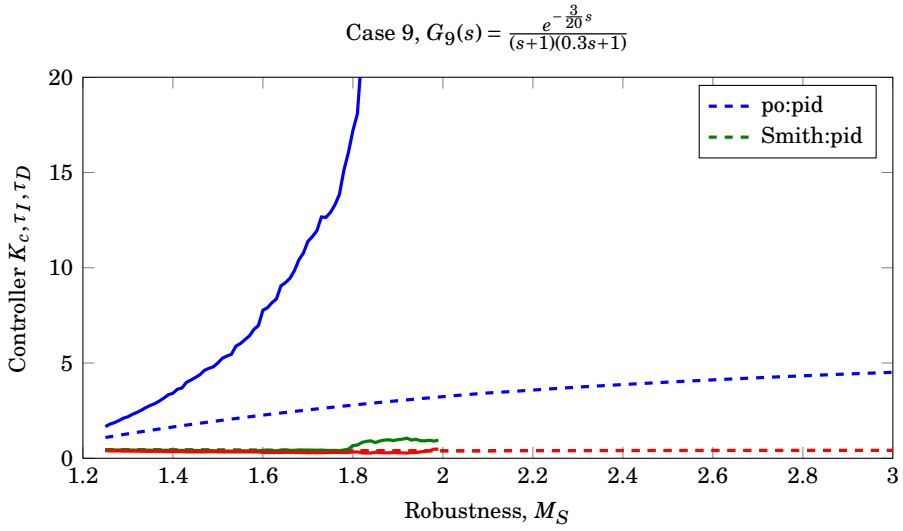


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.8 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_8 = \frac{e^{-\frac{2}{5}s}}{(s+1)(0.8s+1)}$.

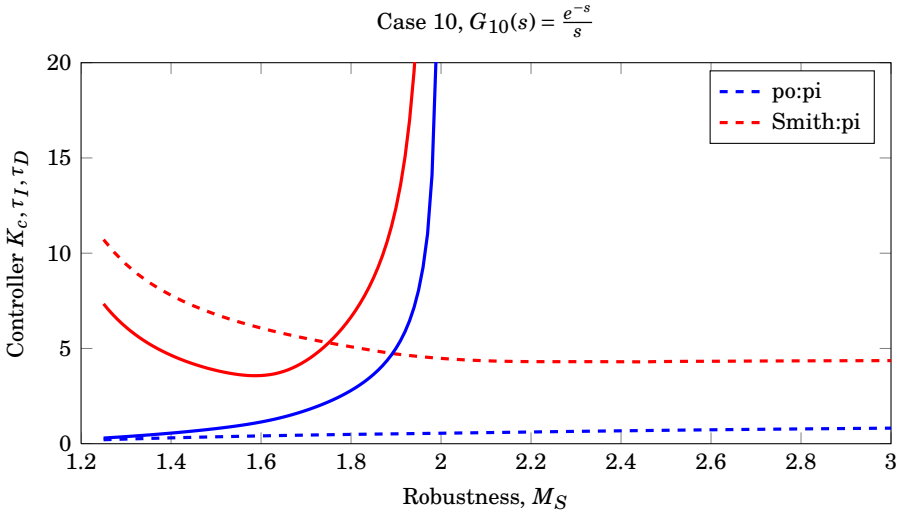


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

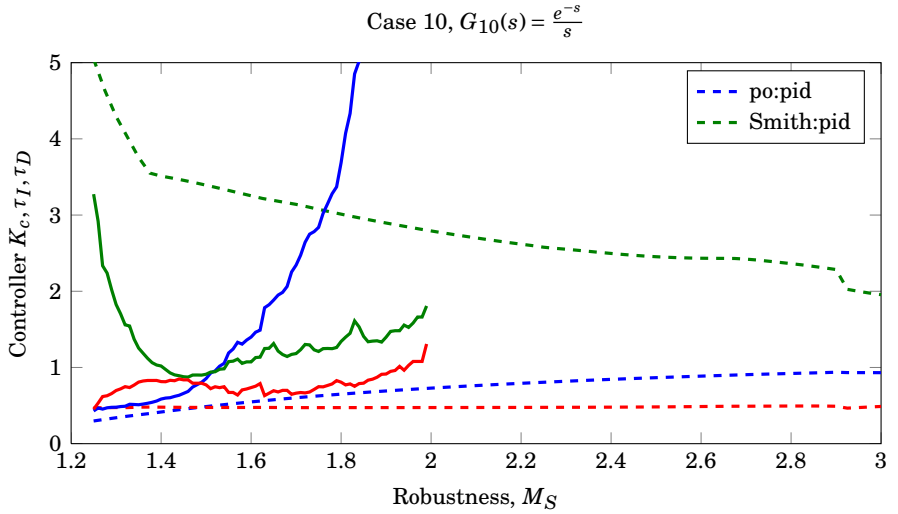


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.9 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_9 = \frac{e^{-\frac{3}{20}s}}{(s+1)(0.3s+1)}$.

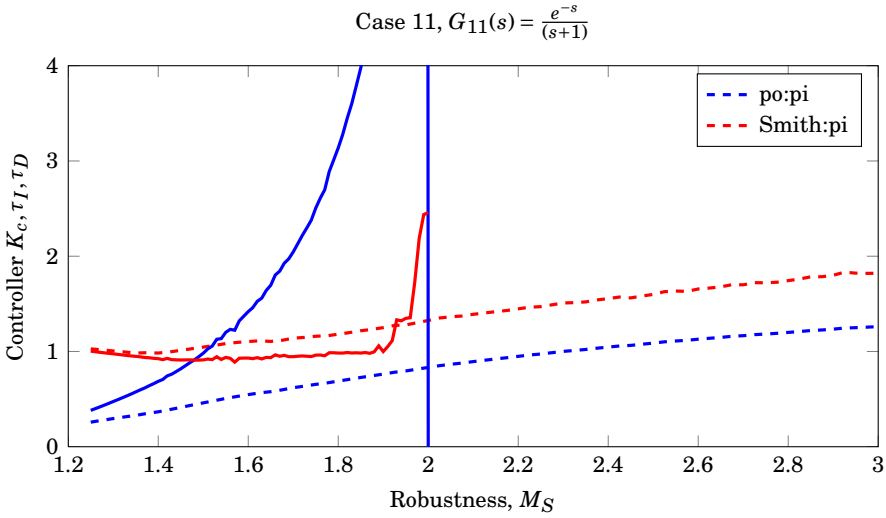


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

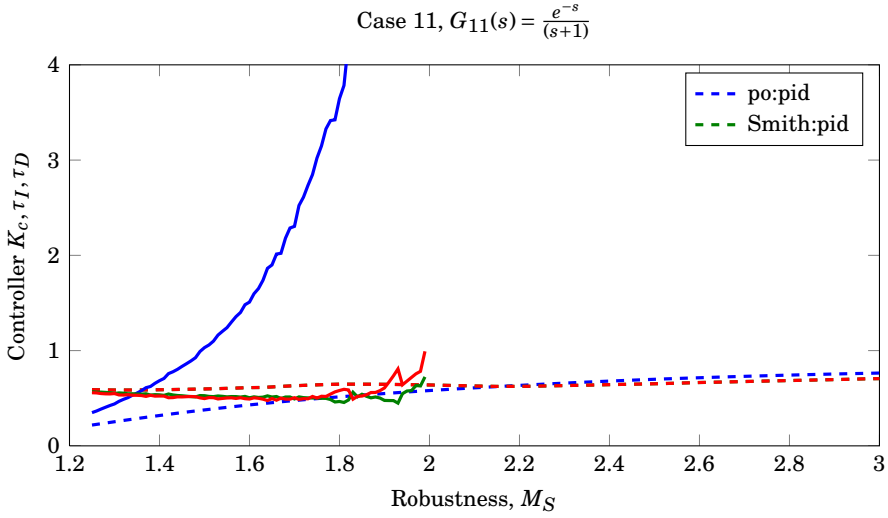


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.10 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_{10} = \frac{e^{-s}}{s}$.

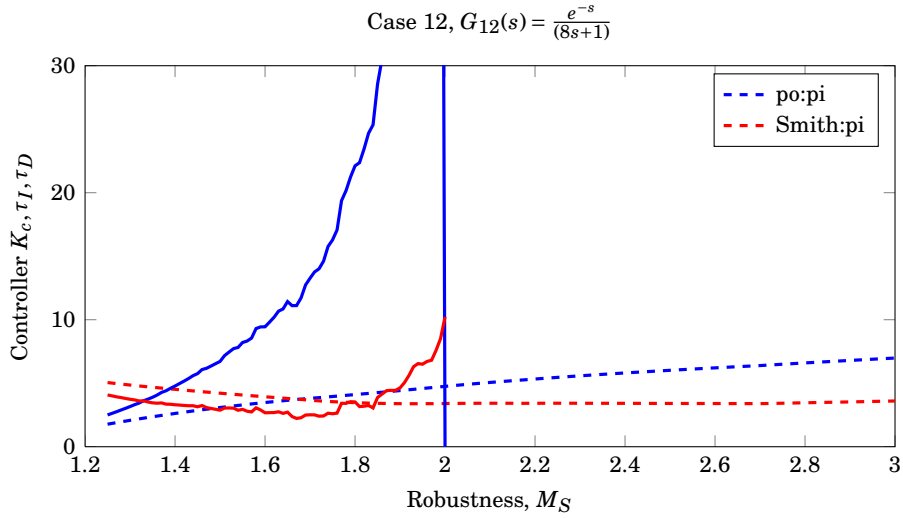


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

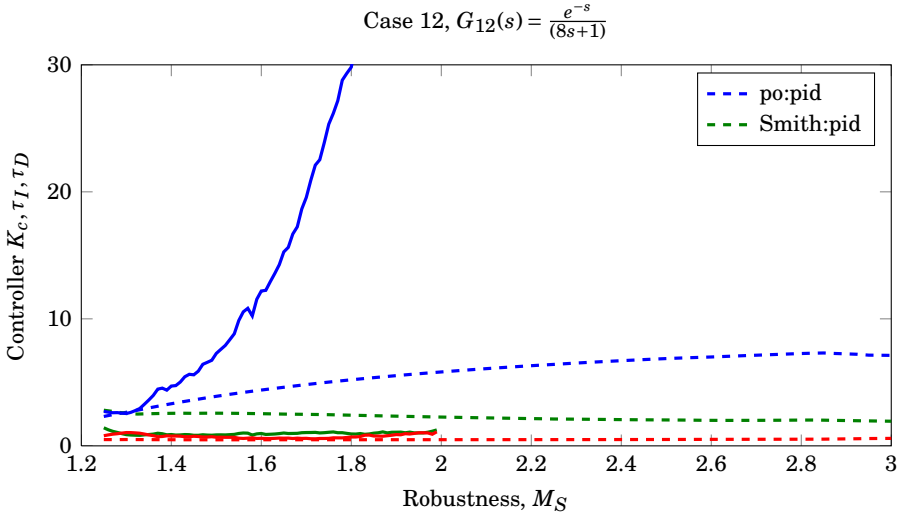


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.11 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_{11} = \frac{e^{-s}}{(s+1)}$.

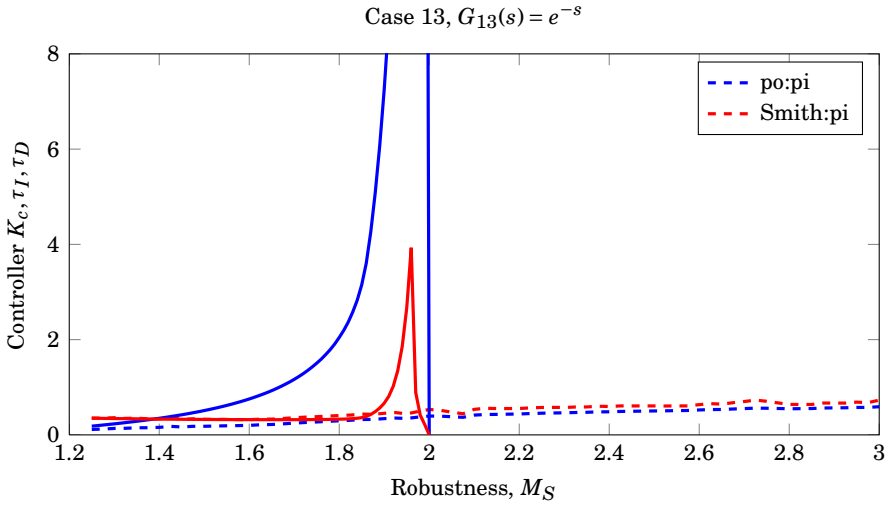


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

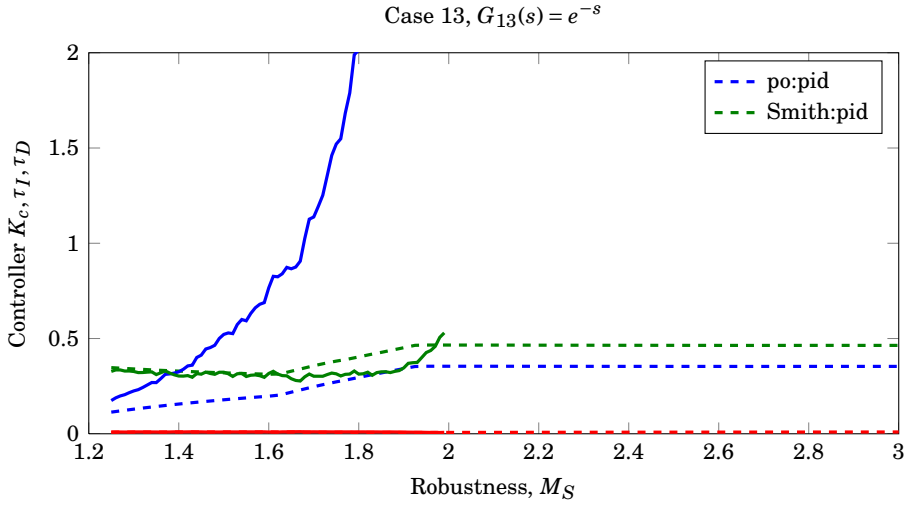


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.12 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_{12} = \frac{e^{-s}}{(8s+1)}$.

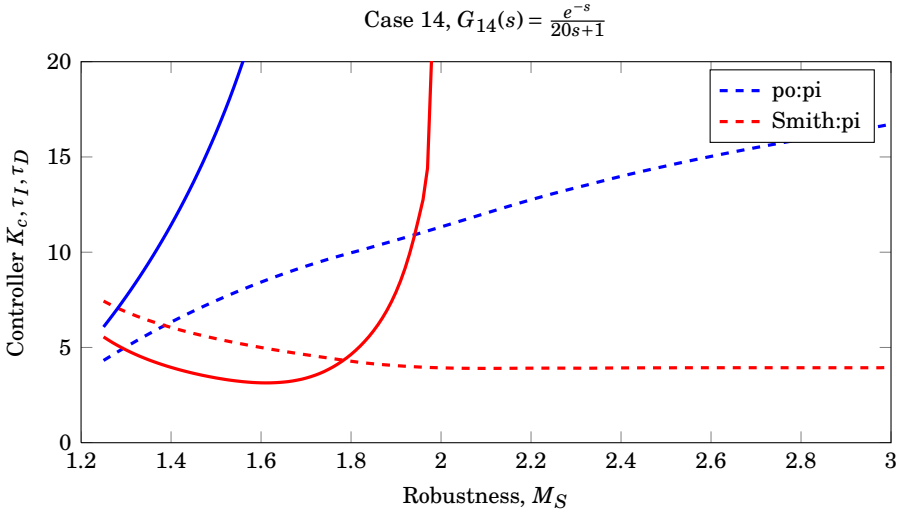


(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).

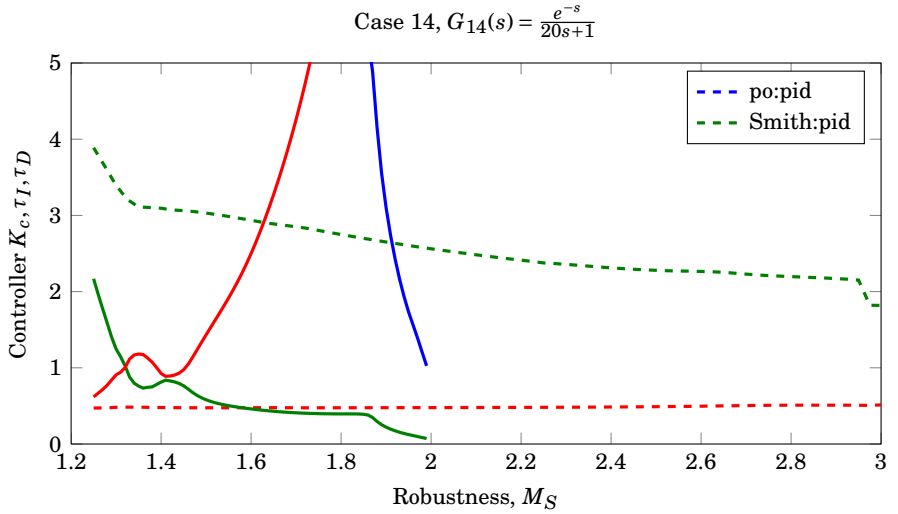


(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.13 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_{13} = e^{-s}$.



(a) Controller gain (K_c, blue color) and integral time (τ_I, red color) for the Pareto optimal PI controller (dashed line) and the Pareto optimal Smith predictor PI controller, as functions of robustness (M_S).



(b) Controller gain (K_c, blue color), integral time (τ_I, red color) and derivative time (τ_D, green color) for the Pareto optimal PID controller (dashed line) and the Pareto optimal Smith predictor PID controller, as functions of robustness (M_S).

Figure G.14 – Compared controller action for the Pareto optimal PI, PID and Smith predictor PI and PID controller for $G_{14} = \frac{e^{-s}}{20s+1}$.

APPENDIX H

DERIVATION OF SMITH PREDICTOR

H.1 Introduction

In this Chapter the Smith predictor controller is derived and analysed for first-order-plus-time-delay (FOPTD) and second-order-plus-time-delay (SOPTD) process models. The lower limit for robustness measured by the maximum peak value (M_S) of the sensitivity function (S) is derived. A small example to illustrate the limit when applying a primary P controller is then given.

Consider a controller, K, and a process, G, which yield the first-order closed loop response

$$\frac{\widetilde{K}G}{1+\widetilde{K}G} = T \stackrel{!}{=} \frac{1}{\tau_c s+1} e^{-\theta s}, \tag{H.1}$$

where the first-order response has gain $k = 1$, lag time parameter τ_c and a time delay of θ. Solving for the controller yields

$$\widetilde{K} = \frac{T}{G - GT}. \tag{H.2}$$

Assuming G has a time delay equal to T, G may be expressed as $G = G_\circ e^{-\theta s}$. Thus, the expression in Equation (H.2) can be written

$$\widetilde{K} = \frac{\frac{1}{\tau_c s+1}}{G_\circ - G_\circ \frac{1}{\tau_c s+1} e^{-\theta s}} = \frac{\frac{1}{G_\circ}\frac{1}{\tau_c s}}{1+\frac{1}{\tau_c s}\left(1-e^{-\theta s}\right)}. \tag{H.3}$$

If G is a first-order process, then the numerator of Equation (H.3) is a PI controller, and for a second-order process G the numerator is a cascade PID

Table H.1 – Processes and their respective PI or PID internal Smith predictor cascade controller given from the derivation of the Smith predictor.

Process, G		Controller, K	τ_I	τ_D
$\frac{k}{\tau s+1}e^{-\theta s}$	$\Longrightarrow$	$k\frac{\tau s+1}{\tau_c s}$	$\tau_I=\tau_c=\tau$	$\tau_D=0$
$\frac{k}{(\tau_1 s+1)(\tau_2 s+1)}e^{-\theta s}$	$\Longrightarrow$	$k\frac{(\tau_1 s+1)(\tau_2 s+1)}{\tau_c s}$	$\tau_I=\tau_c=\tau_1$	$\tau_D=\tau_2$

controller, given in Table H.1. This controller is often denoted the primary controller of the Smith predictor.

From Equation (H.3), the complete controller structure for the Smith predictor is achieved by recognising that $KG_\circ = \frac{1}{\tau_c s}$ and denoting the process model to be the Smith predictor model, $\widetilde{G}$. Then,

$$\widetilde{K} = \frac{K}{1+K\widetilde{G}_\circ\left(1+e^{-\theta_\circ s}\right)}. \tag{H.4}$$

It is noteworthy to observe that from the specification of the controller response, the sensitivity and complementary sensitivity peak values are bounded to

$$M_T - 1 \le M_S \le M_T + 1, \quad M_T = 1 \quad \text{for} \quad G = \frac{1}{\tau_c s+1}e^{-\theta s}. \tag{H.5}$$

Thus the values of M_S for the Smith predictor is limited to $M_S \le 2$. This is illustrated for a pure time delay process with a proportional controller in Section H.1.1.

H.1.1 Smith P Control Robustness Limit Example

Consider the pure time delay process $G_{10}(s) = e^{-s}$. The corresponding Smith predictor controller structure with a P-controller is

$$\widetilde{K} = \frac{K_c}{1+K_c(1-e^{-s})}, \tag{H.6}$$

where K_c is the P-controller gain and the process model $\widetilde{G} = 1$. The sensitivity function for the closed-loop system is

$$S = \frac{1}{1+G\widetilde{K}} = \frac{1+K_c(1-e^{-s})}{1+K_c}. \tag{H.7}$$

As $s = i\omega$, the following relation yields from Euler's formula

$$e^{-s} = e^{-i\omega} = \cos(\omega) - i\sin(\omega), \tag{H.8}$$

which substituted for S yields

$$S = \frac{1 + K_c - K_c\big(\cos(\omega) - i\sin(\omega)\big)}{1 + K_c}. \tag{H.9}$$

By collecting the complex terms and applying some trigonometric identities, the norm of S may be expressed as

$$\|S\| = \frac{K_c}{1 + K_c}\left[\frac{1}{K_c^2} + \frac{2}{K_c}\big(1 + \cos(\omega)\big) + 2\cos(\omega) + 2\right]^{\frac{1}{2}}. \tag{H.10}$$

As an increase in controller gain increases the response time, but reduces robustness of the system, the limit where $K_c \to \infty$ is of interest. Consequently, the M_S value for $G(s) = e^{-s}$ is

$$M_S = \max_{\omega}\left(\lim_{K_c \to \infty} \|S\|\right) = 2. \tag{H.11}$$

The complementary sensitivity for the pure time delay process can in a similar manner be shown to always be

$$M_T = \max_{\omega}\left(\lim_{K_c \to \infty} \|T\|\right) = 1, \tag{H.12}$$

independent of the frequence ω. From the relation $S + T = 1$, it is confirmed that for the pure time delay process and a Smith predictor structure with P-control, $M_S \leq 2$ for all ω.

APPENDIX I

SIMC TUNING RULES FOR THE SMITH PREDICTOR

I.1 Introduction

An analysis to find boundaries for the closed-loop time constant (τ_c) when applying the SIMC tuning rules for the purpose of tuning a Smith predictor, is performed. The SIMC tuning rules are given in Equation (I.1). Routh's stability criterion is assumed to perform a stability analysis of the Smith predictors internal non-delayed feedback system. The analysis is based on (Grimholt, 2013).

$$K_c = \frac{1}{k}\frac{\tau_1}{(\tau_c+\theta)}, \tag{I.1a}$$

$$\tau_I = \min[\tau_1, 4(\tau_c+\theta)], \tag{I.1b}$$

$$\tau_D = \tau_2. \tag{I.1c}$$

K_c is the controller gain and τ_I and τ_D are the integral and derivative time. k, τ_1, τ_2 and θ are the process gain, first and second order lag time constants and the time-delay, respectively.

The Smith predictor controller structure is derived in Appendix H and is

$$\widetilde{K} = \frac{K}{1+K\widetilde{G}_\circ\left(1+e^{-\theta_\circ s}\right)}. \tag{I.2}$$

Here, $\widetilde{K}$ is the complete Smith predictor controller structure, K denote the primary controller, and the process model is $\widetilde{G} = \widetilde{G}_\circ e^{-\theta_\circ s}$.

I.2 Analysis

Suppose the underlaying process behaviour is known such that the dynamic part of the model is perfect, then $\widetilde{G}_\circ = G_\circ$. The modelled time-delay is $\theta_\circ$ and the real process time delay is $e^{-\theta s}$. Assume the first-order Taylor approximation of $e^{-\theta s} \approx 1 - \theta s$. The closed-loop transfer function for a one-degree-of-freedom feedback scheme is

$$\mathrm{CL} = \frac{\widetilde{K}G}{1+\widetilde{K}G}. \tag{I.3}$$

Substituting $\widetilde{K}$ from Equation (I.2) and rearranging, results in the closed loop function

$$\mathrm{CL} = \frac{\widetilde{K}G}{1+\widetilde{K}G} = \frac{KG}{1+KG_\circ\left(1+e^{-\theta s}-e^{-\theta_\circ s}\right)} = \frac{KG}{1+KG_\circ(1+\underbrace{\theta_\circ s-\theta s}_{-\Delta\theta s})}, \tag{I.4}$$

where the deviation between the modelled time-delay and the true time-delay is $\Delta\theta = \theta - \theta_\circ$.

I.2.1 Lag Dominated Process

Consider a second-order-plus-time-delay (SOPTD) lag dominated nominal process model

$$G = G_\circ e^{-\theta s} = \frac{k}{(\tau_1 s+1)(\tau_2 s+1)}e^{-\theta s} \tag{I.5}$$

The SIMC rules are to control $G_\circ$, which yield the cascade controller settings

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c}, \tag{I.6a}$$

$$\tau_I = 4\tau_c, \tag{I.6b}$$

$$\tau_D = \tau_2. \tag{I.6c}$$

Then,

$$KG_\circ = \left(\frac{1}{k}\frac{\tau_1}{\tau_c}\left[\frac{4\tau_c s+1}{4\tau_c s}\right]\right)\left(\frac{k}{(\tau_1 s+1)(\tau_2 s+1)}\right) = \frac{\tau_1(4\tau_c s+1)}{4\tau_c^2(\tau_1 s+1)}. \tag{I.7}$$

Note that it does not matter whether one consider PI control of a first-order-plus-time-delay (FOPTD) process or PID control of a SOPTD process, as the derivative term cancels the second-order term of the process model.

The general stability criterion states that a closed-loop system will be stable only if all of the roots in the characteristic equation have negative real parts. Routh's stability criterion states that "a necessary and sufficient condition for all roots of the characteristic equation to have negative real parts is that all of the elements in the left column of the Routh array are positive" (Seborg, Edgar, and Mellichamp, 2004). The characteristic equation is in this context the denominator of Equation (I.4),

$$1 + KG_{\circ}(1 - \Delta\theta s). \tag{I.8}$$

Substituting Equation (I.7) into Equation (I.8), yields

$$\underbrace{\left(4\tau_1\tau_c^2 - 4\tau_1\tau_c\Delta\theta\right)}_{a_2}s^2 + \underbrace{\left(4\tau_c^2 + 4\tau_1\tau_c - \tau 1\Delta\theta\right)}_{a_1}s + \underbrace{\tau_1}_{a_0}, \tag{I.9}$$

where a_2, a_1 and a_0 are the Routh coefficients, assumed to be strictly positive. This also follows from the Routh array, and the stability criterion thus is

$$4\tau_1\tau_c^2 - 4\tau_1\tau_c\Delta\theta > 0, \tag{I.10a}$$

$$4\tau_c^2 + 4\tau_1\tau_c - \tau_1\Delta\theta > 0, \tag{I.10b}$$

$$\tau_1 > 0. \tag{I.10c}$$

Equations (I.10a) and (I.10c) yields $\tau_c > \Delta\theta > 0$. Equation (I.10b) is solved explicit in terms of τ_c, and a graphical illustration of $\tau_c > f(\tau_1, \Delta\theta)$ with $\Delta\theta, \tau_1 \in [0, 5]$ is given in Figure I.1. The blue plane is $\tau_c = \Delta\theta$, which is added for reference. It is clear that Equation (I.10b) holds if $\tau_c > \Delta\theta$.

I.2.2 Time-Delay Dominated Process

Following the approach of Chapter I.2.1, the cascade controller produced by the SIMC rules is

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c}, \tag{I.11a}$$

$$\tau_I = \tau_1, \tag{I.11b}$$

$$\tau_D = \tau_2, \tag{I.11c}$$

which yield the characteristic equation

$$(\tau_c - \Delta\theta)s + 1. \tag{I.12}$$

To achieve a stable system, $\tau_c - \Delta\theta > 0$, thus $\tau_c > \Delta\theta$.

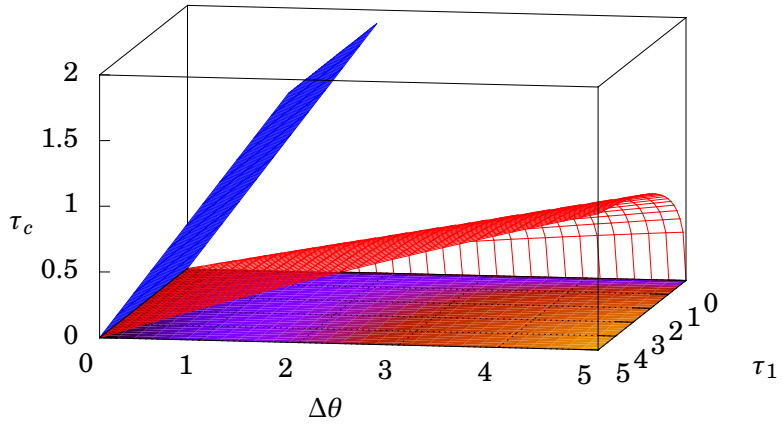


Figure I.1 – Graphical illustration of the solution of Equation (I.10b) expressed on the form $\tau_c > f(\tau_1, \Delta\theta)$ (red surface) with $\tau_c = \Delta\theta$ (blue surface) for reference.

I.3 Smith Predictor SIMC Rules Summary

The SIMC tuning rules can be used to tune a Smith predictor structure with a FOPTD or SOPTD process model. By determine the maximum time-delay modelling error, $\Delta\theta$, and letting the closed-loop time constant $\tau_c > \Delta\theta$, the SIMC tuning rules for a Smith predictor is

$$K_c = \frac{1}{k}\frac{\tau_1}{\tau_c}, \tag{I.13a}$$

$$\tau_I = \min[\tau_1, 4\tau_c], \tag{I.13b}$$

$$\tau_D = \tau_2. \tag{I.13c}$$

APPENDIX J

CONTROLLER SOLUTIONS

J.1 Alternative Parallel Controller

The controller transfer function in the Laplace domain is

$$\frac{u}{e_i} = \mathsf{K}\left((1-\mathsf{I}-\mathsf{D}) + \frac{\mathsf{I}}{s} + \mathsf{D}s\right). \tag{J.1}$$

By gathering of the terms, the controller transfer function takes the form of a second order polynomial in s. Solving for for s yields

$$s = \frac{-(1-\tau_I-\tau_D) \pm \sqrt{(1-\tau_I-\tau_D)^2 - 4\tau_I\tau_D}}{2\tau_D}, \tag{J.2}$$

where squared term $(\tau_I-\tau_D-1)^2-4\tau_D$ represents a parabolic cylinder. $f(\tau_I,\tau_D)$. A plot of $f(\tau_I,\tau_D),(\tau_I,\tau_D) \in [0,1]$ is given in Figure J.1, and it is evident that the given domain yields negative solutions. Thus the controller solutions may be complex, that is $s \in \mathbb{C}$.

J.2 Cascade Controller

The controller transfer function in the Laplace domain is

$$\frac{u}{e_i} = K_c\left(\frac{\tau_I s+1}{\tau_I s}\right)(\tau_D s+1). \tag{J.3}$$

From the same arguments as in Section J.1, the square term in the solution of the second order polynomial contains a parabolic cylinder, $g(\tau_I,\tau_D) = (\tau_I-\tau_D)^2$. A plot of $g(\tau_I,\tau_D),(\tau_I,\tau_D) \in [-1,1]$ is given in Figure J.2. As $g \geq 0 \forall (\tau_I,\tau_D)$, the solutions of s are always real, that is, $s \in \mathbb{R}$.

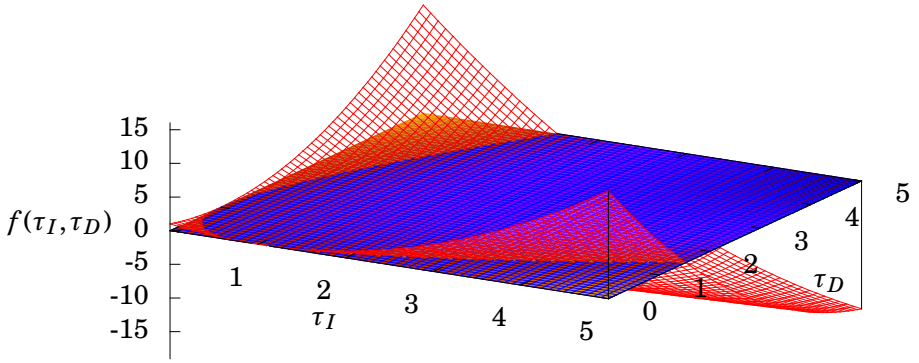


Figure J.1 – Plot of the parabolic cylinder $f(\tau_I,\tau_D),(\tau_I,\tau_D) \in [0,1]$.

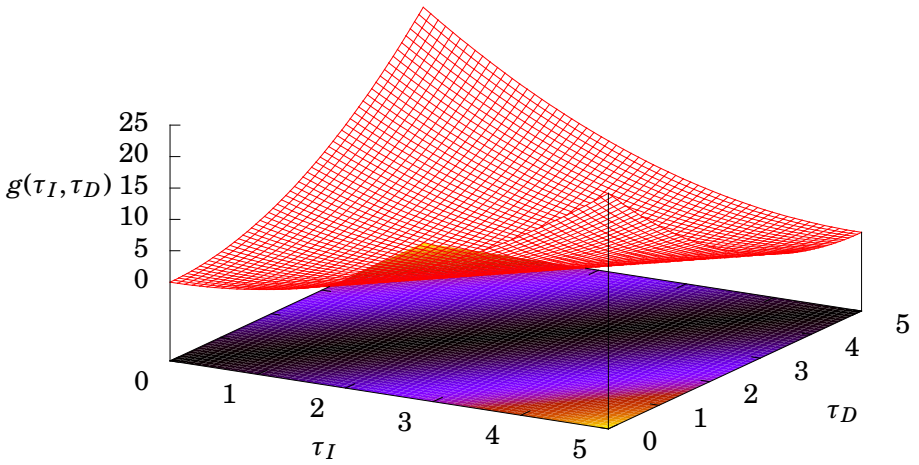


Figure J.2 – Plot of the parabolic cylinder $g(\tau_I,\tau_D),(\tau_I,\tau_D) \in [-1,1]$.

Appendix K

Controller Conversion

The different parameterisations of the controllers yield different tuning. Even so, the controllers should be equal independent of the parameterisation chosen. This is not the case when the zeros of the parallel controllers are complex, as the cascade controller parameterisation doesn't allow for complex controller solutions, see Appendix J.

To transform one controller into the other, the controller parameterisations are compared. The following chapters clarifies the conversion rules.

Parallel to alternative parallel parameterisation Conversion between the tuning parameters for the standard parallel PID-controller and the alternative configuration is done by

$$K_c^\star = \mathsf{K}(1-\mathsf{I}-\mathsf{D}), \tag{K.1a}$$

$$\tau_I^\star = \frac{1-\mathsf{I}-\mathsf{D}}{\mathsf{I}}, \tag{K.1b}$$

$$\tau_D^\star = \frac{\mathsf{D}}{1-\mathsf{I}-\mathsf{D}}. \tag{K.1c}$$

Conversion from Cascade Parameterisation The basis for the calculations are the equality of the alternative parallel parameterisation and the cascade parameterisations

$$\mathsf{K}\left((1-\mathsf{I}-\mathsf{D})+\frac{\mathsf{I}}{s}+\mathsf{D}s\right) = K_c\left(\frac{\tau_I s+1}{\tau_I s}\right)(\tau_D s+1), \tag{K.2}$$

where $(\mathsf{K}, \mathsf{I}, \mathsf{D})$ and (K_c, τ_I, τ_D) are controller gain and integral and derivative time for the alternative parallel and cascade parameterisations, respectively. By solving Equation (K.2) with respect to the alternative parallel parameters, the following relations yield:

$$\mathsf{K} = K_c \left(1 + \frac{\tau_D + \tau_I \tau_D + 1}{\tau_I}\right) \tag{K.3a}$$

$$\mathsf{I} = \frac{K_c}{\mathsf{K}} \frac{1}{\tau_I} \tag{K.3b}$$

$$\mathsf{D} = \frac{K_c}{\mathsf{K}} \tau_D \tag{K.3c}$$

Appendix L

Matlab Main Script

The main MATLAB script used to perform all computations in this thesis were too long for LaTeX to handle[a], but the optimisation routine is listed in Listing L.1. The basic idea of the main script is to provide the target (Pareto optimisation, SIMC computation, disturbance computation, etc.) along with the controller structure (P, PI or PID) and its parameterisation. The "Chriss" parameterisation in the program correspond to the alternative parallel controller described in Chapter 2.3. Further, the derivative filter and the filter constant should be set, along with several other parameters. Together they form the `parm` parameter struct, which guides MATLAB through several of the functions used. For instance, the initial value function, `initValues.m`, is using the parameters to navigate through several `switch` statements to find the correct initial values for a given problem.

Listing L.1 – Main MATLAB script

```
1 while exitFlag
2 for Ms_s = Ms_span
3   parm.Ms_s = Ms_s ;                % Updating Ms setpoint
        value
4   Jfun      = @(ctrl)costFun(ctrl,parm) ;        % Updating
        cost
5   cFun      = @(ctrl)conFun(ctrl,parm);   % Updating
        constraints
```

a) Well, actually it's limitations in the framed evnironment in `mcode` that's the source of the problem.

```
6   initial(i,:) = ctrl ; % Saving starting point for
        optimization
7   runOpt = true ;
8   if ctrlr == 3 % Just for P-control
9     sNum = 5 ;
10    % try
11    %     tic ;
12    %     [ctrl J optFlag] = fmincon(Jfun,ctrl,[],[],[],...
13    %                                [],lb,[],cFun,opt) ;
14    %                     % Optimizing from last optimal point
15    %     sTime(i) = toc ;       % Ending CPU time measurement
16    %     if optFlag < 1
17    %         sNum = 5 ;
18    %     end
19    % catch
20    %     sNum = 5
21    % end
22    if sNum == 5
23        tic ;
24        MsFun = @(ctrl) parm.Ms_s - Ms(ctrl,parm) ;
25        opt = optimset('Display','off',...
26                       'UseParallel','Always') ;
27        [ctrl dmy optFlag] = fzero(MsFun, ctrl, opt) ;
28        J = costFun(ctrl, parm) ;
29        % [ctrl J optFlag] = fmincon(Jfun,ctrl,[],[],[],...
30        %                            [],lb,[],cFun,opt);
31        sTime(i) = sTime(i) + toc ;          % Ending CPU time
32    end % if sNum
33  else
34    if runGradientfree
35      runOpt = false ;            % No standard opt. routine
36      Kc = ctrl(1); ID = ctrl(2:end);
37      Jfun = @(ID)noGrad([Kc ID], parm) ;
38      try
39          tic ;
40          [ID J optFlag] = ...
41                           fminsearch(Jfun, ID, sOpt) ;
42          sTime(i) = toc ;
43      catch
44          pFlag = false ;
45      end % try
46      if pFlag
47          fprintf('Success! (%.2f s)\n', sTime(i))
48          MsFun = @(Kc) Ms_s - Ms([Kc ID], parm) ;
49          tic
50          Kc = fzero(MsFun, Kc) ;
51          sTime(i) = sTime(i) + toc ;
52          ctrl = [Kc ID] ;
53      else
```

```
54          % Running standard opt routine.  Keep  the  faith!
55          pFlag = true ;
56          runOpt = true ;
57        end % pFlag
58      end % runGradientfree
59      while runOpt
60        while sNum < 3
61          try
62            if runTomlab
63              tic ;
64              Prob = ProbDef ;
65              Prob.Warning = 0;    % Turning off warnings.
66              % Prob.Solver.Tomlab = 'npsol' ;
67              [ctrl J optFlag] = fmincon(@Jfunc, ...
68                  ctrl,[],[], [],[],lb,[],@conFunc,...
69                  opt,Prob) ;
70              sTime(i) = toc ;
71            else % if not Tomlab
72              % ctrl = simc(G.theta,parm) ; % setting simc as
                    initial point
73              tic ;
74              [ctrl J optFlag] = fmincon(Jfun,ctrl,[],[],[],...
75                                         [],lb,[],cFun,opt);
76                              % Optimizing from last optimal point
77              sTime(i) = toc ;    % Ending CPU time measurement
78            end % tomLab
79            if optFlag < sFlags{sNum}
80              fprintf('%s failed. Error: %s \n', ...
81                      solvers{sNum}, lasterr)
82              sNum = sNum + 1 ;
83            elseif min(ctrl) < 1e-3 && modelId < 10
84              name = {'Kc' 'I' 'D'} ;
85              fprintf('%s failed. ', solvers{sNum})
86              fprintf('Error: %s approx zero.\n', ...
87                      name{find(ctrl==min(ctrl))})
88              % Hot start from previous working values
89              if i == 1; ctrl = init(3,:) ;
90              else; ctrl = minTune(i-1,:); end;
91              sNum = sNum + 1 ;
92            else % this is done anyway
93              tic ;
94              [ctrl J optFlag] = fmincon(Jfun,ctrl,...
95                                         [],[],[],[],lb,[],...
96                                         cFun,opt);
97                                % Optimizing from last optimal point
98              sTime(i) = sTime(i) + toc ;
99              if optFlag < sFlags{sNum}
100               fprintf('%s failed. ', ...
101                       solvers{sNum})
```

```
102               sNum = sNum + 1 ;
103             else
104               runOpt = false ;      % Opt success. Ending while
105               break
106             end
107           end % optFlag < sFlags
108         catch
109           fprintf('%s failed. Error: %s',solvers{sNum},...
110                                                   lasterr)
111           sNum = sNum + 1 ;
112         end
113       end % while
114         if sNum == 3
115           fprintf('Using %s... ', solvers{sNum})
116           Kc = ctrl(1); ID = ctrl(2:end);
117           Jfun = @(ID)noGrad([Kc ID], parm) ;
118           try
119               tic ;
120               [ID J optFlag] = ...
121                               fminsearch(Jfun, ID, sOpt) ;
122               extrat = toc ;
123               sTime(i) = sTime(i) + extrat ;
124           catch
125               pFlag = false ;
126               fprintf('fminsearch failed.\nGiving up.\n')
127               fprintf('Error message:\n'); disp(lasterr)
128               runOpt = false ;
129               exitFlag = false ;
130               break
131           end % try
132           if pFlag
133             fprintf('Success! (%.2f s)\n', extrat)
134             sTime(i) = sTime(i) + extrat ;
135           else
136             pFlag = true ;
137           end % if pFlag
138           MsFun = @(Kc) Ms_s - Ms([Kc ID], parm) ;
139           Kc = fzero(MsFun, Kc) ;
140           ctrl = [Kc ID] ;
141           Jfun = @(ctrl)costFun(ctrl, parm) ;   % Updating cost
142           cFun = @(ctrl)conFun(ctrl, parm);    % Updating cons.
143           try
144             fprintf('Second run sqp start... ')
145             tic ;
146             [ctrl J optFlag] = fmincon(Jfun, ctrl, [], ...
147                                        [], [], [], lb,...
148                                        [], cFun, opt) ;
149             extrat = toc ;
150           catch
```

```
151             pFlag = false ;
152             fprintf('Second run sqp failed\n')
153             ctrl = [Kc ID];
154           end
155           if pFlag
156             tic ;
157             [ctrl J optFlag] = fmincon(Jfun, ctrl, [], ...
158                                        [], [], [], lb,...
159                                        [], cFun, opt) ;
160             extrat = extrat + toc ;
161             fprintf('Success! (%.2f s)\n', extrat)
162             sTime(i) = sTime(i) + extrat ;
163           else
164             pFlag = true ;
165           end % if pFlag
166           C = controller(ctrl, parm) ;
167           MsCalc = Ms(ctrl, parm) ;
168           if (MsCalc > Ms_s + 1e-4)||(MsCalc < Ms_s - 1e-4)
169           fprintf('Ms values differ at control calculation.\n')
170             fprintf('Ms_s = %0.5f, Ms = %.5f\n', Ms_s, MsCalc)
171           end
172           runOpt = false ; % Opt success. Ending while.
173         end % if sNum == 3
174       end % while runOpt
175     end % if ctrlr == 3
176   if exitFlag == false; break; end;
177   C            = controller(ctrl, parm) ;           %
        Controller
178   minTune(i,:) = ctrl ;              % Storing controller
        tuning
179   Msc(i)       = Ms(ctrl, parm) ;          % Calculating Ms
        value
180   Mtc(i)       = Mt(ctrl, parm) ;        % Corresponding Mt-
        value
181   ctrlRoot{i}  = zero(C.tf) ;             % Find controller
        zeros
182   IAEtune(i,:) = iae(ctrl, parm) ;           % Saving IAE
        weights
183   Jtune(i,:)   = J    ;             % Storing cost function
        value
184   flags(i,:)   = optFlag ;                    % Storing exit
        flag
185   timeleft     = sum(sTime)/i*(length(Ms_span)-i)/60 ;
186                  % Estimation of time left for computations [min
                       ]
187   if ctrlr == 1
188     fprintf('%.2f\t%.2f\t%.2f\t%.2f\t%.3f\t%.3f\t%d\t%d\t%0.2f\
          t\t%0.2f',...
189     J,ctrl,Ms_s,Msc(i), optFlag, ...
```

```
190     length(Ms_span)-i,sTime(i),timeleft)
191   elseif ctrlr == 2
192     fprintf('%.2f\t%.2f\t%.2f\t%.3f\t%.3f\t%d\t%d\t%0.2f\t\t%0
            .2f',...
193     J,ctrl,Ms_s,Msc(i), optFlag, ...
194     length(Ms_span)-i,sTime(i),timeleft)
195   else
196     fprintf('%.2f\t%.2f\t%.3f\t%.3f\t%d\t%d\t%0.2f\t\t%0.2f',
            ...
197     J,ctrl,Ms_s,Msc(i),optFlag, ...
198     length(Ms_span)-i,sTime(i),timeleft)
199   end
200   fprintf('\n')
201   i = i + 1 ;                              % Incrementing counter
202   sNum = sNum_s ;
203 end % Ending for-loop
204 exitFlag = false ;
205 end % Ending while-loop
```

APPENDIX M

MATLAB SUPPORT FUNCTIONS

M.1 Optimisation Constraints Function: `conFun.m`

Function computing and returning the constraints needed for `fmincon` optimisation. `conFun` accepts a vector with controller tunings (`ctrl`) and the parameter struct (`parm`) as arguments and returns inequality and equality constraints for the problem.

Listing M.1 – Function providing constraints to `fmincon`.

```
1  %% Header
2  % Purpose:  Creating the process transfer function G (struct).
3  % Author:   Axel Lodemel Holene (based on work done by Martin
4  %           Foss (2012)).
5  % E-mail:   axel.holene@gmail.com
6  % Date:     February 2013
7  % About:    h       nonlinear inequality constraints
8  %           h_eq    nonlinear equality constraints
9  %           Ms_s    Setpoint value for Ms
10 %
11
12 function [h h_eq] = conFun(ctrl, parm)
13     h    = [] ;            % No nonlinear inequality
           constraints
14     G    = parm.G ;
15     C    = controller(ctrl, parm) ;
16     h_eq = parm.Ms_s - max(abs(freqresp(1/(1 + G.tf*C.tf), ...
```

```
17                              logspace(-4,4,40000)))) ;
18 end
```

M.2 Controller Function: `controller.m`

Function computing the controller transfer function. `controller.m` accepts either one single argument, the `parm` struct, or a combination of controller tuning vector (`ctrl`) and the parameter struct. The first of the two are added for convenience when finding SIMC tunings, as an initial set of tuning parameters is unnecessary for this purpose. To navigate to the correct controller configuration, the parameters `cType`, `ctrlr` and `cNum` are used to navigate through a set of nested `switch` statements. The returned variable is a struct with the controller tunings and the controller transfer function in separate fields. If a Smith predictor controller is generated, the struct also will contain the transfer function for the controller within the Smith predictor controller structure.

Listing M.2 – Controller function for generating struct with controller tunings and transfer functions as separate fields.

```
1  %% Header
2  % Purpose:  Creating the process transfer function G (struct).
3  % Author:   Axel Lodemel Holene (based on work done by Martin
4  %           Foss (2012)).
5  % E-mail:   axel.holene@gmail.com
6  % Date:     February 2013
7  % About:    h       nonlinear inequality constraints
8  %           h_eq    nonlinear equality constraints
9  %           Ms_s    Setpoint value for Ms
10 %
11
12 function [h h_eq] = conFun(ctrl, parm)
13     h    = [] ;            % No nonlinear inequality
           constraints
14     G    = parm.G ;
15     C    = controller(ctrl, parm) ;
16     h_eq = parm.Ms_s - max(abs(freqresp(1/(1 + G.tf*C.tf), ...
17                             logspace(-4,4,40000)))) ;
18 end
```

M.3 Cost Function: costFun.m

Function for computing the cost scalar of the objective function for the optimisation problem are given in Listing M.3. The function accepts a controller tuning vector and the parameter struct as arguments, and returns the weighted cost.

Listing M.3 – Cost function calculation routine.

```
%% Header
% Purpose:  Cost function
% Author:   Axel Lodemel Holene
% E-mail:   axel.holene@gmail.com
% Date:     February 2013
% About:    J = [Jw_y Jw_d]*[IAE_y IAE_d]./[IAEw_y IAEw_d]
%           ctrl =          struct of controller starting points
%           Jweights   ([Jw_y Jw_d])       Cost function weights
%           IAEweights ([IAEw_y IAEw_d])           IAE weighting

function J = costFun(ctrl, parm)
    iae0 = iae(ctrl,parm) ;
    J = parm.Jweights*(iae0./parm.IAEweights) ;
end
```

M.4 Half Rule Implementation: halfrule.m

The `halfrule` function given in Listing M.4 applies the "half rule" to a second order system and returns a struct with the reduced and original system parameters and transfer functions.

Listing M.4 – "Half rule" system reduction of second order systems.

```
%% Header
% Purpose:  Using the half-rule to reduce second order tfs to
%           first order tfs
% Author:   Axel Lodemel Holene.
% E-mail:   axel.holene@gmail.com
% Date:     March 2013
% About:    G: second order model
%           H: reduced first order model

function H = halfrule(G)
```

```
11      s = zpk('s') ;
12      if G.case > 9           % No half rule for 1st order
            systems
13          H = G ;
14      else
15          H.ordG = G ;
16          H.k = G.k ;
17          H.tau1 = G.tau1 + G.tau2/2 ;
18          H.theta = G.theta + G.tau2/2 ;
19          H.tfnd = H.k/(H.tau1*s + 1) ;
20          H.tf = H.tfnd*exp(-H.theta*s) ;
21      end
22  end
```

M.5 Integrated Absolute Error Calculation: `iae.m`

The `iae` function computes the absolute integrated error given a set of controller tunings and the parameter struct, by applying either the MATLAB native `step` function or the SIMULINK block diagram given in Appendix N.

Listing M.5 – Integrated absolute error response for a given controller tuning and set of parameters.

```
1  %% Header
2  % Purpose:  Function calculating the integrated absolute error
3  %           as a function of controller tuning.
4  % Author:   Axel Lodemel Holene
5  % E-mail:   axel.holene@gmail.com
6  % Date:     February 2013
7  % About:
8
9  function iae0 = iae(ctrl, parm)
10     G = parm.G ;
11     C = controller(ctrl, parm);
12     try
13         S = 1/(1+G.tf*C.tf) ;   % Output feedback loop tf(do->e
               )
14         Sd = -G.tf/(1+G.tf*C.tf);% Input feedback loop tf(di->e
               )
15     catch
16         fprintf('This is iae. Error in loop TFs.\n')
17     end
18     %% Output response
19     if parm.simCost == 0 % Don't use Simulink
20         try
21             [e t] = step([S; Sd], parm.simTime) ;
```

```
22              iae0 = trapz(t, abs(e))' ;
23          catch
24              iae0 = [100; 100] ;% Penalize bad tuning from
                    solver
25              if parm.dispErrors
26                  fprintf('This is costFun catch. ')
27                  fprintf('Error in step function: %s\n', ...
28                          lasterr) ;
29              end
30          end
31      else % Use simulink
32          try
33              simOut = sim('simulink/iaeBlock', ...
34                   'SrcWorkspace', 'current') ;
35              iae_y = get(simOut,'iae_y') ;
36              iae_d = get(simOut,'iae_d') ;
37          catch
38              iae_y = 100 ;       % Penalize bad tuning from
                    solver
39              iae_d = 100 ;
40              if parm.dispErrors
41                  fprintf('This is costFun catch. ')
42                  fprintf('Error in simulink simulation: %s\n',
                        ...
43                          lasterr) ;
44              end
45          end
46          iae0 = [iae_y(end); iae_d(end)] ;
47      end
48  end
```

M.6 Initial Values for Optimisation: initValues.m

The function is accepting the parameter struct as argument, returning the corresponding initial values.

Listing M.6 – Initial values function

```
1  %% Header
2  % Purpose:  Returning initvalues for optimization from modelId
3  % Author:   Axel Lodemel Holene
4  % E-mail:   axel.holene@gmail.com
5  % Date:     13. February 2013
6  % About:    init = [iae_y ; iae_d ; pareto]
7
8  function init = initValues(parm)
```

```
9   modelId = parm.modelId ;
10  cType   = parm.cType ;
11  ctrlr   = parm.ctrlr ;
12  cNum    = parm.cNum ;
13
14  switch cType
15    case 1 % PO
16      switch ctrlr
17        case 1 % PID
18          switch cNum
19            case 1 % Chriss values
20              switch modelId
21                case 1
22                  init = [1.8486 0.2976 0.2321 ;
23                          1.9974 0.3233 0.2406 ;
24                          1.1194 0.3275 0.2638 ] ;
25                case 2
26                  init = [2.0566 0.2547 0.2699 ;
27                          2.2294 0.2809 0.2795 ;
28                          1.2524 0.2873 0.3027 ] ;
29                case 3
30                  init = [1.7468 0.3312 0.1996 ;
31                          1.8863 0.3572 0.2076 ;
32                          1.0546 0.3603 0.2309 ] ;
33                case 4
34                  init = [4.4974 0.3266 0.1800 ;
35                          5.2921 0.4244 0.1505 ;
36                          2.6581 0.4077 0.1645 ] ;
37                case 5
38                  init = [3.4497 0.2689 0.2434 ;
39                          3.8393 0.3371 0.2224 ;
40                          2.0308 0.3374 0.2361 ] ;
41                case 6
42                  init = [6.3862 0.3774 0.1251 ;
43                          9.0004 0.5500 0.0867 ;
44                          4.0203 0.4907 0.1032 ] ;
45                case 7
46                  init = [5.9076 0.3270 0.1773 ;
47                          7.4737 0.4608 0.1316 ;
48                          3.4348 0.4083 0.1532 ] ;
49                case 8
50                  init = [4.5005 0.2719 0.2371 ;
51                          5.1083 0.3579 0.2007 ;
52                          2.5717 0.3410 0.2186 ] ;
53                case 9
54                  init = [8.4814 0.3796 0.1219 ;
55                          13.0195 0.5817 0.0740 ;
56                          5.4169 0.5027 0.0941 ] ;
57                case 10
```

```
58                    init = [0.8216 0.0003 0.2429 ;
59                            1.1    0.2    0.25   ;
60                            0.5    0.1    0.3    ] ;
61                  case 11
62                    init = [1.8106 0.3767 0.1508 ;
63                            2.0336 0.4223 0.1568 ;
64                            1.3500 0.4500 0.1350 ] ;
65                  case 12
66                    init = [7.3804 0.0842 0.2155 ;
67                            9.7846 0.2457 0.2261 ;
68                            4.6914 0.1757 0.2437 ] ;
69                  case 13
70                    init = [0.8378 0.7484 0.0044 ;
71                            0.8237 0.7521 0.0015 ;
72                            0.8250 0.7576 0      ] ;
73                  case 14
74                    init = [10.5351 0.0475 0.0019 ;
75                            23.1852 0.2174 0.2357 ;
76                            10.98 0.14  0.26 ] ;
77                end % end switch modelId
78              case 2 % Parallel values
79                switch modelId
80                  case 1
81                    init = [0.8540 0.4185 ;
82                            0.8287 0.4377 ;
83                            0.4180 0.4398 ] ;
84                  case 2
85                    init = [0.8647 0.3781 ;
86                            0.8324 0.4001 ;
87                            0.4158 0.4039 ] ;
88                  case 3
89                    init = [0.8803 0.4439 ;
90                            0.8579 0.4622 ;
91                            0.4338 0.4640 ] ;
92                  case 4
93                    init = [1.7960 0.4054 ;
94                            1.7024 0.4602 ;
95                            0.8020 0.4633 ] ;
96                  case 5
97                    init = [1.3179 0.3681 ;
98                            1.2454 0.4088 ;
99                            0.5983 0.4145 ] ;
100                 case 6
101                   init = [2.7077 0.4360 ;
102                           2.6025 0.5145 ;
103                           1.2037 0.5038 ] ;
104                 case 7
105                   init = [2.1404 0.4022 ;
106                           2.0171 0.4645 ;
```

```
107                         0.9347 0.4627 ] ;
108                 case 8
109                   init = [1.5827 0.3608 ;
110                         1.4804 0.4117 ;
111                         0.6990 0.4152 ] ;
112                 case 9
113                   init = [3.2223 0.4276 ;
114                         3.0833 0.5222 ;
115                         1.4053 0.5020 ] ;
116               end % end switch modelId
117             case 3 % AltParallel values
118               switch modelId
119                 case 1
120               end % end switch modelId
121             case 4 % Cascade values
122               switch modelId
123                 case 1
124                   init = [1 1 1 ;
125                         1 1 1 ;
126                         0.2106 0.7913 0.7913] ;
127                 case 2
128                   init = [1 1 1 ;
129                         1 1 1 ;
130                         0.2349 0.9275 0.9287] ;
131                 case 3
132                   init = [1 1 1 ;
133                         1 1 1 ;
134                         0.2018 0.7017 0.7018] ;
135                 case 4
136                   init = [1 1 1 ;
137                         1 1 1 ;
138                         0.5570 0.6607 0.6605] ;
139                 case 5
140                   init = [1 1 1 ;
141                         1 1 1 ;
142                         0.4077 0.8463 0.8464] ;
143                 case 6
144                   init = [1 1 1 ;
145                         1 1 1 ;
146                         0.8140 0.4777 0.4775] ;
147                 case 7
148                   init = [1 1 1 ;
149                         1 1 1 ;
150                         0.7461 0.6356 0.6351] ;
151                 case 8
152                   init = [1 1 1 ;
153                         1 1 1 ;
154                         0.5475 0.8286 0.8284] ;
155                 case 9
```

```
156                   init = [1 1 1 ;
157                           1 1 1 ;
158                           1.0920 0.4499 0.4492] ;
159                 case 10
160                   init = [1 1 1;
161                           1 1 1;
162                           0.3258 5.8437 0.4221 ] ;
163                 case 11
164                   init = [1 1 1;
165                           1 1 1;
166                           0.4503 1.0024 0.4198 ] ;
167                 case 12
168                   init = [1 1 1;
169                           1 1 1;
170                           2.7234 3.3003 0.4207 ] ;
171                 case 13
172                   init = [0.2 0.32 0 ;
173                           0.2 0.32 0 ;
174                           % 2.7235 3.3011 0.4206 ] ;
175                           0.12  0.35  0.01 ] ;
176                 case 14
177                   init = [1 1 1;
178                           1 1 1;
179                           5.8980 3.8552 0.4780 ] ;
180               end % end switch modelId
181             end % end switch cNum
182           case 2 % PI
183             switch cNum
184               case 1    % Chriss
185                 switch modelId
186                   case 1
187                     init = [0.8540 0.4185 ;
188                             0.8287 0.4377 ;
189                             0.4180 0.4398 ] ;
190                   case 2
191                     init = [0.8647 0.3781 ;
192                             0.8324 0.4001 ;
193                             0.4158 0.4039 ] ;
194                   case 3
195                     init = [0.8803 0.4439 ;
196                             0.8579 0.4622 ;
197                             0.4338 0.4640 ] ;
198                   case 4
199                     init = [1.7960 0.4054 ;
200                             1.7024 0.4602 ;
201                             0.8020 0.4633 ] ;
202                   case 5
203                     init = [1.3179 0.3681 ;
204                             1.2454 0.4088 ;
```

```
205                         0.5983 0.4145 ] ;
206                 case 6
207                   init = [2.7077 0.4360 ;
208                           2.6025 0.5145 ;
209                           1.2037 0.5038 ] ;
210                 case 7
211                   init = [2.1404 0.4022 ;
212                           2.0171 0.4645 ;
213                           0.9347 0.4627 ] ;
214                 case 8
215                   init = [1.5827 0.3608 ;
216                           1.4804 0.4117 ;
217                           0.6990 0.4152 ] ;
218                 case 9
219                   init = [3.2223 0.4276 ;
220                           3.0833 0.5222 ;
221                           1.4053 0.5020 ] ;
222                 case 10
223                   init = [1 0.1 ;
224                           0.6 0.2 ;
225                           0.24 0.07 ] ;
226                 case 11
227                   init = [1.04 0.47 ;
228                           1.02 0.49 ;
229                           0.51 0.49 ] ;
230                 case 12
231                   init = [4.5 0.11 ;
232                           4.23 0.21 ;
233                           2.14 0.16 ] ;
234                 case 13
235                   init = [0.19 3.14 ;
236                           0.1 2 ;
237                           0.44 0.74 ] ;
238                 case 14
239                   init = [1 1 ;
240                           1 1 ;
241                           4.8989 0.1186 ] ;
242               end % end switch modelId
243             case 2
244               switch modelId
245                 case 1
246                   init = [0.8540 0.4185 ;
247                           0.8287 0.4377 ;
248                           0.4180 0.4398 ] ;
249                 case 2
250                   init = [0.8647 0.3781 ;
251                           0.8324 0.4001 ;
252                           0.4158 0.4039 ] ;
253                 case 3
```

```
254                     init = [0.8803 0.4439 ;
255                             0.8579 0.4622 ;
256                             0.4338 0.4640 ] ;
257                   case 4
258                     init = [1.7960 0.4054 ;
259                             1.7024 0.4602 ;
260                             0.8020 0.4633 ] ;
261                   case 5
262                     init = [1.3179 0.3681 ;
263                             1.2454 0.4088 ;
264                             0.5983 0.4145 ] ;
265                   case 6
266                     init = [2.7077 0.4360 ;
267                             2.6025 0.5145 ;
268                             1.2037 0.5038 ] ;
269                   case 7
270                     init = [2.1404 0.4022 ;
271                             2.0171 0.4645 ;
272                             0.9347 0.4627 ] ;
273                   case 8
274                     init = [1.5827 0.3608 ;
275                             1.4804 0.4117 ;
276                             0.6990 0.4152 ] ;
277                   case 9
278                     init = [3.2223 0.4276 ;
279                             3.0833 0.5222 ;
280                             1.4053 0.5020 ] ;
281                   case 10
282                     init = [1 0.1 ;
283                             0.6 0.2 ;
284                             0.24 0.07 ] ;
285                   case 11
286                     init = [1.04 0.47 ;
287                             1.02 0.49 ;
288                             0.51 0.49 ] ;
289                   case 12
290                     init = [4.5 0.11 ;
291                             4.23 0.21 ;
292                             2.14 0.16 ] ;
293                   case 13
294                     init = [0.19 3.14 ;
295                             0.1 2 ;
296                             0.44 0.74 ] ;
297                 end % end switch modelId
298               case 3
299                 % Option never used.
300               case 4
301                 switch modelId
302                   case 1
```

```
303                   init = [0.8540 0.4185 ;
304                           0.8287 0.4377 ;
305                           0.4180 0.4398 ] ;
306                 case 2
307                   init = [0.8647 0.3781 ;
308                           0.8324 0.4001 ;
309                           0.4158 0.4039 ] ;
310                 case 3
311                   init = [0.8803 0.4439 ;
312                           0.8579 0.4622 ;
313                           0.4338 0.4640 ] ;
314                 case 4
315                   init = [1.7960 0.4054 ;
316                           1.7024 0.4602 ;
317                           0.8020 0.4633 ] ;
318                 case 5
319                   init = [1.3179 0.3681 ;
320                           1.2454 0.4088 ;
321                           0.5983 0.4145 ] ;
322                 case 6
323                   init = [2.7077 0.4360 ;
324                           2.6025 0.5145 ;
325                           1.2037 0.5038 ] ;
326                 case 7
327                   init = [2.1404 0.4022 ;
328                           2.0171 0.4645 ;
329                           0.9347 0.4627 ] ;
330                 case 8
331                   init = [1.5827 0.3608 ;
332                           1.4804 0.4117 ;
333                           0.6990 0.4152 ] ;
334                 case 9
335                   init = [3.2223 0.4276 ;
336                           3.0833 0.5222 ;
337                           1.4053 0.5020 ] ;
338                 case 10
339                   init = [1      0.1 ;
340                           0.6    0.2 ;
341                           0.2101 10.7132 ] ;
342                 case 11
343                   init = [1.04 0.47 ;
344                           1.02 0.49 ;
345                           0.51 0.49 ] ;
346                 case 12
347                   init = [4.5   0.11 ;
348                           4.23  0.21 ;
349                           0.5   0.8 ] ;
350                 case 13
351                   init = [0.19 3.14 ;
```

```
352                             0.1   2 ;
353                             0.15 0.40 ] ;
354                   case 14
355                     init = [1.04 0.47 ;
356                             1.02 0.49 ;
357                             0.51 0.49 ] ;
358                 end % end switch modelId
359             end % end switch cNum
360           case 3 % P
361             switch modelId
362               case 1
363                 init = [1 ; 1 ; 0.4232] ;
364               case 2
365                 init = [1 ; 1 ; 1] ;
366               case 3
367                 init = [1 ; 1 ; 1] ;
368               case 4
369                 init = [1 ; 1 ; 1] ;
370               case 5
371                 init = [1 ; 1 ; 1] ;
372               case 6
373                 init = [1 ; 1 ; 1] ;
374               case 7
375                 init = [1 ; 1 ; 1] ;
376               case 8
377                 init = [1 ; 1 ; 1] ;
378               case 9
379                 init = [1 ; 1 ; 1] ;
380               case 10
381                 init = [1 ; 1 ; 1] ;
382               case 11
383                 init = [1 ; 1 ; 1] ;
384               case 12
385                 init = [1 ; 1 ; 1] ;
386               case 13
387                 init = [1 ; 1 ; 1] ;
388             end % end switch modelId
389         end % end switch ctrlr
390       case 2 % SIMC
391         init = 0 ;
392       case 3 % Smith
393         switch ctrlr
394           case 1 % PID
395             switch cNum
396               case 1
397                 switch modelId
398                   case 1
399                     init = [1 1 1 ;
400                             1 1 1 ;
```

```
401                          1.7929 0.3563 0.2657 ] ;
402                  case 2
403                    init = [1 1 1 ;
404                            1 1 1 ;
405                            1.9816 0.3145 0.3023 ] ;
406                  case 3
407                    init = [1 1 1 ;
408                            1 1 1 ;
409                            1.72 0.39 0.23 ] ;
410                  case 4
411                    init = [1 1 1 ;
412                            1 1 1 ;
413                            4.4767 0.4695 0.1505 ] ;
414                  case 5
415                    init = [1 1 1 ;
416                            1 1 1 ;
417                            3.2484 0.3752 0.2279 ] ;
418                  case 6
419                    init = [6.3862 0.3774 0.1251 ;
420                            9.0004 0.5500 0.0867 ;
421                            4.0203 0.4907 0.1032 ] ;
422                  case 7
423                    init = [5.9076 0.3270 0.1773 ;
424                            7.4737 0.4608 0.1316 ;
425                            3.4348 0.4083 0.1532 ] ;
426                  case 8
427                    init = [4.5005 0.2719 0.2371 ;
428                            5.1083 0.3579 0.2007 ;
429                            2.5717 0.3410 0.2186 ] ;
430                  case 9
431                    init = [8.4814 0.3796 0.1219  ;
432                            13.0195 0.5817 0.0740 ;
433                            5.4169 0.5027 0.0941 ] ;
434                  case 10
435                    init = [0.8216 0.0003 0.2429 ;
436                            1.1 0.2 0.25         ;
437                            0.5 0.1 0.3 ] ;
438                  case 11
439                    init = [1.8106 0.3767 0.1508 ;
440                            2.0336 0.4223 0.1568 ;
441                            1.3500 0.4500 0.1350] ;
442                  case 12
443                    init = [7.3804 0.0842 0.2155 ;
444                            9.7846 0.2457 0.2261 ;
445                            4.6914 0.1757 0.2437 ] ;
446                  case 13
447                    init = [0.8378 0.7484 0.0044 ;
448                            0.8237 0.7521 0.0015 ;
449                            0.8250 0.7576 0 ] ;
```

```
450                 end % end switch modelId
451               case 2
452                   % Not used
453               case 3
454                 switch modelId
455                   case 1
456                     init = [1 1 1 ;
457                             1 1 1 ;
458                             0.4570 0.8034 0.6488 ] ;
459                   case 2
460                     init = [1 1 1 ;
461                             1 1 1 ;
462                             0.5134 0.7011 0.7388 ] ;
463                   case 3
464                     init = [1 1 1 ;
465                             1 1 1 ;
466                             0.4312 0.8811 0.5645 ] ;
467                   case 4
468                     init = [1 1 1 ;
469                             1 1 1 ;
470                             1.1372 0.9777 0.3882 ] ;
471                   case 5
472                     init = [1 1 1 ;
473                             1 1 1 ;
474                             0.8662 0.7902 0.5529 ] ;
475                   case 6
476                     init = [1 1 1 ;
477                             1 1 1 ;
478                             1.6325 1.1884 0.2538 ] ;
479                   case 7
480                     init = [1 1 1 ;
481                             1 1 1 ;
482                             1.5061 0.9707 0.3521 ] ;
483                   case 8
484                     init = [1 1 1 ;
485                             1 1 1 ;
486                             1.1327 0.7859 0.4992 ] ;
487                   case 9
488                     init = [1 1 1 ;
489                             1 1 1 ;
490                             1.5 0.2 0.2 ] ;
491                   case 10
492                     init = [1 1 1;
493                             1 1 1;
494                             0.3256 0.1811 0.4291] ;
495                   case 11
496                     init = [1 1 1;
497                             1 1 1;
498                             0.4503 0.9976 0.4199] ;
```

```
499                 case 12
500                   init = [1 1 1;
501                           1 1 1;
502                           2.7235 0.3029 0.4205 ] ;
503                 case 13
504                   init = [1 1 1;
505                           1 1 1;
506                           0.1176 2.7928 0.0170 ] ;
507               end
508             case 4
509               switch modelId
510                 case 1
511                   init = [1 1 1 ;
512                           1 1 1 ;
513                           0.3154 0.8509 0.7143] ;
514                 case 2
515                   init = [1 1 1 ;
516                           1 1 1 ;
517                           0.3459 1.0056 0.8345] ;
518                 case 3
519                   init = [1 1 1 ;
520                           1 1 1 ;
521                           0.3083 0.7493 0.6326] ;
522                 case 4
523                   init = [1 1 1 ;
524                           1 1 1 ;
525                           0.7966 0.7662 0.5323] ;
526                 case 5
527                   init = [1 1 1 ;
528                           1 1 1 ;
529                           0.5723 0.9376 0.7306] ;
530                 case 6
531                   init = [1 1 1 ;
532                           1 1 1 ;
533                           1.2396 0.6113 0.3534] ;
534                 case 7
535                   init = [1 1 1 ;
536                           1 1 1 ;
537                           1.0713 0.7802 0.4870] ;
538                 case 8
539                   init = [1 1 1 ;
540                           1 1 1 ;
541                           0.7553 0.9415 0.6836] ;
542                 case 9
543                   init = [1 1 1 ;
544                           1 1 1 ;
545                           1.6304 0.5997 0.3237] ;
546                 case 10
547                   init = [1 1 1;
```

```
548                             1 1 1;
549                             % 0.2971 5.0103 0.4748 ] ;
550                             % 0.4249 3.1270 0.5373 ] ;
551                             0.3258 5.8437 0.4221 ] ;
552                   case 11
553                     init = [1 1 1;
554                             1 1 1;
555                             0.4503 1.0024 0.4198 ] ;
556                   case 12
557                     init = [1 1 1;
558                             1 1 1;
559                             2.7234 3.3003 0.4207 ] ;
560                   case 13
561                     init = [1 1 1;
562                             1 1 1;
563                             0.19 0.31 0.02 ];
564                             % 2.7235 3.3011 0.4206 ] ;
565                   case 14
566                     init = [1 1 1;
567                             1 1 1;
568                             7.9709    2.0882    0.6502];
569                 end
570               end % end switch cNum
571             case 2 % PI
572               switch cNum
573                 case 1
574                   switch modelId
575                     case 1
576                       init = [0.8540 0.4185 ;
577                               0.8287 0.4377 ;
578                               0.30 0.83 ];
579                     case 2
580                       init = [0.8647 0.3781 ;
581                               0.8324 0.4001 ;
582                               0.3281 0.6769 ] ;
583                     case 3
584                       init = [0.8803 0.4439 ;
585                               0.8579 0.4622 ;
586                               0.3218 0.8720 ] ;
587                     case 4
588                       init = [1.7960 0.4054 ;
589                               1.7024 0.4602 ;
590                               0.8020 0.4633 ] ;
591                     case 5
592                       init = [1.3179 0.3681 ;
593                               1.2454 0.4088 ;
594                               0.5983 0.4145 ] ;
595                     case 6
596                       init = [2.7077 0.4360 ;
```

```
                    2.6025 0.5145 ;
                    1.2037 0.5038 ] ;
          case 7
            init = [2.1404 0.4022 ;
                    2.0171 0.4645 ;
                    0.9347 0.4627 ] ;
          case 8
            init = [1.5827 0.3608 ;
                    1.4804 0.4117 ;
                    0.6990 0.4152 ] ;
          case 9
            init = [3.2223 0.4276 ;
                    3.0833 0.5222 ;
                    1.4053 0.5020 ] ;
          case 10
            init = [1    0.1 ;
                    0.6  0.2 ;
                    0.24 0.07 ] ;
          case 11
            init = [1.04 0.47 ;
                    1.02 0.49 ;
                    0.51 0.49 ] ;
          case 12
            init = [4.5  0.11 ;
                    4.23 0.21 ;
                    2.14 0.16 ] ;
          case 13
            init = [0.19 3.14 ;
                    0.1  2    ;
                    0.44 0.74 ] ;
        end
      case 2
        % Never used
      case 3
        switch modelId
          case 1
            init = [0.8540 0.4185 ;
                    0.8287 0.4377 ;
                    0.30 0.83 ] ;
          case 2
            init = [0.8647 0.3781 ;
                    0.8324 0.4001 ;
                    0.3281 0.6769 ] ;
          case 3
            init = [0.8803 0.4439 ;
                    0.8579 0.4622 ;
                    0.3218 0.8720 ] ;
          case 4
            init = [1.7960 0.4054 ;
```

```
646                             1.7024 0.4602 ;
647                             0.55 0.85 ] ;
648                   case 5
649                     init = [1.3179 0.3681 ;
650                             1.2454 0.4088 ;
651                             0.4305 0.7243 ] ;
652                   case 6
653                     init = [2.7077 0.4360 ;
654                             2.6025 0.5145 ;
655                             1.2037 0.5038 ] ;
656                   case 7
657                     init = [2.1404 0.4022 ;
658                             2.0171 0.4645 ;
659                             0.6162 0.8535 ] ;
660                   case 8
661                     init = [1.5827 0.3608 ;
662                             1.4804 0.4117 ;
663                             0.6990 0.4152 ] ;
664                   case 9
665                     init = [3.2223 0.4276 ;
666                             3.0833 0.5222 ;
667                             0.8475 1.0090 ] ;
668                   case 10
669                     init = [1   0.1 ;
670                             0.6 0.2 ;
671                             0.3 0.14] ;
672                   case 11
673                     init = [1.04 0.47 ;
674                             1.02 0.49 ;
675                             0.51 0.49 ] ;
676                   case 12
677                     init = [4.5  0.11 ;
678                             4.23 0.21 ;
679                             2.14 0.16 ] ;
680                   case 13
681                     init = [0.19 3.14 ;
682                             0.1  2    ;
683                             0.44 0.74 ] ;
684                 end
685               case 4
686                 switch modelId
687                   case 1
688                     init = [0.8540 0.4185 ;
689                             0.8287 0.4377 ;
690                             0.30 0.83 ];
691                   case 2
692                     init = [0.8647 0.3781 ;
693                             0.8324 0.4001 ;
694                             0.3281 0.6769 ] ;
```

```
695                 case 3
696                   init = [0.8803 0.4439 ;
697                           0.8579 0.4622 ;
698                           0.3218 0.8720 ] ;
699                 case 4
700                   init = [1.7960 0.4054 ;
701                           1.7024 0.4602 ;
702                           0.55 0.85 ] ;
703                 case 5
704                   init = [1.3179 0.3681 ;
705                           1.2454 0.4088 ;
706                           0.4305 0.7243 ] ;
707                 case 6
708                   init = [2.7077 0.4360 ;
709                           2.6025 0.5145 ;
710                           1.2037 0.5038 ] ;
711                 case 7
712                   init = [2.1404 0.4022 ;
713                           2.0171 0.4645 ;
714                           0.6162 0.8535 ] ;
715                 case 8
716                   init = [1.5827 0.3608 ;
717                           1.4804 0.4117 ;
718                           0.6990 0.4152 ] ;
719                 case 9
720                   init = [3.2223 0.4276 ;
721                           3.0833 0.5222 ;
722                           0.8475 1.0090 ] ;
723                 case 10
724                   init = [1    0.1 ;
725                           0.6  0.2 ;
726                           0.30 7.34] ;
727                 case 11
728                   init = [1.04 0.47 ;
729                           1.02 0.49 ;
730                           0.38 1.00 ] ;
731                 case 12
732                   init = [4.5  0.11 ;
733                           4.23 0.21 ;
734                           0.5  0.8  ] ;
735                 case 13
736                   init = [0.19 3.14 ;
737                           0.1  2    ;
738                           0.44 0.74 ] ;
739                 case 14
740                   init = [1.04 0.47 ;
741                           1.02 0.49 ;
742                           0.38 1.00 ] ;
743               end % end switch modelId
```

```
744             end % end switch cNum
745         end % end switch ctrlr
746       case 4
747         init = 0 ;
748       case 5
749         init = 0 ;
750     end
751 end
```

M.7 Models Function: model.m

The model function accepts the modelId as argument and return a struct of the appropriate model transfer function and it's parameters.

Listing M.7 – Function generating a struct containing the model transfer function and its corresponding parameters

```
1  %% Header
2  % Purpose:  Creating the process transfer function G (struct).
3  % Author:   Axel Lodemel Holene (based on work done by Chriss
4  %           Grimholt (2011) and Martin S. Foss (2012)).
5  % E-mail:   axel.holene@gmail.com
6  % Date:     February 2013
7  % About:    G =
8  %                   k                              Process
       gain
9  %                   tau1                    Process time
       constant
10 %                   tau2                    Process time
       constant
11 %                   theta                      Process time
       delay
12 %                   tf                 Process transfer
       function
13
14 function G = model(modelId)
15     s = zpk('s') ;                  % Transfer function
           variable
16
17     G.k     = 1   ;                        % Common process
           gain
18     G.tau1  = 1   ;             % Common process time constant
           1
19
20     switch modelId
21         case 1
```

```
22              G.case = 1 ;
23              G.tau2  = 0.5*G.tau1 ;      % Process time constant
                    2
24              G.theta = 1          ;          % Process time
                    delay
25          case 2
26              G.case = 2 ;
27              G.tau2  = 0.8*G.tau1 ;      % Process time constant
                    2
28              G.theta = 1 ;                   % Process time
                    delay
29          case 3
30              G.case = 3 ;
31              G.tau2  = 0.3*G.tau1 ;      % Process time constant
                    2
32              G.theta = 1 ;                   % Process time
                    delay
33          case 4
34              G.case = 4 ;
35              G.tau2  = 0.5*G.tau1 ;      % Process time constant
                    2
36              G.theta = G.tau2/1.5 ;          % Process time
                    delay
37          case 5
38              G.case = 5 ;
39              G.tau2  = 0.8*G.tau1 ;      % Process time constant
                    2
40              G.theta = G.tau2/1.5 ;          % Process time
                    delay
41          case 6
42              G.case = 6 ;
43              G.tau2  = 0.3*G.tau1 ;      % Process time constant
                    2
44              G.theta = G.tau2/1.5 ;          % Process time
                    delay
45          case 7
46              G.case = 7 ;
47              G.tau2  = 0.5*G.tau1 ;      % Process time constant
                    2
48              G.theta = G.tau2/2.0 ;          % Process time
                    delay
49          case 8
50              G.case = 8 ;
51              G.tau2  = 0.8*G.tau1 ;      % Process time constant
                    2
52              G.theta = G.tau2/2.0 ;          % Process time
                    delay
53          case 9
54              G.case = 9 ;
```

```
55              G.tau2  = 0.3*G.tau1 ;     % Process time constant
                    2
56              G.theta = G.tau2/2.0 ;          % Process time
                    delay
57          case 10
58              G.case = 10 ;
59              % G.tau1  = 100 ;
60              G.tau2  = 0 ;
61              G.theta = 1 ;
62              % G.k = 100 ;
63              G.tfnd  = G.k/s ;
64              G.tf    = G.tfnd*exp(-G.theta*s) ;
65              return
66          case 11
67              G.case = 11 ;
68              G.tau1  = 1 ;
69              G.tau2  = 0 ;
70              G.theta = 1 ;
71          case 12
72              G.case = 12 ;
73              G.tau1  = 8 ;
74              G.tau2  = 0 ;
75              G.theta = 1 ;
76          case 13
77              G.case = 13 ;
78              G.tau1  = 0.005 ;
79              G.tau2  = 0 ;
80              G.theta = 1 ;
81          case 14
82              G.case = 14 ;
83              % G.k    = 20 ;
84              G.tau1 = 20 ;
85              G.tau2 = 0 ;
86              G.theta = 1 ;
87      end
88      %% Process transfer functions
89      G.tfnd = G.k/((G.tau1*s + 1)*(G.tau2*s + 1)) ;
90      G.tf   = G.tfnd*exp(-G.theta*s) ;
91  end
```

M.8 Maximum Sensitivity Peak Calculation: Ms.m

The function Ms accepts a vector of controller tuning and the parameter struct, and returns the maximum sensitivity peak value, M_S, for the given model and controller loop.

Listing M.8 – Function computing the maximum sensitivity peak value for a given controller and process model loop.

```
%% Header
% Purpose:  Calculating M_s-value from controller tuning
% Author:   Axel Lodemel Holene
% E-mail:   axel.holene@gmail.com
% Date:     13. February 2013
% About:

function Ms = Ms(ctrl, parm)
    G  = parm.G ;
    C  = controller(ctrl, parm) ;
    Ms = max(abs(freqresp(1/(1 + G.tf*C.tf),...
         logspace(-4,4,40000)))) ;
end
```

M.9 Maximum Complementary Sensitivity Peak Calculation: `Mt.m`

The function `Mt` accepts a vector of controller tuning and the parameter struct, and returns the maximum sensitivity peak value, M_T, for the given model and controller loop.

Listing M.9 – Function computing the maximum sensitivity peak value for a given controller and process model loop.

```
%% Header
% Purpose:  Calculating M_t-value from controller setting
% Author:   Axel Lodemel Holene
% E-mail:   axel.holene@gmail.com
% Date:     13. February 2013
% About:

function Mt = Mt(ctrl, parm)
    G = parm.G ;
    C = controller(ctrl, parm) ;
    Mt = max(abs(freqresp((G.tf*C.tf)/(1+G.tf*C.tf), ...
             logspace(-4,4,40000)))) ;
end
```

M.10 Gradient Free Optimisation Help Function

The help function for gradient free optimisation by the use of `fminsearch` is given in Listing M.10. The function accepts a controller tuning vector and the parameter struct. It determines the controller gain necessary to reach the M_S value specification by the use of `fzero`.

Listing M.10 – `fminsearch` help function for gradient free optimisation.

```
1  % Purpose:  Help function for gradient free opt. with
       fminsearch
2  % Author:   Axel Lodemel Holene
3  % E-mail:   axel.holene@gmail.com
4  % Date:     March 2013
5  % About:    ID = [I D] Vector of integral (and derivative) time
6
7  function J = noGrad(ctrl, parm)
8      Kc = ctrl(1) ;
9      ID = ctrl(2:end) ;
10     MsFun = @(Kc) parm.Ms_s - Ms([Kc ID], parm) ;
11     opt = optimset('Display','off','UseParallel','Always') ;
12     Kc = fzero(MsFun, Kc, opt) ;       % Finding controller
           gain
13     J  = costFun([Kc ID], parm) ;        % Calling cost
           function
14 end
```

M.11 Disturbance Functions

The disturbance functions are functions made when keeping controller and model structs constant was required. The functions are very similar to their ancestors; they are only modified to accept the controller and model struct as arguments instead of computing them within the function.

The disturbance functions are:

- `distCostFun`
- `distMs`
- `distMt`
- `distModel`

The three first functions should be easy to find when considering the parent functions being listed. A description of `distModel` is included for convenience.

M.11.1 Model Disturbance Implementation: `distModel.m`

The `distModel` function, given in Listing M.11, accepts a model struct and a disturbance parameter (`d`). The original model struct is duplicated into the struct returned by the function, and a new field with the new, time-delay perturbed model is saved at the `tf` attribute.

Listing M.11 – Function for changing the time-delay parameter, θ, by a factor of `d`.

```
%% Header
% Purpose:  Function which introduces modelling error to the
%           process transfer function, such that it differs
    from
%           the model used in the Smith Predictor calculations.
% Author:   Axel Lodemel Holene
% E-mail:   axel.holene@gmail.com
% Date:     13. February 2013
% About:    G - model struct returned from model.m
%           d - disturbance factor

function Gse = distModel(G, d)
    s = zpk('s') ;
    Gse         = G ;
    Gse.origMod = G ;
    Gse.theta   = (1 + d)*G.theta ;
    Gse.tf = G.tfnd*exp(-Gse.theta*s) ;
end
```

APPENDIX N

SIMULINK

The SIMULINK flow sheet used for computing the integrated absolute error values for a step load disturbance in the process input and output signal, d_i and d_o, respectively, is given in Figure N.1. S and S_d are the output ($y \rightarrow e$) and input ($u \rightarrow e$) feedback loop to the feedback error, $e = y - r$, where y and r is the output signal and reference signal for the system, respectively.

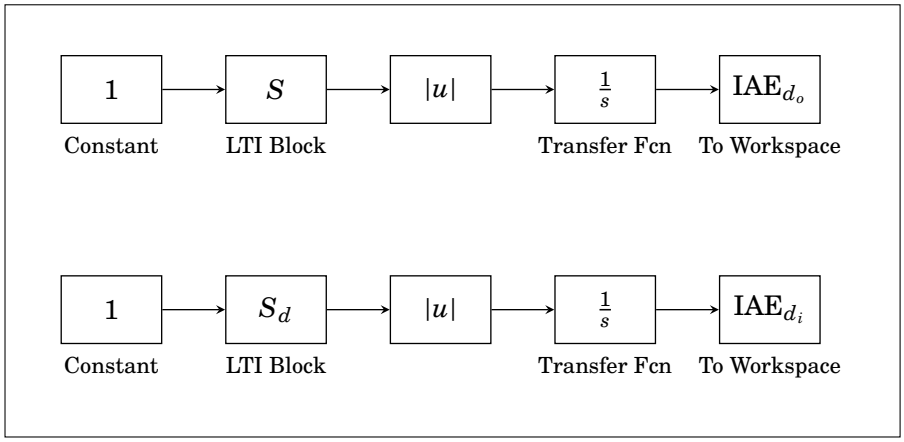


Figure N.1 – SIMULINK flow sheet for computation of the integrated absolute error from a step load disturbance in the process input and output signal.