

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

Interactive tools for computing
multi-dimensional robust stability regions
for simple controllers

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd
Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE (projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Vilém ŽÁN**
Osobní číslo: **A18B0560P**
Studijní program: **B3918 Aplikované vědy a informatika**
Studijní obor: **Kybernetika a řídicí technika**
Téma práce: **Interaktivní nástroje pro výpočet vícerozměrných regionů robustní stability**
Zadávající katedra: **Katedra kybernetiky**

Zásady pro vypracování

1. Analyzujte aktuální stav vývoje on-line nástrojů pro návrh jednoduchých regulátorů.
2. Analyzujte možnosti návrhu PID regulátoru v systému Matlab.
3. Vytvořte GUI pro výpočet a znázornění vícerozměrných regionů robustní stability jednoduchých regulátorů.
4. Vytvořte výpočetní modul pro multikriteriální optimalizaci parametrů.
5. Otestujte vyvinuté nástroje na praktických příkladech.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

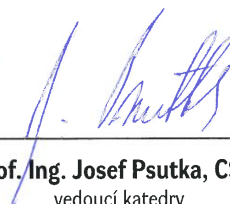
1. Čech, M., Schlegel, M., *Computing PID tuning regions based on fractional-order model set* (2012) IFAC Proceedings Volumes (IFAC-PapersOnline), 2 (PART 1), pp. 661-666.
2. Schlegel, M., Čech, M., *Computing value sets from one point of frequency response with applications* (2005) IFAC Proceedings Volumes (IFAC-PapersOnline), 16, pp. 325-330.
3. J. Königsmarková, M. Čech. *Robust PI/PID parameter surfaces for a class of fractional-order processes*. IFAC-PapersOnLine, Volume 51, Issue 4, 2018, Pages 763-768
4. Čech, M. *Web-Based Fractional PID Controller Design: $\omega\omega$.PIDlab.com*. IFAC-PapersOnLine Volume 51, Issue 4, 2018, Pages 563-568
5. Saurabh Srivastava, V.S. Pandit. *A PI/PID controller for time delay systems with desired closed loop time response and guaranteed gain and phase margins*. Journal of Process Control 37 (2016) 70-77
6. A. Jeya Veronica1 / N. Senthil Kumar1 / Francisco Gonzalez-Longatt2. *Design of Load Frequency Control for a Microgrid Using D-partition Method*. International Journal of Emerging Electric Power Systems. 2020; 20190175
7. Ying J. Huang*, Yuan-Jay Wang. *Robust PID tuning strategy for uncertain plants based on the Kharitonov theorem*. ISA Transactions 39 (2000) 419-431
8. Ehsan Gholamzadeh Nabati, Sebastian Engell. *Online Adaptive Robust Tuning of PID Parameters*
9. S.-Z. Zhao a, M. Willjuice Iruthayarajan b, S. Baskar c, P.N. Suganthan. *Multi-objective robust PID controller tuning using two lbests multi-objective particle swarm optimization*. Information Sciences 181 (2011) 3323-3335

Vedoucí bakalářské práce: **Ing. Martin Čech, Ph.D.**
Výzkumný program 1

Datum zadání bakalářské práce: **15. října 2020**
Termín odevzdání bakalářské práce: **24. května 2021**



Doc. Dr. Ing. Vlasta Radová
děkanka



Prof. Ing. Josef Psutka, CSc.
vedoucí katedry


V Plzni dne 15. října 2020

P R O H L Á Š E N Í

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

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

V Plzni dne 18.května 2021



.....
vlastnoruční podpis

Poděkování

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

Anotace

Cílem této práce je seznámení se s teorií regionů robustní stability, analyzování dostupných nástrojů pro návrh jednoduchých regulátorů a spojení těchto znalostí při vytváření grafického uživatelského rozhraní pro design těchto regulátorů.

V první části práce je vysvětlena teorie řízení a teorie regionů robustní stability. V druhé části práce je analyzovaný současný stav online nástrojů pro návrh regulátorů. Poté jsou analyzovány možnosti návrhu regulátorů v softwaru Matlab. Ve třetí části práce je popsána implementace vlastního navrženého uživatelského rozhraní. Do uživatelského rozhraní je zahrnuta možnost provést multikriteriální optimalizaci. V poslední části práce je validována správná činnost vyvinutého rozhraní na vybraných příkladech.

Klíčová slova: teorie řízení, PID regulace, regiony robustní stability, Nyquistova křivka, Matlab, GUI, multikriteriální optimalizace

Annotation

The aim of this thesis is understanding and describing the theory of robust stability regions, analysis of available simple controller tuning tools, and the combination of this knowledge in order to develop a graphical user interface for the design of these controllers.

In the first part of the thesis, the control theory and the robust stability regions theory is explained. The second part focuses on analysis of the state of the current online controller design tools. Next, the possibilities of controller design in the Matlab software are analyzed. In the third part, the implementation of the designed user interface is described. The ability to perform multi-criteria optimization is included in the user interface. In the final part of this thesis, the correct functionality of the developed interface is validated on selected examples.

Keywords: control theory, PID control, robust stability regions, Nyquist plot, Matlab, GUI, multi-criteria optimization

Contents

1	Introduction	1
1.1	General introduction	1
1.2	Objectives	2
2	Concept and approach	4
2.1	Simple feedback control loop	4
2.2	Robust stability regions for simple controllers	9
2.2.1	Design criteria	10
2.2.2	PI robust stability regions	13
2.2.3	PI ^α robust stability regions	15
2.2.4	Moment-model Set	18
2.2.5	Introduction to the proof of the monotony of $A(\omega)$ and $\varphi(\omega)$	19
2.2.6	Proof of the monotony of $A(\omega)$ and $\varphi(\omega)$	20
2.2.7	Conclusion of the proof	21
3	Analysis of existing tuning tools	22
3.1	Online controller tuning tools	22
3.1.1	Sysquake	22
3.1.2	PID Tuner	25
3.1.3	PIDlab	28
3.2	PID controller tuning methods in Matlab	32
3.2.1	PID Tuner	33
3.2.2	Control System Designer	34
3.2.3	Control System Tuner	37
3.2.4	Conclusion	37
4	Implementation of the designed interactive tool	39
4.1	App Designer	39
4.2	Designed graphical user interface	39
4.2.1	Implementation of the PI control algorithm	41
4.2.2	Visualization of the PI ^α robust stability regions	44
5	Multi-criteria optimization	47
6	Validation	52
6.1	Process set	52
6.2	Coupled tanks	55
7	Conclusion	60
7.1	Thesis results summary	60
7.2	Future works	60
	References	62

1 Introduction

1.1 General introduction

This thesis focuses on studying, describing and understanding the robust stability regions theory, and fractional-order model set based PID controller regions computing method, followed by analysing the current state-of-the-art of the online and Matlab controller tuning methods, resulting in the utilizing the gathered knowledge to develop an interactive graphical user interface (GUI) which would implement the robust stability regions theory for controller tuning and multi-dimensional stability regions visualization.

This thesis starts with a brief introduction to the control theory and robust stability regions theory. The control theory introduction begins with the explanation of the open and closed control loop, system modeling, Proportional-integral-derivative (PID) controllers, and other control structures [1][2]. After that, we will introduce the robust stability regions theory for simple controllers [3]. In this subsection, we will describe how the controller is designed using the Nyquist plot shaping. We will explain, which design requirements are used in order to tune a robust controller, and how we can express the design criteria as Nyquist plot shaping points. Next, we will move to the PI robust stability regions subsection [4] where the PI controller parameters expression will be derived step by step. It will be shown that the PI controller parameters create a frequency parametrized curve which defines the boundary of the robust stability region. In the PI^α robust stability regions subsection, we will introduce the PI^α controller parameters expressions, and we will demonstrate the visualization of the PI^α multi-dimensional regions. We will show the computation of the first encirclement of the robust stability region in the first quadrant of the controller parameters plane for both PI and PI^α controllers which is essential for controller design. Next, Moment-model set theory will be introduced [5][6], resulting in the proof of the monotony of magnitude and phase of the *essentially monotone processes* (Åström and Häglund (2006)) [7]. Together with the Parseval's theorem [8], we will verify assumption about *essentially monotone processes* being the ideal robust stability region method validation set [9][10].

Next section focuses on the current state-of-the-art of the online controller tuning tools and the Matlab PID controller tuning methods. We will point out three online controller tuning tools – Sysquake [11], PID Tuner [12], and PIDlab [13]. We will describe the structure and functionality of each tool. The controller design methods used in each tool will be mentioned. It will be shown whether any of these tools implements the robust stability regions theory. As for the Matlab methods, we will point out tools from the Control system toolbox [14]. We will analyze whether Matlab software provides manual or automatic algorithms for controller tuning, as well as if it implements robust stability regions method.

In the following section, we will describe the implementation of the designed GUI. We will introduce the platform used for GUI development which is a Matlab app building interface App Designer [15]. Matlab software was chosen mainly due to its convenient algorithms for matrix manipulations and fast numeric computing [16]. After that, we will show the structure and interactive functionality of the tool. We will focus on the implementation of controller design utilizing the Nyquist plot shaping method, robust stability regions algorithms, and invented algorithms computing the first quadrant of the first origin encirclement of the robust stability region. The visualization of the multi-dimensional robust stability regions will be shown here.

In the next section, we will perform the multi-criteria controller parameters (e.g. gain and phase margins) optimization in the designed GUI. We will show that the set of controller parameters satisfying every design criteria for each process is given by the intersection of the robust stability regions. As an example, the multi-criteria optimization will be performed for two processes each with several design criteria.

The last section of the bachelor thesis focuses on the GUI validation. First part of the validation

consists of process set with two design criteria. In the second part, we will perform the validation on the model of real physical coupled tanks system. The results of the validation will be discussed here.

1.2 Objectives

PID control can be found in almost every industrial sector. PID controllers are used mainly due to the simplicity of the control law and necessity to tune only few parameters. Moreover, PID controller parameters have clear physical interpretation [4]. However, many of the deployed PID controllers have been tuned sub-optimally or even use default parameters. This leads to significant annual economic and material losses. Consequently, control loop design is not a resolved issue. Over the years many controller tuning methods have been developed. One of the most popular was the Ziegler-Nichols method (Ziegler and Nichols (1942)). Unfortunately, this method together with the majority of the following PID controller tuning methods is not very reliable [17][18]. These traditional methods are neither systematic nor guarantee fulfillment of design specifications for an exactly given class of process transfer functions [6]. One of the most reputable tuning method was introduced by (Åström and Häglund (2006)). This method uses a large set of processes integrated into the design procedure. However, each method offers only nominal controller parameters. None of them provides an area of all parameters guaranteeing some required closed loop robustness and bandwidth [7][5]. This means that there is no one globally accepted reliable automatic controller tuning method.

The main objective was to create well-arranged interactive controller tuning method capable of automatic computation of suitable controller parameters for wide spectrum of stable non-oscillatory processes and satisfying several design criteria, therefore it could be potentially easily applicable in the industrial practice, or for other commercial purposes. In the field of industry or research, we often operate not only with one but rather with a larger amount of different processes. It is a job for the experts or engineers to find controller parameters which satisfy several number of control requirements for the given set of processes. Due to system uncertainty it is important to design robust controller [3]. This tool provides solution in the form of robust regions, where one robust region represents one particular set of controllers.

The GUI was chosen as the controller tuning tool. The GUI implements the robust stability regions method, and is able to automatically compute the boundary of the controller parameter set for selected process and its design requirement. If we select more processes with more design criteria, the resulting set of controller parameters is obtained as an intersection of all regions. The GUI able to determine whether if it is possible to design controller for selected criteria. The solution exists if region intersection exists. If region intersection does not exist then solution does not exist as well. One of the benefits of this interface is that it is graphical. Graphical method makes very sophisticated controller design available even for someone with not as much experience. The user does not have to know the exact tuning algorithms in order to design a controller. The designing algorithms are executed as a callback functions of GUI components, meaning we are able to design a controller by clicking buttons and dragging plots.

The GUI can be used during standard controller development cycle which is described by V-Model (Figure 1). V-Model describes the process of controller design and implementation in the industrial practice. In the industry a modification of controller settings is often needed, because system dynamic changes which is caused by time degradation. In this kind of situation we do not have access to the workstation, that was used for the initial controller construction and initial simulations and measurements. There is no time to properly recreate and adjust our controller design process and carry out all necessary simulations, such as Model in the loop (MIL), Software in the loop (SIL), Processor in the loop (PIL) or Hardware in the loop (HIL) which are verification steps taking place before the controller is deployed into the hardware for production. MIL, SIL, PIL and HIL are all essential parts of development process [19].

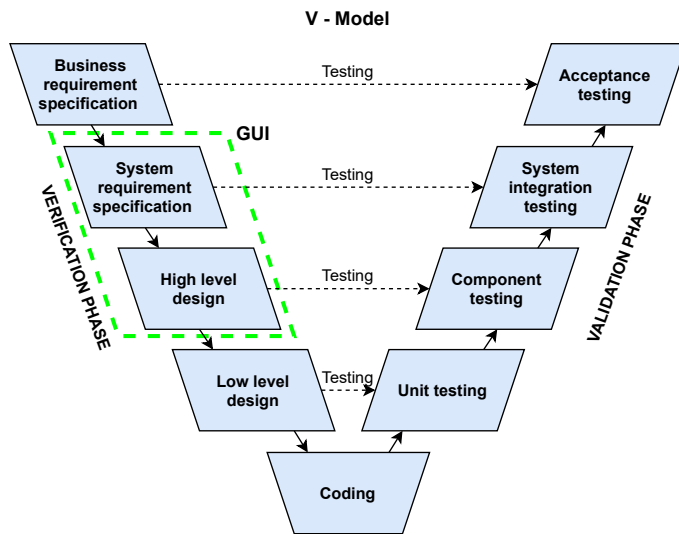


Figure 1: V-Model

expert system analysis and construction of mathematical or software model in a form which would satisfy the system requirements. The technological possibilities are discussed by experts or engineers in this sections. The coding step follows containing the implementation of developed system. This step is common for verification and validation phase. The validation phase follows, where each section validates the relevant verification phase section. The first part of validation phase is Unit testing, where bugs are eliminated at code or unit level. Component testing handles validation of High level design, meaning whether the communication and data operations works correctly. System integration test consists of application functionality, inter-dependency and communication testing. It is usually client's job to perform the load and performance testing, stress testing, regression testing etc. The Acceptance testing is the final step of V-Model. It validates if the whole application meets the business expectations and determines whether the system is ready for use in real time.

Every designed controller is checked against the original specification when designing controller according to V-Model. If the model does not satisfy the requirements, the counterexample is returned. The model can be redesigned until the requirements are met. Meaning that the model design is improved before the implementation phase. This is extremely critical phenomenon because design errors and inaccuracies are otherwise implemented and detected in a testing phase. The later a defect is detected the higher repair costs are. Without checking the design, a defect can even remain undetected after the test phase. Such solution is unreliable and potentially dangerous because its behaviour does not meet the requirements [20].

The GUI would be a part of the System requirement specification and High level design. It lets the user to define a set of design criteria and design a robust controller. Ideally, we would be able to fully integrate the GUI into the design procedure, and link it properly with other design steps.

V-Model consists of two phases: verification phase and validation phase. We start in the verification phase at the upper left corner in the Business requirements specification section, where the client requirements containing the ideal system behaviour are summed up. The System requirements specification follows which is an interpretation of Business requirements. At this stage, the system functionality, interface, data operations etc. is described. The user information should be provided in this section in an understandable form for the user. Next part of the verification phase is High level design and Low level design. This sections depict how the implementation will occur, and contain ex-

2 Concept and approach

Before moving to the robust region theory, I would like to present just brief introduction to the control theory. The central and perhaps the most essential concept of the control theory is a system (or a process). System is a set of elements interacting with each other. The control theory deals with describing dynamical systems and designing controllers for them. Time behaviour and structure of the dynamical system can be described by a mathematical model. Mathematical model usually consists of a vector of differential equations. The vast majority of real systems are non-linear, and thus the control theory is much harder to be applicable for the control design. There are no general controller tuning methods for the non-linear processes. The non-linear system has to be linearized in the working point so that the controller designer could work with it. The transfer function is obtained as a Laplace transformation of the differential equation of the linear system. We can create a state-space representation from the transfer function which describes the dynamics of the state vector and output vector.

Once we have a model of the system, we can design a controller. Many processes are unstable, oscillatory, and almost every real process has a time delay. The process controller should drive the system to a desired state, and ensure that the closed loop will be stable. There are of course other requirements on the controller, such as minimizing any delay, overshoot, or steady-state error. When the controller is connected directly to the process, the open loop is obtained (Figure 2).

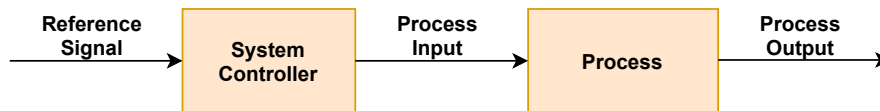


Figure 2: Open loop control schema

However, the open loop control would work well only if we had an exact mathematical model of the real process, and if there were no disturbances affecting the process dynamics. During the open loop control, the controller has no information about the process output. This can be corrected by using the feedback (or closed) control loop. Feedback loop and open loop often combine together to reach better control performance.

2.1 Simple feedback control loop

In this section, the simple feedback control loop together with the proven controller forms will be introduced. The simple feedback control loop (or closed loop) can be seen on the Figure 3. The system and variable notation table has been added for better orientation in the thesis (Table 1). The closed loop controller receives the information about the process output in form of error which is calculated from the measured process variable and the reference signal. This gives the controller ability to reach stability, and a desired steady state much easier.

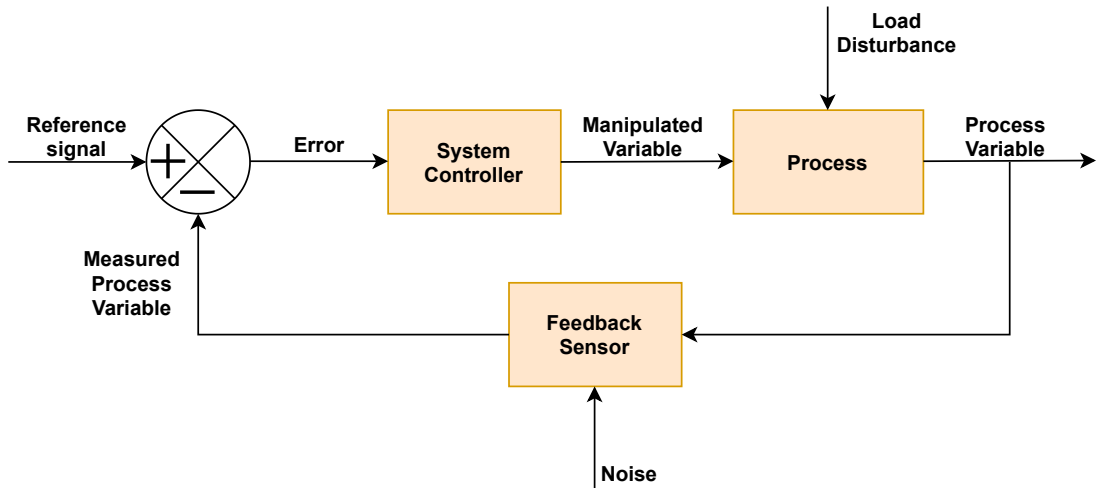


Figure 3: Feedback control loop schema

Process	$P(s)$
Controller	$C(s)$
Set Point	$y_{sp}(t)$
Manipulated (Control) Variable	$u(t)$
Process Variable	$y(t)$
Measured Process Variable	$\hat{y}(t)$
Error	$e(t) = y_{sp}(t) - \hat{y}(t)$
Load Disturbance	$d(t)$
Noise	$z(t)$

Table 1: System and variable notation

The open loop transfer function can be expressed as

$$F_o(s) = C(s)P(s). \quad (1)$$

The closed loop transfer function is defined as

$$F_c(s) = \frac{C(s)P(s)}{1 + C(s)P(s)}. \quad (2)$$

Main objective of the feedback control is to ensure the closed loop stability. There are several algebraic stability criteria (e.g. Routh–Hurwitz stability criterion) and frequency stability criteria (e.g. Mikhailov stability criterion) which can be used to design a stable closed loop system. However the most popular, and probably most important is the Nyquist stability criterion. It is a frequency graphical method which focuses on the course of Nyquist curve in the complex plane. Nyquist curve can be expressed as $L(j\omega) = C(j\omega)P(j\omega)$. It is equivalent to the frequency response of the open loop transfer function. Nyquist criterion says that closed loop is stable if the number of critical point $[-1, j0]$ encirclements in the negative sense is equal to the number of open loop unstable poles. *Note: Poles are the roots of the transfer function denominator, zeros are the roots of the transfer function nominator.* Nyquist stability criterion provides the necessary and sufficient condition for the stability of the closed loop. One of the advantages of the Nyquist method is that it determines the robustness of the stability. If the Nyquist curve passes far from the critical point

$[-1, j0]$, the system has lesser tendency to be unstable.

If we want to design robust controller, we must ensure that all four sensitivity functions are stable. Sensitivity functions can be seen in the Table 2.

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

Table 2: Sensitivity functions

The sensitivity functions are given by the formulas:

$$S(s) = \frac{1}{1 + C(s)P(s)}, \quad (3)$$

$$T(s) = \frac{C(s)P(s)}{1 + C(s)P(s)}, \quad (4)$$

$$CS(s) = \frac{C(s)}{1 + C(s)P(s)}, \quad (5)$$

$$PS(s) = \frac{P(s)}{1 + C(s)P(s)}. \quad (6)$$

It has been found empirically that a useful controller structure is represented by Proportional Integral Derivative (PID) controller (Åström et al. (1995)). The "textbook" version of the PID algorithm can be described by the control law formula:

$$u(t) = K \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right), \quad (7)$$

where $u(t)$ is the control variable, $e(t)$ is the control error. The control variable is thus a sum of three terms: the P-term (which is proportional to the error), the I-term (which is proportional to the integral of the error), and the D-term (which is proportional to the derivative of the error). The controller parameters are proportional gain K , integral time T_i , and derivative time T_d [1]. The detailed schema of the PID controller with parallel structure can be seen on the Figure 4.

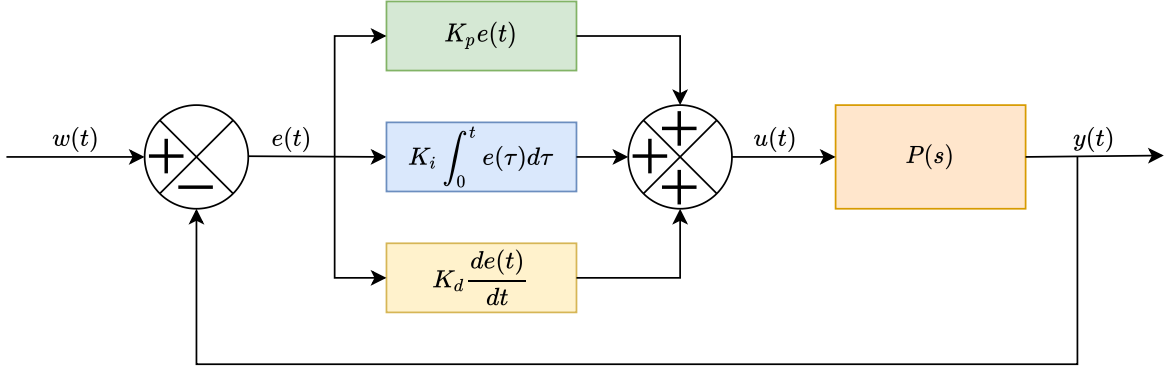


Figure 4: Non-interacting PID controller form
Note: Variable $w(t)$ represents reference signal.

In the case of pure proportional control, the reduced proportional form of PID controller can be used. This reduced form is called P controller which can be described by the control law formula:

$$u(t) = Ke(t) + u_b, \quad (8)$$

where u_b is a bias.

In the case of integral control, the PID controller is reduced to PI controller. Control law of the PI controller can be expressed as

$$u(t) = K \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau \right). \quad (9)$$

During the controller tuning, the integral time constant T_i is often substituted with the integral gain K_i , $K_i = \frac{1}{T_i}$. The main objective of the integral action is to make sure that the process output agrees with the set point in steady state which means that the steady-state error will always be zero. [1].

In the case of derivative control, the PID controller reduction in the form of PD controller is required. Control law of the PD controller can be expressed as

$$u(t) = K \left(e(t) + T_d \frac{de(t)}{dt} \right). \quad (10)$$

The PD controller is used mainly to improve the closed-loop stability. PD control is proportional to the predicted process output where the prediction is made by extrapolating the error [1].

Applying Laplace transformation results in following controller transfer functions:

- P controller:

$$C_P(s) = K. \quad (11)$$

- PI controller:

$$C_{PI}(s) = K \left(1 + \frac{1}{T_i s} \right). \quad (12)$$

- PD controller:

$$C_{PD}(s) = K (1 + T_d s). \quad (13)$$

- PID controller:

$$C_{PID}(s) = K \left(1 + \frac{1}{T_i s} + T_d s \right). \quad (14)$$

Since the relative order of the PID controller in the form (14) is lesser than zero, it is not physically feasible to construct such controller. To solve this problem, the filtration of the derivative term is added. The PID controller formula is then given as

$$C_{PID}(s) = K \left(1 + \frac{1}{T_i s} + \frac{T_d s}{\frac{T_d}{N} s + 1} \right), \quad (15)$$

where N is the fixed parameter determining the time constant of a derivative term filter. The value of N is usually chosen according to the noises in the measured signals [9].

Recently mentioned controllers can be used in the single degree-of-freedom (1-DOF) feedback control systems. However in the practice, the structure of controlled process is more complicated or not-observable. In order to control such structure, we would need to influence the set point response, and thus obtain more freedom in process control. We would achieve that by including set point weighting on the proportional and derivative terms. The set point weighting is implemented in the two degree-of-freedom (2-DOF) PID control structure (Figure 5). The standard 2-DOF PID control law is described by the ISA form as

$$U(s) = K \left(b Y_{sp}(s) - Y(s) + \frac{1}{T_i s} (Y_{sp}(s) - Y(s)) + \frac{T_d s}{\frac{T_d}{N} s + 1} (c Y_{sp}(s) - Y(s)) \right), \quad (16)$$

where $Y(s)$, $Y_{sp}(s)$ and $U(s)$ are Laplace transforms of $y(t)$, $y_{sp}(t)$ and $u(t)$. Parameters b and c are weightings that influence the set point response. The used values of b and c are typically 0 or 1 in commercial controllers [1]. *Note: If $b = c = 1$, then 2-DOF structure is equal to the 1-DOF structure.*

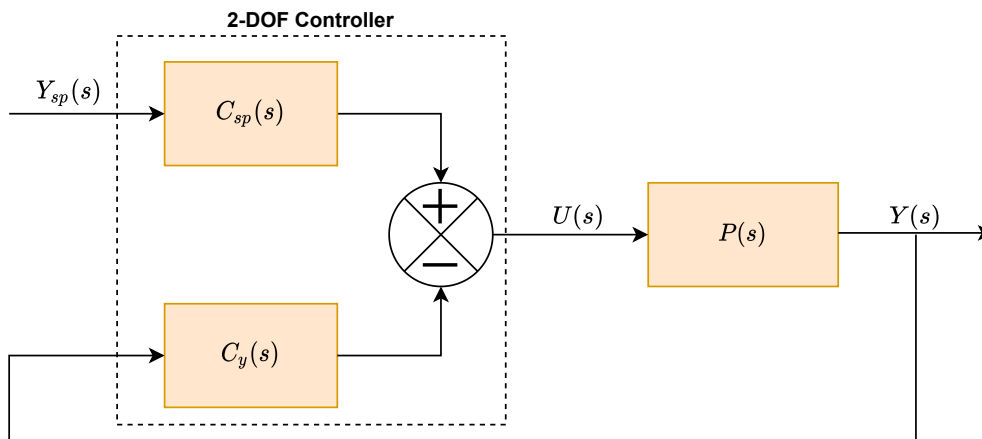


Figure 5: 2-DOF PID controller structure

Despite its benefits, the PID controller is not the only controller form that is used in the engineering practice. Choosing PID control is appropriate in situations when we consider the system to be a black-box which means that we do not know, or do not need its state space model. This happens for example when we control temperature or pressure. However very often the control loop is not isolated, and we have to deal with load disturbances. PID control loops often combine together where behaviour of one controller affects the behaviour of another. This is a common case in process control, or in energetics. However if the system is considered to be a white-box, we can operate with the state space model. In this case we can use more complex form of control algorithm called Linear-quadratic regulator (LQR). Linear quadratic regulator comprises of a state estimator and a state feedback loop. It is suitable for controlling processes with measurable state variables for example in robotics because the position, velocity, or acceleration of the robot is measurable. However some state variables does not have to be physically feasible. In such case, the LQR controller would not be optimal controller choice.

In some cases we can combine more controllers together and create a cascade control. Cascade control is used when there are more than one measurements, but only one control variable is available. In general, cascade control is recommended for slow processes which are controlled by means of a relatively fast process. Cascade control is effective against disturbances having a measurable effect before the process output [2].

Despite the fact that there are many types of controllers, in this thesis we will use mainly controllers from the PID "family", mainly due to their simplicity and efficiency.

2.2 Robust stability regions for simple controllers

In the field of process control there is often a need to design 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. The resulting changes affect the system dynamics meaning that the system technical parameters (e.g. stiffness, damping, or torque coefficients) obtained from the system identification are being changed over the time. This leads to a stage where previously designed controller may not function correctly. To prevent this situation from happening, it is important to satisfy certain design requirements. It is optimal to select more design criteria at one time.

In order to improve the robustness, the controller could be tested on the model set of the processes which implements the system uncertainty. The nominal model is only an approximation of a

real system. If the load disturbance appears in the system, the nominal model will not be ideal for the controller design. In this case we would have the set of design criteria and the model set. The aim would be to find a set of all possible controllers suitable for each system from the model set and for each design requirement. This controller set could be expressed as a robust stability region of the controller parameters. All controllers satisfying every design criteria for all systems would be obtained as an intersection of the partial controller sets. If the intersection exists, it contains an infinite number of controllers. If it does not exist (the intersection is empty), it means that there is not a suitable controller. In the case that the intersection is not empty, we can select optimal controller according to the optimization criteria. Knowing the boundary of the controller set, the optimization would select controller from the set of suitable controllers. The task hierarchy of the controller design implementing the robust stability region theory is shown on the Figure 6.

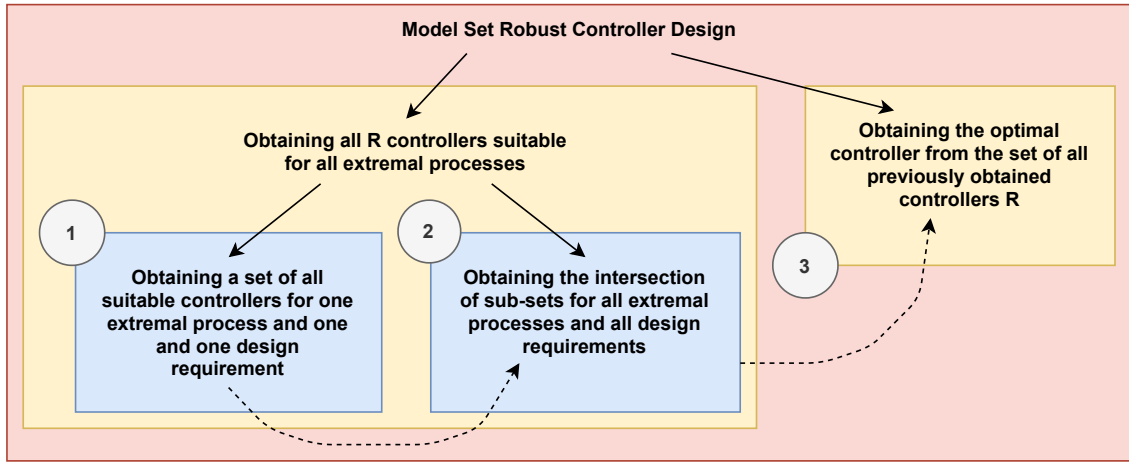


Figure 6: Robust controller design task hierarchy with the set model approach and the solution procedure

2.2.1 Design criteria

While designing the control loop, there are six main 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

The disturbance rejection parameter ε_S could be expressed as the upper magnitude limit of the sensitivity function $S(j\omega)$ for the frequencies $\omega \in [0, \omega_S]$ (Subfigure 7a). The bandwidth parameter ε_T is the upper magnitude limit of the complementary sensitivity function $T(j\omega)$ in the frequency interval $\omega \in [\omega_T, \infty]$ (Subfigure 7b).

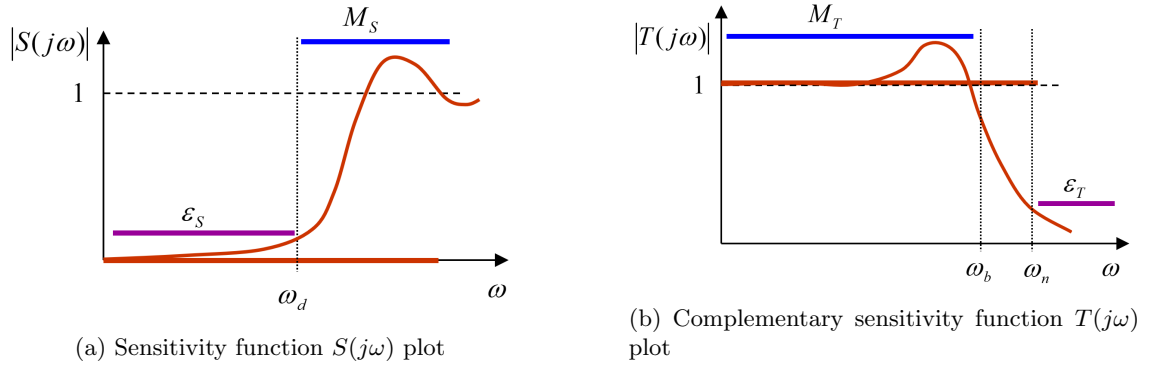


Figure 7: Real sensitivity functions $S(j\omega)$, $T(j\omega)$ constraint requirements [3]

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

$$|S(j\omega)| = 0, \forall \omega, \quad (17)$$

$$|T(j\omega)| = 1, \forall \omega. \quad (18)$$

However the reality is more complex than theory. Due to consequences of the Bode integral formula [21], we are only able to satisfy real sensitivity functions constraint requirements (Figure 7) given as following:

$$|S(j\omega)| < \varepsilon_S, \forall \omega \in [0, \omega_d], \quad (19)$$

$$|S(j\omega)| < M_S, \forall \omega, \quad (20)$$

$$|T(j\omega)| < M_T, \forall \omega, \quad (21)$$

$$|T(j\omega)| < \varepsilon_T, \forall \omega \in [\omega_n, \infty]. \quad (22)$$

This sensitivity functions constraints can be displayed in the complex plane in the form of M -circles and ε -circles (Figure 11). The center and radius of the M_S -circle and ε_S -circle is $[-1, j0]$, and $1/M_S$, $1/\varepsilon_S$ respectively. The center and radius of the M_T -circle and ε_T -circle can be obtained as

$$c = -\frac{M_T^2}{M_T^2 - 1}, \quad (23)$$

$$r = \frac{M_T^2}{|M_T^2 - 1|}, \quad (24)$$

$$c_\varepsilon = -\frac{\varepsilon_T^2}{\varepsilon_T^2 - 1}, \quad (25)$$

$$r_\varepsilon = \frac{\varepsilon_T^2}{|\varepsilon_T^2 - 1|}. \quad (26)$$

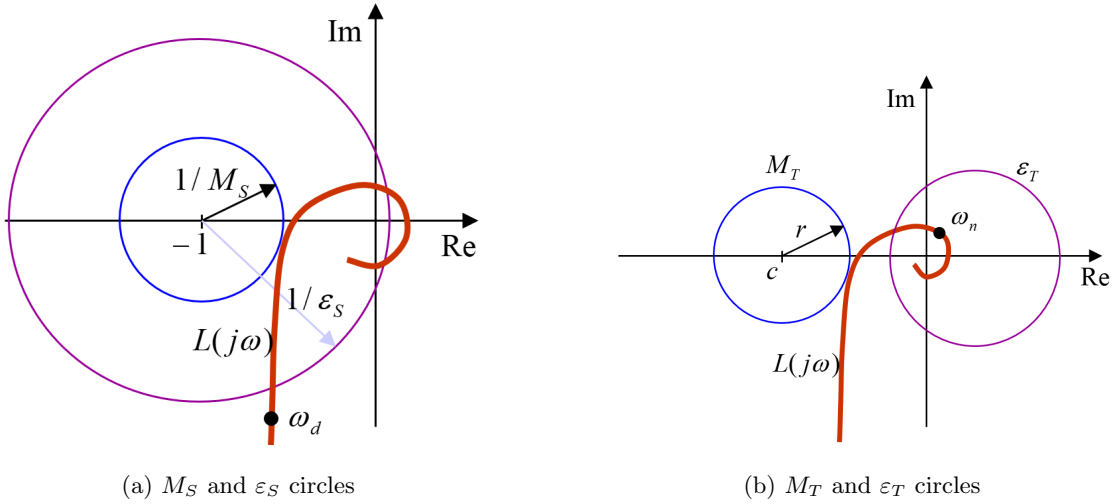


Figure 8: Nyquist shaping according to the M -circles and ε -circles [3]

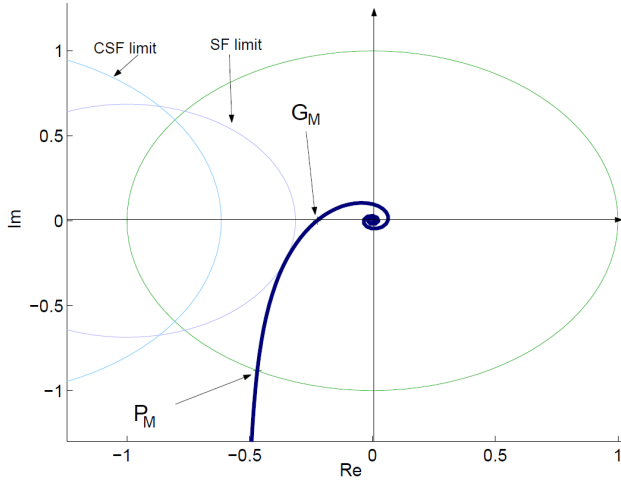


Figure 9: Nyquist plot shaping [3]

low-frequency disturbance rejection is equal to the equation (19). To satisfy this requirement, the Nyquist curve has to be outside of the ε_S -circle until its frequency reaches the ω_S frequency (Figure 8a). The bandwidth requirement corresponds with the equation (22). It is met when the Nyquist curve is inside the ε_T -circle from the moment when the frequency of $L(j\omega)$ equals ω_T (Figure 8b).

Shaping points representing GM and PM are special case of the general shaping points in the complex plane. General shaping point can be put 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. However we can select any general shaping point. The main reason why we select the shaping point X is that from its real and imaginary coordinate together with the real and imaginary parts of the controlled process we are able to compute every possible controller parameters which ensure that the shaping point X lies on the right of the passing Nyquist curve (for increasing frequency ω). In the next subsection, the PI controller parameters have been derived.

According to the Nyquist stability criterion, the closed loop system is stable when the Nyquist curve passes on the right side of the the critical point $[-1, j0]$. If the open loop is stable, then all of the frequency domain design criteria mentioned in table (3) can be expressed as shaping points in the complex plane which represent limits of the Nyquist curve $L(j\omega) = C(j\omega)P(j\omega)$ (Figure 9). In order to satisfy GM and PM, the Nyquist curve has to be placed on the right 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 11) with increasing frequency. The

2.2.2 PI robust stability regions

The final form of the PI controller parameter expression appears in M. Čech's dissertation [3]. The detailed step by step procedure of derivation will be described in this section as an extension of what is stated in the dissertation.

Let $P(s) = \frac{num(s)}{den(s)}$ be the transfer function of a stable non-oscillatory process. Let $C(s) = K + \frac{K_i}{s}$ be the PI controller connected to the process, with K and K_i being the controller gains. Let $L(s) = C(s)P(s)$ be the open loop transfer function (Nyquist curve). If we put $s \stackrel{!}{=} j\omega$ we obtain the frequency response of the system $P(j\omega)$ as

$$P(j\omega) = a(\omega) + jb(\omega), \quad (27)$$

where j is the imaginary unit, $a(\omega)$ is real and $b(\omega)$ is complex part of the frequency response. The frequency response of PI controller can be expressed as

$$C(j\omega) = K(\omega) + \frac{K_i(\omega)}{j\omega}. \quad (28)$$

Frequency response $C(j\omega)$ can be normalized by placing the imaginary unit into the numerator:

$$C(j\omega) = K + \frac{K_i(\omega)}{j\omega} = K(\omega) + \frac{K_I(\omega)}{j\omega} \cdot \frac{j}{j} = K(\omega) + \frac{jK_i(\omega)}{j^2\omega} = K(\omega) - \frac{jK_i(\omega)}{\omega}. \quad (29)$$

By putting the frequency responses in the open loop, we get the Nyquist curve given as

$$L(j\omega) = C(j\omega)P(j\omega) = \left(K(\omega) - j\frac{K_i(\omega)}{\omega} \right) \cdot (a(\omega) + jb(\omega)). \quad (30)$$

Nyquist curve can be put equal to an arbitrary shaping point in the complex plane $X = u + jv$. This is shown in the formula

$$L(j\omega) = \left(K(\omega) - j\frac{K_i(\omega)}{\omega} \right) \cdot (a(\omega) + jb(\omega)) = u + jv. \quad (31)$$

Next, we separate the equation into the real and imaginary part as

$$\Re\{L(j\omega)\} : K(\omega)a(\omega) + \frac{K_i(\omega)b(\omega)}{\omega} = u, \quad (32)$$

$$\Im\{L(j\omega)\} : \cancel{K}(\omega)b(\omega) - \cancel{K}\frac{K_i(\omega)a(\omega)}{\omega} = \cancel{K}v. \quad (33)$$

The proportional gain $K(\omega)$ can be expressed from the equation (33) as

$$K(\omega) = \frac{K_i(\omega)a(\omega)}{\omega b(\omega)} + \frac{v}{b(\omega)}. \quad (34)$$

The expression of the parameter $K(\omega)$ can be substituted into the equation (32), and thus the integral gain $K_i(\omega)$ can be expressed as following:

$$\left(\frac{K_i(\omega)a(\omega)}{\omega b(\omega)} + \frac{v}{b(\omega)} \right) \cdot a(\omega) + \frac{K_i b(\omega)}{\omega} = u, \quad (35)$$

$$K_i(\omega) \cdot \left(\frac{a^2(\omega) + b^2(\omega)}{\omega b(\omega)} \right) = u - \frac{va(\omega)}{b(\omega)} / \cdot \left(\frac{\omega b(\omega)}{a^2(\omega) + b^2(\omega)} \right), \quad (36)$$

$$K_i(\omega) = \frac{ub(\omega) - va(\omega)}{\cancel{b(\omega)}} \cdot \frac{\cancel{\omega b(\omega)}}{a^2(\omega) + b^2(\omega)}, \quad (37)$$

$$K_i(\omega) = \frac{\omega(ub(\omega) - va(\omega))}{a^2(\omega) + b^2(\omega)}. \quad (38)$$

Finally, the acquired expression of $K_i(\omega)$ is being substituted into the proportional gain expression $K(\omega)$ from the equation (34):

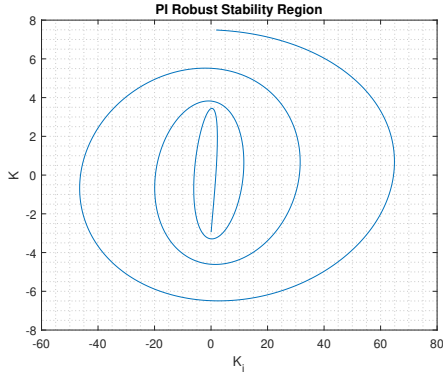
$$\begin{aligned} K(\omega) &= \frac{K_i(\omega)a(\omega)}{\omega b(\omega)} + \frac{v}{b(\omega)} = \frac{\frac{\omega a(\omega) \cdot (ub(\omega) - va(\omega))}{a^2(\omega) + b^2(\omega)} + v\omega}{\omega b(\omega)} = \frac{\frac{a(\omega)b(\omega)u\omega - a^2(\omega)v\omega + a^2(\omega)v\omega + b^2(\omega)v\omega}{a^2(\omega) + b^2(\omega)}}{\omega b(\omega)} = \\ &= \frac{a(\omega)b(\omega)u\omega + b^2(\omega)v\omega}{(a^2(\omega) + b^2(\omega)) \cdot \omega b(\omega)} = \frac{\cancel{\omega b(\omega)} \cdot (ua(\omega) + vb(\omega))}{(a^2(\omega) + b^2(\omega)) \cdot \cancel{\omega b(\omega)}} = \frac{ua(\omega) + vb(\omega)}{a^2(\omega) + b^2(\omega)}. \end{aligned} \quad (39)$$

This way we obtained the general expressions of the $K(\omega)$ and $K_I(\omega)$ parameters of the PI controller given by the formulas

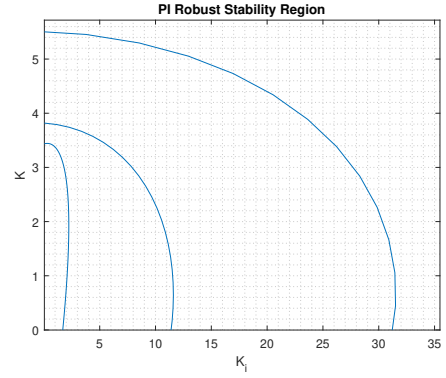
$$K(\omega) = \frac{ua(\omega) + vb(\omega)}{a^2(\omega) + b^2(\omega)}, K > 0, \quad (40)$$

$$K_i(\omega) = \frac{\omega(ub(\omega) - va(\omega))}{a^2(\omega) + b^2(\omega)}, K_i > 0. \quad (41)$$

Obtained expressions (40), (41) define ω parametrized curve in the K, K_i plane with the parameter ω . This curve separates the K, K_i plane into the regions (Figure 10a). Because we defined the parameters K and K_i as a positive numbers, we accept only the regions from the first quadrant (Figure 10b). $K(\omega)$ and $K_i(\omega)$ for stable non-oscillatory processes with dead time create twisted curve where the number of regions increases with the increasing frequency ω . Points in each region encircle the shaping point X the same number of times. Our aim is to find the region which contains the parameter combination ensuring that the Nyquist curve passes on the right of the shaping point X (and thus the design requirement is satisfied). This region is obtained when the $K(\omega), K_i(\omega)$ curve encircles the origin of the K, K_i plane for the first time in the first quadrant (Figure 11a) [4].



(a) Robust PI regions



(b) Regions in the first quadrant

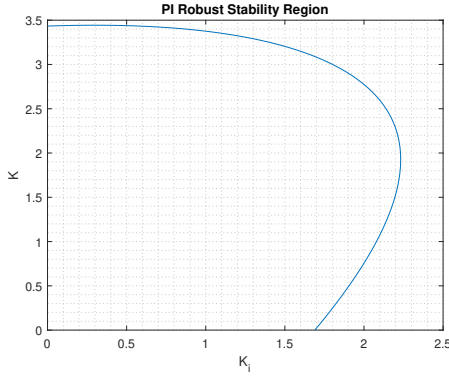
Figure 10: PI robust stability regions for process $P_1(s)$: $u = -0.5, v = -0.5$

The exemplary regions are plotted for two systems $P_1(s)$ and $P_2(s)$ given as

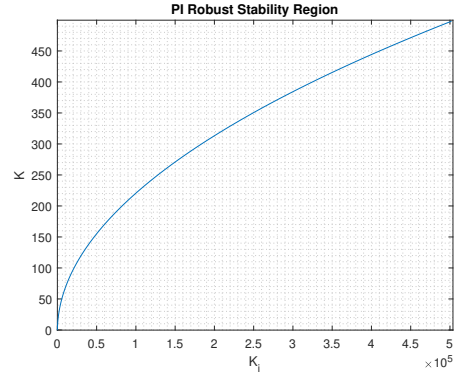
$$P_1(s) = \frac{s+1}{s^2+5s+6}e^{-2s}, \quad (42)$$

$$P_2(s) = \frac{s+1}{s^2+5s+6}, \quad (43)$$

with the design requirement $u = -0.5, v = -0.5$ which is equal to the $PM = 45^\circ$.



(a) Process $P_1(s)$ with time delay



(b) Process $P_2(s)$ without time delay

Figure 11: PI robust stability regions for simple processes $P_1(s), P_2(s)$: $u = -0.5, v = -0.5$

2.2.3 PI^α robust stability regions

In this section we focus on introducing the mathematics behind the PI^α control algorithm which was used for the computing of the multi-dimensional robust stability regions and its visualization. The PI^α controller represents a formal extension of the PI controller. The expressions for PI^α controller parameters can be obtained similarly. Let $P(s) = \frac{num(s)}{den(s)}$ be stable non-oscillatory process. Let $P(j\omega) = a(\omega) + jb(\omega)$ be the frequency response of the given process. Suppose PI^α controller connected to the process in the form

$$C(s) = K + \frac{K_i}{s^\alpha}, \quad (44)$$

where $\alpha \in \mathbb{R}^+$ is the parameter representing controllers fractional order. It has a clear interpretation. If we think of PI^α controller as a band pass filter, then the α parameter influences the steepness of the filter from both sides of the frequency spectrum [3].

The Nyquist curve is obtained by creating the open loop from frequency responses of the process and controller. Again, the Nyquist loop can be shaped by general shaping point in the complex plane $X = u + jv$ which can be expressed as

$$L(j\omega) = C(j\omega)P(j\omega) = \left(K(\omega) + \frac{K_i(\omega)}{(j\omega)^\alpha} \right) \cdot (a(\omega) + jb(\omega)) = u + jv. \quad (45)$$

From the equation (45), the expressions for PI^α controller gains K and K_i parametrized by ω are computed as

$$K(\omega) = \frac{a(\omega)v \cos \phi + b(\omega)v \sin \phi + a(\omega)u \sin \phi - b(\omega)u \cos \phi}{(a^2(\omega) + b^2(\omega)) \sin \phi}, \quad K > 0, \quad (46)$$

$$K_i(\omega) = \frac{\omega^\alpha (ub(\omega) - va(\omega))}{(a^2(\omega) + b^2(\omega)) \sin \phi}, \quad K_i > 0, \quad (47)$$

where $\phi = \frac{1}{2}\alpha\pi$ [3]. Obtained expressions define an area in the K, K_i, α space. The area represents the robust stability regions. Multiple regions in all four quadrants can be seen on the Figure 12. Again, we would concentrate on the first quadrant, because of the conditions $K, K_i, \alpha > 0$. On the Figure 13 is the regions first encirclement of the origin. For the demonstrative example, transfer function $P_3(s)$ was used given by the formula

$$P_3(s) = \frac{s+1}{s^2+5s+6} e^{-0.2s}. \quad (48)$$

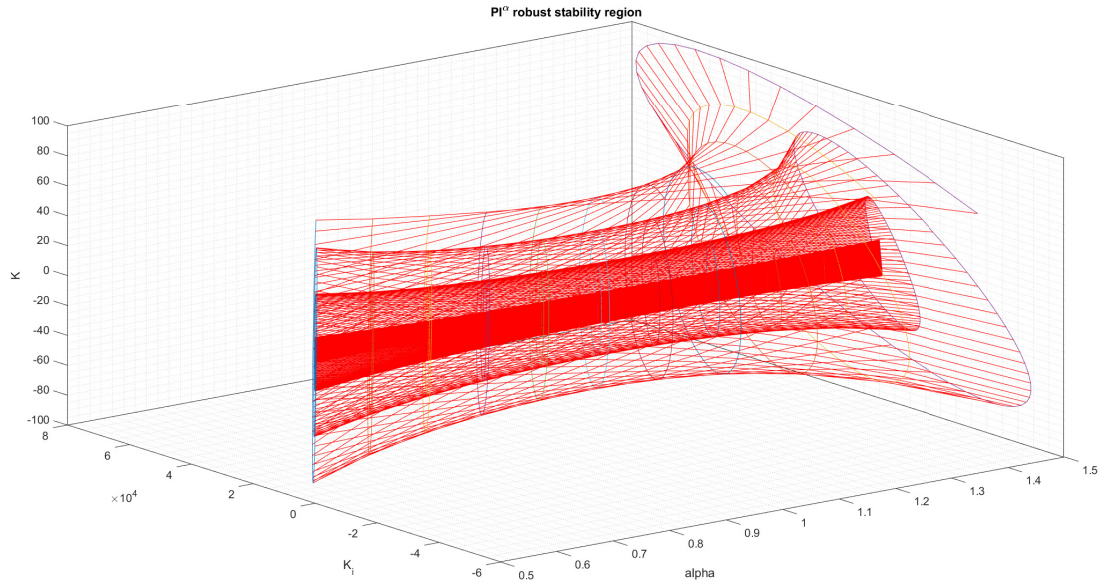


Figure 12: PI^α robust stability region plot: process $P_3(s)$ (48), shaping point: $[u, v] = [-0.5, -0.5]$, $\alpha = [0.5, 0.6, 0.7, \dots, 1.5]$, $\omega = [10^{-1}, \dots, 10^2]$, 1000 frequency samples

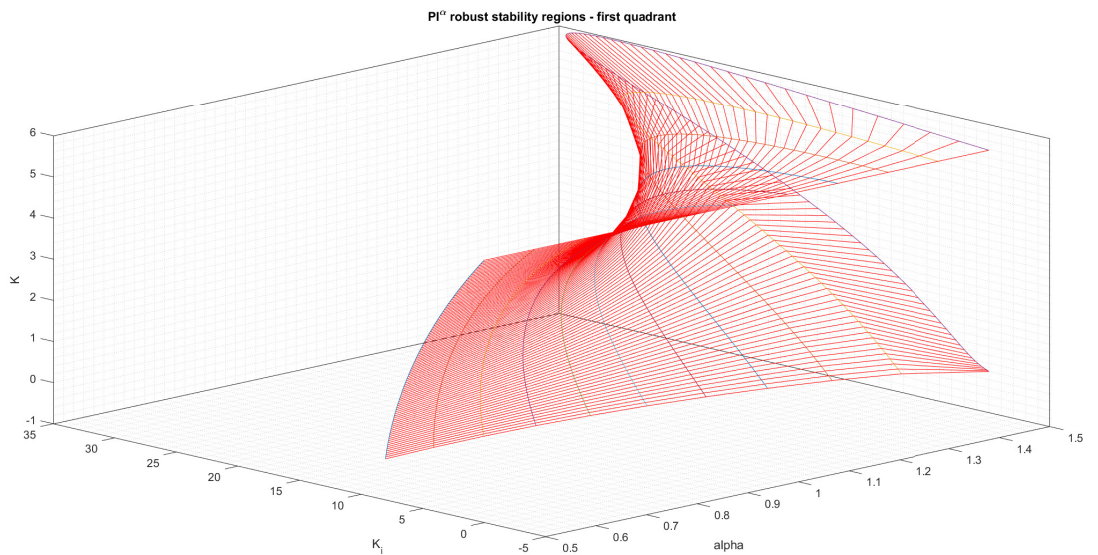


Figure 13: First quadrant of the PI^α robust stability region plot: process $P_3(s)$ (48), shaping point: $[u, v] = [-0.5, -0.5]$, $\alpha = [0.5, 0.6, 0.7, \dots, 1.5]$, $\omega = [10^{-1}, \dots, 10^2]$, 100 frequency samples

The two-dimensional projection of the PI^α regions appears on the Figure 14. *Note: When the $\alpha = 1$, the PI^α and PI controller forms are equal.*

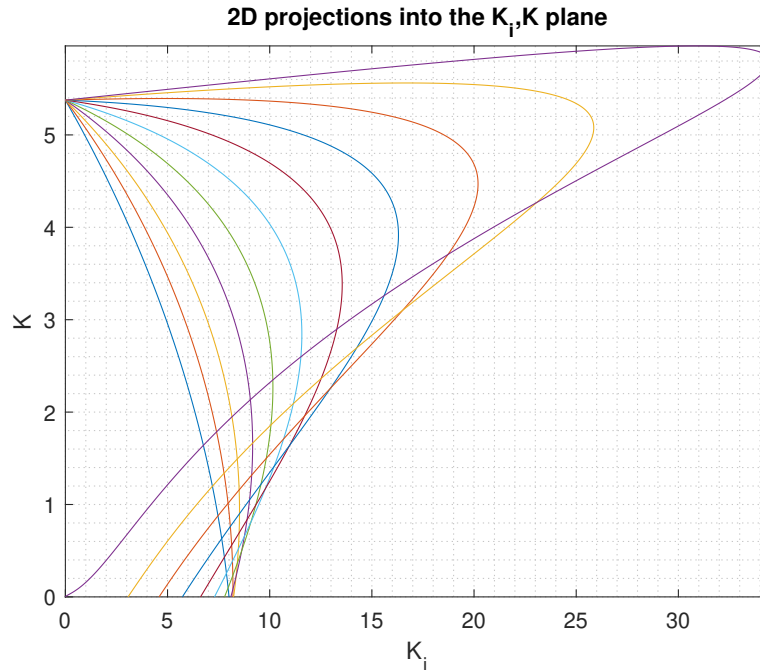


Figure 14: 2D projection of the PI^α robust stability region into the K_i, K plane plot: process $P_3(s)$ (48), shaping point: $[u, v] = [-0.5, -0.5]$, $\alpha = [0.5, 0.6, 0.7, \dots, 1.5]$, $\omega = [10^{-1}, \dots, 10^2]$, 1000 frequency samples

2.2.4 Moment-model Set

The large number of real processes has monotone behaviour in the time domain. In the following sections, we will show how the monotony of these processes in frequency and time domain is related.

The majority of *essentially monotone processes* (Åström and Häglund (2006)) can be described by the formula

$$P(s) = \frac{K}{\prod_{i=1}^p (\tau_i s + 1)^{n_i}}, \quad (49)$$

where p is arbitrary integer number and $K, \tau_i, n_i, i = 1, 2, \dots, p$ are positive real numbers. The equation (49) also contains the systems with dead-time [9][10].

In the practice, the process can be described by the characteristic numbers $\{\kappa, \mu, \sigma^2\}$ which can be computed from the first three impulse response moments [6]. The impulse response moments are defined as

$$m_i = \int_0^\infty t^i h(t) dt, \quad i = 0, 1, 2. \quad (50)$$

The characteristic numbers $\{\kappa, \mu, \sigma^2\}$ are given as

$$\kappa = \int_0^{\infty} h(t)dt = m_0, \quad (51)$$

$$\mu = \frac{\int_0^{\infty} th(t)dt}{\int_0^{\infty} h(t)dt} = \frac{m_1}{m_0}, \quad (52)$$

$$\sigma^2 = \frac{\int_0^{\infty} (t - \mu)^2 h(t)dt}{\int_0^{\infty} h(t)dt} = \frac{m_2}{m_0} - \frac{m_1^2}{m_0^2}. \quad (53)$$

From the characteristic numbers, we are able to obtain a Moment-model set. Moment-model set can be given as $\mathcal{S}^{n,m}(\kappa, \mu, \sigma^2)$, where n is the total order of the process, and m is the minimum allowed order of each fractional pole [5]. In the frequency domain, the model set processes create a connected area called value set. The value set boundary is generated by so called extremal transfer functions (Figure 15).

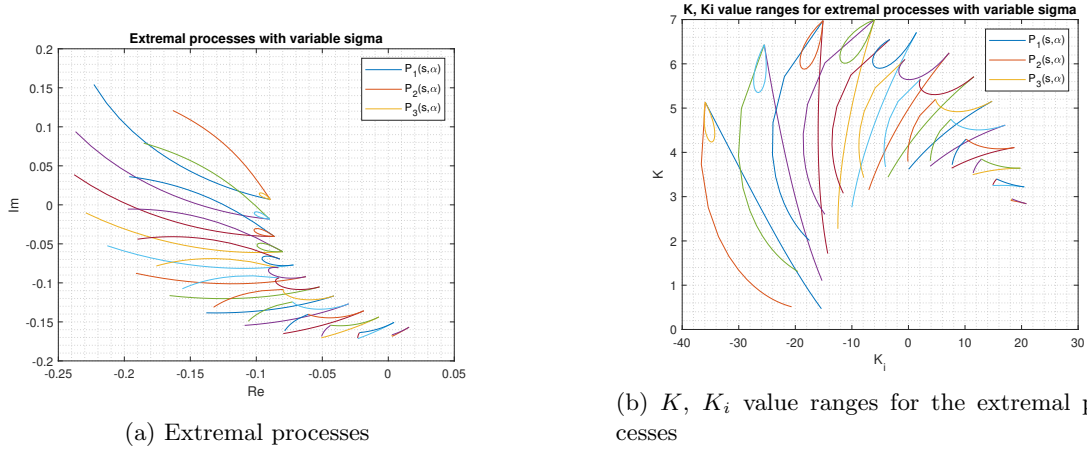


Figure 15: Generating extremal processes in Matlab: $\omega = 6$, $\sigma^2 = [0.35, 0.40, 0.45, \dots, 0.95]$, $n = 10$, $m = 1$

In the following sections, we will show that the magnitude $A(\omega)$ and phase $\varphi(\omega)$ of any arbitrary process belonging into the process set given by the equation (49) are monotonous functions, and then we will show that the monotony in the frequency domain is transmissible into the time domain.

2.2.5 Introduction to the proof of the monotony of $A(\omega)$ and $\varphi(\omega)$

As it has been said earlier, huge number of real processes can be described by the formula (49). Let us take an example of a transfer function which meets the criteria given by this formula as

$$P_1(s) = \frac{K_1}{(\tau_1 s + 1)}. \quad (54)$$

Formula (54) can be expressed as a product of similar transfer functions $P_i(s)$, $i = 1, \dots, N$, where N is the number of the given processes. Transfer functions $P_i(s)$ may differ in constants K and τ and in the exponent of the denominator.

For each process $P_i(s)$ we are able to determine real and imaginary part in the frequency domain as

$$P_i(j\omega) = a_i(\omega) + j \cdot b_i(\omega), \quad (55)$$

where $a_i(\omega)$ is real part of the process, $b_i(\omega)$ is imaginary part of the process, j is imaginary unit and ω is frequency, $[\omega] = \text{rad} \cdot \text{s}^{-1}$.

Real and imaginary parts of the simplest example $P_1(s)$ (Formula (54)) are given as

$$a_i(\omega) = \frac{K_i}{\tau_i^2 \omega^2 + 1}, \quad (56)$$

$$b_i(\omega) = \frac{-K_i \tau_i \omega}{\tau_i^2 \omega^2 + 1}. \quad (57)$$

From real and imaginary part of the process we can easily obtain magnitude $A(\omega)$ as

$$A(\omega) = \sqrt{a^2(\omega) + b^2(\omega)}. \quad (58)$$

For $i = 1, \dots, n$, $n \in \mathbb{N}$ we can obtain magnitudes from formulas

$$A_i(\omega) = \frac{K_i}{\sqrt{(1 + \tau_i^2 \omega^2)}}, \quad (59)$$

$$\frac{dA_i(\omega)}{d\omega} = \frac{-K_i \tau_i^2 \omega}{(1 + \tau_i^2 \omega^2)^{\frac{3}{2}}}. \quad (60)$$

Similarly we can obtain phase from formulas

$$\varphi_i(\omega) = \arctan(-\tau_i \omega), \quad (61)$$

$$\frac{d\varphi_i(\omega)}{d\omega} = -\frac{\tau_i}{(\tau_i \omega^2 + 1)}. \quad (62)$$

Whereas $K_i, \tau_i \in \mathbb{R}^+$, then the derivatives $\frac{dA_i(\omega)}{d\omega}$ and $\frac{d\varphi_i(\omega)}{d\omega}$ will always be negative on the domain $D = (0; +\infty)$. This means that $A_i(\omega)$ and $\varphi_i(\omega)$ are monotonuous decreasing function $\forall \omega \in D$.

2.2.6 Proof of the monotony of $A(\omega)$ and $\varphi(\omega)$

Let $A_1(\omega), A_2(\omega)$ be monotonuous decreasing functions on the domain $D = (0; +\infty)$. Then for all $\omega_1, \omega_2 \in D$ with $\omega_1 \leq \omega_2$ we know that $A_1(\omega_1) \geq A_1(\omega_2)$ and $A_2(\omega_1) \geq A_2(\omega_2)$. If we multiply those inequalities, we get $A_1(\omega_1) \cdot A_2(\omega_1) \geq A_1(\omega_2) \cdot A_2(\omega_2)$ which is equivalent to $(A_1 \cdot A_2)(\omega_1) \geq (A_1 \cdot A_2)(\omega_2)$.

This proof can be extended to multiplication of n monotonuous decreasing functions. Let $A_1(\omega), A_2(\omega), \dots, A_n(\omega)$ where $n \in \mathbb{N}$ be monotonuous decreasing functions on the domain $D =$

$(0; +\infty)$. Then for all $\omega_1, \omega_2 \in D$ with $\omega_1 \leq \omega_2$ we know that $A_1(\omega_1) \geq A_1(\omega_2)$, $A_2(\omega_1) \geq A_2(\omega_2)$, \dots , $A_n(\omega_1) \geq A_n(\omega_2)$. If we multiply those inequalities, we get $A_1(\omega_1) \cdot A_2(\omega_1) \cdot \dots \cdot A_n(\omega_1) \geq A_1(\omega_2) \cdot A_2(\omega_2) \cdot \dots \cdot A_n(\omega_2)$ which is equivalent to $(A_1 \cdot A_2 \cdot \dots \cdot A_n)(\omega_1) \geq (A_1 \cdot A_2 \cdot \dots \cdot A_n)(\omega_2)$.

Let us prove that for $n + 1$ monotonuous decreasing functions the product still will be monotonuous decreasing function. Let us assume, that previous statement works up to n monotonous decreasing functions, if we prove, that it also works for $n + 1$ monotonous decreasing functions, we will have proven that it works for any number of monotonous decreasing functions.

Let $A_{n+1}(\omega)$ be monotonuous decreasing function on the domain $D = (0; +\infty)$. Let $A_N(\omega)$ be the product of n monotonuous decreasing functions: $A_N(\omega) = A_1(\omega) \cdot A_2(\omega) \cdot \dots \cdot A_n(\omega) = (A_1 \cdot A_2 \cdot \dots \cdot A_n)(\omega)$. Then for all $\omega_1, \omega_2 \in D$ with $\omega_1 \leq \omega_2$ we know that $A_N(\omega_1) \geq A_N(\omega_2)$ and $A_{n+1}(\omega_1) \geq A_{n+1}(\omega_2)$. If we multiply those inequalities, we get $A_N(\omega_1) \cdot A_{n+1}(\omega_1) \geq A_N(\omega_2) \cdot A_{n+1}(\omega_2)$ which is equivalent to $(A_N \cdot A_{n+1})(\omega_1) \geq (A_N \cdot A_{n+1})(\omega_2)$.

The result $A_R(\omega) = A_N(\omega) \cdot A_{n+1}(\omega) = (A_N \cdot A_{n+1})(\omega)$ will be yet again monotonuous decreasing function.

The proof is similar for the phase φ .

2.2.7 Conclusion of the proof

In the previous section, we have proven that the magnitude $A(\omega)$ and the phase $\varphi(\omega)$ of the process set defined by the equation (49) are monotonous functions on the domain $D = (0; +\infty)$. It can be shown that if we apply the Parseval's theorem [8], the monotony in the frequency domain transitions into the monotony in the time domain. Parseval's theorem is given by the formula

$$\int_{-\infty}^{\infty} |x(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(\omega)|^2 d\omega, \quad (63)$$

where $x(t)$ is the impulse response of a process, and the $X(\omega)$ is the continuous Fourier transform of the impulse response $x(t)$. This implies that the time response of the system described by the equation (49) is also monotonous which proves the essential monotonic assumption. The monotony of the impulse response of the process could be helpful for robust stability region computation. The regions computed from the monotonic processes have smooth shape which would be beneficial for the intersection region computation. This would make the systems from the process set (49) potentially suitable testing set for the robust stability region controller design method.

3 Analysis of existing tuning tools

In this section, we have focused on the current state-of-the-art of the online controller tuning tools and Matlab controller tuning tools. The research was made, and few controller designing tools have been described. In the following subsections, the structure and functionality of each selected tuning tool is analysed.

3.1 Online controller tuning tools

This section is focused on already existing online controller tuning methods. The main objective was to analyse how these methods operate in time and frequency domain, which systems are they able to control, what type of input data they require and what types of controllers do they enable to design – whether it is PI, PD or PID controller with one or two degrees of freedom or perhaps LQR controller etc. Several methods of tuning PID controllers for satisfactory behavior are used in practice. Most of them are only semi-empirical methods. There exist very few really systematic procedures applicable for more complex systems as non-minimum-phase systems, unstable systems and systems with significant time delay [22]. Next question which comes with more complex controller design analysis is whether these tools enable multi-criteria optimization and if they do so by implementing the robust stability regions, and furthermore whether these tools are able to operate with fractional order transfer functions or if they allow fractional order controllers design. It was also important to find out whether these methods are up to date.

In terms of frequency domain, it would be important to find out whether these methods allow us to design the controller by shaping the Nyquist or Bode plot of the closed loop or each sensitivity function and thus implement stability margins such as Gain margin, Phase margin or Stability margin for a closed loop, or whether it is possible to exactly specify these frequency domain requirements. Most of methods freely accessible on the Internet rely more on controller tuning in the time domain than in the frequency domain. However, by neglecting the frequency domain we lose the ability to develop robust controller and thus integrate systems uncertainty. We have no other choice but to deal with the time domain where we often apply heuristic approach in order to tune the controller. Moreover, majority of the currently used sophisticated methods do not allow to effectively define several design requirements at the same time. In addition, the methods often work only with one nominal model of the controlled process [23]. Nominal process is the process that is considered to be the model of the real system. It is usually given in the form of transfer function. However the reality is more complex, nonlinear etc. and it is impossible to cover it by one transfer function. Frequency response of a system at given frequency ω is uncertain, and could be expressed as a range of values in the complex plane. So in order to design a robust controller it is optimal to generate set of N testing processes. If we are able to design the controller for all of these N testing processes we get the robust controller which is more likely to be used on the real plant.

In order to analyse currently used online methods We made a research and pointed out three online tools - PIDlab, Sysquake and PID Tuner. All of those tools are freely available on the internet. In the next subsections the functionalities and interfaces of these tools are described. We also pointed out the advantages and disadvantages of the mentioned methods.

3.1.1 Sysquake

First tool we want to mention is Sysquake which is an innovative, powerful and flexible software for understanding systems, solving problems, and designing products. What makes it unique is its unparalleled graphical interactivity. The GUI is very simple and intuitive. Multiple level Undo permits the user to experiment without the fear of losing performances obtained so far. The graphical interactive abilities of this tool are the main aspect why Sysquake was chosen to be included in the thesis. Through it, the user can see how the controlled process behaves in the frequency

and time domain in an intuitive and understandable form. Moreover the displaying methods have been implemented very efficiently in native machine code. The mathematical interpreter itself is very fast in a way that the figures are updated nearly instantaneously. Through its interactive graphical interface Sysquake shows different ways to represent a controlled system. It allows user to shape the Nyquist plot, move poles and zeros in the Root Locus, drag the Bode plot and thus change systems gain, bandwidth, GM, PM etc. and many more. This aids in understanding how quantities are related to each other. This way we can observe how the system responds to our manipulations with different domains and plots. Sysquake comes with a rich set of applications for a broad range of areas, including: Automatic control, Analog and digital filters, Identification of model parameters and model validation, Robotics, Statistics, Physics, Demography, Finance [11].

In order to design a controller in Sysquake we can use several applications in the form of SQ files. SQ file is a Sysquake module with specific interface and functionality. Or instead of using already existing SQ files we can build our own tuning graphical interface in Sysquakes Application Builder which allows the creation of stand-alone applications.

First mentioned tuning interface is PID_ct.sq (Figure 16) where continuous-time PID controller can be tuned. Note: The equivalent module PID_dt.sq exists for discrete-time PID controller tuning.

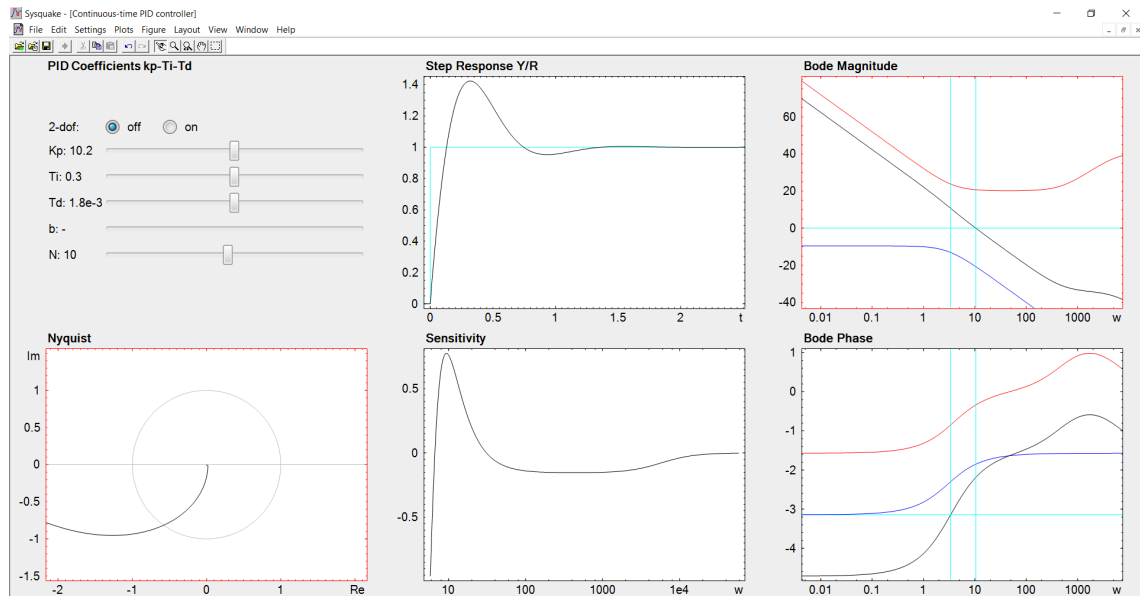


Figure 16: Continuous-time PID controller – Sysquake

In this example, the input transfer function is given as $P(s) = \frac{1}{s-3}$ which represents an unstable first order system. The interface consists of parameters of designed PID controller K_p , T_i , T_d , b (see in section 2.1) and N which are located in the upper left corner and can be changed by moving the buttons from left to right. There is also option to switch to 2DOF version of PID. Next to the parameters section there are several plots - Nyquist plot, Step response, Sensitivity function and Bode Amplitude and Phase plot. After changing the PID parameters Sysquake immediately recomputes all plots. However the Nyquist and Bode Amplitude plot can be shaped by mouse clicking and dragging the closed loop function and thus the PID parameters and other plots are recomputed as well. In the Bode plot there are turquoise lines which help designer to determine GM and PM from Amplitude and Phase plot. Through its interactive graphical interface we can observe the effect of parameter change especially on Nyquist and Bode plot, and vice versa. By shaping Nyquist plot and dragging Bode Amplitude plot we are able to get required values of GM and PM. However we are not able to define these stability margins and get the shape of Nyquist

and Bode plot as an output.

Next module we would like to introduce is PIDBasics (Figure 17). The module gives time and frequency domain views of the responses of a closed-loop system consisting of a PID controller and a process model. Many process models can be selected separately, controller parameters can be changed interactively, and the resulting responses are displayed instantaneously [11]. Set of process transfer functions (Figure 18) matches the process test batch used by K. J. Åström and T. Häglund [24]. Many of the process control systems can be represented by these processes.

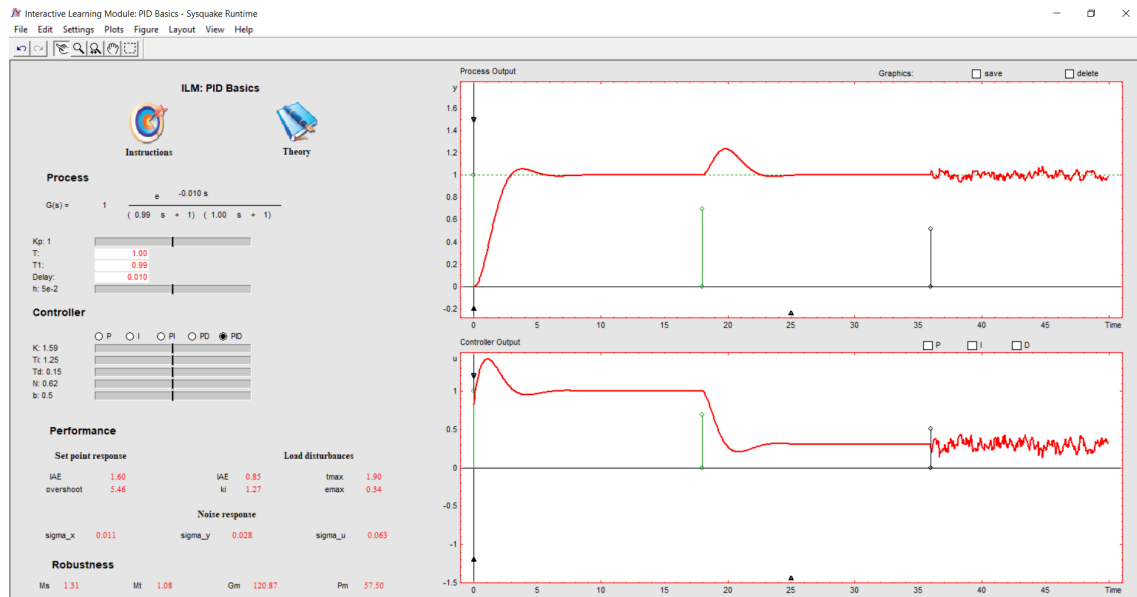


Figure 17: Interactive learning module: PID Basics – Sysquake

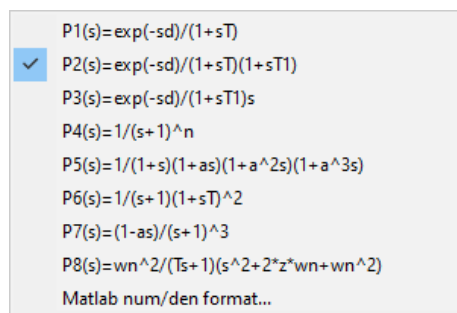


Figure 18: PID Basics Process Transfer Function Set – Sysquake

The input transfer function is $P(s) = \frac{e^{-Ds}}{(T_1s+1)(T_2s+1)}$ which is second order system with time delay. The system parameters can be changed in the Process section on the left side of the application. Under the the Process section there is a Controller section where the type of controller can be selected and its parameters can be tuned by dragging the switch. Under the Controller section there Performance section where set point response, load disturbances and noise response characteristic numbers are being computed. Under the Performance section there is Robustness section where are stability margins GM, PM, and sensitivity function peaks Ms, Mt. On the right side there is a Process and Controller output. In this section we are able to change disturbances by dragging the end of the green line up and down. Similarly we

can change the noise intensity by dragging the end of the blue line. In this module we can easily observe how changes of PID parameters interact with noise and load disturbances, and vice versa. This is great for learning and understanding relations between important quantities. Although there are many options of input processes, there is no possibility to select more processes at a same time. Again the stability margins can not be set, they can only be obtained by controller parameters setting or disturbance and noise shaping.

In the conclusion, Sysquake brings many benefits in terms of interactive graphical user interface

where the impacts of PID tuning are visible immediately in many domains. It provides possibilities to build applications thanks to Application builder. It is a multi-platform, easy to integrate online tool with comprehensive documentation. Latest version of Sysquake was released in December 2019 and thus it is relatively actual method. However it lacks some critical abilities in terms of robust controller design. There is no possibility to define one or more stability margins and thus run multi-criteria optimization. There is no possibility to input more transfer functions at a same time and design a controller that would be able to control multiple systems with multiple design criteria.

3.1.2 PID Tuner

Next tool we would like to mention is PID Tuner. PID Tuner is freely available web application which is executable in browser. PID Tuner relies on systematic procedure consisting of four designing steps. It allows the user to use the step response data from their system and follow the wizard step-by-step. On each step the user is free to make decisions in order to improve tuning, however the default PID Tuner settings often work just fine.

In the first step, the data are imported (Figure 19). Data can be imported as a N -dimensional vector. Input data consist of sampled time vector, input signal vector and output signal vector. On the right side there is a plot of input and output data in time domain.

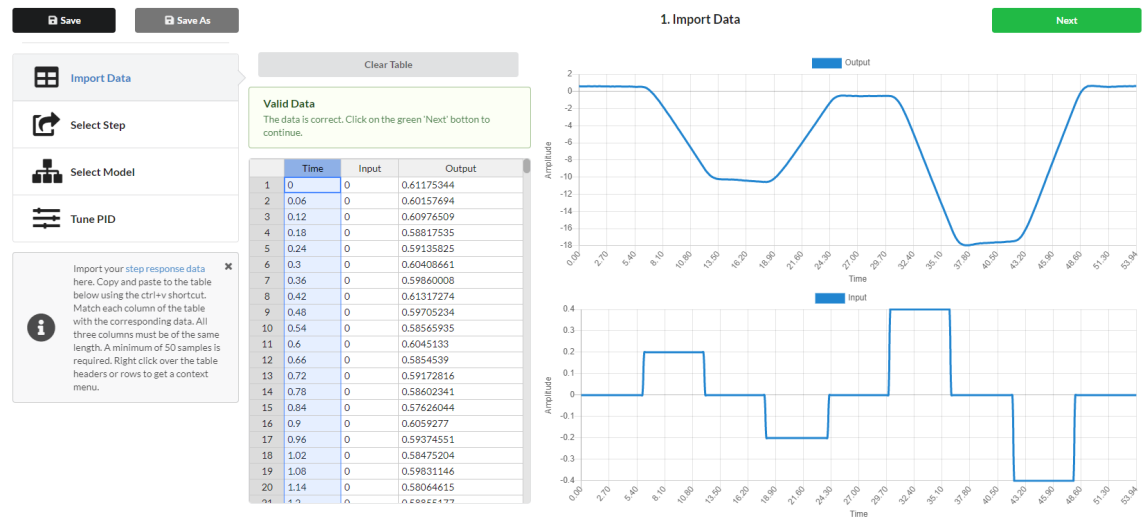


Figure 19: Data import – PID Tuner

In the second step, the user selects time range of the input and output signal to be used for controller tuning (Figure 20). The step response data usually contain multiple steps, but for tuning purposes, only one step is needed. It is optimal to select the smallest time range, where only one step response data is contained. PID Tuner uses pre-computed time ranges, but the user can define custom ranges by dragging the orange vertical lines.



Figure 20: Step select – PID Tuner

In the third step, the model that best fits the process given by step response data is automatically selected (Figure 21). On the left side it is possible to choose different model from identified models. The controller designer should be aware of the laws of physical reality of the system and thus select the type of model which is the best representation of the real system. For this example the integrator with lag is selected as the best fitting model. After the model is selected its parameters can be manually updated. Change is confirmed by pressing enter.

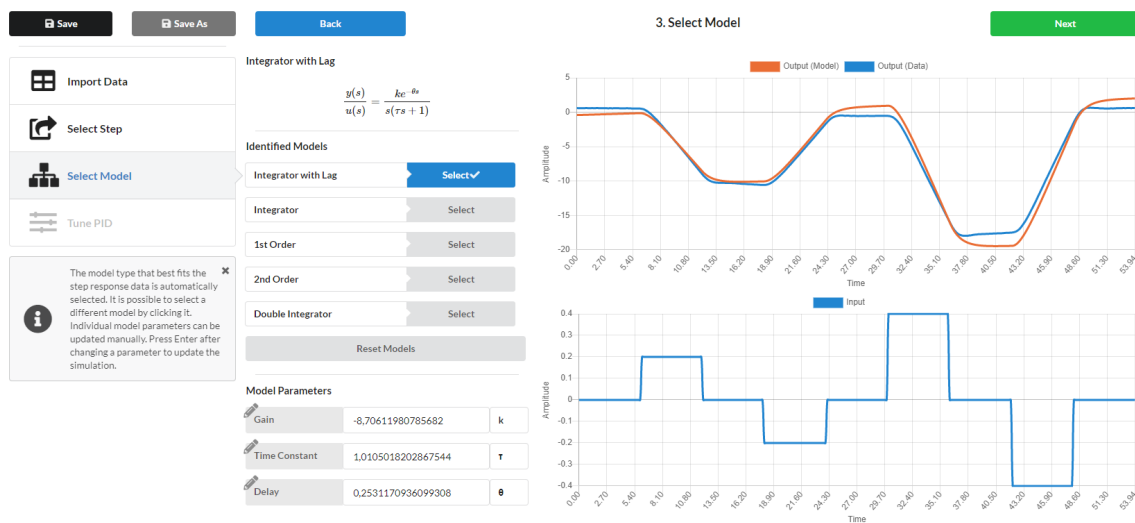


Figure 21: Model select – PID Tuner

The final step focuses on PID tuning. This step is divided into two sections - Time Step (Figure 22) and Bode Plot (Figure 23). At the beginning we get automatically computed PID gains which provide a starting point for PID tuning. It is up to the user whether they decide to change the given gains. Gains can always be reset to initial values, they can be also scaled all at once by dragging the "Scale Gains" button. There is also possibility to change set point and add load disturbances. Every change of parameters which is made is immediately projected into Time Step and Bode Plot graphs.

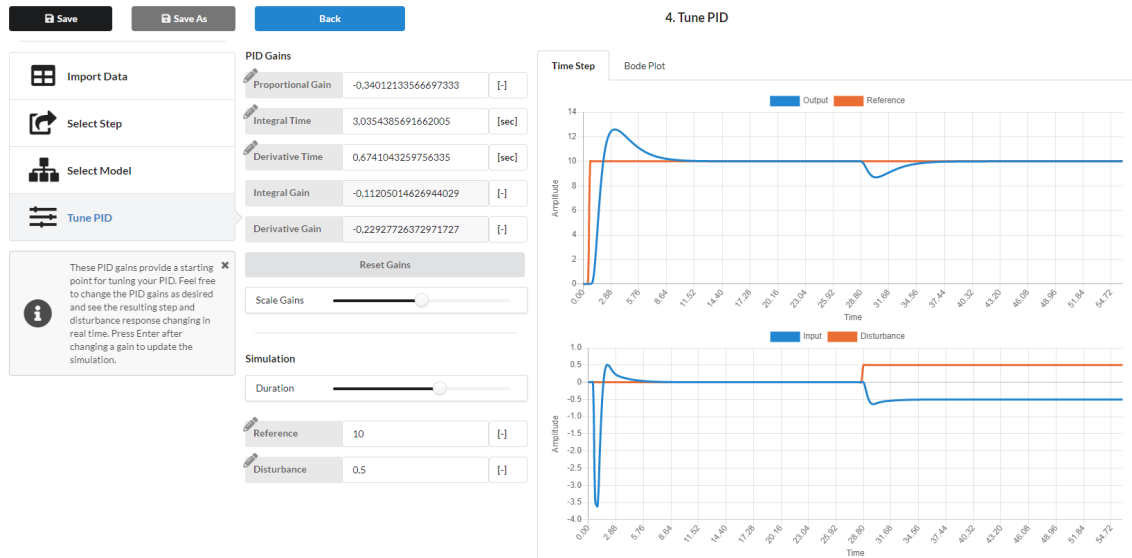


Figure 22: Time step PID tuning – PID Tuner

In the Bode Plot we can see the stability margins which are computed from the frequency domain where they are represented by the orange lines. However we are not able to select and redefine these stability margins.

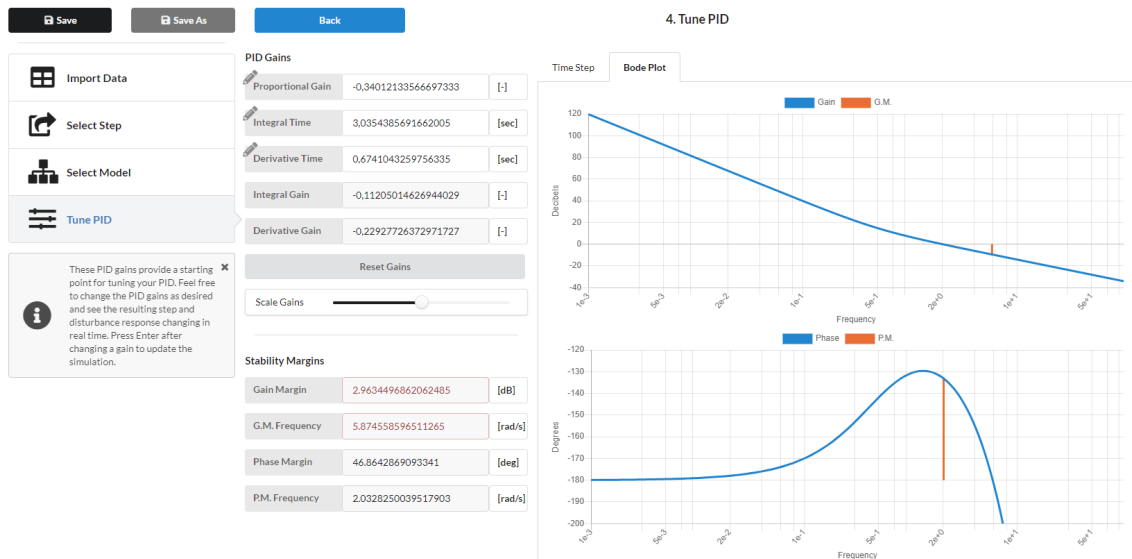


Figure 23: Bode plot PID tuning – PID Tuner

To conclude, PID Tuner has many benefits, such as Copy-paste data import, Data validation and visualization, Multi-model data fit, Auto-tuned PID gains, Interactive closed-loop simulations [12]. It is a browser application, there is no need to download any additional content. PID Tuner has its own Forum with General PID Discussion and PID Tuner Help. Last post on the Forum is from February 4, 2021, meaning that this tool is actual. It has also accessible GitHub repository. However same as Sysquake (3.1.1), the PID Tuner does not allow user to define more systems at the same time and select multiple criteria for each system. It does not have integrated uncertainty and therefore it can not create robust controller. In terms of interactivity, it does not possess as

many interactive features as Sysquake. We have decided to include this tuner in this thesis because of its simplicity, well-arranged graphical design, and a fact that we start with the step response data.

3.1.3 PIDlab

PIDlab is a simple Internet PID controller tuning tool in the form of Java applet. The reason why we included PIDlab in this thesis is that it is based on the robustness regions method, and thus it enables to perform robust controller tuning and design. It allows user to shape the Nyquist plot by selecting multiple general shaping points (including GM and PM) from the complex plane and by sensitivity functions shaping. PIDlab also allows user to define arbitrary number of processes. Thanks to the essential design multi-criteria specification such as stability margins, loop bandwidth, disturbance rejection, sensitivity functions and Nyquist plot shaping we are able to optimize the designed controller. Next to that PIDlab provides necessary procedures for process identification from experimental data, advanced PID feedback loop simulator and can be connected directly to REXYGEN target devices. PIDlab application consists of four sections: Experimental data section, Process section, Controller section and REX section [4].

We start in the Process section where we can add one or more processes in the form of transfer function. The processes can be inserted in various forms such as Bode (contains time constants of numerator and denominator), zeros/poles (contains zeros/poles of the transfer function), num/den (contains coefficients of numerator and denominator in descending power) and SOPDT (second order plus dead time). We can also add gain, transport delay and astaticism into our given process. After inserting a process, its frequency response is plotted in the form of Nyquist plot. In terms of numeric simulation, it is possible to select sample step size. This could solve the problem while plotting the Nyquist plot of the dead time process which is often sparsely sampled when using the default frequency setting. Each inserted process can be edited or deleted later. For the practical example, we used a second order transfer function with time delay given as $P(s) = \frac{e^{-0.2s}}{(0.7s+1)(1.2s+1)}$.

After the process has been defined we move into the Controller section (Figure 24). Here we can select a specific type of controller (PI, PD, PID or FPID) and specify filter coefficient N and the ratio $f = \frac{T_d}{T_i}$ [25]. Then we can define gain margin and phase margin criteria either in the Design specification or by clicking into the complex plane in the Nyquist plot shaping window. There is also possibility to define the values of peaks of sensitivity transfer functions M_s and M_t which influence the radius of m-circles or we can simply drag the m-circles by mouse in the Nyquist plot shaping window. For each inserted stability margin requirement the robust region is plotted in the Robustness Regions window. The resulting region satisfying each stability margin requirement is given by the intersection of all computed regions. The intersection area represents the controller parameters K , K_i from which we can select our controller by mouse click. After the controller parameters are selected PIDlab automatically plots and the Nyquist curve in a way that the selected gain and phase margins same as the circles representing the peaks of sensitivity functions would be positioned on the left side of the Nyquist curve, the sensitivity functions are plotted in the bottom right corner of the Controller section. Controller parameters as well as stability margins and sensitivity functions peaks can be later updated. Controller parameters shown in the Figure 24 have been obtained for: $N = 7$, $f = 0.25$, $PM = 60.951$, $GM = 3.567$, $M_s = 1.532$, $M_t = 1.600$.

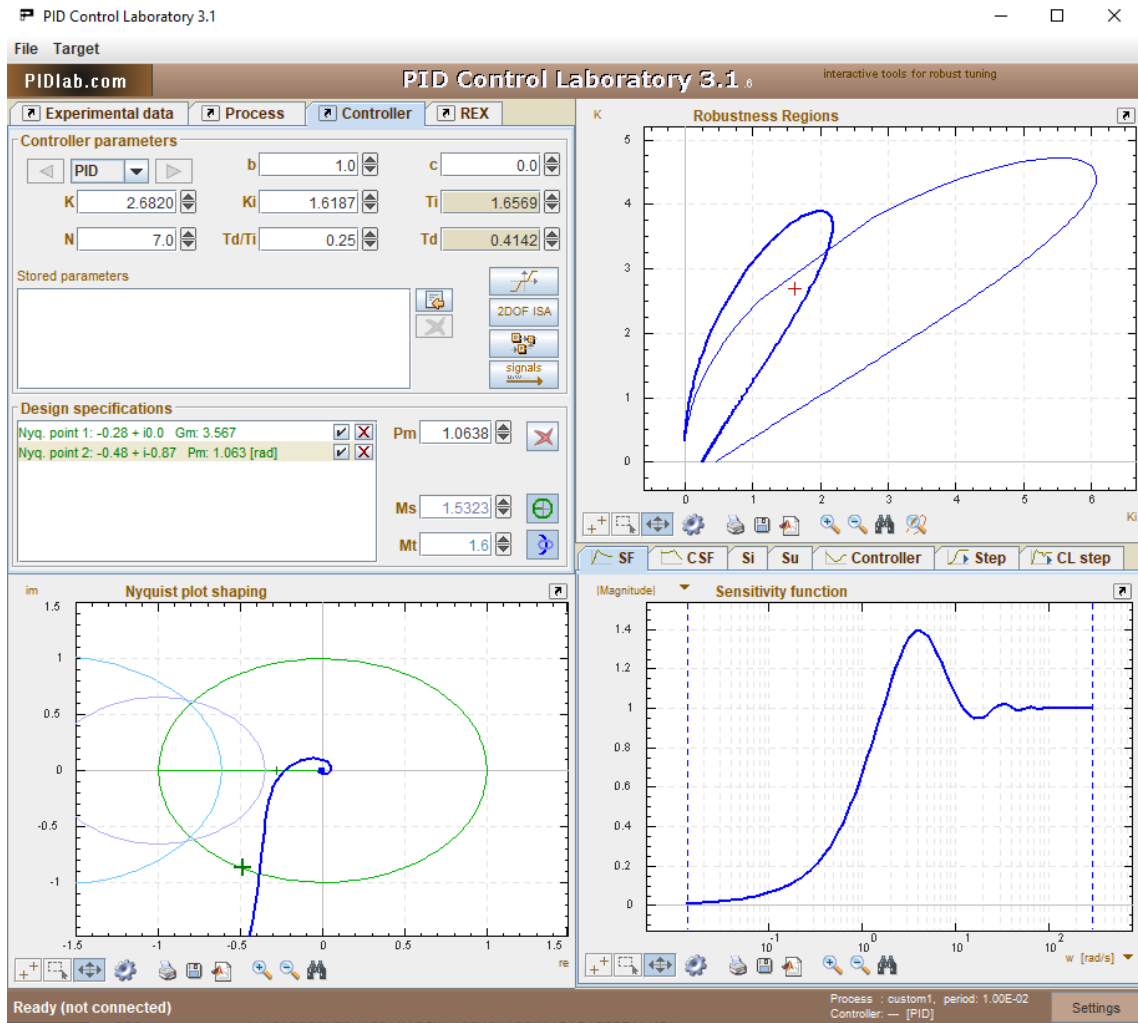
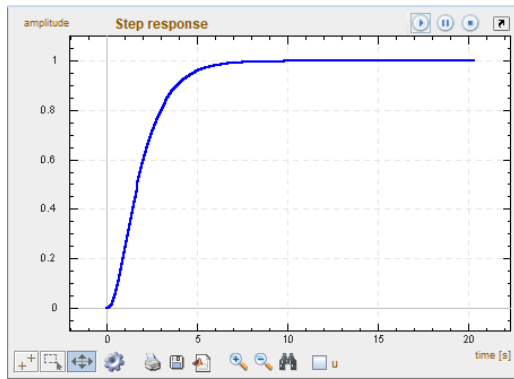
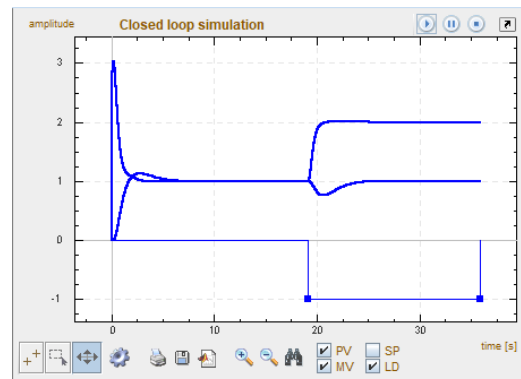


Figure 24: Controller section – PIDlab

We can simulate open and closed loop step response (Figure 32) which can be optimized in the performance menu according to the optimization criteria such as ISE, IAE or ITAE.



(a) Open loop step response



(b) Closed loop step response

Figure 25: Step Responses – PIDlab

In terms of graphical interface options there is possibility to move axis by dragging, use the auto scale button, lock/unlock individual axis by mouse click its labels, change the frequency ranges in plots by mouse and so on. We can change the axes ranges with buttons under each graph. When the button 'auto' is pressed, the ranges of the current plot are set automatically. The best way to zoom is to define the zoom rectangle by mouse dragging. In the lower part of the applet is the settings panel. Here, the ranges for all frequency characteristics can be changed. We can also set the simulation time and the sampling period of discretisation of the process and the controller. The status line can be very useful. Actual applet state information as so as the short help about important components are printed here [25].

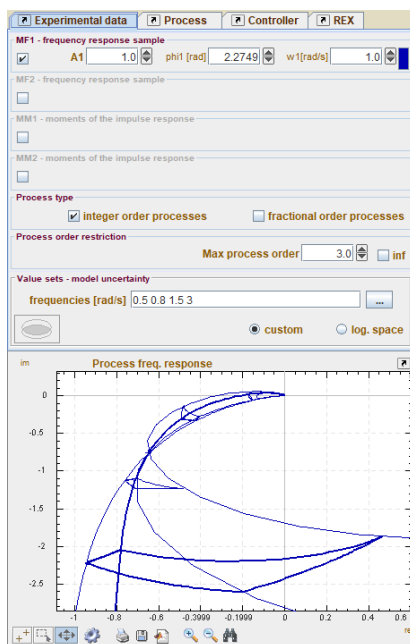


Figure 26: Experimental data section – PIDlab

are being developed. Although PIDlab is suitable primarily for non-oscillatory or slightly oscillatory linear system with dead time, it is possible to define oscillatory linear system design its controller. However there are still some gaps in terms of sensitivity functions and Nyquist plot

If we want to define process from experimental data we can switch to the Experimental data section (Figure 26). There are multiple ways to input measured data, e.g. in the form of moments of the impulse response or by one sample of the frequency response. Then we select fractional or integer order type of model. PIDlab automatically computes and plots uncertainties for specified frequencies and provides value sets which we can use as a bounding processes for robust controller design. The controller designing phase is then similar to previous example.

In the REX section, the designed controller can be connected directly to REX target device and work with PIDU, PIDMA and PIDAT controllers to perform real-time control.

PIDlab does not enable automatic tuning. There is always interaction between designer and the app which is not optimal for novices who lack experience in controller design. The application is monolithic, we can not change the structure of the closed loop. It is important to mention that PIDlab is not actual, however newer versions

shaping. It is an expert task to manipulate 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 $C(s)$ on certain frequency interval 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. This requirement alongside with other mentioned closed loop requirements can often be expressed by the general condition

$$\|H(s, k)\|_\infty < \gamma, \quad (64)$$

where $H(s, k)$ denotes a stable closed-loop-related transfer function, $\|H(s)\|_\infty \triangleq \sup |H(j\omega)|$ and γ is the design parameter. The following performance specifications are considered:

$$\|W_S(s)S(s, k)\|_\infty < \gamma_S, \quad (65)$$

$$\|W_T(s)T(s, k)\|_\infty < \gamma_T, \quad (66)$$

where $S(s, k) = \frac{1}{(1+L(s, k))}$ is the sensitivity function, $T(s, k) = \frac{L(s, k)}{(1+L(s, k))}$ is the complementary sensitivity function, $W_S(s)$ and $W_T(s)$ are assumed to be stable rational functions with no poles on the imaginary axis (these functions are called weighting functions). Note that for our purposes these standard assumptions on the weighted function are not always necessary. For the special case $W_S(s) = W_T(s) = 1$ and $\gamma_S = M_S$, $\gamma_T = M_T$, (65) and (66) are converted to well known conditions (see e.g. [Åström et al. (1998)])

$$\|S(s, k)\|_\infty < M_S, \quad (67)$$

$$\|T(s, k)\|_\infty < M_T. \quad (68)$$

If we choose

$$W_S(\omega) = \begin{cases} 1 & \text{for } \omega \in [0, \omega_S] \\ 0 & \text{otherwise} \end{cases}, \quad (69)$$

and

$$W_T(\omega) = \begin{cases} 1 & \text{for } \omega \in [\omega_T, \infty) \\ 0 & \text{otherwise} \end{cases}, \quad (70)$$

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

$$|S(j\omega)| \leq \epsilon_S, \text{ for } \omega \in [0, \omega_S], \quad (71)$$

and

$$|T(j\omega)| \leq \epsilon_T, \text{ for } \omega \in [\omega_T, \infty), \quad (72)$$

respectively [22]. The implementation of mentioned sensitivity functions criteria would allow us to improve robustness of the designed controller.

In the conclusion, PIDlab is probably the most complete Internet controller tuning tool out of the mentioned tools. It allows user to design integer order and fractional order, 1DOF or 2DOF controllers for multiple design requirements. PIDlab is unique in that we start from the requirements and we finish with the controller set that satisfy these requirements from which we can chose ideal controller. The designed controller can be later optimized by adding more stability margins, thus the multi-criteria optimization is enabled. It is capable of solving disturbance rejection for specified frequency range and the bandwidth specification. It is able to operate with frequency domain in the great depth, it implements uncertainty, and through application of robust region theory it is able to create a robust controller. It is also very graphically interactive tool, almost every parameter can be either exactly numerically defined or it can be obtained from plot shaping. PIDlabs virtual tools can be used for teaching, research and development and commercial purposes [13].

3.2 PID controller tuning methods in Matlab

This section was focused on Matlab controller design methods. The objective was similar as in the previous section (3.1), to analyse the time and frequency domain behavior and tuning ability, the input processes and data possibilities, the interactivity and interface, the variety of controller types these methods provide. In the context of this thesis it would be important to point out if the Matlab methods enable multi-criteria optimization, and whether they implement robust stability region theory. Finally we would analyse whether it is possible to include fractional order processes and/or controller.

Controller tuning methods in Matlab are implemented in Control System Toolbox (CST). This toolbox contains algorithms and apps for systematically analyzing, designing, and tuning linear control systems. The user can specify required continuous or discrete system as a transfer function, state-space, zero-pole-gain, or frequency-response model. Continuous models can be easily discretized. Model of the given system can be further simplified by reducing its order. Through CST we can model systems that are single-input single-output (SISO) or multiple-input multiple-output (MIMO). System models can be transferred to a Matlab graphical programming environment Simulink into block diagrams where they can easily be connected in series, parallel, or feedback. The visualization of system behavior in the time domain and frequency domain is well-arranged. CST provides methods encapsulated in apps, functions and plots which let user to:

- Compute system characteristics such as rise time, overshoot, and settling time and analyze its stability.
- Analyze and visualize system behavior in the time and frequency domains, using step response, impulse response, Bode, Nichols, Nyquist, singular value, and zero-pole plots.
- Inspect characteristics such as rise time, settling time, and maximum overshoot.
- Compute gain margin, phase margin, and crossover frequencies.
- Examine pole and zero locations of dynamic systems graphically and numerically.
- Calculate the damping ratio, natural frequency, and time constant of the poles of a linear model [14].

Apps can be opened in Matlab Apps Tab in Control System Design and Analysis section. List of apps contains: Control System Designer, Control System Tuner, Fuzzy Logic Designer, Linear System Analyzer, Model Reducer, MPC Designer, Neuro-Fuzzy Designer, PID Tuner, System Identification. From the mentioned apps we would like to point out PID Tuner, Control System Designer and Control System Tuner due to their relevancy to the topic of this thesis.

3.2.1 PID Tuner

Matlab app PID Tuner provides functions to automatically tune PID controller gains in order to achieve balance between performance and robustness. It could be opened by clicking its icon in App Tab or by command `pidTuner(plant)`, where `plant` is the transfer function or state-space of our system. We start with defining the plant model in Matlab script. As an example, we used unstable process given as $P(s) = \frac{(s+1)}{(s-2)^2}$. As the PID Tuner opens up (Figure 27), it automatically computes the controller gains. In the left corner we can select required controller type including 1DOF, 2DOF and FO controllers. We can also select Parallel or Standard form. It is possible to tune discrete PID controllers as well. For this example we chose standard PID controller. By clicking Show Parameters icon we can visualize controller, performance and robustness parameters. There are two interactive sliders in the upper middle section of this app – Response Time and Transient Behavior (Time Domain) which is equivalent to Bandwidth and Phase Margin (Frequency Domain). By dragging these sliders we adjust the system performance and robustness. Most of the screen is covered by Step or Bode plots which are automatically recomputed after the sliders have been dragged. It is possible to plot Plant, Open-loop, Reference tracking, Controller effort, Input Disturbance Rejection, and Output Disturbance Rejection Step/Bode plots. Once we are satisfied with the result we can export the PID controller into the Matlab where it will be represented as a `pid` object where it can be further adjusted or transferred to Simulink. For this example we obtained gains: $K = 12.9236$, $K_i = 28.4278$, $K_d = 0.4017$.

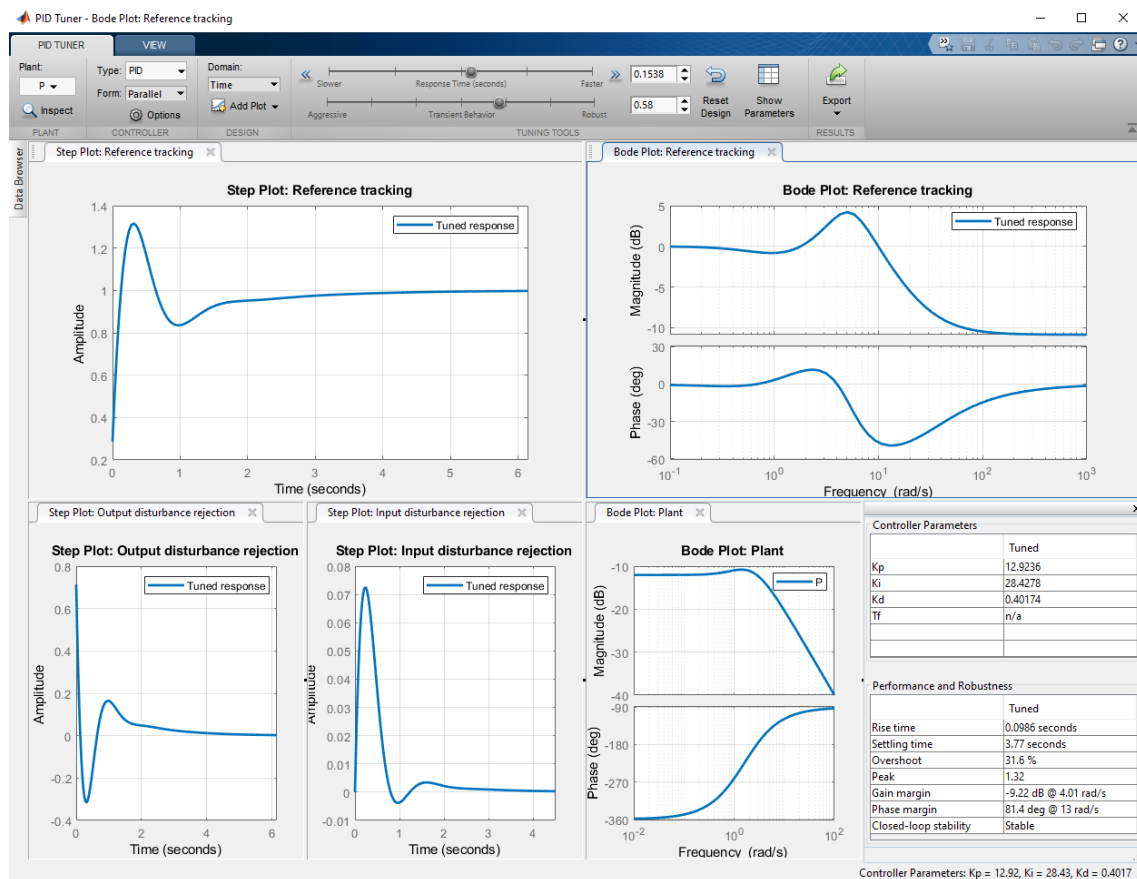


Figure 27: PID Tuner – Matlab Control System Toolbox

To conclude, in this tool we can design the controller performance for optimal set point tracking and disturbance rejection by specifying tuning parameters, such as desired response time and phase margin. One of this tools main benefit is that it is automatic. The initial automatically computed parameter estimation is usually very good. The interface is well-arranged. The dynamic parameter computation is very fast. However despite its interactivity and graphical design, in terms of stability margins we can only define Phase margin by dragging the sliders. The Gain margin can only be estimated from the Bode plot, we can not define it. Any other general shaping point can not be neither defined nor selected from any plot. There is no possibility to shape Nyquist plot, nor view the M-circles. We can only define one process at a time and we can not select more shaping points at a time; thus we can not optimize our controller based on multi-criteria optimization. We are unable to integrate uncertainty into our model. Despite its advantages PID Tuner would probably not be applicable in the practice.

3.2.2 Control System Designer

Control System Designer lets user tune SISO controller using graphical and automated tuning methods. User defines input system in Matlab which can be imported to Control System Designer. It could be opened in App Tab, or similarly as in previous section (3.2.1), by command `controlSystemDesigner(plant)`, where plant is the transfer function or state-space of the given system. As an example we used the same process as in previous section $P(s) = \frac{(s+1)}{(s-2)^2}$. When the app is opened, we can see several plots for graphical tuning (Figure 28). We can choose Bode plot, Nyquist plot, Root Locus plot, Step response and Impulse response of the various signals. In the

Step response, the essential step characteristics can be shown such as Settling Time, Rise Time, or Steady State. In the Edit Architecture tab we can select from different controller architectures including 1DOF, 2DOF, series, or parallel architecture. Here we can import new processes or already existing controllers. In the Tuning Method list we can choose from several PID tuning editors such as Bode, Root Locus or Nichols Editor in case we wanted to use Ziegler–Nichols PID designing method. We focused on the Bode and Root Locus methods. In the Bode plot we can change the gain of our controller by dragging the open loop Bode plot and changing its magnitude. If we add zero or pole to our controller in the Compensator Editor, we can change their value either by dragging them over the curve in Bode or Root Locus Editor or by directly editing their specific values. In the Compensator Editor we can also add integrator, differentiator, lead, or lag compensator. The complete structure of the designed controller can be found in the downer left corner. For the example we obtained controller in the form $C(s) = 38.707 \frac{(s+0.0125)(s+111.8)}{s(s+29.29)}$. This is the result of manual PID tuning.

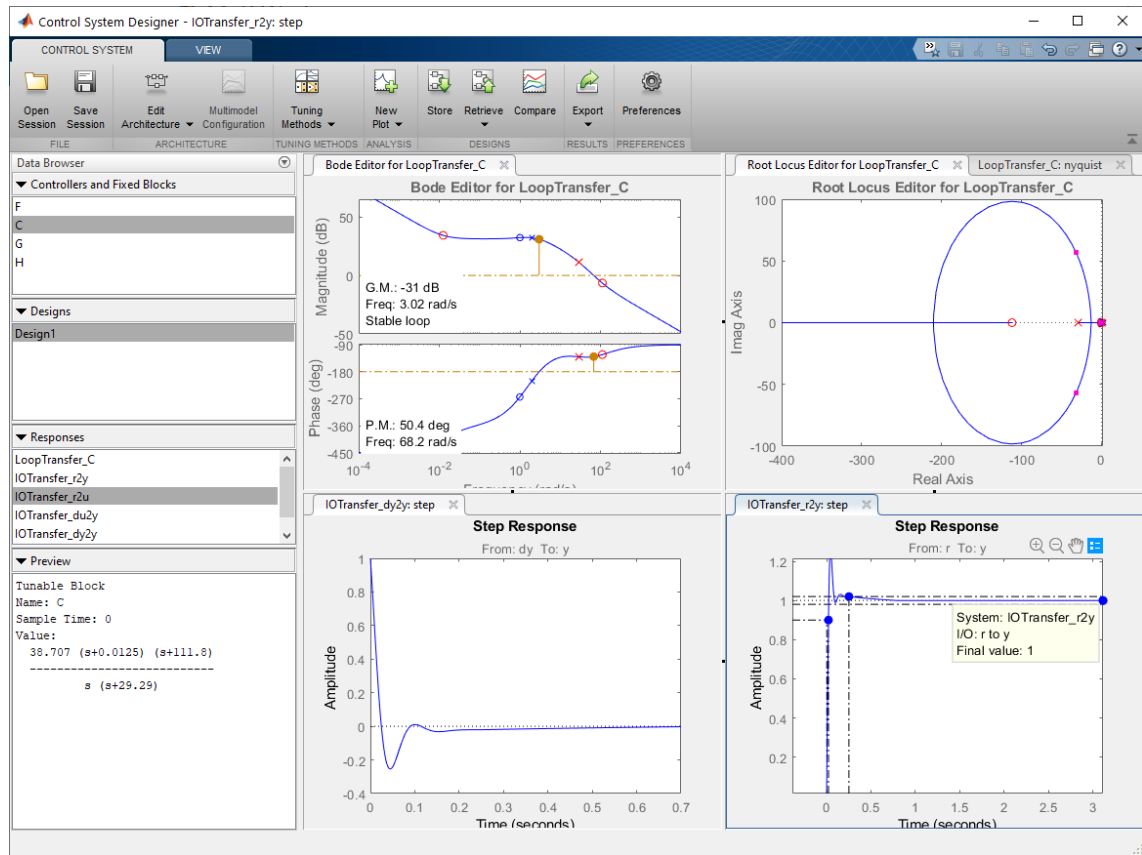


Figure 28: Control System Designer – Matlab Control System Toolbox

We might not be satisfied with the reached GM and PM which can not be specified in Bode Editor. In this case we can switch to automatic PID tuning methods. In the automated PID tuner we receive automatically tuned PID controller which can be further optimized in the Optimization Based Tuning window (Figure 29). Here we can select multiple design requirements including Bode magnitude upper and downer limits, GM and PM limits, Closed-Loop peak gain, Damping ratio, Natural frequency and Settling time. In the Loop Shaping window we can select the maximal bandwidth and optimize our controller.

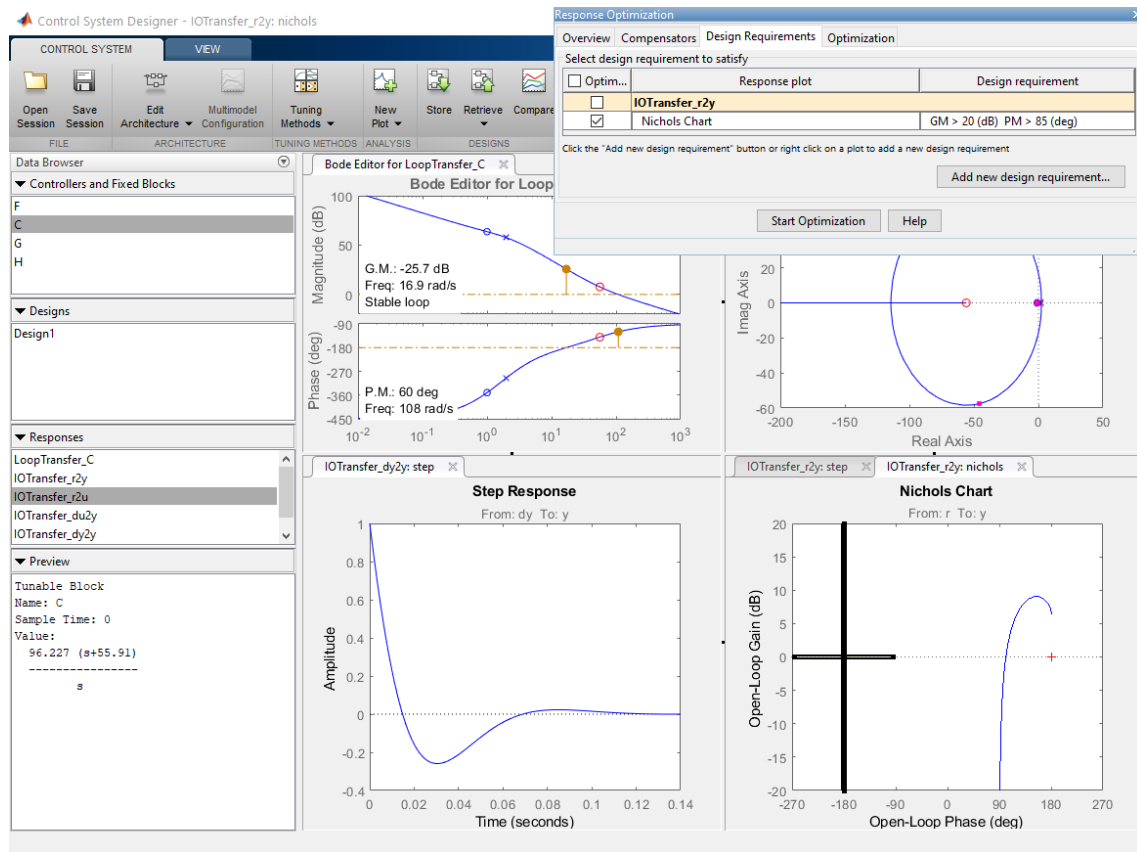


Figure 29: Optimization based tuning in the Control System Designer – Matlab Control System Toolbox

Every design can be stored and compared with other designs. Designed controller data can be exported as an `pid` object to Matlab workspace, or as a `.mat` file, or they could be sent to block diagram in Simulink where they can be used for further analyse and/or simulations.

In the conclusion, the Control System Designer provides very well-arranged interactive GUI reliable for simple processes. Every change made by the user is immediately projected into the design by dynamically recomputing other plots, stability margins and controller parameters. The Matlab computation is very fast. It is very general tool suitable for manual controller tuning. The user can either drag the plots and obtain controller zeros, poles, and gain or enter specific zeros, poles, and gain values and obtain essential plots in the time and frequency domain. There are no predefined forms of PID controllers from which we can chose. It is up to the designer what values of gain, zeros and poles to include in the controller. This approach probably would not work in the practice, because it is complicated. In the industrial practice we often choose PID controller due to its simplicity and effectiveness. PID controller is commonly used and understood, because it is able to sufficiently control large set of systems. It is unnecessary to add more parameters. The stability margins are computed automatically. However during the manual tuning there is no possibility to specify them. In the automated tuning we receive easiest possible controller solution for the given process. In the Optimization Based Tuning section we can specify multiple design criteria not only for closed loop transfer function, but for all four sensitivity functions. It is great to be able to optimize one sensitivity function. However if we optimize one sensitivity function we will definitely, due to Bode Integral Formula [21], worsen the other sensitivity functions. For the PM and GM design requirements we can only select interval greater than input value. There is no possibility to include upper limit of the stability margins. Optimization is very effective for simple processes with

small amount of design criteria. However for multiple design requirements the optimization progress takes very long time and becomes very computationally expensive. If we lack PID designing experience we can create unsatisfiable requirement set and the optimization process will never stop recomputing parameters and after 100 iterations it will automatically end. Furthermore, the system uncertainty at given frequency is not integrated in Control System Designer. So in the case we wanted use Control System Designer we would probably find the parameter set that stabilizes the given process. Then we would perform local optimization because we have no parameter region defining their stable intervals.

3.2.3 Control System Tuner

Final tool we wanted to mention from the CST is Control System Tuner. It is similar to the Control System Designer (3.2.2) with the difference that it uses automatic tuning algorithms. It lets user design any control system architecture including multiple fixed-order, fixed-structure SISO or MIMO control elements such as gains, PID controllers, or low-order filters distributed over any number of feedback loops [26]. Loops could be joint in a multiloop control system. It is less general than Control System Designer. Instead of creating our own controller, we can select the PID controller and specify its gains. Control System Tuner offers user the possibility to design a controller that is robust to changes in plant dynamics. These changes appear as parameter variations, variations in operating conditions, and sensor or actuator failures. This way we are able to integrate the uncertainty into our system. The parametric uncertainty can be defined in Matlab by the command `ureal()`. This way we can select the uncertainty percentage for each system parameter. The next step is to create a block diagram in Simulink containing the defined parametric uncertainty. Simulink uses this parameters and creates the uncertain state space representation which can be loaded into the Control System Tuner. Control System Tuner then tunes the PID controller against the uncertain state space system. Next step is to select from multiple design requirements such as tracking performance, disturbance rejection, noise amplification, closed loop damping, closed loop pole locations, and stability margins. User can specify the must-have requirements (design constraints) and the remaining requirements (objectives). Control System Tuner then optimizes PID parameters according to the chosen requirements and plots time and frequency responses for all variations of parameter values according to the their uncertainty percentage. We can export the designed closed loop transferfunction containing the robust controller as a `robust` object into the Matlab. In the Matlab we can compare former nominal model with the robust model. The designed robust PID controller works often better for the worst process parameter set than the controller designed for the nominal model.

The Control System Tuner was mentioned in this thesis mainly because of its ability to add uncertainty into the process model. It is able to specify multiple criteria. However the situation is same as in the previous section (3.2.2). The incautious criteria selection may result in endless optimization cycle because the controller satisfying the chosen criteria may not exist. Another point is, that by optimizing the controller against one design requirement can worsen its performance against other important requirements. There is no controller parameter region which would restrict the allowable parameters for robust control.

3.2.4 Conclusion

The main advantage of Matlab PID controller tuning methods is the common Matlab environment. Matlab is a programming language with specified objects, methods and data operations. We can simply export our progress from one Matlab app to another without any difficult data manipulation. We can analyse our data in a Matlab script using Matlab commands. The environment itself is well-arranged and interactive. Graphical elements are objects which can be accessed by standardized commands. The dynamic computations are very quick. In terms of the quality of graphical design and overall interactivity, this ranks Matlab above the previous mentioned tuning methods (3.1). Moreover, the Matlab software is actual, new versions are released every year, and it has comprehensive documentation accessible online. It allows to integrate percentage uncer-

tainty into the system model.

On the other hand, there are few significant disadvantages of the Matlab PID controller tuning methods. It is not free and open-source. More importantly unlike PIDlab, it does not include the robust stability region tuning method. There is no controller parameter boundary delimiting the set of parameter values that would provide stability for the given process. Thus, we might define a design requirement set for which the controller does not exist. In this case the optimization often ends in endless cycles. This is a problem, if the user is lacking the experience in the field of control theory. Matlab does not know the controller parameter set boundary for the given requirements because they are the product of the optimization. Next disadvantage is, that the user can select nor multiple stability margins nor general shaping points. There is also no option to design a controller for multiple processes at one time. To summarize, despite being the most advanced mentioned tuning method, the Matlab controller design software is lacking some very important approaches such as robust stability region method or H_∞ region approach. It would be suitable for simpler plants with the optimization being only local.

4 Implementation of the designed interactive tool

In this section, the process of GUI implementation will be described. This includes the description of construction and structure of the interface, the implementation of PI and PI^α tuning algorithms (mentioned in Section 2.2) resulting in visualization of the robust stability regions, and inclusion of the multi-criteria optimization through finding the intersection robust stability region for all design criteria belonging to each system.

The GUI was developed in Matlab app building interface App Designer. The MathWorks software was chosen primarily due to its convenient implemented algorithms for matrix manipulations and fast numeric computing. Apart from the strength and speed of the Matlab compiler, the MathWorks software gathers large set of control theory algorithms in predefined functions (such as `bode()` for Bode plot, or `tf()` for transfer function declaration). MathWorks offers user friendly app developing environment for beginners in interface building. Several tuning apps with elaborate interactive interface have already been developed through Matlab methods (read more in Section 3.2). Another reason why I have chosen Matlab environment is that I am familiar with Matlab programming language.

4.1 App Designer

First, I would like to introduce the App Designer environment. It integrates the two primary tasks of app building – laying out the visual components of a graphical user interface (GUI) and programming app behavior. It is the most recent environment for building apps in Matlab. It allows to drag and drop visual components to the design canvas using alignment hints to get a precise layout. One of the main advantages of App Designer is that it automatically generates the object-oriented code that specifies the app’s layout and design. It contains the the integrated version of the Matlab Editor where the functionality of the app can be defined. It uses the Code Analyzer which automatically checks if any coding problem appears [15].

The core of the app building consists of components (buttons, check boxes, trees, drop-down lists etc.). Smaller components can be organized by using the container components (for example tabs, or panels). Each Component has predefined structure and appearance. It is up to the user to define its functionality and behaviour. This can be done by adding component callbacks or by custom mouse and keyboard interactions that execute when a user interacts with the app.

App Designer lets user create a standalone application.

4.2 Designed graphical user interface

In this section, the functionality of the designed GUI will be discussed. The implementation of PI robust stability region control algorithm together with the visualization of the three-dimensional PI^α robust stability regions will be described here.

The designed GUI consists of multiple App Designer components. Majority of them are UIAxes containing function plots. Some of the data are stored in the UITables where they are easily accessible. The GUI components are organized into several sections. Specifically: Define Process section, System Nyquist Plot section, Robust Stability Region Plot section, Sensitivity Functions Step Responses section, System Step Response section, Sensitivity Functions Bode Diagrams section (Figure 30).

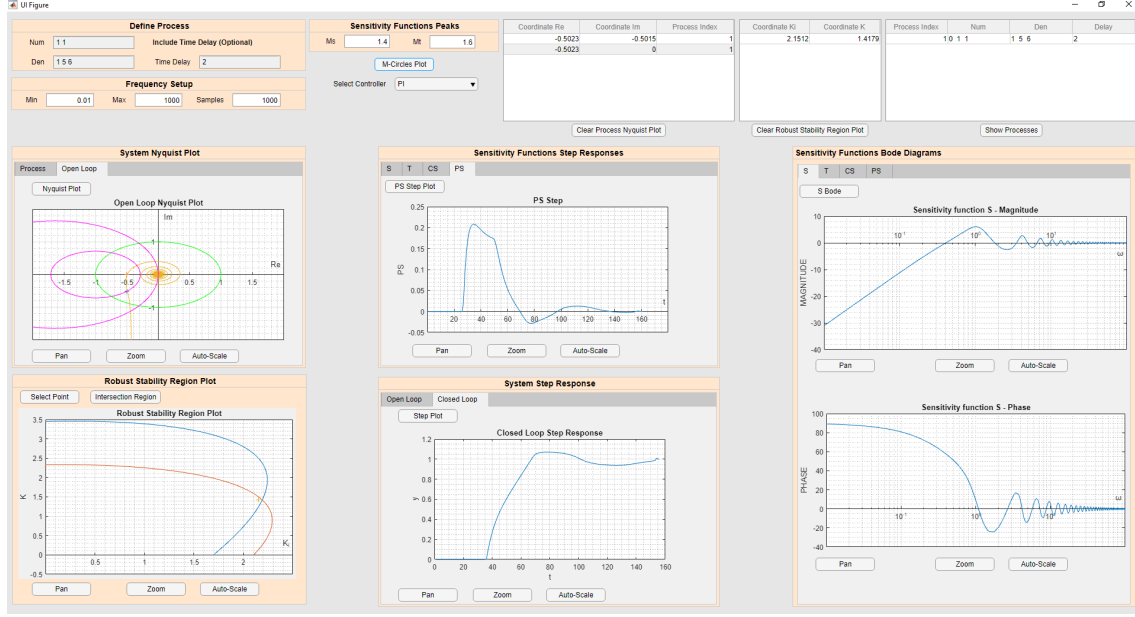


Figure 30: Designed graphical user interface

The process can be defined in the Define Process section (Figure 31a). In this section, the numerator, denominator and the time delay of the given process are inserted separately. As the form of data input, the Text Edit Field block was used. Input is later converted from string to double. Numerator and Denominator block accepts the polynomial coefficients of the process transfer function numerator and denominator. The Time Delay Edit block accepts the time constant D which is inserted into the expression e^{-Ds} . It is user choice, whether to include Time Delay or leave the block empty. It is important to mention that this application is suitable for stable non-oscillatory systems with relative order greater than 0. This means that $deg(den) > deg(num)$.

Transfer functions $P_1(s)$, $P_2(s)$ given by the equations

$$P_1(s) = \frac{s + 1}{s^2 + 5s + 6} e^{-2s}, \quad (73)$$

$$P_2(s) = \frac{s + 1}{s^2 + 5s + 6}, \quad (74)$$

were used as a demonstrative example.

After the system is inserted, it is stored into the system database from where it can be later accessed. Each system obtains its index which is also stored. System database is implemented as a table (Figure 31b). In the table we can see the index, numerator, denominator and time delay for each given system. To show the processes click "Show Process" button.

(a) Define process section

Process Index	Num	Den	Delay
1	1 1	1 5 6	2
2	0 1	1 5 6	1
3	1	1 3	3

(b) Table with stored processes

Figure 31: Define and store process sections

(a) Sensitivity functions peaks M_s , M_t

(b) Frequency interval and sampling

Figure 32: Configuration of the system parameters

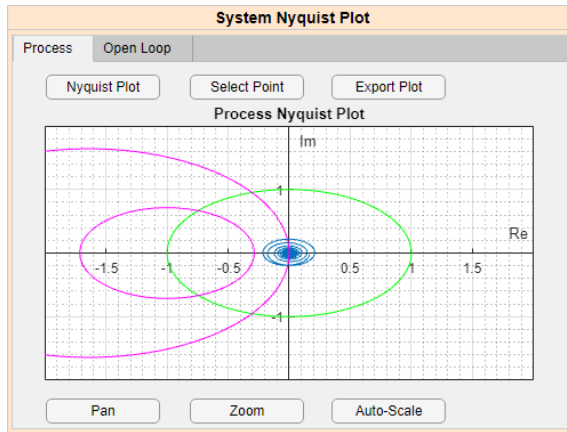


Figure 33: Process Nyquist plot

After the process is declared, we can show its Nyquist plot (Figure 33). Unit circle together with M -circles can also be seen on the plot. Radius of the M_s -circle as well as the radius of the M_t -circle can be modified through input of the sensitivity functions maximal values M_s and M_t (Figure 32a). The implicit values are $M_s = 1.4$, $M_t = 1.5$. Plotting frequency of the process Nyquist curve can also be modified (Figure 32b) by changing the initial and end value as well as the sample rate of a logarithmically spaced frequency vector.

Next step is to select a controller. This GUI implements PI and PI^α robust stability regions, so it is possible to choose either of the PI or PI^α

controller.

4.2.1 Implementation of the PI control algorithm

In this section, we suppose that PI controller was selected for the process control. After the controller is selected, we can add Nyquist shaping points representing required controller design criteria. Shaping points can be selected by clicking the "Select Point" button (Figure 33 above the graph). The "Select Point" button opens figure with the unit circle and M -circles in separate window. The shaping points selection is done by mouse clicking in the plot. Mouse click creates a cross on the position of the selected point. The relative coordinates of the mouse click are gathered by Matlab function `ginput()`. Coordinates of the selected point are stored with the matching process index in the table (Figure 34).

Coordinate Re	Coordinate Im	Process Index
-0.5023	-0.5015	1
-0.5023	0	1

Figure 34: Table of shaping point coordinates: Process $P_1(s)$ (73), Criteria: $PM \approx 45^\circ$, $GM \approx 2$

Coordinates together with the real and imaginary parts of the current process are then inserted into the formulas for computing controller gains $K(\omega)$, $K_i(\omega)$. For PI controller the formulas are

given by equations (40), (41). It was shown earlier that $K(\omega)$ and $K_i(\omega)$ represent region boundary in K, K_i plane. Meaning that for each shaping point (which is equal to the design requirement) the robust stability region is plotted in the Robust Stability Region Plot section (Figure 35a). Since with the increasing frequency the number of origin encirclement increases, it was essential to find the first encirclement around the origin in the first quadrant. The implemented algorithm uses Matlab function `fzero()` which finds the zero value of the input mathematical function. The condition of the first quadrant was met without a problem since only the positive values of $K(\omega)$, $K_i(\omega)$ were accepted. The condition on the first encirclement was harder to meet, since the frequency samples are distributed differently for different processes and different shaping points. Function `fzero()` operates on the vicinity of the certain value meaning that it was necessary to select initial frequency for the algorithm to search for the zero values. If the $K(\omega) = 0$ then for increasing frequency ω we would be able to find the $K_i(\omega)$ coordinate marking the intersection point of the parametric curve with the K_i axis. Now, if we increased the frequency on the rationally selected interval we would reach the point where $K_i(\omega) = 0$, and for that frequency we would be able to obtain the value of the $K(\omega)$ coordinate which would mark the intersection point of the parametric curve with the K axis. This method was used in order to get the first encirclement of the origin in the first quadrant.

There is a possibility to visualize the intersection region by clicking the "Intersection Region" button in the Robust Stability Region Plot section (Figure 35b). This region was obtained through Matlab functions implementing the set operations. The two-dimensional shape is created by connecting the K , K_i axes and the parametric curve $K(\omega)$, $K_i(\omega)$ by the function `polyshape()`. This repeats for each plotted curve. The intersection is executed by the function `intersection()` which operates with the cell array of the `polyshape` objects.

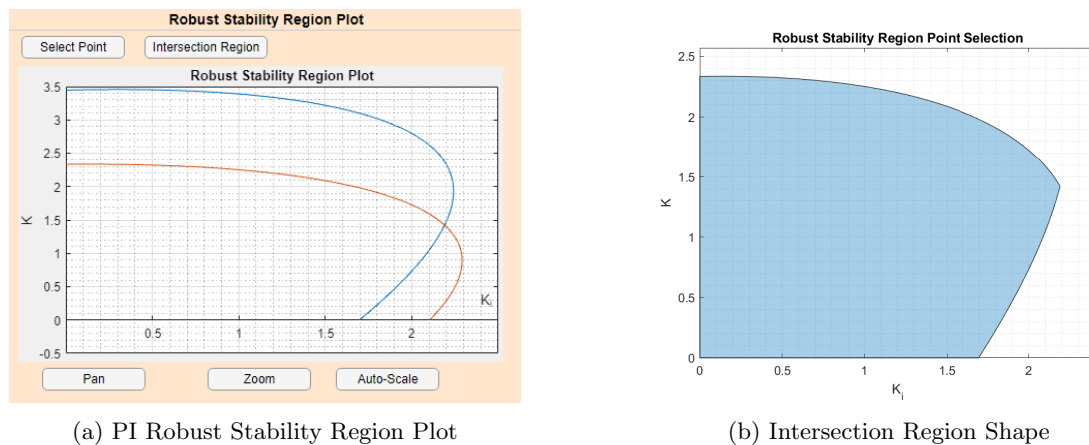


Figure 35: PI robust stability region plot section: Process $P_1(s)$ (73), Criteria: $PM \approx 45^\circ$, $GM \approx 2$ (Figure 34)

Coordinate K_i	Coordinate K
2.1617	1.4179

Figure 36: Table of K , K_i coordinates

Next step is to select a controller from the Robust Stability Region Plot. The controller parameters are selected by clicking "Select Point" button. This button opens separated window where we can select the coordinates which are gathered by Matlab function `ginput()` and stored in the table (Figure 36) where they are easily accessible.

After the controller parameters are selected, the essential time and frequency characteristics can be plotted. First of all, it would be recommended to plot the Nyquist curve of the Open Loop system (Figure 37). In this figure we can check whether the Nyquist curve passes on the right of the selected shaping points and of the critical point $[-1, j0]$. This plot contains all inserted processes together with all selected shaping points for each process. The unit circle and the M -circles are

also included in the plot.

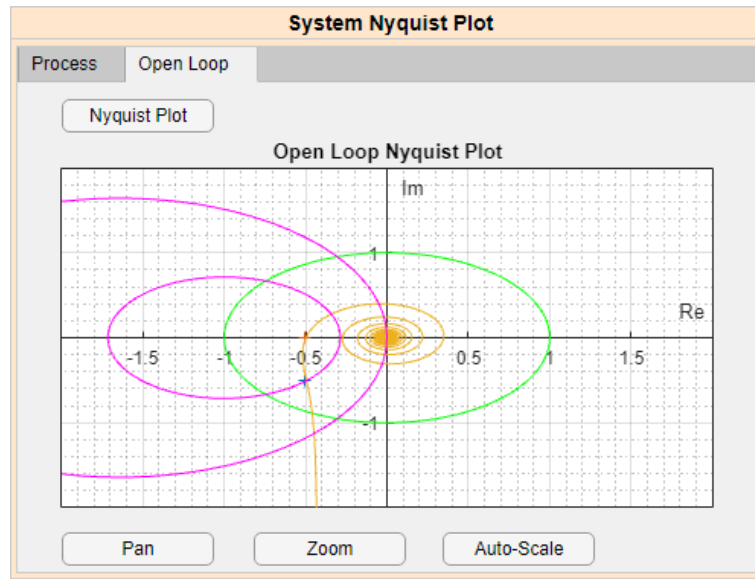
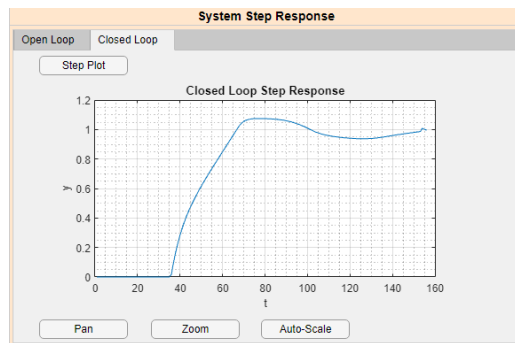
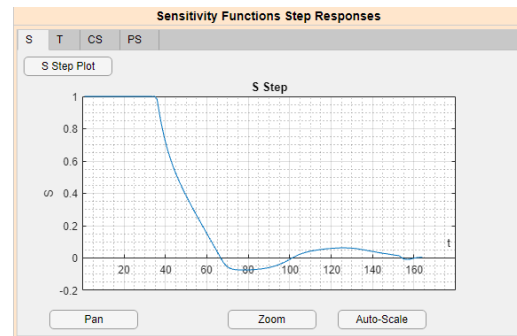


Figure 37: Open loop Nyquist plot



(a) Step response of the open and closed Loop



(b) Step response of the sensitivity functions

Figure 38: Time characteristics of the process

The step responses of open loop, closed loop and sensitivity functions can be seen on the right of the System Nyquist Plot section (Figure 38). Step responses are plotted for each inserted system with the last selected controller parameters being used. The user can use "Zoom", "Pan", or "Auto-Scale" button to manipulate with the plots.

On the right of the step responses, there are Bode diagrams of the sensitivity functions (Figure 39). Bode diagrams are one of the frequency domain characteristics. Again, the Bode diagrams are plotted for each inserted process and for each design criteria. The controller uses the last set of the selected parameters. The frequency axis uses the logarithmic scale which is typical for the Bode plot. Same as for time responses, it is possible to zoom, pan, and auto-scale each Bode plot by clicking the buttons.

This summarizes the process of controller tuning in the designed interactive graphical interface which implements the PI control algorithm through robust stability regions.

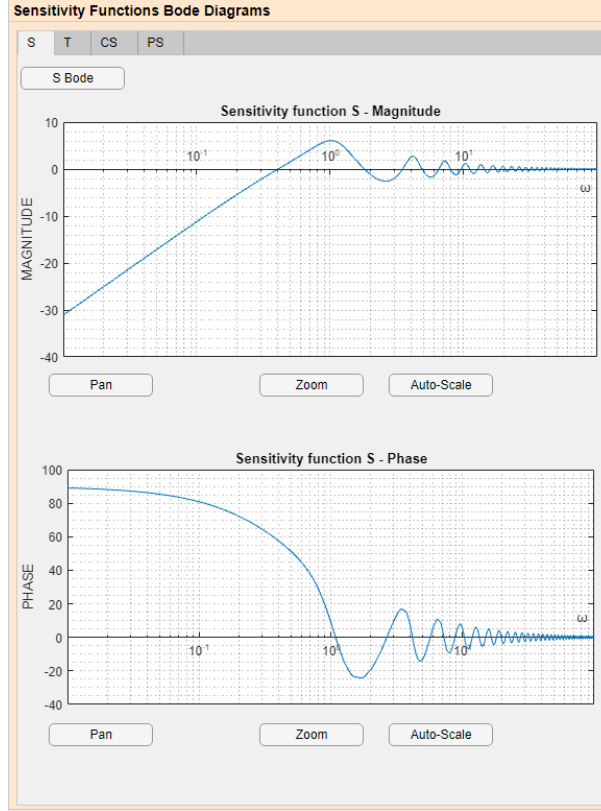


Figure 39: Bode Diagrams of the Sensitivity Functions

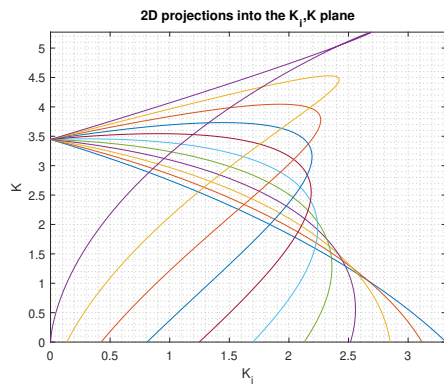
4.2.2 Visualization of the PI^α robust stability regions

This section focuses on the case when the PI^α controller was selected. PI^α control algorithm is mentioned in Section (2.2.3). After the controller is selected, we can add design criteria in the form of the Nyquist shaping points. Same as in the previous section, this functionality is implemented in the callback function of the "Select Point" button in the System Nyquist Plot section which opens figure with the unit circle and M -circles in separate window. After selecting the required criteria by clicking in the plot, the coordinates of the mouse click are stored together with the process index in the table.

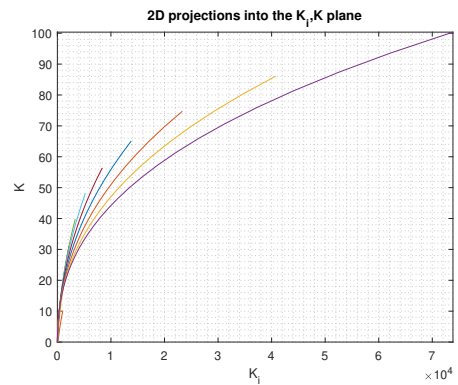
The coordinates together with the real and imaginary parts of the current process are inserted into the formulas (46), (47) containing the $K(\omega)$ and $K_i(\omega)$ controller gains expressions. The region boundary described by the $K(\omega)$, $K_i(\omega)$ parametric curve is expanded into the three-dimensional space by the vector α (Section 2.2.3). Since the plane curved in the 3D space encircles the origin multiple times, it was essential to find a method which would compute the 3D regions in the first quadrant. The procedure was similar to the previous section. Again, the Matlab function `fzero()` was used. Only difference was that the procedure repeated for every element of the vector α . The output of the "Select Point" button callback function are two figures. First of them contains the mesh graph of the regions first encirclement of the origin in the first quadrant. Second figure contains the projection of the region into the K, K_i plane.

The following figures show the three-dimensional region in the first quadrant (Figures 41, 42) and its projection into the K, K_i plane (Figure 40). Two transfer functions were used as a demonstrative example. One with time delay $P_1(s)$ (73), second without time delay $P_2(s)$ (74). The shaping point was identical for both of them $[u, v] \approx [-0.5, -0.5]$. The α was implemented

as: $\alpha = [0.5, 0.6, 0.7, \dots, 1.5]$. Plotting frequency was set as 100 logarithmically spaced samples in the range $\omega = [10^{-1}, \dots, 10^2]$.



(a) Process $P_1(s)$ (73) with time delay



(b) Process $P_2(s)$ (74) without time delay

Figure 40: Projection of the PI^α robust stability regions into the K, K_i plane

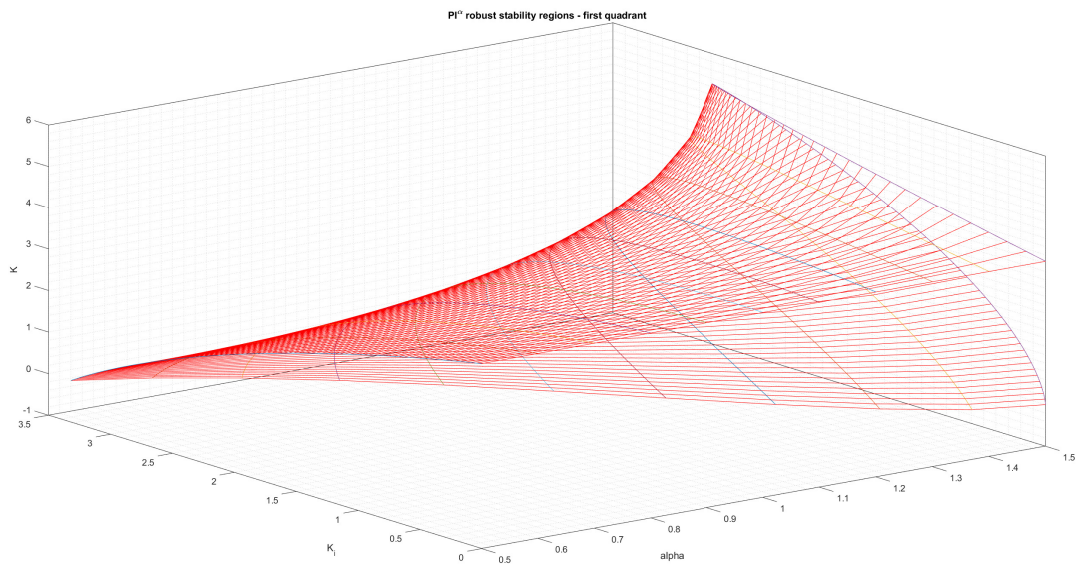


Figure 41: PI^α robust stability region: Process $P_1(s)$ (73) with time delay

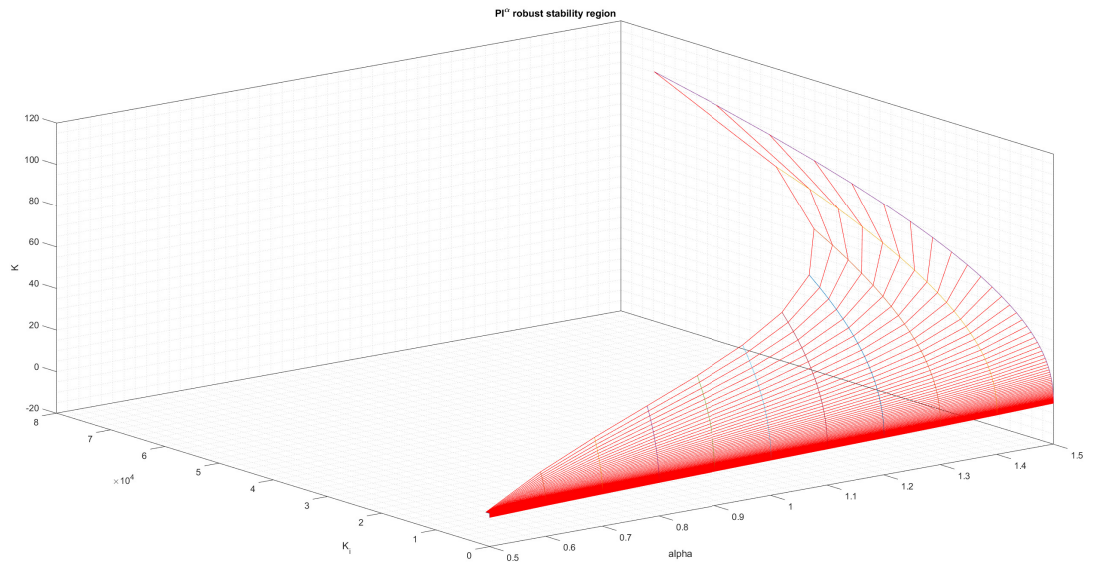


Figure 42: PI^α robust stability region: Process $P_2(s)$ (74) without time delay

This way the designed GUI implements the computing and visualizing of the three-dimensional robust stability regions for the PI^α controller. The GUI does not implement the intersection of the three-dimensional PI^α regions, nor the 3D point selection representing the PI^α controller parameters, as it was not the main aim of this thesis. The main reason why this functionality was not implemented is its computational expensiveness. However, mentioned three-dimensional operations could be part of the future works.

5 Multi-criteria optimization

One of the main aims of this thesis was to design a module which would implement the multi-criteria parameter optimization. This section is focused on explaining how the multi-criteria optimization was implemented into the GUI, as well as demonstrating the optimization on a practical examples.

GUI enables us to insert any number of processes. For each process, we can select as many Nyquist shaping points as we want. Each of the point represents given design requirement. The robust stability region is obtained for each design requirement. The intersection region of all regions represents the set of controller parameters which satisfy all of the design criteria for all processes. GUI implements the region intersection for PI controller. The intersection of found regions is displayed in the separated window where it can be stored or exported. As an example, the multi-criteria optimization was performed for the process $P_1(s) = \frac{1}{2s^2+3s+4}e^{-1.5s}$.

After the process is inserted, the shaping points $X_1 = [u_1, v_1] = [-0.5, 0]$, $X_2 = [u_2, v_2] = [-0.3088, -0.9446]$ were selected. Shaping points X_1, X_2 represent design criteria $GM \geq 2$, $PM \geq 71^\circ$. For each criteria the robust region is obtained (Figure 43a). The intersection region shows in the separated figure after clicking the "Intersection Region" button (Figure 43b). The intersection region shows which controller parameters can be selected in order to satisfy the defined design requirements.

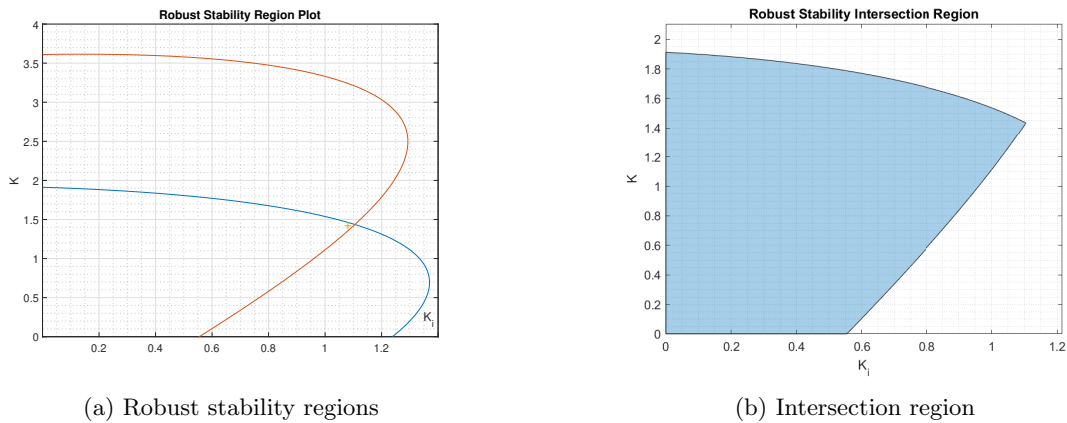
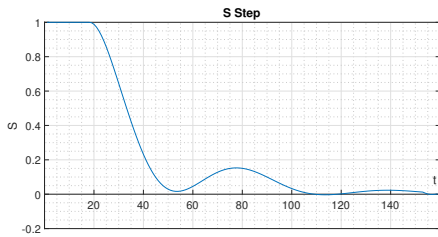
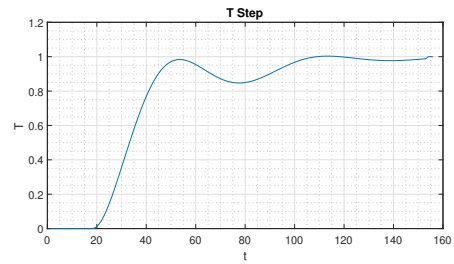


Figure 43: PI robust stability region plot: Process $P_1(s)$, Criteria: $GM \geq 2$, $PM \geq 71^\circ$

The coordinates $K = 1.4193$, $K_i = 1.0809$ have been chosen as the PI controller parameters. In this case, the nearly maximal value of K_i was chosen from the region. The reason for this choice is that the maximal K_i value guarantees the faster step response but amplifies the overshoot. Since we selected point from the region, we have found controller which satisfies mentioned design criteria. Now we can visualize the time and frequency characteristics (Figures 44, 45, 46). *Note: All plots have been exported to Matlab figures.*

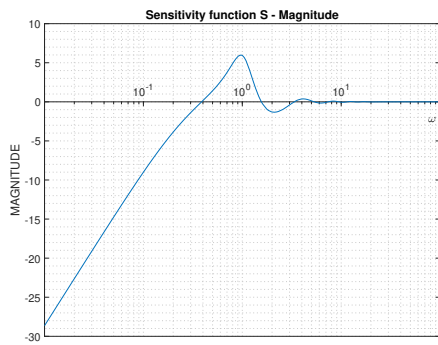


(a) $S(s)$ step response

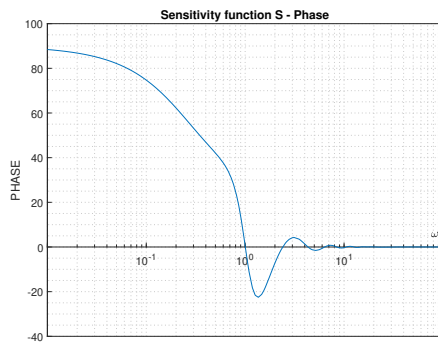


(b) $T(s)$ step response

Figure 44: Step responses of the sensitivity functions $S(s)$, $T(s)$ for the process $P_1(s)$, $[K, K_i] = [1.4193, 1.0809]$

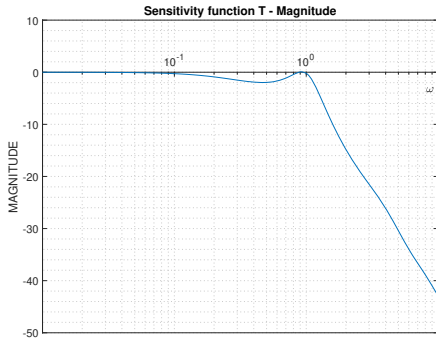


(a) $S(j\omega)$ magnitude

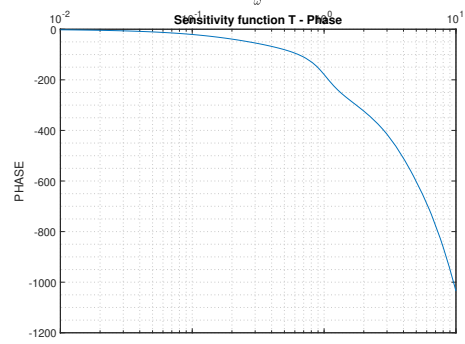


(b) $S(j\omega)$ phase

Figure 45: Bode diagram of the sensitivity function $S(j\omega)$ for the process $P_1(s)$, $[K, K_i] = [1.4193, 1.0809]$



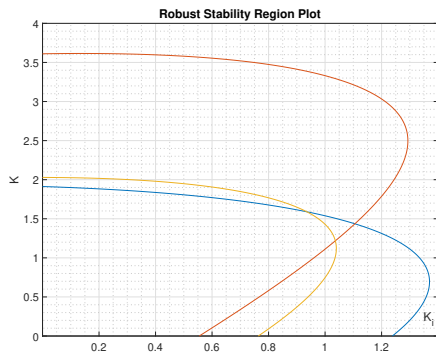
(a) $T(j\omega)$ magnitude



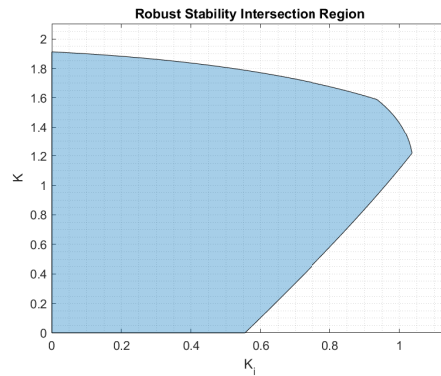
(b) $T(j\omega)$ phase

Figure 46: Bode diagram of the sensitivity function $T(j\omega)$ for the process $P_1(s)$, $[K, K_i] = [1.4193, 1.0809]$

All time and frequency characteristics have a correct shape. That means we have obtained PI controller parameter setting which satisfies design criteria $GM \geq 2$, $PM \geq 71^\circ$. However we can add more design criteria or even more processes. After adding another design criteria for the $P_1(s)$ in the form of shaping point $X_3 = [u_3, v_3] = [-4.009, -0.3965]$ we can see that the intersection changes (Figure 47).



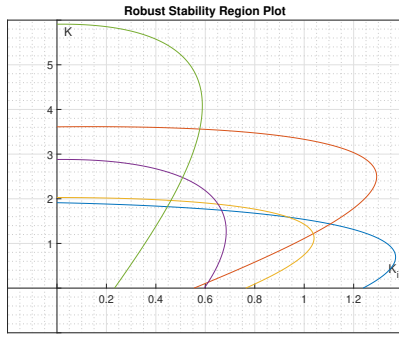
(a) Robust stability regions



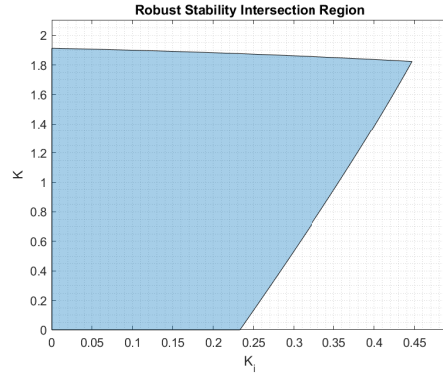
(b) Intersection region

Figure 47: PI robust stability region plot: Process $P_1(s)$, Shaping points: $X_1 = [u_1, v_1] = [-0.5, 0]$, $X_2 = [u_2, v_2] = [-0.3088, -0.9446]$, $X_3 = [u_3, v_3] = [-4.009, -0.3965]$

We can add new process $P_1(s) = \frac{s+1}{s^2+5s+6}e^{-8s}$ with its set of design requirements $Y_1 = [u_{y1}, v_{y1}] = [-0.5, 0]$, $Y_2 = [u_{y2}, v_{y2}] = [-0.3088, -0.9446]$ which are equal to the X_1, X_2 shaping points. The intersection changes again (Figure 48).



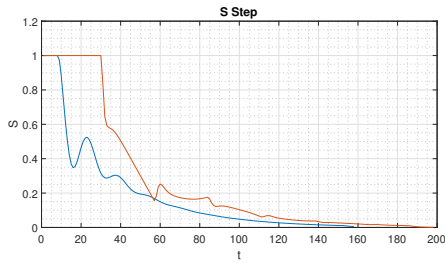
(a) Robust stability regions



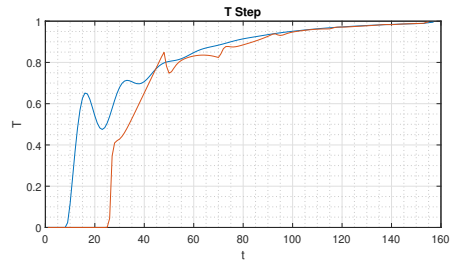
(b) Intersection region

Figure 48: PI robust stability region plot: Processes $P_1(s)$, $P_2(s)$, Shaping points: $X_1 = [-0.5, 0]$, $X_2 = [-0.3088, -0.9446]$, $X_3 = [-4.009, -0.3965]$, $Y_1 = [-0.5, 0]$, $Y_2 = [-0.3088, -0.9446]$

Coordinates $[K, K_i = 1.7293, 0.4118]$ have been selected from the intersection region. Obtained PI controller $C(s) = 1.7293 + \frac{0.4118}{s}$ is able to control both processes $P_1(s)$, $P_2(s)$ while satisfying design criteria for each process. Time and frequency characteristics can be seen on the Figures 49, 50,

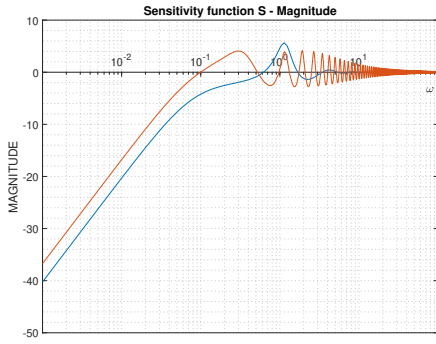


(a) $S(s)$ step response

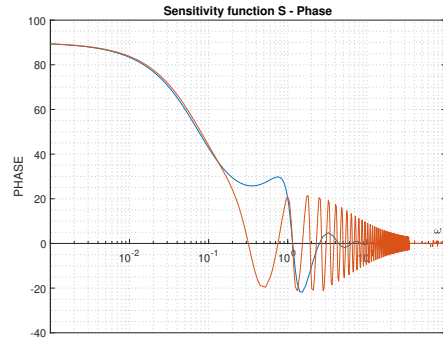


(b) $T(s)$ step response

Figure 49: Step responses of the sensitivity functions $S(s)$, $T(s)$ for the processes $P_1(s)$, $P_2(s)$, $[K, K_i = 1.7293, 0.4118]$

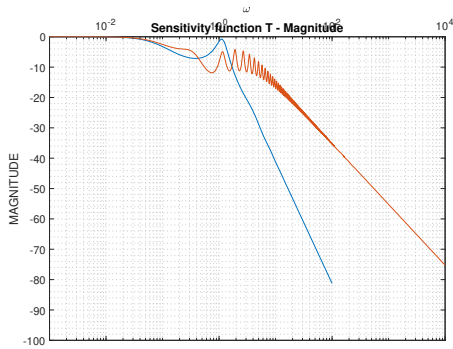


(a) $S(j\omega)$ magnitude

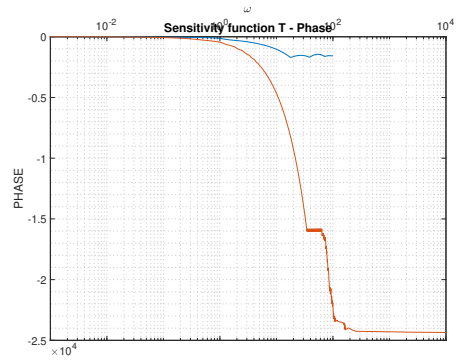


(b) $S(j\omega)$ phase

Figure 50: Bode diagram of the sensitivity function $S(j\omega)$ for the processes $P_1(s)$, $P_2(s)$, $[K, K_i = 1.7293, 0.4118]$



(a) $T(j\omega)$ magnitude



(b) $T(j\omega)$ phase

Figure 51: Bode diagram of the sensitivity function $T(j\omega)$ for the processes $P_1(s)$, $P_2(s)$, $[K, K_i = 1.7293, 0.4118]$

This is how the multi-criteria optimization is implemented in the GUI. It is possible to insert any number of processes and for each select any number of design criteria. Implemented GUI methods then compute the intersection region from which we can choose the controller gains. The intersection region can be exported in several formats. It would be a part of the future works to operate with the object representing the bitmap of the intersection region. It would be possible to compute the optimal parameters from the bitmap according to the performance criteria of the control loop, such as Integral time absolute error (ITAE), Integral square error (ISE), Integral time square error (ITSE) and Integral absolute error (IAE).

6 Validation

This section focuses on the GUI validation. It was shown in the earlier demonstrations that the GUI works for several processes with and without time delay. As it was described earlier, using the Nyquist shaping method is appropriate for stable non-oscillatory systems (Section 2.2.2) with relative order greater than 0 (Section 4.2). The positive relative order requirement means that only systems which have physical representation can be used for the controller design in this app.

6.1 Process set

In the first step of the validation, the set of stable non-oscillatory processes $G = \{G_1(s), G_2(s), G_3(s), G_4(s), G_5(s), G_6(s)\}$ have been selected for the GUI testing. The processes were given as

$$G_1(s) = \frac{1}{s+1}e^{-0.3s}, \quad (75)$$

$$G_2(s) = \frac{1}{s^2+4s+3}e^{-s}, \quad (76)$$

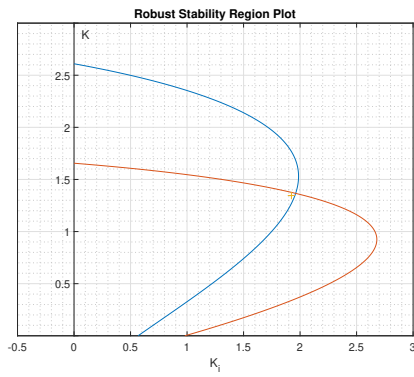
$$G_3(s) = \frac{s+1}{s^2+4.5s+4.5}e^{-0.8s}, \quad (77)$$

$$G_4(s) = \frac{1}{s^3+7.6s^2+17.99s+12.74}e^{-1.5s}, \quad (78)$$

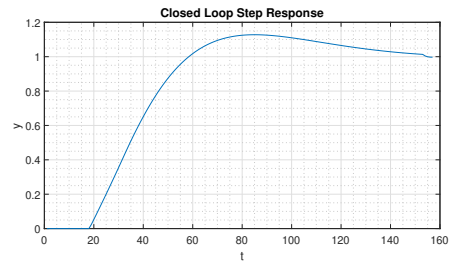
$$G_5(s) = \frac{s+2}{s^3+7.6s^2+14.69s+8.45}e^{-0.7s}, \quad (79)$$

$$G_6(s) = \frac{s^2+6s+9}{s^3+9.5s^2+24s+18}e^{-2s}. \quad (80)$$

Selected process set matches some processes of the process test batch used by K. J. Åström and T. Häglund [24]. All of the processes contain time delay because the time delay is present in the real processes. Design criteria $GM \geq 3.5575$ and $PM \geq 45^\circ$ have been chosen for all processes. PI controller was selected for the controller design. The robust stability regions together with the closed loop step responses for each process can be seen on the Figures 52, 53, 54, 55, 56, 57. The point with the highest value of K_i coordinate was selected from every intersection region in order to obtain the fastest step response.

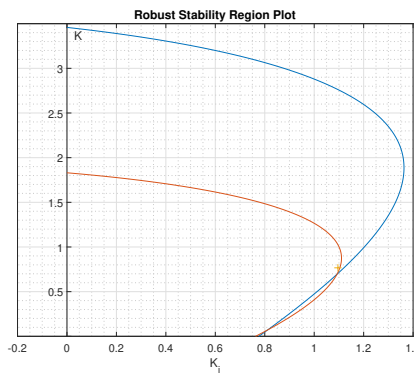


(a) Robust stability region plot

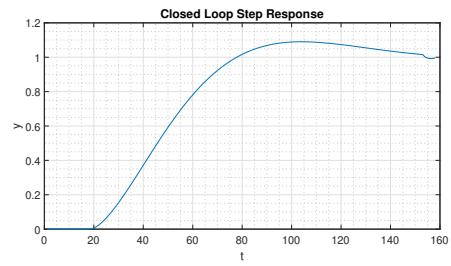


(b) Closed loop step response

Figure 52: Robust stability region plot and closed loop step response: Process $G_1(s)$, Design criteria: $GM \geq 3.5575$, $PM \geq 45^\circ$, $[K, K_i] = [1.3456, 1.9234]$

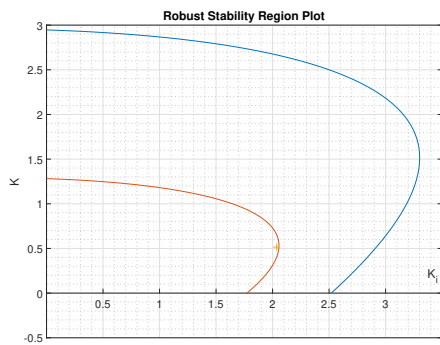


(a) Robust stability region plot

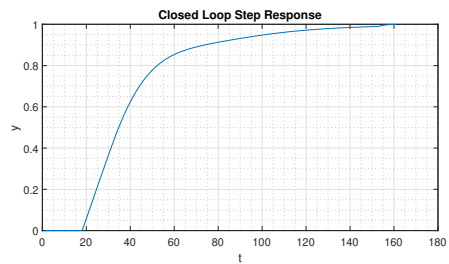


(b) Closed loop step response

Figure 53: Robust stability region plot and closed loop step response:: Process $G_2(s)$, Design criteria: $GM \geq 3.5575$, $PM \geq 45^\circ$, $[K, K_i] = [0.7669, 1.0959]$

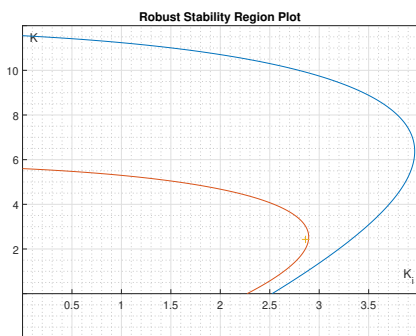


(a) Robust stability region plot

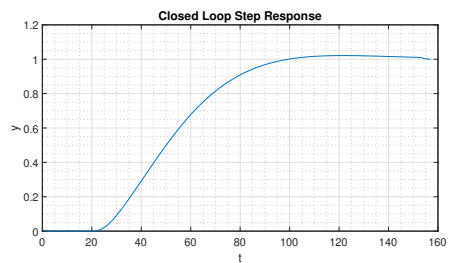


(b) Closed loop step response

Figure 54: Robust stability region plot and closed loop step response: Process $G_3(s)$, Design criteria: $GM \geq 3.5575$, $PM \geq 45^\circ$, $[K, K_i] = [0.5153, 2.0363]$

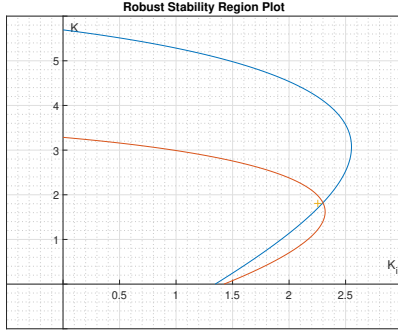


(a) Robust stability region plot

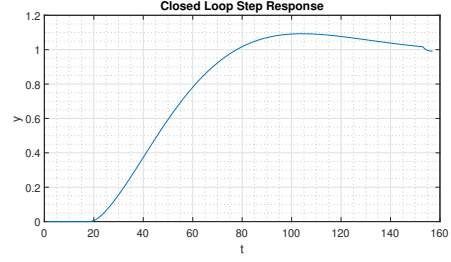


(b) Closed loop step response

Figure 55: Robust stability region plot and closed loop step response: Process $G_4(s)$, Design criteria: $GM \geq 3.5575$, $PM \geq 45^\circ$, $[K, K_i] = [2.4286, 2.8618]$

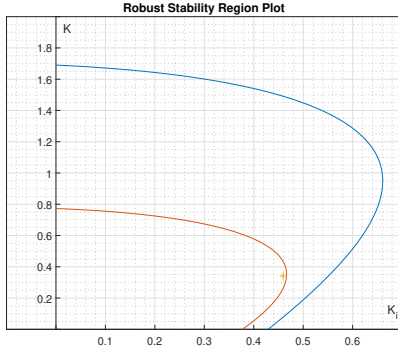


(a) Robust stability region plot

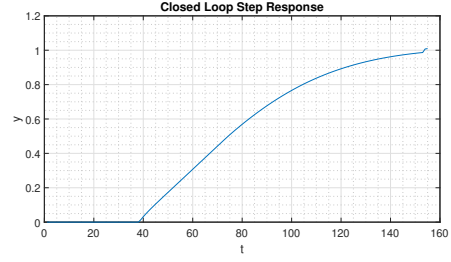


(b) Closed loop step response

Figure 56: Robust stability region plot and closed loop step response: Process $G_5(s)$, Design criteria: $GM \geq 3.5575$, $PM \geq 45^\circ$, $[K, K_i] = [1.8061, 2.2540]$



(a) Robust stability region plot



(b) Closed loop step response

Figure 57: Robust stability region plot and closed loop step response: Process $G_6(s)$, Design criteria: $GM \geq 3.5575$, $PM \geq 45^\circ$, $[K, K_i] = [0.3411, 0.4594]$

We can see that the closed loop step responses always reach the set point for each process of the set G .

6.2 Coupled tanks

In the second step of the validation, the real process has been selected as a practical example. The real process consisted of coupled tanks (Figure 58). The system can be described by two non-linear equations obtained by the application of the Bernoulli's equations as

$$\frac{dH_1(t)}{dt} = -\frac{1}{S}c_pS_p\sqrt{2g(H_1(t) - H_2(t))} + \frac{1}{S}Q_1(t), \quad (81)$$

$$\frac{dH_2(t)}{dt} = \frac{1}{S}c_pS_p\sqrt{2g(H_1(t) - H_2(t))} - \frac{1}{S}c_2S_2\sqrt{2gH_2(t)}. \quad (82)$$

Each equation expresses changing of the liquid level in each vessel. In order to control this system, we need to linearize it. The linearized state space can be expressed as

$$\begin{bmatrix} \dot{H}_1 \\ \dot{H}_2 \end{bmatrix} = \begin{bmatrix} -\frac{c_p \cdot S_p \cdot \sqrt{2g}}{2S \cdot \sqrt{H_{10} - H_{20}}} & \frac{c_p \cdot S_p \cdot \sqrt{2g}}{2S \cdot \sqrt{H_{10} - H_{20}}} \\ \frac{c_p \cdot S_p \cdot \sqrt{2g}}{2S \cdot \sqrt{H_{10} - H_{20}}} & -\left[\frac{c_p \cdot S_p \cdot \sqrt{2g}}{2S \cdot \sqrt{H_{10} - H_{20}}} + \frac{c_2 \cdot S_2 \cdot \sqrt{2g}}{2S \cdot \sqrt{H_{20}}} \right] \end{bmatrix} \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Delta Q_1,$$

For a specific set of parameters we can express the state-space of the linearized model of the first level where $y(t) = H_1(t)$ as

$$A = \begin{bmatrix} -0.1 & 0.1 \\ 0.1 & -0.16 \end{bmatrix}, \quad B = \begin{bmatrix} 400 \\ 0 \end{bmatrix}, \quad C_1 = [1 \quad 0], \quad D_1 = [0]$$

Similarly, the state-space of the linearized model of the second level where $y(t) = H_2(t)$ is given as

$$A = \begin{bmatrix} -0.1 & 0.1 \\ 0.1 & -0.16 \end{bmatrix}, \quad B = \begin{bmatrix} 400 \\ 0 \end{bmatrix}, \quad C_2 = [0 \quad 1], \quad D_2 = [0]$$

Transfer functions $F_1(s)$, $F_2(s)$ can be obtained from the state-spaces by using the formula $C(sI - A)^{-1}B$ as

$$F_1(s) = \frac{400s + 64}{s^2 + 0.26s + 0.006}, \quad (83)$$

$$F_2(s) = \frac{40}{s^2 + 0.26s + 0.006}. \quad (84)$$

In order to control the coupled tanks system, we add the water pump powered by the direct current electric engine. The water pump can be approximated as $F_{wp}(p) = \frac{1,5 \cdot 10^{-4}}{1 + 0,5p}$. Next, we connect the water pump in series with both vessels. We obtain transfer functions $P_1(s)$ and $P_2(s)$ where $P_1(s)$ represents the water pump connected to the first vessel, $P_2(s)$ represents the the water pump connected to the second vessel. Transfer functions $P_1(s)$ and $P_2(s)$ are given by the following formulas:

$$P_1(s) = F_{wp}(s) \cdot F_1(s) = \frac{1,5 \cdot 10^{-4}}{1 + 0,5p} \cdot \frac{400s + 64}{s^2 + 0.26s + 0.006} = \frac{0.6s + 0.096}{5s^3 + 11.3s^2 + 2.63s + 0.06}, \quad (85)$$

$$P_2(s) = F_{wp}(s) \cdot F_2(s) = \frac{1,5 \cdot 10^{-4}}{1 + 0,5p} \cdot \frac{40}{s^2 + 0.26s + 0.006} = \frac{0.06}{5s^3 + 11.3s^2 + 2.63s + 0.06}. \quad (86)$$

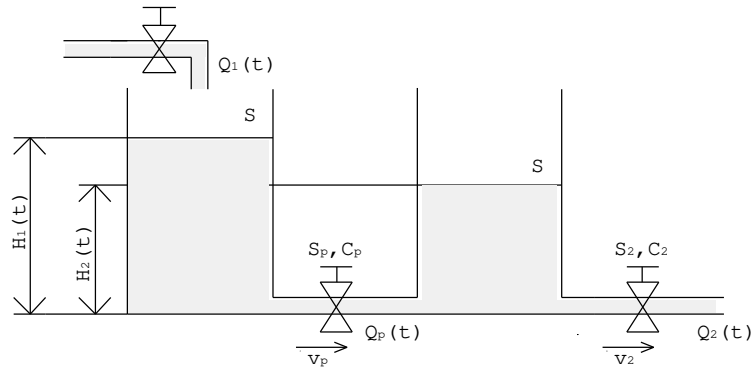


Figure 58: Coupled tanks system

The process $P_2(s)$ was selected as a practical example for the GUI validation. After the process was inserted into the GUI, its Nyquist plot has been visualized (Figure 59) and the PI controller was selected. The design criteria for the process were: $GM \geq 10$, $PM \geq 62^\circ$, $M_s \leq 1.4$. The requirement on the sensitivity function peak $M_s \leq 1.4$ was approximated by three shaping points located on the M_s -circle. Next, the robust stability regions were computed for each design criteria (Figure 60a). The controller parameters with nearly maximal K_i coordinate have been selected from the intersection region: $[K, K_i] = [4.0147, 0.1187]$. On the Figure 60b we can see that the Nyquist curve passes on the right of the every selected shaping point. On the Figure 61 we can see that sensitivity functions $S(s)$, $T(s)$ reach required values 0, and 1 in approximately 150 time units. Bode diagrams of the sensitivity functions $S(s)$, $T(s)$ can be observed on the Figures 62, 63.

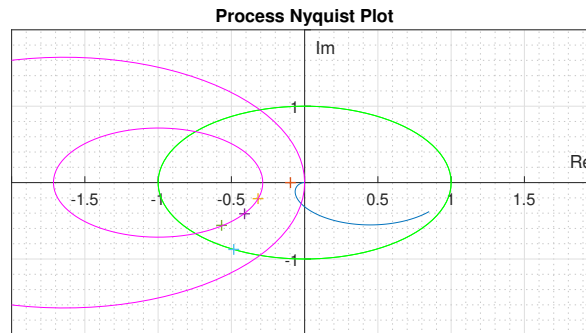
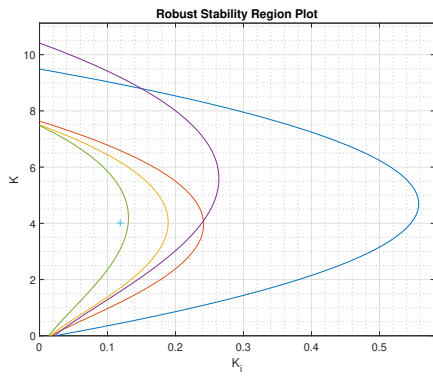
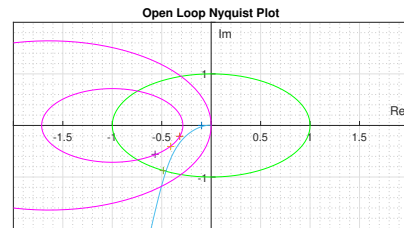


Figure 59: Nyquist plot of the $P_2(s)$ with the design criteria set: $GM \geq 10$, $PM \geq 62^\circ$, $M_s \leq 1.4$

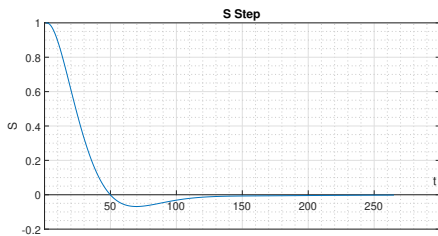


(a) Robust stability region plot: Process $P_2(s)$

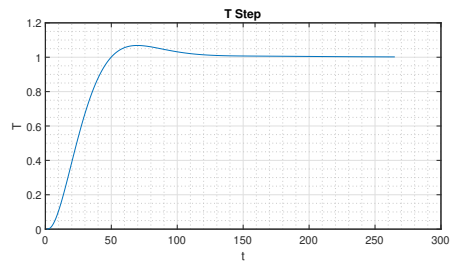


(b) Nyquist plot of the open loop

Figure 60: Robust stability region plot and open loop Nyquist plot: Process $P_2(s)$, Design criteria: $GM \geq 10$, $PM \geq 62^\circ$, $M_s \leq 1.4$, $[K, K_i] = [4.0147, 0.1187]$

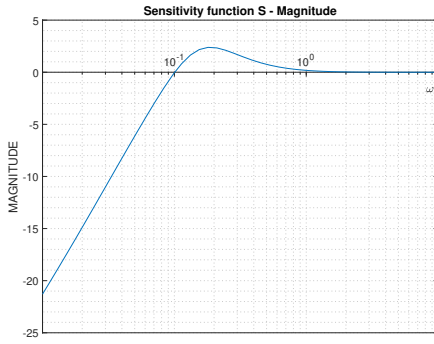


(a) $S(s)$ step response

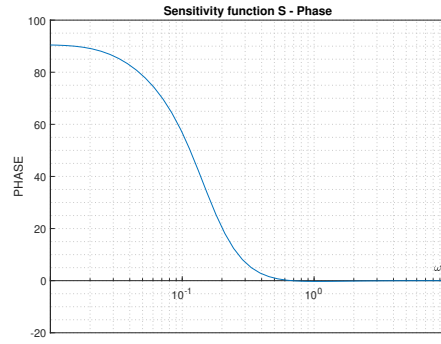


(b) $T(s)$ step response

Figure 61: Step responses of the sensitivity functions $S(s)$, $T(s)$: Process $P_2(s)$, Design criteria: $GM \geq 10$, $PM \geq 62^\circ$, $M_s \leq 1.4$, $[K, K_i] = [4.0147, 0.1187]$

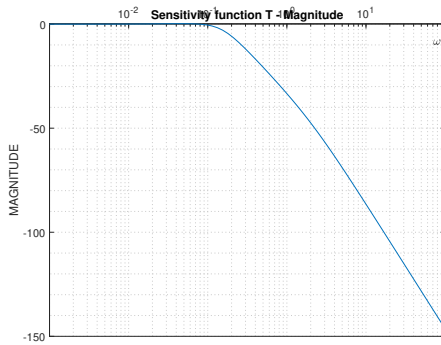


(a) $S(j\omega)$ magnitude

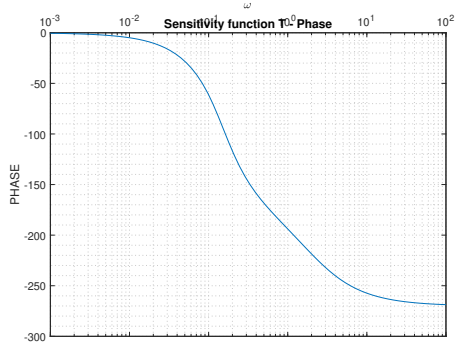


(b) $S(j\omega)$ phase

Figure 62: Bode diagram of the sensitivity function $S(j\omega)$: Process $P_2(s)$, Design criteria: $GM \geq 10$, $PM \geq 62^\circ$, $M_s \leq 1.4$, $[K, K_i] = [4.0147, 0.1187]$



(a) $T(j\omega)$ magnitude



(b) $T(j\omega)$ phase

Figure 63: Bode diagram of the sensitivity function $T(j\omega)$: Process $P_2(s)$, Design criteria: $GM \geq 10$, $PM \geq 62^\circ$, $M_s \leq 1.4$, $[K, K_i] = [4.0147, 0.1187]$

From the time and frequency response we can claim that the designed PI controller $C(s) = 4.0147 + \frac{0.1187}{s}$ satisfies the given design requirements.

In this GUI, it would be possible to design a controller for a range of work points. We would perform the linearization of the process in those work points. Next, we would insert the linearized processes into the app. Finally, the controller would be designed as an intersection of the regions for each design requirement for all linearized processes.

In this two steps the GUI validation has been completed first for the set of processes with time delay, second for the practical example of real physical system.

7 Conclusion

In this section we will discuss the results of this bachelor thesis, and point out which tasks could be a part of the future works.

7.1 Thesis results summary

The main aim of this thesis was to study robust stability regions theory and the current state of the controller tuning tools, and design a graphical user interface which would implement the robust stability regions controller tuning method.

In the first section, we have made brief introduction into the control theory where we described the PID controller and simple feedback control loop. Together with the control theory, we introduced the robust stability regions theory where we explained the controller design using the Nyquist plot shaping. We have shown the detailed step by step derivation of PI controller parameters $K(\omega)$, $K_i(\omega)$ expressions. We have developed an algorithm which helped us visualize the first encirclement of the multidimensional regions in the first quadrant. This method was used to visualize both PI and PI^α robust stability regions. Next, the fractional-order theory has been introduced including the proof of the monotony of $A(\omega)$ and $\varphi(\omega)$ of the essentially monotone processes. By utilizing the Parseval's theorem, we have shown that the monotony in the frequency domain is transmissible into the time domain, proving the monotony of the time response of the essentially monotone processes which would make them good robust stability region controller design method validation set.

In the second section, we have focused on the current state-of-the-art of the online controller tuning tools followed by the current state of the Matlab PID controller tuning methods. We have described the structure and functionality, and pointed out the advantages and disadvantages of each mentioned tool. It has been shown that none of the current tools, except PIDlab, does not implement the robust stability region method for controller design. It was shown that the absence of the robust stability region during the controller design could lead to infinite optimization cycle because the controller does not have to exist for selected design criteria.

In the third section, the GUI implementation in the Matlab environment App Designer was described. Designed GUI uses the Nyquist plot shaping method, robust stability region method, as well as the invented algorithms computing the first quadrant of the first origin encirclement of $K(\omega)$, $K_i(\omega)$ curve. GUI provides the set of controller parameters (if they exist) to the user from which the desired parameters can be selected by mouse click making it interactive. GUI implements the PI controller design and PI^α multidimensional regions visualization.

In the fourth section, we have performed the multi-criteria optimization where the set of controller parameters satisfying the chosen design requirements is given by the intersection region of the robust stability regions for each requirement. The multi-criteria optimization was performed first for one process with multiple criteria, and after that another process with its own design criteria has been added to the first.

In the last step of the thesis, we have validated the designed tool. First, the set of stable non-oscillatory processes with time delay has been selected for the testing. Second, the real physical system consisting of coupled tanks has been selected for the validation. The validation finished successful in every case.

7.2 Future works

Despite the fact that this thesis brought several results, it also laid the foundation for future works.

In the first step, we could include other controller forms such as P, PID, and fractional-order controller $PI^\alpha D^\beta$. The interactivity of the GUI could be improved, more object handles dynamically recomputing the plots by mouse dragging could be added. Next feature we could include is computation of the the optimal parameters from the bitmap of intersection region according to the performance criteria (ITAE, ISAE, ISE etc.). We could also focus on the computation of the 3D regions intersection.

More importantly we would like to focus on implementing the process identification. This would add the possibility to input and create a mathematical model from the measured data. Next, we would focus on connecting the GUI to the target device using communication protocols. On the target device we would be able to simulate time responses of the open and closed loop, and thus validate whether the design form of controller is compatible with the real process. The ideal step would be to create a module which would generate a code containing the control algorithm for the target device.

References

- [1] Karl Johan Åström and Tore Hägglund. *PID controllers: theory, design, and tuning*, volume 2. Instrument society of America Research Triangle Park, NC, 1995.
- [2] Alexandre C Dimian, Costin Sorin Bildea, and Anton A Kiss. *Integrated design and simulation of chemical processes*. Elsevier, 2014.
- [3] Martin Čech. Návrh robustních regulátoru s omezenou strukturou pro systémy neceločíselného řádu, 2008.
- [4] M Čech. Web-based fractional pid controller design: [www. pidlab. com](http://www.pidlab.com). *IFAC-PapersOnLine*, 51(4):563–568, 2018.
- [5] M Čech and M Schlegel. Computing PID tuning regions based on fractional-order model set. *IFAC Proceedings Volumes*, 45(3):661–666, 2012.
- [6] 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.
- [7] Karl Johan Åström, Tore Hägglund, and Karl J Astrom. *Advanced PID control*, volume 461. ISA-The Instrumentation, Systems, and Automation Society Research Triangle Park, 2006.
- [8] GH Hardy and EC Titchmarsh. A note on parseval’s theorem for fourier transforms. *Journal of the London Mathematical Society*, 1(1):44–48, 1931.
- [9] Königsmarková J Čech, M and M Schlegel. Robust PID tuning rules for a class of fractional-order processes, 2008.
- [10] J Königsmarková and M Čech. Robust PI/PID parameter surfaces for a class of fractional-order processes. *IFAC-PapersOnLine*, 51(4):763–768, 2018.
- [11] Calerga. <https://calerga.com/products/Sysquake/index.html>. Accessed: 15-03-2021.
- [12] PID tuner. <https://pidtuner.com/#/>. Accessed: 15-03-2021.
- [13] PIDlab. <https://www.pidlab.com/cs/>. Accessed: 15-03-2021.
- [14] Control System Toolbox. <https://www.mathworks.com/products/control.html>. Accessed: 21-03-2021.
- [15] Matlab App Designer. <https://www.mathworks.com/products/matlab/app-designer.html>. Accessed: 27-04-2021.
- [16] Matlab. <https://www.mathworks.com/products/matlab.html>. Accessed: 21-03-2021.
- [17] John G Ziegler, Nathaniel B Nichols, et al. Optimum settings for automatic controllers. *trans. ASME*, 64(11), 1942.
- [18] Miloš Schlegel. Exact revision of the ziegler-nichols frequency response method. In *Proc. of IASTED Int. Conf. on control and applications, Cancun, Mexico*, 2002.
- [19] 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.
- [20] Tomáš Ausberger, Karel Kubíček, Pavla Medvecová, Tomáš Myslivec, and Milan 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. IEEE, 2020.

- [21] Henrik Sandberg and Bo Bernhardsson. A bode sensitivity integral for linear time-periodic systems. *Automatic Control, IEEE Transactions on*, 50:2034 – 2039, 01 2006.
- [22] Miloš Schlegel and Pavla Medvecová. Design of PI controllers: H_∞ region approach. *IFAC-PapersOnLine*, 51(6):13–17, 2018.
- [23] M Čech. Návrh robustních regulátor s omezenou strukturou pro systémy neceločíselného řádu. *Plzeň: Disertační práce, ZČU Plzeň*, 65:67, 2008.
- [24] Karl Johan Åström and Tore Hägglund. Revisiting the Ziegler–Nichols step response method for PID control. *Journal of process control*, 14(6):635–650, 2004.
- [25] Milos Schlegel and Martin Cech. Internet PID controller design: [www. pidlab. com](http://www.pidlab.com). *IBCE*, 4:1–6, 2004.
- [26] Tuning with Control System Tuner. <https://www.mathworks.com/help/slcontrol/tuning-with-control-system-tuner.html>. Accessed: 29-03-2021.