Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

Advanced Tools for Interactive Design of Simple Controllers

Plzeň, 2023

Bc. Vilém Žán

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Vilém ŽÁN**
Osobní číslo:	**A21N0130P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Téma práce:	**Pokročilé nástroje pro interaktivní návrh jednoduchých regulátorů**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Analyzujte současný stav průmyslových regulátorů a technické prostředky jejich ladění.
2. Sestrojte identifikační modul pro získání modelu reálného systému.
3. Vyviňte nástroj hledající optimální robustní regulátor s využitím integrálních kritérií optimality.
4. Prokažte funkčnost nástroje pro systémy s neurčitostí.
5. Validujte výsledky na reálném systému.

Rozsah diplomové práce:	**40-50 stránek A4**
Rozsah grafických prací:	
Forma zpracování diplomové práce:	**tištěná**
Jazyk zpracování:	**Angličtina**

Seznam doporučené literatury:

1. Astrom, K. J., Hägglund, T., & Astrom, K. J. (2006). Advanced PID control (Vol. 461). Research Triangle Park: ISA-The Instrumentation, Systems, and Automation Society.
2. Astrom, K. J., & Murray, R. M. (2008). Feedback systems. Princeton Univer.
3. Yurkevich, V. D. (Ed.). (2011). Advances in PID control. BoD-Books on Demand.
4. Schlegel, M., & Medvecová, P. (2018). Design of PI controllers: H-inf region approach. IFAC-papersonline, 51(6), 13-17.
5. Matušů, R., Şenol, B., & Pekař, L. (2020). Robust PI control of interval plants with gain and phase margin specifications: Application to a continuous stirred tank reactor. IEEE Access, 8, 145372-145380.
6. Čech, M., & Schlegel, M. (2013, February). Generalized robust stability regions for fractional PID controllers. In 2013 IEEE International Conference on Industrial Technology (ICIT) (pp. 76-81). IEEE.
7. Schlegel, M., Mertl, J., & Čech, M. (2003). Generalized robustness regions for PID controllers. Process Control, 3, 108.
8. Astrom, K. J., & Hägglund, T. (2004). Revisiting the Ziegler-Nichols step response method for PID control. Journal of process control, 14(6), 635-650.
9. Rojas, J. D., Arrieta, O., & Vilanova, R. (2021). Industrial PID Controller Tuning. Springer International Publishing.
10. Smuts, J. F. (2011). Process Control for Practitioners: How to Tune PID Controllers and Optimize Control Loops. OptiControls.

Vedoucí diplomové práce:	**Ing. Martin Čech, Ph.D.**
	Výzkumný program 1
Datum zadání diplomové práce:	**1. října 2022**
Termín odevzdání diplomové práce:	**22. května 2023**

[illegible]

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

L.S.

[illegible]

Doc. Ing. Miloš Železný, Ph.D.
děkan

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 1. října 2022

<u>P R O H L Á Š E N Í</u>

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 18. května 2023

[illegible]
....
vlastnoruční podpis

Poděkování

Tímto bych chtěl poděkovat vedoucímu své práce panu Ing. Martinu Čechovi, Ph.D. za odborné vedení, za cenné a četné rady a konzultace a za vstřícný přístup.

Dále bych chtěl poděkovat Bc. Daliboru Májovi za poskytnutý inkubátor, na kterém mohla být tato práce validována.

Anotace

Cílem této práce je návrh nástrojů pro identifikaci modelu reálného procesu, návrh robustního regulátoru pro procesy s neurčitostí, optimalizaci chování zpětnovazební smyčky v časové oblasti.

V první části práce je vysvětlena část teorie řízení, která byla použita pro autorovo řešení. Autorův přístup je zaměřen na metodu ladění regulátorů pomocí robustních regionů stability pro PI regulátory, procesy s neurčitostí, množinový model a integrální kritéria optimality v časové oblasti. V druhé části práce je analyzovaný současný stav nástrojů pro návrh regulátorů a identifikaci procesů. Ve třetí části je popsána implementace identifikačního modulu, návrh robustního regulátoru pro procesy s neurčitostí a nástroj pro optimalizaci chování zpětnovazebné smyčky v časové oblasti pomocí integrálních kritérií. Je zde také představena metoda gradientní optimalizace. V závěrečné části této práce je ověřena funkčnost vyvinutých nástrojů na reálném systému.

Klíčová slova: teorie řízení, PID regulace, PI regulátor, regiony robustní stability, neurčitost, experimentální identifikace, integrální kritéria optimality, Matlab, GUI, optimalizace, gradientní optimalizace

Annotation

The aim of this thesis is to develop tools for process model identification, robust controller design for processes with uncertainty, and closed-loop performance optimization in the time domain.

In the first part, the control theory used in the author's approach is explained. The approach is focused on the controller tuning method using the robust stability regions for PI controllers, processes with uncertainty, the Model set approach, and the time domain integral criteria of optimality. In the second part, the state-of-the-art of the current controller design and process identification tools is analyzed. In the third part, the implementation of the identification module, robust controller design for processes with uncertainty, and a tool for closed-loop optimization in the time domain using the integral criteria is described. The gradient optimization method is shown here. In the final part, the functionality of the developed tools is validated on a real system.

Keywords: control theory, PID control, PI controller, robust stability regions, uncertainty, experimental identification, integral criteria of optimality, Matlab, GUI, optimization, gradient optimization

Contents

1 Introduction **1**

1.1 General Introduction . . . 1

2 Concept and Approach **4**

2.1 Process and Controller in the Open Loop . . . 4

2.2 Feedback Control . . . 4

2.2.1 PID Control . . . 6

2.3 Controller Tuning . . . 7

2.3.1 Design Criteria . . . 7

2.3.2 Nyquist Plot Shaping . . . 8

2.4 Robust Stability Regions for Simple Controllers . . . 10

2.4.1 PI Robust Stability Regions . . . 11

2.5 Uncertainty in Process Control . . . 12

2.5.1 Identification Part . . . 14

2.5.2 Model Set . . . 15

2.5.3 Robust Controller Design . . . 16

2.6 Integral Criteria of Optimality . . . 17

3 State of the Art of Current Controller Design and Process Identification Tools **19**

3.1 PID H_∞ Designer . . . 19

3.1.1 Conclusion . . . 23

3.2 REXYGEN Blocks . . . 24

3.2.1 Conclusion . . . 25

3.3 Matlab Identification Tools . . . 25

3.3.1 System Identification Toolbox . . . 26

4 Implementation of Advanced Tools for Simple Controller Design **29**

4.1 Previously Developed Controller Design Tool . . . 29

4.2 Identification Module . . . 31

4.2.1 Process Model Identification Using Matlab Functions . . . 31

4.2.2 FOPMDT Process Model Identification . . . 32

4.2.3 Model Uncertainty . . . 37

4.3 Robust Controller Design for Process with Uncertainty . . . 38

4.4 Closed-Loop Time Domain Performance Optimization Tool . . . 42

4.4.1 Integral Criteria Standard Optimization . . . 47

4.4.2 Integral Criteria Gradient Optimization . . . 50

4.4.3 Comparison of Standard and Gradient Optimization Methods . . . 58

5 Validation on Real Process **63**

5.1 Validation of Estimated Matlab Process Model . . . 64

5.2 Validation of Estimated FOPMDT Process Model . . . 65

6 Conclusion **68**

6.1 Future Works . . . 69

References **71**

1 Introduction

Process control is by no means a nontrivial discipline. Nowadays, the feedback loop containing a proportional-integral-derivative (PID) controller forms the basis of modern process control [6, 24, 58]. The PID controllers are used mainly due to the simplicity of the control law leading to the necessity to tune maximally three parameters. Moreover, PID controller parameters have clear physical interpretation [16, 66]. Besides the process control, the PID controllers are also used in energetic, chemical, robotics, or *e.g.*, embedded systems[1, 5, 23].

Literature states that PID controllers make up almost 90% of all controllers [8]. The majority of them (97%) do not use the derivative term, i.e. are just proportional-integral (PI) types. Considering the history of PID control, its importance in industrial processes, and the software tools available for detecting and fixing poor performance, it is surprising, that still many of these PI controllers (70%) are tuned improperly or work just with default parameters [17, 33, 58, 67].

If used properly, the feedback control can bring huge energy and material savings, product quality, and other economic benefits [7]. This idea has been the main motivation behind the controller tuning methods that have made an appearance over the years.

One of the oldest and still the most popular controller tuning method is the Ziegler-Nichols method [8, 50, 68]. Consequently, during the evolution of industrial technology, many PID controller tuning methods have emerged. Unfortunately, the vast majority of these methods, including the Ziegler-Nichols method, often lead to an unstable feedback loop, and thus these methods are not very reliable [51].

Hence it was necessary to find more exact analytical approaches [67]. General overview of PID control system analysis, analytical or numerical design methods in time and frequency domain, and technology were introduced *e.g.*, in [2, 35].

Over time, also model uncertainties were considered in the process identification, mainly to reflect the unmodelled dynamics of the real system. In the past, huge amounts of different identification methods have been developed mainly based on standard methods of linear identification [36]. However, these methods are not too suitable for automatic procedures because of the strong dependency on the model structure. Moreover, these methods provide nominal models without any estimation of model uncertainty. In process control, it is useful to have some *a priori* information about the physical nature of the real system. If the *a priori* information is combined with the knowledge of the first few process characteristic numbers obtained from the experimental identification, it is possible to compute the exact frequency domain limits on the process transfer function. These limits play a key role in the robust controller design [15, 56].

This led to the development of methods for robust control, *e.g.*, the robust stability regions method [55]. The robust controller design based on robust stability regions brings a lot of principal trade-offs which are discussed in [27]. Moreover, it has a wide range of applications, such as a battery-super-capacitor energy storage system, a continuous stirred tank reactor, or a large wind turbine with communication delays [32, 38, 64, 67].

1.1 General Introduction

This thesis is focused on the identification of processes with uncertainty, robust controller design, and closed-loop time domain performance optimization. In this thesis, we have added a module for process identification, a tool for closed-loop time domain performance optimization, and we have designed a robust controller for process with uncertainty.

The robust controller design tool was introduced in the author's previous work [66]. Hence, this method implements tools in the interactive graphical user interface (GUI) which was developed by the author in the MathWorks app building interface App Designer [61]. The GUI uses powerful Matlab commands to automatically perform all complex calculations used in the implementation, making it an easy-to-use for any user.

The developed GUI uses robust stability regions as the controller design method. Firstly, the controller design is carried out in the frequency domain. The design criteria are defined as shaping points in the complex plane for the Nyquist curve [55]. It is possible to select multiple shaping points for the Nyquist curve. A set of controllers satisfying one design requirement for one process is represented by a robust stability region. The solution is in the form of an intersection of all robust stability regions (if it exists). The intersection area contains an infinite number of controllers [50]. Here are all controllers satisfying given shaping conditions [19, 21].

In this thesis, we will measure the time domain performance of each closed loop from the intersection area. All closed loops containing all controllers from the region will be evaluated using the integral criteria of optimality, for example, Integral Error (IE), Integral of Time-weighted Absolute Error (ITAE) *etc.* [43]. Then we will search for the closed loop with a minimal value of the given criterion. We will use standard and gradient optimization methods.

If we want to control a process we should understand process dynamics and its limitations in the time and frequency domain. However, the mathematical model of the controlled process is not always known, and therefore it needs to be identified. There are several methods for experimental identification. We should also know which controllers, tuning techniques and right tools. In this thesis, we will work with Matlab [63]. Matlab is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks, which also provides applications in practice. Specifically, it can be, for example, the automotive industry. Matlab can be supplemented with special tools for automatic code generation, thanks to which the entire SW development process is significantly faster. [29, 34].

It is also worth noting that process identification for purposes of automatic tuning of industrial controllers is an important area of automatic control that attracts both researchers and control engineers. Common requirements on such identification are quickness and simplicity of the corresponding procedure [56].

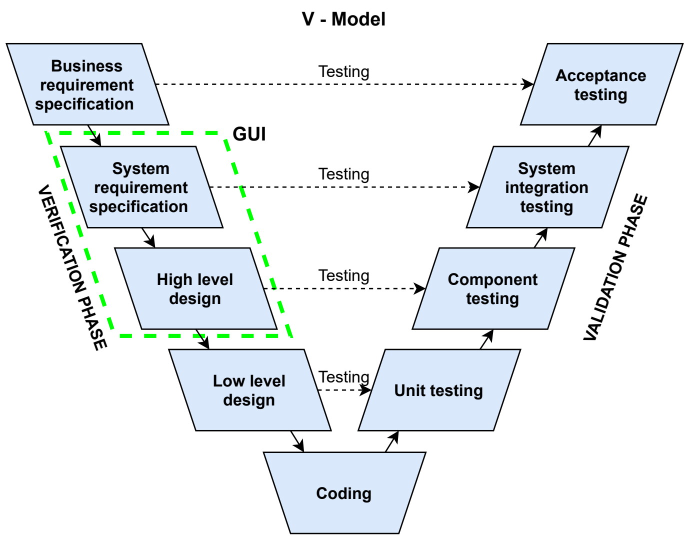


Figure 1: The role of the designed GUI in the SW development process described by the V-Model

The classic product or SW development process is shown in Figure 1 and is a well-known graph - the so-called V-diagram [29]. This graph thus links the processes of design, development and subsequent testing of the given product. As part of the design, a model is built, for example in Matlab or Simulink, where it is immediately tested. This phase is called Model in the Loop (MIL). The created and further described GUI tool, whose inclusion is shown in green in Figure 1 could also be included here. The next step is programming the given function according to the model, i.e. creating SW. It is then tested at its level, which is referred to as Software in the Loop (SIL). For example, unit tests or integration tests could be included here. At the moment when the compiled SW is uploaded to the unit, it is time to test this unit with real communications. This phase is referred to as Hardware in the Loop (HIL). Acceptance tests or customer tests come last [11, 29, 34].

In recent times, efforts to extend this process to other stages can also be encountered, where the Model Checking method is particularly worth mentioning [11]. This method deals with formal verification and verification that the created model has not deviated from predefined requirements. Using this method, test scenarios can also be generated that can be used across all phases of testing. This idea has already proven its value when the safety systems of a nuclear power plant were tested with its help [9, 10, 12].

This entire process is of course supplemented by quality control and at the same time several standards are observed, based on the industry in which we operate. It is also worth mentioning that different standards are used in the automotive industry than in the development of SW for a nuclear power plant. The goal of all these actions is to create the highest quality and safest product possible. The amount of testing is of primary importance in that the earlier an error is discovered in the process, the easier and cheaper the overall fix [11, 34].

This thesis is organized as follows: The concept and approach used in this thesis are formulated in Section 2. This chapter describes feedback control, frequency domain design requirements, the robust stability region design method, uncertainty modeling, Model set approach, process identification, and the time domain integral criteria of optimality. State of the art of present-day controller design and process identification software tools is the subject of Section 3. Three tools analysed in this thesis are: PID H_∞ Designer [45], REXYGEN [46], and MathWorks' System identification toolbox [62]. Section 4 is focused on the implementation of developed tools in this thesis. In this Section, we will show the previously developed robust controller design tool [66], the module for process identification, then we will show the robust controller design for processes with uncertainty, and last, we will introduce the tool for optimal robust controller search where we will compare standard and gradient optimization method. Section 5 provides the validation of developed tools on a real process. Section 6 contains concluding remarks as well as possible topics for future works.

2 Concept and Approach

Our approach focuses on robust feedback (or closed-loop) control [5]. The concept of the robust controller design is to find all controllers satisfying all design requirements for all processes. First, the design requirements are defined in the frequency domain as shaping points for the Nyquist curve. The solution comes in the form of the intersection of the robust stability regions, one for each design requirement for each process [15, 54].

In this thesis, we will also focus on processes with non-structural uncertainty. Hence no mathematical model is exactly accurate, the uncertainty gives us information about the relative accuracy of the process model [50]. The uncertainty will be implemented using the Model set approach [53, 57]. To obtain the Model set, we will use the identification method estimating the process characteristic numbers which parameterize the Model set. The provided result of the identification method will be a set of processes that will later be used for the robust controller design [20].

The next task is to find the optimal controller from the intersection region. Optimal closed-loop performance leads to the minimal value of the given time domain integral criterion of optimality [7, 8, 65]. In this thesis, we will introduce a gradient method searching for the optimal controller, and compare it to the standard optimization method.

The following chapters bring just a brief introduction to the control theory used in this thesis.

2.1 Process and Controller in the Open Loop

In the field of process control, we are dealing with dynamic processes (or systems) which we want to control. A process can have various forms, such as mechanical process, thermodynamic process, chemical process, et cetera. In general, a dynamic process could be defined as a process in which behavior changes in time, or as a process with a memory [41].

In order to operate with the given process we need to describe it with an appropriate model. A model is a simplified representation of reality. The control theory uses a mathematical model for describing system behavior which can be based on a differential equation. Another form of the process model, which will be shown in this thesis, is a transfer function that can be obtained as a Laplace transformation of the differential equation of the linear system. To perform analysis and various simulations, we need a linear model of the observed system. Hence the vast majority of real systems are non-linear, and there are no general controller tuning methods for the non-linear processes, the non-linear system has to be linearized.

Once we have a model of the system, we can design a controller. The controller is designed according to specific design criteria. The properly tuned robust controller should ensure the stability and sufficient quality of the closed-loop system. When the controller is connected directly to the process, the open loop is obtained. A simple scheme of process and controller connected in an open loop can be seen in Figure 2 [8].

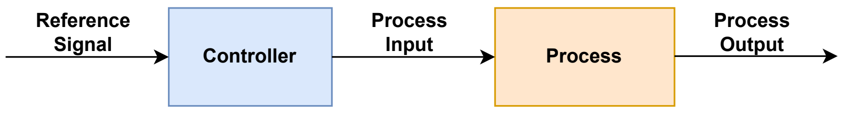


Figure 2: Open-loop schema consisting of process and controller models

However, in practice, the open-loop control is very difficult to execute. First, it requires a completely accurate model for the control system to achieve the exact desired response. Second, it can be easily influenced by disturbances that will dramatically alter the response of the system. Both problems can be fixed by feedback control [25].

2.2 Feedback Control

The feedback control is based on correcting actions on the difference between the desired and actual performance. The use of feedback has often resulted in vast improvements in system capability [3]. The fundamental requirement for the feedback loop is the ability to follow reference signal, and to compensate the effects of load disturbance [4].

In Figure 3, we can see the classical structure of a feedback control loop [8]. The process model is considered linear, and it is described by a transfer function $P(s)$. The $C(s)$ block represents the linear model of the controller [67].

Table 1 shows the notation of signals which appear in the closed-loop system.

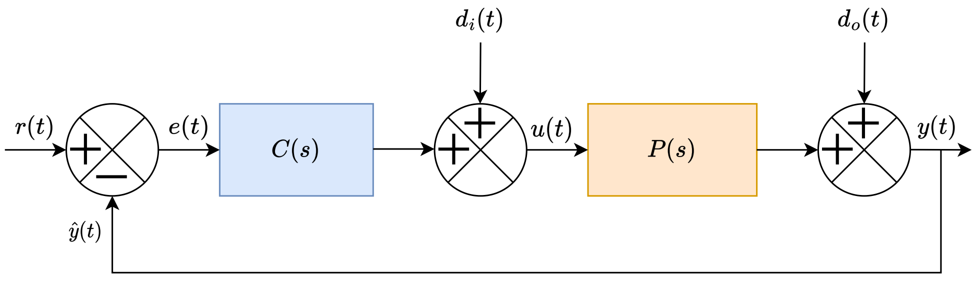


Figure 3: Feedback control loop schema

$P(s)$	Process
$C(s)$	Controller
$r(t)$	Reference signal
$d_i(t)$	Input disturbance
$d_o(t)$	Output disturbance
$u(t)$	Controller output
$y(t)$	Process output
$\hat{y}(t)$	Measured process output
$e(t)$	Control error

Table 1: Closed-loop signals notation

The performance and robustness of the closed loop can be well analyzed from the behavior of the sensitivity functions. These functions have to be stable to achieve closed-loop stability. When designing a robust controller, the following sensitivity functions are often shaped and constrained to reach required process performance. Sensitivity functions are mentioned in Table 2. Since there are four sensitivity functions, they are called the Gang of Four [3].

Sensitivity function	$S(s)$
Complementary sensitivity function	$T(s)$
Control sensitivity function	$CS(s)$
Input sensitivity function	$PS(s)$

Table 2: Sensitivity functions

The control sensitivity function can be also called the noise sensitivity function. Similarly, the input sensitivity function is also called the load disturbance function [4]. Most importantly, the sensitivity function $S(s)$ represents the effects of output disturbance $d_o(t)$ on the process output $y(t)$, and the complementary sensitivity function $T(s)$ shows how is the reference signal $r(t)$ transferred to the process output $y(t)$ [8].

The sensitivity functions are given by the following transfer functions

$$S(s) = \frac{1}{1+C(s)P(s)},\ T(s) = \frac{C(s)P(s)}{1+C(s)P(s)},\ CS(s) = \frac{C(s)}{1+C(s)P(s)},\ PS(s) = \frac{P(s)}{1+C(s)P(s)}. \quad (1)$$

2.2.1 PID Control

It has been found empirically that a useful controller structure is represented by Proportional Integral Derivative (PID) controller. PID controllers are found in large numbers in all industries. The controllers come in many different forms. The PID controller has several important functions: it provides feedback, it has the ability to eliminate steady state offsets through integral action, it can anticipate the future through derivative action [5].

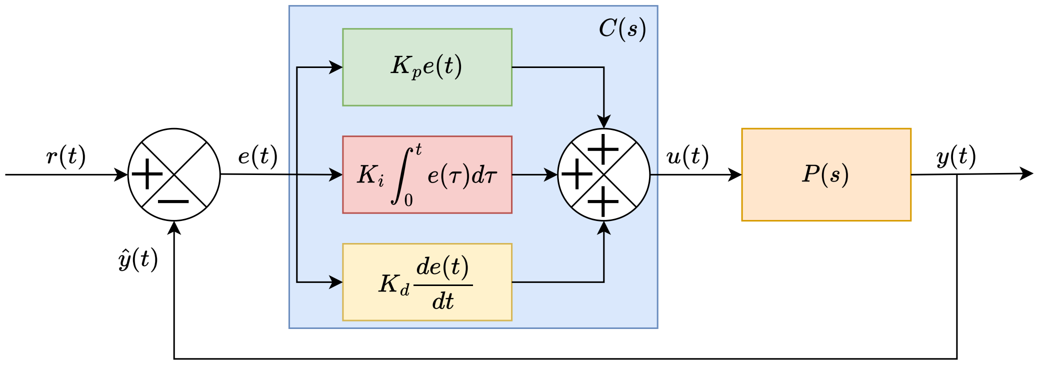


Figure 4: Non-interactive form of PID controller

The control law for a non-interactive form of PID controller is given by the formula

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right), \tag{2}$$

where K is gain, T_i is integral time constant, T_d is derivative time constant. This is an ideal version of the PID controller, hence it has a non-filtered derivative term. However, an ideal derivative is not causal, so implementations of PID controllers include additional low-pass filtering for the derivative term to limit the high-frequency gain and noise. PID controller with the filtered derivative term can be expressed as

$$C_{PID_f}(s) = K\left(1 + \frac{1}{T_i s} + \frac{T_d s}{\frac{T_d}{N}s + 1}\right). \tag{3}$$

Very often, we have to operate with the closed-loop system in the frequency domain. It is better to express the PID controller with a non-filtered derivative as a transfer function given by the formula

$$C_{PID}(s) = K_p + \frac{K_i}{s} + K_d s, \tag{4}$$

where K_p is proportional gain, K_i is integral gain, K_d is derivative gain. This form is obtained after the Laplace transformation of (2).

Most PID controllers do not use derivative action, so they should strictly speaking be called PI controllers. However, the term PID is used as a generic term for this class of controllers [8]. The transfer functions for P, PI and PD controllers are given as

$$C_P(s) = K_p, \quad C_{PI}(s) = K_p + \frac{K_i}{s}, \quad C_{PD}(s) = K_p + K_d s. \tag{5}$$

The proportional term serves as a static controller, the derivative term helps speed up the system response, and the integral term helps reduce the steady-state error.

2.3 Controller Tuning

Although the PID controller is unquestionably the most commonly used control algorithm, it seems surprising that there exists no generally accepted design method for this controller [6]. Moreover, vast majority of generally used design methods, such as Ziegler-Nichols [68], are outdated and can lead to unstable feedback loop. These methods often work with nominal model of the process which does not include uncertainty. Thus, they do not provide robust controller.

However, in the field of process control there is often a need to design a robust controller. One of the main reasons is the necessity to compensate the load disturbances affecting the system behaviour. Other practical reasons are that the controlled system components are wearing of over the time, or when another system or controller is being added to the currently controlled system causing changes in the system behaviour. These changes affect the system dynamics, because the technical parameters of the system (*e.g.* stiffness, damping, or torque coefficients) obtained from the system identification change over the time. This leads to a stage where previously designed controller may not function correctly [66].

First, in order to tune any controller, it is necessary to specify a design criteria for the feedback loop. Our approach focuses on criteria in the frequency domain.

2.3.1 Design Criteria

Design criteria represent requirements for the closed-loop system. It is an expert task to decide what type of criterion we want to choose and how to set its value to achieve required behavior of the closed loop. However, in the vast majority of cases, the most important is the closed-loop stability of a given system [13]. Furthermore, very often, it is required that the closed loop satisfies multiple design criteria at once.

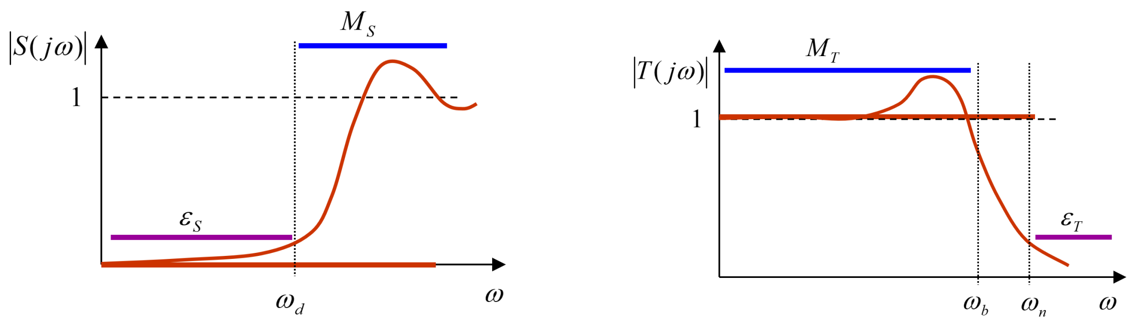
In this thesis, we will firstly define design criteria in the frequency domain. In the Table 3, there can be seen six frequency domain design criteria.

Gain margin	GM
Phase margin	PM
Sensitivity function $S(j\omega)$ maximal value	M_S
Complementary sensitivity function $T(j\omega)$ maximal value	M_T
Low-frequency disturbance rejection	ε_S
Bandwidth of the control loop	ε_T

Table 3: Design criteria

First two design criteria, GM and PM, express how far is the Nyquist curve from the critical point $[-1, j0]$. The remaining criteria, M_S, M_T, ε_S and ε_T, create limits for the magnitude of sensitivity functions $S(j\omega)$ and $T(j\omega)$, and for the behaviour of Nyquist curve in form of the M-circles and ε-circles.

The disturbance rejection parameter ε_S could be expressed as the magnitude limit of the sensitivity function $S(j\omega)$ on the low frequencies $\omega \in [0, \omega_d]$ (Subfigure 5a). The bandwidth parameter ε_T represents magnitude limit of the complementary sensitivity function $T(j\omega)$ in the frequency interval $\omega \in [\omega_n, \infty]$ (Subfigure 5b) [66].



(a) Magnitude of $S(j\omega)$ (b) Magnitude of $T(j\omega)$

Figure 5: Design requirements expressed as magnitude constraints for Sensitivity function $S(j\omega)$ and Complementary sensitivity function $T(j\omega)$ [15]

Ideal sensitivity functions $S(j\omega)$, $T(j\omega)$ requirements are

$$|S(j\omega)| = 0,\ \forall\omega, \quad |T(j\omega)| = 1,\ \forall\omega. \tag{6}$$

However, due to the physical limitations, these requirements can not be satisfied. This problem is covered by Bode integral formula [49] which can be expressed as

$$\int_0^\infty \ln|S(j\omega)|d\omega = \int_0^\infty \ln\left|\frac{1}{1+L(j\omega)}\right| d\omega = \pi\sum Re(p_k) - \frac{\pi}{2}\lim_{s\to\infty} sL(s), \tag{7}$$

where p_k are the unstable poles of $L(s)$. If $L(s)$ has at least two more poles than zeros, and has no unstable poles (is stable), the equation simplifies to

$$\int_0^\infty \ln|S(j\omega)|d\omega = 0. \tag{8}$$

This equality shows that if sensitivity to disturbance is suppressed at some frequency range, it is necessarily increased at some other range. This has been called the "waterbed effect." [40] In practice, we are operating on low frequencies where $|S(j\omega)|$ and $|T(j\omega)|$ are close to desired values 0 and 1.

Since there are physical limitations for $S(j\omega)$ and $T(j\omega)$, we are only able to satisfy "real" requirements for sensitivity functions constraints (Figure 5) according to the equations

$$|S(j\omega)| < \varepsilon_S,\ \forall\omega \in [0, \omega_d], \quad |S(j\omega)| < M_S,\ \forall\omega, \tag{9}$$

$$|T(j\omega)| < M_T,\ \forall\omega, \quad |T(j\omega)| < \varepsilon_T,\ \forall\omega \in [\omega_n, \infty]. \tag{10}$$

2.3.2 Nyquist Plot Shaping

Sensitivity functions constraints mentioned in equations (9) and (10) can be displayed in the complex plane in the form of M-circles and ε-circles (Figure 6). The center and radius of the M_S-circle is given as

$$c_S = [-1, j0], \quad r_S = \frac{1}{M_S}. \tag{11}$$

The center and radius of the ε_S-circle is

$$c_{\varepsilon_S} = [-1, j0], \quad r_S = \frac{1}{\varepsilon_S}. \tag{12}$$

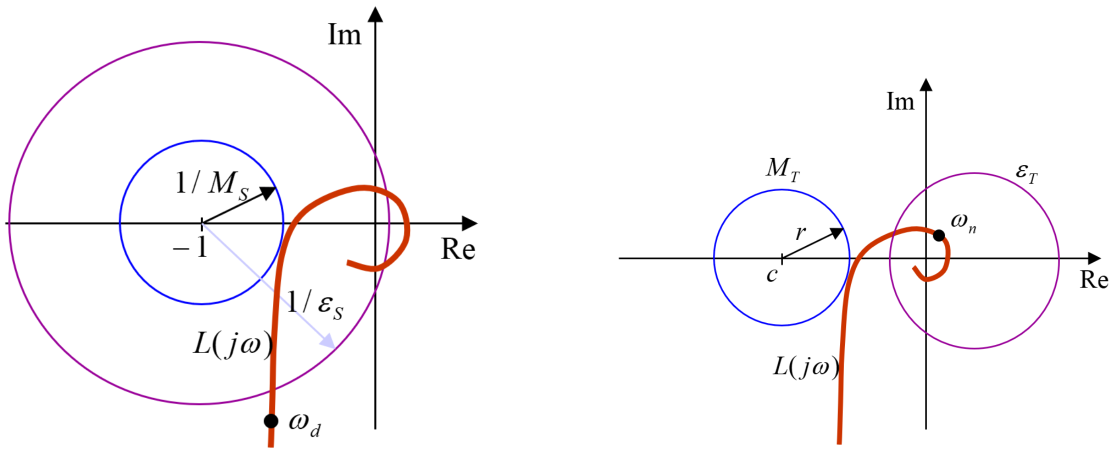
The center and radius of the M_T-circle can be expressed as

$$c_T = \left[-\frac{M_T^2}{M_T^2 - 1}, j0\right], \quad r_T = \frac{M_T^2}{|M_T^2 - 1|}. \tag{13}$$

The center and radius of the ε_T-circle can be obtained as

$$c_{\varepsilon_T} = \left[-\frac{\varepsilon_T^2}{\varepsilon_T^2 - 1}, j0\right], \quad r_{\varepsilon_T} = \frac{\varepsilon_T^2}{|\varepsilon_T^2 - 1|}. \tag{14}$$

In Figure 6 we can see that M-circles and ε-circles create boundaries for the Nyquist curve of the open loop system. This way we can perform Nyquist plot shaping.



(a) M_S and ε_S circles (b) M_T and ε_T circles

Figure 6: Nyquist plot shaping according to the M-circles and ε-circles [15]

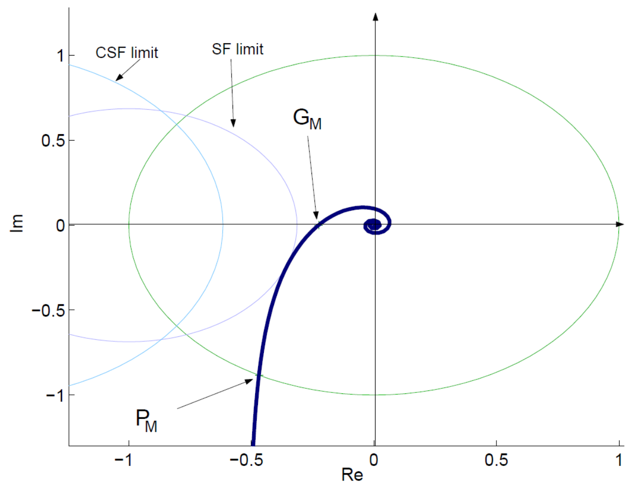


Figure 7: Nyquist plot shaping [15]

According to the Nyquist stability criterion, the number of counterclockwise encirclements about the critical point $[-1, j0]$ must be equal to the number of open-loop unstable poles [42]. Hence, the Nyquist curve passes on the right side of the $[-1, j0]$ for the stable open loop. If the open loop is stable, then the frequency domain design criteria GM, PM, M_S, M_T, ε_S, ε_T mentioned in Table 3 can be represented as shaping points in the complex plane.

Shaping points representing this criteria are special case of the general shaping points in the complex plane. General shaping point can be expressed as $X = u + jv$. If X appears on the real axis ($v = 0$), then $1/u$ is equal to the GM. If X lies on the unit circle (or when $u = v$), then the $\arctan(v/u)$ is equal to the PM. M-circles and ε-circles can be represented as a set of shaping points. However we can select any general shaping point as a design requirement.

These shaping points then represent limitations for the Nyquist curve $L(j\omega) = C(j\omega)P(j\omega)$ (Figure 7). In order to satisfy GM and PM, the Nyquist curve has to be placed on the right side of the GM and PM shaping points. The requirement on the sensitivity functions peaks M_S and M_T is satisfied when the Nyquist curve does not enter the M-circles (Figure 6). To satisfy low-frequency disturbance rejection ε_S, the Nyquist curve has to be outside of the ε_S-circle until its frequency reaches the ω_d frequency (Figure 6a). Requirement for the bandwidth of the control loop ε_T is met when the Nyquist curve is inside the ε_T-circle from the frequency ω_n (Figure 6b) [66].

2.4 Robust Stability Regions for Simple Controllers

The main reason why we select the shaping point X is that from its real and imaginary coordinates together with the real and imaginary parts of the open loop L, we can compute all possible PID controller parameters, ensuring that the shaping point X lies on the right of the passing Nyquist curve, and thus, the design requirement represented by X is satisfied (for stable processes). This set of all admissible PID parameters is called the robust stability region, and it is the center of our robust controller design method. An illustrative example can be seen in Figure 8.

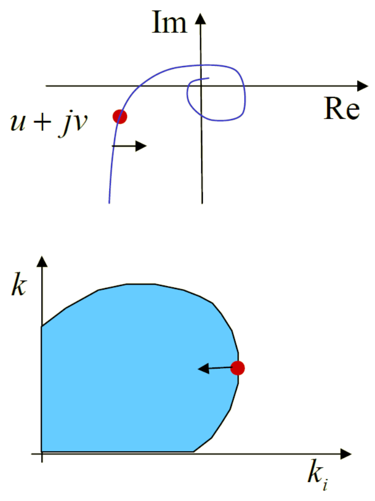


Figure 8: Nyquist plot shaping [52]

If we put

$$L(j\omega) \overset{!}{=} X, \tag{15}$$

$$C(j\omega)P(j\omega) \overset{!}{=} u + jv, \tag{16}$$

we can derive the analytical expression for the boundary of the robust stability region in the $[K_p, K_i]$ plane. In order to do so, we need the frequency response of the controller $C(j\omega)$ and process $P(j\omega)$. Frequency response of the controller depends on the type of controller. In this thesis, the PI controller will be used to compute robust stability region. Frequency response of a process $P(s)$ can be obtained by putting $s \overset{!}{=} j\omega$, and thus $P(s) \overset{!}{=} P(j\omega)$. Frequency response can be expressed as

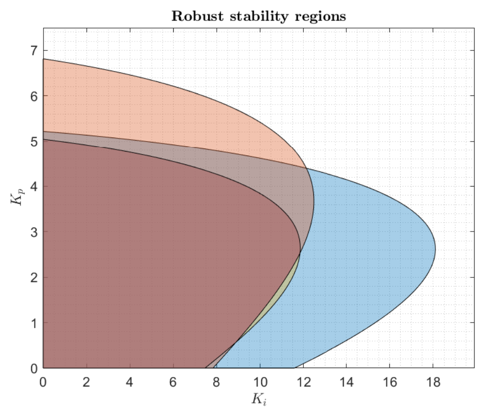
$$P(j\omega) = a(\omega) + jb(\omega), \tag{17}$$

where $a(\omega)$ is real and $b(\omega)$ is imaginary part of the frequency response, j is the imaginary unit.

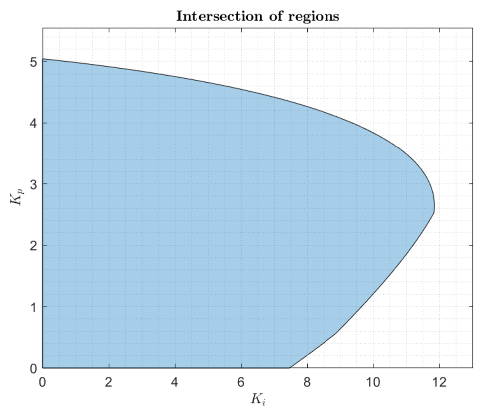
After the design criteria are selected, our design method obtains set of all possible controllers satisfying this requirement in the form of robust stability region. In the practice, it is often required to select more design criteria at one time. Our method provides this possibility. For each design requirement there is one region. If we want to find all controllers satisfying multiple design requirements, we need to find the intersection of regions. However, the intersection does not have to exist. An example of three robust stability regions each associated to one design requirement can be seen in Figure 9a. In Figure 9b, we can see the intersection of these regions.

Selecting one or more design requirements for more processes is also possible. Our tool can compute each robust stability region for each of M processes for each of N design requirements and then find its intersection - if it exists. One of the most significant advantages of this method is that it finds all solutions for the specified problem in the form of the intersection of regions. Otherwise, it easily shows that there is no solution because there is no intersection. Another significant advantage is that the boundary of the region can be analytically computed, and thus its calculation can be algorithmized.

The intersection region consists of an infinite number of controllers [15, 65]. However, only one controller can be used for our implementation. The subsequent task is to find the optimal controller which minimizes the time domain integral criteria of optimality. In Section 2.6, the time domain integral criteria of optimality will be introduced.



(a) Robust stability regions



(b) Intersection of regions

Figure 9: Illustration of robust stability regions and its intersection

2.4.1 PI Robust Stability Regions

Frequency response of the PI controller is given by formula

$$C(j\omega) = K_p(\omega) + \frac{K_i(\omega)}{j\omega}, \tag{18}$$

where $K_p(\omega)$ and $K_i(\omega)$ are PI controller parameters parametrized by frequency ω, $\omega \in (0, \infty)$. After substituting into (15), we get

$$\left(K_p(\omega) + \frac{K_i(\omega)}{j\omega}\right) \cdot (a(\omega) + jb(\omega)) \overset{!}{=} u + jv. \tag{19}$$

From this equation we can get the analytic expression of the $K_p(\omega)$ and $K_i(\omega)$ parameters of the PI controllers boundary [15]. Proof has been made in authors previous work [66]. $K_p(\omega)$ and $K_i(\omega)$ are given by formulas

$$K_p(\omega) = \frac{ua(\omega) + vb(\omega)}{a^2(\omega) + b^2(\omega)}, \quad K_p > 0, \tag{20}$$

$$K_i(\omega) = \frac{\omega\left(ub(\omega) - va(\omega)\right)}{a^2(\omega) + b^2(\omega)}, \quad K_i > 0. \tag{21}$$

Similarly, the analytic expressions are defined for the PID controller parameters $K_p(\omega)$, $K_i(\omega)$, and $K_d(\omega)$. However, for the PID controller, the region boundary computation becomes non-trivial. Hence, the PID robust stability region is a three-dimensional object. If we want to compute its border, numerical problems occur. The border of the region comes in the form of the solution of higher-order polynomial roots which are numerically computed. Moreover, it is difficult to visualize the 3D object and to operate with it in a user-friendly manner.

Another possibility is to define the ratio $f = T_i/T_d$ between the integral and derivative time constants, and the filter in the derivative term N. The ratio f is usually near 0.25 and the filter in derivative term is often chosen in the interval N $\langle 2, 10 \rangle$ according to the signal/noise ratio in the individual control application [15, 68]. This approach results in the region appearing in the $[K_i, K_p]$ plane in two dimensions. This approach leads to a more user-friendly solution, hence, operating with the 2D plot is much less difficult. However, two more design parameters need to be set in the form of f and N.

2.5 Uncertainty in Process Control

In this thesis, we will also focus on process uncertainty. There are various ways of describing uncertainty. It can be divided into two main branches, structural and non-structural uncertainty.

As for the structural uncertainty, the system model is considered accurate except for the values of its parameters. Uncertainty in the parameters can be caused by external conditions or imprecise measurement. A significant part of the study of the process with structural uncertainty is the analysis of its stability. This study is covered in Kharitonov's theorem (Kharitonov 1978). The stability of the dynamical process corresponds with the position of the characteristic polynomial roots (or poles) in the complex plane. Roots of the characteristic polynomial can be obtained from the equation

$$s^n + a_{n-1}s^{n-1} + a_{n-2}s^{n-2} + \cdots + a_1 s + a_0 = 0, \tag{22}$$

where n is the process order, and $a_i \in \left\langle a_i^{\min}, a_i^{\max} \right\rangle$, $\forall i = 0, 1, \ldots, n-1$ express the structural uncertainty. For the dynamical process to be stable, its poles must be located in the left half plane of the complex plane $\forall a_i$. It has been proven that it is sufficient to test only four so-called Kharitonov polynomials [30], which is by no means a great result of stability analysis.

However, the structural uncertainty can be impractical. Hence, it does not operate in the process's frequency domain and does not consider the real physical constraints of the process model. It supposes that the structural uncertainty is the same on all frequencies.

In this thesis, we will deal with non-structural uncertainty. We will switch to the frequency domain. Hence, it is usually assumed that the frequency response of the "true" system lies for each frequency on regions of the complex plane [20, 48]. Non-structural uncertainty includes the process dynamics and its physical limitations in the form of frequency-dependent intervals containing a frequency response of the set of process models. This set is called a Model set, and it can be described as

$$P(s) = P_0(s)\left(1 + W_m(s)\Delta\right), \quad ||\Delta||_\infty \leq 1 \tag{23}$$

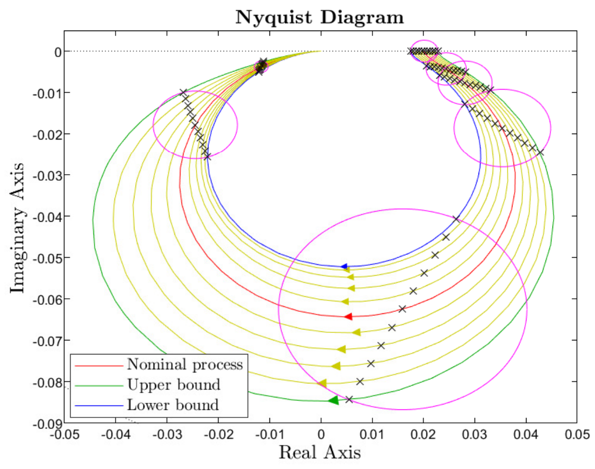


Figure 10: Non-structural uncertainty in the frequency domain

where $P_0(s)$ is the nominal model of the process, which is obtained by a simple identification algorithm (*e.g.*, minimum mean square error method), $W_m(s)$ is the weighting function which defines the process uncertainty for all frequencies ω, Δ determines the area of the unit circle. This form of non-structural uncertainty is called multiplicative. Another form of non-structural uncertainty is the additive form, which can be expressed as

$$P(s) = P_0(s) + W_a(s)\Delta, \tag{24}$$

where $W_a(s)$ is the weighting function representing the additive uncertainty. The multiplicative model can be easily recomputed to additive form and vice versa.

While working with the process with non-structural uncertainty, we want to achieve robust stability and sufficient control quality. These requirements can be expressed as weighting functions $W_S(s)$ and $W_T(s)$ for the sensitivity functions $S(s)$ and $T(s)$. The weighting functions $W_S(s)$ and $W_T(s)$ are assumed to be stable rational functions with no poles on the imaginary axis. The main function of the weighting functions is to suppress sensitivity functions on certain frequency intervals. That leads to improved reduction of load disturbances and set point tracking [55].

This requirement alongside other mentioned closed-loop requirements can often be expressed by the general condition

$$\|H(s)\|_\infty < \gamma, \tag{25}$$

where $H(s)$ denotes a stable closed-loop-related transfer function, $\|H(s)\|_\infty \triangleq \sup_{\forall\omega} |H(j\omega)|$ and γ is the design parameter [39]. The H_∞ norm represents the maximum singular value of the function. The following performance specifications are considered:

$$\|W_S(s)S(s)\|_\infty < \gamma_S, \tag{26}$$

$$\|W_T(s)T(s)\|_\infty < \gamma_T, \tag{27}$$

where γ_S, and γ_T represent design parameters for sensitivity functions $S(s)$, and $T(s)$ [55]. With this definition of sensitivity function requirements, we can specify the conditions for robust stability and robust control quality.

Robust stability can be expressed as

$$C(s),\ P_0(s) \text{ are stable } \wedge\ ||W_T(s)T_0(s)||_\infty < 1, \tag{28}$$

where $C(s)$ is controller, $P_0(s)$ is the nominal process model, $W_T(s)$ is weighting function, and $T_0(s)$ is the nominal complementary sensitivity function.

Control quality can be expressed as

$$||W_S(s)S_0(s)||_\infty < 1, \tag{29}$$

where $W_S(s)$ is the weighting function for the nominal sensitivity function $S_0(s)$.

Often, these two requirements merge into one, the robust control quality, which can be defined as

$$\Big|\Big|\ |W_T(s)T_0(s)| + |W_S(s)S_0(s)|\ \Big|\Big|_\infty < 1. \tag{30}$$

After sufficient robust control quality is achieved, it is still wise to improve the robustness of the designed controller. We can specify certain sensitivity function criteria which would allow us to do so.

The previously mentioned assumptions on the weighting functions are not always necessary. For the special case $W_S(s) = W_T(s) = 1$ and $\gamma_S = M_S$, $\gamma_T = M_T$, (26) and (27) are converted to well-known conditions [6, 55]

$$\|S(s)\|_\infty < M_S, \tag{31}$$

$$\|T(s)\|_\infty < M_T. \tag{32}$$

If we choose

$$W_S(\omega) = \begin{cases} 1 & \text{for } \omega \in [0, \omega_S] \\ 0 & \text{otherwise} \end{cases}, \tag{33}$$

$$W_T(\omega) = \begin{cases} 1 & \text{for } \omega \in [\omega_T, \infty) \\ 0 & \text{otherwise} \end{cases}, \tag{34}$$

we obtain other very useful, but not so often used criteria

$$|S(j\omega)| \le \epsilon_S,\ \omega \in [0, \omega_S], \tag{35}$$

$$|T(j\omega)| \le \epsilon_T,\ \omega \in [\omega_T, \infty). \tag{36}$$

This approach is implemented in the controller design tool PID H$_\infty$ Designer which will be mentioned further in Section 3.1 [55].

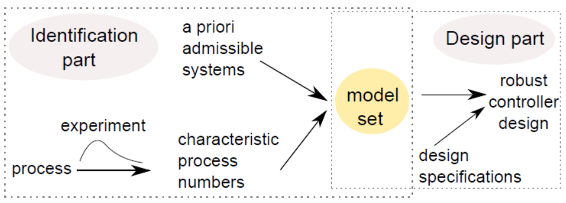
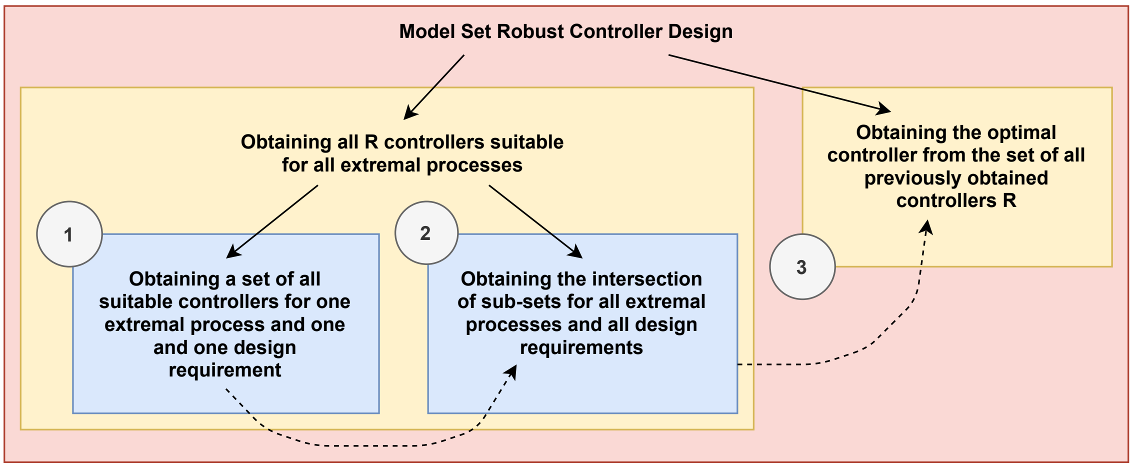


Figure 11: Model set approach diagram [20]

Besides, this thesis will focus on the Model set of either fractional order (FO) or integer order (IO) processes. FO processes will be mentioned; hence, in accordance with the majority of works in the process control field, it is assumed that the real process can be described by a multiple fractional order pole model [20]. The fractional order model contains fractional derivatives. They can be described by transfer functions $F(s)$, in which there are non-integer powers of s, i.e. they are irrational. An example of the FO model is the heat dissipation model, which can be expressed as $F(s) = K/\sqrt{sT}$ [15, 44]. Another example of the FO model can be the relationship between force and deformation for the visco-elastic materials having combined properties of rigid and flexible elements [28].

To construct the Model set, we will use the identification method combining the knowledge of *a priori* admissible systems and three characteristic process numbers measured from the input/output data.

Then we will find the extremal processes of the Model set, which define the boundaries of the process uncertainty. Then, the main aim is to design one robust controller $C(s)$ capable of satisfying all design requirements for all extremal processes. Figure 11 shows the diagram describing the Model set identification and robust controller design. Figure 12 shows the task hierarchy of the robust controller design.



Model Set Robust Controller Design
Obtaining all R controllers suitable for all extremal processes
Obtaining the optimal controller from the set of all previously obtained controllers R
1
Obtaining a set of all suitable controllers for one extremal process and one and one design requirement
2
Obtaining the intersection of sub-sets for all extremal processes and all design requirements
3

Figure 12: Robust controller design task hierarchy with the Model set approach and the solution procedure

2.5.1 Identification Part

The identification part in the Model set approach combines the *a priori* information about the admissible processes and experimentaly measured characteristic process numbers.

Huge number of admissible processes can be described by transfer function shown in [22]. This transfer function covers the majority of essentially monotone processes [7], it can be expressed as

$$P(s) = \frac{K}{\prod_{i=1}^{p}(\tau_i s + 1)^{n_i}}, \tag{37}$$

where p is arbitrary integer number and K, τ_i, n_i, $i = 1, 2, \dots p$ are positive real numbers. The equation (37) also contains the systems with dead-time [7, 18, 22, 31].

This theoretical information is then combined with experimental information. During the experiment, the process is excited by the rectangle pulse and its response is measured. The experimental information consists of three characteristic numbers $\{\kappa, \mu, \sigma^2\}$ obtained from the first three impulse response moments. These impulse response moments are generally defined as

$$m_i = \int_0^\infty t^i h(t) dt, \; i = 0, 1, 2, \tag{38}$$

where $h(t)$ is the impulse response. In practice, moments m_i are often computed from the response on a rectangle pulse on the input. Output is then recomputed as if the experiment was carried out with impulse response [15]. The characteristic numbers $\{\kappa, \mu, \sigma^2\}$ are given as

$$\kappa = \int_0^\infty h(t) dt = m_0, \tag{39}$$

$$\mu = \frac{\int_0^\infty t h(t) dt}{\int_0^\infty h(t) dt} = \frac{m_1}{m_0}, \tag{40}$$

$$\sigma^2 = \frac{\int_0^\infty (t-\mu)^2 h(t) dt}{\int_0^\infty h(t) dt} = \frac{m_2}{m_0} - \frac{m_1^2}{m_0^2}. \tag{41}$$

The computation of characteristic numbers is reasonable for monotone processes. For oscillatory processes, it would make more sense to compute natural frequency [15].

It has been proved in Čech (2008) that for the process (37) these moments result in the following relations

$$\kappa = K, \; \mu = \sum_{i=1}^{p} \tau_i n_i, \; \sigma^2 = \sum_{i=1}^{p} \tau_i^2 n_i. \tag{42}$$

Meaning from a control point of view, κ is equal to process static gain, and μ represents the residual time constant. Without loss of generality, the process can be normalized in gain and time, thus $\bar{\kappa} = 1$, $\bar{\mu} = 1$ and $\bar{\sigma}^2 = \sigma^2/\mu^2$. The remaining parameter $\bar{\sigma}^2$ then has a meaning similar to normalized dead time [20].

2.5.2 Model Set

From the characteristic numbers, we can obtain a Model set (or Moment-model set). Model set can be expressed as $\mathcal{S}^{n,m}(\kappa, \mu, \sigma^2)$, where $n \in \mathbb{R}^+$ is the total order of the process, and $m \in \mathbb{R}^+$ is the minimum allowed order of each fractional pole. Moment-model set contains admissible transfer functions. Admissible transfer function $P(s)$ has to be in the form (37) and $n_i \geq m, \forall i, \sum_{i=1}^{p} n_i \leq n$, and it has to be consistent with experimental data and thus, to fulfill (42). For $n \geq 2m$, the Model set $\mathcal{S}^{n,m}(\kappa, \mu, \sigma^2)$ is not empty if and only if

$$\frac{1}{n} \leq \frac{\sigma^2}{\mu^2} \leq \frac{1}{m}. \tag{43}$$

If this inequality is satisfied, then the Model set contains an infinite number of processes for given characteristic numbers $\{\kappa, \mu, \sigma^2\}$. After mapping into the frequency domain, the Model set creates a connected area called Value set [18, 20]. The Value set is defined as

$$\mathcal{V}_\omega^{n,m}(\kappa, \mu, \sigma^2) = \left\{ P(j\omega) : P(s) \in \mathcal{S}^{n,m}(\kappa, \mu, \sigma^2) \right\}, \; \forall \omega > 0. \tag{44}$$

The process uncertainty limits are defined by the Value set boundary. Value set boundary in the complex plane is denoted as $\partial\mathcal{V}_\omega^{n,m}(\kappa, \mu, \sigma^2)$ and it is generated by so-called extremal transfer functions. Extremal transfer function is an admissible transfer function defined as $P(s) \in \mathcal{S}^{n,m}(\kappa, \mu, \sigma^2)$ for which exists $\omega > 0$ such, that $P(j\omega) \in \partial\mathcal{V}_\omega^{n,m}(\kappa, \mu, \sigma^2)$. Set of all extremal transfer function is denoted as $\mathcal{S}_E^{n,m}(\kappa,\mu,\sigma^2)$. This set is independent of frequency ω if the conditions (37) and (42) are met.

Since the processes of the order less than one do not have an equivalent in the real world, and they extend the Model set uncertainty, it is acceptable to define the minimum pole order m as $m = 1$. However, the maximum process order n does not need to be restricted because the Model set uncertainty converges very quickly for $n \to \infty$, and the generated extremal processes are quite easier to simulate in the time domain. Therefore, the normalized Model set depends only on $\bar{\sigma}^2$. It can be denoted as $\mathcal{S}^{\infty,1}(\bar{\sigma}^2)$. The set of its extremal processes can be denoted as $\mathcal{S}_E^{\infty,1}(\bar{\sigma}^2)$. The set $\mathcal{S}^{\infty,1}(\bar{\sigma}^2)$ is non-empty if and only if $\bar{\sigma}^2 \in \langle 0, 1\rangle$ [20]. If $n \to \infty$ and $m = 1$, then for any $\omega > 0$ the Value set boundary $\partial\mathcal{V}_\omega$ of the normalized Model set $\mathcal{S}^{\infty,1}(\bar{\sigma}^2)$ consists of three arcs $P_i(s = j\omega, \alpha), \alpha \in I_i, i = 1, 2, 3$. Arcs are defined as

$$P_1(s,\alpha) = \frac{e^{-(1-\sigma\sqrt{\alpha})s}}{\left(\frac{\sigma}{\sqrt{\alpha}}s+1\right)^{\alpha}}, \tag{45}$$

$$P_2(s,\alpha) = \frac{1}{\left(\frac{\alpha-\sqrt{\alpha m[(m+\alpha)\sigma^2-1]}}{\alpha(m+\alpha)}s+1\right)^{\alpha}\left(\frac{m+\sqrt{\alpha m[(m+\alpha)\sigma^2-1]}}{m(m+\alpha)}s+1\right)^{m}}, \tag{46}$$

$$P_3(s,\alpha) = \frac{1}{\left(\frac{m-\sqrt{\alpha m[(m+\alpha)\sigma^2-1]}}{m(m+\alpha)}s+1\right)^{m}\left(\frac{\alpha+\sqrt{\alpha m[(m+\alpha)\sigma^2-1]}}{\alpha(m+\alpha)}s+1\right)^{\alpha}}. \tag{47}$$

Intervals $I_i, i = 1, 2, 3$ are given as

$$I_1 = \left\langle m, \frac{1}{\sigma^2}\right\rangle,\ I_2 = \left\langle \max\left\{m, \frac{1-m\sigma^2}{\sigma^2}\right\}, \infty\right\rangle,\ I_3 = \left\langle \max\left\{m, \frac{1-m\sigma^2}{\sigma^2}\right\}, \frac{1}{\sigma^2}\right\rangle. \tag{48}$$

The mentioned equations for computing extremal processes for both IO and FO types of the Model set can be seen in detail in [50, 53, 57]. In Section 2.5.3, the set of extremal processes will be used for robust controller design. In Figure 13, we can see an example of value sets for varying frequency ω and fixed σ^2, and vice versa.

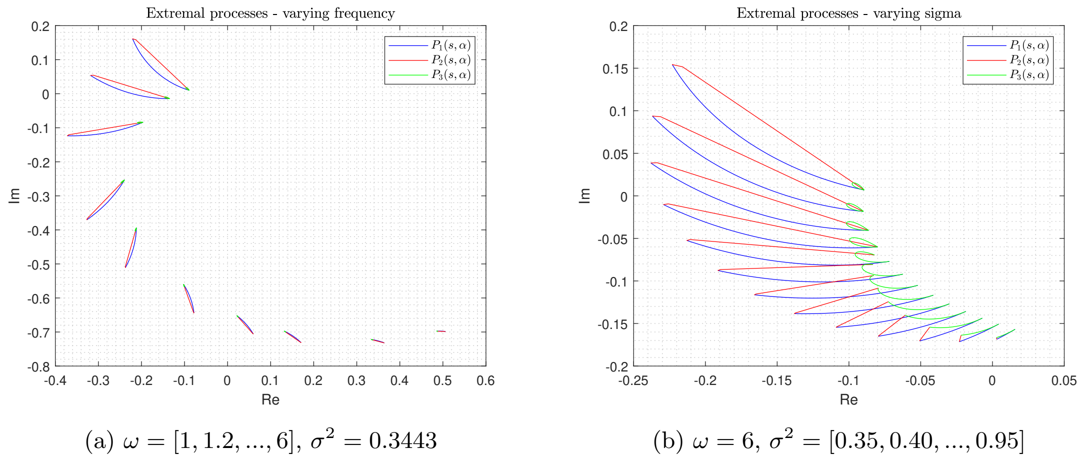


Figure 13: Extremal processes for varying frequency and σ^2; $\kappa = 1$, $\mu = 1$, $n = 10$, $m = 1$

2.5.3 Robust Controller Design

As it was shown in Figure 11, the last part of the Model set approach is the robust controller design. We want to obtain the set of controllers satisfying all selected design criteria for all processes from the Model set according to the diagram in Figure 12.

It can be proved that to find a solution, it is sufficient to consider only the extremal processes $\mathcal{S}_E^{\infty,1}(\bar{\sigma}^2)$ forming the boundary of the range of values in the frequency domain exactly defined in Section 2.5.2 [20]. This is a significant benefit since we do not have to consider all processes from the Model set. For the numerical computation of the region, the Value set boundary is sampled. Each arc is then represented by M processes.

Our aim is to obtain set controllers satisfying N design criteria for M extremal processes. This set is given by the intersection of sub-sets for each design requirement for each process. Each set of controllers (or robust stability region) is obtained by the procedure described in Section 2.4.

A robust controller is any controller from the final set. However, we are able to obtain the optimal robust controller by numerically evaluating the integral criteria of optimality for parameters from the final set. The optimal robust controller leads to the minimal value of chosen integral criterion. Integral criteria implemented in this thesis are described in Section 2.6. In Section 4.4.2, the gradient computation of integral criteria will be shown.

2.6 Integral Criteria of Optimality

After the resulting region is obtained, it contains an infinite number of controllers stabilizing the closed-loop system according to the chosen design criteria (for details see Section 2.3.1). However, we want to select one controller which leads to optimal closed-loop performance in the time domain. The closed-loop performance (or control quality) can be measured by the integral criteria of optimality [43]. The optimal closed-loop performance can be achieved by minimizing the integral criteria.

The integral criteria are time domain criteria that are widely used in academic journal papers and simulation studies [8, 15, 65, 67]. The following six criteria will be considered in this thesis

- **IE** – Integral of Error

$$IE = \int_0^{+\infty} e(t)dt, \tag{49}$$

- **ITE** – Integral of Time Error

$$ITE = \int_0^{+\infty} t \cdot e(t)dt, \tag{50}$$

- **ITAE** – Integral of Time Absolute Error

$$ITAE = \int_0^{+\infty} t \cdot |e(t)|dt, \tag{51}$$

- **IAE** – Integral of Absolute Error

$$IAE = \int_0^{+\infty} |e(t)|dt, \tag{52}$$

- **ISE** – Integral of Square Error

$$ISE = \int_0^{+\infty} e^2(t)dt, \tag{53}$$

- **IGSE** - Integral of Generalized Square Error

$$IGSE = \int_0^{+\infty} [e^2(t) + \alpha \cdot \dot{e}^2(t)]dt, \tag{54}$$

where $e(t)$ is the control error, and α is a weighting parameter for the squared first derivative of control error in (54). The higher value of α leads to the higher value of IGSE criterion [65].

In this thesis, the obtained robust stability region is sampled, and for all controllers, the closed-loop performances are numerically evaluated according to the described criteria. For each integral criterion, the minimum is obtained. We have implemented standard and gradient optimization methods for integral criteria minimization. This optimization process will be described in detail in Section 4.4.

Important to mention, the complexity of the optimization process does not depend on the choice of integral criteria. The optimization could be performed for various signals, such as closed-loop output $y(t)$ or controller output $u(t)$. It would be possible to define new previously unspecified integral criteria in the form of a norm of the selected signal. This criterion would be then used for minimization. The optimization complexity would not increase. In Section 4.4, the 3D visualization of 2 and ∞ norms for closed-loops from the resulting region will be shown.

3 State of the Art of Current Controller Design and Process Identification Tools

Part of this thesis was the analysis of the state of the art of current tools and methods for process identification and controller design.

In terms of controller design, an analysis of some current controller design tools was made in the bachelor thesis [66]. That analysis was mainly focused on PIDlab [26], Calerga software Sysquake [14], and powerful Matlab controller design tools, such as Control System Tuner [60] and Control System Toolbox [59].

PIDlab focuses on the interactive design of the robust controller using Nyquist plot shaping, resulting in the robust stability regions containing all solutions for specified design criteria (if this problem has a solution) [26]. Although PIDlab is suitable primarily for non-oscillatory or slightly oscillatory linear systems with dead time, it is possible to define an oscillatory linear system and design a controller. However, there are still some gaps in terms of sensitivity functions and Nyquist plot shaping. It is an expert task to operate with the m-circles. It is not guaranteed that PIDlab will prevent the Nyquist function to encircle the m-circles from the top which would lead to system and controller instability. Moreover, Nyquist plot shaping becomes more complicated if the given system is unstable. There is also no implemented solution for suppressing sensitivity functions $S(s)$ and $T(s)$ on certain frequency intervals which would improve the reduction of load disturbances and set point tracking. This method is not implemented for various reasons, one of them being its computational expensiveness [55].

The main advantage of the Sysquake was the interactive display of the process control variables in the time and frequency domain. Matlab tools provide great results in terms of computational time; they support multiple controller design methods and closed-loop optimization methods.

However, both environments only work with a nominal process model and do not provide all solutions for the specified problem. This unawareness of the set of all solutions can lead to a situation when the user can create unsatisfiable requirements for the closed loop. This results in the endless optimization process, where the stability of the closed loop can be endangered [66].

This thesis focuses on controller design and process identification. Hence, a few tools providing identification methods will be mentioned here. However, we will focus on the controller design as well.

In this thesis, we will focus on the PID H_∞ Designer [45]. It provides a user-friendly environment for robust controller design in a few steps and other useful features for process analysis and identification.

Next, we will analyze some blocks from the REXYGEN library from the REX Controls company[46]. Some provide process identification methods, and others controller tuning methods.

Lastly, we will focus on the System identification toolbox from MathWorks [62]. This toolbox provides Matlab functions, Simulink blocks, and a user-friendly interface with many interesting features for process identification.

3.1 PID H_∞ Designer

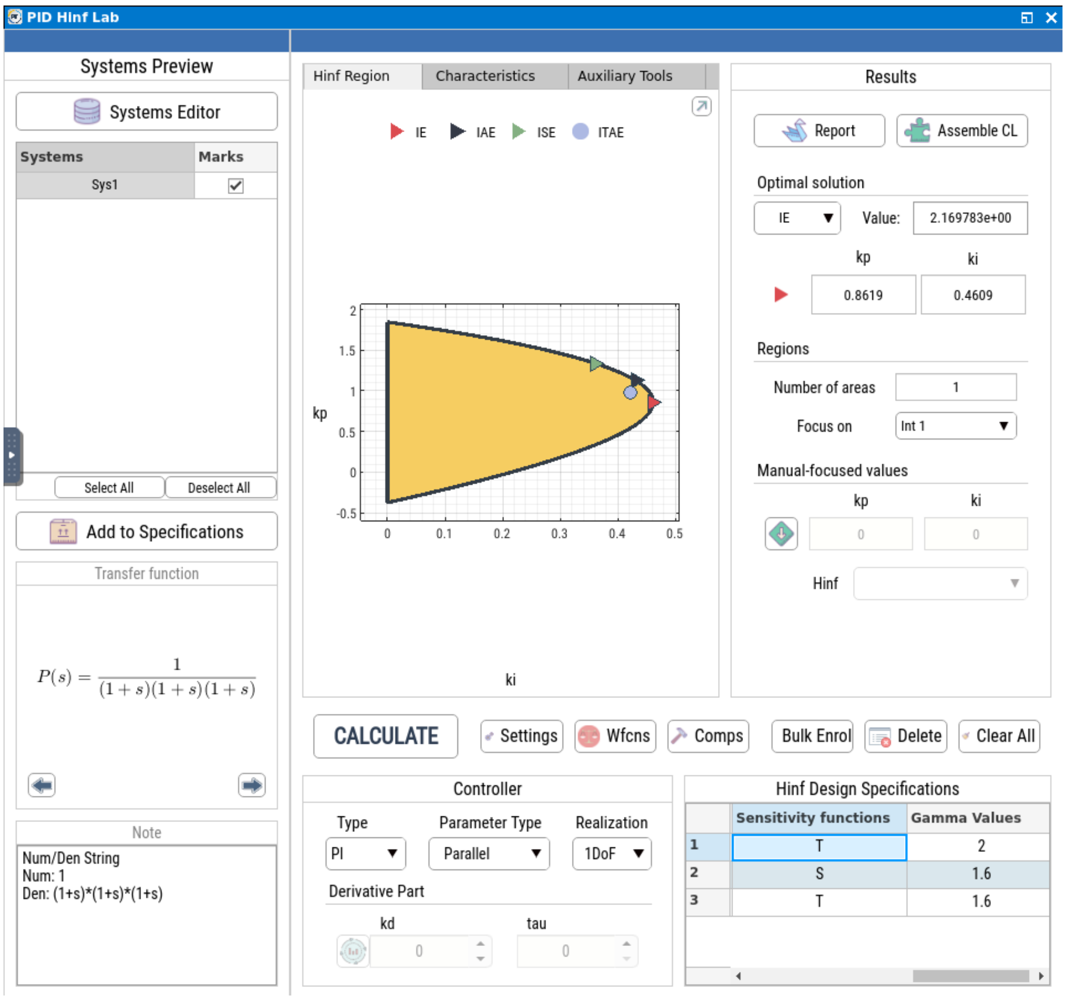
First and perhaps most significant tool for robust controller design is the PID H_∞ Designer developed by REX Controls company. Aim of this tool is to easily design and tune robust and optimal PID controller. Controller optimality is evaluated by time domain integral criteria, the PID H_∞ Designer implements Integral of Error, Integral of Absolute Error, Integral of Square Error and Integral of Time Absolute Error as integral criteria (IE, IAE, ISE, ITAE). The Designer also provides many tools for process and controller analysis [45].

The controller design is based on defining limits γ or weighting functions $W_S(s)$ and $W_T(s)$ representing restrictions for sensitivity function $S(s)$ and complementary sensitivity function $T(s)$ according to the equations (26), (27), (31), (32), (33), (34), (35), (36) mentioned in Chapter 2.5. It is possible to select weighting function of limit for $CS(s)$, $PS(s)$ as well. Correct sensitivity function shaping will lead to the required behavior of the closed loop.

We start in the System Editor, where the process model is defined as a transfer function. It is possible to add multiple processes. It is also possible to go to the System Identification section and upload experimental input/output data into the file manager. PID H_∞ Designer uses Matlab commands to identify the process transfer function from the i/o data. There are several models of the transfer function, such as FOPDT, SOPDT, Second order with zero plus dead time, oscillating second order plus dead time, general transfer function, or black box. We can also add parametric uncertainty to our model after clicking the P.U. button.

Then the controller type is selected in the Controller section. H_∞ Designer provides serial and parallel PI or PID controller versions. We can choose between one or two degrees of freedom realization. In the Hinf Design Specification, we are setting the constraints for any of the sensitivity functions $S(s)$, $T(s)$, $CS(s)$, $PS(s)$ for any defined system we choose. We can choose either a constant value or a specific transfer function as a constraint. Surprisingly, we can not select simple stability margins GM or PM for the Nyquist curve, as it was possible in the environment for robust controller design PIDlab which was previously developed at the Faculty of Applied Sciences [26]. Design requirement GM or PM would have to be expressed as the weighting function $W(s)$ or maximal magnitude γ for the sensitivity functions. That would be an expert task for the controller designer.

Requirements for the sensitivity functions are then processed. The result is a robust stability region formed by the intersection of regions each belonging to a selected constraint for a sensitivity function for each process. Then we can select the optimal solution from the list of integral criteria (IE, IAE, ISE, ITAE). H_∞ Designer automatically finds the optimal values of the PID controller which minimize selected integral criteria. An example can be seen in Figure 14.



Systems	Marks
Sys1	✓

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

Note
Num/Den String
Num: 1
Den: (1+s)*(1+s)*(1+s)

Optimal solution: IE Value: 2.169783e+00

	kp	ki
▶	0.8619	0.4609

Regions
Number of areas: 1
Focus on: Int 1

Manual-focused values

kp	ki
0	0

Hinf:

Controller
Type: PI
Parameter Type: Parallel
Realization: 1DoF
Derivative Part
kd: 0
tau: 0

Hinf Design Specifications

	Sensitivity functions	Gamma Values
1	T	2
2	S	1.6
3	T	1.6

Figure 14: Controller tuning in PID H_∞ Designer [45]

Apart from this very effective and user-friendly controller design, the H_∞ Designer provides useful Characteristics and Auxiliary Tools.

First of the Characteristics is the Hinf Regions plot showing each region for each requirement. Next there is Intersection Boundaries plot which shows the intersection of all regions. All characteristics from this two plots can be seen in the All Curves Plot. In figure Nyquist of Open Loop we can see M-circles and the Nyquist curve of the open loop. Here we can observe the Nyquist loop shaping. In the Amplitude Frequency Response we can display amplitude response of each sensitivity function as well as the open loop. Step responses of measured variable $y(t)$, control variable $u(t)$ and control error $e(t)$ can be seen in Characteristics Step Response of Closed Loop and Step Response of Internal Closed Loop signals. Characteristics Response of Closed Loop to Input Step Disturbance and Response of Internal Closed Loop signals to Input Step Disturbance show responses of $y(t)$, $u(t)$ and $e(t)$ to the input step disturbance.

First of the Auxiliary Tools is called Find the Minimum Gamma Value. There is an iterative algorithm that computes the minimal value of the infinity norm for sensitivity and complementary sensitivity function with respect to the selected design criteria for sensitivity functions. It is possible to preset the top and bottom limit of the gamma value. This tool then shows the region of controller parameters for the minimal gamma value.

Next tool is the Multipoint Analysis which shows performance of the control loop with controller minimizing the value of integral criteria (IE, IAE, ISE, ITAE) either in time or frequency domain. We can see here the Nyquist plot of open loop, step response of closed loop and response to input step disturbance and amplitude frequency response. We can also select new point from the $[K_i, K_p]$ plane and the characteristics will be computed for it as well.

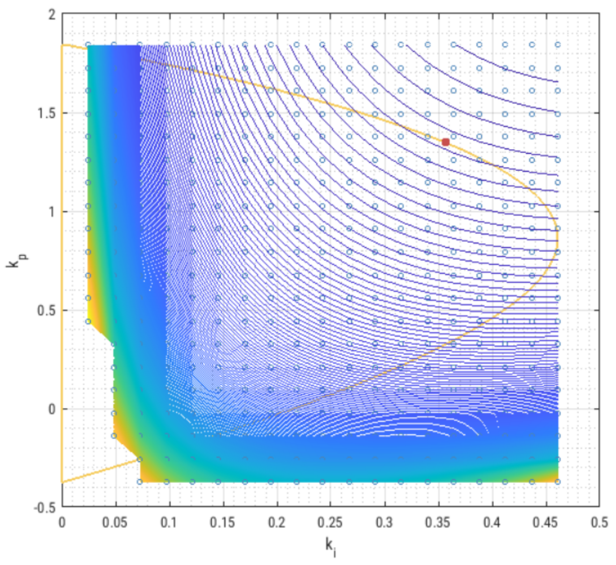


Figure 15: Performance Criteria Contour Line in PID H_∞ Designer [45]

Next tool is the Performance Criteria Contour Line which shows computed values of the integral criteria (IE, ISE or ITAE) for each sample of the intersection of regions as a surface of a 3D function above the region. It is possible to compute integral criteria for signals associated to reference tracking, input disturbance rejection or both at the same time. We can also select the optimal solution - this will highlight the minimal value of the selected integral criterion. To increase the precision of the graph, we can ad additional contour lines. It is possible to change grid density and contour line levels. Default value of grid density is 20, value of contour line levels is 300. If we set these numbers higher, the computation time will rise significantly. We can see that the integral criteria are being computed even for $[K_i, K_p]$ values which are not in the intersection of regions. This may or may not be helpful. The graph for the integral criteria has an offset in the K_i axis by 0.25 where the graph is not plotted. This has been preset by the developers of the H_∞ Designer because the values of the integral criteria for $K_i < 0.25$ were to high and would ultimately distort the 3D plot.

Next, there is the Open Loop Value-Set Region/s tool. It can generate processes from the one point of the frequency response Model set and its static gain. There are several parameters which values can be set by user, such as r - magnitude, φ - offset, ω_1 - measured frequency, n - order of the process, # - samples, and ω - frequency of the frequency response Model set. Processes from the Model set can then be added to the specification, and a solution according to design requirements can be computed for them in form of the intersection of regions. This tool can find which of these processes has the lowest robust. It is accomplished by finding the active frequency.

On this frequency, the frequency response of the closed loop from the Value set is equal to the M-circle, which represents the design requirement. This tool can visualise the moment Model set from the characteristic numbers κ, μ, σ^2 which can be obtained from PIDMA block from the REXYGEN environment.

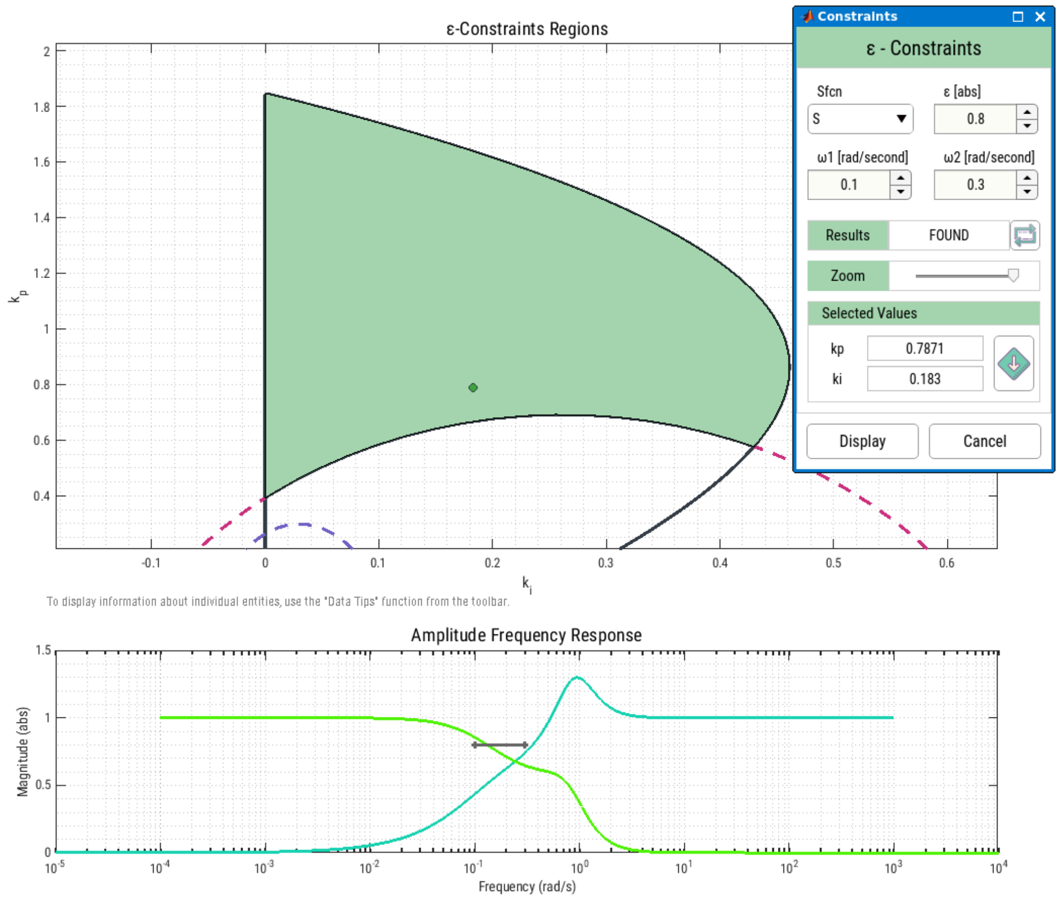
Multiparametric Analysis is the next tool. It provides several characteristics for each value of parameters Gamma, kd and tau. Parameter kd is the derivative gain, tau is a derivative time constant. Among the characteristics, there are Hinf Regions, Amplitude Frequency Response, Step Response of Closed Loop and Response of Closed Loop to Input Step Disturbance. This visualization can be very helpful for the controller designer because it provides better understanding of the systems behaviour with respect to the changes in the process parameters gamma, kd and tau. This tool serves for the analysis of PID controller design. The PID controller design using the H_∞ approach is more sophisticated than the PI controller design. PID controller has three parameters K_p, K_i and K_d, which causes a problem when solving the quartic equation to obtain the region's boundary [50]. Thus, the PID controller can be static, meaning that the K_d is selected *a priori*. Otherwise, numerical optimization methods must be applied to find PID regions from the quartic equation. These methods eventually lead to a sub-optimal solution. Furthermore, the solution has to exist for the PI controller to exist for the PID controller. Meaning, that if there is a process that can not be regulated by a PI controller, but could be by a PID controller, then the optimization would not detect the solution for the PID controller.

This is a different approach compared to the environment PIDlab [26]. In the PIDlab, the user could adjust the ratio T_i/T_d. This had a practical impact because it was user-friendly and well captured the feedback loop's physical reality, *e.g.*, the ratio 1/4 led to a band-stop filter.

Next auxiliary tool is called Tolerance Circles. It plots Nyquist curve of the nominal process together with several tolerance circles according to the user input. User can also change the value of gamma which represents the limitation for the $T(s)$. Default value is 20 tolerance circles and 1.2 for gamma.

Another tool is the ε-Constrains which allows user to set magnitude limit ε to the sensitivity function $S(s)$ on the frequency interval from ω_1 to ω_2. This tool then shows ε-Constrains region for $S(s)$ in $[K_i, K_p]$ plane which contains all possible controllers leading to that performance of $S(s)$. In Figure 16 we can see an example of ε-Constrains region for $S(s)$ for $\omega_1 = 0.1$ rad/second, $\omega_2 = 0.3$ rad/second, $\varepsilon = 0.8$. It can be well observed that a significant piece of region is missing when compared to the region in Figure 14. This is caused exactly by selected constrains for $S(s)$. The resulting solution is the intersection of the space outside the circles representing constrains for $S(s)$ with the region. This tool very well reflects how the requirements in the amplitude frequency domain affect the global solution for a robust controller.

Last, there is the Signal Response tool which simulates system output for several various types of signals on the input. Among the signals there is for example sine wave, step signal, impulse signal *etc.* Signals amplitude, frequency, bias *etc.* can be modified by user. As the input we can select signals $r(t)$, $d_i(t)$, $d_o(t)$, as the output there are signals $y(t)$, $e(t)$, $u(t)$.


ε-Constraints Regions
k_p
k_i
To display information about individual entities, use the "Data Tips" function from the toolbar.
Amplitude Frequency Response
Magnitude (abs)
Frequency (rad/s)
Constraints
ε - Constraints
Display
Cancel

Sfcn: S
ε [abs]: 0.8
ω1 [rad/second]: 0.1
ω2 [rad/second]: 0.3
Results: FOUND
Zoom
Selected Values
kp: 0.7871
ki: 0.183

Figure 16: ε-Constrains region for sensitivity function $S(s)$ in the PID H$_\infty$ Designer [45]

3.1.1 Conclusion

This tool provides many advantages in terms of robust PID controller design. The most significant advantage is that it finds all solutions for specified design requirements in the form of H$_\infty$ region. It provides an optimal controller by finding the minimum of the time domain integral criteria. Every computation is automatic. The next advantages of PID H$_\infty$ Designer are that it considers process uncertainty. It supports powerful Matlab commands for process identification and control. It provides useful characteristics in the time and frequency domain and additional tools for analyzing the problem we are currently dealing with. It is available online, and it is user-friendly.

Another advantage of this tool is the management of data. Parameters and variables can be stored in a file manager. Later, they can be transferred among the sections.

The disadvantage would be that we can not select simple stability margins, such as GM or PM, for the Nyquist curve, leaving us with an expert task to define design requirements as requirements for the sensitivity functions.

Another disadvantage would be the computational time of some functions. For example, the computation time of integral criteria over the region. This tool does not include methods for computational time optimization (*e.g.*, gradient methods). However, these methods often end in a local minimum, leaving us with a sub-optimal solution.

Also, saving the figures is a little bit more complicated. The figure is saved as an encrypted text. The user has to paste this text to another tool which translates it into the desired format.

3.2 REXYGEN Blocks

REXYGEN software from REX Controls company is widely used in various fields of automation, process control and robotics. REXYGEN Studio is a graphical programming environment with a generous library of so-called function blocks with implemented algorithms typical for process control [46]. There are blocks that implement controller tuning algorithms or identification algorithms. Some of them will be mentioned in this section.

The first block is the State controller for 2nd order system with frequency autotuner (SC2FA) (Figure 18a). It implements process identification and design of state controller for second order system with frequency auto-tuner. Two points of frequency response with a given phase delay are measured during the identification experiment. Identification is initialized by the rising edge of the signal on the input `RUN`. Its output is a harmonic signal with frequency ω increasing from ω_b to ω_f]. User must set phase delays φ_1 and φ_2. Their default values are $\varphi_1 = -60°$ and $\varphi_2 = -120°$, respectively, but these can be changed to arbitrary values within the interval $(-360°, 0°)$, where $\varphi_1 > \varphi_2$. From this phase delays, the identification algorithm obtains two points of frequency response which are successively used to compute the controlled process model in the form of

$$F(s) = \frac{b_1 s + b_0}{s^2 + 2\xi\Omega s + \Omega^2}, \tag{55}$$

where $\Omega > 0$ is the natural (undamped) frequency, ξ, $0 < \xi < 1$, is the damping coefficient and b_1, b_0 are arbitrary real numbers.

After successful identification, it is possible to generate the frequency response of the controlled system model, which is initiated by a rising edge at the `MFR` input.

Then, if we want to design a state controller for the identified process model (55), we have to switch to the "Controller mode". This mode has manual and automatic sub-modes. After the identification, the state controller design is performed automatically. A simplified inner structure of the frequency auto-tuning part of the controller can be seen in Figure 17. The diagram below shows the state feedback, observer and integrator anti-wind-up. The diagram does not show that the controller design block automatically adjusts the observer and state feedback parameters after the identification experiment [47].

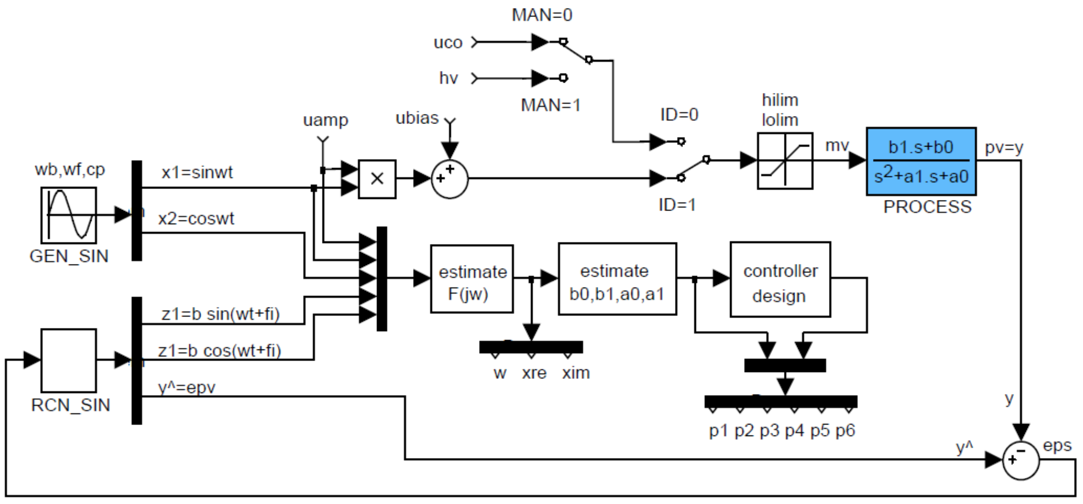


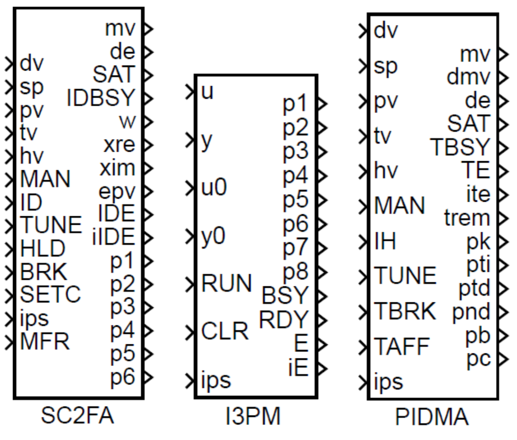
Figure 17: Diagram of the frequency auto-tuner implemented in the block SC2FA [47]

Next block is called Identification of a three parameter model (I3PM) (Figure 18b). The I3PM block is based on the generalized moment identification method. It takes input/output data, and provides a three parameter model of the system described by characteristic numbers $\{\kappa, \mu, \sigma^2\}$ mentioned in Section 2.5.2. It can also identify process in form of FOPDT model [47].

The last block picked for analysis in this thesis is PID controller with moment autotuner (PIDMA) (Figure 18c). This block extends the control function of the standard PID controller through the built-in auto-tuning feature. Before the start of the auto-tuner, the operator has to reach the steady state of the process at a suitable working point (in manual or automatic mode) and specify the required type of controller type (PI or PID) and other tuning parameters.

The identification experiment is started by the input `TUNE` and stopped by the input `TBRK`. In this mode, first of all, the noise and possible drift gradient are estimated during the user-specified time. Then the rectangle pulse is applied to the input of the process, and the first three process moments are identified from the pulse response. The pulse amplitude is set by the parameter `amp`. The pulse is finished when the process variable `pv` deviates from the steady value more than the `dy` threshold defines. The threshold is an absolute difference. Therefore it is always a positive value. The tuning experiment's duration depends on the process's dynamic behavior. The remaining time to the end of the tuning is provided by the output `trem`.

If the identification experiment is finished correctly and the input `ips` equals zero, then the optimal parameters immediately appear on the block outputs. In the opposite case, the output `ite` specifies the experiment error more closely. Other values of the `ips` input are reserved for custom-specific purposes. At the end of the experiment, the function of the controller depends on the current controller mode. If the `TAFF = on`, the designed controller parameters are immediately accepted [47].


SC2FA
dv sp pv tv hv MAN ID TUNE HLD BRK SETC ips MFR
mv de SAT IDBSY w xre xim epv IDE iIDE p1 p2 p3 p4 p5 p6
I3PM
u y u0 y0 RUN CLR ips
p1 p2 p3 p4 p5 p6 p7 p8 BSY RDY E iE
PIDMA
dv sp pv tv hv MAN IH TUNE TBRK TAFF ips
mv dmv de SAT TBSY TE ite trem pk pti ptd pnd pb pc

(a) SC2FA (b) I3PM (c) PIDMA

Figure 18: Identification and controller design blocks in the REXYGEN environment [47]

Illustration of mentioned blocks for process identification and controller design can be seen in Figure 18.

3.2.1 Conclusion

REXYGEN is a real-time control software. It is capable of the automatic tuning of the PID controller. In terms of identification, it implements the three-parameter model of the system, and thus it takes process uncertainty into account. Apart from the REXYGEN Studio, this software has more essential aspects. REXYGEN operates on a wide range of supported platforms, such as Raspberry Pi. It supports multiple communication protocols, such as Modbus, OPC UA, OPC DA, MQTT and SQL through ODBC [46].

However, in terms of controller design, it is more oriented toward the implementation of the already designed controller on a target device. The environment is not primarily determined for controller design or process identification. It might be complicated and unnecessary for users to use this environment for controller design and process identification. That is because, the controller design part is covered in the PID H_∞ Designer.

3.3 Matlab Identification Tools

Last part of the research was the analysis of Matlab identification tools. There are several commands for process identification in Matlab.

First, `idproc` model represents a system as a continuous-time process model with estimable coefficients. `idproc(Type)` creates a continuous-time process model with estimable parameters and sets the `Type` property. `Type` specifies aspects of the model structures, such as the number of poles in the model, whether the model includes an integrator, and whether the model includes a time delay. More generally, `idproc` can represent process models with up to three poles and one zero [63].

Next, the command `tfest` estimates the continuous-time transfer function from the input/output data. It is possible to specify the number of poles and the number of zeros. The input-output delay can be specified as well, or estimated by the command **delayest**. Input/output data can be from the time or frequency domain. This syntax can be used for SISO and MISO systems [63].

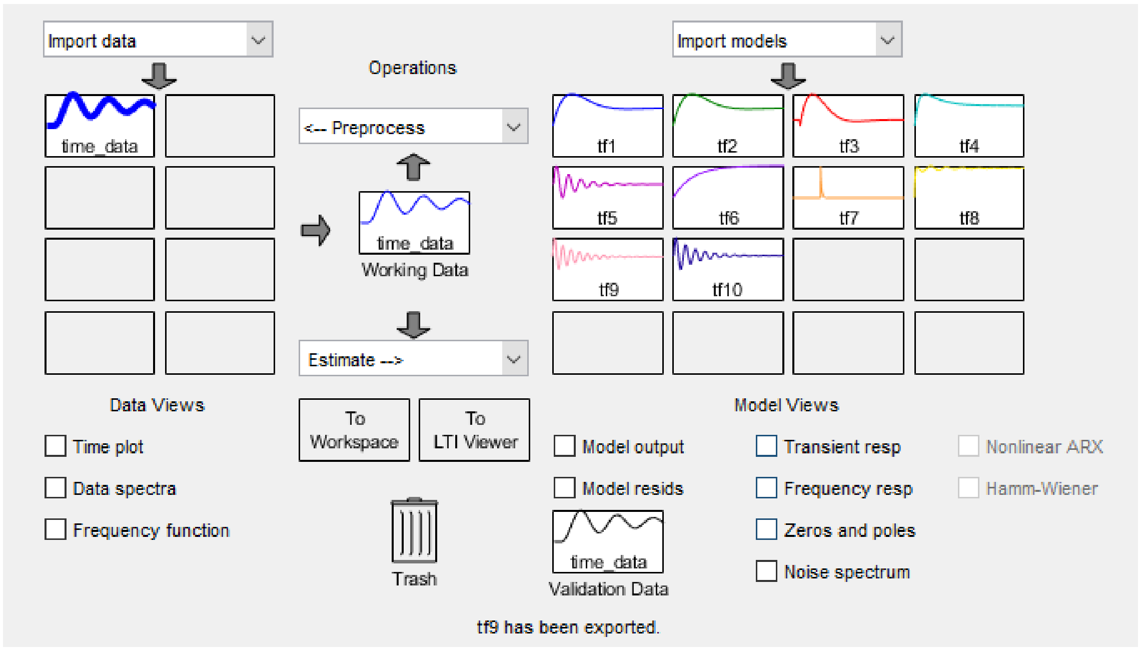
In the following section, we will focus mainly on the System identification toolbox, an interactive environment which includes mentioned commands for process identification.

3.3.1 System Identification Toolbox

One of the most up-to-date system identification toolboxes is the System identification toolbox designed by Mathworks. It is an app suitable for dynamic system modeling, time-series analysis, and data forecasting. It automatically estimates transfer functions, process models, and state-space models from either time domain data (i/o data) or frequency domain data (real and imaginary parts of process frequency response). In this toolbox, we can work either in continuous or discrete time. Besides process estimation, the toolbox can automatically generate C/C++ code for online estimation algorithms to target embedded devices [62].

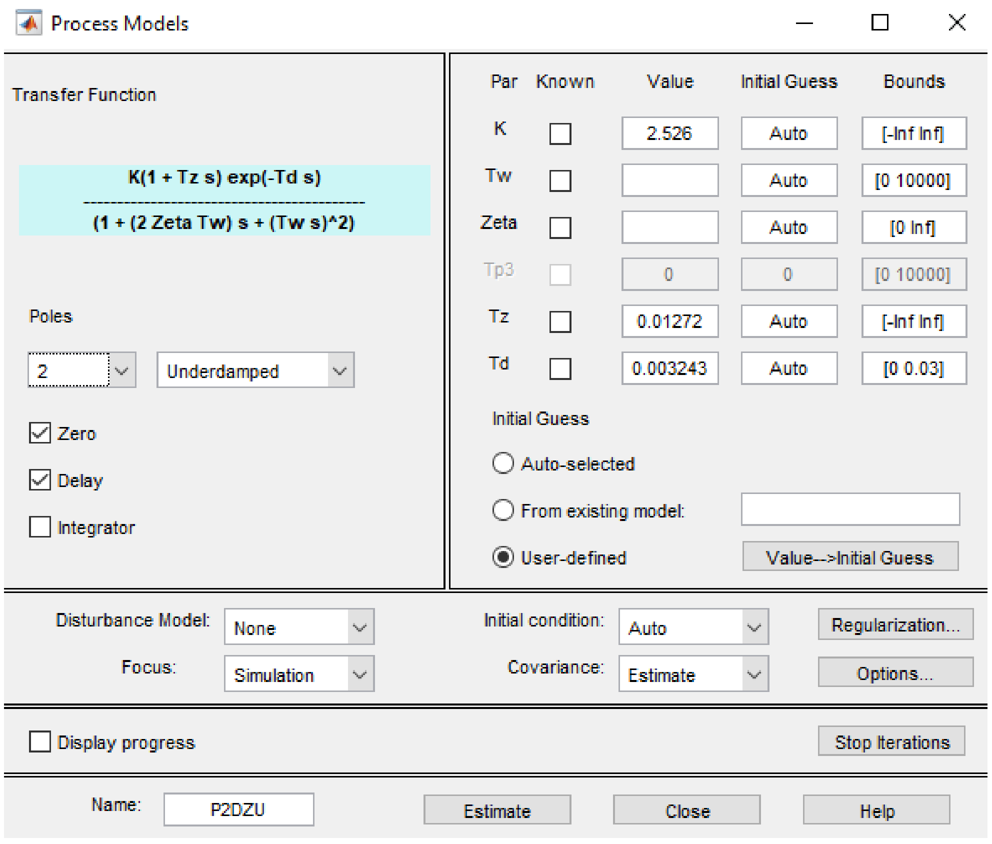
First, we import the data from the Matlab workspace. Then we have to select a form of model that we want to estimate. For instance, if we choose a Transfer Function Model, the tool lets us specify the number of poles and zeros of the estimated transfer function. It is also possible to include a delay. The delay can be fixed or automatically set by the solver. We can try to estimate several transfer functions, all of which can be seen in Model Views. An example is depicted in Figure 19.

If we want to use simple transfer functions with a fixed structure to approximate our process, we can select Process Models in the drop-down list of estimation options (Figure 20). A process model can have up to 3 poles. We can specify the system's damping, delay and whether or not it should have zero or an integrator. If we know the values of some parameters, we can fix their values or their bounds.



tf9 has been exported.

Figure 19: Matlab: System Identification Toolbox



Process Models

Transfer Function

```
      K(1 + Tz s) exp(-Td s)
-----------------------------------
(1 + (2 Zeta Tw) s + (Tw s)^2)
```

Poles: 2, Underdamped

☑ Zero

☑ Delay

☐ Integrator

Par	Known	Value	Initial Guess	Bounds
K	☐	2.526	Auto	[-Inf Inf]
Tw	☐		Auto	[0 10000]
Zeta	☐		Auto	[0 Inf]
Tp3	☐	0	0	[0 10000]
Tz	☐	0.01272	Auto	[-Inf Inf]
Td	☐	0.003243	Auto	[0 0.03]

Initial Guess

Auto-selected [✓ User-defined]

From existing model:

Value-->Initial Guess

Disturbance Model: None

Focus: Simulation

Initial condition: Auto

Covariance: Estimate

Regularization...

Options...

☐ Display progress

Stop Iterations

Name: P2DZU

Estimate Close Help

Figure 20: Matlab: System Identification Toolbox

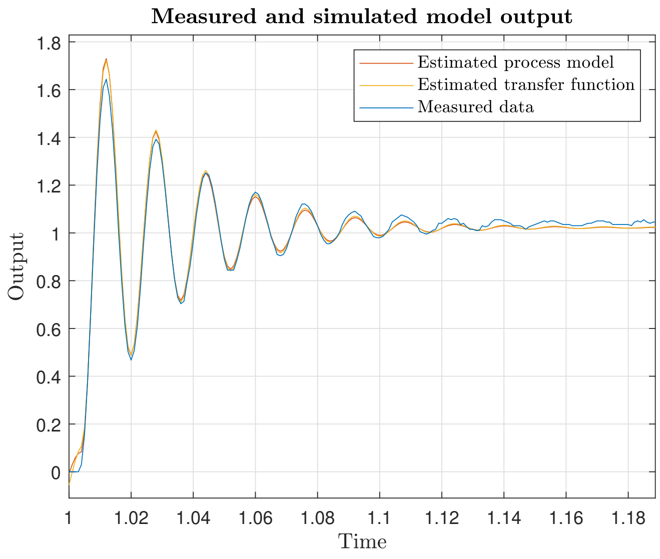


Figure 21: Comparison of measured data and identified systems

In Figure 21, we can see step responses of the measured process and two estimated process models (Transfer Function Model and Process Model). Transfer functions of these models have been estimated as

$$P_{TF}(s) = \frac{(21.92s + 1.579e05)}{(s^2 + 70.31s + 1.545e05)} e^{-0.003s}, P_{PM}(s) = \frac{(2.799e - 07s + 1.023)}{(6.457e - 06s^2 + 0.0004309s + 1)} e^{-0.00304s}.$$

Note: It is possible to estimate this models using commands `tfest` and `procest` in the Matlab Command Window.

The System Identification Toolbox and the models provide information about estimation accuracy. Three parameters give accuracy: Fit to Estimation Data (FED), Final Prediction Error (FPE) and Mean Square Error (MSE). In Table 4, we can see results for mentioned estimated models.

Estimated model	FED	FPE	MSE
Transfer Function	90.24%	0.0006705	0.0006826
Process Model	89.82%	0.0006304	0.0005856

Table 4: Accuracy of estimated models in System Identification Toolbox

These statistics show that the quality of both estimates is similar. Moreover, Matlab provides information about the uncertainty of estimated transfer function parameters. The command `getpvec` returns parameters of estimated process. It also returns the 1 standard deviation value of the uncertainty associated with the system parameters [63].

Another way to represent uncertainty in Matlab is the covariance matrix of estimated parameters. It is obtained by the command `getcov`. Covariance matrices of parameters from estimated transfer functions were computed as

$$\hat{\boldsymbol{P}}_{PM} = \begin{bmatrix} 0.0000 & -0.0000 & -0.0000 & -0.0001 & -0.0001 \\ -0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.0000 & 0.0000 & 0.0000 & 0.0004 & 0.0004 \\ -0.0001 & 0.0000 & 0.0004 & 0.2503 & 0.2503 \\ -0.0001 & 0.0000 & 0.0004 & 0.2503 & 0.2503 \end{bmatrix},$$

$$\hat{\boldsymbol{P}}_{TF} = \begin{bmatrix} 0.0001 & -0.0056 & -0.0000 & -0.0054 & 0.0000 \\ -0.0056 & 1.7358 & 0.0005 & 0.7815 & 0.0000 \\ -0.0000 & 0.0005 & 0.0000 & 0.0002 & 0.0000 \\ -0.0054 & 0.7815 & 0.0002 & 0.7089 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix} \cdot 10^6.$$

From the covariance matrices $\hat{\boldsymbol{P}}_{PM}$ and $\hat{\boldsymbol{P}}_{TF}$ we can see that the uncertainty of P_{TF} parameters is significantly bigger than uncertainty of the P_{PM} parameters, despite the estimated transfer functions P_{TF} and P_{PM} have been identified with almost the same precision (Table 4), and their response

From this results we can conclude that the representation of uncertainty in Matlab can turn out to be inconsistent. Moreover, it does not include *a priori* information about admissible processes which is essential for the Model set approach defined in Section 2.5. Therefore, it does not include extremal processes which create uncertainty boundary in the frequency domain.

4 Implementation of Advanced Tools for Simple Controller Design

This thesis aimed to improve the graphical user interface (GUI) for robust controller design, previously designed by the author [66]. The GUI was developed in the Matlab app building environment App Designer using Matlab version 9.4 (R2018a). In this thesis, the GUI was redesigned using Matlab version 9.10 (R2021a) [61, 63]. To access the GUI, the user has to enter command `appdesigner` to the Matlab Command Window.

The previously designed GUI implemented algorithms for robust controller design. It provided automatic computation of robust stability regions and their intersection for the PI controller. The GUI also provided visualization and analysis of 3D robust stability regions for PI$^\alpha$ controllers. A short description of the developed controller design tool will be mentioned in Section 4.1.

In this thesis, the GUI was extended by a module for process identification. This module implements standard Matlab commands for process identification and the first-order-plus-multiple-dead-time (FOPMDT) model identification method. It includes process uncertainty in the form of the Model set. The identification module will be introduced in Section 4.2.

In Section 4.3, we will show the possibility of robust controller design for processes with uncertainty in the GUI.

The last extension was a closed-loop time domain performance optimization tool which was searching for an optimal robust controller using the integral criteria of optimality. This tool implements standard Matlab commands and the gradient method for finding extrema. It will be described in Section 4.4.

4.1 Previously Developed Controller Design Tool

The tool for controller design implemented the robust region as described in Sections 2.3 and 2.4. In Figure 22, we can observe robust controller design and closed-loop characteristics in the GUI. The nominal process $P(s)$ used as an example has a form

$$P(s) = \frac{s+1}{s^2+5s+6}e^{-2s}. \tag{56}$$

The process was inserted and its coefficients were shown in table by clicking the `Load Process` button and `Show Processes` button, respectively. After that, the user could select multiple shaping points in the complex plane representing design requirements. This tool computed one robust region for each inserted process and each selected requirement. Then, the user had to choose a point from the intersection of regions. This point represented the $[K_i, K_p]$ coordinates of robust PI controller leading to closed-loop behavior satisfying all design criteria for all processes. The closed-loop performance could then be analyzed from multiple characteristics and plots.

Another useful feature was the visualization of three-dimensional PI$^\alpha$ robust regions. The PI$^\alpha$ controller is defined as

$$C(s) = K + \frac{K_i}{s^\alpha}, \tag{57}$$

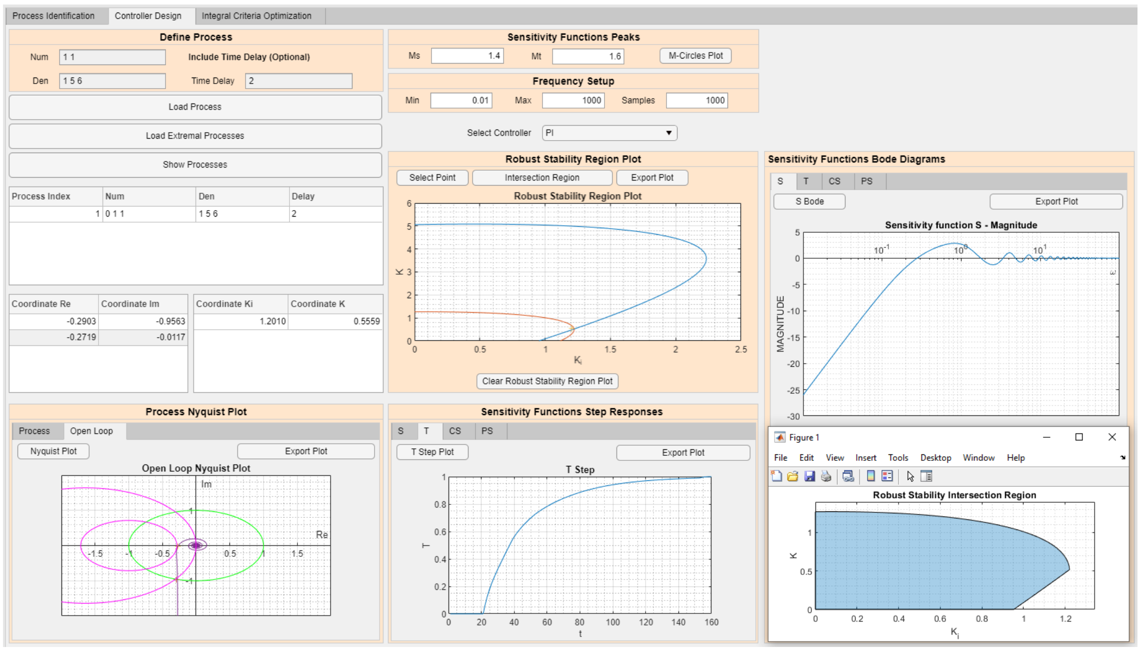
$$C(j\omega) = K(\omega) + \frac{K_i(\omega)}{(j\omega)^\alpha}, \tag{58}$$

where $\alpha \in \mathbb{R}^+$ is the parameter representing controllers fractional order. Controller gains $K(\omega)$ and $K_i(\omega)$ can be obtained as

$$K(\omega) = \frac{a(\omega)v\cos\phi + b(\omega)v\sin\phi + a(\omega)u\sin\phi - b(\omega)u\cos\phi}{(a^2(\omega)+b^2(\omega))\sin\phi}, \tag{59}$$

$$K_i(\omega) = \frac{\omega^\alpha(ub(\omega) - va(\omega))}{(a^2(\omega)+b^2(\omega))\sin\phi}, \tag{60}$$

where $\phi = \left(\frac{1}{2}\alpha\pi\right)$, $a(\omega)$ and $b(\omega)$ are real and imaginary part of the process, u and v are real and imaginary parts of the shaping point [15, 66]. An example of the PI$^\alpha$ robust region can be seen in Figure 23.


Process Identification
Controller Design
Integral Criteria Optimization
Define Process
Load Process
Load Extremal Processes
Show Processes
Sensitivity Functions Peaks
M-Circles Plot
Frequency Setup
Robust Stability Region Plot
Select Point
Intersection Region
Export Plot
Clear Robust Stability Region Plot
Sensitivity Functions Bode Diagrams
S
T
CS
PS
S Bode
Sensitivity function S - Magnitude
MAGNITUDE
ω
Process Nyquist Plot
Process
Open Loop
Nyquist Plot
Open Loop Nyquist Plot
Im
Re
Sensitivity Functions Step Responses
T Step Plot
T Step
t
Figure 1
File Edit View Insert Tools Desktop Window Help
Robust Stability Intersection Region
K
Ki

Num: 1 1 Include Time Delay (Optional)

Den: 1 5 6 Time Delay: 2

Ms: 1.4 Mt: 1.6

Min: 0.01 Max: 1000 Samples: 1000

Select Controller: PI

Process Index	Num	Den	Delay
1	0 1 1	1 5 6	2

Coordinate Re	Coordinate Im
-0.2903	-0.9563
-0.2719	-0.0117

Coordinate Ki	Coordinate K
1.2010	0.5559

Figure 22: Robust PI controller design in the GUI for process (56)

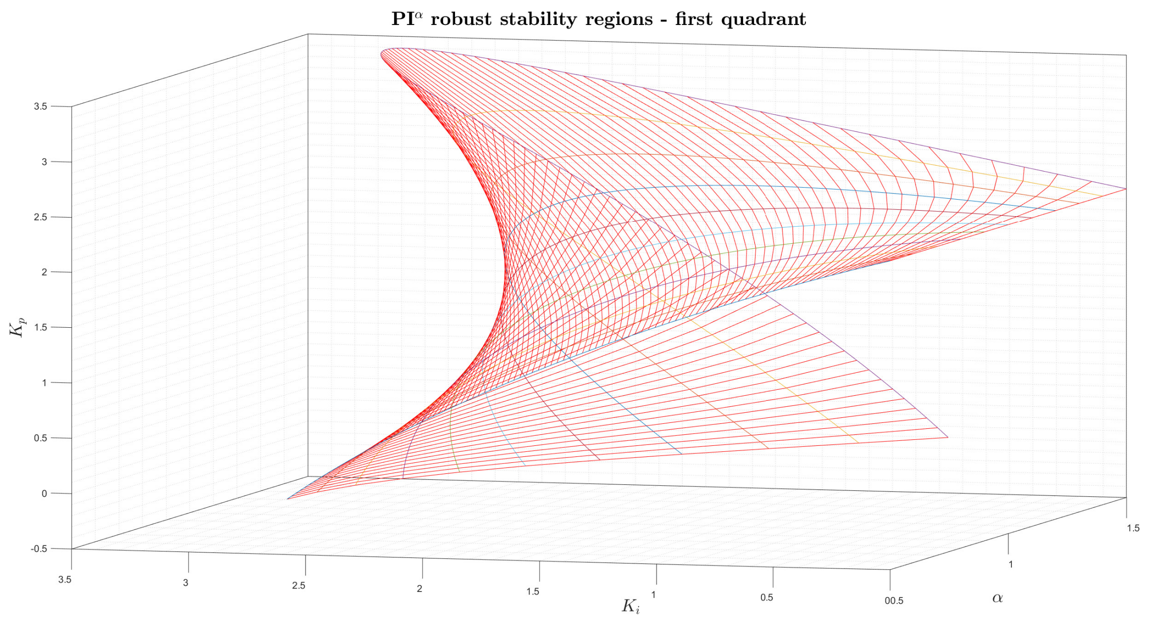
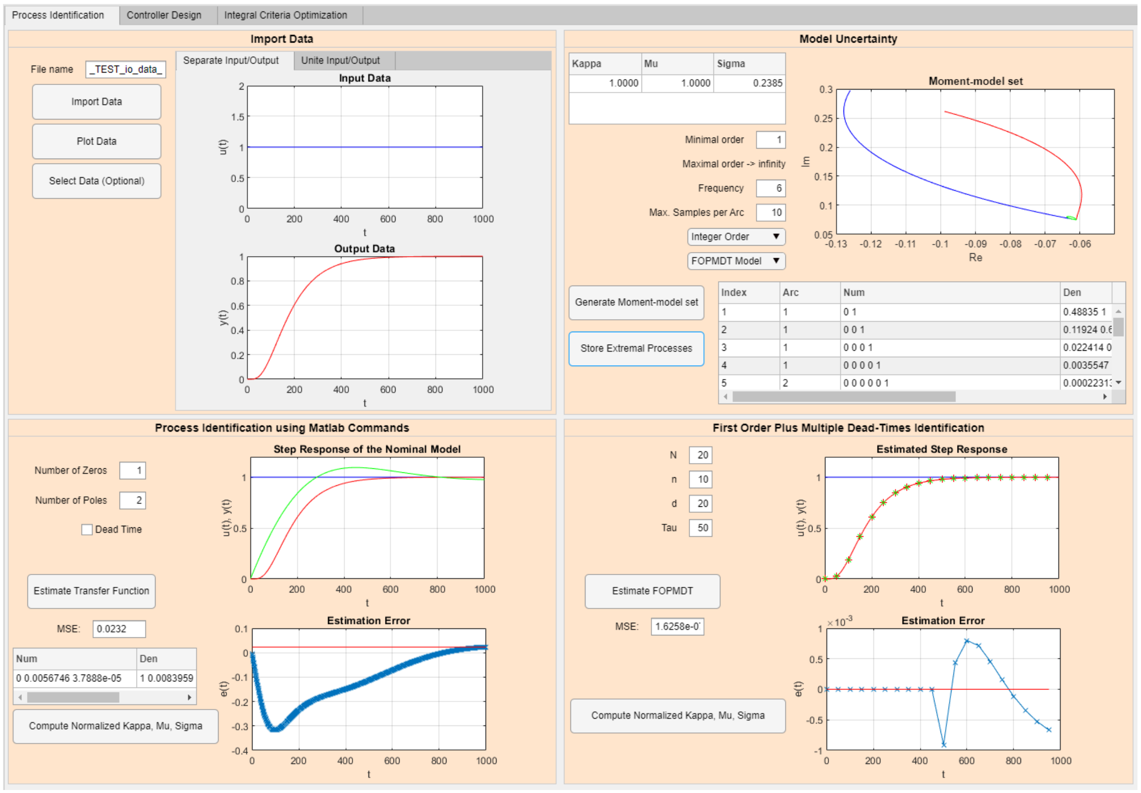


Figure 23: PI$^\alpha$ robust stability region for process (56)

4.2 Identification Module

The identification module consists of four sections (Figure 24). First, there is a section for input/output (i/o) data import. The data should contain the measurement of impluse or step response of the real process. It is beneficial, if the i/o data are in the steady state at the beginning, and at the end of an experiment.

After clicking the `Import Data` button, we can select a data file. Data have to be ordered into two columns, first for input data, and second for output data. It is recommended to use `.csv`, `.xlsx`, or `.txt` as a data file format. Data have to be in the same folder as the GUI app, otherwise, error occurs. After the file with i/o data is selected, it is important to wait a few seconds, otherwise, an interrupt error appears. After importing the data, they can be plotted by the `Plot Data` button. The user can also crop the data by clicking the `Select Data` button. The i/o data are then stored as a global variable which can be easily accessed in the GUI. After the data are imported, we can identify the process model.


Process Identification
Controller Design
Integral Criteria Optimization
Import Data
Import Data
Plot Data
Select Data (Optional)
Separate Input/Output
Unite Input/Output
Input Data
Output Data
Model Uncertainty
Moment-model set
Generate Moment-model set
Store Extremal Processes
Process Identification using Matlab Commands
Step Response of the Nominal Model
Estimation Error
Estimate Transfer Function
Compute Normalized Kappa, Mu, Sigma
First Order Plus Multiple Dead-Times Identification
Estimated Step Response
Estimation Error
Estimate FOPMDT
Compute Normalized Kappa, Mu, Sigma

File name: _TEST_io_data_

Kappa	Mu	Sigma
1.0000	1.0000	0.2385

Minimal order: 1
Maximal order -> infinity
Frequency: 6
Max. Samples per Arc: 10
Integer Order
FOPMDT Model

Index	Arc	Num	Den
1	1	0 1	0.48835 1
2	1	0 0 1	0.11924 0.6
3	1	0 0 0 1	0.022414 0
4	1	0 0 0 0 1	0.0035547
5	2	0 0 0 0 0 1	0.00022313

Number of Zeros: 1
Number of Poles: 2
☐ Dead Time
MSE: 0.0232

Num	Den
0 0.0056746 3.7888e-05	1 0.0083959

N: 20
n: 10
d: 20
Tau: 50
MSE: 1.6258e-0

Figure 24: The identification module

4.2.1 Process Model Identification Using Matlab Functions

The first method provides a nominal model identified using Matlab command `tfest` [63]. We can enter the desired number of poles, number of zeros, sampling period and whether we want the dead time or not. The identification procedure is initialized by clicking the `Estimate Transfer Function` button. After the computation is finished, the response of the identified model is compared with the data. The estimation error and mean square error (MSE) are evaluated. The estimation error $e[k]$ is given as

$$e[k] = y[k] - z[k], \tag{61}$$

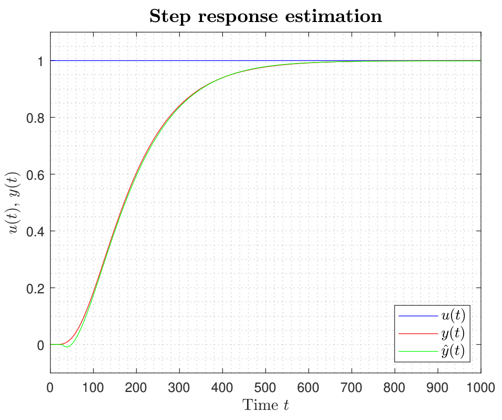
where $y[k]$ is the sample of output data, $z[k]$ is the sample of estimated output. The MSE is defined as

$$\text{MSE} = \frac{1}{N}\sum_{k=1}^{N}\left(y[k]-z[k]\right)^2, \; k = 1,\dots,N. \tag{62}$$

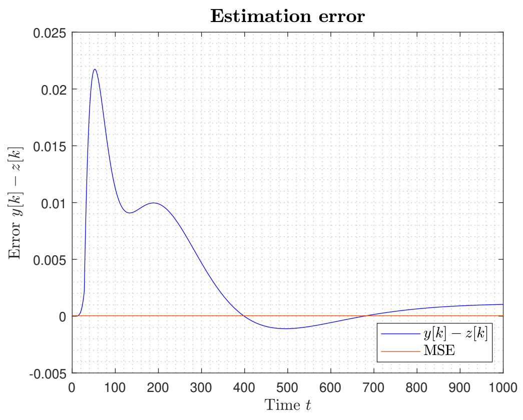
The coefficients of the process model are stored in a table and can be later used for controller design.

In Figure 25a we can see the step response of process $y(t)$ and estimated model response $\hat{y}(t)$ using Matlab function `tfest`. Number of poles n_p and number of zeros n_z were selected as $n_p = 2$ and $n_z = 1$, respectivelly. Dead time was not selected. In Figure 25b we can observe the estimation error $e[k]$ and MSE of the estimated model.

Since it is an expert task to estimate the correct number of poles and zeros from the i/o data, the identified process model is not always accurate. This method serves more as a comparison to the following method in Section 4.2.2.



(a) Step response estimation



(b) Estimation error

Figure 25: Process identification using Matlab function `tfest`

In the GUI, there is a possibility to include model uncertainty in the form of a Model set. The normalized characteristic numbers $\left\{\bar{\kappa},\bar{\mu},\bar{\sigma}^2\right\}$ are automatically computed after clicking the `Compute Normalized Kappa, Mu, Sigma` button. The GUI provides reliable results for the i/o data obtained from impulse response. Characteristic numbers are later used for Model set computation. The section in the GUI which implements the Model set will be described in Section 4.2.3.

4.2.2 FOPMDT Process Model Identification

The next method for process experimental identification implemented in this thesis is based on the assumption that the process model can be sufficiently approximated by the first-order-plus-multiple-dead-time (FOPMDT) model, for details, see [56]. This method consists of two steps. The first one is the FOPMDT model identification, and the second is the computation of the characteristic numbers $\{\kappa,\mu,\sigma^2\}$ from the identified transfer function. The FOPMDT model can be described by the formula

$$\hat{P}(s) = \frac{1}{\tau s+1}\sum_{i=0}^{n} A_i e^{-ids}, \tag{63}$$

where τ is a time constant with smoothing functionality, d is a time constant representing sampling period, A_i is the magnitude representing the weight of the i-th dead-time, n represents the number of dead-times used in the approximation. The detailed structure of the FOPMDT model can be seen in Figure 26.

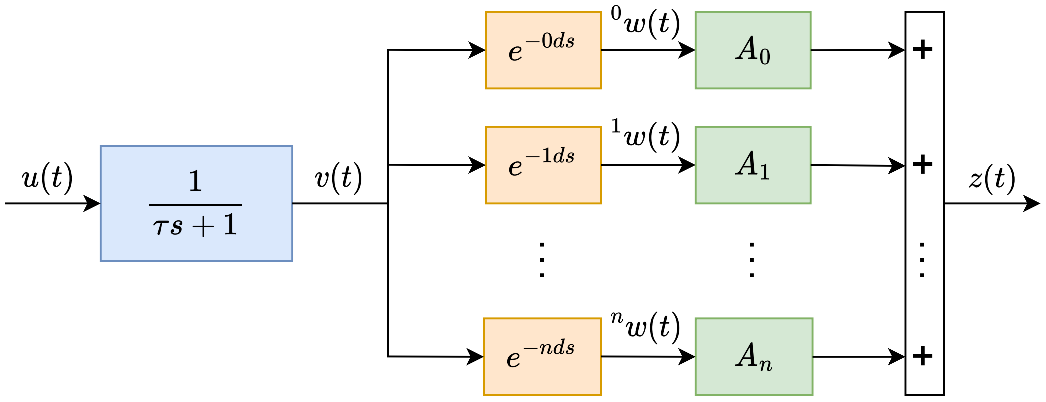


Figure 26: FOPMDT model structure

Involved continuous signals $u(t)$, $v(t)$, ${}^iw(t)$ and $z(t)$ can be sampled as

$$u_k = u(kd), \tag{64}$$
$$v_k = v(kd), \tag{65}$$
$${}^iw_k =^i w(kd), \tag{66}$$
$$z_k = z(kd). \tag{67}$$

From the diagram in Figure 26 we can determine the time change of continuous signal $v(t)$ as a derivative $\dot{v}(t)$ given as

$$\dot{v}(t) = \frac{1}{\tau}v(t) + \frac{1}{\tau}u(t). \tag{68}$$

After discretization we obtain

$$v_{k+1} = \alpha v_k + (1-\alpha)u_k, \tag{69}$$

where $\alpha = e^{-d/\tau}$. Next, the term ${}^iw(t)$ expressing the i-th delayed signal is represented as

$${}^iw(t) = v(t - id). \tag{70}$$

Discretized form is given as

$${}^iw_k = v_{k-i}. \tag{71}$$

Continuous and discrete forms of FOPMDT model output are expressed as

$$z(t) = \sum_{i=0}^{n} A_i^i w(t), \tag{72}$$
$$z_k = \sum_{i=0}^{n} A_i^i w_k. \tag{73}$$

It is assumed that the number of dead times n as well as both time constants τ and d are known. In our GUI it is up to the user to define its values. Parameter d represents sampling. If we set its value is too small, we can lose information about the process dynamics. If its value is too high, it can lead to resampling which can cause numeric problems. Parameter τ represents a smoothing function. It is good to select it in relation to d.

If it is supposed firstly that both time constants τ and d are known, then the magnitudes $A_0, \ldots, An$ may be determined by the mean square errors method. After the square-errors criterion is minimized, we obtain the function

$$I(\alpha) \triangleq \min_{A_0,\ldots,A_n} \left\{ \sum_{k=0}^{N} (y_k - z_k)^2 \right\}, \tag{74}$$

where y_k and z_k are discrete responses of the process and model, respectively, to the common input u_k, N is the number of samples. In the GUI, user can define the number of samples. However the higher its value is, the lower the estimation accuracy. The final model parameters of identification are determined by repeated minimization of $I(a)$ with respect to indeterminate variable $\alpha \in [0, 1]$ [56].

In Figure 27 we can observe a comparison of continuous response $y(t)$, discrete response y_k and estimated discrete response z_k to sine, relay, impulse and step input signal $u(t)$. Parameters were selected as $d = 30$, $\tau = 50$, $n = 20$, $N = 20$. In Figure 28 we can see the estimation error $e[k]$ and MSE for each estimated response. Estimation error and MSE were computed according to (61) and (62), respectively. Since the values of $e[k]$ and MSE are very low, we can assume that the identification was successful.

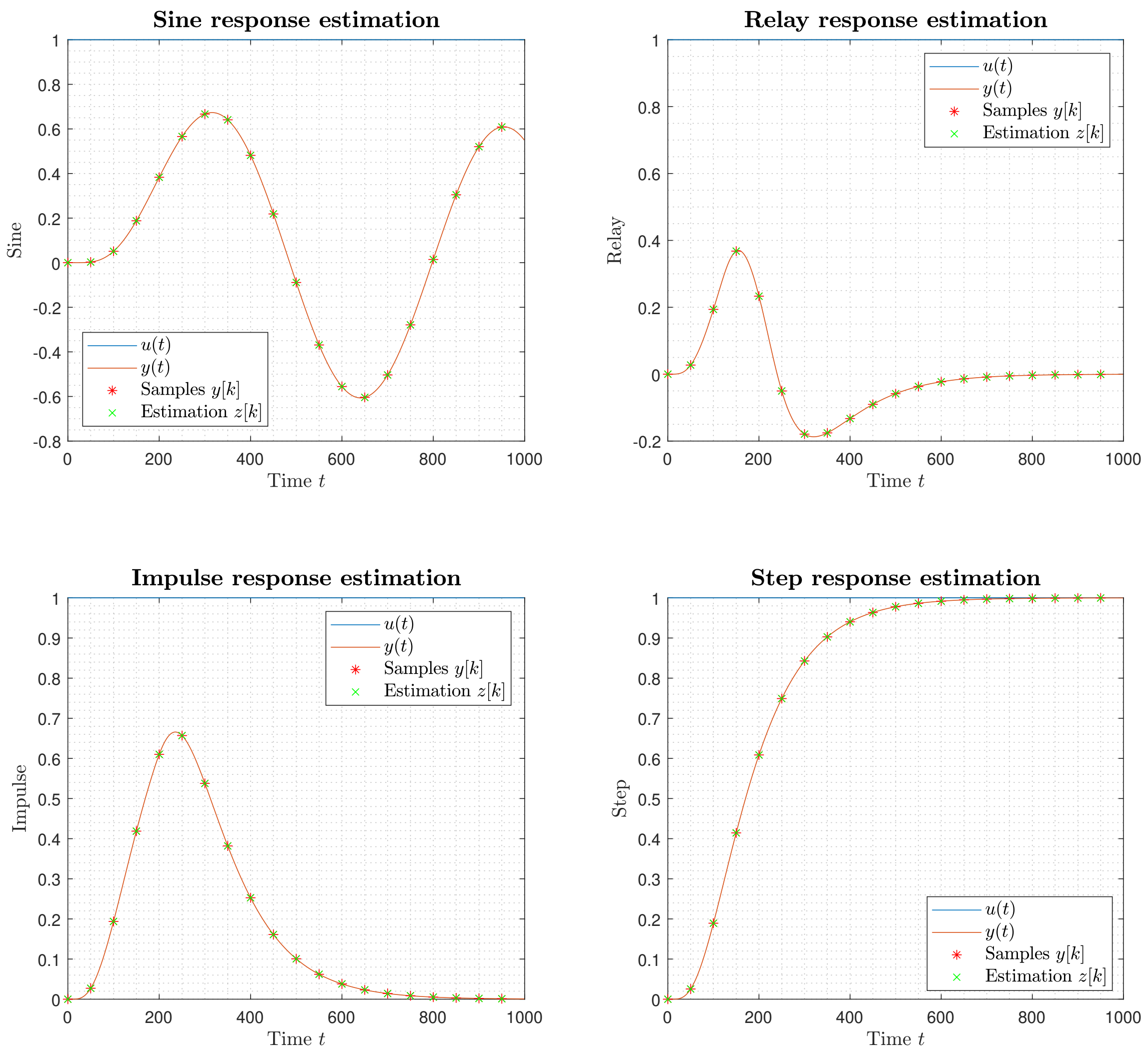


Figure 27: Process identification using FOPMDT model with sine, relay, impulse and step input signal, parameters were $d = 30$, $\tau = 50$, $n = 20$, $N = 20$

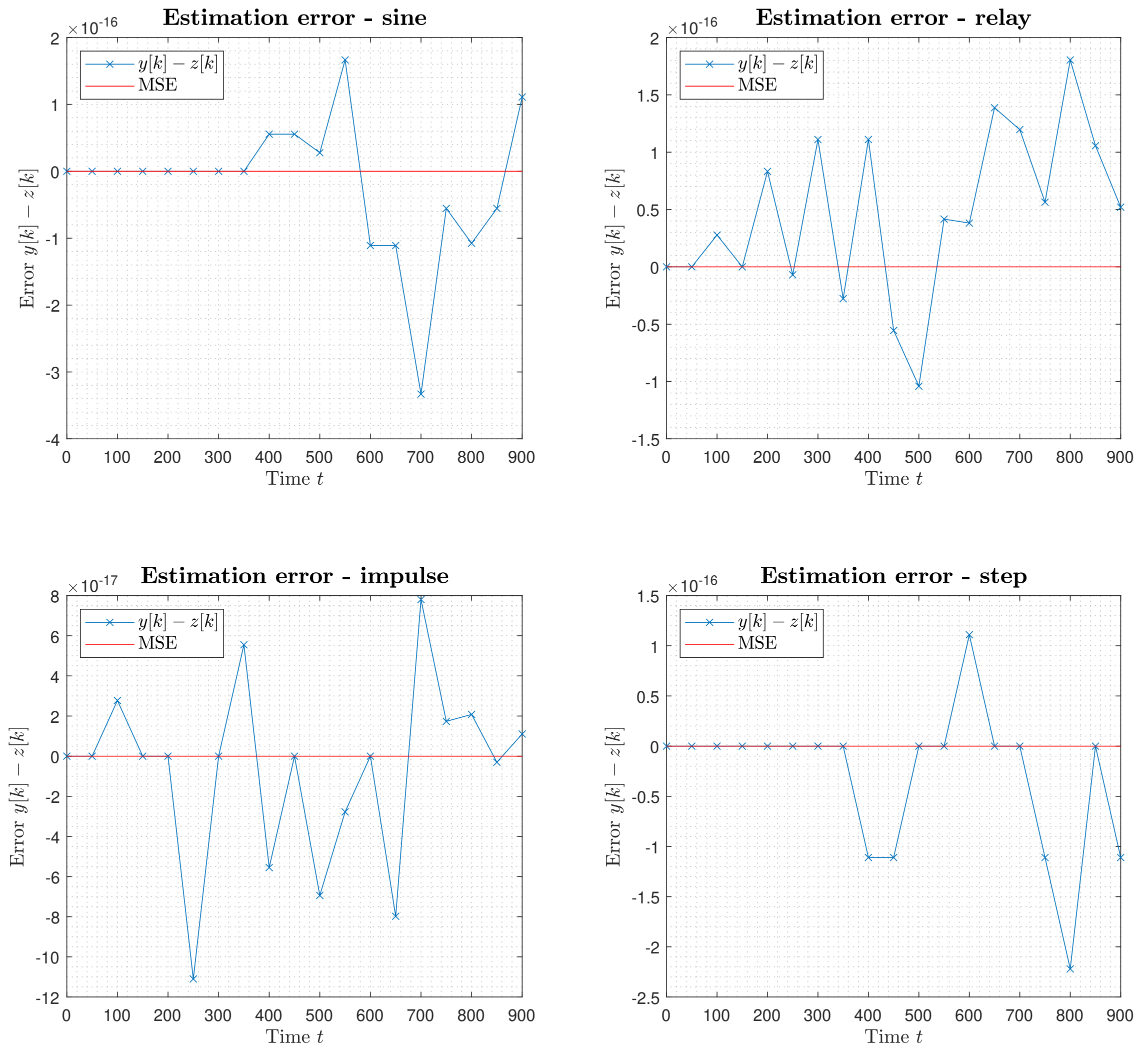


Figure 28: Estimation error $e[k] = y[k] - z[k]$ of the FOPMDT model for sine, relay, impulse and step input signal, parameters were $d = 30$, $\tau = 50$, $n = 20$, $N = 20$

Final step of this identification method is the computation of the characteristic numbers of the process. First four moments m_0, m_1, m_2, m_3 of the auxiliary transfer function

$$\sum_{i=0}^{n} A_i e^{-ids} \tag{75}$$

are defined as

$$m_0 = \sum_{i=0}^{n} A_i, \quad m_1 = d \cdot \sum_{i=0}^{n} i \frac{A_i}{m_0}, \quad m_2 = d^2 \cdot \sum_{i=0}^{n} i^2 \frac{A_i}{m_0}, \quad m_3 = d^3 \cdot \sum_{i=0}^{n} i^3 \frac{A_i}{m_0}. \tag{76}$$

The corresponding characteristic numbers of the process model $\sigma_0, \sigma_1, \sigma_2, \sigma_3$ are defined as

$$\sigma_0 = m_0, \quad \sigma_1 = m_1 + \tau, \quad \sigma_2 = m_2 - m_1^2 + \tau^2, \quad \sigma_3 = \frac{1}{2}m_3 - \frac{3}{2}m_1 m_2 + m_1^3 + \tau^3. \tag{77}$$

Then, if the assumptions described in Section 2.5.1 and 2.5.2 are met, the three characteristic numbers $\{\kappa, \mu, \sigma^2\}$ are given as

$$\kappa \triangleq \sigma_0, \ \mu \triangleq \sigma_1, \ \sigma^2 \triangleq \sigma_2. \tag{78}$$

The condition (37) then transforms to

$$P(s) = \frac{K}{\prod_{i=1}^{p}(\tau_i s + \alpha)^{n_i}}. \tag{79}$$

The characteristic numbers $\{\kappa, \mu, \sigma^2\}$ are then used for the Value set boundary computation according to the (45), (46), (47). In the GUI, the normalized characteristic numbers are automatically computed after clicking the `Compute Normalized Kappa, Mu, Sigma` button.

This method works well for processes with monotone response, and it is suitable only for the cases where identification starts in steady state. That does not conflict with the GUI which is designed to operate with processes with monotone response [56, 66].

4.2.3 Model Uncertainty

The last part of the identification module covers the process model uncertainty in the form of a Moment-model set. It is supposed that the assumptions described in Section 2.5.1 and 2.5.2 are met.

The section for model uncertainty can be seen in the up-right corner of the identification module. The whole procedure is implemented as a callback function of the `Generate Moment-model set` button. First, it loads the characteristic numbers $\{\kappa, \mu, \sigma^2\}$ from the table. Then, the Value set boundary is computed according to the equations (45), (46), (47).

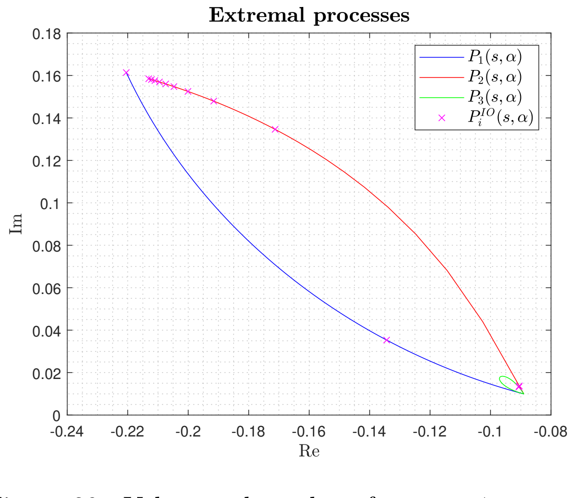


Figure 29: Value set boundary for $m = 1$, $n \to \infty$, $\omega = 6$

It is up to the user to specify a minimal order m, maximal order n, frequency ω, and sampling of the Value set boundary. It is possible to select a fractional or integer order version of the Value set. It can be also specified whether we want to work with Matlab or FOPMDT model.

In Figure 29 we can see the three arcs creating a Value set boundary for process minimal order $m = 1$, maximal order $n \to \infty$ and frequency $\omega = 6$. The Value set boundary is plotted for fractional order extremal processes $P_1(s, \alpha)$, $P_2(s, \alpha)$ and $P_3(s, \alpha)$ as it was described in (45), (46), (47). The boundary is also sampled creating integer order extremal processes $P^{IO}_{\{1,2,3\},i}(s, \alpha)$.

The set of IO extremal processes $P^{IO}_{\{1,2,3\},i}(s, \alpha)$ can be stored in the table by clicking the `Store Extremal Processes` button. This process set can be later used for robust controller design which will be described in Section 4.3.

4.3 Robust Controller Design for Process with Uncertainty

Since this thesis focuses on processes with uncertainty, part of the implementation focused on including the robust controller design for processes with uncertainty to the GUI.

Process uncertainty is included in the form of normalized Moment-model set $\mathcal{S}^{n,m}(\kappa, \mu, \sigma^2)$ in the Process Identification section in the GUI, where the characteristic numbers are obtained from the i/o data.

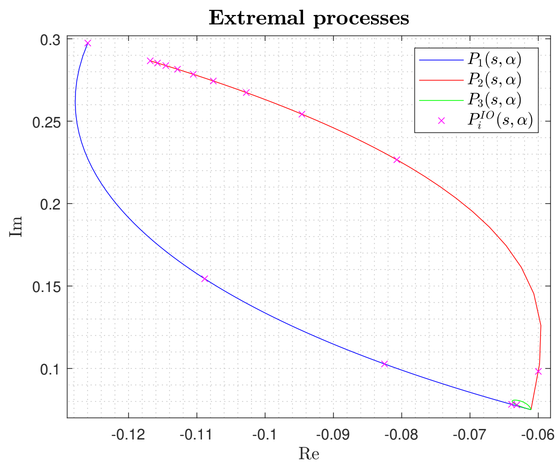


Figure 30: Value set boundary for $\{\bar{\kappa}, \bar{\mu}, \bar{\sigma}^2\} = \{1, 1, 0.2385\}$, $m = 1$, $n \to \infty$, $\omega = 6$, max. 10 samples per arc

As an example, we will use the characteristic numbers set

$$\{\bar{\kappa}, \bar{\mu}, \bar{\sigma}^2\} = \{1, 1, 0.2385\}. \tag{80}$$

After clicking the `Generate Moment-model set` button we are able to obtain the Value set boundary represented by FO extremal processes $P_1(s,\alpha)$, $P_2(s,\alpha)$ and $P_3(s,\alpha)$, see equations (45), (46) and (47). Since Matlab does not support fractional order transfer functions, each arc of the Value set boundary is sampled, creating a set of IO extremal processes $P^{IO}_{\{1,2,3\},i}(s,\alpha)$ as described in Section 4.2.3. For $n \to \infty$ it is possible to select the maximal value of samples per arc. In Figure 30 we can see three arcs represented by frequency responses of FO processes $P_1(s,\alpha)$, $P_2(s,\alpha)$, $P_3(s,\alpha)$ and IO processes $P^{IO}_{\{1,2,3\},i}(s,\alpha)$.

The coefficients of the IO extremal process set $P^{IO}_{\{1,2,3\},i}(s,\alpha)$ are then stored by `Store Extremal Processes` button. For our example, the IO extremal process set has a form

$$P^{IO}_{1,1}(s,\alpha) = \frac{1}{0.4884s+1}e^{(-0.512s)}, \tag{81}$$

$$P^{IO}_{1,2}(s,\alpha) = \frac{1}{0.1192s^2+0.6907s+1}e^{(-0.309s)}, \tag{82}$$

$$P^{IO}_{1,3}(s,\alpha) = \frac{1}{0.02242s^3+0.2385s^2+0.8459s+1}e^{(-0.154s)}, \tag{83}$$

$$P^{IO}_{1,4}(s,\alpha) = \frac{1}{0.003555s^4+0.05824s^3+0.3578s^2+0.9767s+1}e^{(-0.0233s)}, \tag{84}$$

$$P^{IO}_{2,1}(s,\alpha) = \frac{1}{0.0002231s^5+0.00631s^4+0.07014s^3+0.3808s^2+s+1}, \tag{85}$$

$$\vdots$$

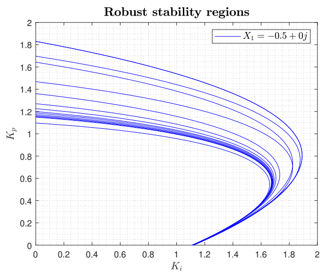
$$P^{IO}_{3,1}(s,\alpha) = \frac{1}{8.667e-05s^5+0.004959s^4+0.06676s^3+0.3807s^2+s+1}. \tag{86}$$

This process Model set can be loaded in the Controller Design section by clicking the `Load Extremal Processes` button. After the set is loaded, we can see the process model coefficients in the table (Figure 34). Now, it is our task to select the design criteria in the form of shaping points in the complex plane. This functionality is implemented as a callback function of the `Select Point` button above the Process Nyquist Plot. As an example, we have selected three shaping points $X_1 = u_1 + jv_1$, $X_2 = u_2 + jv_2$, $X_3 = u_3 + jv_3$ each representing one design requirement. Shaping points and matching design requirements are mentioned in Table 5.

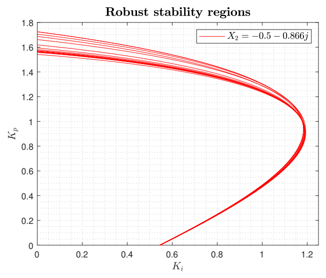
Index of shaping point	Coordinates	Design criteria
1. point	$X_1 = -0.5 + 0.000j$	GM = 2
2. point	$X_2 = -0.5 - 0.866j$	PM = 60°
3. point	$X_3 = -0.5 - 0.375j$	$M_S = 1.6$

Table 5: Shaping points

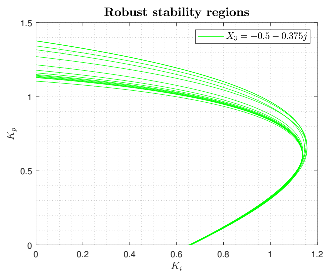
The GUI then automatically computes a robust stability region for each process from the $P^{IO}_{\{1,2,3\},i}(s,\alpha)$ set for each design requirement. Robust stability regions for design criteria X_1, X_2, X_3 can be seen in Figure 31.



(a) 1. shaping point X_1



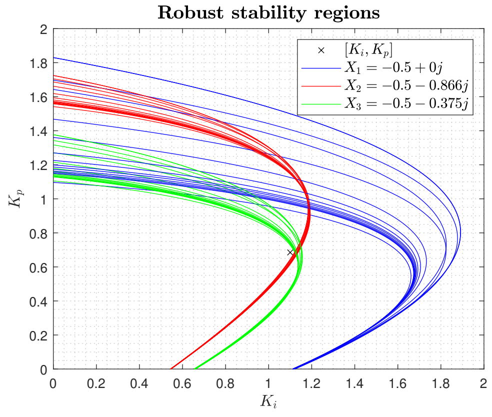
(b) 2. shaping point X_2



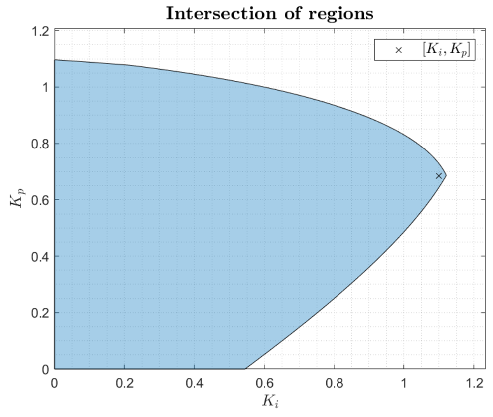
(c) 3. shaping point X_3

Figure 31: Robust stability regions for sampled IO extremal processes $P^{IO}_{\{1,2,3\},i}(s,\alpha)$ for shaping points X_1, X_2, X_3 representing design requirements described in Table 5

In the GUI, the regions are visualized in one plot (Figure 32a). After clicking the `Intersection Region` button, we can see the intersection (if it exists) of these regions (Figure 32b). In order to finish the robust controller design, we have to click the `Select Point` button to select $[K_i, K_p]$ coordinates from the intersection of robust stability regions, and thus to satisfy defined design requirements for the closed-loop. In this example, we have selected coordinates $[K_i, K_p] = [1.1, 0.685]$. It is recommended to select the point with the highest value of the K_i coordinate, hence, it leads to the minimal value of the IE criterion.



(a) Robust stability regions



(b) Intersection of regions

Figure 32: Intersection of Robust stability regions for sampled IO extremal processes $P^{IO}_{\{1,2,3\},i}(s,\alpha)$ for design requirements X_1, X_2, X_3 described in Table 5 together with the selected PI controller coordinates $[K_i, K_p] = [1.1, 0.685]$

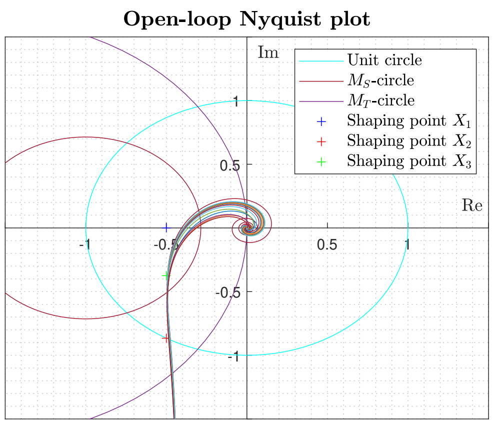


Figure 33: Open-loop Nyquist plot

This way we have designed the robust PI controller satisfying all defined design criteria for all IO extremal processes.

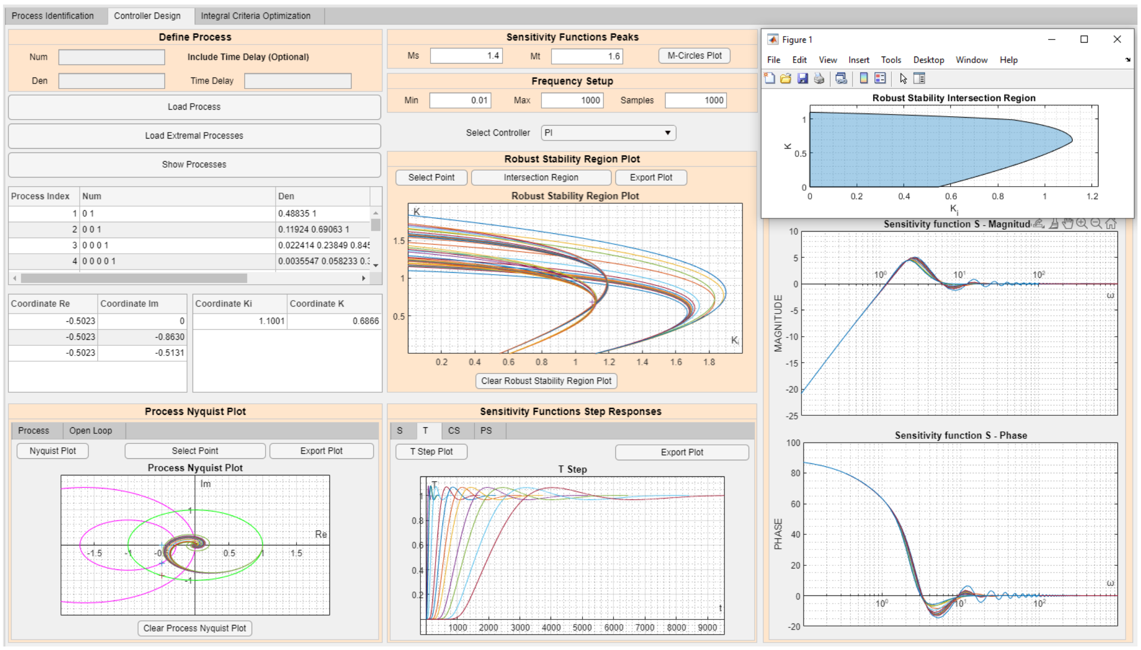
In Figure 33 we can see the Nyquist plot of the open loop in the form

$$L_i(s) = P^{IO}_{\{1,2,3\},i}(s,\alpha)C(s), \qquad (87)$$

where $i = 1, \ldots, N$, N is the number of extremal IO processes, and $C(s)$ is the designed robust PI controller

$$C(s) = K_p + \frac{K_i}{s} = 0.685 + \frac{1.1}{s}. \qquad (88)$$

In Figure 33 we can see that the Nyquist curve of the open loop passes on the right side of the specified design criteria X_1, X_2, X_3. Thus, according to the theory described in Section 2.4, those design criteria have been satisfied.



Process Identification
Controller Design
Integral Criteria Optimization
Define Process
Include Time Delay (Optional)
Load Process
Load Extremal Processes
Show Processes
Sensitivity Functions Peaks
M-Circles Plot
Frequency Setup
Robust Stability Region Plot
Select Point
Intersection Region
Export Plot
Clear Robust Stability Region Plot
Figure 1
File Edit View Insert Tools Desktop Window Help
Robust Stability Intersection Region
Sensitivity function S - Magnitude
MAGNITUDE
Sensitivity function S - Phase
PHASE
Process Nyquist Plot
Process
Open Loop
Nyquist Plot
Clear Process Nyquist Plot
Sensitivity Functions Step Responses
S
T
CS
PS
T Step Plot
T Step

Num:

Den:

Time Delay:

Ms: 1.4

Mt: 1.6

Min: 0.01

Max: 1000

Samples: 1000

Select Controller: PI

Process Index	Num	Den
1	0 1	0.48835 1
2	0 0 1	0.11924 0.69063 1
3	0 0 0 1	0.022414 0.23849 0.845
4	0 0 0 0 1	0.0035547 0.058233 0.3

Coordinate Re	Coordinate Im
-0.5023	0
-0.5023	-0.8630
-0.5023	-0.5131

Coordinate Ki	Coordinate K
1.1001	0.6866

Figure 34: Robust PI controller design for process with uncertainty in the GUI

The GUI further provides characteristics for verification of the closed-loops performance as can be seen in Figure 34. The set of closed-loop transfer functions is given as

$$F_i(s) = \frac{L_i(s)}{1 + L_i(s)} = \frac{P^{IO}_{\{1,2,3\},i}(s,\alpha)C(s)}{1 + P^{IO}_{\{1,2,3\},i}(s,\alpha)C(s)},\ i = 1,\dots,N. \tag{89}$$

In Figures 36 and 36 we can see the time and frequency characteristics of closed-loops each consisting of one $P^{IO}_{\{1,2,3\},i}(s,\alpha)$ process and designed PI controller.

In Figure 36 the user can observe step responses of closed-loop sensitivity functions $S(s)$, $T(s)$, $CS(s)$, $PS(s)$. From this Figure, we can see that all four sensitivity functions have required behavior.

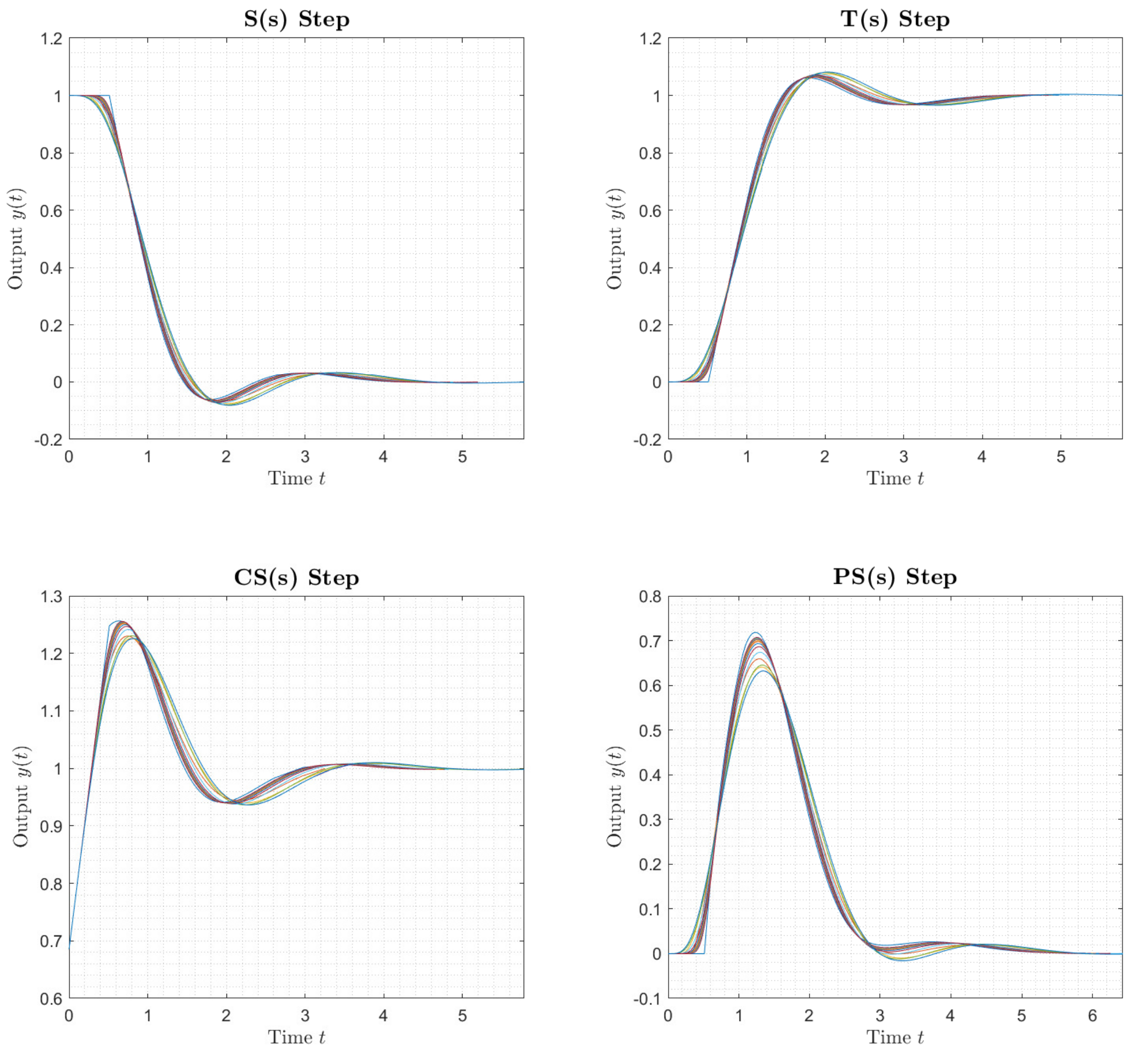


Figure 35: Closed-loop sensitivity functions $S(s)$, $T(s)$, $CS(s)$, $PS(s)$ step responses, PI controller parameters $[K_i, K_p] = [1.1, 0.685]$

In Figure 36 we can see the sensitivity functions $S(s)$, $T(s)$, $CS(s)$, $PS(s)$ magnitude and phase frequency responses in the form of Bode plot. Bode plots are plotted on the logarithmic scale on the frequency interval $\omega \in \langle 10^{-1}, 10^{3} \rangle$ rad/s. Due to the numerical issues the results of $T(s)$ and $PS(s)$ phase responses are slightly distorted.

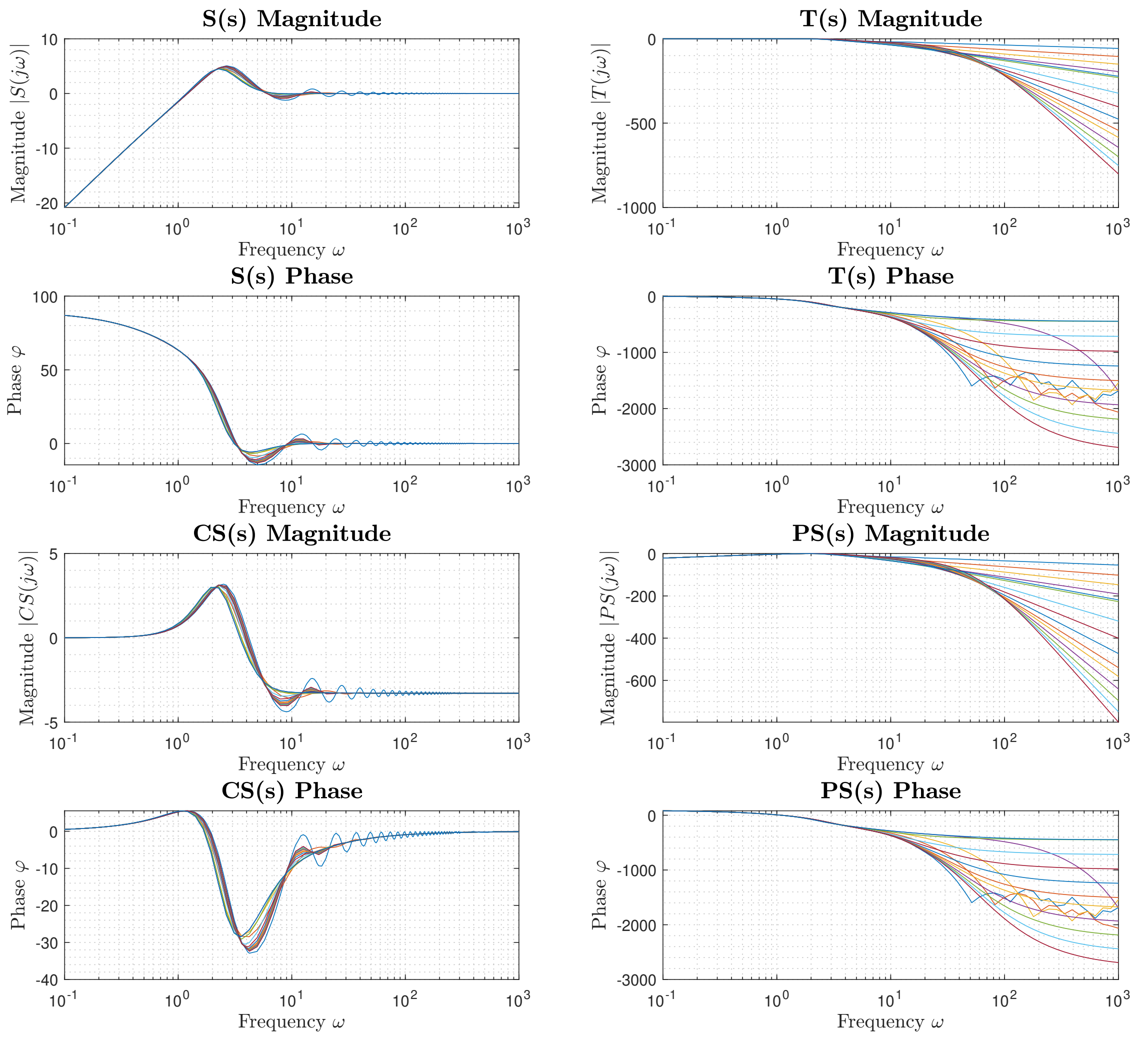


Figure 36: Closed-loop sensitivity functions $S(s)$, $T(s)$, $CS(s)$, $PS(s)$ frequency responses, PI controller parameters $[K_i, K_p] = [1.1, 0.685]$

4.4 Closed-Loop Time Domain Performance Optimization Tool

The next task of this thesis was to develop a tool for optimal robust controller search using the integral criteria of optimality mentioned in Section 2.6. This tool improves the author's previous work firstly introduced in [67]. It focuses on measuring the closed-loop performance in the time domain. Due to the computational and implementation complexity, this tool works only for one process model, and for an arbitrary number of design criteria.

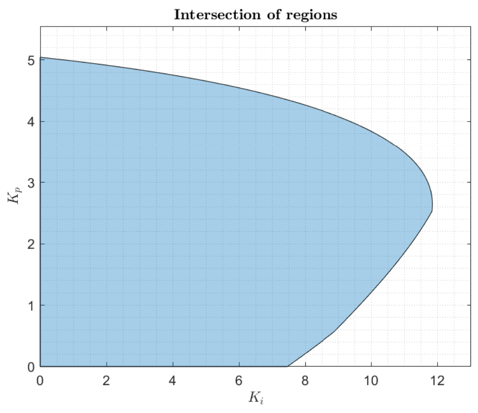
The aim of this tool is to sample the resulting intersection of robust stability regions and for each $[K_i, K_p]$ coordinate compute the control error $e(t)$ of the closed-loop. The control error $e(t)$

is then used for computation of IE, ITE, ITAE, IAE, ISE, and IGSE according to the equations (49), (50), (51), (52), (53), and (54). Next, the GUI automatically finds the optimal coordinates leading to minimal values of each integral criterion. In this tool, we have implemented standard minimization using the command `min` and gradient minimization.

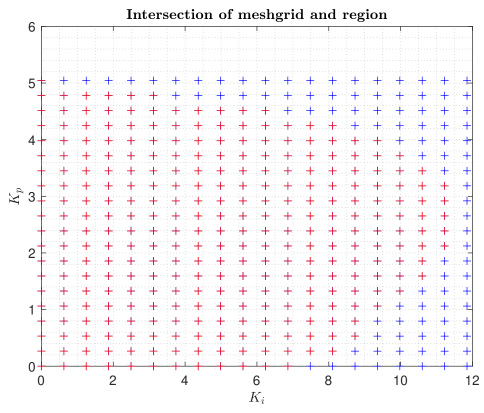
To illustrate the functionality of this tool, we have selected the process model

$$P(s) = \frac{s+1}{s^2+5s+6} e^{-0.2s}, \tag{90}$$

and three design criteria X_1, X_2, X_3 mentioned in Table 5 for optimal robust controller search. After the robust stability regions are computed for each design requirement, their intersection is automatically evaluated and stored as a Matlab `polyshape` object (Figure 37a). In order to evaluate the six mentioned integral criteria it is important to sample the intersection region. The region was sampled as a mesh grid matrix with its limits set as the maximal values of K_i, K_p coordinates. However, this sampling did not respect the irregular shape of the region creating a rectangular grid. A cubic approximation of the sampled region had to be performed on the obtained grid. As a result, it is then possible to approximate any irregular shape of the region for its further processing. The sampled area is shown in Figure 37b. Default sampling has been set to 20x20 samples per rectangular mesh grid which results in approximately three hundred samples of region coordinates $[K_i, K_p]$ [67].



(a) The resulting intersection of regions



(b) Sampled area of the intersection

Figure 37: Intersection of robust stability regions and its sampling

In the GUI, the intersection area sampling is done in the Integral Criteria Optimization section by clicking the `Create Meshgrid` and `Sample Region` buttons. Grid resolution can be selected as 20x20, 40x40, or 80x80 samples. The cubic approximation of the sampled region had one significant disadvantage since it did not include its border. In general, the shape of the intersection area border is irregular. Therefore, the borderline had to be additionally sampled and added to the cubical approximation of the area.

After the intersection of regions was sampled, all obtained controllers were used for closed-loop simulations. The performance of all closed-loops was evaluated according to the chosen integral criteria of optimality. In Figure 38 we can see the three-dimensional surface of all six assumed integral criteria.

The graphs depicted in Figure 38 show that all chosen criteria provide similar information. The more fundamental difference between them is only in the steepness as the criterion grows. This is primarily due to the contemplated shape of the system that was used as an example. In the case of an oscillatory system, it would be possible to obtain graphs with various shapes. However, the GUI focuses on essentially monotone processes, therefore, the surface shape will always be similar. The criteria for ITE and ITAE achieve the highest values for example system. Furthermore, for example, the ISE criterion provides the steepest course. Also worth mentioning here is the design parameter α in the IGSE criterion. It serves as a weight of the first derivative of control error $\dot{e}(t)$. Its default value is $\alpha = 0.5$. In the GUI user can adjust its value. For increasing α, the IGSE reaches higher values [67].

These graphs thus provide an interesting insight into the evaluation of the resulting region. Their results are used in further described optimization processes where we are searching for optimal $[K_i, K_p]$ coordinates which lead to the minimal value of each integral criteria of optimality. In Section 4.4.1 we will describe standard optimization. In Section 4.4.2, the gradient optimization method will be introduced.

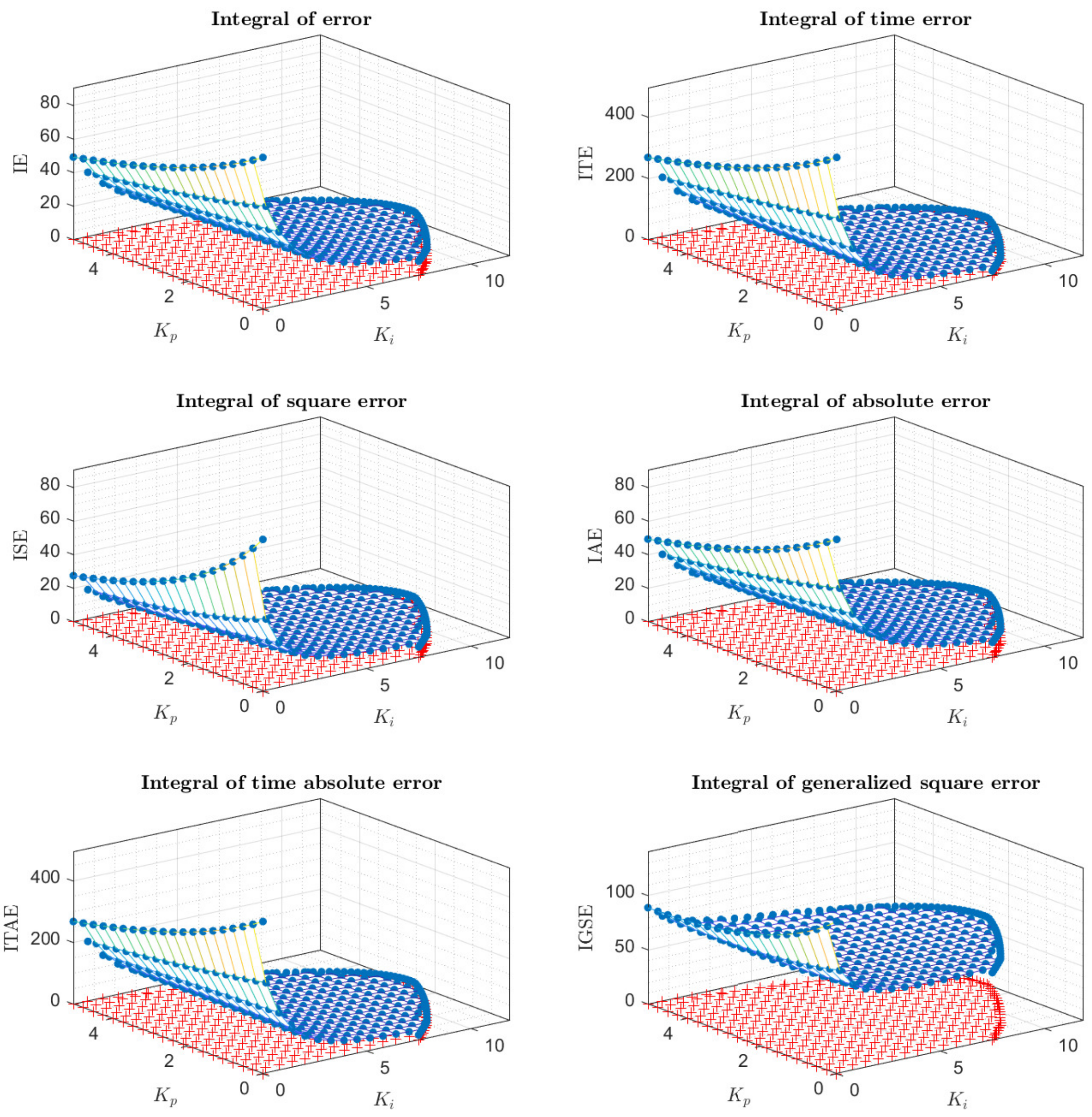


Figure 38: 3D surface of integral criteria for the intersection of robust regions. The GUI can visualize IE, ITE (on the top), ISE, IAE (in the middle), ITAE and IGSE (at the bottom) [67]

In authors previous work, the subsequent step in the region evaluation was the usage of the H_2 and H_∞ norms for each closed-loop system $H(s)$ [67]. Both norms can be used as design criteria for time domain optimization. The H_2 norm can be expressed as

$$||H(s)||_2 \triangleq \left(\frac{1}{2\pi} \int_{-\infty}^{+\infty} |H(j\omega)|^2 d\omega \right)^{\frac{1}{2}}. \tag{91}$$

The H_2 signal norm represents the energy it contains. Thus, it is equal to the ISE criterion. The H_∞ norm can be expressed as

$$\|H(s)\|_\infty \triangleq \sup_{\forall\omega} |H(j\omega)| \tag{92}$$

If the system is a stable single-input-single-output system then the H_∞ norm represents the peak gain, the largest value of the frequency response magnitude.

The calculations were performed according to the formulas (91) and (92) for all closed-loop systems from the intersection region. Even in this case, the resulting region was composed of approximately three hundred controllers. The values obtained for both norms are displayed in Figure 39. In Figure 39 on the left, the resulting region is evaluated using the H_2 norm. On the right, the resulting region is evaluated using the H_∞ norm. The results for the H_2 norm show that as the value of K_i and K_p increases, the total energy of the system also increases. In the case of the results for the H_∞ norm, it is clear that almost all controllers are evaluated at 1. This means that all obtained controllers stabilize the system and reach the required value. In this manner, it converges to the value of the set point. Only the values for $K_i = 0$ do not achieve the value of 1. These controllers do not attain the required value when it is caused by the absence of the integration component in the controller. Consequently, only the proportional component is present there, which is not enough to reach the set point. In this case, the considered PI controllers became P controllers. This fact stems mainly from the shape of the resulting region [67].

Since the H_∞ norm shows the maximum gain, it could be used for overshoot minimization. In the case of an oscillatory process $P(s)$, the 3D surface of H_∞ plot could have various shapes, and we would be able to select a controller leading to the minimal overshoot.

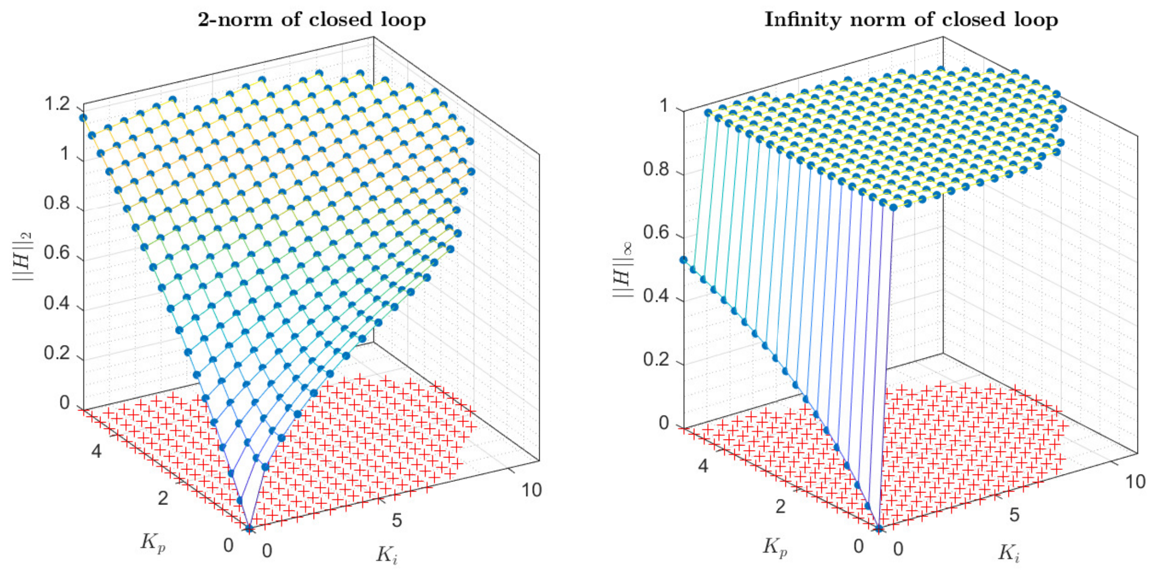


Figure 39: 3D visualisation of $||H||_2$ and $||H||_\infty$ of the closed-loop obtained from the parameters of the final robust region and defined nominal process. On the left there is 2-norm of the closed-loop. On the right there is ∞-norm of the closed-loop [67]

4.4.1 Integral Criteria Standard Optimization

The first minimization method is simple. It starts with computing closed-loop transfer functions for all controllers from the intersection region. Then it computes the integral criteria for all closed-loops from the intersection and finds their minimal value using Matlab command `min`.

The resulting $[K_i, K_p]$ coordinates and values of integral criteria can be seen in Table 6. From the results, we can see that the $[K_i, K_p]$ coordinates are similar in all cases, except for ISE and IGSE criteria, where the values for K_i are lower, and for K_p higher. The final minimal value of ISE is the lowest when compared with other criteria. The highest minimal value belongs to the IAE criterion. The GUI can visualize the step responses of optimal closed-loops for each integral criterion. Optimal closed-loop performances can be seen in Figure 40. From this plot, we can see that the step response peak for ISE and IGSE is smaller than for other criteria.

Figure 41 shows the minimal values of integral criteria in the 3D graphs generated for each criterion. 2D visualization can be seen in Figure 42. In all graphs, the minimum value is represented by a green dot. The region consists of 353 samples.

Figure 43 shows implemented standard optimization algorithm in the GUI.

Integral criteria	K_i value	K_p value	Integral criteria minimal values
IE	$K_i = 11.8485$	$K_p = 2.6582$	IE = 5.0856
ITE	$K_i = 11.7818$	$K_p = 2.4776$	ITE = 3.4821
ITAE	$K_i = 11.8445$	$K_p = 2.5743$	ITAE = 3.6358
IAE	$K_i = 11.7761$	$K_p = 2.9057$	IAE = 5.2245
ISE	$K_i = 10.9092$	$K_p = 3.5302$	ISE = 3.2147
IGSE	$K_i = 11.0747$	$K_p = 3.4553$	IGSE = 3.3550

Table 6: Integral criteria standard optimization results

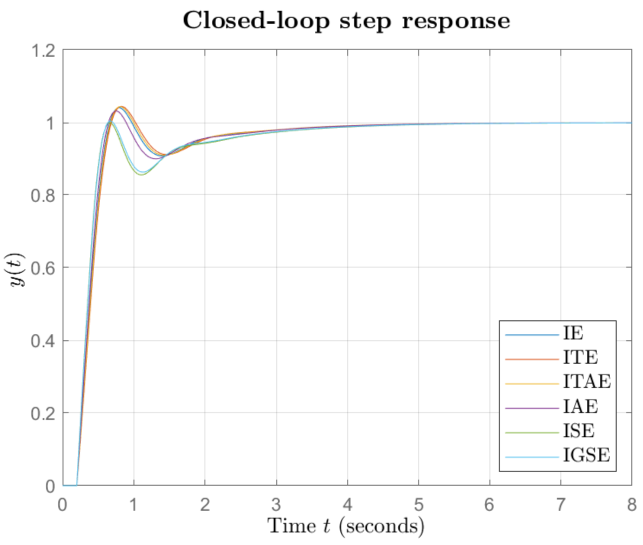


Figure 40: Standard optimization: Step responses of closed-loops containing optimal PI controllers in the sense of minimal corresponding integral criterion

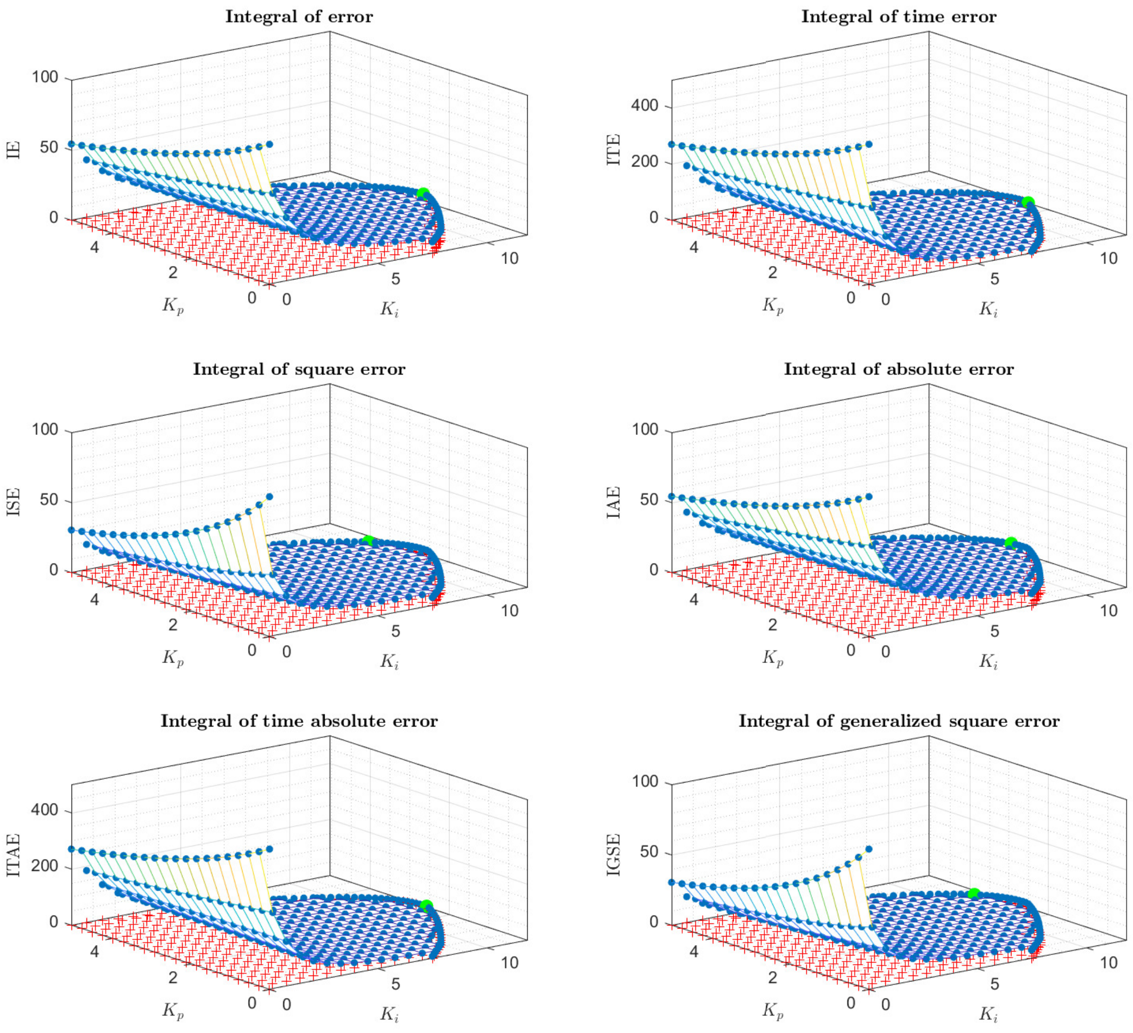


Figure 41: Standard optimization: Minimal values of integral criteria of optimality shown in 3D surfaces for 353 grid samples

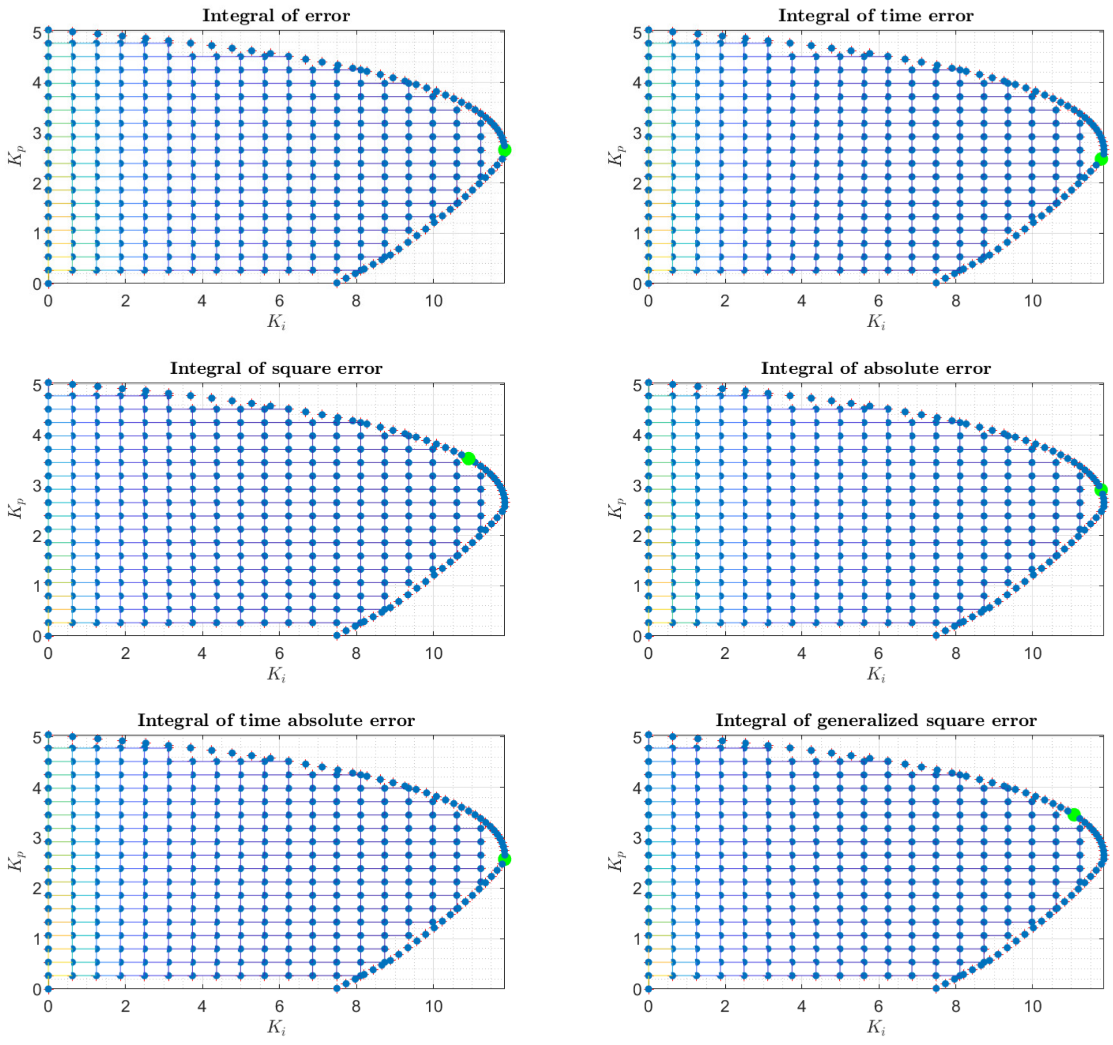
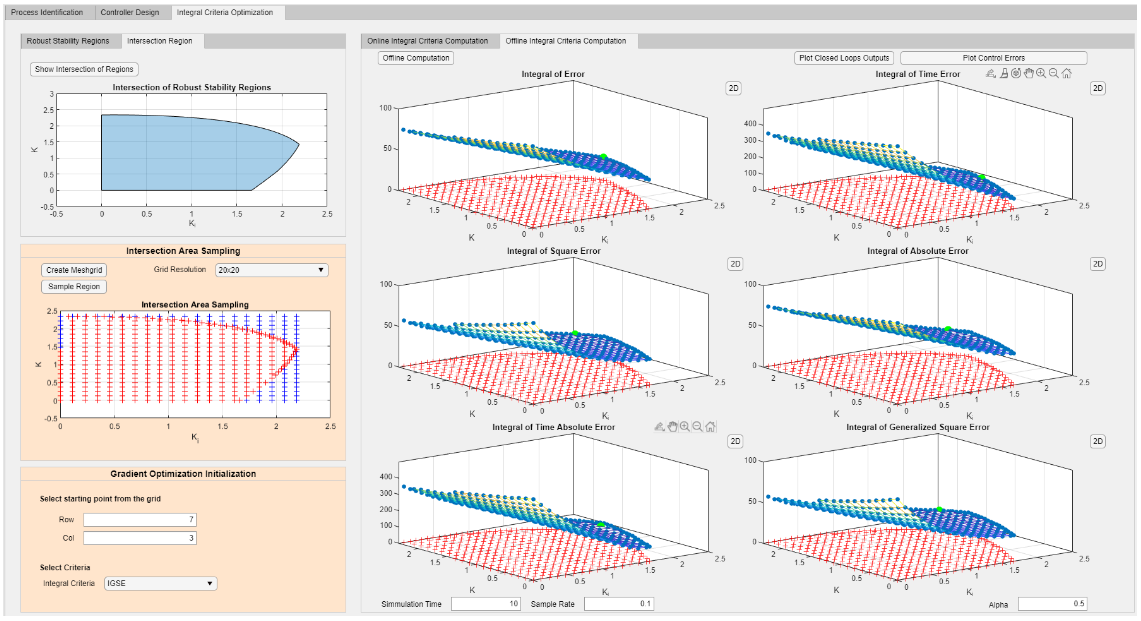


Figure 42: Standard optimization: Minimal values of integral criteria of optimality shown in 2D for 353 grid samples


Process Identification
Controller Design
Integral Criteria Optimization
Robust Stability Regions
Intersection Region
Show Intersection of Regions
Intersection of Robust Stability Regions
Intersection Area Sampling
Create Meshgrid
Sample Region
Gradient Optimization Initialization
Online Integral Criteria Computation
Offline Integral Criteria Computation
Offline Computation
Plot Closed Loops Outputs
Plot Control Errors
Integral of Error
Integral of Time Error
Integral of Square Error
Integral of Absolute Error
Integral of Time Absolute Error
Integral of Generalized Square Error
2D

Grid Resolution 20x20

Select starting point from the grid

Row 7

Col 3

Select Criteria

Integral Criteria IGSE

Simmulation Time 10

Sample Rate 0.1

Alpha 0.5

Figure 43: Integral criteria standard optimization in the GUI

The biggest advantage of this method is that it provides a global minimum of the integral criterion. The resulting $[K_i, K_p]$ parameters as well as the minimal value of chosen criterion will be the same after every simulation. The main disadvantage is the computational complexity and therefore the long duration of the algorithm, hence the integral criteria have to be computed for all closed-loops from the intersection area. With increasing mesh grid sampling the computational duration significantly increases as well.

4.4.2 Integral Criteria Gradient Optimization

The second implemented minimization method uses a gradient approach. The gradient optimization method does not compute integral criteria for all closed-loops from the sampled intersection region. This method is initiated in one $[K_i, K_p]$ point from the region. For this point, the closed-loop and selected integral criterion are computed. Next, the algorithm selects the nearest neighboring $[K_i, K_p]$ points and uses these controller parameters for the computation of corresponding closed-loops and integral criteria. The algorithm selects the $[K_i, K_p]$ point with the lowest value of the chosen integral criterion, and repeats the previous step. In this manner, the algorithm searches for minimal criterion value according to the maximal negative gradient. In the GUI, it is possible to minimize previously mentioned criteria IE, ITE, ITAE, ISE, IAE, and IGSE. It is also possible to select starting point from the sampled grid. The choice of the starting point influences the results of this method.

As an illustrative example, we have run the gradient optimization algorithm three times, each time with a different starting point. The mesh grid contained 20x20 samples, and the starting points were

$$[X_1, Y_1] = [3, 7],\ [X_2, Y_2] = [1, 1],\ [X_3, Y_3] = [1, 20], \tag{93}$$

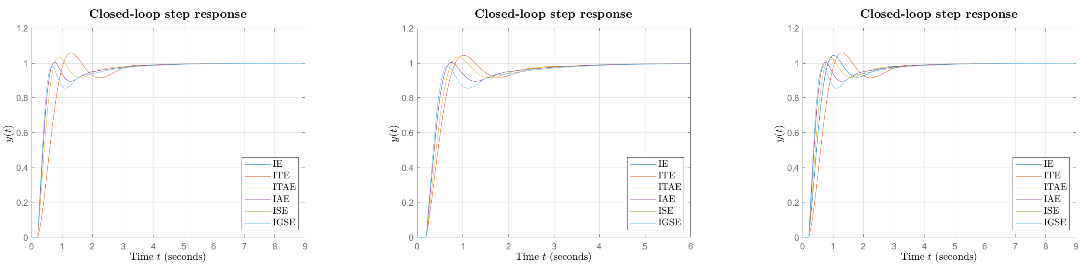
where X_i, and Y_i represent the column, and the row of the mesh grid respectively. Note: Starting point in origin $[X_0, Y_0] = [0, 0]$ would be at the bottom left of the intersection region. The resulting $[K_i, K_p]$ coordinates and values of integral criteria for each simulation can be seen in Table 7. The results show that for IE and ITE, the $[K_i, K_p]$ coordinates and integral criteria minimal values are not always the same for each starting point. Since this method does not have information about all closed-loops from the intersection area, these results were expected. For other integral criteria, the results are consistent for each starting point. The final minimal value of ISE is the lowest when compared with other criteria. The highest minimal value belongs to the IE criterion in the case when the algorithm starts in the point $[X_3, Y_3] = [1, 20]$. Again, the GUI can visualize the step responses of optimal closed-loops for each integral criterion. Optimal closed-loop performances for each starting point can be seen in Figure 44. All three graphs look very similar. The only difference is in the shape of closed-loop response minimizing IE and ITE criteria.

Figures 45, 47, and 49 show the process of gradient optimization in the 3D graphs leading to a local minimum for each criterion for starting points $[X_1, Y_1]$, $[X_2, Y_2]$, and $[X_3, Y_3]$. Figures 46, 48, and 50 provide 2D visualizations of the gradient method. In all graphs, the starting point is represented by a pink dot, the path to the local minimum is marked by green dots, and blue dots represent the nearest neighbors which had been selected during the iterations of gradient optimization.

Figure 51 shows implemented gradient optimization algorithm in the GUI.

Integral criteria	K_i value	K_p value	Integral criteria minimal values
IE_1	$K_i = 11.2252$	$K_p = 2.1234$	$\text{IE}_1 = 5.36251$
IE_2	$K_i = 11.2252$	$K_p = 2.1234$	$\text{IE}_2 = 5.3625$
IE_3	$K_i = 10.6016$	$K_p = 1.5925$	$\text{IE}_3 = 5.6726$
ITE_1	$K_i = 8.7307$	$K_p = 0.5308$	$\text{ITE}_1 = 4.8886$
ITE_2	$K_i = 10.6016$	$K_p = 1.5925$	$\text{ITE}_2 = 3.8491$
ITE_3	$K_i = 8.7307$	$K_p = 0.5308$	$\text{ITE}_3 = 4.8886$
ITAE_1	$K_i = 11.2252$	$K_p = 2.1234$	$\text{ITAE}_1 = 3.8212$
ITAE_2	$K_i = 11.2252$	$K_p = 2.1234$	$\text{ITAE}_2 = 3.8212$
ITAE_3	$K_i = 11.2252$	$K_p = 2.1234$	$\text{ITAE}_3 = 3.8212$
IAE_1	$K_i = 11.2252$	$K_p = 2.9196$	$\text{IAE}_1 = 5.3680$
IAE_2	$K_i = 11.2252$	$K_p = 2.9196$	$\text{IAE}_2 = 5.3680$
IAE_3	$K_i = 11.2252$	$K_p = 2.9196$	$\text{IAE}_3 = 5.3680$
ISE_1	$K_i = 10.6016$	$K_p = 3.4505$	$\text{ISE}_1 = 3.2514$
ISE_2	$K_i = 10.6016$	$K_p = 3.4505$	$\text{ISE}_2 = 3.2514$
ISE_3	$K_i = 10.6016$	$K_p = 3.4505$	$\text{ISE}_3 = 3.2514$
IGSE	$K_i = 10.6016$	$K_p = 3.4505$	IGSE = 3.38593
IGSE	$K_i = 10.6016$	$K_p = 3.4505$	IGSE = 3.38593
IGSE	$K_i = 10.6016$	$K_p = 3.4505$	IGSE = 3.38593

Table 7: Integral criteria gradient optimization results for all three starting points



(a) Starting point $[X_1, Y_1]$ (b) Starting point $[X_2, Y_2]$ (c) Starting point $[X_3, Y_3]$

Figure 44: Gradient optimization: Step responses of closed-loops containing optimal PI controllers in the sense of minimal corresponding integral criterion

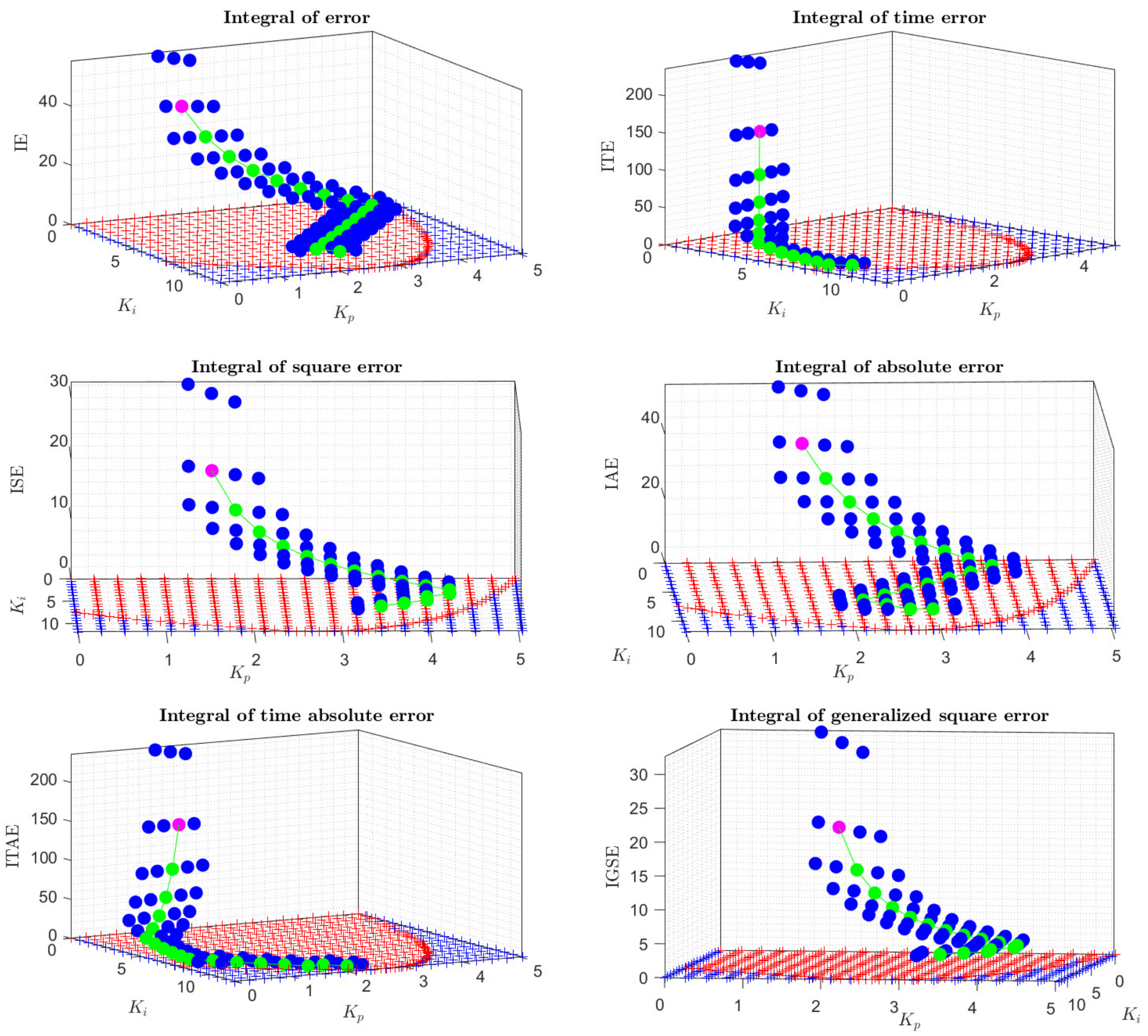


Figure 45: Gradient optimization: Minimal values of integral criteria of optimality shown in 3D surfaces for 353 grid samples

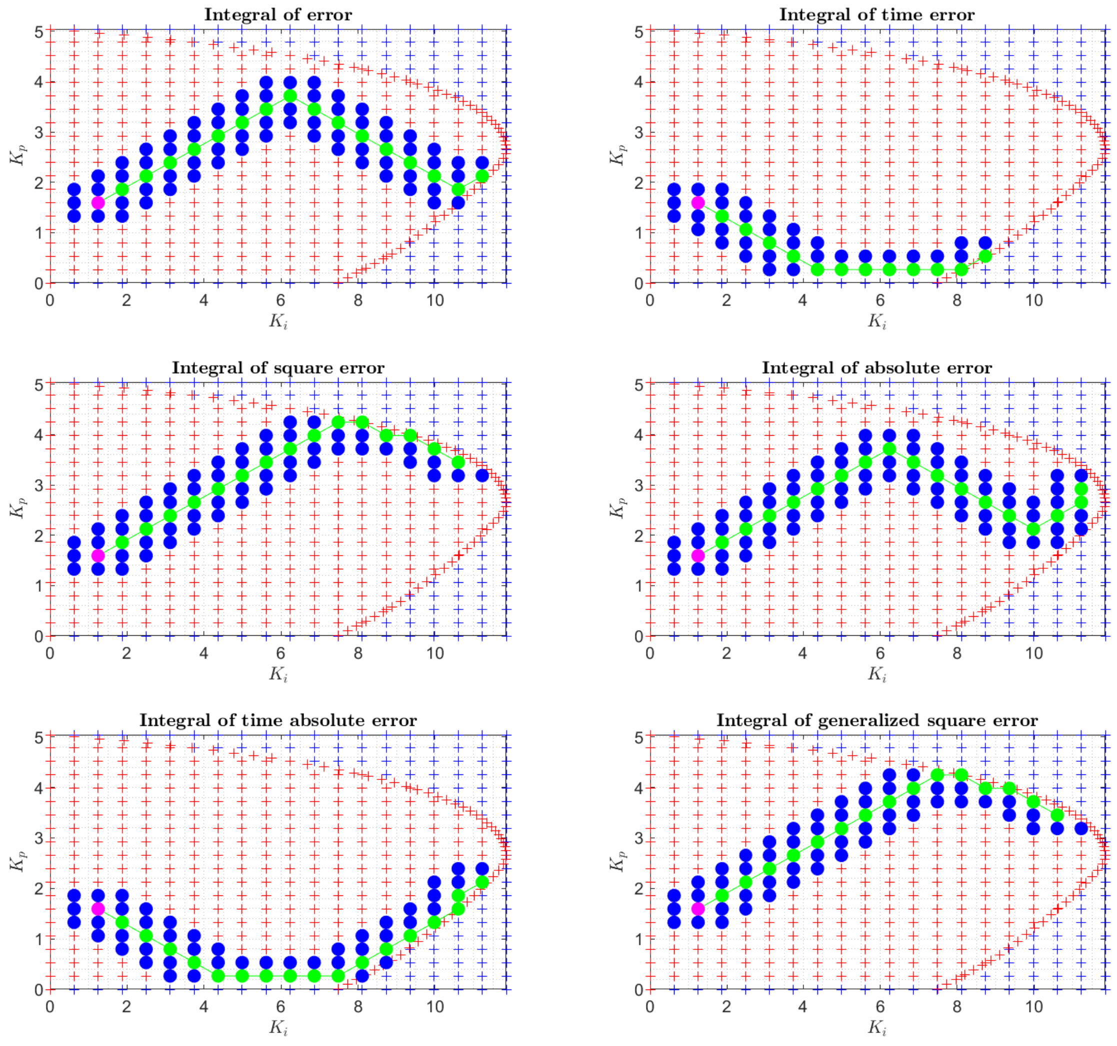


Figure 46: Gradient optimization: Minimal values of integral criteria of optimality shown in 2D for 353 grid samples

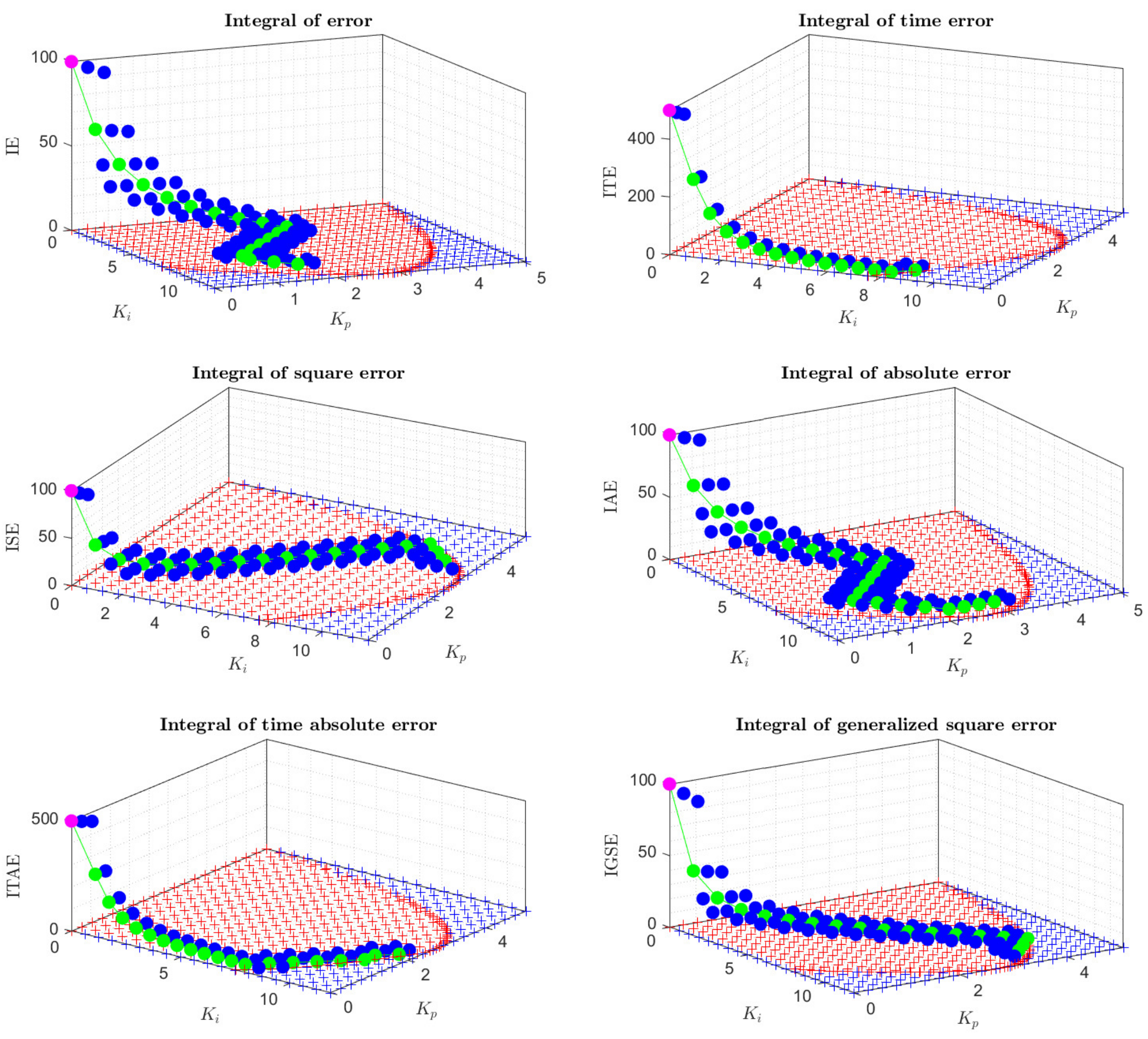


Figure 47: Gradient optimization: Minimal values of integral criteria of optimality shown in 3D surfaces for 353 grid samples

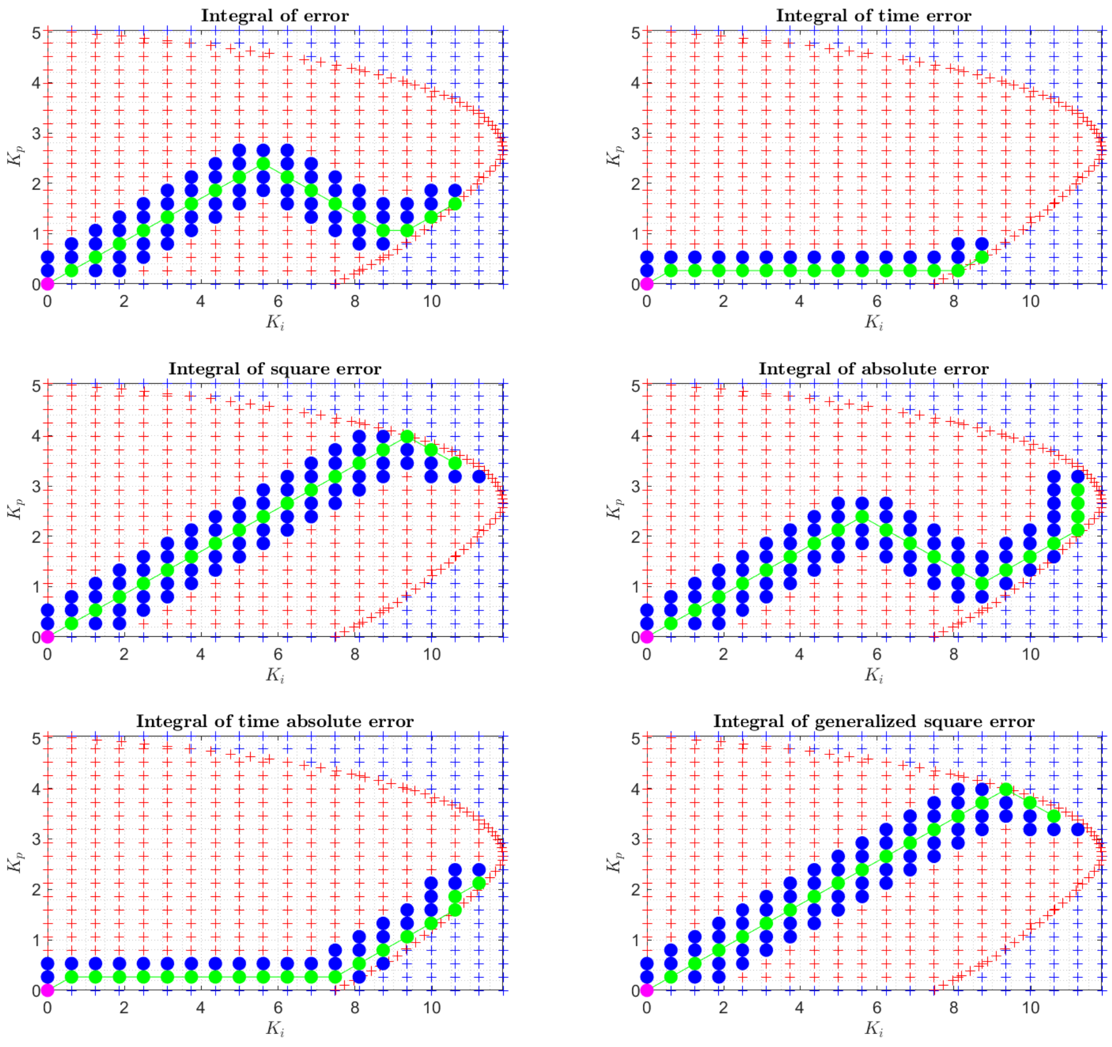


Figure 48: Gradient optimization: Minimal values of integral criteria of optimality shown in 2D for 353 grid samples

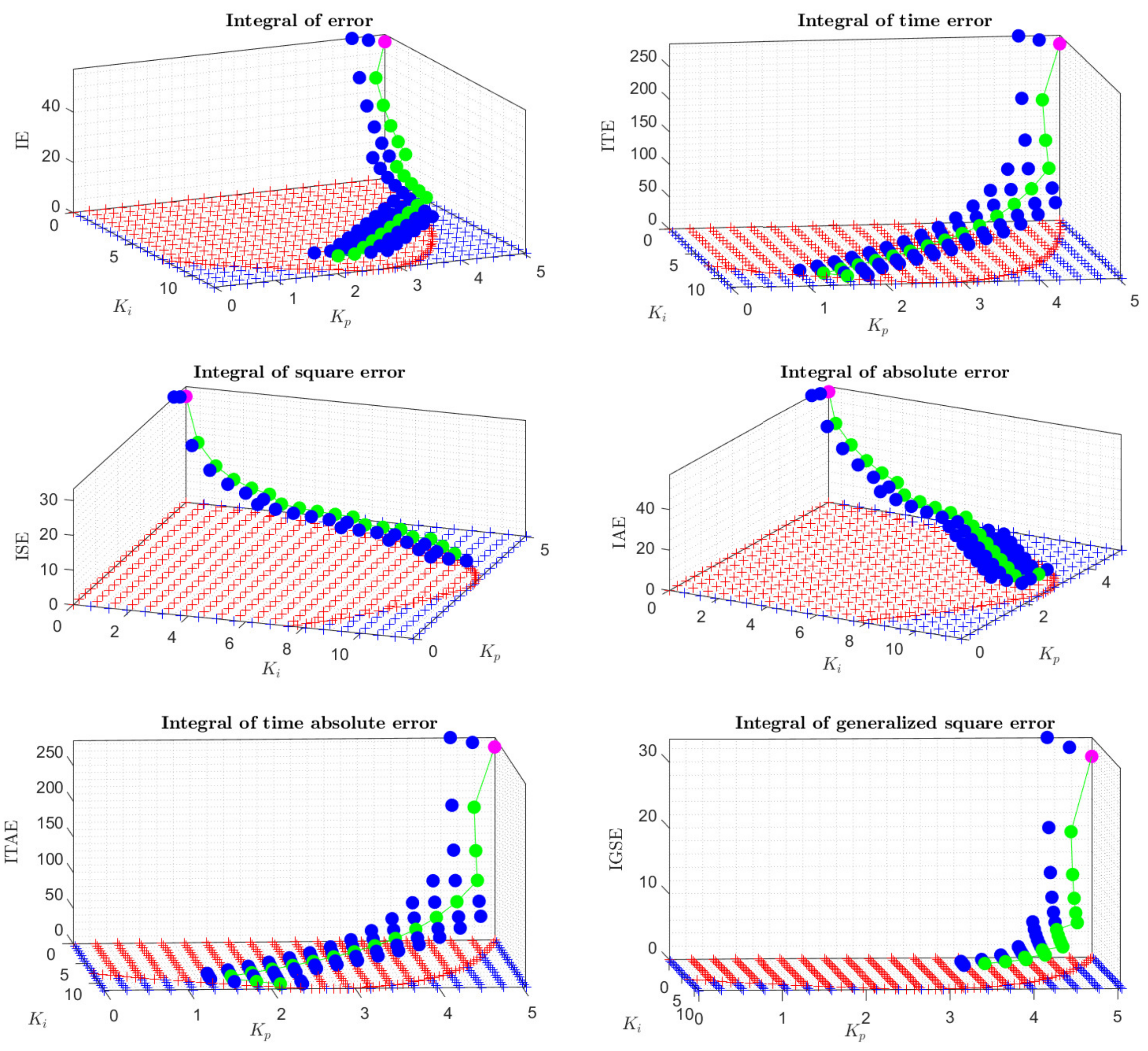


Figure 49: Gradient optimization: Minimal values of integral criteria of optimality shown in 3D surfaces for 353 grid samples

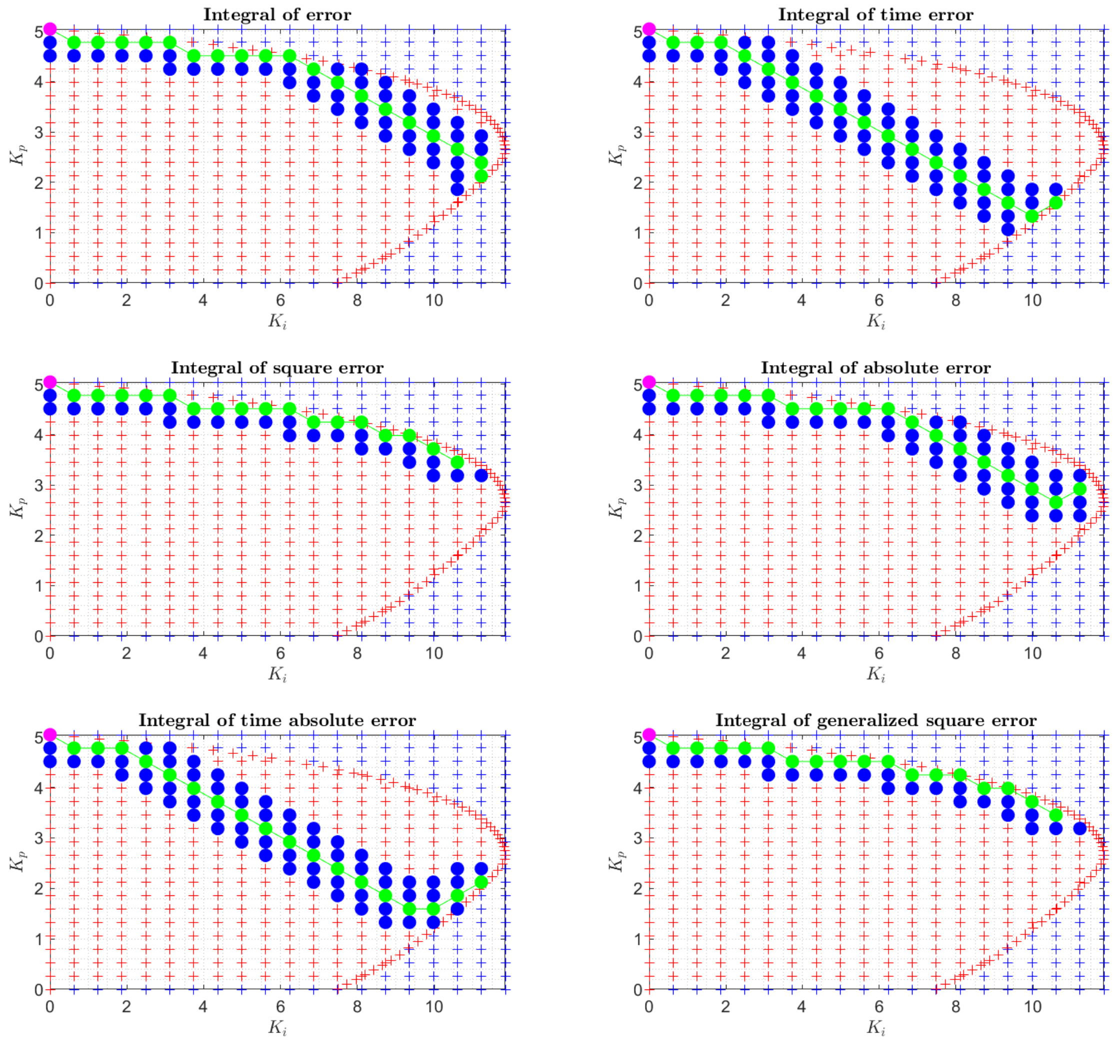
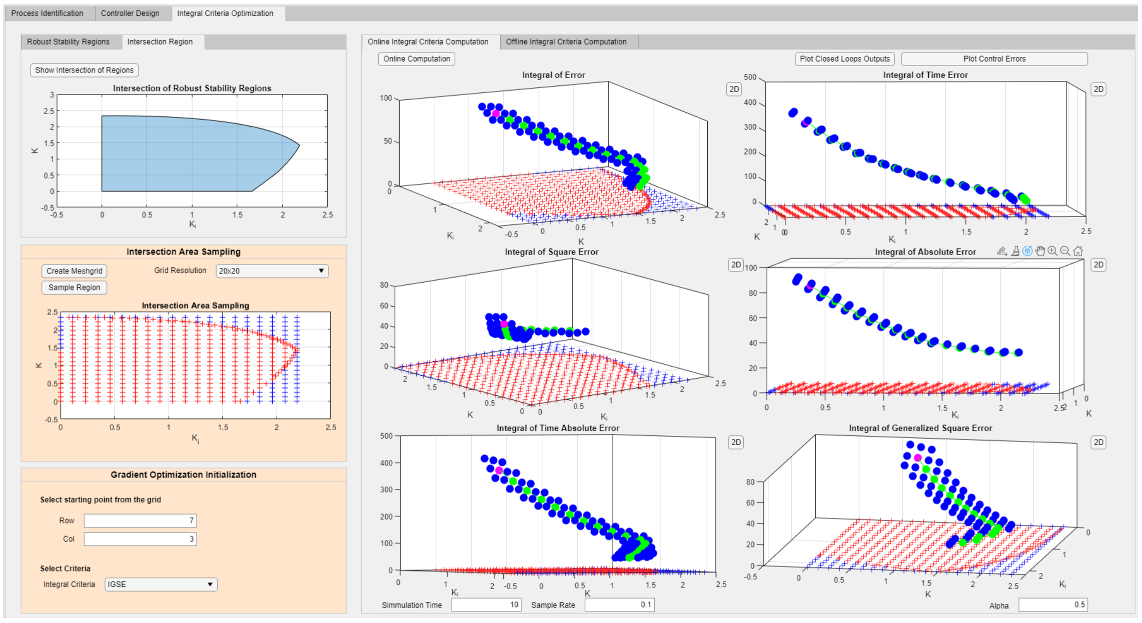


Figure 50: Gradient optimization: Minimal values of integral criteria of optimality shown in 2D for 353 grid samples


Process Identification
Controller Design
Integral Criteria Optimization
Robust Stability Regions
Intersection Region
Show Intersection of Regions
Intersection of Robust Stability Regions
Intersection Area Sampling
Create Meshgrid
Sample Region
Gradient Optimization Initialization
Online Integral Criteria Computation
Offline Integral Criteria Computation
Online Computation
Plot Closed Loops Outputs
Plot Control Errors
Integral of Error
Integral of Time Error
Integral of Square Error
Integral of Absolute Error
Integral of Time Absolute Error
Integral of Generalized Square Error
2D

Grid Resolution 20x20

Select starting point from the grid

Row 7

Col 3

Select Criteria

Integral Criteria IGSE

Simmulation Time 10

Sample Rate 0.1

Alpha 0.5

Figure 51: Integral criteria gradient optimization in the GUI

From these results, we can see that the starting point plays a significant role. It can influence the shape and distance of the path to the minimum. The closer it is to the supposed minimum, the shorter the path and the computational time. Perhaps the biggest disadvantage of this method is that it leads to a local minimum. Meaning, the resulting $[K_i, K_p]$ parameters and the minimal value of chosen criterion will not be the same after every simulation. Moreover, due to the irregular shape of the sampled borderline, it could not be included. Hence, the algorithm operates with the regular rectangular grid. Since, often the optimal $[K_i, K_p]$ parameters lie on the border of the region, this creates slight discrepancies between this method and the standard optimization method mentioned earlier.

However, the biggest advantage of the gradient optimization method is its small computational complexity and thus the algorithm duration. The computational duration is significantly smaller when compared to the standard minimization, as it will be shown in Section 4.4.3.

4.4.3 Comparison of Standard and Gradient Optimization Methods

In this section, the comparison of the two mentioned methods will be shown. In the following experiment, the optimization is executed for both gradient and standard methods. For an illustrative example, the optimization algorithms were performed for the IE criterion. Starting point $[X, Y] = [3, 7]$ was selected for the gradient method. This experiment consisted of three phases. In each phase we have increased the mesh grid sampling, starting at 20x20 samples, followed by 40x40 samples, and finishing with 80x80 samples. In each phase, we have run 5 simulations for standard and gradient optimization. We have mostly focused on the differences in computational time and the final minimum value of the IE criterion. For a better comparison of both methods, we have excluded the irregular borderline from the standard optimization method because the gradient method can not operate with it.

Note: All calculations and simulations were performed on a computer with CPU I7 - 8700k 3.70GHz, 16GB RAM, 2TB HDD and Windows 10 64bit.

In the first phase, the simulation was performed on the 20x20 mesh grid samples. The elapsed time of each simulation for both methods can be seen in Table 8. From this table, we can see that the computational time of standard optimization has approximately double the value when compared to the duration of gradient optimization.

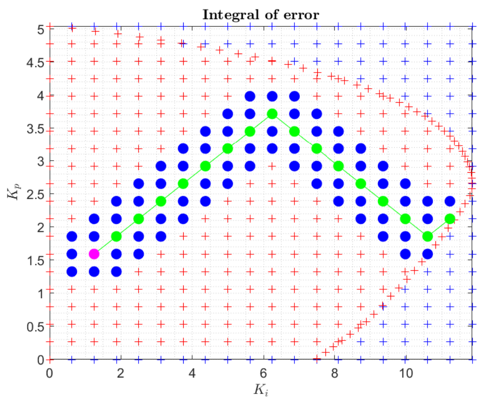
Optimization method	Elapsed time of each simulation				
Gradient method	4.241939 s	4.045740 s	3.882915 s	4.041694 s	4.128104 s
Standard method	8.479173 s	8.252737 s	8.402302 s	8.200030 s	8.420921 s

Table 8: Results of the first phase, mesh grid sampling: 20x20

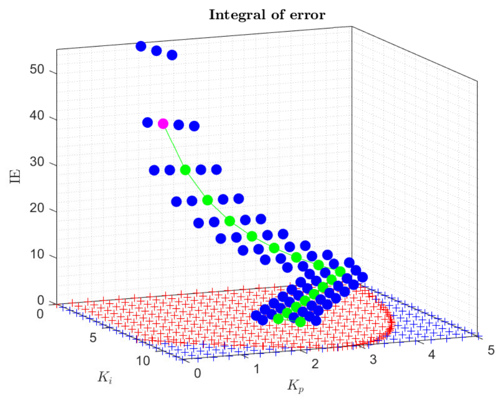
The IE minimum values for gradient optimization method $\mathrm{IE}_{\mathrm{G}}^{\{\min\}}$, and standard optimization method $\mathrm{IE}_{\mathrm{S}}^{\{\min\}}$ were

$$\mathrm{IE}_{\mathrm{G}}^{\{\min\}} = 5.3625,\ \mathrm{IE}_{\mathrm{S}}^{\{\min\}} = 5.3625. \tag{94}$$

We can see that the final values of the IE criterion are the same for both methods. Even despite the gradient method providing only a local minimum. Visual results of these methods can be seen in Figure 52, and 53 respectively.

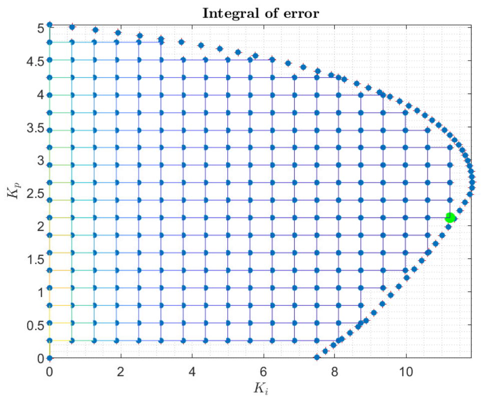


(a) 2D view

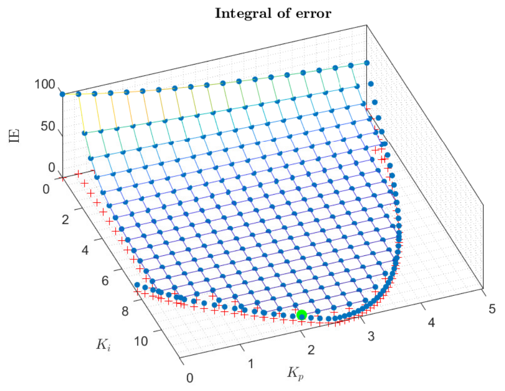


(b) 3D view

Figure 52: IE gradient minimization, grid dimension 20x20



(a) 2D view



(b) 3D view

Figure 53: IE standard minimization (border excluded), grid dimension 20x20

In the second phase, the simulation was executed for the 40x40 mesh grid sampling. The elapsed time of each simulation for both methods can be seen in Table 9. This table shows that the computational time value of standard optimization is almost four times higher than the computational time value of gradient optimization.

Optimization method	Elapsed time of each simulation				
Gradient method	7.003348 s	6.961408 s	6.913999 s	6.759694 s	6.840178 s
Standard method	28.552841 s	28.174117 s	26.324839 s	27.072495 s	27.188183 s

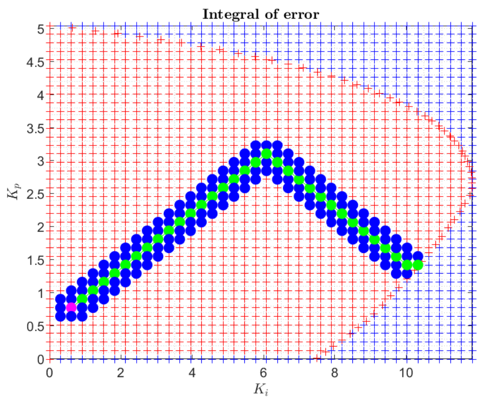
Table 9: Results of the second phase, mesh grid sampling: 40x40

The IE minimum values for gradient optimization method $\mathrm{IE}_{\mathrm{G}}^{\{\min\}}$, and standard optimization method $\mathrm{IE}_{\mathrm{S}}^{\{\min\}}$ were

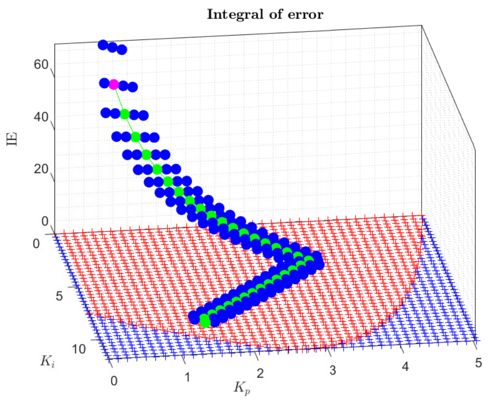
$$\mathrm{IE}_{\mathrm{G}}^{\{\min\}} = 5.8204,\ \mathrm{IE}_{\mathrm{S}}^{\{\min\}} = 5.2161. \tag{95}$$

We can see that the final value of the IE criterion is different for each method. This is caused because the gradient method provides only a local minimum. When compared to the previous phase results (94), we can see that the $\mathrm{IE}_{\mathrm{G}}^{\{\min\}}$ value increased, and the $\mathrm{IE}_{\mathrm{S}}^{\{\min\}}$ value decreased. These results indicate that the higher number of samples does not have to improve the gradient method performance.

Visual results of these methods can be seen in Figure 54, and 55 respectively.

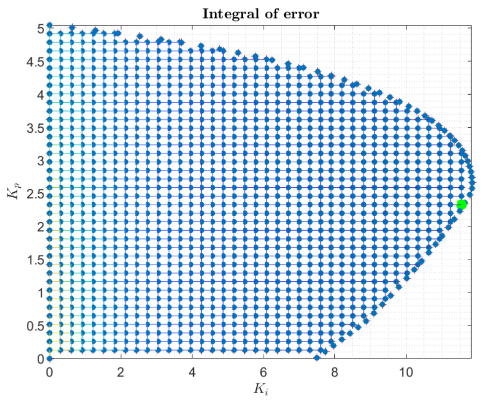


(a) 2D view

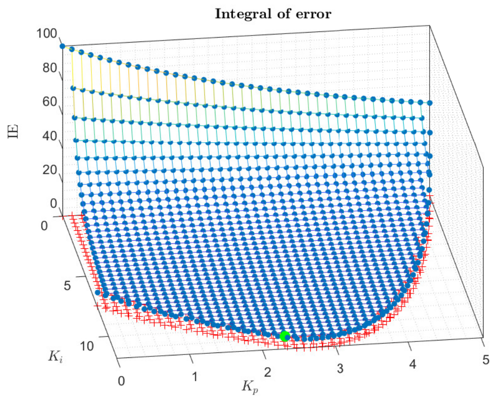


(b) 3D view

Figure 54: IE gradient minimization, grid dimension 40x40



(a) 2D view



(b) 3D view

Figure 55: IE standard minimization (border excluded), grid dimension 40x40

Finally, in the third phase, the simulation was run for the 80x80 mesh grid dimension. The elapsed time of each simulation for both methods can be seen in Table 10. According to the table, the computational time value of standard optimization is almost ten times higher than the computational time value of gradient optimization.

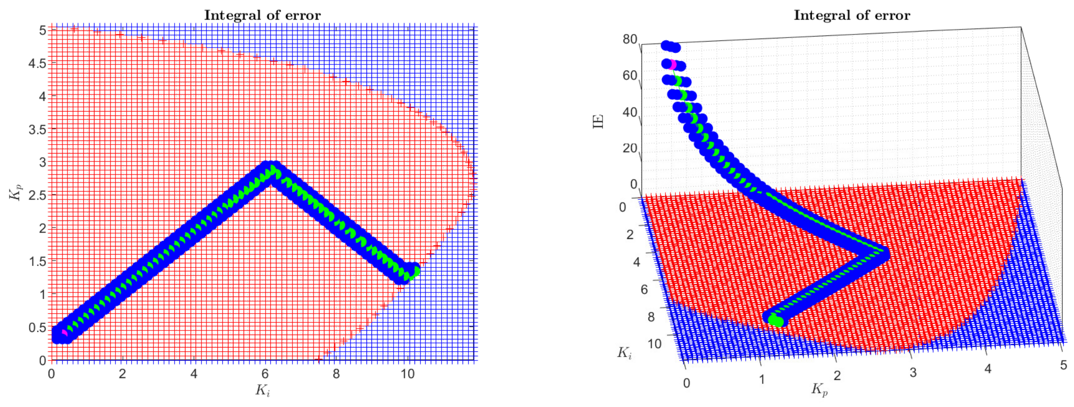
Optimization method	Elapsed time of each simulation				
Gradient method	13.092232 s	12.715285 s	13.024894 s	13.217139 s	13.194521 s
Standard method	103.117455 s	104.984474 s	101.238183 s	103.115569 s	27.188183 s

Table 10: Results of the third phase, mesh grid sampling: 80x80

The IE minimum values for gradient optimization method $\mathrm{IE}_{\mathrm{G}}^{\{\min\}}$, and standard optimization method $\mathrm{IE}_{\mathrm{S}}^{\{\min\}}$ were

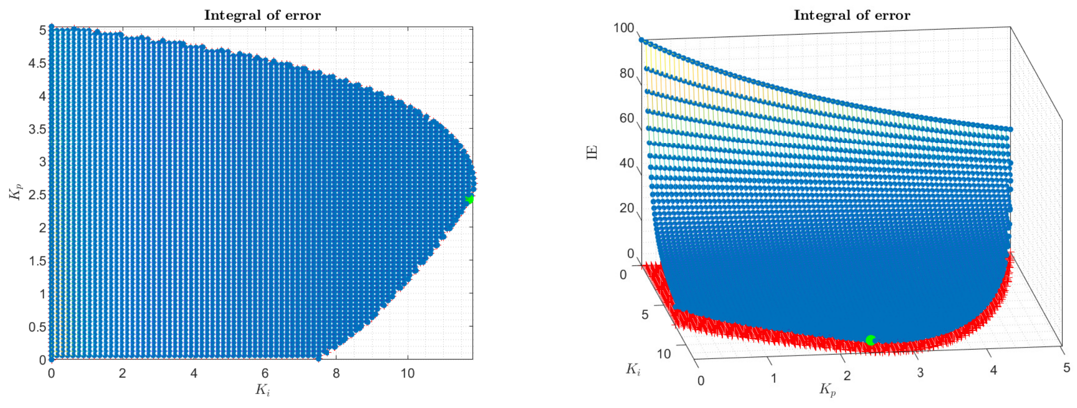
$$\mathrm{IE}_{\mathrm{G}}^{\{\min\}} = 5.8941,\ \mathrm{IE}_{\mathrm{S}}^{\{\min\}} = 5.1486. \tag{96}$$

These results show that the final value of the IE criterion is different for each method. Again, this difference is caused because the gradient method provides only a local minimum. We can see that the $\mathrm{IE}_{\mathrm{G}}^{\{\min\}}$ value again increased, and the $\mathrm{IE}_{\mathrm{S}}^{\{\min\}}$ value again decreased when compared to the previous phase results (94). Visual results of these methods can be seen in Figure 56, and 57 respectively.



(a) 2D view (b) 3D view

Figure 56: IE gradient minimization, grid dimension 80x80



(a) 2D view (b) 3D view

Figure 57: IE standard minimization (border excluded), grid dimension 80x80

To conclude this comparison, both methods have some advantages and disadvantages. The standard method provides a global minimum. However, it has to compute the integral criterion for each closed loop of the intersection area. The gradient method provides a sub-optimal solution in the form of a local minimum. On the other hand, it does not have to compute the integral criterion for all closed loops from the intersection. This creates a significant difference in the computational time between the two methods. However, the more grid samples are selected, the harder it gets for the gradient method to find (or at least get close to) the global minimum. Moreover, the borderline for the standard minimization method was excluded for the results to be better comparable. If it was included, the final minimal value of IE would be lower for the standard method.

5 Validation on Real Process

For the validation of this thesis, we have chosen the incubation device (Figure 58) where we have measured the air temperature. The air is heated by the metal heating unit which serves as the actuator. A detailed description of the device can be seen in [37]. The incubation device is connected to the Monarco HAT and Raspberry Pi HW. The communication with the local PC is provided by REXYGEN SW [46].

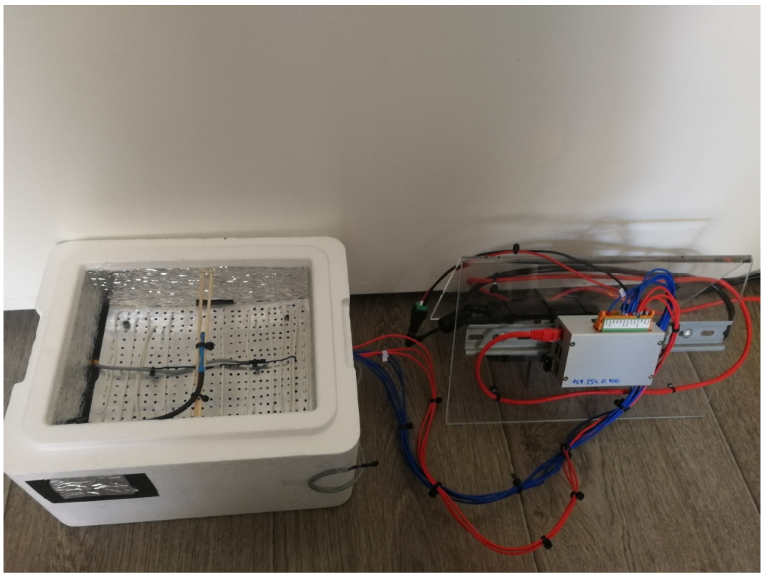


Figure 58: Open incubator with the Monarco HAT and Raspberry Pi [37]

The first part of the validation was the experimental identification of the process. As a process variable, we have selected the air temperature. In order to perform the identification, we had to measure the step response of the air temperature.

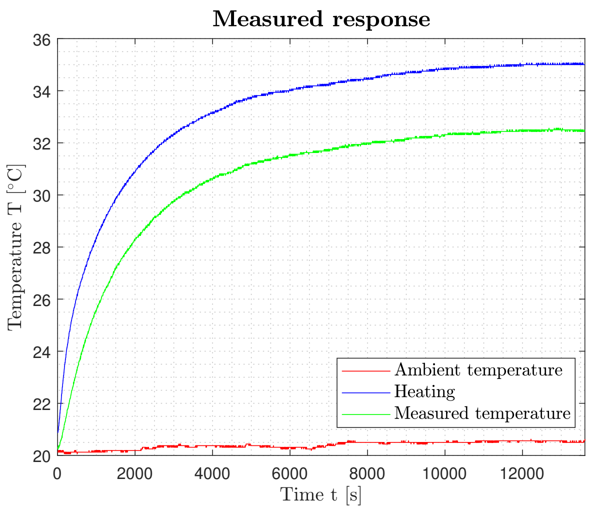


Figure 59: Measured response of the air temperature in the incubation device

We have set the power of the heating unit to 30% which is equivalent to the air temperature $T_{air} = 32.5$ °C in the steady state. During the experiment, the average ambient temperature was $T_{amb} = 20.25$ °C. In Figure 59, we can see the measured response of the temperature in the incubator. The air temperature starts at 20.25 °C and finishes at 32.5 °C. The duration of this experiment is in the interval from 14:27:58 to 18:14:49 (13611 seconds).

The measured data $\boldsymbol{x}$ were later normalized according to the equation

$$\frac{(\boldsymbol{x} - \text{offset})}{\text{amplitude}} = \frac{(x - 20.25)}{(32.5 - 20.25)}. \quad (97)$$

The data from step response were used for Matlab and FOPMDT process estimation.

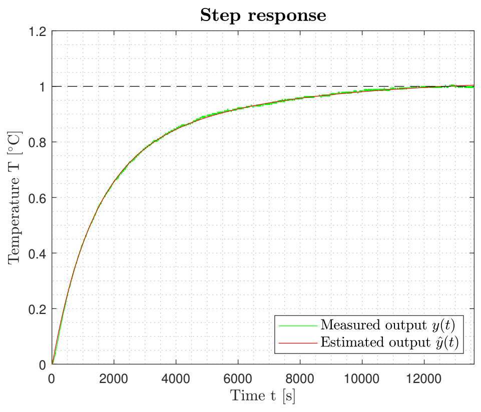
The second part of the validation was the robust controller design. For the controller design we have used design criteria $X_1 = -0.5 + 0.000j$ (GM $= 2$), $X_2 = -0.5 - 0.866j$ (PM $= 60°$), $X_3 = -0.5 - 0.375j$ ($M_S = 1.6$) which were mentioned in Table 5.

5.1 Validation of Estimated Matlab Process Model

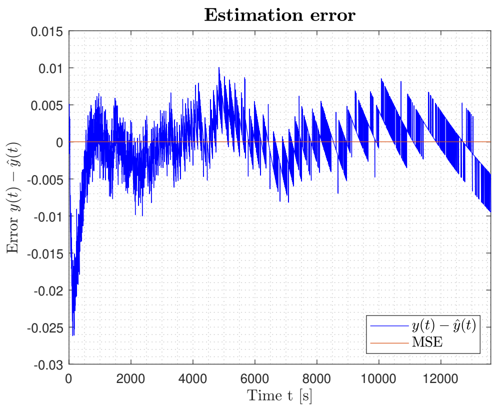
The transfer function estimated using Matlab commands was

$$\hat{P}(s) = \frac{(0.0001229s + 6.542e-09)}{s^2 + 0.0001991s + 6.372e-09} e^{-72s} \tag{98}$$

The comparison of measured and estimated process response using Matlab commands can be seen in Figure 60a. The estimation error is depicted in Figure 60b. From these pictures, we can assume that the identification was precise.



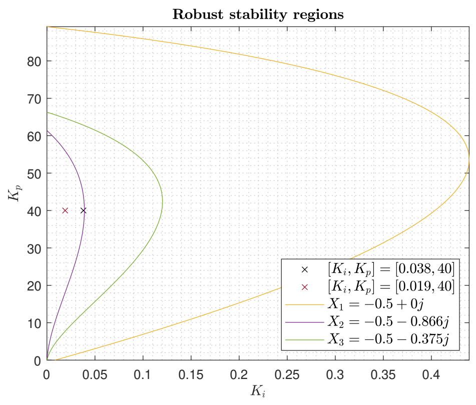
(a) Estimated step response



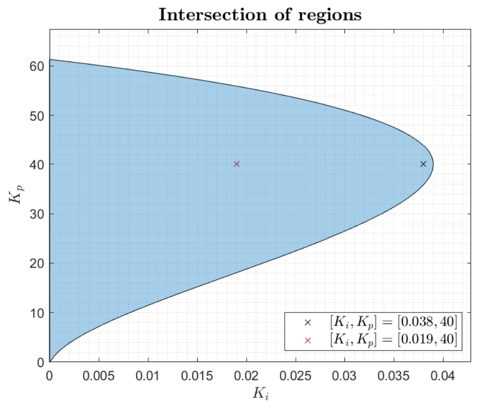
(b) Estimation error

Figure 60: Comparison of measured and estimated process response using Matlab commands, and the estimation error

The robust stability regions corresponding with the design criteria X_1, X_2, X_3 for the process model (98), and its intersection can be seen in Figure 61a, and 61b respectively. From the intersection area, we have selected the PI controller parameters $[K_i, K_p] = [0.038, 40]$ and $[K_i, K_p] = [0.019, 40]$.



(a) Robust stability regions

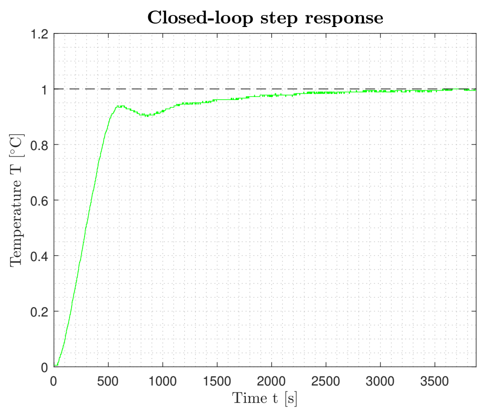


(b) Intersection of regions

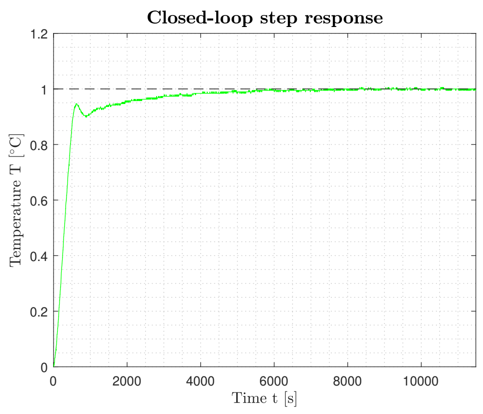
Figure 61: Robust stability regions and its intersection together with the selected PI controller parameters $[K_i, K_p] = [0.038, 40]$ and $[K_i, K_p] = [0.019, 40]$

Both PI controllers were used in the closed-loop system connected to the incubator through REXYGEN. The closed-loop step response was measured for both PI controllers. In Figure 62a, and 62b we can observe the step responses of closed-loops with PI controller parameters $[K_i, K_p] = [0.038, 40]$, and $[K_i, K_p] = [0.019, 40]$ respectively. From both graphs, we can see that the measured value of the air temperature reaches the set point. However, for the second closed loop, the set-point is reached after a significantly longer time (approx 8000 seconds) when compared to the first closed loop (approx. 3500 seconds). For the first closed loop, the rise time is approximately 4 times quicker than the original non-controlled rise time. For the second closed loop, the rise time is approximately 1.5 times quicker than the original non-controlled rise time.

In Figure 63a, and 63b we can observe the courses of manipulated variable in percents for the first, and second closed loop respectively.

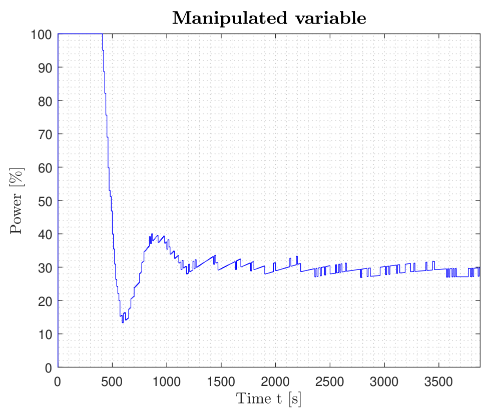


(a) $[K_i, K_p] = [0.038, 40]$

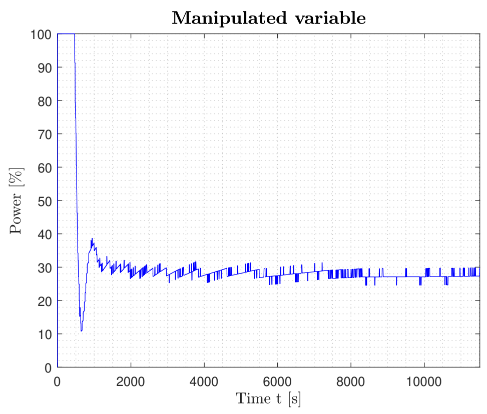


(b) $[K_i, K_p] = [0.019, 40]$

Figure 62: Closed-loop step responses containing PI controller parameters designed for X_1, X_2, X_3 design criteria for the Matlab model (98)



(a) $[K_i, K_p] = [0.038, 40]$



(b) $[K_i, K_p] = [0.019, 40]$

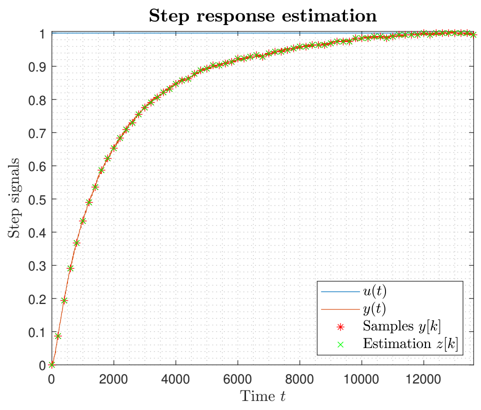
Figure 63: Manipulated values (in percents) for both step responses

5.2 Validation of Estimated FOPMDT Process Model

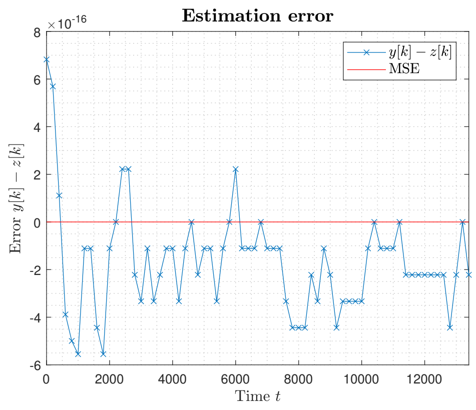
The estimation using the FOPMDT model was performed for the parameters: $N = 69$, $d = 30$, $n = 1000$, $\tau = 50$ according to the procedure described in 4.2.2. The estimated transfer function had a form

$$\hat{P}(s) = \frac{1}{50s + 1} \sum_{i=0}^{1000} A_i e^{-i30s}. \qquad (99)$$

Again, the measured and estimated process responses can be seen in Figure 64a. The estimation error is depicted in Figure 64b. The identification was precise, the estimation error has significantly lesser values than in the previous case.



(a) Estimated step response



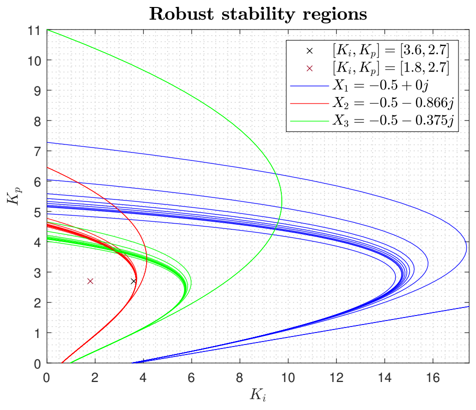
(b) Estimation error

Figure 64: Comparison of measured and estimated process response using FOPMDT, and the estimation error

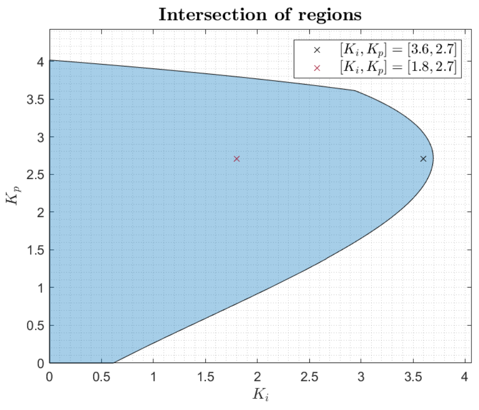
In this part of the validation, we have considered process uncertainty. We have computed the three characteristic numbers for the FOPMDT model according to the (76), (77), and (78). The characteristic numbers were $\{\kappa = 0.9887, \mu = 354.4626, \sigma^2 = 9.1671e+04\}$. The normalized characteristic numbers set had a form $\{\bar{\kappa} = 1, \bar{\mu} = 1, \bar{\sigma}^2 = 0.7296\}$. We have considered $n \to \infty$ and $m = 1$. Thus, we have obtained the normalized Model set $\mathcal{S}^{\infty,1}(\bar{\sigma}^2)$.

Next, we have computed the extremal processes creating the Value set boundaries $\partial\mathcal{V}_\omega$ according to the (45), (46), and (47). From the extremal processes, we have chosen 12 IO transfer functions $P^{IO}_{\{1,2,3\},i}(s,\alpha)$, $i = 1,\ldots,12$, for the robust controller design.

The robust stability regions corresponding with the design criteria X_1, X_2, X_3 for the obtained IO extremal process set $P^{IO}_{\{1,2,3\},i}(s,\alpha)$, and its intersection can be seen in Figure 65a, and 65b respectively. From the intersection area, we have selected the PI controller parameters $[K_i, K_p] = [3.6, 2.7]$ and $[K_i, K_p] = [1.8, 2.7]$.



(a) Robust stability regions



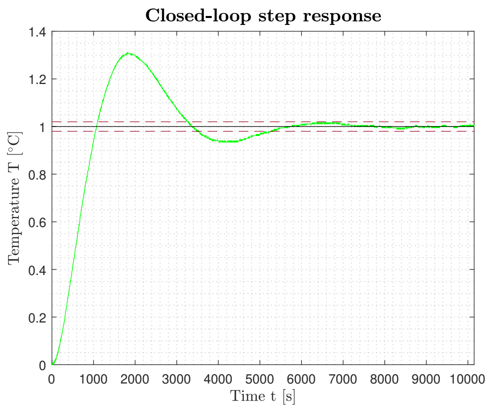
(b) Intersection of regions

Figure 65: Robust stability regions and its intersection together with the selected PI controller parameters $[K_i, K_p] = [3.6, 2.7]$ and $[K_i, K_p] = [1.8, 2.7]$

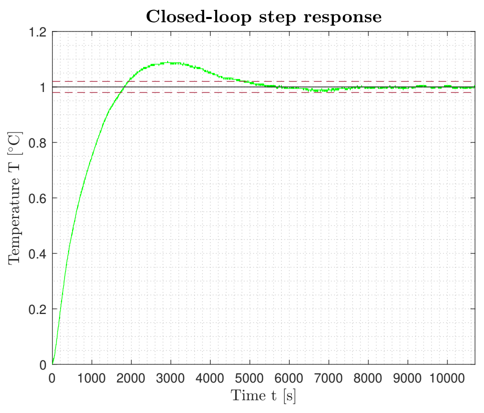
After the PI controller parameters were selected, it was important to denormalize them. The denormalized controller parameters were $[K_i, K_p] = [0.0038, 2.7309]$ and $[K_i, K_p] = [0.0019, 2.7309]$

The closed-loop step response was measured on the incubator through REXYGEN for both PI controllers. In Figure 66a, and 66b we can observe the step responses of closed-loops with PI controller parameters $[K_i, K_p] = [0.0038, 2.7309]$, and $[K_i, K_p] = [0.0019, 2.7309]$ respectively. From the first graph, we can see that the measured output has approx. 30% overshoot. After approx. 5250 seconds, the measured output stays in the 2% tolerance band. The rise time is around 1000 seconds. In the second graph, we can see that the overshoot is smaller than in the previous case reaching approx. 9% of the set-point value. After approx. 4885 seconds, the measured output stays in the 2% tolerance band. The rise time takes around 1800 seconds. We can observe that the $K_i = 0.0038$ leads to a more oscillatory course of the step response, higher overshoot, and shorter rise time.

In Figure 67a, and 67b we can observe the course of manipulated variable in percents for the first, and second closed loop respectively. The first graph shows, that the manipulated variable for $K_i = 0.0038$ does not go above 75% of the power. Higher values of the manipulated variable lead to a higher overshoot in the closed-loop step response. In the second graph, the manipulated variable for $K_i = 0.0019$ does not exceed 42.5% of the power. The course of the manipulated variable is less oscillatory than in the first graph, leading to the lower overshoot of the closed-loop step response.

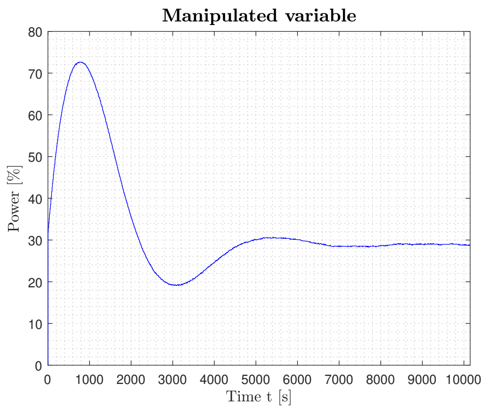


(a) $[K_i, K_p] = [0.0038, 2.7309]$

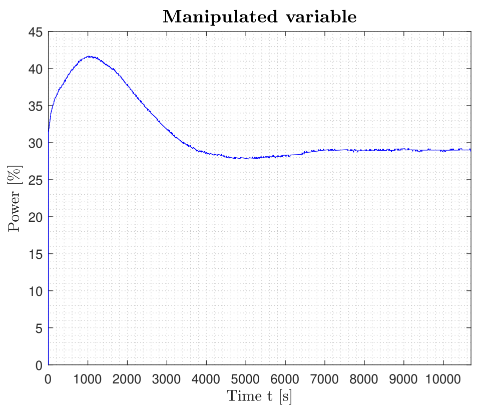


(b) $[K_i, K_p] = [0.0019, 2.7309]$

Figure 66: Closed-loop step responses containing PI controller parameters designed for X_1, X_2, X_3 design criteria for FOPMDT model



(a) $[K_i, K_p] = [0.0038, 2.7309]$



(b) $[K_i, K_p] = [0.0019, 2.7309]$

Figure 67: Manipulated values (in percents) for both step responses

6 Conclusion

The main aim of this thesis was to develop tools for process model identification, robust controller design for processes with uncertainty, and closed-loop performance optimization in the time domain. These tools have been successfully implemented in the GUI developed earlier by the author.

The first part focuses on the theoretical base of the author's approach. Here, we have described the robust controller tuning method based on the robust stability regions for PI controllers, the modeling of the process non-structural uncertainty, the Model set approach which consists of the experimental identification of the process moments, the construction of the model and Value set and the computation of the extremal processes for FO and IO processes, and the robust controller design, and finally we have mentioned the time domain integral criteria of optimality.

The concept of the used method began with defining the design criteria in the frequency domain, these criteria were then represented as the shaping points for the Nyquist curve. From the experimental identification and the *a priori* assumptions, the moment Model set was obtained, and the models of extremal processes were computed. Subsequently, for each design requirement for each process model, one robust stability region was obtained. The solution emerged as an intersection of all robust stability regions (if it existed). The intersection area contained all controllers which satisfied all design requirements for all processes. From the intersection area, one controller was selected according to the minimal value of the given integral criterion which led to the optimal time domain performance.

The second part of this thesis focused on the current state-of-the-art of controller design and process identification tools. We have described the structure and functionality and pointed out the advantages and disadvantages of each mentioned tool.

First, we have analyzed the PID H_∞ Designer. Its main advantages were finding all solutions (if the solution exists) using the H_∞ regions, finding optimal controller in the time domain using the integral criteria, including the uncertainty to the process model, automatic computations, data management, tools for closed-loop performance analysis, *etc.* Among the disadvantages was the computational time of some functions (*e.g.*, the computational time of the integral criteria over the region), or the inability to select simple stability margins for the Nyquist curve (*e.g.*, GM or PM).

Second, we have analyzed the functionality of some blocks in the REXYGEN studio. However, it has been found, that it is more oriented toward the implementation of the already-designed controller on a target device.

The third step was the analysis of Matlab's System identification toolbox. Although this toolbox provided good results in the process model estimation and important information about the precision of model estimation, the representation of uncertainty could turn out to be inconsistent, it did not include the extremal processes which create uncertainty boundary in the frequency domain.

In the third part of this thesis, we have described the implementation of developed tools. These tools extended the previously developed GUI.

The first extension was the identification module. This module included the standard Matlab commands for process identification which lead to a nominal process model, and the FOPMDT identification method which resulted in the process model including the non-structural uncertainty represented by the extremal processes obtained according to the Model set approach.

Next, the ability to design a robust controller for processes with uncertainty was shown. From the extremal processes, the samples containing IO transfer functions were selected for the controller design. Then three frequency domain controller requirements were specified, and for each design requirement for each process from the IO extremal process set, the robust stability region was obtained. Intersection was successfully found and one controller with approximately highest value of K_i coordinate was selected. Time domain and frequency domain sensitivity functions characteristics were depicted, all of which showing acceptable results. Thus, the robust controller design for process with uncertainty was successfully performed.

The last part of the implementation described the closed-loop time domain performance optimization tool. This tool was designed to search for an optimal robust controller by minimizing the integral criteria of optimality. This tool extended the author's previous work in which the intersection area of the robust stability regions was sampled, creating the mesh grid of $[K_i, K_p]$ parameters. For each PI controller sample, the closed loop was constructed and the selected integral criterion was computed resulting in 3D graphs depicting the integral criteria of optimality for the whole intersection area. In this thesis, we have worked with six integral criteria of optimality: IE, ISE, ITE, ITAE, IAE and IGSE. In this thesis, two optimization methods were shown.

First was the standard optimization method. It computed each integral criterion for all samples of the intersection region and then found the controller with the smallest value of each integral criterion. This method led to the global minimum of each integral criterion.

The second was the gradient optimization method. It started at some point of the sampled $[K_i, K_p]$ grid where it computed selected integral criterion. Then it computed the selected integral criterion for its nearest neighbors and selected the lowest value. This process repeated until the algorithm reached the local minimum of the integral criterion. Several experiments have been performed with different starting points. The final values of each criterion were not always the same, however, they were very similar, since they were located in the same close neighborhood. This was caused due to the shape of the 3D region. The next limitation of the gradient optimization method was the inability to include the region's boundary. In our experiments, the true minimal value of each integral criterion often appeared on the region's boundary (as it was shown for the standard optimization method). However, the final values of the gradient optimization method were in the close neighborhood of the value on the region's boundary.

Then, the standard and gradient optimization methods were compared for the changing mesh grid sampling, starting at 20x20 samples, followed by 40x40 samples, and finishing with 80x80 samples. For both methods, the IE criterion was minimized. The final IE values were very similar for both methods. The differences were given by the limiting attributes of the gradient optimization method. However, the computation time of the gradient method was significantly lower than that of the standard method. For the 20x20 samples, the difference between the methods was around 4 seconds, for the 40x40 samples, the difference was approximately 20 seconds, and for the 80x80 samples, the difference was around 90 seconds. Thus, despite the gradient method providing a sub-optimal solution in the form of a local minimum, and not including the region's boundary, it leads to significantly shorter computational time.

In the last part of the thesis, we have validated the designed tool on the incubation device. We have modeled the process response with sufficient precision using the Matlab commands as well as the FOPMDT method. For both cases, the same set of controller design requirements has been selected. For both models, we have obtained robust stability regions, and from the intersection area, we have selected two controllers, one with higher and one with lower integral gain. These controllers were used in the closed-loop systems connected to the incubator through REXYGEN.

At first, we have estimated the nominal process model using the Matlab commands. Then we selected two sets of PI controller parameters from the intersection region. Both PI controllers were tested on the incubation device. The first PI controller led to approximately 4 times quicker rise time, second PI controller led to approximately 1.5 times quicker rise time.

Second, we have computed the three characteristic numbers for the FOPMDT model from which the IO extremal processes were obtained. Robust stability regions and the intersection was computed for all extremal processes and the selected design requirements. From the intersection area, two sets of PI controller parameters were selected. Both PI controllers were tested on the incubation device. The first PI controller led to approximately 30% overshoot in the step response, 1000 seconds-long rise time, and 5250 seconds until the output was in the 2% tolerance band. The manipulated variable did not reach 75% of the heating power. The second PI controller has a significantly lower overshoot in the step response (approximately 9%), longer rise time (1000 seconds), and shorter duration before the output was in the tolerance band (4885 seconds). The manipulated variable of the second closed loop did not exceed 42.5% of the heating power. Thus, the implemented robust controller design method was successfully validated.

6.1 Future Works

Despite this thesis brought several results, it also laid the foundation for some future works. Here are some ideas which could be captured in the future works.

First, the PID and FO PI$^{\alpha}$D$^{\beta}$ robust stability regions could be included in the controller design method. Another tool could be implemented which would compute the intersection of the three-dimensional regions. With the improving Matlab SW, it could be possible to operate with a 3D graph and perhaps select a set of controller parameters from it.

Second, the Value set boundary for a finite total order of the process n could be implemented. The whole structure of the IO value sets could be implemented as well, for in this thesis only the vertices of IO value sets are implemented.

Next, in the closed-loop time domain performance optimization tool, in the case of more complex shapes, it might be necessary to improve the approximation method of the shape of this irregular region. Moreover, it would be ideal to include the region's border in the gradient optimization method.

Interesting would be the extension of the controller design section for the FO processes. Although this thesis provides the identified FO Model set, the commands for FO process models are not a standard part of the Matlab SW. However, there are advanced library-extensions where these commands are implemented. Thus, this functionality could be implemented in the future.

References

[1] T. H. Akkermans and S. G. Stan. Digital servo ic for optical disc drives. *Control Engineering Practice*, 9(11):1245–1253, 2001.

[2] K. H. Ang, G. Chong, and Y. Li. Pid control system analysis, design, and technology. *IEEE transactions on control systems technology*, 13(4):559–576, 2005.

[3] K. J. Åström. Control system design lecture notes for me 155a. *Department of Mechanical and Environmental Engineering University of California Santa Barbara*, 333, 2002.

[4] K. J. Åström. Feedback fundamentals. 2019.

[5] K. J. Åström and T. Hägglund. *PID controllers: theory, design, and tuning*, volume 2. Instrument society of America Research Triangle Park, NC, 1995.

[6] K. J. Åström and T. Hägglund. The future of pid control. *Control engineering practice*, 9(11):1163–1175, 2001.

[7] K. J. Åström, T. Hägglund, and K. J. Astrom. *Advanced PID control*, volume 461. ISA-The Instrumentation, Systems, and Automation Society Research Triangle Park, 2006.

[8] K. J. Åström and R. M. Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2008.

[9] T. Ausberger, K. Kubíček, P. Medvecová, and T. Myslivec. Test case generation for function block diagram based on blocks' predefined behaviour. In *2021 23rd International Conference on Process Control (PC)*, pages 206–211, 2021.

[10] T. Ausberger, K. Kubíček, P. Medvecová, T. Myslivec, and M. Štětina. Analytic method for automatic test case generation for function block diagram. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1799–1826, 2020.

[11] T. Ausberger, K. Kubíček, P. Medvecová, T. Myslivec, and M. Štětina. Model checking application on function block diagram model. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1807–1814, 2020.

[12] T. Ausberger, K. Kubíček, P. Medvecová, and J. Wolf. Verification of a safety-related i&c system for nuclear power plant by model checking, test case generation and automatic testing. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2022.

[13] N. Bazylev Dmitry, A. Margun Alexei, A. Zimenko Konstantin, K. A. Sergeevich, D. Ibraev Denis, and Č. Martin. Approaches for stabilizing of biped robots in a standing position on movable support. *Journal Scientific and Technical Of Information Technologies, Mechanics and Optics*, 97(3):418–425, 2015.

[14] Calerga Sarl. Calerga. `https://calerga.com/products/Sysquake/index.html`. Accessed: 15-03-2021.

[15] M. Čech. Návrh robustních regulátorů s omezenou strukturou pro systémy neceločíselného řádu. *Plzeň: Disertační práce, ZČU Plzeň*, 65:67, 2008.

[16] M. Čech. Web-based fractional pid controller design: www. pidlab. com. *IFAC-PapersOnLine*, 51(4):563–568, 2018.

[17] M. Čech, A.-J. Beltman, and K. Ozols. Pushing mechatronic applications to the limits via smart motion control. *Applied Sciences*, 11(18):8337, 2021.

[18] M. Čech, J. Königsmarková, and M. Schlegel. Robust PID tuning rules for a class of fractional-order processes, 2008.

[19] M. Čech and M. Schlegel. Interval pid tuning rules for a fractional-order model set. *IFAC Proceedings Volumes*, 44(1):5359–5364, 2011.

[20] M. Čech and M. Schlegel. Computing pid tuning regions based on fractional-order model set. *IFAC Proceedings Volumes*, 45(3):661–666, 2012.

[21] M. Čech and M. Schlegel. Generalized robust stability regions for fractional pid controllers. In *2013 IEEE International Conference on Industrial Technology (ICIT)*, pages 76–81. IEEE, 2013.

[22] A. Charef, H. Sun, Y. Tsao, and B. Onaral. Fractal system as represented by singularity function. *IEEE Transactions on automatic Control*, 37(9):1465–1470, 1992.

[23] L. Desborough and R. Miller. Increasing customer value of industrial control performance monitoring-honeywell's experience. In *AIChE symposium series*, number 326, pages 169–189. New York; American Institute of Chemical Engineers; 1998, 2002.

[24] A. C. Dimian, C. S. Bildea, and A. A. Kiss. *Integrated design and simulation of chemical processes*. Elsevier, 2014.

[25] J. J. DiStefano, A. R. Stubberud, and I. J. Williams. *Feedback and control systems*, volume 2. McGraw-Hill, 2012.

[26] Faculty of applied sciences, Department of cybernetics. PIDlab. `https://www.pidlab.com/cs/`. Accessed: 15-03-2021.

[27] O. Garpinger, T. Hägglund, and K. J. Åström. Performance and robustness trade-offs in pid control. *Journal of Process Control*, 24(5):568–577, 2014.

[28] N. Heymans. Fractional calculus description of non-linear viscoelastic behaviour of polymers. *Nonlinear dynamics*, 38:221–231, 2004.

[29] K. Karel. Modelově orientovaný vývoj softwaru: řízení spojky automatické převodovky kamionů, 2019.

[30] V. Kharitonov. Asymptotic stability of an equilibrium position of a family of systems of linear differential equations. *Differential'nye Uraveniya*, 14:1483–1485, 1978.

[31] J. Königsmarková and M. Čech. Robust PI/PID parameter surfaces for a class of fractional-order processes. *IFAC-PapersOnLine*, 51(4):763–768, 2018.

[32] Y. Kozhushko, D. Pavković, T. Karbivska, P. Safronov, and O. Bondarenko. Robust control of battery-supercapacitor energy storage system using kharitonov theorem. In *2020 IEEE 14th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG)*, volume 1, pages 550–555. IEEE, 2020.

[33] W. Krajewski, A. Lepschy, and U. Viaro. Designing pi controllers for robust stability and performance. *IEEE transactions on control systems technology*, 12(6):973–983, 2004.

[34] K. Kubíček, M. Čech, and J. Škach. Continuous enhancement in model-based software development and recent trends. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 71–78, 2019.

[35] Y. Li, K. H. Ang, and G. C. Chong. Pid control system analysis and design. *IEEE Control Systems Magazine*, 26(1):32–41, 2006.

[36] L. Ljung. System identification-theory for the user 2nd edition ptr prentice-hall. *Upper Saddle River, NJ*, 1999.

[37] D. Máj. Regulace teploty na modelu inkubačního zařízení. 2022.

[38] R. Matušů, B. Şenol, and L. Pekař. Robust pi control of interval plants with gain and phase margin specifications: Application to a continuous stirred tank reactor. *IEEE Access*, 8:145372–145380, 2020.

[39] D. McFarlane and K. Glover. A loop-shaping design procedure using H_∞ synthesis. *IEEE transactions on automatic control*, 37(6):759–769, 1992.

[40] A. Megretski. The waterbed effect, 2004.

[41] A. V. Mokshin, R. M. Yulmetyev, and P. Hänggi. Simple measure of memory for dynamical processes described by a generalized langevin equation. *Physical review letters*, 95(20):200601, 2005.

[42] H. Nyquist. Regeneration theory. *Bell system technical journal*, 11(1):126–147, 1932.

[43] F. Pan, H. Liao, J. Luo, and Y. Xue. Itae-optimal pi controller based on genetic algorithm for low-order process with large time delays. In *2014 20th International Conference on Automation and Computing*, pages 134–139. IEEE, 2014.

[44] I. Podlubny. *Fractional differential equations.* 1999.

[45] REX Controls, s.r.o. PID Hinf Designer. `https://www.pidlab.com/cs/`. Accessed: 3-2-2023.

[46] REX Controls, s.r.o. REXYGEN. `https://www.rexygen.com/`. Accessed: 16-3-2023.

[47] REX Controls, s.r.o. *Function Blocks of REXYGEN: Reference manual, version 2.50.12.* 2022.

[48] H. Rotstein, N. Galperin, and P.-O. Gutman. Set membership approach for reducing value sets in the frequency domain. *IEEE transactions on automatic control*, 43(9):1346–1350, 1998.

[49] H. Sandberg and B. Bernhardsson. A bode sensitivity integral for linear time-periodic systems. *Automatic Control, IEEE Transactions on*, 50:2034 – 2039, 01 2006.

[50] M. Schlegel. *New approach for robust design of industrial controllers.* PhD thesis, Habilitation thesis, University of West Bohemia, Pilsen, 2000.

[51] M. Schlegel. Exact revision of the ziegler-nichols frequency response method. In *Proc. of IASTED Int. Conf. on control and applications, Cancun, Mexico*, 2002.

[52] M. Schlegel. *Regulační smyčka.* 2023.

[53] M. Schlegel, P. Balda, and M. Štetina. Robust pid autotuner–method of moments. *Automatizace*, 46(4):242–246, 2003.

[54] M. Schlegel and M. Čech. Computing value sets from one point of frequency response with applications. In *Proceedings of the 16th IFAC world congress*, volume 1, 2005.

[55] M. Schlegel and P. Medvecová. Design of PI controllers: H_∞ region approach. *IFAC-papersonline*, 51(6):13–17, 2018.

[56] M. Schlegel and J. Mertl. Process identification for automatic tuning of industrial controllers. *Process Control 2010*, 2010.

[57] M. Schlegel and O. Večerek. Robust design of smith predictive controller for moment model set. *IFAC Proceedings Volumes*, 38(1):427–432, 2005.

[58] J. F. Smuts. *Process Control for Practitioners: How to Tune PID Controllers and Optimize Control Loops.* OptiControls, 2011.

[59] The MathWorks Inc. Control System Toolbox (R2021a). `https://www.mathworks.com/products/control.html`. Accessed: 21-03-2021.

[60] The MathWorks Inc. Control System Tuner (R2021a). https://www.mathworks.com/help/slcontrol/tuning-with-control-system-tuner.html. Accessed: 29-03-2021.

[61] The MathWorks Inc. Matlab App Designer (R2021a). https://www.mathworks.com/products/matlab/app-designer.html. Accessed: 07-03-2023.

[62] The MathWorks Inc. System Identification Toolbox (R2021a). https://www.mathworks.com/products/sysid.html. Accessed: 21-11-2022.

[63] The MathWorks Inc. Matlab version: 9.10.0 (R2021a), 2021.

[64] O. Turksoy, S. Ayasun, Y. Hames, and Ş. Sönmez. Computation of robust pi-based pitch controller parameters for large wind turbines. *Canadian journal of electrical and computer engineering*, 43(1):57–63, 2019.

[65] H. Unbehauen. Controller design in time-domain. 2011.

[66] V. Žán. Interaktivní nástroje pro výpočet vícerozměrných regionů robustní stability. 2021.

[67] V. Žán, K. Kubíček, and M. Čech. Design of robust pi controller by combining robustness regions with time-domain criteria. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2022.

[68] J. G. Ziegler, N. B. Nichols, et al. Optimum settings for automatic controllers. *trans. ASME*, 64(11), 1942.