KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Optimal linear feedback controller design

software tools & mechatronic applications

Maarten Verbandt

Dissertation presented in partial fulfillment
of the requirements for the degree of
Doctor of Engineering Science (PhD): Mechanical Engineering

January 2019

Optimal linear feedback controller design

software tools & mechatronic applications

Maarten VERBANDT

Examination committee:
Prof. dr. ir. P. Wollants, chair
Prof. dr. ir. J. Swevers, supervisor
Prof. dr. ir. G. Pipeleers, supervisor
Prof. dr. ir. J. De Schutter
Prof. dr. ir. W. Michiels
Prof. dr. ir. J. Schoukens
Prof. dr. ir. E. Demeester
dr. ir. B. Demeulenaere
(Atlas Copco Airpower n.v.)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Mechanical Engineering

January 2019

Voorwoord

Nog altijd niet schoolmoe en in de overtuiging dat ons op vlak van regeltechniek nog niet alles was verteld, vatte ik zo'n 4,5 jaar geleden een doctoraat aan. Zoals iedereen wist ik toen niet zo heel goed waar ik aan begon. De eerste uren, dagen en weken was het eerder zoeken dan wel onderzoeken. Gelukkig waren Jan en Goele er nog om mij een duwtje in de gepaste richting te geven. Even stilstaan bij hun aanwezigheid gedurende deze periode is dus wel gepast. In de eerste plaats ben ik uiteraard dankbaar dat ze mij met zoveel enthousiasme aan boord van het MECO team hebben gehaald. Wat ik echter nog belangrijker acht is hun manier van omgaan: ze slaagden er in hun expertise op een bescheiden manier over te brengen, zetten de zeilen bij zodat iedereen op professioneel vlak zo goed mogelijk tot zijn recht kwam en hadden eens zoveel aandacht voor het mentale welbevinden van hun onderzoekers. Waarschijnlijk zijn er zo nog dingen waar ik niet eens aan denk omdat ik al die jaren verwend ben geweest en ze nu als evidentie beschouw. Bij deze dus een welgemeende dankjewel voor alle aandacht, tijd en goesting die jullie aan de dag hebben gelegd.

Tijdens de eerst dagen en weken maakte ik natuurlijk ook kennis met de onderzoeksgroep en in het bijzonder met mijn bureaugenoten! Meteen werd ik ingelicht over de lange traditie van lopen en taart eten in 01.023. Niet veel later trok ik mijn loopschoenen aan en namen Keivan en Ruben mij op sleeptouw. Als toen nog ongeoefende loper deden de bergjes in het bos van Heverlee toch een beetje pijn en de brandende septemberzon hielp ook niet echt. Waarschijnlijk is het op aandringen van Ruben, Keivan maar toch vooral ook Friedl geweest dat ik ben blijven doorzetten, wat uiteindelijk zijn vruchten afgeworpen heeft. Met Ruben als vaste buddy werd twee keer per week het bos onveilig gemaakt en kon er gediscussiëerd worden over de meest uiteenlopende zaken. Geweldige momenten waar ik met blijdschap aan terugdenk. Wat ik na mijn vuurdoop nog niet wist was dat Tim, a.k.a. onze Iron Man, dan nog moest komen. Was het nu Tim of Friedl, of allebei? Minstens één van hen heeft me besmet met de triatlon microbe. Een fietstocht op zondag was geen excuus om maandag niet mee te gaan lopen. Uiteindelijk heeft het mij wel onvergetelijke ervaringen opgeleverd met als uitschieters de triatlon van Leuven en Eupen, waar Tim mij tijdens het lopen gezwind dubbelde. Op sportief vlak heeft mijn doctoraat dus een niet te onderschatten impact gehad. Maar zoals reeds aangehaald was er ook de traditie van de taart. Veel uitleg vergt dit niet: één keer per week was er taart, al dan niet zelf gebakken. Misschien is het wel leuk om er even wat cijfers bij te halen. Naar schatting heb ik (en dus ook Tim en Ruben) 40 kg gebak verorberd, wat neerkomt op zo'n 81 000 kcal ofwel een slordige 1120 km lopen. Gelukkig hing er dus een sportieve atmosfeer in het bureau. Bedankt 01.023 voor deze en vele andere aspecten die het leven in onze bureau kleur gaven.

Beetje bij beetje tikte de tijd weg en kreeg mijn werk meer richting, een richting

waar onder anderen Bram interesse in toonde. Hij durfde de samenwerking met de academische wereld aan en zette zich hiervoor ten volle in. Dit leidde zowaar tot een 'consultancy' opdracht. Alleen al door de benaming kreeg ik het lichtjes benauwd. Niet lang na aankomst werden de eerste testen uitgevoerd, hetgeen de spanning nog een tikkeltje opdreef. Even later kwam de bevrijdende bevestiging dat de testen goed verliepen! Een grote opluchting en tevens het begin van een mooie samenwerking. Een speciaal woordje van dank aan het adres van Bram lijkt me dus wel op zijn plaats. Jij hebt me met open armen ontvangen, vertrouwen geschonken en een hoop inzichten gegeven, zowel op technisch als op menselijk vlak, waarvoor een dikke merci.

Omstreeks die tijd moet ook mijn eerste conferentie, de Benelux Meeting, geweest zijn. Meer nog dan onze vakkennis werden tijdens deze conferentie onze teamgeest en cohesie aangescherpt. Overdag kwam MECO trouw naar alle presentaties van zijn leden kijken en 's avonds vloeide de nodige hoeveelheid Duvel en ice tea om de geest toch wat te ontlasten. Dit brengt mij tot een bedankje aan het adres van de hele onderzoeksgroep. Niet alleen waren jullie verantwoordelijk voor de positieve sfeer waarvan ik dagelijks kon genieten, jullie hebben ook de initiële doelstelling van mijn doctoraat ingelost. Van bewegingsplanning over lerende controle tot optimale lineaire controle, elk facet van de regeltechniek werd wel door iemand afgedekt. Door presentaties en discussies hebben jullie mij stukje bij beetje een beter beeld gegeven van wat de wereld van regeltechniek allemaal omvat, hetgeen de reden was waarom ik in de eerste plaats aan mijn doctoraat begon. Dank u! In het bijzonder wil ik ook Laurens bedanken. Als drijvende kracht achter het ROCSIS project heeft hij zich mijn werk eigen gemaakt en de taak op zich genomen mijn werk voort te zetten en te verbeteren. Vertrekken in de wetenschap dat je werk niet voor niets is geweest doet deugd en jij bent daarvan de belichaming.

De jaren vlogen voorbij. Verse krachten kwamen het team versterken en anderen gingen op zoek naar nieuwe uitdagingen. Er werden softwaresprints gelanceerd, projecten op poten gezet en de eerste MECO retreat vond plaats. Voor ik het wist was ik in het laatste stadium van mijn doctoraat aanbeland en werden de lijnen uitgetekend voor het schrijven van deze tekst. Dat laatste werd evenwel uitgesteld naar aanleiding van de op persoonlijk vlak belangrijkste gebeurtenis van 2018: de geboorte van Arthur. Hoewel je invloed op mijn doctoraat beperkt was wil ik ook jou even bedanken, al is het maar om mijn aandacht af te leiden wanneer het schrijven weer eens niet wou vlotten. Als ik dan toch persoonlijk word is het misschien ook de gelegenheid om alle familie en vrienden te bedanken die af en toe de moed hadden om te vragen hoe het met 'het doctoraat' ging en daar een paar minuten later waarschijnlijk alweer spijt van hadden. Een speciale vermelding gaat uiteraard uit naar mama en papa. Papa, jij bent volgens mij toch mee verantwoordelijk voor mijn interesse in het technische, hiervan getuige de bouw van onze waterraket of het zweefvliegtuig. Mama, bedankt om zo lang mijn kleren te willen wassen en mij te leren taart te bakken om mijn

bureau niet teleur te stellen.

Na een paar maanden van slapeloze nachten, niet door doctoraatszorgen maar door Arthur, was de eerste versie van mijn tekst klaar. Dat betekende dat er niet veel later een preliminaire verdediging op het programma stond. Hoewel ik meestal een koele kikker ben als het op dergelijke zaken aankomt, was ik toch niet helemaal ontspannen. Na een korte presentatie maakte de spanning snel plaats voor een boeiende discussie met nieuwe inzichten vanuit verschillende invalshoeken, interessante vragen en terechte opmerkingen. Bij deze wil ik dan ook uitdrukkelijk de jury bedanken voor de moeite die ze duidelijk heeft gedaan om mijn tekst met zorg te lezen en te voorzien van commentaren om deze tot een hoger niveau te tillen.

Bij deze eindigt dan ook mijn verhaal, voorlopig althans. Op 23 januari 2019 wordt het laatste bedrijf opgevoerd. Hoe dat afloopt weet op dit ogenblik niemand, maar iedereen kan erbij zijn. Ik hoop dat jij er ook bent.

Ooit schreef een wijs man “Achter elke doctorandus staat een sterke vrouw”. Hij had gelijk Katrien, maar jij vervulde meer dan die ene rol. Als mijn privé arts bracht je mij keer op keer met een lach het nieuws dat ik weer eens een vaccinatie moest krijgen. Tetanus na een foutje in de afdaling op de mountainbike, gele koorts voor Kenia, tekenencefalitis voor Slovakije, kinkhoest voor Arthur, noem maar op. Niets krijgt mij nu nog klein! En als ik dan toch eens geveld was door de griep kwam jij aandraven met een wonderpilletje diclofenac. Je was ook mijn persoonlijke reisagent. Van Kenia tot Berlijn, van Malta tot Slovakije, van Slovenië (niet te verwarren met Slovakije) tot Ijsland. We bezochten het inmiddels tot steenpuin herschapen Azure Window, je sleurde me mee naar een afgelegen wijnkelder ergens in Slovenië en toonde me de grote trek in de Serengeti. Ik kijk al uit naar wat je nog voor mij in petto hebt. Met je vaardigheden als planner en hamsteraar, zorgde je er voor dat ik ook thuis niets tekort kwam. Ik moest me geen zorgen maken over babykleertjes of een geboortelijst en als je naar de winkel was geweest stonden de chocomelk, drinkyoghurt en een overvloed aan pasta al op mij te wachten. Hopelijk komt aan deze dingen geen eind nu mijn doctoraat afloopt, behalve dan misschien die vaccinaties, dat mag stoppen.

Maarten

Abstract

In view of our never-ending pursuit of maximum welfare, automation has become key in our modern day society. Rapid developments within the flourishing electronics industry boost the applicability of control systems making them truly omnipresent. Although modern electronics are opening the door for more complex control strategies that improve a system's performance, industry clings to its beloved PID controller. How to overcome this reluctance towards advanced control strategies is therefore the central question within part I of this work.

The first reason that comes to mind is the absence of sufficient software support. In view of this observation, the Linear Control Toolbox (Verbandt, Jacobs, Singh, et al., 2018) constitutes the cornerstone of this work. The Linear Control Toolbox is an open-source Matlab-based software package that focuses on advanced feedback controller design strategies and supports the control engineer throughout the entire design procedure. To improve the user experience, the toolbox provides a signal-based modeling language that allows the user to express his or her intentions in a natural way. Also, various identification and controller design algorithms are interfaced in a uniform manner so that different designs are readily explored.

Unfortunately, having decent software support only partially solves the problem. These advanced feedback controller design strategies require a true paradigm shift within the control community. Whereas traditional methods derive closed-loop properties from the open loop, the more advanced methods operate directly on the closed loop. Although this approach should feel more intuitive, decades of open-loop thinking have created a serious bias that has to be conquered. To show the advantages of a closed-loop design paradigm and tip the balance in its favor, several mechatronic case studies are presented and experimentally validated.

During the experimental validation, two open-source software packages have come into existence, which are showcased in part II of this work. To speed-up developments on embedded mechatronic platforms, the MECO-CSI (Verbandt, 2018) toolchain emerged. By providing software for both the embedded platform and a desktop pc, the embedded platform becomes accessible, speeding up the otherwise tedious design phase. ProjectEagle (Van Parys and Verbandt, 2018) is the code name of the second open-source project that was started and is concerned with the problem of indoor localization. By combining image processing techniques and ideas from the internet-of-things paradigm, a distributed localization system that consists of several smart camera modules was developed.

Beknopte samenvatting

Door ons streven naar als maar meer welvaart is automatisatie een belangrijk aspect geworden in de hedendaagse maatschappij. Snelle ontwikkelingen op gebied van elektronica geven de toepasbaarheid van regelsystemen ook nog een duwtje in de rug, waardoor ze bijna letterlijk overal aanwezig zijn. Hoewel moderne elektronica de deur opent voor meer gevorderde controlestrategieën die de huidige systemen kunnen verbeteren houdt de industrie vast aan haar geliefkoosde PID-regelaar. De centrale vraag doorheen deel I van dit werk luidt dan ook: "Hoe kunnen we de terughoudendheid tegenover gevorderde controlestrategieën wegwerken?".

De eerste reden die men voor deze terughoudendheid kan bedenken is de afwezigheid van voldoende ondersteuning op gebied van software. Hiertoe is de Linear Control Toolbox (Verbandt, Jacobs, Singh e.a., 2018) ontwikkeld dewelke de hoeksteen vormt van dit werk. De Linear Control Toolbox is een Matlabtoolbox toegespitst op gevorderde controlestrategieën op basis van terugkoppeling, en biedt ondersteuning doorheen het hele ontwerpproces. Om het de gebruiker zo gemakkelijk mogelijk te maken hanteert deze toolbox een signaal-gebaseerde modeleertaal die natuurlijk aanvoelt. Daarbij biedt de toolbox op een uniforme manier tal van algoritmes aan voor de identificatie van systemen en het ontwerp van regelaars, wat het mogelijk maakt om snel verschillende ontwerpen te vergelijken.

Spijtig genoeg lost een betere software-ondersteuning het probleem slechts gedeeltelijk op. De beoogde gevorderde strategieën voor het ontwerp van regelaars vereisen ook een heuse ommezwaai wat de denkwijze van de controlegemeenschap betreft. Waar de traditionele methodes de karakteristieken van de gesloten lus dienen af te leiden uit de open lus, werken gevorderde methodes rechtstreeks op de gesloten lus. Hoewel deze laatste aanpak veel directer is en dus natuurlijker zou moeten aanvoelen, heeft het jarenlange open-lusdenken gezorgd voor een vooringenomenheid die nu moet overwonnen worden. Om de voordelen van een gesloten-lusaanpak toch duidelijk te maken worden verschillende mechatronische gevalstudies voorgesteld en experimenteel gevalideerd.

Als resultaat van deze experimentele validatie, werden twee open-source softwarepakketten uitgewerkt, dewelke in deel II van deze tekst worden voorgesteld. Om ontwikkelingen op weinig toegankelijke, ingebedde mechatronische platformen te versnellen kwam het MECO-CSI (Verbandt, 2018) softwarepakket tot stand. Door software aan te bieden voor zowel het ingebedde platform als een standaard pc wordt het ingebedde platform toegankelijk gemaakt waardoor het anders moeilijke ontwerp op dergelijk platform wordt versneld. ProjectEagle (Van Parys en Verbandt, 2018) is het tweede open-source project en biedt een antwoord op het probleem van localisatie in gebouwen. Door de combinatie van beeldverwerkingstechnieken en ideeën rond

internet-of-things kwam een gedistribueerd localisatiesysteem tot stand, bestaande uit slimme cameramodules.

Contents

Part I

Optimal Linear Feedback Controller Design

1

Introduction

1.1 Motivation

In view of our never-ending pursuit of maximum welfare, automation has become key in our modern day society. Coal and gas stoves are systematically replaced by central heating systems that are able to maintain a comfortable temperature throughout a winter's day. Dishwashers and washing machines are taking care of time-consuming daily chores, freeing time for amusement and relaxation. Airplanes already display a high degree of autonomy and cars are following in their footsteps.

This increase in degree of automation is reflected by the number of sensors installed in various products. Whereas the first electronic washing machine contained only a sensor to measure the rotational speed of the drum, today's washing machines incorporate a broad variety of sensors such as a temperature sensor for the water, a water level sensor, a sensor to determine the load or even a dirt sensor. Looking at cars, the cost of onboard electronics is increasing from 10% in the 1980s to about 35% today, and is expected to increase even further to about 50% in the year 2030. This is even more striking when considering the fact that electronics are becoming cheaper at an enormous rate.

This correlation between the number of sensors and the level of automation is explained by the fact that automation can be considered the skill of interpreting measurements and taking a suitable action. This process of acting in response to measurements is known as feedback control or closed loop control. The words feedback or closed loop reflect the fact that the measurements are influenced by the action and vice versa. Control signifies the aim to make the system behave as desired. This desire is usually referred to as the reference, for instance a target velocity for a driving car or a target temperature for the water of a washing machine. What action is appropriate given some measurement and reference is determined by the control law or controller and depends heavily on the application. This is where the control engineer comes into play. His or her task is to determine this relation to obtain adequate performance. Though a rich variety of such relations can be thought of, the linear control law claims the lion's share because of its simplicity and sound mathematical foundation.

In the 1980s however, it was proposed to change the design paradigm in order to achieve optimal instead of adequate performance. This approach shows advantages in comparison to the conventional design techniques but was never adopted by industry. This is in sharp contrast to the continuous academic research to improve and extend this methodology. It is clear that a number of challenges prevent this technique from being adopted.

A first challenge is the practical applicability of the methodology itself. Many numerical routines to synthesize optimal control laws exist, but since they mostly originate from academia, they are not adapted to industrial needs. Whereas the job should only consist of coming up with an optimization problem, the industrial practitioner is in addition confronted with the cumbersome task of transforming the problem to an equivalent form that is accepted by the specific routine. The complexity of the routines offered to industry is seemingly of such a level that their industrial value vanishes.

A secondary yet similar challenge arises from the fact that the proposed methodology is model-based. Although high-fidelity models are often available as a result of the system's design stage, they are usually not suited for control because of their nonlinear nature, their excessive size or the modeling errors. Carrying out a linear identification of the system is the safest answer, but comes at the cost of doing additional experiments. Reducing this cost to a minimum is found to be an additional requirement to get the proposed methodology adopted.

The third challenge consists of the paradigm shift that is needed to design controllers. Whereas conventional methods consider the open loop to derive stability and performance properties, the proposed methodology requires the control engineer to think in terms of the closed-loop transfer functions. Although the latter is a direct approach, many years of open-loop thinking made us accustomed to this indirect approach which is now hard to abandon.

1.2 Contributions

This thesis tries to answer the various challenges postulated in section 1.1. The following paragraphs summarize the contributions that were made in view of tackling these challenges.

1.2.1 Development of the Linear Control Toolbox

In an attempt to answer challenges one and two, this thesis presents the Linear Control Toolbox. The Linear Control Toolbox provides a higher level of abstraction, compara-

ble to other design tools such as YALMIP (Löfberg, 2012) or CasADi (Andersson et al., 2018). By doing so, various controller design and identification techniques become accessible via a unified interface that requires less in-depth knowledge and accelerates the process of designing an optimal feedback controller. The lower threshold to adopt this methodology and the reduced development time should boost the appeal of the optimal feedback controller design as a methodology and pave the way for industrial acceptance.

1.2.2 Various mechatronic case studies

In order to tackle the third challenge, a series of mechatronic case studies is being presented in this work. Although the considered setups originate from different domains, a similar design approach applies to each of these cases. This observation should tip the balance in favor of closed-loop design approaches because the latter bring along this high degree of systematicity. As a by-product, closed-loop methods are able to make abstraction of the specific system, which is especially useful for complex applications where open-loop design is not at all straightforward.

1.3 Outline

Following this introduction are five substantive chapters related to the contributions of this thesis. The relation between these chapters is characterized by Figure 1.1.

Chapter 2 introduces the mathematical background of closed-loop controller design to the reader. It does not contain any of the contributions stated in section 1.2, but provides a gentle stepping stone to the remainder of the text. Chapter 3 presents the Linear Control Toolbox, which is considered the first contribution of part I of the thesis. It mainly highlights the features of this toolbox and provides corresponding code examples. Next, three mechatronic case studies are presented, which relate to the second contribution of section 1.2. Chapter 4 describes the design and experimental validation of an LPV controller on a lab-scale overhead crane, chapter 5 repeats this exercise on a high-speed rotordynamic test setup and chapter 6 is concerned with the velocity control of a ball-balancing robot. A conclusion is drawn in the last chapter of this part.

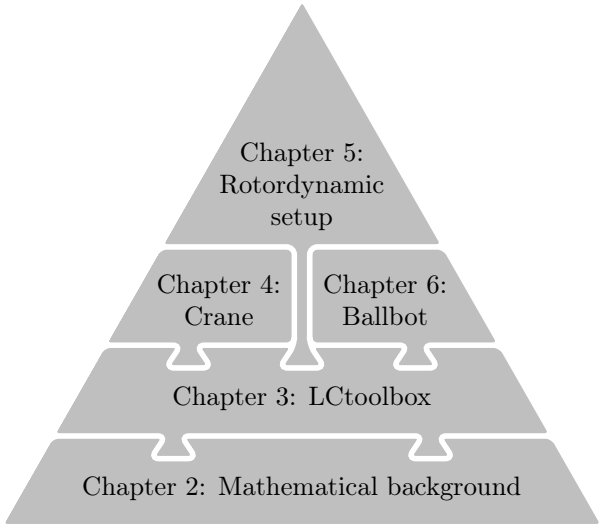


Figure 1.1 · Schematic overview of the relation between the different chapters of part I.

2

Mathematical background

This chapter introduces the reader to the field of optimal feedback controller design. The reader is taken on a journey through "the wonder years 1980–1995 of $\mathcal{H}_\infty$ control" (Meinsma, 2016). Along the way, the mathematical background is gradually built up and practical aspects are highlighted. A strong emphasis is put on globally optimal solutions. Results are shown in a continuous time setting, but their discrete-time counterparts also exist. First, the most basic formulation of the optimal $\mathcal{H}_\infty$ controller design problem is introduced, followed by the most influential solution strategies. Next, the basic problem is extended in various ways, e.g. by allowing constraints or by introducing different performance measures. It is shown how to obtain a convex reformulation of the general problem, under certain assumptions. The third section presents yet another extension in which the system matrices are allowed to vary as a function of some predefined parameter. Again, it is explained how to address this parameter dependency in the controller design. The chapter is closed by introducing other approaches which only partially reformulate the problem or solve the problem as is. Although these problems are nonconvex and only locally optimal solutions are obtained, these methods produce good results in practice.

2.1 The mixed sensitivity problem and its solutions

This section introduces the mixed sensitivity problem, which can be considered the cornerstone of optimal feedback controller design. The mixed sensitivity problem considers the controller to be the solution of an optimization program, hence the term optimal feedback controller design. The mixed sensitivity problem is introduced first, followed by an illustration of the complexity of the optimization problem. Next, the most influential solution strategies are presented, one based on algebraic Riccati equations (AREs) and two formulations in terms of linear matrix inequalities (LMIs). An assessment of these different approaches concludes the section.

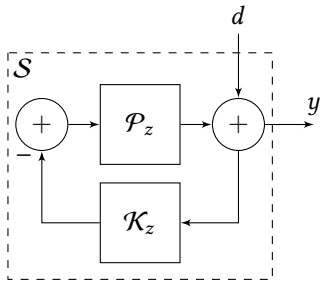


Figure 2.1 · The closed loop studied by Zames. Here, the subscript z refers to Zames.

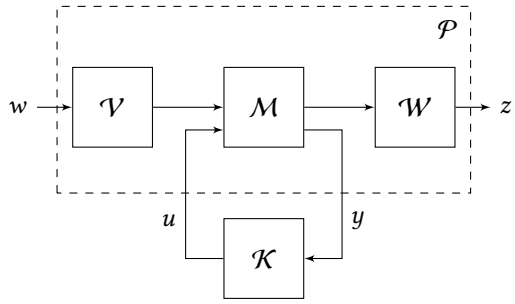


Figure 2.2 · The generalized closed loop as it is known nowadays.

2.1.1 Introduction of the $\mathcal{H}_\infty$ norm and the mixed sensitivity problem

The original formulation of the mixed sensitivity problem dates back to 1981. When studying the influence of a disturbance, d, on the closed-loop system depicted by Figure 2.1, Zames (1981) figured that it is what he called *the sensitivity* that requires attention. To define the sensitivity, the L^2 signal space is introduced. A real-valued signal $s(t)$ is an element of the L^2 signal space if

$$\|s\|_2 \triangleq \int_0^\infty |s(t)|^2 \, \mathrm{dt} < \infty. \tag{2.1}$$

In that case, the total energy of the signal on the interval $[0, \infty[$ is said to be $\|s\|_2$. Based on this notion of energy, he defined the sensitivity, $\bar{S}$, as (2.2), i.e. the maximum energy transfer from some disturbance d to the measured output y.

$$\bar{S} \triangleq \sup \left\{ \|y\|_2 / \|d\|_2 : d \in L^2, \|d\|_2 \neq 0 \right\} \tag{2.2}$$

The importance of definition (2.2) is not to be underestimated, as it is now formally known as the definition of the $\mathcal{H}_\infty$ norm of a system, $\|\cdot\|_\infty$. In Zames' case $\bar{S} = \|\mathcal{S}\|_\infty$ in which $\mathcal{S}$ denotes the transfer function from d to y. Apart from the time domain interpretation, there also exists a frequency domain interpretation (2.3): the $\mathcal{H}_\infty$ norm of a stable system equals the maximum singular value over all frequencies.

$$\|\mathcal{S}\|_\infty = \max_\omega \bar{\sigma}(\mathcal{S}(j\omega)) \tag{2.3}$$

For single-input-single-output (SISO) systems, the maximum singular value simplifies to the maximum magnitude.

His findings also showed that the optimal feedback controller, $\mathcal{K}_z^*$, in terms of withstanding some arbitrary disturbance, minimizes the sensitivity, or more formally:

$$\mathcal{K}_z^* = \underset{\mathcal{K}_z}{\operatorname{argmin}} \left\| \mathcal{S}\left(\mathcal{K}_z\right) \right\|_\infty \tag{2.4}$$

Problem (2.4) is the first occurrence of what is now known as the mixed sensitivity problem. By considering a multidimensional y composed of several signals y_i, the problem can be interpreted as taking a mixture of several sensitivities $\mathcal{S}_i$ into account. This notion is what the mixed sensitivity problem owes its name to.

2.1.2 Formal problem formulation

The fairly concrete case studied by Zames was later generalized to the more generic setting depicted by Figure 2.2. $\mathcal{P}$ is usually referred to as the generalized plant and has a parameterization prescribed by a state space model for which (2.5) is a shorthand notation. Without loss of generality D_{yu}^p is assumed to be zero (Safonov and Limebeer, 1988). Similarly, the control law, $\mathcal{K}$, is parameterized by (2.6). Note that from here on, superscripts refer to a system and subscripts denote inputs and outputs.

$$\mathcal{P} = \left[\begin{array}{c|cc} A^p & B_w^p & B_u^p \\ \hline C_z^p & D_{zw}^p & D_{zu}^p \\ C_y^p & D_{yw}^p & 0 \end{array}\right] \tag{2.5}$$

$$\mathcal{K} = \left[\begin{array}{c|c} A^k & B^k \\ \hline C^k & D^k \end{array}\right] \tag{2.6}$$

The resulting closed-loop dynamics $\mathcal{G} : w \to z$ are described by (2.7).

$$\begin{aligned} \mathcal{G} &= \left[\begin{array}{cc|c} A^p & 0 & B_w^p \\ 0 & 0 & 0 \\ \hline C_z^p & 0 & D_{zw}^p \end{array}\right] + \left[\begin{array}{cc|c} 0 & B_u^p & \\ I & 0 & \\ \hline 0 & D_{zu}^p & \end{array}\right] \left[\begin{array}{cc} A^k & B^k \\ C^k & D^k \end{array}\right] \left[\begin{array}{cc|c} 0 & I & 0 \\ C_y^p & 0 & D_{yw}^p \end{array}\right] \\ &= \left[\begin{array}{c|c} \hat{A}^p & \hat{B}_w^p \\ \hline \hat{C}_z^p & \hat{D}_{zw}^p \end{array}\right] + \left[\begin{array}{c} \hat{B}_u^p \\ \hline \hat{D}_{zu}^p \end{array}\right] \mathcal{K} \left[\begin{array}{c|c} \hat{C}_y^p & \hat{D}_{yw}^p \end{array}\right] \\ &= \left[\begin{array}{c|c} A^g & B^g \\ \hline C^g & D^g \end{array}\right] \end{aligned} \tag{2.7}$$

Indeed, it is readily seen that by considering d to be w, z equal to y, $\mathcal{W}$ and $\mathcal{V}$ both a unit matrix and

$$\mathcal{M} = \begin{bmatrix} I & -\mathcal{P}_z \\ I & -\mathcal{P}_z \end{bmatrix}, \tag{2.8}$$

Figure 2.1 can be considered a special case of Figure 2.2. A subtle difference between Zames' initial interpretation and that of the modern day control engineer is the

understanding of d and y. Zames only considered d to be a perturbation of y. However, a broader interpretation of the concepts disturbance and measurement led to the introduction of exogenous inputs, w, and outputs, z. These need not be actual disturbances or measurements, but can for instance be a reference signal or the actuation signal respectively. This notion transforms the original problem formulation which only considered robustness to a more general tool to shape different closed-loop transfer functions.

The mixed sensitivity problem as we know it today is given by (2.9) and is usually rewritten in terms of a slack variable γ (2.10).

$$\underset{\mathcal{K}}{\text{minimize}} \quad \left\|\mathcal{G}\right\|_\infty \tag{2.9}$$

$$\Updownarrow$$

$$\underset{\gamma,\mathcal{K}}{\text{minimize}} \quad \gamma \tag{2.10a}$$

$$\text{subject to} \quad \left\|\mathcal{G}\right\|_\infty \leqslant \gamma \tag{2.10b}$$

From an optimization point of view, problem (2.9) is generally hard to solve as is. To illustrate this, consider a system described by transfer function (2.11).

$$\mathcal{P} = \frac{85\,(s+1)\,(s+0.1)\left(s^2+2s+43.25\right)}{(s+10)\,(s^2+2s+82)\,(s^2+2s+101)} \tag{2.11}$$

Suppose the loop is closed by a proportional controller, i.e. $\mathcal{K} = D^k$. Let us now assess stability of the resulting closed loop. From classical control theory, it is well known that a system is stable if and only if all poles have a negative real part. The evolution of the closed-loop poles as D^k goes from 0 to ∞ is shown in Figure 2.3. Having a look at Figure 2.3, one can observe a pole pair which leaves the left half-plane for values of $D^k \in [D^-, D^+]$. This essentially means the set of stabilizing controllers consists of two disjoint regions $[0, D^-[$ and $]D^+, \infty[$. This example therefore shows how the set of stabilizing controllers is in general nonconvex (Boyd and Vandenberghe, 2004) which in turn makes (2.9) nonconvex. From Figure 2.4 one can infer that problem (2.9) is also nonsmooth. This figure shows the real part of the two complex conjugate closed-loop poles which are moving towards each other and eventually separate on the real axis as the value of D^k increases. It is observed that the gradient of the real part of these closed-loop poles differs for D^k coming from 0 or from ∞, which makes the stability condition a nonsmooth function of the controller variables. Again, this has a great impact on the solution strategy because standard Newton-type methods can no longer be employed (Boyd and Vandenberghe, 2004).

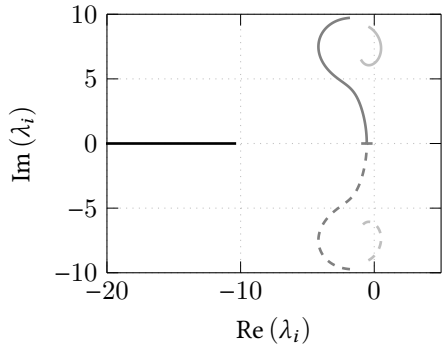


Figure 2.3 · Evolution of the closed-loop poles as a function of the proportional controller gain D^k.

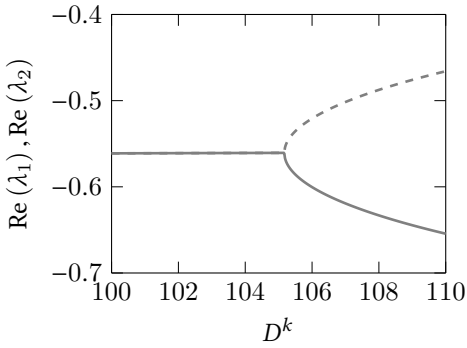


Figure 2.4 · Evolution of the closed-loop complex conjugate poles which move toward each other and become two real poles as D^k increases.

2.1.3 Solution based on AREs

Although problem (2.10) was stated in 1981, it was only in 1989 that Doyle et al. (1989) published a true solution. Rather than solving (2.10) directly, they considered the suboptimal mixed sensitivity problem, i.e. finding an admissible [(1)] controller that satisfies

$$\left\|\mathcal{G}\right\|_\infty \leqslant \gamma. \tag{2.12}$$

[(1)] An admissible controller yields an internally stable and well-posed closed loop (Peters et al., 2009). Admissibility implies the technical notion of well-posedness, so from a practical point of view, 'admissible' could be replaced by 'stabilizing'.

They showed that necessary and sufficient conditions for the existence of an admissible controller that satisfies (2.12) are: $\exists X, Y \in \mathbb{S}^+$ so that (2.13a) - (2.13c) hold, where $\mathbb{S}^+$ denotes the set of positive semi-definite matrices and $\rho(\cdot)$ indicates the spectral radius.

$$A^{p\,T}X + XA^p + X\left(\gamma^{-2}B_w^p B_w^{p\,T} - B_u^p B_u^{p\,T}\right)X + C_z^{p\,T}C_z^p = 0 \tag{2.13a}$$

$$A^pY + YA^{p\,T} + Y\left(\gamma^{-2}C_z^{p\,T}C_z^p - C_y^{p\,T}C_y^p\right)Y + B_w^p B_w^{p\,T} = 0 \tag{2.13b}$$

$$\rho(XY) < \gamma^2 \tag{2.13c}$$

Based on the solution of algebraic Riccati equations (AREs) (2.13a)-(2.13b), which can be solved efficiently e.g. by Arnold and Laub (1984), and condition (2.13c), one is able to set up a simple bisection scheme which searches for the minimal value γ^* of γ. This procedure is sketched by Algorithm 1.

Algorithm 1 · Bisection procedure to attain the optimum of problem (2.10). $\underline{\gamma}$ and $\overline{\gamma}$ denote the lower resp. upper bound on the optimal value γ^*.

$\gamma^* \leftarrow 0, \gamma \leftarrow \left(\underline{\gamma} + \overline{\gamma}\right)/2$
while $\overline{\gamma} - \underline{\gamma} > \Delta\gamma$ **or** $\gamma^* = 0$ **do**
  if $\exists X, Y \in \mathbb{S}^+$ satisfying (2.13a) - (2.13c) **then**
    $\overline{\gamma} \leftarrow \gamma, \gamma^* \leftarrow \gamma$
  else
    $\underline{\gamma} \leftarrow \gamma$
  end if
  $\gamma \leftarrow \left(\underline{\gamma} + \overline{\gamma}\right)/2$
end while

Furthermore, Doyle et al. (1989) show that given a combination of γ, X and Y satisfying (2.13), one of the admissible controllers is given by (2.14). Provided a bisection was carried out to obtain γ^* and the corresponding X^* and Y^*, (2.14) yields an optimal control law that solves problem (2.9).

$$\left[\begin{array}{c|c} A^p + \left(\gamma^{-2}B_w^p B_w^{p\,T} - B_u^p B_u^{p\,T}\right)X + ZC_y^p & -Z \\ \hline -B_u^{p\,T}X & 0 \end{array}\right] \tag{2.14}$$

with

$$Z = -\left(I - \gamma^{-2}YX\right)^{-1} YC_y^{p\,T} \tag{2.15}$$

At this point, it is also important to stress that the solutions provided by Doyle et al. (1989) assume so-called full-order controllers, i.e. the number of controller states equals the amount of states within the plant. This assumption essentially renders the mixed sensitivity problem convex (Apkarian and Noll, 2017), allowing the bisection procedure to obtain a globally optimal solution.

2.1.4 The $\mathcal{H}_\infty$ norm as a matrix inequality

Although Doyle et al. (1989) provided a solution to the mixed sensitivity problem in terms of AREs, researchers kept on investigating other approaches. This led to a completely different branch of solutions based on matrix inequalities. This approach has its roots in the bounded real lemma (Gahinet and Apkarian, 1994), which states the following equivalence:

$$\left\|\mathcal{G}\right\|_\infty < \gamma \;\Leftrightarrow\; \exists P \in \mathbb{S}^+ : Z \prec 0 \tag{2.16}$$

where

$$Z = \begin{bmatrix} A^{g\,T}P + PA^g & PB^g & C^{g\,T} \\ \star & -\gamma I & D^{g\,T} \\ \star & \star & -\gamma I \end{bmatrix}, \tag{2.17}$$

where a $\star$ denotes elements that renders the matrix symmetric. Within this formulation, P can be considered a Lyapunov matrix for the system enforcing stability of $\mathcal{G}$ because $A^{g\,T}P + PA^g \prec 0$.

It is easy to see that by means of the bounded real lemma, the mixed sensitivity problem (2.10) can be reformulated in terms of matrix inequalities, as shown by (2.18).

$$\begin{aligned} \underset{\gamma,\mathcal{K},P}{\text{minimize}} \quad & \gamma && \text{(2.18a)} \\ \text{subject to} \quad & P \succ 0 && \text{(2.18b)} \\ & Z \prec 0 && \text{(2.18c)} \end{aligned}$$

Important to note is that matrix inequality (2.18c) contains the closed-loop matrices. Substituting (2.7) in (2.18c) results in (2.19).

$$Z(\gamma,\mathcal{K},P) = \begin{bmatrix} \hat{A}^{p\,T}P + P\hat{A}^p & P\hat{B}_w^p & \hat{C}_z^{p\,T} \\ \star & -\gamma I & \hat{D}_{zw}^{p\,T} \\ \star & \star & -\gamma I \end{bmatrix} + Z_0(\mathcal{K},P) + Z_0(\mathcal{K},P)^T \tag{2.19a}$$

$$Z_0(\mathcal{K},P) = \begin{bmatrix} P\hat{B}_u^p \\ 0 \\ \hat{D}_{zu}^p \end{bmatrix} \mathcal{K} \begin{bmatrix} \hat{C}_y^p & \hat{D}_{yw}^p & 0 \end{bmatrix} \tag{2.19b}$$

$Z(\gamma,\mathcal{K},P)$ is linear except for a cross term $P\hat{B}_u^p\mathcal{K} + (P\hat{B}_u^p\mathcal{K})^T$. Therefore, matrix inequality (2.18c) qualifies as a bilinear matrix inequality (BMI) and (2.18) classifies as a nonconvex problem.

2.1.5 LMI reformulation via elimination

A key contribution in solving problem (2.18) in an efficient way was proposed by Gahinet and Apkarian (1994) and Iwasaki and Skelton (1994). They applied the projection lemma to reformulate BMI (2.18c) in terms of two linear matrix inequalities (LMIs) which leads to problem (2.20).

$$\underset{\gamma, X, Y}{\text{minimize}} \quad \gamma \tag{2.20a}$$

$$\text{subject to} \quad \begin{bmatrix} X & I \\ I & Y \end{bmatrix} \succ 0 \tag{2.20b}$$

$$\begin{bmatrix} W_1^T & W_2^T & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} A^{p^T} Y + Y A^p & Y B_w^p & C_z^{p^T} \\ \star & -\gamma I & D_{zw}^{p^T} \\ \star & \star & -\gamma I \end{bmatrix} \begin{bmatrix} W_1 & 0 \\ W_2 & 0 \\ 0 & I \end{bmatrix} \prec 0 \tag{2.20c}$$

$$\begin{bmatrix} V_1^T & 0 & V_2^T \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} A^p X + X A^{p^T} & B_w^p & X C_z^{p^T} \\ \star & -\gamma I & D_{zw}^{p^T} \\ \star & \star & -\gamma I \end{bmatrix} \begin{bmatrix} V_1 & 0 \\ 0 & I \\ V_2 & 0 \end{bmatrix} \prec 0 \tag{2.20d}$$

$$\operatorname{rank}\left(\begin{bmatrix} X & I \\ I & Y \end{bmatrix} \right) \leq n_p + n_k, \tag{2.20e}$$

where

$$\begin{bmatrix} W_1 \\ W_2 \end{bmatrix} = \operatorname{null}\left(\begin{bmatrix} C_y^p & D_{yw}^p \end{bmatrix} \right), \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \operatorname{null}\left(\begin{bmatrix} B_u^{p^T} & D_{zu}^{p^T} \end{bmatrix} \right),$$

n_p and n_k are the order of the generalized plant resp. the controller and $Y, X \in \mathbb{S}^{n_p}$ are slack variables that correspond to the upper left subblocks of the Lyapunov matrix P and P^{-1} respectively. The resulting optimization problem is linear apart from the rank constraint (2.20e). However if $n_k \geq n_p$, this constraint is automatically satisfied and problem (2.20) becomes linear. Usually the equality is preferred since it results in the lowest controller order.

The keen observer might have noticed the absence of the controller variables in problem (2.20), hence 'reformulation via elimination'. The optimal controller itself is computed in a second step based on (γ^*, X^*, Y^*). First, two full-column-rank matrices $M, N \in \mathbb{R}^{n_p \times n_k}$ are computed so that (2.21) holds. One of the optimal P^* is then obtained as the unique solution of linear equation (2.22).

$$M N^T = I - X^* Y^* \tag{2.21}$$

$$\begin{bmatrix} Y^* & I \\ N^T & 0 \end{bmatrix} = P^* \begin{bmatrix} I & X \\ 0 & M^T \end{bmatrix} \tag{2.22}$$

Given $\gamma^{*(2)}$, the solution of (2.20), and P^*, constructed via (2.22), the optimal controller parameters are found by solving feasibility problem (2.23), which is now linear.

$$Z(\gamma^*, \mathcal{K}, P^*) \prec 0 \tag{2.23}$$

This approach makes it possible to obtain a full-order controller by solving two linear programs which can be solved efficiently (Boyd et al., 1994).

2.1.6 LMI reformulation via a change of variables

A second LMI reformulation is based on a nonlinear change of controller variables, which was formally introduced by Scherer et al. (1997). They partitioned the Lyapunov matrix P and its inverse, as indicated by (2.24), from which nonlinear transformation of the controller variables (2.25) is derived.

$$P = \begin{bmatrix} Y & N \\ N^T & * \end{bmatrix}, P^{-1} = \begin{bmatrix} X & M \\ M^T & * \end{bmatrix}, \tag{2.24}$$

$$C = \left[\begin{array}{c|c} A^c & B^c \\ \hline C^c & D^c \end{array}\right] = \begin{bmatrix} YA^pX & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} N & YB_u^p \\ 0 & I \end{bmatrix} \mathcal{K} \begin{bmatrix} M^T & 0 \\ C_y^pX & I \end{bmatrix} \tag{2.25}$$

By means of a congruence transformation, (2.19) becomes linear in C, X and Y (2.26).

$$\begin{aligned} \tilde{Z}(\gamma, C, X, Y) &= \Delta^T Z(\gamma, \mathcal{K}, P)\, \Delta \\ &= \begin{bmatrix} A^pX + XA^{pT} & A^p & B_w^p & XC_z^{pT} \\ A^{pT} & YA^p + A^{pT}Y & YB_w^p & C_z^{pT} \\ B_w^{pT} & B_w^{pT}Y & -\gamma I & D_{zw}^{pT} \\ C_z^pX & C_z^p & D_{zw}^p & -\gamma I \end{bmatrix} \\ &\quad + \Lambda(C) + \Lambda(C)^T \end{aligned} \tag{2.26}$$

with

$$\Delta = \begin{bmatrix} X & I & 0 & 0 \\ M^T & 0 & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}, \tag{2.27}$$

$$\Lambda(C) = \begin{bmatrix} 0 & B_u^p \\ I & 0 \\ 0 & 0 \\ 0 & D_{zu} \end{bmatrix} C \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & C_y^p & D_{yw}^p & 0 \end{bmatrix}. \tag{2.28}$$

(2) In a practical implementation, γ^* is slightly relaxed, which is numerically more stable at the expense of obtaining a slightly suboptimal solution.

The resulting reformulated mixed sensitivity problem is stated in (2.29).

$$\underset{\gamma, C, X, Y}{\text{minimize}} \quad \gamma \tag{2.29a}$$

$$\text{subject to} \quad \begin{bmatrix} X & I \\ I & Y \end{bmatrix} \succ 0 \tag{2.29b}$$

$$\tilde{Z}(\gamma, C, X, Y) \prec 0 \tag{2.29c}$$

To obtain the control law, it suffices to invert transformation (2.25), which is linear in the controller variables. C^*, X^* and Y^* follow immediately from the optimization problem whereas M and N should still be computed. Because $PP^{-1} = I$, there exists a direct relation between M and N

$$YX + NM^T = I, \tag{2.30}$$

with a possible solution $M = I$ and $N = I - YX$. From (2.25), it can be seen that M and N have to be invertible to transform C to the original controller $\mathcal{K}$. Therefore, this approach only applies to the search for full-order controllers.

2.1.7 Three solutions - an assessment

Sections 2.1.3, 2.1.5 and 2.1.6 all provide different solutions to the mixed sensitivity problem. One certainly wonders how they relate to one another, both in terms of applicability as well as flexibility.

One can distinguish two different categories: ARE-based and LMI-based. From a numerical perspective, AREs have the advantage of being algebraic which means it is numerically very efficient to obtain a solution (Arnold and Laub, 1984). On the other hand, LMIs are essentially linear optimization programs for which efficient numerical routines also exist. Moreover LMIs rely on the flexible framework of optimization, which makes it possible to generalize the mixed sensitivity problem by for instance introducing extra constraints. Section 2.2 will elaborate on this thought. Also, LMI formulation (2.29) can be extended to synthesize linear parameter varying (LPV) controllers, which are introduced in section 2.3. The LMI reformulation discussed in 2.1.5, though nonconvex, allows the synthesis of reduced-order controllers via a rank constraint whereas the AREs only address the problem from a full-order perspective. More on the design of reduced-order controllers is found in section 2.4. A summary of this assessment is provided by Table 2.1.

	ARE	**LMI** *elimination*	**LMI** *change of variables*
mixed sensitivity			
convex	yes	yes	yes
conservative	no	no	no
measure	$\mathcal{H}_\infty$	$\mathcal{H}_\infty$	$\mathcal{H}_\infty$
multi-objective			
convex		yes	yes
conservative		yes: LS	yes: LS
measure		$\mathcal{H}_\infty$	$\mathcal{H}_\infty$, $\mathcal{H}_2$, region, ...
reduced order			
convex		no	
conservative		yes: LS	
measure		$\mathcal{H}_\infty$	
parameter varying			
convex			yes
conservative			yes: LS and relaxation
measure			$\mathcal{H}_\infty$, $\mathcal{H}_2$, region, ...

Table 2.1 · Overview of the properties of the three approaches that are used to solve the mixed sensitivity problem and their applicability to other problems. Within this table, LS denotes Lyapunov shaping.

2.2 The multi-objective problem and its solutions

This section introduces the first extension to the mixed sensitivity problem: the multi-objective problem. The insights underlying this extension are presented first, followed by a formal formulation of the optimization problem. Again various solutions from literature are presented, with a multi-objective $\mathcal{H}_\infty$ problem as a running example. This example illustrates how a reformulation via a change in variables is the only practical approach to reformulate the general multi-objective problem.

2.2.1 Extensions to the mixed sensitivity problem

Even before the mixed sensitivity problem was solved, researchers proposed extensions to the original formulation, as described by the following paragraphs.

The multi-channel approach

Kwakernaak (1983) realizes that the original formulation in terms of robustness is readily extended to multiple channels. He proposes to take both the sensitivity and the complementary sensitivity, $\mathcal{T} = I - \mathcal{S}$, into account. On the one hand, $\mathcal{S}$ takes care of robustness as well as the closed-loop bandwidth. On the other hand, $\mathcal{T}$ can be exploited to enforce roll-off on the controller or reduce the influence of sensor noise on the closed loop.

Weighted channels

Kwakernaak (1983, 1993) also introduced the use of filters to shape the closed-loop transfer functions as desired. These filters, also referred to as weights, are introduced before an input or behind an output of interest, as shown in Figure 2.2. For example, $\mathcal{V}$ and $\mathcal{W}$ modify the considered model $\mathcal{M}$:

$$\mathcal{P} = \begin{bmatrix} \mathcal{W} & 0 \\ 0 & I \end{bmatrix} \mathcal{M} \begin{bmatrix} \mathcal{V} & 0 \\ 0 & I \end{bmatrix}. \tag{2.31}$$

The resulting system $\mathcal{P}$ is commonly referred to as the augmented plant. These weights can originate from a rigorous theoretical analysis, typically related to robust stability (Skogestad and Postlethwaite, 2001), but they can also be seen as design parameters. To understand the latter, consider the transfer function

$$\mathcal{T}_{zw} : z \to w = \mathcal{W}\mathcal{G}\mathcal{V}, \tag{2.32}$$

which arises when the loop is closed by controller $\mathcal{K}$. In case of a 1-dimensional w and z, $\mathcal{V}$ can be assumed 1 without loss of generality. An analysis of the constraint $\left\|\mathcal{T}_{zw}\right\|_\infty \leqslant 1$ now shows that:

$$\begin{aligned} \left\|\mathcal{W}\mathcal{G}\right\|_\infty \leqslant 1 &\Leftrightarrow \left|\mathcal{W}\left(j\omega\right)\mathcal{G}\left(j\omega\right)\right| \leqslant 1,\ \forall\omega \\ &\Rightarrow \left|\mathcal{W}\left(j\omega\right)\right|\left|\mathcal{G}\left(j\omega\right)\right| \leqslant 1,\ \forall\omega \\ &\Leftrightarrow \left|\mathcal{G}\left(j\omega\right)\right| \leqslant 1/\left|\mathcal{W}\left(j\omega\right)\right|,\ \forall\omega \end{aligned} \tag{2.33}$$

From (2.33), it follows that $\mathcal{W}^{-1}$ can be interpreted as a frequency-dependent bound on the transfer function $\mathcal{G}$, which is graphically illustrated in Figure 2.5. This notion allows the control engineer to design a weight, such that a closed-loop transfer function of interest behaves as desired. Take for instance the sensitivity defined by Zames as the transfer function from input d to output y, from here on denoted by $\mathcal{T}_{yd}$. At low frequencies, the controller should suppress disturbances and so $\mathcal{T}_{yd} \to 0$ for $\omega \to 0$. This behavior can be enforced by choosing $\mathcal{W} = a/s^n$. In this case, the design engineer has to choose an appropriate order n as well as the cross-over frequency of the weight, which is tuned via a.

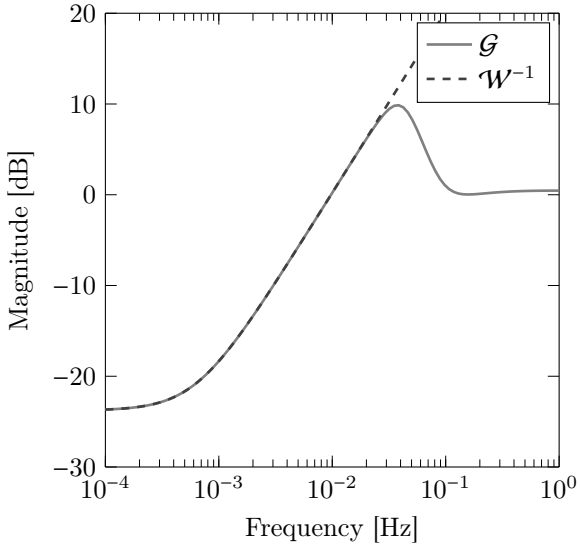


Figure 2.5 · Graphical illustration of the inverse of the weights acting as constraints for a $\mathcal{H}_\infty$ controller design. The transfer function (solid line) is constrained by the inverse of the weight (dashed line).

It is important to note that the weight, although not physically present, is considered a part of the generalized plant. However, since the controller has no influence on the stability of the weights, the generalized plant is not stabilizable if a weight contains an unstable pole. Therefore, the poles in $s = 0$ are typically shifted to the left half-plane, e.g. a/s^n becomes $a/(s+\epsilon)^n$ with $\epsilon > 0$, although Köroğlu (2013) proposes a way to overcome this difficulty.

Allowing constraints

Another interesting idea was proposed by Hara and Katori (1985). They indicate the need to impose constraints on the original mixed sensitivity problem. Indeed, in a practical setting the control loop almost always has to obey certain specifications and only the remaining freedom can be used to optimize some other criteria.

Multiple performance measures

Also new measures for a system's performance were suggested. The original formulation in terms of the $\mathcal{H}_\infty$ norm was mainly driven by the robustness

characterization. However, one can come up with other measures that are useful in a variety of settings. One of the more common alternatives is a system's $\mathcal{H}_2$ norm, which can be interpreted as the total output energy in response to a series of impulses spanning the input space (Stoorvogel, 1993). Also worth mentioning are the LMI regions proposed by Chilali and Gahinet (1996) which can be employed to constrain the poles of the closed loop in specific regions of the complex plane. This is useful to for instance impose sufficient damping within the closed loop or ensure a minimum natural frequency of the closed-loop poles. Other more exotic measures such as the peak of the impulse response, settling time or peak-to-peak gain are discussed in Scherer et al. (1997) but these usually result in conservative results and hence are not common.

2.2.2 The multi-objective mixed sensitivity problem

The combination of all of these insights and ideas leads to a general formulation of the optimal controller design often referred to as the multi-objective mixed sensitivity problem (2.34), which is immediately formulated in terms of the performance slack variables π_i. The sets $\mathbb{I}_o$ and $\mathbb{I}_c$ denote the indices related to the objectives and the constraints respectively.

$$\begin{aligned} \underset{\pi_i, \mathcal{K}}{\text{minimize}} \quad & \sum_{i \in \mathbb{I}_o} \alpha_i \pi_i \\ \text{subject to} \quad & f_i\left(\mathcal{G}_{z_i w_i}\right) \leqslant \pi_i, \quad \forall i \in \mathbb{I}_o \cup \mathbb{I}_c \\ & \pi_i = \pi_{i0}, \qquad\quad \forall i \in \mathbb{I}_c \end{aligned} \tag{2.34}$$

Problem (2.34) makes it possible to weigh multiple objectives expressed in terms of different performance measures f_i, applied to different input-output channels $\mathcal{G}_{z_i w_i}$. Changing the values of α_i moves the optimal controller along the Pareto-optimal curve. A set of constraints is expressed in a similar fashion where the performance is bounded by π_{i0}. Solving this optimization problem is obviously more challenging than the traditional mixed sensitivity problem (2.10). However, the solutions presented in 2.1.3, 2.1.5 and 2.1.6 might serve as a starting point to tackle this more general formulation.

To make the following sections more tangible, the example of a multi-objective $\mathcal{H}_\infty$ controller design is addressed here. The goal is to optimize the $\mathcal{H}_\infty$ performance of input-output channel $\mathcal{G}_{z_1 w_1}$, subject to an $\mathcal{H}_\infty$ constraint on $\mathcal{G}_{z_2 w_2}$.

$$\underset{\pi_1, \mathcal{K}}{\text{minimize}} \quad \pi_1 \tag{2.35a}$$

$$\text{subject to} \quad \left\|\mathcal{G}_{z_1 w_1}\right\|_\infty \leqslant \pi_1 \tag{2.35b}$$

$$\left\|\mathcal{G}_{z_2 w_2}\right\|_\infty \leqslant \pi_2 \tag{2.35c}$$

2.2.3 Solution based on AREs

Sadly, this section is fairly short because to the best of my knowledge, there exists no general solution of the multi-objective mixed sensitivity problem in terms of AREs. The main reason is that the majority of performance criterions described in 2.2.1 cannot be described in terms of AREs. In that regard, the $\mathcal{H}_\infty$ and $\mathcal{H}_2$ system norms are the exceptions, leading to some special cases for which a solution in terms of AREs does exist. The most important are the $\mathcal{H}_\infty$ design problem as discussed in 2.1.3 and its $\mathcal{H}_2$ counterpart. The solution to the latter is for instance also described in Doyle et al. (1989). Moreover, there are even results for the mixed $\mathcal{H}_2$ /$\mathcal{H}_\infty$ problem, as presented by Bernstein and Haddad (1989).

2.2.4 LMI reformulation via elimination

As already illustrated by section 2.1.5, the projection lemma is capable of eliminating the controller variables from a single matrix inequality. This property makes it suited to reformulate the simple mixed sensitivity problem in terms of a set of LMIs. However, this elegance disappears when other performance criteria are considered because the majority of performance criteria requires multiple matrix inequalities to hold. Likewise, considering several objectives and constraints complicates the controller synthesis because this gives rise to a multitude of matrix inequalities, even if the performance indicators are all the same. A notable exception is the problem formulated in (2.35). Provided a single Lyapunov matrix is considered, it is possible to combine the matrix inequalities that impose (2.35b) and (2.35c) into one matrix inequality. This makes the elimination of controller variables applicable again.

This discussion shows that the elimination of controller variables is a neat approach when only $\mathcal{H}_\infty$ norms are involved, but breaks down when other performance criteria are considered.

2.2.5 LMI reformulation via a change of variables

The second approach to reformulate problem (2.35) in terms of LMIs is based on the nonlinear change of variables (2.25). Because this approach effectively linearizes each nonlinear matrix inequality, it is readily applied to problem (2.35), resulting in a tractable formulation, provided the same Lyapunov matrix P is used for all performance measures. The latter is understood from (2.25): since the controller $\mathcal{K}$ depends on the Lyapunov matrix P via relation (2.24), introducing a different Lyapunov matrix for each performance measure would lead to different controllers, which is of course not a feasible solution. The choice of employing a single Lyapunov matrix might render the solution conservative but seems a stringent requirement to

obtain a convex formulation: the same requirement already appeared when applying the projection lemma to problem (2.35).

Moreover, Scherer et al. (1997) point out that the same change in variables is readily applicable to a rich set of performance measures. This property makes the reformulation based on a change of variables the preferred method in a general multi-objective setting.

2.3 Extension to LPV controller design

A very interesting class of models is that of the linear parameter varying (LPV) models. These models are structured similarly to LTI models, but incorporate nonlinearities under the form of a, possibly time varying and multidimensional, parameter $\varphi \in \Phi$, as indicated by equation (2.36).

$$\mathcal{P} = \left[\begin{array}{c|c} A(\varphi) & B(\varphi) \\ \hline C(\varphi) & D(\varphi) \end{array}\right] \tag{2.36}$$

The set of possible parameter trajectories Φ is usually characterized by a pair of bounds on the parameter φ and its time derivative $\dot{\varphi}$:

$$\Phi = \left\{\varphi \in L^2 \mid (\varphi(t), \dot{\varphi}(t)) \in \mathbb{K}, \forall t \in [0, \infty[\right\}, \tag{2.37}$$

where

$$\mathbb{K} = \left[\underline{\varphi}_1, \overline{\varphi}_1\right] \times \cdots \times \left[\underline{\varphi}_{n_\varphi}, \overline{\varphi}_{n_\varphi}\right] \times \left[\underline{\dot{\varphi}}_1, \overline{\dot{\varphi}}_1\right] \times \cdots \times \left[\underline{\dot{\varphi}}_{n_\varphi}, \overline{\dot{\varphi}}_{n_\varphi}\right] \tag{2.38}$$

with n_φ the dimension of the parameter vector.

LPV systems can also be regarded as a generalization of a classical interpolated system, which is the current industrial standard for parameter varying systems (Gahinet and Apkarian, 2013). The wide acceptance of interpolated control laws is rooted rather in engineering practice than in mathematical foundation. Though the interpolation of locally stable LTI controllers might be intuitive, the stability of the closed loop under arbitrary parameter variations is not guaranteed. To assess the stability of the closed loop, one usually resorts to extensive simulations for different parameter trajectories (Rugh and Shamma, 2000).

Modern LPV controller design can, because of its structure, rely on a similar wealth of controller analysis and synthesis tools as the LTI case. This makes it possible to for instance ensure stability and performance of the closed loop for arbitrary parameter variations at the design stage, instead of assessing these after the controller design. Because of these appealing properties, the abundant research in LPV controller design comes as no surprise (see Hoffmann and Werner (2015) for an excellent survey).

The following sections will give a short overview of how the LMI formulations from section 2.2.5 also lead to a solution of the LPV controller design problem.

2.3.1 LMI reformulation in an LPV setting

Once again, matrix inequalities ensure stability and performance of the closed loop. However due to the parameter dependency, the matrix inequalities differ slightly from their LTI counterparts. More specifically, because the Lyapunov matrix depends on a varying parameter, the stability condition becomes (Chen and Tu, 1995):

$$\exists P : \mathbb{R}^{n_\varphi} \rightarrow \mathbb{S}^+ : A^g(\varphi)^T P(\varphi) + P(\varphi) A^g(\varphi) + \frac{\partial P}{\partial \varphi}\dot{\varphi} \prec 0, \ \forall (\varphi, \dot{\varphi}) \in \mathbb{K}. \tag{2.39}$$

This change is easily included in the performance LMIs, for instance, the matrix inequalities in (2.17) that guarantee $\mathcal{H}_\infty$ performance now become (2.40).

$$P(\varphi) \succ 0 \tag{2.40a}$$

$$\begin{bmatrix} A^g(\varphi)^T P(\varphi) + P(\varphi) A^g(\varphi) + \frac{\partial P}{\partial \varphi}\dot{\varphi} & P(\varphi) B^g(\varphi) & C^g(\varphi)^T \\ \star & -\gamma I & D^g(\varphi)^T \\ \star & \star & -\gamma I \end{bmatrix} \prec 0 \tag{2.40b}$$

As already established in section 2.2, the nonlinear change of controller variables is the most suited methodology to tackle the multi-objective controller design problem. However, because the stability condition changed, the transform proposed in (2.25) no longer linearizes the matrix inequality. However the slightly altered version (2.41) does so (Apkarian and Adams, 1998). Note that in case of a constant Lyapunov matrix, (2.41) equals (2.25).

$$C = \begin{bmatrix} YA^pX + X\dot{Y} + N\dot{M} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} N & YB_u^p \\ 0 & I \end{bmatrix} \mathcal{K} \begin{bmatrix} M^T & 0 \\ C_y^p X & I \end{bmatrix} \tag{2.41}$$

Making abstraction of the actual matrix inequalities, the nonlinear change of variables always results in LMIs

$$L\left(\mathcal{P}(\kappa), C(\kappa), X(\kappa), Y(\kappa), \dot{X}(\kappa), \dot{Y}(\kappa)\right) \succ 0 \tag{2.42}$$

in terms of the plant $\mathcal{P}$, the controller C and the slack variables X and Y and their derivatives $\dot{X}$ and $\dot{Y}$, with κ as a shorthand notation for $(\varphi, \dot{\varphi})$. The principal difficulty of LPV controller design is to satisfy (2.42) for all $\kappa \in \mathbb{K}$, making (2.42) infinite dimensional. This problem can be considered a robust optimization problem to which standard relaxation techniques apply. The goal is to construct a finite set of constraints that is sufficient for the infinite dimensional constraint to hold, or more formally:

$$L_i \succ 0, \forall i \in [1, m] \ \Rightarrow \ L(\kappa) \succ 0, \forall \kappa \in \mathbb{K}. \tag{2.43}$$

If such a set can be constructed, the LPV design problem boils down to again solving a set of LMIs. The following sections will give an overview of such relaxation techniques.

2.3.2 Sum-of-squares relaxations

One of the most intuitive relaxation techniques is called the sum-of-squares (SOS) method. As the name suggests, the goal is to certify positivity of a polynomial $l(\kappa)$ by decomposing it in a sum of squared polynomials:

$$l(\kappa) = \sum_i l_i(\kappa)^2 . \tag{2.44}$$

If it is possible to obtain such a representation of $l(\kappa)$, then obviously $l(\kappa) \geqslant 0,\ \forall \kappa \in \mathbb{R}^{n_\kappa}$. Because positivity is usually not a global requirement, extensions were formulated to assert local positivity only (Putinar, 1993), which reduces conservatism. Scherer and Hol (2006) extended these results to the positive definiteness of polynomial matrices. Lasserre (2009) shows how finding a SOS representation with respect to a given monomial basis can be cast into a semidefinite optimization problem, making the SOS approach a valuable instrument to obtain a tractable reformulation of (2.43).

2.3.3 Spline relaxations

Another interesting method is based on a spline parameterization. Suppose

$$l(\kappa) = \sum_i l_i b_i(\kappa), \tag{2.45}$$

where $\{b_1, \ldots, b_n\}$ is a basis on the domain $\mathbb{K}$. Suppose $b_i(\kappa) > 0, \forall \kappa \in \mathbb{K}$, then it is easy to see that:

$$l_i > 0, \forall i \in [1, m] \implies l(\kappa) > 0, \forall \kappa \in \mathbb{K} \tag{2.46}$$

Moreover (2.45) and (2.46) are readily extended to matrix valued splines and matrix inequalities, resulting in (2.43). The resulting conditions and their conservatism depend on the choice of basis functions. It can be shown that using a Bernstein basis is equivalent to the well established Pólya relaxation (Polya, 1928; De Caigny et al., 2009; Hilhorst, Pipeleers, Michiels, Oliveira, et al., 2016). However, recent work has shown the potential of a B-spline basis. Whereas the Bernstein basis can only be extended by elevating the degree of the basis functions, B-splines also allow knot insertions that lead to a faster reduction in conservatism (Van Loock et al., 2016; Hilhorst, Lambrechts, et al., 2016; Lambrechts and Pipeleers, 2017).

2.4 Nonconvex controller design

A great deal of attention has gone to the *convexification* of various optimal controller synthesis problems. As already stated in section 2.1.3, a convex problem can only be obtained when a full-order controller is considered. This immediately closes the door for another interesting design problem, i.e. structured controller design. As the name suggests, the goal is to optimize a controller with predefined structure. This comprises the design of reduced-order controllers[3] as well as the optimization of a parameterized controller such as the infamous PID controller (Ziegler and Nichols, 1993), an observer-based controller (Luenberger, 1971) or a decentralized controller.

As section 2.1.5 already showed, it is possible to obtain a reformulation of the mixed sensitivity problem in terms of LMIs and a rank constraint. Taking this rank constraint into account explicitly allows the synthesis of reduced-order controllers, at the expense of convexity. Examples of research in this direction can be found in Grigoriadis and Skelton (1996) and Orsi et al. (2006).

Section 2.2.4 already indicated that this reformulation is only obtained in case of $\mathcal{H}_\infty$ norms. Moreover, this reformulation does not allow imposing an arbitrary structure. Therefore, it is reasonable to omit the reformulation in terms of LMIs and solve the BMIs directly. In that case, the controller's state space matrices are directly optimized, which allows the designer to fix the controller's structure up front. An application of this approach is for instance described by Henrion et al. (2005) and Tran Dinh et al. (2012).

Solving the optimal feedback controller design problem as is, i.e. without reformulation in terms of matrix inequalities or transformation of the optimization variables, is another option. As a consequence, the optimization problem is not only nonconvex, but also nonsmooth, as already indicated by section 2.1.2. Because objectives and constraints are allowed to be nonsmooth and nonconvex, there is virtually no restriction on the permitted performance measures. Moreover, the number of optimization variables is kept to a minimum because: i) no slack variables such as a Lyapunov matrix are used and ii) the order of the reduced-order controller can be chosen smaller than its full-order counterpart, which is especially significant in case of a high-order plant. Different solvers to tackle these nonconvex nonsmooth optimization problems have been developed. `Hinfstruct` is mainly based on Apkarian and Noll (2006), and has been designed especially for the design of feedback controllers. Another package is called `HIFOO` (Burke et al., 2006; Arzelier et al., 2010), which is a frontend for the generic solver `HANSO`. Although the algorithms should be capable of synthesizing structured controllers, the interface only supports the design of reduced-order controllers. Recently, a novel solver, called `GRANSO` (Curtis et al., 2017), was

[3] Sufficient LMI conditions do exist for this design problem, e.g. Trofino (2009) and Hilhorst et al. (2015)

developed and applied to the synthesis of reduced-order controllers (Benner et al., 2018). Although these solvers only guarantee local optimality, the solutions they provide are usually very practical (Apkarian and Noll, 2017).

2.5 Conclusion

This chapter introduced the reader to the field of optimal feedback controller design. Emphasis was put on obtaining convex reformulations of the original nonconvex and nonsmooth problem. It is illustrated how the nonlinear change of controller variables proves to be the most versatile of all approaches since it is capable of not only solving the mixed-sensitivity problem, but also the multi-objective problem, possibly in an LPV setting. For the sake of completeness, nonsmooth and nonconvex approaches to the optimal feedback controller design were also included in the discussion.

3

Linear Control Toolbox

This chapter presents the Linear Control Toolbox (Verbandt, Jacobs, Singh, et al., 2018), a freely available Matlab-based software package primarily focused on the design of optimal feedback controllers for LTI and LPV systems. The first section motivates the development of this new software package. Next, the different modules which reside within the toolbox are introduced: i) the modeling module, which handles general system manipulations ii) the identification module to extract an analytical model from measured data and iii) the controller design module. To conclude, the new toolbox is compared to the alternatives by means of an LPV case study. This chapter is based on Verbandt, Jacobs, Turk, et al. (2018).

3.1 Aim and scope

Although optimal feedback controller design dates back to the 1980s, the majority of practitioners is reluctant to adopt this *new* paradigm, the reason being the absence of facilitating software tools. Casting the controller design into an optimization problem should be the only burden of the control engineer, but various other intricacies end up on the designer's plate, e.g. constructing the generalized plant or choosing a suitable numerical optimization routine.

The motivation for this new toolbox is twofold. First, existing tools are primarily focused on control, tailored to a specific design problem. This forces the user to switch toolboxes to for instance identify a system or to design a controller. On the contrary, the Linear Control Toolbox supports the control engineer throughout the different steps involved in controller design both in an LTI and an LPV setting. It provides the functionality to identify a system, facilitates general manipulations with systems, enables the user to declare a control configuration and a list of specifications, automatically solves the optimal controller design problem and provides frequency and time domain validation tools. The second driver is to support recent developments in LPV controller design, based on B-splines as described by Hilhorst, Lambrechts, et al. (2016). This method allows parameter dependencies to be described by B-splines which are more flexible than the traditionally employed polynomials. Since B-splines

encompass polynomials, traditional analysis and design methods can still be interfaced with this new toolbox.

3.2 Modeling module

At the core of the presented toolbox resides a general system modeling module. This module adopts a new design paradigm which makes a clear distinction between systems and models. A system reflects a physical entity, e.g. a pump, an electrical circuit or a chemical process. All of these have inputs and outputs which are intrinsically related. This relationship is never truly known but one or more models of the same system might be available. Therefore a system can be regarded as a collection of models which all refer to the same physical entity. The advantage of this approach is that different models can be consulted when most relevant. For instance, the controller design might be based on a simplified linear model whereas a time-domain simulation of a high-fidelity nonlinear model is preferable for a validation. Using a custom modeling module also gives the opportunity to introduce new model classes such as a general nonlinear model, a gridded (LPV) model or a parameter dependent state-space model.

The following paragraphs shed a brighter light on the toolbox's understanding of systems and signals, models and parameters, and their mutual relation.

3.2.1 Systems and signals

Within the Linear Control Toolbox, systems and signals serve the purpose of describing a configuration, i.e. how different components interact with each other. Systems are created via the function call

```
> P = IOSystem(ni,no)
```

where `ni` and `no` denote the number of inputs and outputs. Inputs and outputs are internally stored as a custom `Signal` object and can be accessed via:

```
> P.in, P.out
```

The user is also encouraged to use signal aliases which make the code easier to read:

```
> u = P.in, y = P.out
```

Moreover, new signals can be constructed either via the `Signal()` call or as a linear combination of other signals, for example:

```
> r = Signal(n)
```

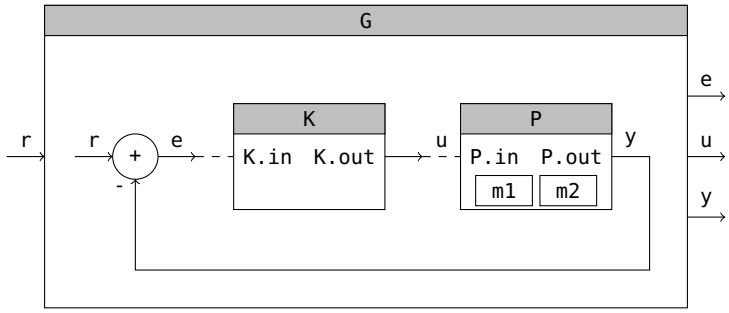


Figure 3.1 · Schematic of the plant constructed throughout section 3.2. Systems `P` and `K` are connected to form system `G` with input `r` and outputs `e`, `u`, and `y`. Two models for `P` are available and added to the system. The system `K` remains empty because no control law is available.

```
> e = r - y
```

where the optional argument `n` indicates the dimension of the signal. Signals serve the purpose of connecting systems to one another. A connection is established by simply equating the signals with the == operator and creating a new system from the subsystems and the connection list:

```
> c1 = (K.in == e)
> c2 = (K.out == u)
> G = IOSystem(P,K,[c1;c2])
```

In this case, the new system `G` is constructed from `P` and `K`, arranged in a standard error feedback configuration. A schematic representation is found in Figure 3.1.

3.2.2 The scheduling parameter

Models provide a relation between the inputs and outputs of a system. This relation can be either time independent or depending on a parameter. To this end, the `SchedulingParameter` is introduced:

```
> p = SchedulingParameter(name,range,rate)
```

This object has three attributes: a name, a range indicating the bounds of parameter values and a range for the rate of variation, keeping track of the minimum and maximum change of the parameter value over time. Special cases include fixed parameter values (rate of variation $[0, 0]$) and unbounded rate of variation, $]-\infty, \infty[$. Under the hood, a `SchedulingParameter` is a B-spline based identity function.

Therefore the `SchedulingParameter` supports all operations that yield piecewise polynomial functions i.e. the addition and multiplication. This allows the user to easily declare complex parameter dependencies, for example the A-matrix of a state-space model:

```
> Ap = [p^2, 1;p^3 - 2*p, 0]
```

3.2.3 Design models

Because the controller design is based on linear parametric models, the most important model class is the general `LFTmod`. It incorporates a pair of linear fractional transforms applied to an underlying descriptor state-space model, as depicted in Figure 3.2. Because E is allowed to be singular, the toolbox is able to handle improper models, which are particularly useful when it comes to the controller design (Masubuchi, 2007; Feng and Su, 2014; Verbandt et al., 2016). The descriptor state-space model can also describe LPV dynamics since the matrices involved need not be constant. The resulting state-space model is given by (3.1).

$$\left[\begin{array}{c|c} M_{22} & M_{23} \\ \hline M_{32} & M_{33} \end{array}\right] + \left[\begin{array}{c|c} M_{21} & M_{24} \\ \hline M_{31} & M_{34} \end{array}\right] \begin{bmatrix} F_u & 0 \\ F_l M_{41} F_u & F_l \end{bmatrix} \left[\begin{array}{c|c} M_{12} & M_{13} \\ \hline M_{42} & M_{43} \end{array}\right] \tag{3.1a}$$

$$F_u = N_u \left(I - M_{11} N_u\right)^{-1}, \; F_l = N_l \left(I - M_{44} N_l\right)^{-1} \tag{3.1b}$$

A corresponding constructor is provided as:

```
> mlft = LFTmod(M,Nu,Nl,E,Ts)
```

where `M` is a 4-by-4 cell-array and `Nu`, `Nl` and `E` are matrices of corresponding sizes. It is readily seen that providing empty feedback matrices `Nu` and `Nl` reduces the `LFTmod` to a standard descriptor state-space model. The latter is made directly available via the call:

```
> mdss = DSSmod(A,B,C,D,E,Ts)
```

where `A`, `B`, `C`, `D` and `E` are standard state-space matrices, `Ts` denotes the sample time in seconds. In case `E` happens to be the unit matrix, the `DSSmod` further reduces to a standard state-space model which is constructed straightaway as:

```
> mss = SSmod(A,B,C,D,Ts)
```

Although the implementation relying heavily on the `LFTmod` might look over-complicated, there are two advantages in doing so. First, more complex operations such as closing a loop become very simple on models in the LFT form (Doyle et al., 1991). Second, the `LFTmod` is able to represent the complex model structures which arise naturally from an LPV controller design (Apkarian and Adams, 1998).

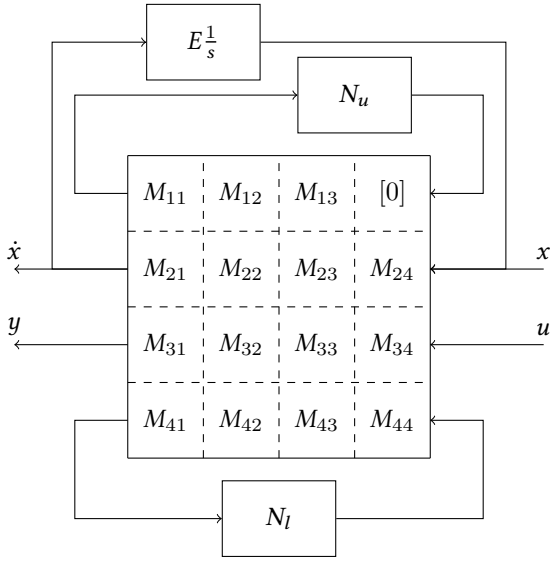


Figure 3.2 · Graphical representation of the `LFTmod` in the Linear Control Toolbox. Two LFTs are included to describe a more general class of models. The dynamics are closed by $E\frac{1}{s}$ so that improper models are allowed.

3.2.4 Validation models

For validation purposes, the toolbox provides more dedicated model classes such as nonlinear models or nonparametric models like a sampled frequency response.

To capture arbitrary nonlinear dynamics, e.g. to perform a time domain validation, the `ODEmod` is introduced. This model incorporates a general differential algebraic equation[(1)] (DAE) which is driven by u and p, i.e. the input to the system and the parameter value. The state x is updated via equation (3.2a). Note that x might be empty, e.g. for a saturation block. Equation (3.2b) returns the output, y, of the model.

$$E(x,u,p)\,\dot{x} = f(x,u,p) \tag{3.2a}$$

$$y = g(x,u,p) \tag{3.2b}$$

An `ODEmod` is constructed by:

```
> mode = ODEmod(f,g,E,p)
```

[(1)] Although one might expect this model to be called `DAEmod`, the model is named `ODEmod` because the term ODE is deemed more widespread than DAE.

where `f`, `g` and `E` are function handles with two or three inputs, depending on whether a scheduling parameter is involved. In case these function handles do depend on a scheduling parameter, the latter is also required as an input argument of the model.

The presented toolbox also offers the possibility to work with nonparametric FRFs. The `FRDmod` class is built on top of Matlab's `frd` class, transferring all original functionality:

```
> mfrd = FRDmod(resp,freq)
```

The `Gridmod` represents a gridded implementation of an LPV model. Take for instance a series of local identification experiments which each yield an `FRDmod` linked to a particular parameter value. These are readily combined in a `Gridmod` via the function call:

```
> mgrd = Gridmod({frd1,frd2,frd3},[pval1,pval2,pval3])
```

The main advantage is that all operations performed on the `Gridmod` are carried out on all underlying elements at once, e.g. computing an element-wise norm or plotting a series of Bode diagrams.

3.2.5 Adding models to systems

Once one or more models have been created, they should be linked to the corresponding system. To do so, the function `add` is used:

```
> P.add(mss,mode,mfrd)
```

Thereafter, `P.content()` is no longer empty. Though standard plotting functionality like `bode` or `step` is supported for individual models, calling these functions on systems is more powerful. This shows all responses on a single graph, facilitating the comparison between different (closed-loop) models.

3.3 Identification module

The system identification module provides the necessary functionality to convert measured time or frequency domain data into parametric models. It interfaces the Matlab implementations (Pintelon and Schoukens, 2012b) of the frequency domain system identification methods in detail elaborated in Pintelon and Schoukens (2012a).

Although the focus currently lies on LTI identification, LPV identification is supported via the State-space Model Interpolation of Local Estimates (SMILE) (De Caigny et al., 2011). The latter is a technique that interpolates LTI models with a user-specified set

of basis functions of the scheduling parameter. B-spline basis functions are used here in order to comply with the B-spline based LPV controller design within the toolbox.

3.3.1 Excitation signals and measurement data

The first step of the system identification procedure is the generation of excitation signals. At present, the toolbox only supports the generation of multisine signals (Pintelon and Schoukens, 2012a). The command

```
> u = Multisine('label', 'experiment1', 'fs', 100,...
       'fwindow', [0.01 2], 'freqres', 0.01, 'type', 'odd')
```

creates a multisine, sampled at 100 Hz, exciting a linearly spaced frequency grid in the interval [0.01, 2] Hz, with a frequency resolution of 0.01 Hz. This choice automatically determines the period of the multisine, in this case 100 s. The option 'type' allows the user to construct an odd multisine instead of the default full realization. The multisine `u` is a so-called `TimeSignal` object that contains all information that is characteristic for this specific signal. Additional name-value pairs allow the user to specify more options, e.g. the phase distribution of the signal, its amplitude spectrum, etc. `TimeSignal` objects feature useful methods; the user can for example plot the signal spectrum by

```
> plotSpectrum(u, 'dft')
```

The actual signal content (one period of the multisine in this case) is obtained by

```
> [s,t] = signal(u)
```

Once the excitation has been applied, the experimental data is imported in the toolbox. This is done by wrapping the measurements into `TDMeasurementData` objects. In case of a system with two inputs and two outputs, a `TDMeasurementData` object is constructed as follows:

```
> meas1 = TDMeasurementData('label','meas1','excitation',...
              [u1 u2],'data', [x1 x2 y1 y2], 'datalabels',...
              {'x1', 'x2', 'y1', 'y2'}, 'periodic', true)
```

where `u1` and `u2` are objects of the aforementioned class `TimeSignal` representing the first and the second input. Similar to `TimeSignal`, the toolbox supports several relevant operations on `TDMeasurementData` objects. For example, to avoid the effect of transients in the FRF estimation, one may decide to keep the last 7 periods of the measurements:

```
> meas1 = clip(meas1, 'lastnper', 7)
```

Complementary to TDMeasurementData, the FDMeasurementData class handles non-parametric frequency response function (FRF) measurements. FRF measurements can either be loaded directly into Matlab or can be calculated based on time domain data from a TDMeasurementData object. The latter is discussed in the following paragraph.

3.3.2 Nonparametric frequency response function estimation

The toolbox supports two methods to estimate the system's FRF based on multiple measurements obtained with different random multisine realizations: the robust detection algorithm for nonlinearities (Pintelon and Schoukens, 2012a, section 4.3.1) implemented as Robust_NL_Anal (Pintelon and Schoukens, 2012b) intended for SISO systems and the robust local polynomial method (Pintelon and Schoukens, 2012a, section 7.2.2) implemented as RobustLocalPolyAnal (Pintelon and Schoukens, 2012b) for MIMO systems. These methods combine measurements from different random multisine excitations to average out both the effects of nonlinear distortions and measurement noise. They yield an FRF estimate, also referred to as the best linear approximation, the sample total variance (contributions of measurement noise and nonlinear distortions) and the sample noise variance.

The nonparametric FRF estimate based on e.g. four different measurements is obtained by calling the routine nonpar_ident with the arguments as shown below:

```
> IOlabels.input = {'x1', 'x2'};
> IOlabels.output = {'y1', 'y2'};
> FRF = nonpar_ident(meas1, meas2, meas3, meas4,
          IOlabels, 'RobustLocalPolyAnal');
```

where 'x1', 'x2', 'y1' and 'y2' refer to the data labels provided to the TDMeasurementData objects. The result, FRF, is a FDMeasurementData object.

3.3.3 Parametric LTI model identification

The nonparametric FRF of the system is further used to estimate a parametric model. For the time being, the toolbox comes with the routine MIMO_ML, a maximum likelihood identification algorithm (Pintelon and Schoukens, 2012a), MIMO_NLS, a nonlinear least-squares fitting algorithm allowing an arbitrary frequency weighting weight, and also interfaces Matlab's routine tfest. These methods estimate common denominator transfer function models. The function param_ident parses the data for each of these routines and is called as shown below:

```
> settings = struct('denh', 2, 'denl', 0, 'numh', 2,...
                'numl', 0, 'W', weight, 'Ts', 0.01)
```

```
> model = param_ident('data', FRF, 'method', 'MIMO_NLS',...
              'settings', settings)
```

where `denh` (`numh`) and `denl` (`numl`) in `settings` represent the highest and lowest degree of the denominator (numerator) of the underlying transfer functions, respectively. `numh` and `numl` are $n_o \times n_i$ matrices, with n_o and n_i the number of system outputs and inputs, respectively.

3.3.4 LTI state-space model interpolation

To obtain an LPV model the aforementioned procedure is repeated for different fixed values of the scheduling parameter yielding a set of parametric LTI models. Subsequently, an LPV state-space model is obtained in two steps. First, the identified LTI models are packed in a `Gridmod`:

```
> mgrd = Gridmod(models,schGrid)
```

where `models` is a cell array of local LTI models and `schGrid` is the set of corresponding values of the scheduling parameter.

Second, an LPV state-space model is obtained from the gridded model through an interpolation carried out by the SMILE technique (De Caigny et al., 2011) and a user defined basis:

```
> basis = BSplineBasis(range,degree,knots)
> mlpv_interp = SSmod(mgrd,basis,schParam)
```

3.4 Controller design module

Since one of the goals of the presented toolbox is to facilitate the design of optimal controllers, another key module is the controller design module. Though optimization problem (2.34) might still be comprehensible to a control practitioner, the actual implementation is not. Two main difficulties arise. First, the control engineer has to choose a solver implementation which fits the problem at hand. This task is not obvious as it depends on several criteria, for instance: Does the problem involve hard constraints? Is the problem formulated with $\mathcal{H}_\infty$ norms only? Is there a constraint on the controller structure? The second hurdle is providing the right arguments to the routine. Not only do these arguments vary from routine to routine, but also the construction of the usually required augmented plant itself might cause problems.

The following sections describe how the Linear Control Toolbox is designed to facilitate formulating the problem as well as obtaining a solution. Currently, the Linear

Control Toolbox supports $\mathcal{H}_\infty$ and $\mathcal{H}_2$ performance criteria in both the objective and constraints, in an LTI as well as an LPV setting. In case the problem is LTI, reduced order controller design is also possible.

3.4.1 Optimal controller formulation

In order to simplify the problem formulation, the Linear Control Toolbox offers the `Channel` and `Norm` classes. A `Channel` is an abstract representation of a transfer function with an input `in` and output `out` and is constructed as:

```
> T = Channel(in,out,name)
```

or via the shorthand notation:

```
> T = out/in
```

The argument `name` is an optional human-readable identifier which is only used for plotting purposes. Multidimensional `in` and `out` are allowed, yielding a MIMO channel.

The second ingredient is a frequency-dependent weight. Although weights can in principle be any transfer function, they often boil down to standard filters. Therefore, the presented toolbox offers the `Weight` class which implements these standards. As pointed out in section 2.2.1, the inverse of the weights can be regarded as upper bounds to the closed-loop transfer functions in the SISO case. Therefore `Weight` employs an inverse definition allowing the user to interpret the specified transfer functions as constraints for the optimization rather than frequency-dependent weights for the closed-loop transfer functions. The simplest possible weight is a constant, constraining the peak of a transfer function:

```
> W = Weight.DC(value)
```

Since control engineers typically prefer a logarithmic scale, `value` is by default interpreted in [dB]. In case `value` is to be interpreted as is, the argument `'linear'` should be supplied. Due to the inverse definition, the actual magnitude of `W` will be `-value` [dB].

```
> W = Weight.LF(fco,ord,lfgain)
```

puts emphasis on lower frequencies and is therefore suited to shape a sensitivity transfer function. At low frequencies, its magnitude approaches `-lfgain` [dB]. `ord` poles are inserted so that the cross-over frequency becomes `fco`. Again, natural parameters are employed to shape the weight. Its counterpart `Weight.HF` also exists, amplifying high frequencies which is for instance applicable to ensure noise suppression.

In combination with a weighting function, the `Channel` yields a `Norm` object. The latter is then used to formulate the optimization problem in a natural way, i.e. visually corresponding to (2.34). A `Norm` is readily constructed via:

```
> N = Norm(W,T,p)
```

where `W` and `T` are the weight and channel respectively. In this case the output of `T` is weighted by `W`. To have weighted inputs which is especially relevant in a MIMO setting, the arguments `W` and `T` are swapped. The third argument `p` specifies which norm is being employed and can take two values: `2` or `inf`. In case `p` is omitted, the default `inf` is employed. In that case, the shorthand notation

```
> N = W*T
```

is also available.

3.4.2 Obtaining a solution

Once the control loop and design objectives have been formulated, the function `solve` is called on the constructed closed-loop system to dispatch the optimization problem to an appropriate solver. A schematic representation of the controller design is shown in Figure 3.3.

```
> S = Channel(r,e,'S'), T = Channel(r,y,'T')
> obj = W1*S
> constr = [W2*S<=1,W3*T<=1]
> [G,k1,info] = G.solve(obj,constr,K,opts)
```

`obj` and `constr` are the objective and constraints which are readily constructed from the appropriate `Norms`. The argument `K` reflects the optimization variable. Because choosing a suitable solver requires expert knowledge, this process is automated by the Linear Control Toolbox. First, the toolbox generates a list of available solvers (table 3.1), where the more tailored solvers come first and the more generic solvers last. Next, this list is pruned depending on criteria such as the presence of constraints or a parameter dependency in the plant, to retain only solvers that are applicable to the specific problem. Because tailored solvers benefit from specific optimizations and therefore tend to perform better than more generic implementations, the first remaining solver is selected by default. However, the options `FullOrderSolver` and `ReducedOrderSolver` allow an experienced user to overrule the automatic selection.

In case the optimization succeeds, the solution, `k1` is automatically added to the optimization variable, `K`. Along with the actual solution, `solve` produces a report on the internal optimization process, `info`. This allows the user to assess the quality of the solution. The easiest way (only for SISO controllers) is to call:

	$\mathcal{H}_\infty$	$\mathcal{H}_2$	RO	HC	LPV	implementation
`hinfsyn`	+					section 2.1.3
`mixedHinfsyn`	+			+		section 2.2.4
`compute_FO_mix`	+	+		+		section 2.2.5
`compute_RO_mix`	+	+	+	+		Hilhorst et al. 2015
`systune`	+	+	+	+		section 2.4
`synLPV`	+	+		+	+	section 2.3

Table 3.1 · Overview of the different solvers in the Linear Control Toolbox and their applicability. In this table, RO and HC denote 'reduced order' and 'hard constraint' respectively.

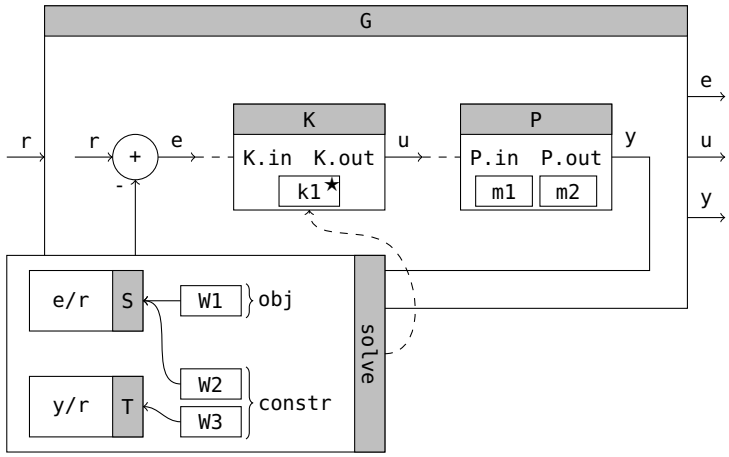


Figure 3.3 · Schematic of the controller design. Channels `S` and `T` are constructed first. Together with the appropriate weights, they yield several norm objects which form the objective and constraints for the optimization procedure. The resulting controller `k1` is automatically added to the optimized system `K`.

```
> bodemag(info)
```

showing the closed-loop frequency responses along with the constraints. For MIMO designs, `sigma` is provided, which plots the weighted closed-loop singular values.

Since controller design is mostly an iterative process, the Linear Control Toolbox supports multiple designs in a straightforward manner. Here too the system-model approach proves itself useful. Subsequent `solve` calls with different objectives and constraints will come up with different control laws, i.e. models, which are all added to the controller system `K`. Therefore, the closed-loop system `G` carries multiple models which are readily compared.

Another more elaborate case would be the design of multiple decentralized controllers. Contrary to Matlab's `hinfstruct`, the Linear Control Toolbox does not support a single-step design procedure for this case. However, the intuitive design procedure of sequential loop closure is easily carried out. First, the inner controller is designed:

```
> [CL,~,info1] = CL.solve(obj_in, constr_in, K_in)
```

Once `K_in` has an implementation, the inner control loop is fully known. Thereafter, the outer loop can be shaped, which is done in a very similar way:

```
> [CL,~,info2] = CL.solve(obj_out, constr_out, K_out)
```

This yields a control law for `K_out` which is the second part of the decentralized controller. This shows that sequentially calling `solve`, each time indicating another system in the decentralized control structure, gradually constructs the overall controller.

3.5 A comparative case study

This section investigates the advantages and drawbacks of the Linear Control Toolbox compared to the state of the art, i.e. Matlab's Robust Control Toolbox (Mathworks, 2018) and LPVtools (Hjartarson et al., 2015). The comparison is based on a case study described by Gahinet et al. (1995) which concerns the control of a missile. The details on the model and the design objectives are followed by a discussion on the problem formulation in each of the toolboxes. Finally an assessment of the underlying methodologies for the controller synthesis is done, based on the quality of the solution as well as the speed at which it is obtained.

3.5.1 Model and design objectives

The controlled system is chosen to be the missile model described in Gahinet et al. (1995) which is described by equation (3.3). The system consists of one control input, δ_m, which represents the fin deflection. The measured outputs are the normalized vertical acceleration, α_{zv}, and the missile's pitch rate, q. Moreover, the missile's dynamics change under influence of two aerodynamical coefficients, $Z_\alpha \in [0.5\ 4]$ and $M_\alpha \in [0\ 105]$, which are the scheduling parameters for the LPV controller design.

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \alpha_{zv} \\ q \end{bmatrix} = \left[\begin{array}{cc|c} -Z_\alpha & 1 & 0 \\ -M_\alpha & 0 & 1 \\ \hline -1 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right] \begin{bmatrix} \alpha \\ q \\ \delta_m \end{bmatrix} \tag{3.3}$$

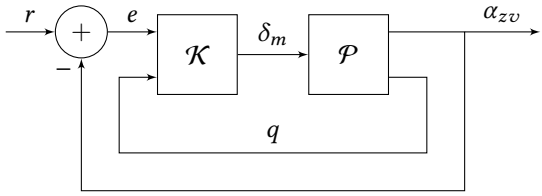


Figure 3.4 · Control configuration for the vertical acceleration control of a missile. The control output δ_m is based on the the tracking error e and the missile's pitch rate q as an additional measurement. (adapted from Gahinet et al. (1995))

The control configuration is depicted in Figure 3.4. The normalized vertical acceleration's tracking error e is fed back together with the missile's pitch rate q. Based on these measurements, the control law computes the fin deflection. The goal is to achieve accurate tracking for the normalized vertical acceleration while taking into account robustness requirements. Therefore the design objective, given by (3.4), consists of two parts. Adequate reference tracking is ensured by taking $\mathcal{S} : r \to e = \alpha_{zv} - r$ into account. The second term in the objective encourages robustness since the latter is guaranteed if $\left\|\mathcal{W}_2\mathcal{U}\right\|_\infty \leq 1$ with $\mathcal{U} : r \to \delta_m$ (Skogestad and Postlethwaite, 2001).

$$\text{minimize} \left\| \begin{matrix} \mathcal{W}_1\mathcal{S} \\ \mathcal{W}_2\mathcal{U} \end{matrix} \right\|_\infty \tag{3.4}$$

with

$$\mathcal{W}_1 = \frac{2.01}{s + 0.201} \tag{3.5a}$$

$$\mathcal{W}_2 = \frac{9.678s^3 + 0.029s^2}{s^3 + 1.203e4s^2 + 1.136e7s + 1.066e10} \tag{3.5b}$$

3.5.2 Constructing an LPV model

The first step in the controller design is to enter the missile's LPV model in the different toolboxes (see code example 3.1). LPVtools and the Linear Control Toolbox employ a very similar approach when it comes to declaring an LPV model. Each of the toolboxes provides a parameter object: LPVtools comes with `tvreal` and `pgrid` whereas the Linear Control Toolbox offers `SchedulingParameter`. This approach makes it very intuitive to declare for instance a parameter varying state-space matrix. Consequently, the resulting code resembles the model description in (3.3) which is helpful to the user. Matlab's Robust Control Toolbox employs a very different strategy inspired by the polytopic nature of the supported models. The vertices of the polytope are

defined as a series of LTI models which combined with a description of the parameter domain result in a polytopic system. This approach requires more care as the order of vertices has to correspond to the list of LTI models. Also, the resulting code shows less resemblance to the model description (3.3) which might be perceived by the user as more challenging.

```
% Linear Control Toolbox
Z = SchedulingParameter('Z',[0.5 4]);
M = SchedulingParameter('M',[0 105]);
P = SSmod([-Z 1;-M 0],[0;1],[-1 0;0 1],[0;0]);

% LPVtools
Z = tvreal('Z',[0.5 4]); % or pgrid
M = tvreal('M',[0 105]); % or pgrid
P = ss([-Z 1;-M 0],[0;1],[-1 0;0 1],[0;0]);

% Robust Control Toolbox
pv = pvec('box',[0.5 4; 0 105]);
s0 = ltisys([0 1;0 0],[0;1],[-1 0;0 1],[0;0]);
sZ = ltisys([-1 0;0 0],[0;0],zeros(2),[0;0],0);
sM = ltisys([0 0;-1 0],[0;0],zeros(2),[0;0],0);
P = psys(pv,[s0 sZ sM]);
```

Code example 3.1 · Comparison of the Linear Control Toolbox, LPVtools and the Robust Control Toolbox when constructing the missile's LPV model.

3.5.3 Declaring the control configuration

To proclaim the control configuration depicted in Figure 3.4, each of the discussed toolboxes provides its own signal-based method. The Linear Control Toolbox uses a syntax as discussed in section 3.2.1 and distinguishes itself from the other toolboxes by using explicit `Signal` objects and `IOSystem` objects that carry input and output signals. LPVtools uses `sysic` as a connection front-end which is a string-based approach. The Robust Control Toolbox offers similar functionality through `sconnect`, also parsing a series of strings to construct the interconnected system. Code example 3.2 shows the construction of the control configuration within the different toolboxes.

```
% Linear Control Toolbox
Psys = IOSystem(P);
Ksys = IOSystem(2,1);
r = Signal; dm = Ksys.out;
a = Psys.out(1); q = Psys.out(2);
e = r - a;
conn = [Psys.in == dm; Ksys.in == [e;q]];
G = IOSystem(Psys,Ksys,conn);
```

```
 9
10  obj = [W1*(e/r);W2*(dm/r)];
11  constr = [];
12  [G,K] = G.solve(obj,constr);
13
14  % LPVtools
15  systemnames = 'P W1 W2';
16  inputvar = '[r{1}; dm{1}]';
17  outputvar = '[W1 ; W2 ; r-P(1) ; P(2)]';
18  input_to_P = '[dm]';
19  input_to_W1 = '[r-P(1)]';
20  input_to_W2 = '[dm]';
21  cleaupsysic = 'yes';
22  Paug = sysic;
23  [K,gam] = lpvsyn(Paug,2,1);
24
25  % Robust Control Toolbox
26  Paug = sconnect('r','W1;W2','K:e=r-P(1);P(2)','P:K',P,'W1:e',W1,'W2:K',W2);
27  [gam,K] = hinfgs(Paug,[2,1]);
```

Code example 3.2 · Comparison of the Linear Control Toolbox, LPVtools and the Robust Control Toolbox when constructing the control configuration and solving the optimization problem.

Although a comparison of these different implementations depends heavily on the user's preferences, it is tried to make it as objective as possible. Table 3.2 summarizes the conclusions of this assessment. The first criterion is the readability of the produced code. In that regard the Linear Control Toolbox and LPVtools would surpass the Robust Control Toolbox since the former rely on additional variables to declare the plant. The ability to supply convenient names for the signals also adds to the readability of the code. This feature is present in both the Robust Control Toolbox and the Linear Control Toolbox which in that regard outperform LPVtools. Moreover, the Linear Control Toolbox and the Robust Control Toolbox also include the controller in the formulation of the control configuration which feels more natural than constructing an open loop to which the controller still needs to be connected, as is the case with LPVtools. On the other hand, LPVtools and the Robust Control Toolbox impose a more structured declaration than the Linear Control Toolbox which helps preventing unconnected models or unresolved connections.

When it comes to expressing the control objectives, it is usually required to extend the control configuration with appropriate weights to obtain the augmented plant. Therefore, these weights are already included in the control configuration `Paug` when using LPVtools or the Robust Control Toolbox. Contrary to its alternatives, the Linear Control Toolbox provides a higher level of abstraction to formulate the optimization problem as described by section 3.4.1. Based on this optimization problem, the

augmented plant is constructed behind the scenes. This allows the user to declare simply the actual control configuration rather than the augmented plant which is merely a mathematical construct.

Generally speaking, the Linear Control Toolbox is very much oriented to the user's comfort, mainly improving the readability of the code. The Robust Control Toolbox sits at the other end of the spectrum, usually resulting in compact code intended for expert users. LPVtools holds the middle ground providing tools to simplify the controller design to some extent. Although these conclusions are drawn based on a specific example, they hold in general since the majority of the control configurations strongly resembles the example.

	RCtoolbox	**LPVtools**	**LCtoolbox**
overall readability	-	+	+
additional variables	-	+	+
signal aliases	+	-	+
controller present	+	-	+
imposed structure	+	+	-
problem formulation	-	-	+

Table 3.2 · Comparison of the user interfaces provided by the Robust Control Toolbox, LPVtools and the Linear Control Toolbox.

3.5.4 Solving the control problem

The missile control problem is now passed to the solvers in the different toolboxes in order to compare the quality of the solution as well as the speed at which it is obtained. However, when passing the problem to LPVtools, an error is thrown related to a rank deficient D_{yw} matrix in the augmented plant. Following the instructions provided by Megretski (2006), a regularization input was added to the control configuration leading to a successful solution. The Linear Control Toolbox's solver was invoked with the `var_deg = 0` resulting in a constant Lyapunov matrix. LPVtools' `lpvsyn` was executed with the `'MinGamma'` flag to avoid a relaxation of the optimal value which would lead to an unfair comparison. Because LPVtools offers two parameter representations, `tvreal` and `pgrid`, both solutions are presented.

The results are gathered in Table 3.3. In terms of optimality, the Linear Control Toolbox and the Robust Control Toolbox show comparable performance whereas LPVtools lags behind with both solution strategies. This is also clearly visible from Figure 3.5, particularly from the sensitivity: the red and green curves show a lower bandwidth than the yellow and blue curves, which implies that LPVtools reaches a lower degree of optimality. This also shows in the step response subject to a spiral

parameter trajectory (3.6) depicted in Figure 3.6. The two controllers designed with LPVtools are slower than the other two responses.

$$\begin{cases} Z(t) = 2.25 + 1.70\exp(-4t)\cos(100t) \\ M(t) = 50 + 49\exp(-4t)\sin(100t) \end{cases} \tag{3.6}$$

The opposite holds when comparing the solver times. Here LPVtools as well as the Robust Control Toolbox outperform the Linear Control Toolbox. Moreover the Linear Control Toolbox's B-spline-based LPV solver is an order of magnitude slower than its competitors. However the time it takes to solve the underlying SDP is only 0.96 s which is more in line with LPVtools and the Robust Control Toolbox. The 16 s overhead is caused by the controller synthesis tools being based on the 'OptiSpline' toolbox by Lambrechts and Gillis (2018). This toolbox eases the manipulation of splines at the expense of an extra cost to export the underlying SDP. A direct implementation of the LMIs would put the B-spline based method amongst its competitors.

	RCtoolbox	**LPVtools-tvreal**	**LPVtools-pgrid**	**LCtoolbox**
$\|\cdot\|_\infty$	0.2054	0.2778	0.6490	0.2021
t [s]	0.73	0.18	1.8612	17.23 (0.96)

Table 3.3 · Comparison of the optimality and solver time of the Robust Control Toolbox, LPVtools and the Linear Control Toolbox.

Although this case study would suggest there are only minor differences between the different toolboxes regarding the quality of the solution, it should be noted that the case study was chosen so that all toolboxes were able to solve the same problem. Both LPVtools and the Linear Control Toolbox are capable of solving a broader variety of problems than the Robust Control Toolbox. They can handle polynomial and B-spline parameterized models respectively and deal with constrained parameter variations explicitly. Moreover, the Linear Control Toolbox is also capable of enforcing hard constraints on the solution and automatically selects an appropriate solver for the problem at hand which are clear advantages from a user's point of view. Each of the aforementioned considerations complicates a thorough comparison of the different toolboxes when it comes to solving the optimal controller design problem, making it virtually impossible to draw general conclusions.

3.6 Conclusion

This chapter has highlighted the main features of the Linear Control Toolbox. Three complementary modules, i.e. the system modeling module, identification module and controller design module, as well as the different validation tools assist the

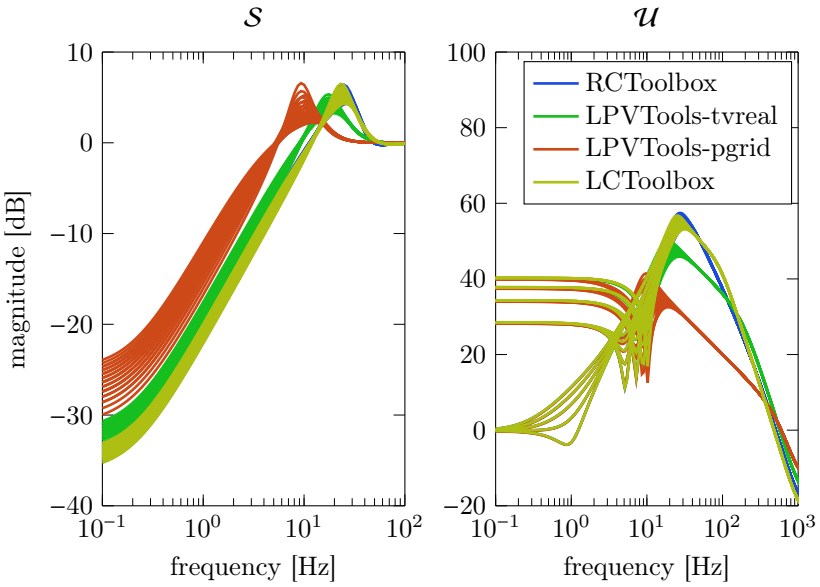


Figure 3.5 · Comparison of the closed-loop transfer functions within the three controller design toolboxes.

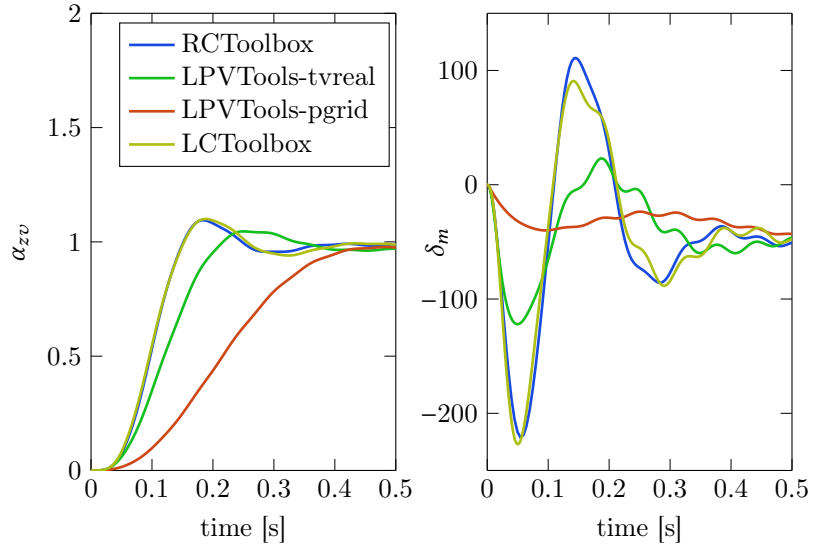


Figure 3.6 · Step response of the missile's autopilot on a spiral parameter trajectory (3.6).

practicing control engineer along the journey towards a high-performance model-based $\mathcal{H}_2$ /$\mathcal{H}_\infty$ controller. The most important data structures have been presented and important design choices were justified. Based on a two-parameter missile control case study, the Linear Control Toolbox was compared to its competitors: LPVtools and Matlab's Robust Control Toolbox. It was shown that the Linear Control Toolbox stands out when it comes to the variety of problems it is able to solve as well as formulating the control problem in a readable manner. The experimental validation of the Linear Control Toolbox is presented in the following chapters. Chapter 4 demonstrates the use of the toolbox on an overhead crane setup, focusing on the necessary code. Chapters 5 and 6 showcase other experimental results obtained with the toolbox.

4

Control of an overhead crane

This chapter demonstrates the use of the Linear Control Toolbox (see Chapter 3) when designing an LPV controller for an overhead crane system. The first part covers the physical modeling of the system resulting in a nonlinear model. This model is well suited for validation but not for controller design. This is why a second model is derived, based on a series of identification experiments. These result in local LTI models that are interpolated to obtain an LPV model. Based on the latter, a parameter varying controller is designed and validated both in simulation and experimentally.

4.1 Physical modeling and identification

In this section, various models to describe the overhead crane system are derived. First, a high-fidelity nonlinear model is derived, based on physical grounds. Because the resulting nonlinear model is not directly suited for a linear controller design, a system identification is carried out. This procedure ultimately yields an LPV model with the cable length as the scheduling parameter.

4.1.1 Physical modeling

The overhead crane that is the subject of this chapter is depicted schematically in Figure 4.1. The base of the system consists of a velocity-controlled trolley with a time constant τ of 10 ms and reference u. The encoder on the trolley's track provides a measurement of x_0. The trolley is equipped with a hoisting mechanism that allows the system to vary the cable length l, which is also being measured. Moreover, l is constrained between 0.3 m and 0.8 m with a maximum hoisting speed of $0.1\,\mathrm{m\,s^{-1}}$. Another sensor on the hoisting mechanism measures the load angle θ. These measurements are combined to obtain a measurement for the output of interest x:

$$x = x_0 + l \sin(\theta) \tag{4.1}$$

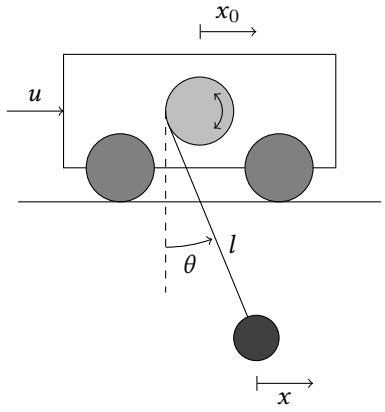


Figure 4.1 · Schematic of the considered overhead crane. The cart is driven by the velocity reference u. x_0, θ and l are measured.

The dynamics of the overhead crane can be derived by means of the Euler-Lagrange equation. The total kinetic energy T and potential energy U in the system are:

$$T = \frac{1}{2} m_0 \dot{x}_0^2 + \frac{1}{2} m \left(\left(l\dot{\theta} \cos\theta + \dot{x}_0 \right)^2 + (l\dot{\theta} \sin\theta)^2 \right) \tag{4.2a}$$

$$U = -mgl \cos\theta \tag{4.2b}$$

By means of the Lagrangian $L = T - V$, the differential equations governing the dynamics are derived via:

$$M_0 = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} \tag{4.3a}$$

$$F_0 = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{x}_0} \right) - \frac{\partial L}{\partial x_0}, \tag{4.3b}$$

where F_0 is the force acting on the cart and M_0 is the torque acting on the pendulum. This torque results from the viscous friction that exists within the hoisting mechanism and is modeled as $M_0 = -c\dot{\theta}$. Substituting (4.2) in (4.3) gives (4.4) under the assumption of a quasi-static cable length.

$$-c\dot{\theta} = ml^2 \ddot{\theta} + ml \cos\theta \ddot{x}_0 + mgl \sin\theta \tag{4.4a}$$

$$F_0 = (m_0 + m)\, \ddot{x}_0 + ml\ddot{\theta} \cos\theta - ml\dot{\theta}^2 \sin\theta \tag{4.4b}$$

Equation (4.4a) relates the pendulum's swing angle to the movement of the cart. Equation (4.4b) gives the required force F_0 to move the cart given the system's states.

This force is implicitly generated by the feedback loop that controls the cart's speed. Therefore, equation (4.4b) should be replaced by the closed-loop cart dynamics, given by:

$$\ddot{x}_0 = \tau^{-1}(r - \dot{x}_0) \tag{4.5}$$

The nonlinear dynamics of the system are thus given by:

$$\begin{cases} \ddot{x}_0 = -\tau^{-1}(\dot{x}_0 - u) \\ \ddot{\theta} = l^{-1}\left(\tau^{-1}(\dot{x}_0 - u)\cos\theta - g\sin\theta - c\dot{\theta}\right) \\ x = x_0 + l\sin\theta \end{cases} \tag{4.6}$$

Code example 4.1 describes the declaration of the crane. First, a SISO system, `crane`, is constructed. Next, the scheduling parameter `l` is introduced followed by a declaration of the nonlinear equations of motion. As a last step, the nonlinear model `nl` is added to the system `crane`.

```
% constructing the crane
Crane = IOSystem(1,1);

% scheduling parameter l
range = [0.3,0.8];
rate = [-0.1,0.1];
l = SchedulingParameter('l',range,rate);

% nonlinear model
f = @(x,u,p) [x(2);-(x(2)-u)/tau;x(4);...
        ((x(2)-u)*cos(x(3))/tau-g*sin(x(3))-c*x(4))/p];
g = @(x,u,p) [x(1) + sin(x(3))*p];
nl = ODEmod(f,g,l,[1,1,4]);
nl.name = 'nonlin';
Crane.add(nl);
```

Code example 4.1 · Declaration of the crane system and construction of the nonlinear model.

4.1.2 Identification of an LPV model

Because the previously derived nonlinear model is not directly suited for a linear controller design, an identification experiment is set up to derive several LTI models for a set of fixed cable lengths. This approach implicitly assumes that the nonlinearities are of a level that is negligable compared to the system's linear approximation. This assumption is not explicitly validated since the practical value of a linear approximate model has already been shown by others, e.g. by Hilhorst, Pipeleers, Michiels, Oliveira,

et al. (2016). Code example 4.2 describes how the toolbox creates an excitation signal and how it is exported to drive the actual setup. The experimental settings are defined first: the sample frequency `fs`, the number of samples of the periodic excitation signal `N`, the frequency resolution `df` and the range of excited frequencies `frange` (all frequencies specified in Hz). Based on these settings, an excitation signal is generated. A full random-phase multisine with amplitude $0.05\,\mathrm{m\,s^{-1}}$ is opted for. This signal is then written to a text file using Matlab's `dlmwrite`, which in turn is fed to the overhead crane.

```
% identification settings
fs = 100; N = 4096; df = fs/N; range = [df,5];
l = 50; %[cm] adapted for each experiment

% exication signals
exc = 0.05*Multisine('label','musin','fs',fs,'fwindow',range,'freqres',df);
dlmwrite(filename,exc.signal_.Value);
```

Code example 4.2 · Code necessary to generate and export a multisine excitation signal for the overhead crane setup.

Once an experiment has been carried out, the data is imported in the Linear Control Toolbox. `TDMeasurementData` is provided as the standard time domain data container and is constructed as described by code example 4.3. First, the control value `u` and the measured output `y` are read from the data file 'exp_50.nc', created during the experiment. Next they are packed together with the originally constructed excitation signal in the `TDMeasurementData` object. Moreover, the signals are labeled 'input' and 'loadpos' and the data is marked as being periodic.

```
u = ncread('exp_50.nc','controller.ControlValues.0');
y = ncread('exp_50.nc','plant.Measurements.0');

% pack data in a TDMeasurementData object
data = TDMeasurementData('label','exp','excitation',exc,'data',[u,y],...
  'datalabels',{'input','loadpos'},'periodic',true);
```

Code example 4.3 · Code necessary to import time domain measurement data in the Linear Control Toolbox.

In order to estimate the LTI models for different cable lengths, a two-step approach is used. This is described by Code example 4.4.

```
% nonparametric estimation
labels.input = 'input';
labels.output = 'loadpos';
npmod_50 = nonpar_ident(data{:},labels,'Robust_NL_anal');

% parametric estimation
```

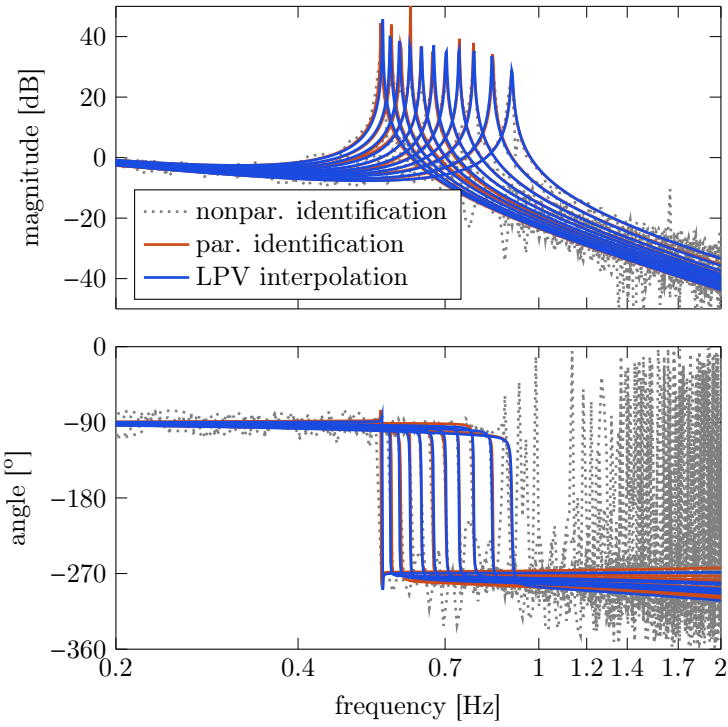


Figure 4.2 · Bode diagram of the different models involved in the identification process. The dotted curves indicate the estimated nonparametric FRF for fixed cable lengths, in red the corresponding LTI models and in blue the evaluation of the interpolated LPV model at the corresponding cable lengths. The B-spline's four degrees of freedom are used to optimally interpolate the 11 local LTI models.

```
set = struct('Ah',3,'Al',1,'Bh',1,'Bl',0,'Ts',0);
lmod_50 = param_ident('data',npmod_50,'method','MIMO_NLS','settings',set);
```

Code example 4.4 · Estimation of a fixed-length LTI model for the overhead crane.

In the first step, the measured data is fed to `nonpar_ident` which estimates a nonparametric FRF. The input and output label indicate which measurements are to be used and `'Robust_NL_anal'` indicates the employed method. Note that `clip` is used to only retain the last three periods of the measured response thereby suppressing the influence of the transient on the resulting frequency response.

The nonparametric estimates for various cable lengths are shown on Figure 4.2. It is

observed that the identification results degrade towards higher frequencies. However, from a control point of view, it is mainly the resonance frequency that needs a decent estimate, which justifies continuing with these models. The second step involves the estimation of a parametric model. To this end, `param_ident` is invoked on `npmod_50` using the method `'MIMO_NLS'` and a structure with some identification settings. The latter contains in this case, the order of the denominator and numerator, `Ah` and `Bh`, the number of integrators and differentiators `Al` and `Bl`, and the sample time `Ts` in seconds.

The previously described sequence of creating an excitation signal, loading the measured data and estimating a model is repeated for cable lengths from 0.3 m to 0.8 m in steps of 0.05 m. The different LTI models are combined in one `Gridmod`, as it serves naturally as a sampled version of an underlying LPV model. Given a B-spline basis and a scheduling parameter, the gridded model is readily transformed into an LPV state-space model through a SMILE interpolation (De Caigny et al., 2011). This LPV model is then added to the system `crane`. This is described in Code example 4.5. Figure 4.2 shows the bode diagrams of the different models involved: the fixed-length FRFs, the fixed-length LTI models and the varying length LPV model.

```
% generation of local LTI models
lgrid = 0.3:0.05:0.8;
lmod = {lmod_30,lmod_35,...,lmod_75,lmod_80};

% LTI models to Gridmod
gmod = Gridmod(lmod,{'l',lgrid});

% Gridmod to LPV state-space
basis = BSplineBasis([0.3,0.8],2,3);
lpv = SSmod(gmod,basis,l);
lpv.name = 'lpv';
crane.add(lpv);
```

Code example 4.5 · Interpolation of local LTI models to obtain an LPV model.

4.2 Controller design and validation

This section describes the controller design for the overhead crane system. First, the control configuration is specified, followed by the objective and the constraints. The corresponding optimization problem is readily solved using the Linear Control Toolbox, yielding an optimal control law. Next, this control law is validated both in frequency and in time domain.

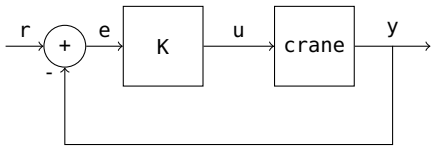


Figure 4.3 · Schematic of the control configuration of the overhead crane.

4.2.1 Controller design

Designing a controller involves two steps: proclaiming the control configuration and stating the design specifications. The presented toolbox facilitates both by offering a suitable syntax.

First, the configuration depicted in Figure 4.3 is built. The corresponding code is found in Code example 4.6. Since the system `crane`, has already been declared, the only system still needed is the controller, `K`. After introducing the reference signal `r`, aliases are provided for the measured output of the crane, `y`, the control signal, `u` and the error signal, `e`, which improves the readability. Finally, the set of connections is passed on to a new system together with the subsystems involved, resulting in the desired configuration. Note that at this point, the closed-loop transfer functions are not accessible because the controller `K` does not yet contain any models.

```
K = IOSystem(1,1);

r = Signal();
y = crane.out; u = K.out;
e = r - y;

conn = [K.in == e; crane.in == u];
G = IOSystem(crane,K,conn);
```

Code example 4.6 · Declaration of the control configuration in the Linear Control Toolbox.

Second, the list of specifications is provided, as shown in Code example 4.7. Two requirements are imposed: the controller's high-frequency roll-off should be −20 dB per decade and the bandwidth should be at least 0.15 Hz. The transfer functions of interest are therefore $r \to e$ and $r \to u$, also referred to as the sensitivity $\mathcal{S}$ and the input sensitivity $\mathcal{U}$. Roll-off is guaranteed by constraining the weighted input sensitivity. The weight $\mathcal{W}_1$ is designed so that high frequencies become dominant in the $\mathcal{H}_\infty$ criterion, effectively suppressing them. By choosing a first order weight, the requirement of high-frequency controller roll-off is met. Similarly, the sensitivity's low frequencies are amplified by $\mathcal{W}_2$ in order to obtain better tracking behavior. The cross-over frequency of $\mathcal{W}_2$ is chosen 0.15 Hz to satisfy the design requirement.

The remaining freedom is used to maximize the damping of the closed loop, i.e. penalizing $\mathcal{W}_3\mathcal{S}$ with constant $\mathcal{W}_3$. The latter is a pragmatic way to ensure a certain degree of robustness of the controller. Explicitly modeling the uncertainty during the system identification is a valuable alternative that might improve the closed-loop performance, but comes at the cost of additional modeling effort and is therefore avoided. By calling `solve` on the closed loop `G`, along with the design specifications, the optimal controller is computed and automatically added to system `K`.

```
S = Channel(r,e,'Sensitivity');
U = Channel(r,u,'Input Sensitivity');

W1 = Weight.HF(300,1,-20);
W2 = Weight.LF(0.15,2,-40);
W3 = Weight.DC(10);

[G,C,sol] = G.solve(W3*S,[W1*U<=1,W2*S<=1]);
```

Code example 4.7 · Stating the design specifications and computing the optimal controller.

4.2.2 Frequency domain validation

The Linear Control Toolbox also offers some validation tools, allowing the user to critically assess the design. Because the controller design was based on an $\mathcal{H}_\infty$ criterion, the first thing to do is to check how the closed-loop transfer functions relate to the imposed constraints. By simply calling `bodemag(sol,S,U)`, the closed-loop transfer functions are shown along with the weights, as depicted in Figure 4.4. It is clearly visible that all constraints are met: the closed-loop transfer functions (solid lines) have a smaller magnitude than their respective weights (dashed lines).

4.2.3 Time domain validation

Time domain simulations may yield additional insight in the behavior of the controller. Although standard commands such as `step` or `impulse` are available within the Linear Control Toolbox, the most powerful command is `sim`. This function carries out a time domain simulation for a system, provided a time-dependent input and parameter. Because a system may contain several models, responses are readily compared. Figure 4.5 shows the simulation result when the cart moves 0.3 m while the load is lowered from 0.4 m to 0.7 m at maximum velocity and executes the opposite maneuver after a few seconds (Code example 4.8). In this case, two simulations are shown: one for the LPV model and one for the nonlinear model. As expected, only small differences

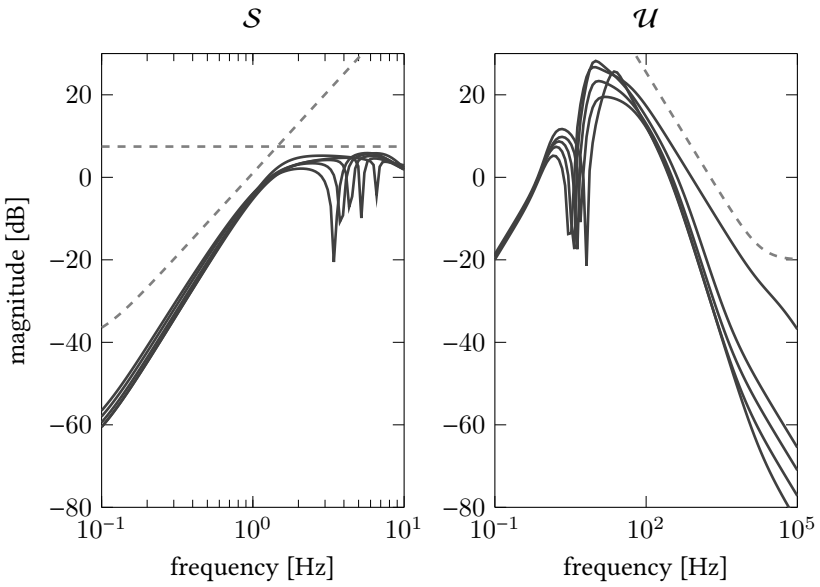


Figure 4.4 · Frequency domain validation of the LPV controller design for the overhead crane example. The closed-loop frequency responses are solid, the constraints are dashed.

occur between the two responses. This observation indicates that the interpolated LPV model describes the nonlinear dynamics sufficiently accurate. Second, the tracking behavior looks adequate and the resonance frequency is sufficiently dampened.

```
% time domain validation
ref = @(t) pfun(t);
l = @(t) lfun(t);
[out,t] = sim(G(y,r),ref,l,30);
```

Code example 4.8 · Validation of the LPV controller design in time domain.

To experimentally validate the designed B-spline LPV controller, it is exported to a C++ implementation with Eigen (Jacob and Guennebaud, 2018) as a linear algebra back-end and a custom implementation of De Boor's algorithm (De Boor, 2001) to evaluate the B-splines. Since the controller was designed in continuous time, a backward Euler scheme is employed to update the control law at a rate of 100 Hz. Figure 4.5 shows the measured load position as well as the control signal along the simulations. It is readily seen that the experimental results match the simulations very well both qualitatively and quantitatively.

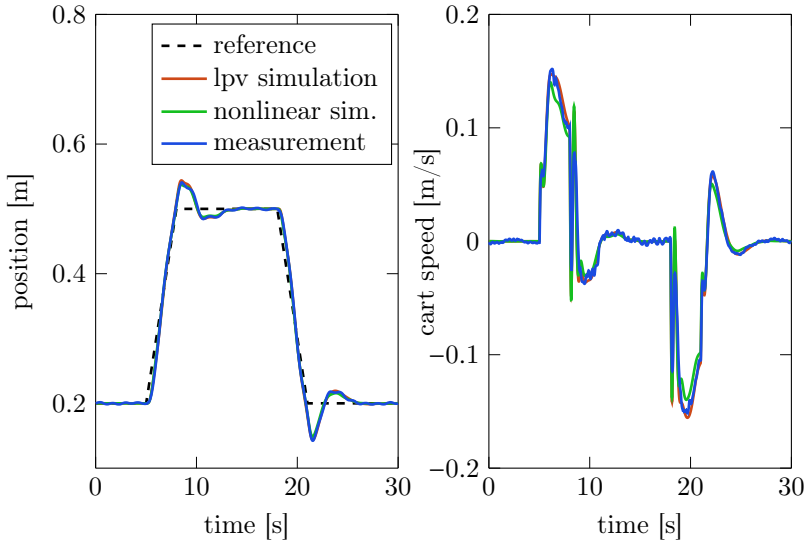


Figure 4.5 · Response of the closed loop when lowering the load from 0.4 m to 0.7 m and moving the load 0.3 m simultaneously for a time span of 3 s. After 10 s, the motion is repeated in the opposite direction so that the load returns to its initial position.

4.3 Conclusion

This chapter demonstrated the use of the Linear Control Toolbox on an overhead crane system. First, a nonlinear model of the system was derived to serve as a high-fidelity validation model. Next, a series of local LTI models was identified and interpolated to obtain an LPV model of the overhead crane with the cable length as a scheduling parameter. The LPV model was used to design an LPV controller, capable of positioning the load while it is being hoisted or lowered. Its behavior was verified by investigating the closed-loop transfer functions and by comparing a simulation of the closed loop with the nonlinear model and the experimental results.

5

Control of a rotordynamic test setup

This chapter describes the design of controllers for the radial active magnetic bearings (AMBs) of a high-speed rotordynamic test setup. The axial bearing remains untouched. In the first section, a physical model of a simplified AMB is derived, leading to insight in the system. Also the field of rotor dynamics is briefly introduced. Next, an identification of the radial bearings is carried out yielding a nonparametric and parametric model, which are compared to the physical model. Section 5.2 covers the controller design and interprets the results. To conclude, the controller is validated both in the frequency domain and the time domain.

5.1 Physical modeling and identification

Because the dynamics of a simple 1-dimensional AMB already provide important insight in the nature of a magnetically levitated rotor, such a system is covered first. Section 5.1.2 points out that only the differential magnetic field exerts a net force, leading to a differential AMB setup. To analyze the effect of rotation and rotor flexibility, a simple rotor dynamical model is introduced. These parts are mainly based on Schweitzer (2009), chapters 4 and 7. Next, reliable nonparametric and parametric models are derived by means of an identification procedure and compared to the physical model.

5.1.1 Dynamics of an AMB

The system discussed in this section is depicted in Figure 5.1. The system consists of a rod with high relative permeability μ. The position of the rod is described by the coordinate z. The rod is enclosed by a pair of electromagnets, referred to by $j \in \{1, 2\}$, with cross sectional area A, N windings with a total resistance R, applied voltage v_j and resulting current i_j. In between the rod and the electromagnets lies a nominal air gap of size s_0.

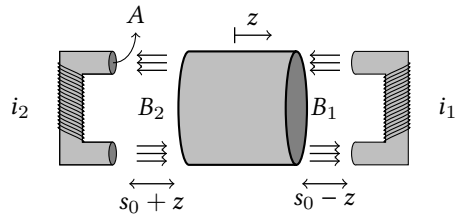


Figure 5.1 · Schematic of a 1-dimensional AMB. A metal rod of high relative permeability is positioned in between two electromagnets with a nominal air gap s_0 on both sides. Each coil carries a current i_j that results in a magnetic flux density B_j. The attraction force results in a movement z of the rod.

Applying Ampère's loop law to the electromagnets under the assumption that $\mu \to \infty$ yields an expression for the magnetic flux density B_j:

$$B_j = \frac{\mu_0 N i_j}{2 s_j}, \tag{5.1}$$

where μ_0 denotes the permeability of the air gap and s_j is the size of the actual air gap: $s_1 = s_0 - z$ and $s_2 = s_0 + z$. Because the eventual driver of the system is the applied voltage, a relation between v_j, i_j and B_j is required. To this end, Faraday's law is consulted:

$$\begin{aligned} NA\dot{B}_j &= v_j - i_j R \\ &= v_j - \frac{2 s_j R}{\mu_0 N} B_j. \end{aligned} \tag{5.2}$$

This differential equation describes the electrical side of the considered system: it relates the applied voltage to the magnetic flux density. The coupling with the mechanical side lies within s_j since it depends on the translation of the rod z. To resolve this coupling, it is required to compute the force F acting on the rod, since:

$$F = m\ddot{z}, \tag{5.3}$$

with m the mass of the rod. A possible way of obtaining an expression for F in terms of the electrical variables is via the potential energy U_j stored in the air gap. Given the flux density, U_j is computed via

$$U_j = 2 \frac{B_j^2}{2\mu_0} A s_j. \tag{5.4}$$

The force derivable from a potential energy storage is computed via the spacial derivative, so

$$F_j = -\frac{dU_j}{dz} = -\frac{A B_j^2}{\mu_0} \frac{ds_j}{dz}. \tag{5.5}$$

This means that the total force acting on the rod equals

$$F = F_1 + F_2 = \frac{A}{\mu_0}\left(B_1^2 - B_2^2\right). \tag{5.6}$$

The complete system is thus described by a triplet of nonlinear differential equations, i.e. equation (5.2) for $j = \{1, 2\}$ and the combination of (5.3) and (5.6).

5.1.2 Dynamics of a differential AMB

An elegant formulation is obtained when applying a coordinate transformation based on the sum and difference of the electrical variables. The proposed transformation is as follows:

$$b^- = \frac{s_0}{\mu_0 N}\left(B_1 - B_2\right),\ b^+ = \frac{s_0}{\mu_0 N}\left(B_1 + B_2\right) \tag{5.7a}$$

$$v^- = \left(v_1 - v_2\right)/2,\ v^+ = \left(v_1 + v_2\right)/2 \tag{5.7b}$$

$$i^- = \left(i_1 - i_2\right)/2,\ i^+ = \left(i_1 + i_2\right)/2. \tag{5.7c}$$

The governing dynamics then become:

$$\ddot{z} = \frac{2L}{ms_0} b^- b^+ \tag{5.8a}$$

$$\dot{b}^- = -\frac{R}{L} b^- + \frac{R}{L}\frac{1}{s_0} z b^+ + \frac{1}{L} v^- \tag{5.8b}$$

$$\dot{b}^+ = -\frac{R}{L} b^+ + \frac{R}{L}\frac{1}{s_0} z b^- + \frac{1}{L} v^+, \tag{5.8c}$$

with $L = 0.5\mu_0 N^2 A s_0^{-1}$ the nominal inductance of each electromagnet.

To gain more insight in the set of nonlinear differential equations (5.8), the model is linearized around its equilibrium state. Putting all derivatives in (5.8) to zero together with $z = 0$ results in the following equilibrium conditions:

$$0 = b_0^- b_0^+ \tag{5.9a}$$

$$0 = -R b_0^- + v_0^- \tag{5.9b}$$

$$0 = -R b_0^+ + v_0^+. \tag{5.9c}$$

All i_j can be assumed positive because both negative and positive currents exert a pulling force. In that case, (5.9a) amounts to $b_0^- = 0$. The additional equilibrium

conditions are easily derived and the linearized model becomes:

$$\ddot{\delta z} = \frac{K_i}{m}\delta b^- \tag{5.10a}$$

$$\dot{\delta b}^- = -\frac{R}{L}\delta b^- - \frac{R}{L}\frac{K_s}{K_i}\delta z + \frac{1}{L}\delta v^- \tag{5.10b}$$

$$\dot{\delta b}^+ = -\frac{R}{L}\delta b^+ + \frac{1}{L}\delta v^+, \tag{5.10c}$$

where the δ-variables represent small perturbations with respect to the equilibrium state, $K_i = 2Lb_0^+ s_0^{-1}$ denotes the actuator gain and $K_s = -K_i b_0^+ s_0^{-1}$ reflects the nominal actuator stiffness.

An examination of the linearized dynamics shows that it consists of two decoupled parts. Equations (5.10a) and (5.10b) describe the rod position δz as a function of the differential quantities. This was to be expected because it is the differential mode of F_1 and F_2 that exerts a net force on the rod. Equation (5.10c) governs the common mode dynamics. Although these do not influence δz, the presence of a nonzero common mode magnetic field is vital since $b_0^+ = 0$ makes δz uncontrollable. Having a fixed current bias on the two electromagnets is the traditional way of generating b_0^+. The main advantage of this approach is that the amplifiers have to supply only positive currents, at the expense of additional resistive losses due to the bias current. Another elegant solution is to opt for electromagnets with a permanent magnetic core, invoking a fixed common mode magnetic field $b_0^+ \neq 0$. In that case, the electromagnets can be wired together so that $i_1 = -i_2$ and only one current amplifier is needed. This approach avoids resistive losses resulting in a higher effiency of the system and requires only one instead of two current amplifiers. However, this one current amplifier now has to be capable of supplying a bidirectional current and the construction of the bearing is complicated by the forces exerted by the permanent magnets.

Either approach directly eliminates equation (5.10c) from the modeled dynamics, resulting in a model $\mathcal{M} : \delta v^- \to \delta z$, given by

$$\mathcal{M} = \left[\begin{array}{ccc|c} -\frac{R}{L} & -\frac{R}{L}\frac{K_s}{K_i} & 0 & \frac{1}{L} \\ 0 & 0 & 1 & 0 \\ \frac{K_i}{m} & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \end{array}\right]. \tag{5.11}$$

The characteristic equation to determine the poles of $\mathcal{M}$ is given by equation (5.12).

$$\lambda^3 + \frac{R}{L}\lambda^2 + \frac{R}{L}\frac{K_s}{m} = 0 \tag{5.12}$$

By means of the Routh-Hurwitz criterion (see Gopal (2002), section 5.4), it can be shown that the open loop system is unstable. The criteria for the roots of (5.12) to be

in the open left half-plane are:

$$\frac{R}{L} > 0 \tag{5.13a}$$

$$\frac{R}{L}\frac{K_s}{m} > 0 \tag{5.13b}$$

$$\frac{R}{L} \cdot 0 > \frac{R}{L}\frac{K_s}{m}. \tag{5.13c}$$

Because $K_s < 0$, (5.13b) does not hold. Moreover, the absence of a damping force proportional to $\dot{\delta z}$ leads to poles on the imaginary axis, explaining the inevitable violation of (5.13c). It is clear that an AMB system displays an inherently unstable nature that makes the need for control obviously more stringent.

5.1.3 Dynamics of a suspended rotor

Because this chapter is devoted to a magnetically levitated rotordynamic test setup, it is also important to have a notion of rotor dynamics. This text only covers the basics of this topic to add insight to the results. First, consider a rigid rotor, as depicted in Figure 5.2. The states considered are the inclination and the displacement of the rotor, both in the xz-plane and the yz-plane:

$$\chi = \left[\alpha_y, s_x, \alpha_x, s_y\right]. \tag{5.14}$$

The linearized dynamics of the rigid rotor, assuming small inclinations, then become

$$\begin{bmatrix} I & 0 & 0 & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & m \end{bmatrix} \ddot{\chi} = -I_z\Omega \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \dot{\chi} + \phi, \tag{5.15}$$

where m represents the rotor mass and I the rotor's mass moment of inertia around an axis perpendicular to z, I_z is the rotor's mass moment of inertia around the z-axis and Ω is the rotational speed of the rotor. The shaft's acceleration in the fixed reference frame is thus influenced by the gyroscopic forces and the external forces ϕ exerted by the AMBs. Given the bearing force vector $F = [F_1, F_2, F_3, F_4]$ and bearing displacements $s = [s_1, s_2, s_3, s_4]$, the generalized force vector ϕ is computed as:

$$\phi = BF = BK_s s = K_s BB^T \chi \tag{5.16}$$

with

$$B = \begin{bmatrix} -a & 0 & b & 0 \\ 1 & 0 & 1 & 0 \\ 0 & -a & 0 & b \\ 0 & 1 & 0 & 1 \end{bmatrix}, \tag{5.17}$$

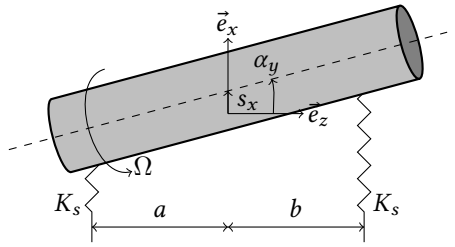


Figure 5.2 · Sketch of a suspended rotor as viewed within the xz-plane. A similar sketch can be made for the yz-plane.

a and b the distances between the rotor's center of gravity and the actuator/sensor planes, and K_s the bearing stiffness, which was found to be negative (see section 5.1.2). This derivation reveals two important characteristics. First, because of the negative bearing stiffness, the open-loop dynamics (5.15) are unstable. Therefore, a stabilizing controller will require phase lead around the cross-over frequency. Second, there exists a coupling between the dynamics in the x and y direction as a result of the gyroscopic forces that in addition varies with the rotational speed. Whereas an LPV framework allows taking this coupling into account rigorously, this is not possible when relying on LTI results. In the latter case, the coupling is usually neglected because gyroscopic forces cannot destabilize the system and typically do not severely deteriorate performance.

In practice, the rotor displays some degree of flexibility. This results in a series of resonance and antiresonance frequencies (Rzadkowski and Sokolowski, 2004). Depending on the stiffness, these flexible modes might end up in the neighborhood of the cross-over frequency, further complicating the controller design.

5.1.4 Identification of the AMB

The AMBs that are the subject of this chapter are part of a high-speed rotordynamic test setup. The system contains a total of five differential AMBs $\mathcal{B}_i$, as depicted in Figure 5.3. Each individual AMB system consists of a pair of differentially connected electromagnets with an integrated noncollocated sensor to measure the rotor position. The radial position of the rotor is controlled by AMBs $\mathcal{B}_1$ to $\mathcal{B}_4$ in two parallel planes: $z = -a$ and $z = b$. Each side consists of two bearings: one to control position along x and one for y. To prevent an asymmetrical load on the bearings, the x and y bearings are positioned at a 45° angle with respect to the gravitational force. As mentioned earlier, the axial bearing $\mathcal{B}_5$ is not studied. Moreover, the results for $\mathcal{B}_3$ and $\mathcal{B}_4$ are from here on omitted because of the high similarity to $\mathcal{B}_1$ and $\mathcal{B}_2$.

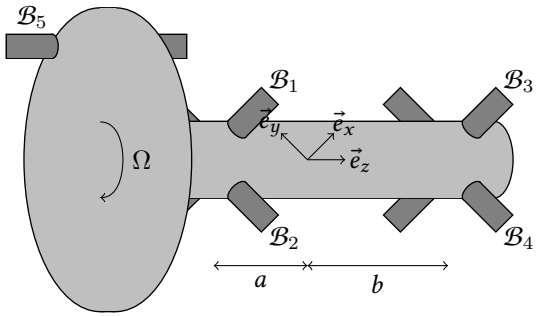


Figure 5.3 · Sketch of the rotordynamic test setup with AMBs.

Because the AMB driver is capable of generating a stepped sine signal, a stepped sine identification (Swevers et al., 1992) is the method of choice. The advantage of a stepped sine identification is the high signal-to-noise ratio, which immediately results in a high-quality nonparametric model at the expense of relatively time-consuming experiments. Also a stepped sine identification does not yield an estimate of the model uncertainty and the level of nonlinearities, making it impossible to assess the validity of a linear model without further experimentation. It is thus assumed that a linear approximation is sufficient in view of the controller design. Because the system is unstable, the identification is carried out in closed loop (see Pintelon and Schoukens (2012a) section 7.2.7). During the identification, the rotor is levitated and an excitation signal is applied to one of the electromagnets while the corresponding position and actuation signals are measured. This ultimately leads to two single-input-single-output (SISO) nonparametric models $\mathcal{B}_1$ and $\mathcal{B}_2$ that are shown in Figure 5.4 and Figure 5.5 respectively. Because of the rotational symmetry of the machine, the two models display a high degree of similarity. Two structural differences can be observed. First, $\mathcal{B}_1$ displays an antiresonance-resonance pair whereas $\mathcal{B}_2$ does not. This effect most probably occurs due to the mounting of the machine on the supporting frame. Second, there is a small difference between the gains of the systems, i.e. the actuator gain K_i. This might be explained by a difference in magnetization of the permanent magnets. Furthermore, both models show a phase decay corresponding to a time delay

$$T_d = \frac{135\pi/180}{2\pi 2000}\text{s} = 0.2\,\text{ms}. \tag{5.18}$$

Also a series of antiresonance-resonance pairs is present at higher frequencies as a result of the flexibility of the rotor.

Traditionally, the next step involves fitting a separate parametric model on each nonparametric model. However, the AMB driver requires a single control law for the two bearings in the same control plane, with the controller's DC gain as the only

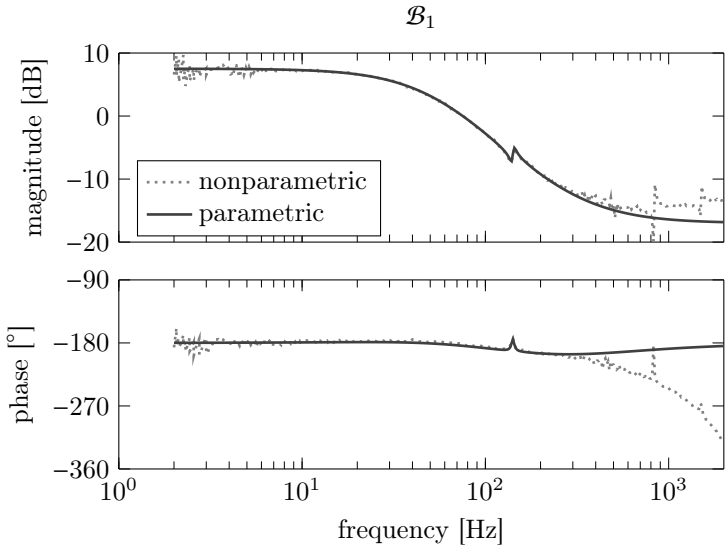


Figure 5.4 · Parametric and nonparametric models for bearing $\mathcal{B}_1$.

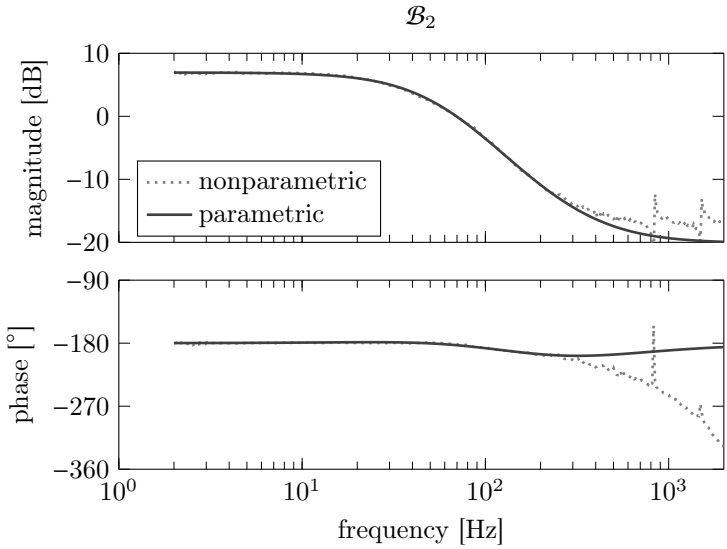


Figure 5.5 · Parametric and nonparametric models for bearing $\mathcal{B}_2$.

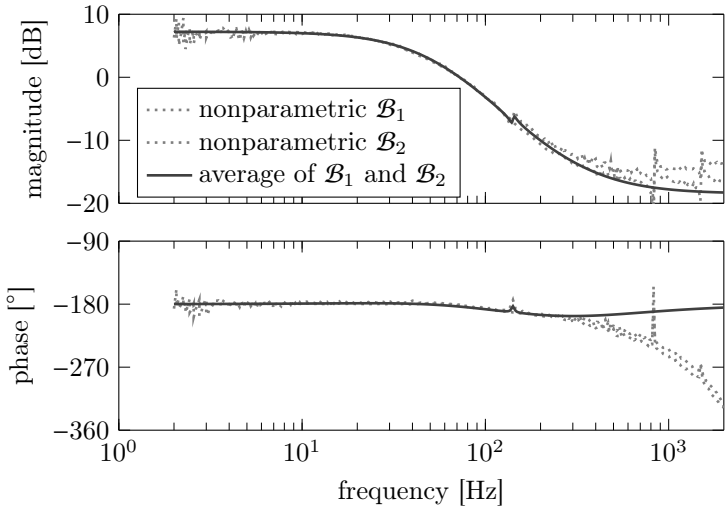


Figure 5.6 · Average parametric model of $\mathcal{B}_1$ and $\mathcal{B}_2$, together with the individual nonparametric models.

difference. This essentially comes down to a robust controller design but because both models are nearly identical (note that the difference in actuator gain can be compensated for), robust stability is not enforced explicitly but checked after the design phase. Because the controller design requires a single model, the average of the two parametric models is used. The average model has in general $2n$ poles where n is the number of poles of each individual model. This in turn leads to a controller of order $2n$, which has twice as many poles as a controller design based on an individual model. To avoid this artificial blow-up of the controller order, the two parametric models are forced to have the same poles so that the average model retains n poles. The two identified parametric models are shown in Figure 5.4 and Figure 5.5. To avoid a high complexity of the resulting model, the fitted frequency range was limited to 250 Hz , which explains the large phase deviation at higher frequencies. This however, is a sensible frequency range for the controller design because the model exhibits an unstable pole around 50 Hz, which is related to the eventual bandwidth of the closed loop. Figure 5.6 shows the resulting average model that is the model used for the controller design.

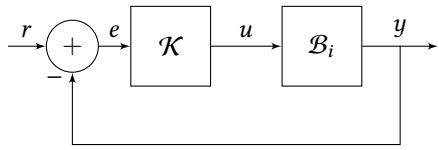


Figure 5.7 · Control loop of a single AMB.

5.2 Controller design and validation

This section describes the controller design for the radial AMBs of a high-speed rotordynamic test setup. Based on the average parametric model, a set of constraints and an objective function, an optimal feedback controller is computed and reflected on. Note that this section only mentions bearings $\mathcal{B}_1$ and $\mathcal{B}_2$, i.e. sensor/actuator plane $z = -a$. The high similarity of the other sensor/actuator plane leads to nearly identical results which are therefore omitted. Afterwards, the controller is validated both in frequency and time domain.

5.2.1 Design

One SISO control loop of the decoupled AMB system is depicted in Figure 5.7. Two transfer functions are distinguished: the sensitivity $\mathcal{S} : r \to e$ and the input sensitivity $\mathcal{U} : r \to u$. The list of specifications for the AMB controller consists of regulations as well as requirements arising from standard practice. First of all, applying ISO 7919-1 to AMB systems results in a constraint on the peak sensitivity

$$\max_{\omega} |\mathcal{S}(j\omega)| < 9.5 \text{ dB}. \tag{5.19}$$

This constraint also adds to the robustness of the controller and is therefore a practical alternative to an explicit uncertainty model, which is in this case not available.

Second, a trade-off is made between adequate tracking on the one hand and actuator effort at high frequencies on the other. The latter can be understood as a measure against actuator wear. The concept of actuator wear is rather vague: there is no way to obtain a hard constraint on the frequency content of the actuation signal to prevent this phenomenon. Therefore, it is proposed to fix the bandwidth at the desired value and exploit the remaining freedom to minimize the actuation signal's high-frequency content. These specifications lead to the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \left\| \mathcal{W}_3 \mathcal{U} \right\|_\infty && \text{(5.20a)} \\ \text{subject to} \quad & \left\| \mathcal{W}_1 \mathcal{S} \right\|_\infty \leqslant 1 && \text{(5.20b)} \\ & \left\| \mathcal{W}_2 \mathcal{S} \right\|_\infty \leqslant 1. && \text{(5.20c)} \end{aligned}$$

Normally, (5.19) immediately leads to $\mathcal{W}_1 = 10^{-9.5/20}$, but because the actual plant differs from the nominal model this constraint will not be satisfied in practice. As a pragmatic solution, (5.19) is made more strict, e.g. $\mathcal{W}_1 = 10^{-9/20}$. Weight

$$\mathcal{W}_2 = \frac{66.2}{s + 20.9} \tag{5.21}$$

is a low-pass butterworth filter of order 1, which enforces a low sensitivity at low frequencies with a bandwidth of 10 Hz and a penalty at 0 Hz of 10 dB. Although arbitrary, the latter is chosen rather low to avoid a big sensitivity peak due to the waterbed effect. These specific values are based on previous designs. Similarly, high-pass butterworth filter $\mathcal{W}_3$ (5.22) penalizes the high-frequency content of $\mathcal{U}$.

$$\mathcal{W}_3 = \frac{1000s^3}{(s + 94250)\,(s^2 + 94250s + 94250^2)} \tag{5.22}$$

Its coefficients are so that the bandwidth and high-frequency penalty are as desired, i.e. 1500 Hz and 60 dB respectively. Note that $\mathcal{W}_3$ is part of the optimization so that the specific values have a limited influence on the result. The order of $\mathcal{W}_3$ is based on previous designs. Figure 5.8 provides a graphical representation of constraints (5.19),(5.21) and (5.22). The solution to optimization problem (5.20) can be obtained by means of the Linear Control Toolbox (see Chapter 3) and is laid out in Figure 5.9.

5.2.2 Frequency domain assessment

Although the optimal controller is merely the result of an optimization problem, a reflection is always appropriate. Two characteristics catch the eye. First, the controller provides phase lead in the region between 100 Hz and 200 Hz, which is necessary to stabilize the unstable system. Second, the controller shows a third order roll-off at high frequencies, which is a direct consequence of $\mathcal{W}_3$.

Figure 5.10 shows the sensitivity and the input sensitivity for bearing $\mathcal{B}_1$ and $\mathcal{B}_2$. Each time, three different models are shown. The prediction based on the average parametric model gives a first indication of the closed-loop behavior and satisfies (5.20). Because the individual bearings behave slightly differently from the nominal response, the loop is also closed with the actual bearing models, i.e. the parametric and nonparametric model. The parametric model is employed to check the stability of the actual closed loop, which is satisfied for both $\mathcal{B}_1$ and $\mathcal{B}_2$. The nonparametric model gives an idea of the actual frequency response, eliminating the fitting error which is present in the parametric model. The individual bearing models suggest a sensitivity peak which is slightly higher than 9.5 dB but still acceptable.

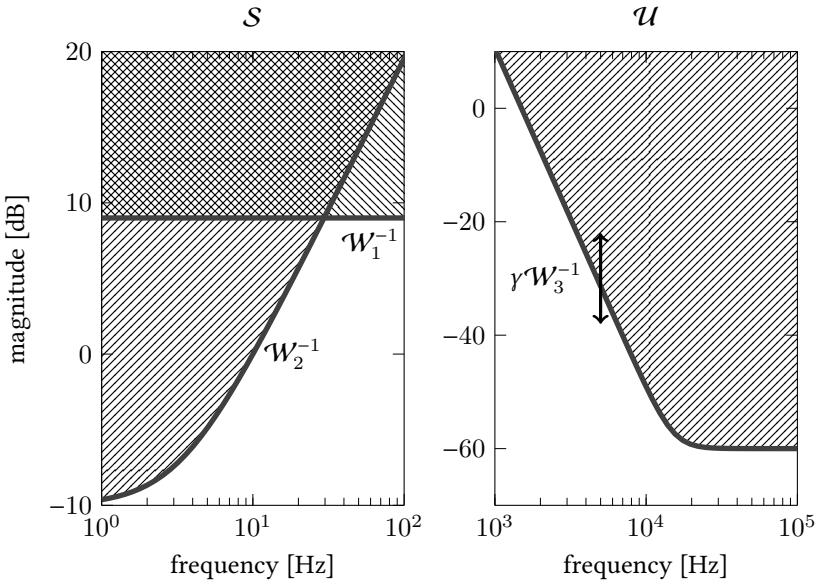


Figure 5.8 · Graphical representation of the regions in which $\mathcal{S}$ and $\mathcal{U}$ are allowed. Note that the optimization slack variable γ causes $\mathcal{W}_3$ to move up or down.

5.2.3 Time domain assessment

The most important criterion to assess the performance of the machine is its behavior at startup. This is especially important because the dominant frequency component of the disturbance signal corresponds to the rotational speed of the machine. This can for instance be caused by an unbalance or roundness error of the rotor. As the rotational speed increases, the uncompensated flexible modes of the shaft might cause a resonance and unacceptable rotor displacements, also know as orbits. Figure 5.11 shows the radius of the orbit r relative to the maximum clearance r_{max} on the $\mathcal{B}_1$-$\mathcal{B}_2$ side when the displayed speed profile is applied to the machine. The orbit radius stays just within the allowed clearance and thus passes the test. Additional testing also reveals that the controller is capable of stabilizing the system up to the maximum rotational speed of 30 krpm without any modification or scheduling.

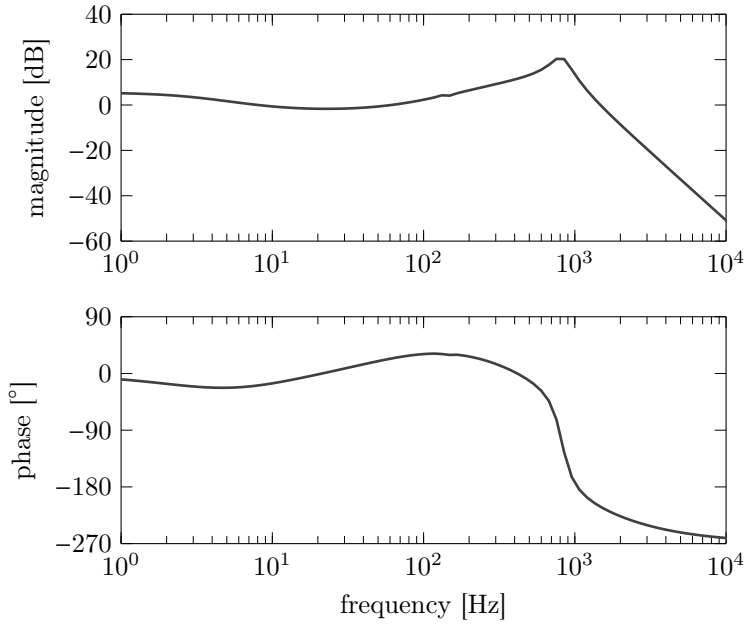


Figure 5.9 · Optimal feedback controller for problem (5.20)

.

5.3 Conclusion

This chapter described the design of controllers for the radial AMBs of a high-speed rotordynamic test setup. Based on a simplified analytical model, the unstable nature of the machine was demonstrated. From the field of rotor dynamics, it was inferred that the rotor's flexibility causes high-frequency antiresonance-resonance pairs and that the system's behavior depends on the rotational speed. The latter however was neglected because parameter varying control is not supported by the controller's firmware. These insights were confirmed by the identification of the actual machine, which lead to a nonparametric and a parametric model of the four individual radial AMBs. The specifications for the controller design were listed leading to an optimal feedback controller design problem. Because of a restriction in the drive's software, a single controller was designed for each pair of bearings, based on the average parametric model. The resulting controller was analyzed and the closed-loop behavior was validated both in the frequency domain and in the time domain.

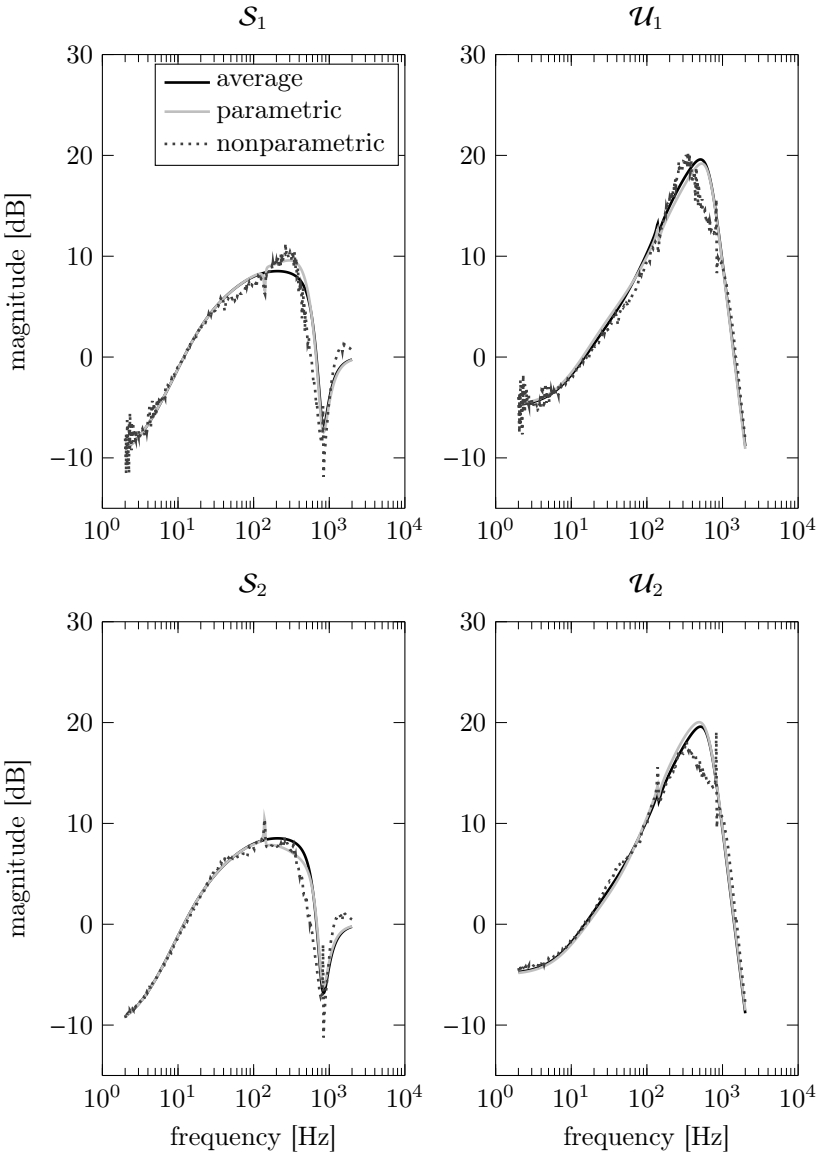


Figure 5.10 · Closed-loop transfer functions for bearings $\mathcal{B}_1$ (top) and $\mathcal{B}_2$ (bottom).

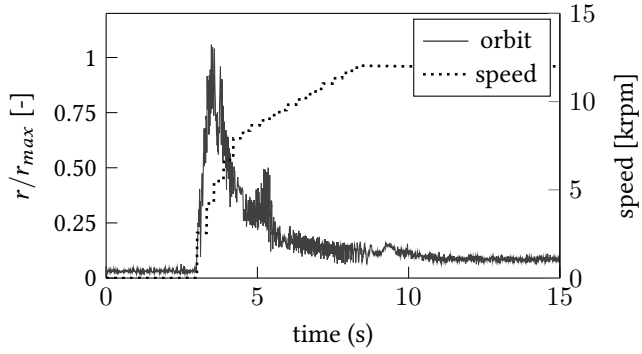


Figure 5.11 · Relative orbit radius of the rotor in the $\mathcal{B}_1$-$\mathcal{B}_2$ plane as a result of speeding the rotor up. The relative orbit stays just within the maximum clearance.

6

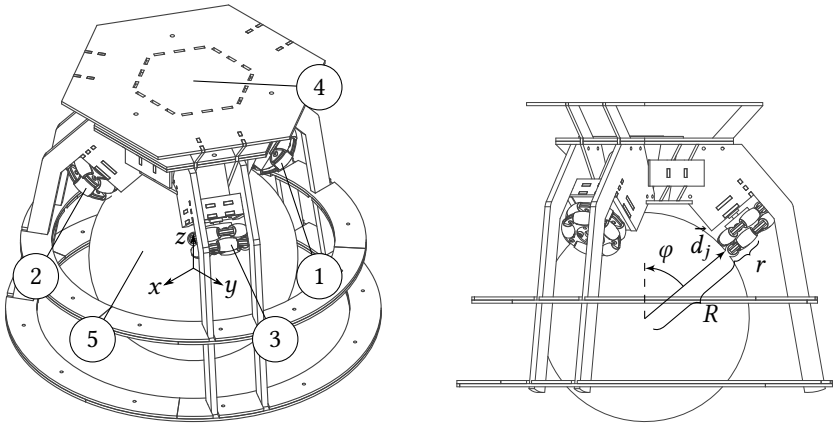
Velocity control of a ball-balancing robot

This chapter presents the design of a velocity controller for a ball-balancing robot. The first step towards a suitable controller consists of the modeling of the system. This is done via both first principles modeling and an experimental identification procedure. The resulting models are then used to synthesize a controller capable of tracking a velocity setpoint. Two different design strategies are investigated: a cascade and a single-step MISO controller design. Finally, the performance of the resulting controllers is experimentally validated and compared.

6.1 Physical modeling and identification

The system that is the subject of this chapter belongs to a relatively new class of mobile robots: the ball-balancing robots (Lauwers et al., 2005). A schematic of the specific design is depicted schematically in Figure 6.1a. The robot's body carries three omniwheels $j \in \{1, 2, 3\}$, actuated by voltage-controlled DC motors and set up in a rotational symmetrical configuration. The body is positioned on top of a ball that is in contact with the three omniwheels and the ground. This contact allows the robot to transfer motion to the ball and effectively change its course. In order to control the system, the body carries an inertial measurement unit (IMU) providing measurements of the body's attitude, i.e. the roll and pitch angles. Velocity measurements are deduced from the rotary encoders that are attached to each of the motors. For more details on the platform, the reader is referred to section 9.4.1.

The robot's reference frame, as indicated by Figure 6.1a, is positioned at the center of the ball and has its z-axis pointing in the opposite direction of the gravitational force. The x-axis lies in between wheels 2 and 3 and the y-axis completes the right-handed reference frame. The positioning of the wheels with respect to the reference frame is fixed via the angle ψ_1 between the projection of the axis of wheel 1 in the xy-plane and the x-axis, so $\psi_1 = \pi$ for this configuration. The angles ψ_2 and ψ_3 corresponding



(a) (b)

Figure 6.1 · Schematic of the ball-balancing robot. Three omniwheel-motor pairs (1)-(3), mounted on the body (4) drive a ball (5).

to wheels 2 and 3 are then given by

$$\psi_j = \psi_1 + (j-1)\,\frac{2\pi}{3}. \tag{6.1}$$

Furthermore, the angle between the wheel's contact point $\vec{d}_j$ and the z-axis is denoted by φ and the radii of the ball and the wheels are labeled R and r, as indicated by Figure 6.1b.

The remainder of this section consists of five parts. First, a static decoupling is proposed to convert the 3D model to three separate 2D models that describe the dynamics in the xz-, yz- and xy-plane. This approach simplifies reasoning about the system and the controller design. Next, a nonlinear model for the 2D equivalent system in the xz- and yz-plane, corresponding to a pitching and rolling motion respectively, is derived. The dynamics in the xy-plane, i.e. a yaw motion, are no longer considered. To gain more insight in the dynamics, the nonlinear 2D model is then linearized around its equilibrium state. In the last two sections, theory meets practice as the system undergoes an identification procedure to obtain models for the system's attitude and velocity.

6.1.1 Decoupling the 3D model

As depicted in Figure 6.1a, the considered robot carries three omniwheels actuating the ball. These wheels are placed in a circular symmetrical fashion, each of them

influencing the dynamics in the xz-, yz- and xy-plane. In order to obtain a decoupled system, it is thus required to distinguish the contribution of each wheel in each of the individual planes. To this end, consider the speed of the ball $\vec{v}_j$ at the contact between wheel j and the ball as a result of the ball's rotational velocity $\vec{\omega}_b = [\omega_x,\ \omega_y,\ \omega_z]^T$ with s_x and c_x as shorthand notations for $\sin(x)$ and $\cos(x)$:

$$\vec{v}_j = \vec{d}_j \times \vec{\omega}_b = R \begin{bmatrix} -\mathrm{c}_\varphi \omega_y + \mathrm{s}_\varphi \mathrm{s}_{\psi_j} \omega_z \\ \mathrm{c}_\varphi \omega_x - \mathrm{s}_\varphi \mathrm{c}_{\psi_j} \omega_z \\ -\mathrm{s}_\varphi \mathrm{s}_{\psi_j} \omega_x + \mathrm{s}_\varphi \mathrm{c}_{\psi_j} \omega_y \end{bmatrix}. \tag{6.2}$$

The tangential component of the contact velocity is equal to the wheel radius r multiplied by the motor speed ω_j so that

$$\begin{aligned} \omega_j &= \frac{1}{r} \vec{v}_j \cdot \begin{bmatrix} -\mathrm{s}_{\psi_j} & \mathrm{c}_{\psi_j} & 0 \end{bmatrix}^T \\ &= \frac{R}{r} \left(\mathrm{c}_\varphi \mathrm{c}_{\psi_j} \omega_x + \mathrm{c}_\varphi \mathrm{s}_{\psi_j} \omega_y - \mathrm{s}_\varphi \omega_z \right) \end{aligned} \tag{6.3}$$

holds for all j. Equation (6.3) yields a triplet of equations that relates the vector of motor speeds $\vec{\omega}_m = [\omega_1,\ \omega_2,\ \omega_3]^T$ to the ball's rotational velocity

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \underbrace{\frac{R}{2r} \begin{bmatrix} 2\mathrm{c}_\varphi \mathrm{c}_{\psi_1} & 2\mathrm{c}_\varphi \mathrm{s}_{\psi_1} & -2\mathrm{s}_\varphi \\ \mathrm{c}_\varphi \left(-\mathrm{c}_{\psi_1} - \sqrt{3}\mathrm{s}_{\psi_1} \right) & \mathrm{c}_\varphi \left(\sqrt{3}\mathrm{c}_{\psi_1} - \mathrm{s}_{\psi_1} \right) & -2\mathrm{s}_\varphi \\ \mathrm{c}_\varphi \left(-\mathrm{c}_{\psi_1} + \sqrt{3}\mathrm{s}_{\psi_1} \right) & \mathrm{c}_\varphi \left(-\sqrt{3}\mathrm{c}_{\psi_1} - \mathrm{s}_{\psi_1} \right) & -2\mathrm{s}_\varphi \end{bmatrix}}_{D} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{6.4}$$

The inverse relation is characterized by D^{-1}:

$$D^{-1} = \frac{r}{3R} \begin{bmatrix} 2\mathrm{c}_{\psi_1} \mathrm{c}_\varphi^{-1} & -\left(\sqrt{3}\mathrm{s}_{\psi_1} - \mathrm{c}_{\psi_1} \right) \mathrm{c}_\varphi^{-1} & \left(\sqrt{3}\mathrm{s}_{\psi_1} - \mathrm{c}_{\psi_1} \right) \mathrm{c}_\varphi^{-1} \\ 2\mathrm{s}_{\psi_1} \mathrm{c}_\varphi^{-1} & \left(\sqrt{3}\mathrm{c}_{\psi_1} - \mathrm{s}_{\psi_1} \right) \mathrm{c}_\varphi^{-1} & \left(-\sqrt{3}\mathrm{c}_{\psi_1} - \mathrm{s}_{\psi_1} \right) \mathrm{c}_\varphi^{-1} \\ -\mathrm{s}_\varphi^{-1} & -\mathrm{s}_\varphi^{-1} & -\mathrm{s}_\varphi^{-1} \end{bmatrix} \tag{6.5}$$

The same set of matrices relates the vector of wheel torques T_w to a torque acting on the ball T_b, so that D effectively decouples the system. By means of D^{-1}, the vector of motor speeds is converted to the ball velocity. The ball velocity is then controlled via the dynamics in the xz- and yz-planes[(1)] separately using a measurement of the attitude and the ball velocity in the respective control planes. The control laws yield a virtual ball torque that is transformed to the individual motor torques via D so that the loop is closed.

[(1)] The decoupling also gives rise to a system in the xy-plane. The latter however behaves fundamentally different from the other two systems because of the absence of a gravitational influence. The controller design of this system is not discussed here because of its simplicity.

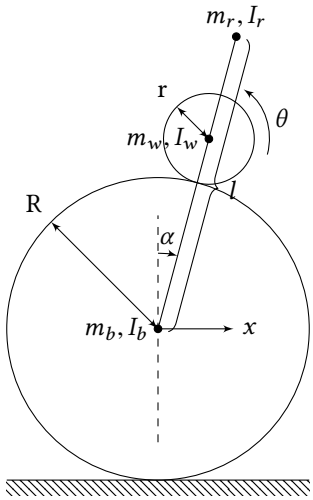


Figure 6.2 · Schematic of the 2D ballbot model in the xz- and yz-plane.

6.1.2 Nonlinear 2D model

Given the kinematic decoupling presented in the previous section, the dynamics in the xz- and yz-plane reduce to the 2D setting depicted in Figure 6.2. The position of the ball with mass m_b and mass moment of inertia I_b is described by the translational coordinate x. The ball is actuated by a virtual wheel with mass m_w and mass moment of inertia I_w. The wheel's total angular displacement is represented by θ. The body of the robot, characterized by mass m_r and mass moment of inertia I_r, rotates with respect to the center of the ball. The distance between their centers of mass is l and the angular displacement with respect to the gravitational force is α. Similarly to section 4.1.1, the Euler-Lagrange equation is used to derive a physical model. The kinetic energy T and the potential energy V are given by

$$\begin{aligned} T = {} & \frac{1}{2} m_b \dot{x}^2 + \frac{1}{2} I_b \left(\frac{\dot{x}}{R} \right)^2 + \frac{1}{2} m_r \left((\dot{x} - l\dot{\alpha} \cos\alpha)^2 + (l\dot{\alpha} \sin\alpha)^2 \right) + \frac{1}{2} I_r \dot{\alpha}^2 \\ & + \frac{1}{2} m_w \left((\dot{x} - (R+r)\,\dot{\alpha} \cos\alpha)^2 + ((R+r)\,\dot{\alpha} \sin\alpha)^2 \right) + \frac{1}{2} I_w \dot{\theta}^2 \end{aligned} \tag{6.6a}$$

$$V = g m_r l \cos\alpha + g m_w (r + R) \cos\alpha, \tag{6.6b}$$

with

$$x = r\theta + R\alpha \tag{6.7}$$

because of the kinematic coupling between the wheel, body and ball. The nonlinear differential equations describing the system's dynamics become

$$F_b = m_e\ddot{x} + \lambda\dot{\alpha}^2\cos\alpha - \kappa\ddot{\alpha} \tag{6.8a}$$

$$T_r = I_e\ddot{\alpha} - g\lambda\sin\alpha - \kappa\ddot{x} \tag{6.8b}$$

with

$$m_e = m_b + m_r + m_w + \frac{I_b}{R^2} + \frac{I_w}{r^2} \tag{6.9}$$

$$I_e = m_r l^2 + I_r + m_w (R+r)^2 + \frac{I_w}{r^2}R^2 \tag{6.10}$$

$$\lambda = m_r l + m_w (R+r) \tag{6.11}$$

$$\kappa = \frac{I_w}{r^2}R + \lambda\cos\alpha. \tag{6.12}$$

Expressions for T_r and F_b as a function of the wheel torque T_w are derived via the coupling of the wheel with the body and the ball:

$$F_b = \frac{1}{r}T_w \tag{6.13a}$$

$$T_r = -\frac{R}{r}T_w. \tag{6.13b}$$

To relate the torque T_w generated by the motor to the applied voltage V, a simplified version (6.14) of the standard model, e.g. Zaccarian (1998), that excludes the high-frequency dynamics coming from the electrical components, is proposed. The model depends on the motor's internal resistance ρ and the motor constant k.

$$\frac{V}{\rho} = \frac{T_w}{k} + \frac{k}{\rho}\dot{\theta} \tag{6.14}$$

It is interesting to see that values of $k \to 0$ make the motor act as a torque source, whereas values of $k \to \infty$ transform the motor in a speed source. Because the motors that are installed on the robot considered here have a very low transmission ratio (1:19), it is fair to assume the motor to act like a speed source so that (6.14) simplifies to

$$V \approx k\dot{\theta} = \frac{k}{r}\left(\dot{x} - R\dot{\alpha}\right). \tag{6.15}$$

By eliminating the wheel torque from equations (6.8) and (6.13), and by substituting (6.15), expression (6.16) is obtained.

$$(\kappa - Rm_e)\frac{r}{k}\dot{V} = \left(R^2 m_e + I_e - 2R\kappa\right)\ddot{\alpha} + \lambda\dot{\alpha}^2\cos\alpha - g\lambda\sin\alpha \tag{6.16}$$

Together, equations (6.15) and (6.16) constitute the nonlinear dynamic equations of a voltage-driven ball-balancing robot.

6.1.3 Linearized model

Based on equations (6.15) and (6.16), the equilibrium conditions are as follows:

$$\alpha = 0 \tag{6.17a}$$
$$\dot{\alpha} = 0 \tag{6.17b}$$
$$\ddot{\alpha} = 0 \tag{6.17c}$$
$$\dot{x} = 0 \tag{6.17d}$$
$$\ddot{x} = 0 \tag{6.17e}$$
$$V = 0 \tag{6.17f}$$
$$\dot{V} = 0 \tag{6.17g}$$

The resulting linearized dynamic equations in terms of small perturbations δ of the equilibrium state, then become

$$(\kappa_0 - Rm_e)\frac{r}{k}\delta\dot{V} = I_\alpha \delta\ddot{\alpha} - g\lambda\delta\alpha \tag{6.18a}$$
$$\delta V = \frac{k}{r}\left(\delta\dot{x} - R\delta\dot{\alpha}\right) \tag{6.18b}$$

with

$$\kappa_0 = \frac{I_w}{r^2}R + \lambda \tag{6.19}$$
$$I_\alpha = R^2 m_e + I_e - 2R\kappa_0. \tag{6.20}$$

First of all, equation (6.18a) is driven by $\delta\dot{V}$ resulting in a differentiator in the attitude dynamics. As a consequence, a controller that achieves decent tracking has to contain an integrator. Second, elaborating I_α results in

$$I_\alpha = m_b R^2 + m_w r^2 + m_r (l - R)^2 + I_b + I_w, \tag{6.21}$$

which is clearly positive. Because λ is also positive, equation (6.18a) produces an unstable pole located at $\sqrt{g\lambda/I_\alpha}$. Therefore, a controller with sufficient phase lead will be necessary to stabilize the system. Third, suppose the goal is to track a non-zero fixed $\delta\alpha$, so that $\delta\dot{\alpha}$ and $\delta\ddot{\alpha}$ are both zero. By combining (6.18a) and (6.18b), it is inferred that this requires a fixed acceleration which is proportional to the attitude $\delta\alpha$.

6.1.4 Attitude identification

The identification of the attitude model happens in a closed-loop setting because the system is unstable. This implies that an initial stabilizing controller has been obtained,

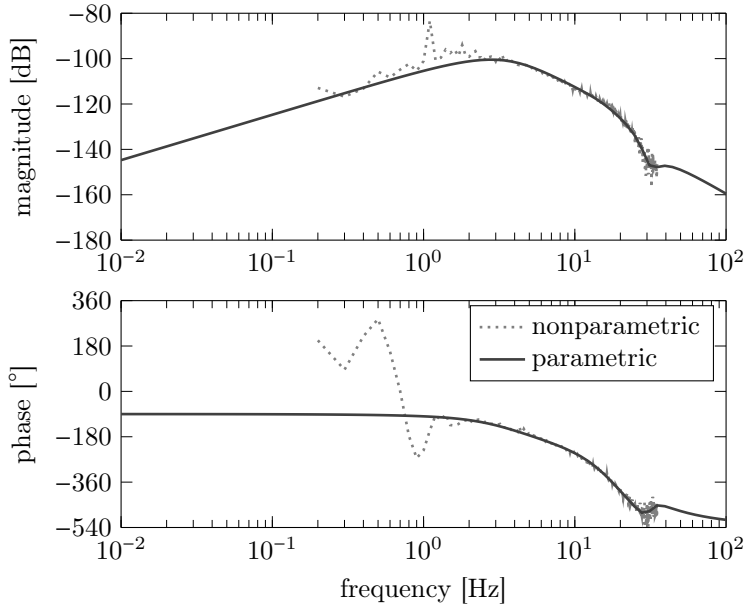


Figure 6.3 · Nonparametric and parametric model obtained via a multisine identification.

e.g. a hand-tuned PID controller. According to (see Pintelon and Schoukens (2012a) section 7.2.7), an unbiased estimate of the dynamics $\mathcal{P}_a : u_a \rightarrow y_a$ is obtained by applying a disturbance d_a to the controller output u_a and by measuring the corresponding attitude y_a. In that case,

$$\mathcal{P}_a = \mathcal{D}_a \mathcal{S}_a^{-1} \tag{6.22}$$

provides the unbiased estimate, where $\mathcal{D}_a : d_a \rightarrow y_a$ and $\mathcal{S}_a : d_a \rightarrow (u_a + d_a)$.

To identify a nonparametric model of $\mathcal{P}_a$, the robust method with random multisine excitations is used (see Pintelon and Schoukens (2012a) section 4.3.1). Based on this nonparametric model, a parametric estimate $\mathcal{P}_a$ is computed via a nonlinear least-squares formalism (see Pintelon and Schoukens (2012a) section 9.9). More details on the settings of the identification experiment are found in the appendix section A.1. The resulting nonparametric and parametric estimates of $\mathcal{P}_a$ are shown in Figure 6.3. It is clear that the estimated models confirm the behavior predicted by the physical model. The open loop contains an unstable pole at 2.65 Hz and displays a clear differentiator. However, at low frequencies the nonparametric model appears to be rather noisy. The explanation lies with the fact that the closed-loop system is not responsive at

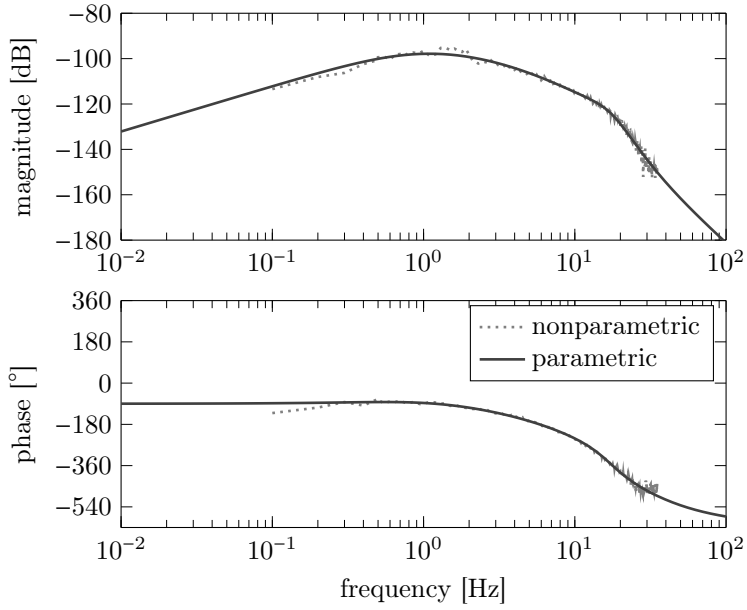


Figure 6.4 · Nonparametric and parametric model obtained via a stepped sine identification.

low frequencies due to the controller's intent to suppress disturbances, resulting in a low signal-to-noise ratio.

To overcome this problem, a stepped sine identification (Swevers et al., 1992) was carried out in the lower frequency region, i.e. $[0.1,\ 2.0]$ Hz. Because this excitation signal packs all of its energy in a single frequency, the highest possible signal-to-noise ratio is obtained at the expense of more time-consuming experiments. More details on the experimental conditions are found in the appendix, section A.2. The resulting model is combined with the higher frequency part of a nonparametric model identified with a multisine signal. This leads to a nonparametric model with a lower level of uncertainty and as a consequence a better parametric model. Both models are shown in Figure 6.4. The new model suggests that the unstable pole is located at a much lower frequency 0.77 Hz and as a consequence the model has a higher gain at low frequencies.

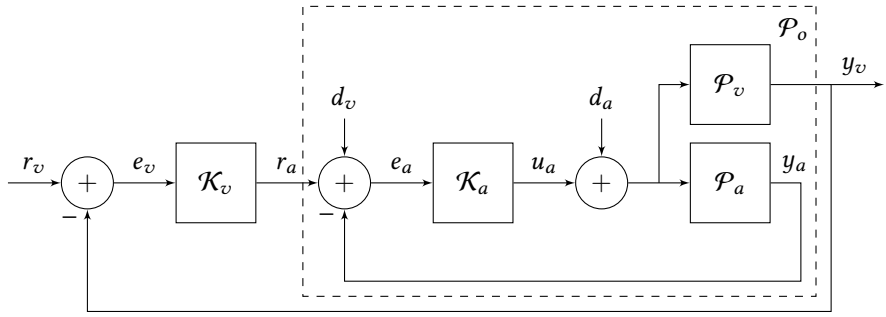


FIGURE 6.5 · Schematic of the cascade control loop. The inner controller $\mathcal{K}_a$ stabilizes the attitude y_a whereas the outer controller $\mathcal{K}_v$ provides an attitude setpoint r_a, based on a reference for the velocity r_v.

6.1.5 Velocity identification

There are two ways to describe the system's velocity response. One way is to consider the velocity to be a result of the same actuation that drives the attitude system identified in section 6.1.4, which essentially yields two systems $\mathcal{P}_a$ and $\mathcal{P}_v$ acting in parallel. Another line of thinking is that the attitude acts as a driver for the velocity, which is justified by the observation that the ball's acceleration is proportional to the attitude (see section 6.1.3). In this case, the latter is opted for because it leads to an intuitive cascade controller setting, depicted by Figure 6.5. The attitude loop is closed first, stabilizing the attitude. Next a model $\mathcal{P}_o$ of the velocity y_v in response to an attitude reference r_a is identified. It should be noted that the attitude controller stabilizing the inner loop is the controller designed in section 6.2.1.

Again a closed-loop multisine identification is performed. Although the attitude loop already stabilizes the system making an open-loop identification procedure possible, a closed-loop identification is preferred. This is because small disturbances e.g. coming from imperfections of the ground's surface, are then counteracted by the velocity controller, resulting in better estimate.

Because of the closed-loop setting, applying a disturbance to the control signal does not result in an equal excitation in all frequency ranges. A decent controller will be able to counteract low-frequency disturbances so that the system is not excited in this frequency range. Higher frequencies are not counteracted and do appear in the measured output's frequency content. The opposite holds when applying the disturbance to the measured output. The closed loop's tracking behavior tries to follow the disturbance, but can only do so up to the closed-loop bandwidth. Higher frequencies disappear from the frequency content. Therefore, it is proposed to identify two nonparametric models, each corresponding to one of the different excitation cases.

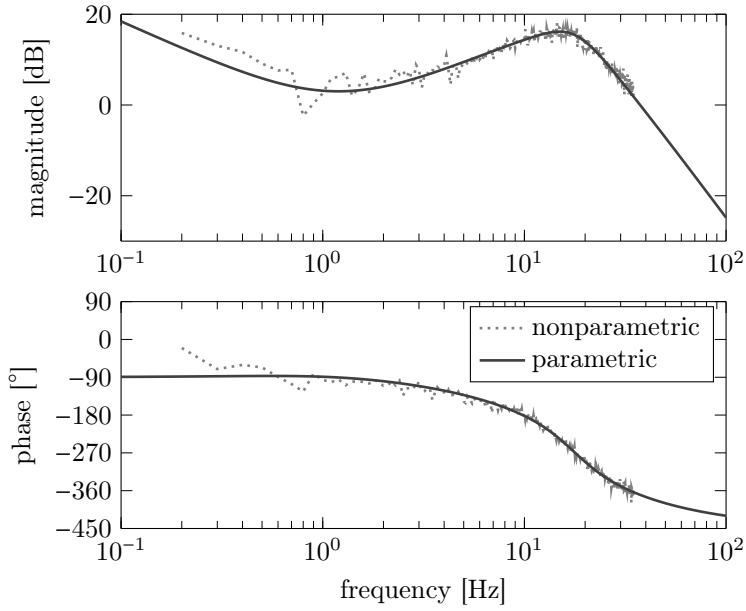


Figure 6.6 · Parametric and nonparametric model of $\mathcal{G}_o$.

The low-frequency part of the one model is then combined with the high-frequency content of the other, resulting in a more reliable estimate of the overall frequency response function. This model is then used to estimate a parametric transfer function via a nonlinear least-squares formalism. The nonparametric and parametric model are shown in Figure 6.6. More details on the identification experiments are given in the appendix, section A.3. Because

$$\mathcal{P}_o = \mathcal{P}_v \mathcal{K}_a \mathcal{S}_a, \tag{6.23}$$

it is possible to separate the different contributions within the identified model. Most interesting is the nonminimum phase zero at 1.44 Hz in $\mathcal{P}_o$ that comes from $\mathcal{S}_a$. Although this zero should in theory have the same natural frequency as the unstable pole of $\mathcal{P}_a$, the identification procedure provides a slightly different estimate. Also $\mathcal{K}_a \mathcal{S}_a$ is constant in the low frequency-range, so that the dominant low-frequency behavior of $\mathcal{P}_o$ is a result of an integrator in $\mathcal{P}_v$. This observation is in line with the earlier derivation that the acceleration of the ball is proportional to the robot's attitude.

6.2 Cascade controller design

Designing a cascade controller is one of the more obvious design strategies for a system with multiple outputs. The idea is to close two control loops subsequently where the outer loop drives the inner loop. It is chosen to have the inner loop control the attitude $\mathcal{P}_a$ so that the outer velocity loop has a stable system $\mathcal{P}_o$ to control. The resulting control configuration is depicted in Figure 6.5. The following sections discuss the controller design for these two control loops. In the remainder of the text, the inputs and outputs of a transfer function are explicitly mentioned because of the somewhat complex nature of the system. An example hereof is $\mathcal{S}_{e_a r_a}$, which denotes the closed-loop transfer function with input r_a and output e_a.

6.2.1 Attitude controller design

The attitude controller design is based on a standard objective and set of constraints. Because the system is unstable and nonlinear, a high degree of robustness is desirable. Although the controller design might benefit from taking the model uncertainty into account explicitly, putting a rather stringent constraint on the peak of the sensitivity $\mathcal{S}_{e_a r_a}$,

$$\left\| \mathcal{S}_{e_a r_a} \right\|_\infty \leqslant 3 \text{ dB}, \tag{6.24}$$

is opted for since it avoids additional modeling effort and proves to be a practical alternative to ensure robustness. Constraint 6.24 results in a weight

$$\mathcal{W}_1 = 10^{-3/20}. \tag{6.25}$$

A second constraint is put on the complementary sensitivity $\mathcal{T}_{y_a r_a}$. To prevent nervous behavior in response to sensor noise, the cross-over frequency of $\mathcal{T}_{y_a r_a}$ is chosen 15 Hz. The corresponding weight

$$\mathcal{W}_2 = \frac{1000 s^3}{(s + 942)\left(s^2 + 942 s + 942^2\right)} \tag{6.26}$$

has a third order roll-off towards high frequencies with a high-frequency penalty of 60 dB. This yields a controller without high-frequency roll-off preventing additional phase lag that has to be compensated for near the cross-over frequency.

The remaining freedom is exploited to maximize the bandwidth of $\mathcal{S}_{e_a r_a}$. A simple first order low-pass filter

$$W_3 = \frac{6.39}{s + 1.14} \tag{6.27}$$

with a low-frequency gain of 15 dB is proposed as a weight. Its cross-over frequency equals 1 Hz, which is chosen to be in line with the natural frequency of the unstable

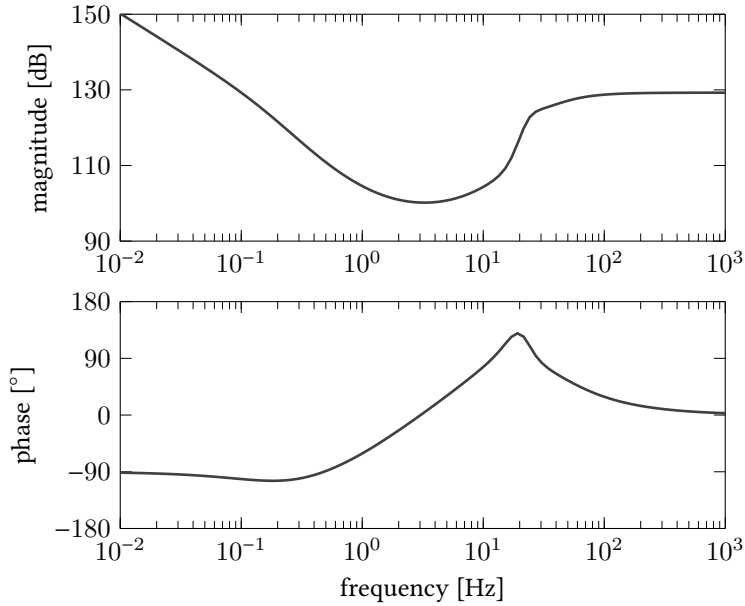


Figure 6.7 · Bode diagram of the attitude controller $\mathcal{K}_a$ for the cascade configuration.

pole. Note that the influence of the exact values is limited because W_3 is being optimized.

The resulting optimization problem

$$\begin{aligned} \text{minimize} \quad & \left\| \mathcal{W}_3 \mathcal{S}_{e_a r_a} \right\|_\infty && \text{(6.28a)} \\ \text{subject to} \quad & \left\| \mathcal{W}_1 \mathcal{S}_{e_a r_a} \right\|_\infty \leqslant 1 && \text{(6.28b)} \\ & \left\| \mathcal{W}_2 \mathcal{T}_{y_a r_a} \right\|_\infty \leqslant 1 && \text{(6.28c)} \end{aligned}$$

is solved with the Linear Control Toolbox (see Chapter 3). However, the objective (6.28a) renders the problem infeasible because an integrator is required in the controller to compensate the differentiator in the model. Though the resulting closed loop can be made input-output stable, it is only marginally internally stable hence the infeasibility. To overcome this issue, the attitude model is precompensated by a PI controller that cancels the internal marginally stable mode so that an optimal controller can be found. The resulting combined controller, shown in Figure 6.7, provides the expected phase lead characteristic for the control of unstable systems. Also high-frequency roll-off is absent as a consequence of the choice of $\mathcal{W}_2$.

6.2.2 Velocity controller design

The controller design for the velocity loop is dominated by the nonminimum phase zero. A nonminimum phase zero imposes an upper limit on the bandwidth of the system (see Skogestad and Postlethwaite (2001), section 5.6). As a rule of thumb this limit is half of the natural frequency of the nonminimum phase zero. Because of this inherent restriction, (6.29) is proposed as a design.

$$\begin{aligned} \text{minimize} \quad & \left\| \mathcal{W}_3 \mathcal{U}_{r_a r_v} \right\|_\infty && \text{(6.29a)} \\ \text{subject to} \quad & \left\| \mathcal{W}_1 \mathcal{S}_{e_v r_v} \right\|_\infty \leqslant 1 && \text{(6.29b)} \\ & \left\| \mathcal{W}_2 \mathcal{S}_{e_v r_v} \right\|_\infty \leqslant 1 && \text{(6.29c)} \end{aligned}$$

The weight

$$\mathcal{W}_1 = \frac{2.65}{s + 0.838} \tag{6.30}$$

is designed to suppress a tracking error at low frequencies. To this end, a simple first order low-pass filter with a DC gain of 10 dB is used. The cross-over frequency is chosen to be 0.4 Hz. Therefore, the constraint pushes the bandwidth of the closed loop but stays well below the limitation imposed by the nonminimum phase zero. The second constraint on $\mathcal{S}_{e_v r_v}$ ensures a reasonable degree of robustness by imposing a limit on the peak sensitivity. Sensible values range from 3 dB to 6 dB. Here, a value of 5 dB is chosen, resulting in

$$\mathcal{W}_2 = 10^{-5/20}. \tag{6.31}$$

The remaining freedom is used to enforce high-frequency roll-off on the controller via $\mathcal{U}_{r_a r_v}$. This limits the influence of sensor noise and unmodeled dynamics typically present at higher frequencies. To this end, a second order high-pass filter $\mathcal{W}_3$ with a high-frequency gain of 60 dB and a cross-over frequency of 0.5 Hz is used.

$$\mathcal{W}_3 = \frac{1000}{s^2 + 99.3\sqrt{2}s + 99.3^2} \tag{6.32}$$

Solving optimization problem (6.29) with the Linear Control Toolbox results in a fairly simple second order lag controller, displayed in Figure 6.8, with poles at $-1.38 \pm 0.20j$. Such a controller was to be expected because i) the design requires an appropriate proportional feedback at low frequencies to meet the constraints on $\mathcal{S}_{e_v r_v}$ and ii) roll-off is required in the neighborhood of the nonminimum phase zero to compensate the increasing magnitude.

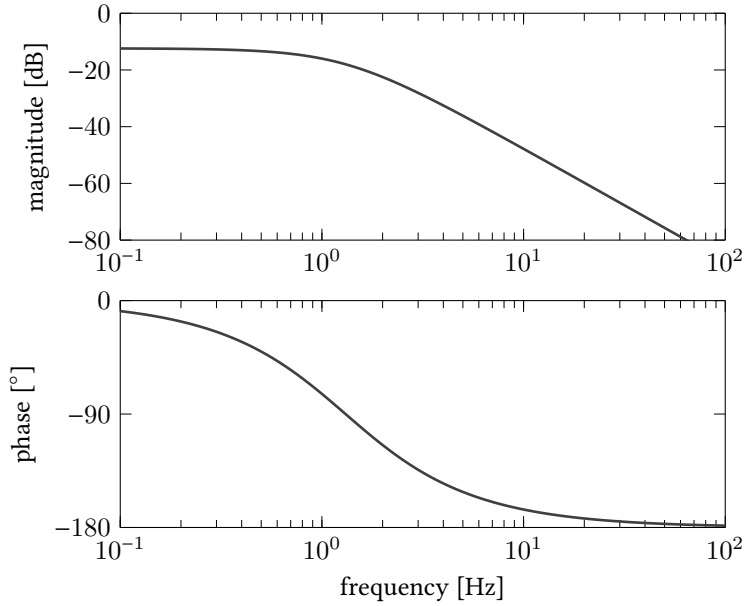


Figure 6.8 · Bode diagram of the velocity controller $\mathcal{K}_v$ for the cascade configuration.

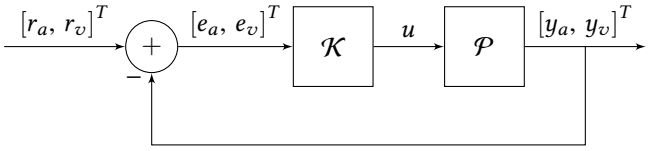


Figure 6.9 · Control configuration for the single-step MISO controller design.

6.3 Single-step controller design

This section describes an alternative to the controller design strategy presented in section 6.2. Instead of the sequential design of a separate attitude and velocity controller, a multiple-input-single-output (MISO) controller is designed in a single step. The corresponding control configuration is depicted in Figure 6.9. To do so, a single-input-multiple-output (SIMO) model of the system is derived from the identification experiments performed in sections 6.1.4 and 6.1.5. Next, a corresponding controller is designed, from here on referred to as the MISO controller.

6.3.1 Model derivation

In order to design a MISO controller, a SIMO model

$$\mathcal{P} = \left[\frac{\mathcal{P}_a}{\mathcal{P}_v}\right] \tag{6.33}$$

is required. A model for $\mathcal{P}_a$ is available as a result of the identification carried out in section 6.1.4. Regarding $\mathcal{P}_v$, no model is directly available because the identification of section 6.1.5 captured the behavior of the inner loop of the cascade setting. However, rearranging equation (6.23) yields an expression for $\mathcal{P}_v$ in terms of the identified system $\mathcal{P}_o$ and the attitude control loop:

$$\mathcal{P}_v = \mathcal{P}_o \mathcal{S}_{e_a r_a}^{-1} \mathcal{K}_a^{-1}. \tag{6.34}$$

Although this produces an estimate of $\mathcal{P}_v$ with relatively high uncertainty because the uncertainties on the estimates of $\mathcal{P}_o$ and $\mathcal{P}_a$ are added, it is assumed that the resulting model is sufficiently accurate for the controller design.

6.3.2 Controller design

The single-step velocity controller design proposed in this section combines the requirements imposed on the attitude and the velocity control loop during the cascade controller design in section 6.2. First, the tracking performance of the controller is guaranteed via a constraint on the velocity sensitivity, similar to (6.29b),

$$\left\| \mathcal{W}_1 \mathcal{S}_{e_v r_v} \right\|_\infty \leqslant 1 \tag{6.35}$$

where

$$\mathcal{W}_1 = \frac{1.89}{s + 0.0189} \tag{6.36}$$

is a first-order low-pass filter with a DC gain of 40 dB and cross-over frequency of 0.3 Hz. The robustness constraints similar to (6.24) and (6.29c)

$$\left\| \mathcal{S}_{e_a r_a} \right\|_\infty \leqslant 2.5\,\text{dB} \tag{6.37a}$$

$$\left\| \mathcal{S}_{e_v r_v} \right\|_\infty \leqslant 5\,\text{dB} \tag{6.37b}$$

are enforced as well resulting in $\mathcal{W}_2 = 10^{-2.5/20}$ and $\mathcal{W}_3 = 10^{-5/20}$. The remaining freedom is used to minimize the actuation signal's high-frequency content. To this end, a second order high-pass filter

$$\mathcal{W}_4 = \frac{10000}{s^2 + 1257\sqrt{2}s + 1257^2} \tag{6.38}$$

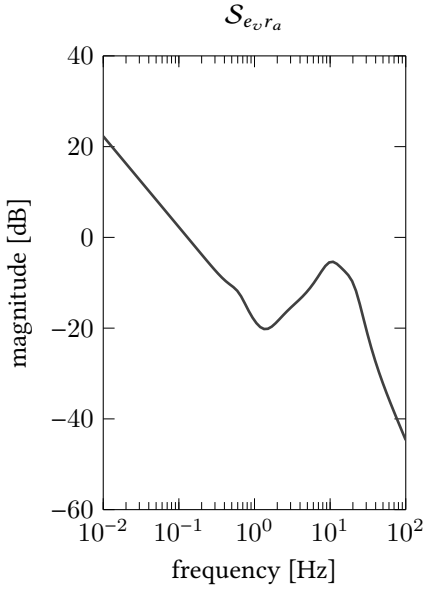


Figure 6.10 · Response of the velocity error e_v to an attitude reference r_a.

with cross-over frequency of 2 Hz and a high-frequency gain of 80 dB is used:

$$\left\| \mathcal{W}_4 \mathcal{U}_{ur_v} \right\|_\infty \leqslant 1. \tag{6.39}$$

The resulting optimization problem

$$\begin{aligned} \text{minimize} \quad & \left\| \mathcal{W}_4 \mathcal{U}_{ur_v} \right\|_\infty && \text{(6.40a)} \\ \text{subject to} \quad & \left\| \mathcal{W}_1 \mathcal{S}_{e_v r_v} \right\|_\infty \leqslant 1 && \text{(6.40b)} \\ & \left\| \mathcal{W}_2 \mathcal{S}_{e_a r_a} \right\|_\infty \leqslant 1 && \text{(6.40c)} \\ & \left\| \mathcal{W}_3 \mathcal{S}_{e_v r_v} \right\|_\infty \leqslant 1 && \text{(6.40d)} \end{aligned}$$

is solved with the Linear Control Toolbox (see Chapter 3). As it turns out, the LMI based solvers are not capable of retrieving a solution to the problem, probably due to the size of the problem, i.e. a generalized plant of 22^{nd} order. In order to find a solution, the toolbox is forced to use `systune`, which does not rely on an LMI reformulation. Because it is unlikely that a 22^{nd} order controller is required to reach sufficient performance, the toolbox is instructed to limit the controller's order to 8. This yields a solution that behaves undesirably in terms of the cross-coupling between r_a and e_v, shown in Figure 6.10. This figure shows how $\mathcal{S}_{e_v r_a}$ behaves like an integrator towards low frequencies. As a consequence, a DC disturbance on the

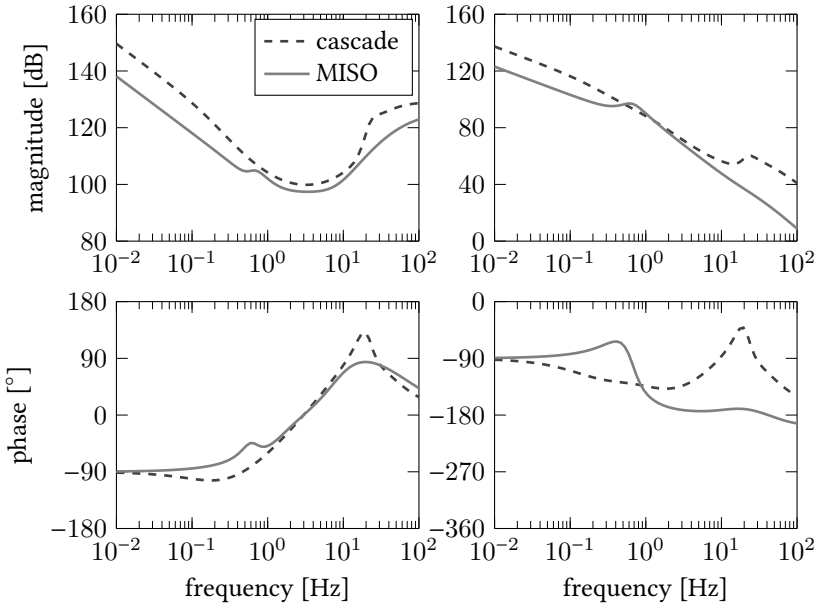


Figure 6.11 · Bode diagram of the cascade and the MISO controller.

attitude coming from e.g. a calibration error or an unbalance of the body results in an arbitrarily large velocity error. This is obviously undesirable behavior and needs to be counteracted. To this end, an additional constraint is put on $\mathcal{S}_{e_v r_a}$:

$$\left\| \mathcal{W}_5 \mathcal{S}_{e_v r_a} \right\|_\infty \leqslant 1. \tag{6.41}$$

The low-pass filter

$$\mathcal{W}_5 = \frac{44.5}{s + 31.5} \tag{6.42}$$

suppresses the low-frequency content to −3 dB up to a frequency of 5 Hz. Figure 6.11 shows the resulting controller along with the cascade controller of section 6.2. The main difference between these controllers appears to be the relative importance of the attitude and the velocity error in the construction of the actuation. In the neighborhood of the cross-over frequency, the cascade controller has approximately 6 dB more gain on the attitude error than the MISO controller. The opposite holds for the velocity error.

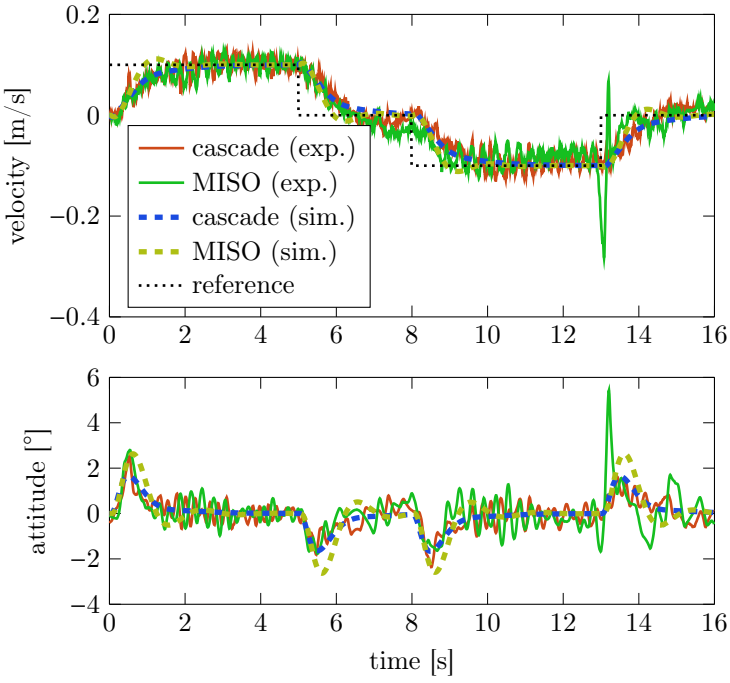


Figure 6.12 · Experimental validation of the cascade and MISO controller. A forward-backward motion in the xz-plane is requested with a constant velocity of $\pm 0.1\,\mathrm{m\,s^{-1}}$.

6.4 Validation

An experimental validation was done to assess the practical value of both the cascade and the MISO controller. The experiment consists of a simple forward-backward motion so that the controller's tracking behavior is observed. As shown in Figure 6.12, the velocity reference in the x-direction is set to $0.1\,\mathrm{m\,s^{-1}}$ and $-0.1\,\mathrm{m\,s^{-1}}$ followed by a standstill. The solid lines indicate the measured response whereas the response predicted by the models is given by the dashed lines. Generally speaking, the measurements follow the trend of the prediction. However, both measurements display a similar vibration that is not predicted by the model. A modeling error coming from the identification or the fact that the linear model is not capable of capturing important nonlinear dynamics are possible explanations but more experiments need to be performed to draw a conclusion. The measurement with the MISO controller also displays interesting behavior after 13 s, which is when the robot is instructed to stop its backward motion. This appears to trigger a large peak in the attitude and

the velocity. This clearly has to do with nonlinearities within the system because such behavior does not occur when ending a forward motion. Similar behavior was reported by Yang et al. (2015). They point out that the normal force on the wheel is proportional to $\cos^{-1}(\varphi)$. Because of the relatively small $\varphi \approx \pi/4$, the normal force becomes insufficient to transfer a high torque from the motors to the ball during swift maneuvers. The reason for this happening only in the backward direction is that in this direction, the body tips in such a way that the effective angle φ reduces. During a forward motion, the body tips in the opposite direction, increasing the normal force on the wheels.

The reason this issue seems to arise only when the loop is closed with the MISO controller can probably be attributed to the individual contributions of the attitude and velocity measurements to the control signal. The MISO controller attaches more importance to the velocity measurement around the cross-over frequency than the cascade controller. However, it is this measurement that becomes corrupted when the wheel slips, degrading the closed-loop performance.

6.5 Conclusion

This chapter presented the main steps taken to come to a velocity controller for a ball balancing robot. First, it was pointed out how the dynamics of the 3D system are decoupled via a linear transformation based on the robot's geometry. This led to two decoupled 2D models for which a nonlinear model was derived. To gain more insight this nonlinear model was linearized around the robot's equilibrium state. It was observed that the system is open-loop unstable and that the input is proportional to the robot's speed, which was confirmed by an identification of the system. Based on the identified models, both a cascade controller and a MISO controller were designed and experimentally validated. The experiments show a response that is in general in line with the predicted response. Furthermore, the effect of the nonlinear torque transfer as a function of the attitude is visible when the loop is closed by the MISO controller.

7

Conclusion

Advanced feedback controller design methodologies have the potential to become a worthy alternative to standard design methodologies. Especially the control of high-end complex systems could benefit from these methodologies. In view of this belief, the previous chapters introduced the Linear Control Toolbox and demonstrated its power on various mechatronic systems. The following sections give a concise summary of these chapters and a series of suggestions to further improve this work.

7.1 Summary

Chapter 3 introduced the cornerstone of part I of this thesis, the Linear Control Toolbox. This open-source Matlab toolbox originated from the observation that advanced feedback controller design strategies do not find their way to industrial application, although they have been around for several decades. The Linear Control Toolbox provides a higher level of abstraction to apply these methods in an attempt to erase the industrial reluctance. Key features are the signal-based modeling framework that allows the user to construct the control configuration, the identification module that interfaces several routines in a uniform manner and the control module that allows the user to specify an optimal feedback controller design problem in a natural way and solve it as such. Moreover the Linear Control Toolbox supports state-of-the-art B-spline-based LPV modeling and controller design techniques, unlocking the true potential of advanced feedback controller design methodologies. A comparative case study considering the design of a missile controller was presented to position this new toolbox with respect to the state-of-the-art.

Because an experimental validation is more convincing than simulation results, the Linear Control Toolbox was extensively tested on real-life setups. Chapter 4 describes the first application of the toolbox. The goal is to control the position of the load of a lab-scale overhead crane. Because the crane is capable of hoisting its load, the system's dynamics depend on the length of the cable. To obtain a model with sufficient confidence, an identification procedure was carried out, eventually yielding a B-spline LPV model. This model was used to design a controller which was then experimentally

validated. The experimental results show great resemblance to the toolbox's predicted response.

Chapter 5 presents a second industrially relevant application, i.e. the control of the radial active magnetic bearings (AMBs) of a high-speed rotordynamic test setup. Mainly the unstable nature of the machine turns this design into a challenging exercise. This property makes it hard to achieve sufficient closed-loop damping, which happens to be the primal design specification. However, the advanced controller design techniques shipped with the toolbox prove to be perfectly suited to this problem. Whereas the unstable nature of the system would have a drastic effect on the open-loop design paradigm, the closed-loop design paradigm remains unchanged. Also, contrary to an open-loop design formalism, the constraint on the closed-loop damping fits neatly in the closed-loop design framework. An experimental validation of the obtained controllers proved their performance meets the specifications.

The last experimental validation, described in chapter 6, was carried out on a ball-balancing robot. Although the setup might be rather academical, the complex mechatronic nature of the system allows the Linear Control Toolbox to show what it is capable of. Not only is the system open-loop unstable, it also deviates from the traditional single-input-single-output (SISO) setting. This allows the Linear Control Toolbox to perform both a two-step cascaded controller design as well as a single-step MISO controller design. Both controllers were experimentally validated and exhibit decent performance, given the limitations of the platform.

Although the Linear Control Toolbox is a relatively new software package, two industrial partners already benefited from it. Company A has been using the Linear Control Toolbox from the beginning and gradually adopted the closed-loop design paradigm up to the point of being capable of totally independently designing controllers. Moreover, the company already has one product running a control algorithm that was designed with the Linear Control Toolbox and more are to come. Company B only recently started showing interest in the closed-loop controller design paradigm. They managed to design an LPV controller for one of their mobile platforms. Also, the industrial impact is stimulated via the Flanders Make ROCSIS SBO-project in which the Linear Control Toolbox is being developed further and is being applied to industrial cases. The Linear Control Toolbox has also become a valuable instrument to teach the closed-loop design paradigm within the KU Leuven master's course 'Advanced model-based control'. The toolbox allows students to focus on the problem formulation instead of the mathematical background, improving their ability to design controllers with this new paradigm.

7.2 Suggestions

Doing research often raises more questions than it answers. This work was no exception. The following paragraphs give an overview of insights and thoughts that have not found their way to this thesis but are worth mentioning.

7.2.1 State-space identification

The identification process is probably the most underappreciated aspect of designing a decent controller. Although first principles models might be available, it happens rarely that the necessary parameters are known with sufficient confidence, making the process of identification invaluable to the control engineer. For understandable reasons, many identification routines are designed to come up with a transfer function model. This however poses a problem to the advanced controller design methodologies, which typically expect state-space models. Whereas a SISO transfer function model is readily transformed in an equivalent state-space model, the generalization to the MIMO case is no longer obvious. Including direct state-space identification techniques is therefore crucial to lift the toolbox's applicability to a higher level.

7.2.2 Clean controllers

An issue that has been meticulously hushed up throughout this thesis is the consequence of the fact that weights have to be proper and stable. This typically leads to poles and zeros located orders of magnitude away from the cross-over region. To some extent, a control engineer is able to take on the process of cleaning this controller, retaining only the valuable part of the controller. However, this operation becomes rather hard in the case of a MIMO controller and even impossible in the case of an LPV controller. The need to cope with this problem at a design level is obvious and triggers research in this direction. To prevent this low- and high-frequency rubbish, Köroğlu (2013) and Masubuchi (2007) propose to tackle the problem at the level of the LMIs. Another approach might be to use intelligent pre- and post-filtering of the controller so that the equivalent weights become stable and proper, much like the idea presented by Meinsma (1995), but how this process should be automated remains an open question.

7.2.3 Robust controller design

An interesting problem is that of a robust controller design. This comes down to finding an LTI controller, capable of stabilizing a family of systems, which is especially useful in a setting where the variation on the system dynamics is known. Several ways of describing this variation exist, but these are only supported by the Linear Control Toolbox to a small extent. The $\mathcal{H}_\infty$-framework allows unstructured uncertainty (see Skogestad and Postlethwaite (2001), chapter 7) which is to some extent supported by the Linear Control Toolbox. However, this approach easily leads to conservative results, especially in a MIMO setting. In order to reduce conservatism in the presence of uncertainty, μ-synthesis emerged as design methodology. Because of the complexity of the problem, only approximate solutions such as the D-K-iteration (see Skogestad and Postlethwaite (2001), section 8.12) exist. Alhtough convergence is not guaranteed, this method has the advantage that it is able to fully exploit the uncertainty's structure, introducing no additional conservatism. Another approach would be to describe the system's dynamics using an LPV model and providing an LTI controller as the design variable. This approach might be perceived as more natural and straightforward since the parametric uncertainty is modeled in a direct manner. A nonconvex BMI solver as the one presented by Tran Dinh et al. (2012) should be able to yield a solution.

7.2.4 Controller deployment

Synthesizing a controller is usually only part of the job. In order to test the design, the controller has to be deployed on the corresponding setup. Depending on the controller's software framework and the controller's parameter dependency, this step's difficulty varies from straightforward to challenging. Exporting a controller designed in Matlab to another Matlab-based software framework such as Dspace or speedgoat can happen fairly quickly. Much more challenging is exporting a controller to some other programming language such as python or C++, especially when the controller is parameter varying and depends on the parameter in a nontrivial fashion. This poses a serious threat to the adoption of LPV design techniques by industry. Therefore, having a way of exporting the synthesized control laws seems like an obvious but compelling requirement.

7.2.5 Feedforward design

The Linear Control Toolbox as it has been described throughout this text focuses heavily on feedback control. However, incorporating feedforward control can substantially improve the overall performance of a system. Therefore, including feedforward design strategies in the toolbox might be an interesting course of action.

Part II

FLEXIBLE COMPONENTS FOR MECHATRONIC SYSTEMS

8

Introduction

8.1 Motivation

According to Google's search engine, engineering is the branch of science and technology concerned with the design, building and use of engines, machines and structures. This definition reveals two important aspects related to the field. On the one hand, a solid scientific basis is required, which can be associated with the design phase of a project. This is where the engineer has to apply his knowledge to come to a theoretical sketch of a possible solution. On the other hand, the definition stresses the technological contributions the engineer has to come up with to obtain a practical implementation. This is closely related to the second phase of an engineering project, i.e. the build phase.

Although the engineer's job is occasionally finished after a single design-and-build step, this is usually not the case. In the majority of the projects, multiple such steps are needed to come to a result that meets the specifications. A reason for this is the inevitable presence of modeling errors, i.e. the fact that some parts of the system do not behave as expected. Examples thereof are unmodeled dynamics that turn out to be more influential than anticipated, a high degree of uncertainty on some of the model parameters such as a material's strength or, let's face it, a human error.

The observation that a practical implementation turns out to almost never behave as predicted by the initial sketch on paper adds to the importance of building the system and validating the performance. In case the project relies upon new scientific insights, this experimental validation is even more important as it might confirm its practical applicability or reveal unexpected shortcomings. A striking example is the introduction of the front-drive A-Class of Mercedes-Benz, a revolutionary short but tall design. Shortly after its release, it flipped over during the so-called moose test, a swift lane-change at a constant moderate speed. Amongst others, the moose test uncovered that Mercedes testing procedures were at that time inadequate to judge the car's stability.

Although engineering at an academical level is mostly focused on theoretical or methodological contributions, a practical assessment of the theory is in general

deemed of high value to the community. Acknowledging the added value of experimental results, the MECO research group invests a large amount of time in practically validating its research. This brings a number of challenges related to different research projects. This part of the thesis addresses three of these challenges.

A first challenge is related to the control architecture of mechatronic systems. Cheap microcontrollers provide the opportunity to have a separate low-level controller running on a dedicated chip. This has several advantages such as a more transparent software architecture due to the physical separation of the different controllers and a guaranteed clock frequency of the low-level controller. The main drawback is the rather limited accessibility of a microcontroller, compared to a regular pc, resulting in a more cumbersome design phase. Streamlining the development of embedded software is therefore considered the first challenge within this part of the thesis.

The second challenge originates from the motion planning research line within the MECO group. In order to experimentally validate the developed strategies, a positioning system that is able to track multiple mobile robots is required. On top of that, the system is required to have a distributed architecture so that it fits within a factory of the future demonstration by Van Parys and Verbandt (2018). Because such a system is not readily commercially available, the development of such a system is considered a second challenge.

The third challenge is the need to incorporate flexibility in experimental setups. This challenge originates from the versatility of proposed solutions, typical for an academic setting. An example from within the control community is the existence of broad variety of control laws. Whereas one algorithm is tailored to run on low-cost embedded hardware (Van Parys and Pipeleers, 2018), another requires a high-end computer to be executed (Mercy et al., 2017). Keeping flexibility in mind leads to setups that are readily reused for different purposes, minimizing the overhead when experimentally validating a new idea.

8.2 Contributions

In an attempt to answer the challenges put forth in section 8.1, several technical contributions to tackle these challenges are presented in the following paragraphs.

8.2.1 Development of the MECO-CSI toolchain

In view of the first challenge of section 8.1, the MECO-CSI toolchain was developed, where CSI stands for control setup interface. This toolchain is designed to facilitate the development of embedded software and as such lowers the threshold to implement

the distributed control architecture proposed in challenge one. To meet this goal, two parts of software are created. MicroOS provides an embedded framework written on top of the Arduino API. This framework seamlessly interacts with QRoboticsCenter, a secondary application that runs on a pc and provides access to the device's internal variables and parameters. This is believed to accelerate the development on embedded hardware. QRoboticsCenter also takes the third challenge of section 8.1 into consideration, i.e. the need for flexibility. By making the internal variables available via a UDP port, it is possible for other programs, possibly running on another pc within the network, to interact with the embedded system.

8.2.2 Development of ProjectEagle

To meet the second challenge of section 8.1, ProjectEagle was started. The system consists of multiple smart cameras typically mounted on the ceiling. The mobile platforms are marked with a specific pattern that is localized by the image processing software. Each individual module broadcasts the position measurements on the network so that other devices within the network can access the data. Both the distributed architecture and the use of a standard network interface make the system easy to integrate and therefore the third challenge of section 8.1 is also met.

8.3 Outline

Besides this introduction and a final conclusion, part II of this thesis contains three chapters. A schematic overview of the relation between these chapters is given in Figure 8.1.

Chapter 9 introduces the MECO-CSI software toolchain, focusing on the fundamental idea. The chapter also provides an overview of the various mechatronic systems that were developed with this toolchain. Chapter 10 highlights the modular localization system, code-named ProjectEagle. Throughout the chapter, the two main aspects of the system are discussed, i.e. the calibration and the detection. To give the reader a feel of the ease with which the aforementioned systems are integrated, a demo within the field of motion planning is presented in chapter 11.

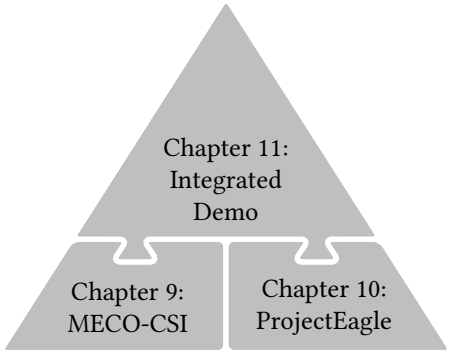


Figure 8.1 · Schematic overview of the interaction between the chapters of part II.

9

MECO-CSI and its applications

This chapter introduces the technical contributions related to the MECO-CSI GitHub project (Verbandt, 2018), which stands for MECO control setup interface. As the name suggests, this is a software toolchain designed to interface various setups that rely on embedded hardware with limited interfacing capabilities. Although the software was mainly designed with control applications in mind, it is also useful for the design of general mechatronic systems. The chapter is structured as follows. First, the aim of the project is explained and the need for this new toolchain is justified. Next, the toolchain's main features are highlighted, introducing the two main pieces of interacting software, MicroOS and QRoboticsCenter. By means of a simple example, the following section shows how MicroOS is used to structure the code of a mechatronic application. To conclude, several mechatronic systems that rely on this software toolchain are presented along with the role of MECO-CSI within the whole.

9.1 The aim of the project

Over the last decades, the cost of computational power has seriously dropped. This acts as a catalyst for low-cost applications that require only a limited amount of computational power. Examples hereof are multirotors, running a simple control algorithm or internet-of-things applications, which usually consist of various elementary modules that communicate with one another. Typically, these systems are equipped with one or more microcontrollers running dedicated software for a specific task. Often this software is referred to as embedded, though this term knows many interpretations. In this text, embedded refers to the limited capabilities when it comes to interaction with the developer. A microcontroller is interfaced via a variety of serial (bus) protocols such as UART, I2C or SPI, but cannot be hooked directly to a keyboard, mouse or screen. This lack of accessibility makes the development of embedded software even more cumbersome than software running on a pc. Also interactions with the system are supposed to happen via serial (bus) communication, which obviously does not feel natural nor practical to a human.

MECO-CSI tries to overcome these difficulties by facilitating the interaction between the developer and the embedded application. As depicted by Figure 9.1, a pc serves as an intermediate link between the embedded application and the user. By means of a serial port, e.g. a USB port or Bluetooth connection, variables and parameters are exchanged between the embedded application and the pc so that they essentially become each other's mirror image. Hence, the onboard variables and parameters are virtually accessible to the user.

This philosophy is embodied by two interacting pieces of software. MicroOS is a software library that runs on the embedded hardware. Written on top of the Arduino API, it not only supports the native Arduino boards, but for instance also boards from the Teensy (PJRC, 2018) and the Maple (LeafLabs, 2018) family. Its counterpart, QRoboticsCenter, is a graphical user interface that makes the variables and parameters accessible on a pc. It is based on the Qt framework (The QT Company, 2018) so that the application runs on various systems such as Windows, Unix and macOS.

The idea of such a toolchain is certainly not new: Mathworks and National Instruments both provide their own way of interfacing microcontrollers. However, relying on commercial software introduces extra costs, limits flexibility and usually brings some toolchain specific overhead, justifying the development of a custom toolchain.

9.2 MECO-CSI – an in-depth discussion

This section presents a detailed description of the MECO-CSI toolchain, focusing on the different features it provides. The onboard variables are discussed first as they constitute the main way of interaction between the platform and the user. Thereafter, it is shown how settings are stored in the persistent (nonvolatile) memory using parameters.

9.2.1 Incoming and outgoing variables

The interaction between the embedded platform and the user happens primarily via the incoming and outgoing variables. Incoming and outgoing reflect the direction of travel as seen from the embedded platform. The incoming variables serve as a driver for the embedded platform, e.g. a setpoint for a control algorithm supplied by the user. The outgoing variables can be employed to provide the user with information about the embedded platform, for instance the voltage applied to the motors or a measured velocity. Both sets of variables consist of eight floating point and four integer numbers. This means that a total of twelve signals can be transferred to and

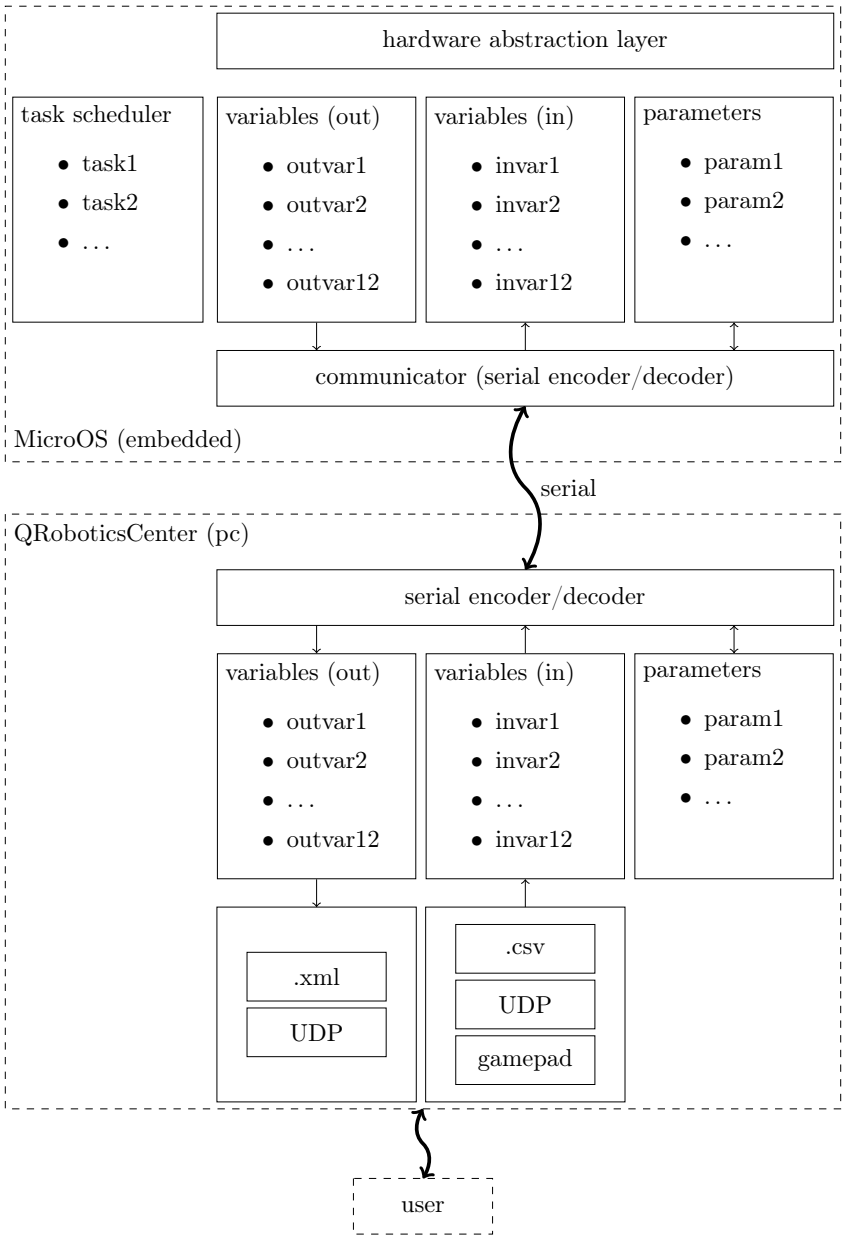


Figure 9.1 · General overview of the MECO-CSI toolchain. The software library MicroOS runs on the embedded application, providing the utilities to gain access to onboard variables and parameters. QRoboticsCenter runs on a pc and interacts with the embedded platform via a serial connection. By doing so, the onboard variables and parameters become available to the end user.

from the system simultaneously, which was found to be a good compromise between the amount of information per message and the transmission rate.

On the embedded platform, MicroOS keeps track of the incoming and outgoing variables via two dedicated memory blocks. The incoming variables are decoded and stored in the reserved memory block so that they can be read via the functions `getGPinFloat()` and `getGPinInt()`. The outgoing variables are set using the functions `setGPoutFloat()` and `setGPoutInt()`. By default, the set of outgoing variables is encoded and transmitted at a rate of 100 Hz.

On the side of the user, QRoboticsCenter takes care of a similar task. The incoming and outgoing variables are again respectively decoded and encoded and stored in reserved memory blocks. Because the memory blocks within QRoboticsCenter reflect the corresponding blocks of the MicroOS library, the user gains virtual access to the variables on the embedded platform. To facilitate the interaction, QRoboticsCenter provides a graphical user interface, depicted in Figure 9.2. The incoming variables, selected in panel (1) are displayed on a running graph. The outgoing variables on the other hand can be set via interface (2). QRoboticsCenter also supports more advanced ways of interaction with the embedded system. It is possible to log the incoming data in an .xml file (see indicator (5)) or pass the data on to a UDP port. The latter allows the user to have another program, e.g. a Matlab or python script, process the data stream by simply reading from this UDP port. Similarly, an outgoing stream of variables can be generated by a .csv file or a UDP port that is linked to another program (see panel (4)).

9.2.2 Persistent parameters

Besides the traditional volatile memory (RAM), most microcontrollers also provide a block of persistent memory (ROM). The persistent memory preserves data even if the power is switched off and becomes useful to store platform-specific settings. To make a clear distinction between volatile variables and variables stored in the persistent memory, the latter are referred to as parameters. A typical example of a parameter would be the identifier of a platform or the gain of a controller.

MicroOS manages the microcontroller's persistent memory by means of storage objects. Two such objects are provided: `floatStorage` for floating point numbers and `intStorage` for integers. A new parameter is declared via the function `add(name)` where `name` is an identifier string. The `get(name)` function in turn loads the variable from the corresponding storage object so that its value can be read or altered.

QRoboticsCenter provides a specific panel (see Figure 9.2 panel (3)) to access the embedded system's parameters. Because the persistent memory is intended to be quasi-static, the user has full control over the exchange of parameter values. This is

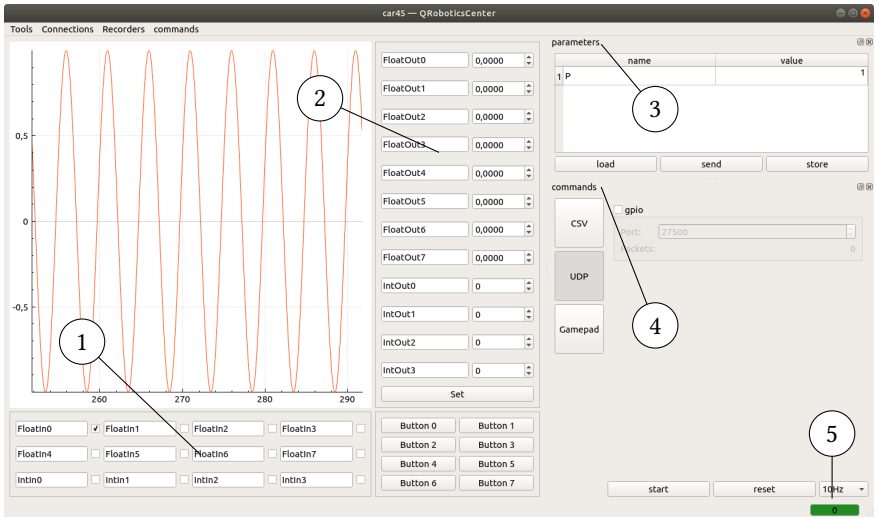


Figure 9.2 · Screenshot of the QRoboticsCenter user interface. The incoming variables are accessed graphically via control panel (1), the outgoing variables set via control panel (2). Panel (3) allows the user to change the onboard parameters. More advanced usage such as the use of UDP ports is supported via panel (4). Indicator (5) signals whether the incoming variables are being logged.

done via the buttons load, send and store. Once the parameters are loaded, the user is able to alter the values of the onboard parameters. The button send updates a local copy of the parameters on the embedded platform. The new values are not written to the persistent memory immediately so that the user is able to experiment with new settings while the old settings are kept in the persistent memory. When satisfied, the user can write the parameters to the persistent memory by clicking the store button.

9.3 MicroOS – a backbone for embedded systems

The MicroOS library provides a sensible software structure for a typical mechatronic system, which is a second way of facilitating the development of embedded applications. To this end, two additional classes are introduced: the hardware abstraction layer (HAL) and the communicator, also visible on overview Figure 9.1. The first paragraphs explain the purpose of each class and how they are extended to satisfy the needs of a specific application. Next the optional robot object is introduced

as well as MicroOS's task scheduler. Throughout this section, code example 9.1 is used to make the discussion more tangible.

```
1  //static RobotHALv1 hal;
2  static RobotHALv2 hal;
3  static Robot robot(&hal);
4  static RobotCommunicator comm(&robot);
5
6  int update(void) {
7    robot.controllerHook();
8    return 0;
9  }
10
11 void setup() {
12   System.setHAL(&hal);
13   System.setCommunicator(&comm);
14   System.addThread(NORMAL, 5000, &update, false);
15   System.start();
16 }
17
18 void loop() {
19   System.run();
20 }
```

Code example 9.1 · Basic code example to explain the role of the HAL, communicator, robot and task scheduler.

9.3.1 The hardware abstraction layer

The hardware abstraction layer (HAL) is used to decouple the software from the specific hardware. This is especially useful when there are two or more hardware configurations that are physically different but functionally the same. A typical example would be a series of PCBs, e.g. versions A and B, where the pinout has changed. By means of inheritance, specific HAL classes that incorporate these changes are implemented, e.g. HAL_A and HAL_B. Switching hardware platforms then only amounts to switching the HAL to the appropriate instance without changing the remainder of the software. This is illustrated by lines 1 and 2 in code example 9.1. Because the original hardware, interfaced via `RobotHALv1`, has changed, the first line is commented and a new HAL, `RobotHALv2`, is introduced. The remainder of the code remains unchanged.

9.3.2 The communicator

The communicator is responsible for decoding and encoding incoming and outgoing messages on the serial communication line. To this end, the MAVLink protocol (Meier, 2018) is used. This is a well established lightweight protocol which is used frequently in the context of embedded systems. The basic implementation of the communicator relies on a set of messages that is only related to the operating system itself. This involves messages to send a heartbeat, to share variables and parameters or to communicate events. In case a set of application-specific messages is required, the original message set is readily extended using the MAVLink tools. To incorporate these new messages into the communicator, again the mechanism of inheritance is proposed, because it allows the developer to handle the new message set, without changing the behavior in response to the basic messages. Lines 4 and 13 of code example 9.1 implement this idea. First, a `RobotCommunicator` object is created, which is an extension of the standard `Communicator`. This object is then passed on to `System` to replace the standard `communicator` object.

9.3.3 The robot

Although the robot object is not strictly necessary to create a running example, it is highly recommended. The robot object gathers all information and functionality that is characteristic to the application itself. This keeps the application specific code separated from the MicroOS library. Take for instance a control-oriented setup. The robot class would then typically implement the control law itself as well as reading the sensors and setting the actuator setpoints. Lines 3 and 6-9 of code example 9.1 reflect this line of thinking. First, a `Robot` object is created. The HAL is supplied as an argument because the robot typically gathers the sensor's measurements and drives the actuators which are interfaced via the HAL. Lines 6 to 9 wrap the robot's main functionality, in this case a control law, in a new function intended to be used later.

9.3.4 Task scheduling

As a by product, MicroOS comes with a task scheduler, allowing the embedded platform to run different tasks in parallel[(1)]. Contrary to other operating systems available for the Arduino framework, e.g. FreeRTOS (Ltd., Real Time Engineers, 2018) and ChibiOS (Di Sirio, 2018), real-time execution of the highest priority task is not guaranteed. Because real-time operation in the strict sense is seldom required and

[(1)]Single core microcontrollers strictly speaking never actually run tasks in parallel. However by dividing the processing power, several tasks (threads) seemingly run in parallel although they are executed one after the other.

not the main purpose of the MicroOS library, a simpler user-friendly task scheduler is opted for. The use of the task scheduler is visible on line 14. On this line, the function `update()`, which wraps the main functionality of the robot, is added to the list of tasks. It is executed with a `NORMAL` priority and a period of 5000 µs.

9.4 Applications

Several mechatronic systems have been designed using the toolchain described in this chapter. The ballbot, a ball balancing robot was the original driver for the development of this toolchain. Later the toolchain also proved itself useful for the mecotron project as well as the ourbot project. The following sections give an overview of these mechatronic systems and the role of MicroOS and QRoboticsCenter within them.

9.4.1 The ballbot

Ball-balancing robots or ballbots are a relatively new type of mechatronic system. The first design was presented by Lauwers et al. (2005). Four actuated rollers are in contact with a ball that acts as an omnidirectional wheel. Each pair of rollers causes a motion in one of two perpendicular directions. Rotation around the robot's central axis is however not possible with this setup. Another influential design was proposed by Kumagai and Ochiai (2008). They mounted three omniwheels (Blumrich, 1974) instead of two pairs of rollers. Although the omniwheels introduce coupling in the dynamics, they enable the robot to both rotate and translate. Although the dynamics are more complicated, the ability to rotate around its central axis makes a ballbot with omniwheels the design of choice for many (Fankhauser and Gwerder, 2010; Yang et al., 2015; Blonk, 2014).

The ballbot presented in this section falls into the category of omniwheeled designs and is shown in Figure 9.3. Three omniwheels are directly driven by three dc motors. The motors are voltage-controlled by means of a TB6612FNG motor driver and a PWM signal. The power comes from two 12 V lipo batteries. The robot is equipped with two types of sensors. The MPU6050 inertial measurement unit (IMU) is capable of directly estimating the attitude, i.e. the roll and pitch angles with respect to the gravitational vector. The motors are equipped with rotary encoders that make it possible to estimate the velocity in the local reference frame.

The software runs on a Teensy 3.1 microcontroller and is based on the MicroOS library. As explained in section 9.3.1, multiple instances of the basic HAL have been written to cope with the changing connections of the electronics over different versions. Also, an extension of the basic communicator was developed to be able to use ballbot-specific

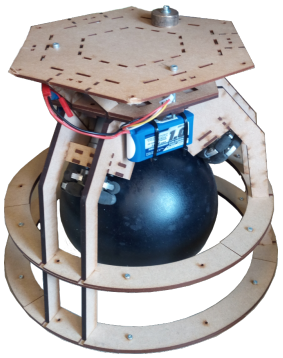


Figure 9.3 · The ballbot, an unstable holonomic platform to experimentally validate optimal feedback controller design strategies.

messages such as a message to share the attitude information. The communicator listens on both the native USB port as well as an additional Bluetooth module. The latter makes wireless communication possible. The ballbot class is introduced to group all application-specific functionality. This comprises the processing of the sensory data, the execution of the control algorithm and the generation of the PWM signals. Via the Bluetooth module, a connection is established with QRoboticsCenter, which allows the user to interact with the system.

9.4.2 The mecotron

The development of the mecotron was inspired by the need to make the basic control theory course at KU Leuven more hands-on (Steinhauser et al., 2017). Because of the mathematical and rather abstract nature of the course, students miss out on the practical value of control. To bridge the gap with reality, the students are challenged by a series of assignments involving the mecotron platform. The mecotron platform, depicted by Figure 9.4 consists of two motors in a differential drive configuration. Additionally, the platform is equipped with a series of sensors. Encoders on the wheels provide measurements of the wheel angles. Infrared sensors at the front and the side give information about the distance to the platform's surroundings.

At the core of the platform resides an Arduino Mega. In order not to burden the students with course-unrelated developments, a software structure is already provided. Again, the MECO-CSI toolchain proves itself useful. First of all, a dedicated robot class is provided, which allows the students to focus on the application itself while the rest of the software remains hidden. Also, tedious debugging procedures via textual output are prevented by the availability of the output variables and the corresponding

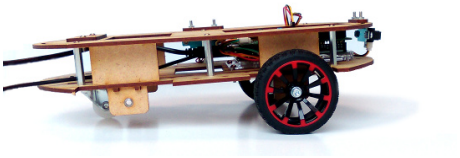


Figure 9.4 · The mecotron, a low-cost mobile platform to get hands-on experience with control.

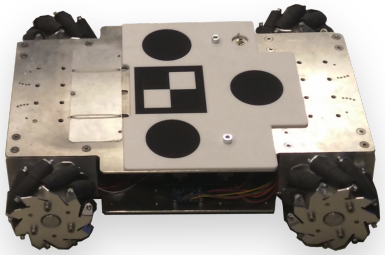


Figure 9.5 · The ourbot, a holonomic mobile platform to test motion control algorithms.

graphical representation. Moreover, students can easily interact with the platform via QRoboticsCenter to for instance set a reference value or load a reference trajectory. Buttons in QRoboticsCenter are linked to corresponding callback functions in the robot class, which come in handy to for instance suspend the control law.

9.4.3 The ourbot

The ourbot, shown in Figure 9.5, was developed in view of the experimental validation of new B-spline based motion planning algorithms (Van Parys and Pipeleers, 2017). This resulted in a holonomic platform with an open interface capable of running various types of control algorithms. The platform is driven by four mecanum wheels (Ilon, 1975), each of them mounted on a dc motor with a rotary encoder. The motors are voltage-controlled by a PWM signal.

Two computational units provide the necessary processing power to run the necessary control algorithms. At the lowest level, a Teensy 3.1 microcontroller takes care of the ourbot's local velocity. Similarly to the ballbot from section 9.4.1, an application-

specific message set and communicator are created to for instance share local encoder measurements or to provide a velocity setpoint in a cartesian space. A kinematic decoupling is used to transform the robot's wheel speeds to the local cartesian space. Three PID controllers compute virtual motor voltages, based on the velocity error in cartesian space. The virtual motor voltages are then transformed to four actual wheel voltages and the corresponding PWM signal. At the higher level, an Odroid XU4 single board computer is responsible for the motion planning and related tasks. It fuses external global position information with the encoder information by means of a Kalman filter, computes a time-optimal trajectory in a receding horizon fashion and executes three PID controllers responsible for tracking the computed trajectory.

9.5 Conclusion

This chapter presented a new software toolchain to ease the development of embedded applications. The MicroOS software library provides a level of abstraction for the basic features of a typical embedded platform such as the serial communication, accessing the nonvolatile memory or the scheduling of tasks. Also the library suggests a structure which applies to a typical embedded application, helping the developer to maintain a clean code-base. Accompanying MicroOS is the graphical user interface, QRoboticsCenter. QRoboticsCenter serves two purposes. First, it provides access to the internal variables of the embedded application, speeding up the development of new applications. Second, it allows sharing the internal variables with other applications such as Matlab or python scripts. Several embedded applications have been successfully developed by means of this new software toolchain. Amongst them are the ballbot, the mecotron and the ourbot.

10

ProjectEagle

This chapter is devoted to ProjectEagle, an open-source GitHub project (Van Parys and Verbandt, 2018) that provides a scalable solution to the problem of absolute indoor positioning. Although a variety of indoor positioning systems has been developed, few of them address the problem of scalability, i.e. the ability to cope with a changing number of objects to be localized in an area changing in size. The proposed system consists of several smart camera modules that are capable of localizing a specific type of marker and broadcast its position on the network. Because the modules are headless[(1)] and typically mounted on the ceiling, an important aspect of the project is its user-friendly interface that facilitates adding or removing a camera module. The remainder of this chapter is structured as follows. First, the development of the software package is motivated. Next, the system is introduced with a focus on the actual hardware. The technical contributions related to the calibration and the deployment of the system are covered in the last part. This chapter is to a large extent based on Van Parys et al. (2018).

10.1 Motivation

As a part of the Industry 4.0 paradigm, there exists a tendency towards flexible manufacturing systems. Contrary to rigid systems, they allocate the available resources, e.g. products or work cells in an optimal and thus demand-varying fashion. In order to do so, one of the requirements is to keep track of all resources' whereabouts. This observation makes localization and tracking of the different resources a vital aspect of a flexible manufacturing system, since the increased degree of flexibility results in a higher level of uncertainty about the resources' positioning with respect to the work floor. Various approaches to tackle the problem of absolute indoor localization have been proposed. Currently LIDAR-based systems have found wide acceptance in industry. Cheap reflectors spread across the factory provide landmarks from which the LIDAR's position is determined (Beinschob and Reinke, 2015). Wireless

(1) The term headless reflects the absence of human interface devices such as a mouse, a keyboard and a monitor so that the computer cannot be accessed directly. Typically, headless platforms are accessed via a (wireless) network.

technology is emerging as an interesting alternative because wireless beacons are becoming increasingly cheap and are reaching centimeter-level accuracies (Stojanović and Stojanović, 2014). LIDAR and wireless-based systems however share a lack of scalability, i.e. the amount of sensors depends on the number of resources, typically rendering the overall system expensive after all. Moreover, these technologies require the resources to carry a sensor, which is not always feasible.

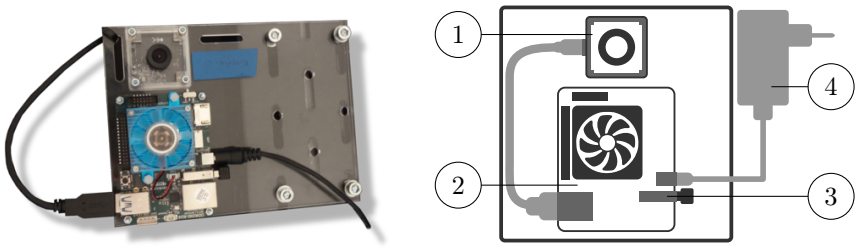
More widespread throughout academia are vision-based systems. Their high information density makes them suited for a variety of tasks such as obstacle detection, product identification or object tracking. With fixed cameras covering the entire workspace, the required number of sensors is decoupled from the number of resources, creating a scalable solution. Examples of such systems are widely available within the robotics literature, especially in a robotic soccer context (Brezak et al., 2008; Velasco et al., 2012; Roque and Doering, 2005). These provide basic insight on how to achieve accurate localization and identify several resources within a single image. Because of their scalability and noninvasive nature, this work proposes the use of several smart camera modules monitoring the work floor. Two challenges are involved in setting up such a system. First, the modules are headless and typically mounted in inaccessible places, e.g. a ceiling, making remote accessibility a must-have. Second, attention should go to the global consistency of the measurements since they are ultimately coming from different devices.

10.2 Overview

The system consists of a series of smart camera modules, connected via a wireless network. A single module is illustrated in Figure 10.1 and consists of three main parts. An Odroid oCam 5MP, equipped with a wide angle lens, provides a video stream of the work floor. This video stream is transmitted via USB to an Odroid XU4 single-board computer, capable of extracting position information at a rate of 10 Hz. By means of a WiFi dongle, the Odroid XU4 is able to broadcast the measurements on to the network so that other devices within the network are able to pick these up. Several of these smart cameras are spread across the work floor to cover the entire space.

10.3 Calibration

This section details on the calibration of the multi-camera system. The first section derives the elementary mathematical relations that are used to calibrate the system. Next, the practical aspect of the calibration is highlighted, i.e. the use of an



(a) (b)

Figure 10.1 · Camera module. A picture of the module is illustrated in (a), while (b) gives an overview of its components: (1) the Odroid oCam 5MP, (2) the Odroid XU4, (3) a WiFi dongle and (4) an AC adapter.

accompanying graphical user interface that provides swift access to the headless modules and runs the semi-automatic calibration procedure remotely.

10.3.1 Theoretical aspects

As with all camera-based localization systems, a good calibration procedure is vital to the accuracy of the position measurements. The camera calibration is usually split in two parts: the intrinsic and extrinsic calibration. The intrinsic calibration relates a 2-dimensional image coordinate to a 3-dimensional coordinate in the camera frame. The extrinsic calibration is concerned with finding the transformation between the camera frame and the world frame. The latter is especially important in a multi-camera system as it is required to achieve mutually consistent measurements. The first paragraph elaborates on the intrinsic calibration. Next it is shown how the world frame is determined. The last paragraph describes how to enforce the same world frame on the other camera modules.

Intrinsic calibration

The intrinsic calibration (Zhang, 2000) is concerned with finding the parameter values of the selected camera model, in this case the standard pinhole camera model:

$$\lambda\, {}_p p = K\, {}_c p, \tag{10.1}$$

with ${}_p p = [u,\, v,\, 1]^T$. Given a scaling factor λ, this model provides the mapping of pixel coordinates $[u,\, v]^T$ to coordinates ${}_c p \in \mathbb{R}^3$, expressed in a frame $\{c\}$ attached to the camera. The pinhole model is characterized by the camera matrix K, which

is determined from a set of training data, usually a series of chessboard pictures. To reduce the effect of lens distortions, $[u, v]^T$ is obtained by first applying a nonlinear transformation to the distorted pixel coordinates. The nonlinear distortion model is determined together with K.

Fixing the world frame

The extrinsic calibration searches for the transformation that converts camera coordinates to world coordinates. Since the location of the world frame is arbitrary, it is fixed through the calibration of the first camera module. Figure 10.2 illustrates how the world frame $\{w\}$ is chosen. The construction of the world frame is based on the location of the ground plane γ:

$$\gamma = \left\{ {}_c p \in \mathbb{R}^3 \mid {}_c p^T \, g = h \right\}, \tag{10.2}$$

where $g = [g_x, g_y, g_z]^T$. The vector g is easily derived as the least-squares solution of the system

$$\begin{bmatrix} {}_c p_1 & -1 \\ {}_c p_2 & -1 \\ \vdots & \vdots \\ {}_c p_n & -1 \end{bmatrix} \begin{bmatrix} g \\ h \end{bmatrix} = 0, \tag{10.3}$$

where the points ${}_c p_i$ are located on the ground. The coordinates of such points are typically available from the intrinsic calibration, so that γ can be assumed to be known. The origin o' of $\{w\}$ lies on the intersection of the ground plane γ and the z-axis of $\{c\}$. The x'-axis of $\{w\}$ follows the intersection of γ with the plane spanned by the x- and z-axis of $\{c\}$. The z'-axis is chosen to be perpendicular to γ, pointing upwards and the y'-axis is determined such that $\{w\}$ is a right-handed frame. Given the equation of the ground plane, the origin ${}_c o$ and unit vectors ${}_c e_x$, ${}_c e_z$ are mapped to ${}_c o'$, ${}_c e_{x'}$, ${}_c e_{z'}$ respectively and their coordinates expressed in {c} are

$${}_c o' = \left[0, \, 0, \, \frac{h}{g_z} \right]^T, \tag{10.4a}$$

$${}_c e_{x'} = \left[\frac{|g_z|}{\sqrt{g_x^2 + g_z^2}}, \, 0, \, -\frac{g_x}{\sqrt{g_x^2 + g_z^2}} \operatorname{sign}(g_z) + \frac{h}{g_z} \right]^T, \tag{10.4b}$$

$${}_c e_{z'} = {}_c o' - \frac{g}{\|g\|_2} \operatorname{sign}(g_z). \tag{10.4c}$$

Using the mapping of these three points, the transformation matrix ${}^c_w T$ from $\{w\}$ to $\{c\}$ is easily determined (Sorkine and Rabinovich, 2016).

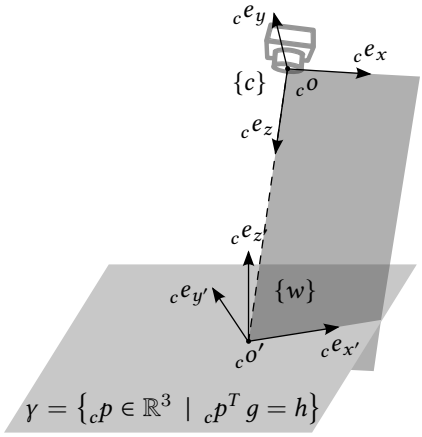


Figure 10.2 · Choice of the world coordinate frame based on the coordinate frame of the first camera.

Extrinsic calibration

The extrinsic calibration of a camera n consists of finding the transformation ${}^{c_n}_{w}T$, i.e. the transformation matrix that expresses a coordinate in a camera frame $\{c_n\}$ to a coordinate in the world frame $\{w\}$:

$$ {}_w p = {}^{c_n}_{w}T \, {}_{c_n} p. \tag{10.5} $$

Now suppose camera module n is added to a fully calibrated system, i.e. all ${}^{c_i}_{w}T$ are known for $i \in \{0, \ldots, n-1\}$. In this case, the extrinsic calibration of the additional camera module can be done via any of the other modules within the system because:

$$ {}^{c_n}_{w}T = {}^{c_i}_{w}T \, {}^{c_n}_{c_i}T. \tag{10.6} $$

To exploit this property, a small overlap between the field of view of the new camera and one camera that has already been extrinsically calibrated, is introduced. A series of chessboard pictures taken from the two cameras now provides two point clouds: one in the world coordinate frame and one in the coordinate frame of the new camera. These point clouds are identical up to a rigid transformation, which is exactly the transformation from the world to the new camera. Once more, the technique of Sorkine and Rabinovich (2016) is employed to compute this transformation. Important to note is that every camera stores its own transformation from the world to its own coordinate system. By doing so, every camera module can function independently while providing consistent measurements.

Figure 10.3 · GUI supporting the calibration of the ProjectEagle camera modules.

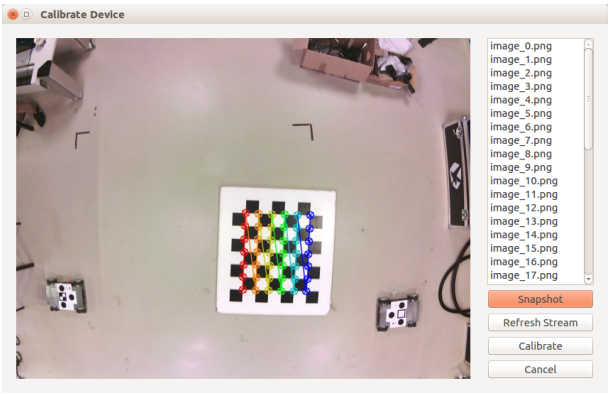


Figure 10.4 · Pop-up window with a live camera feed and buttons to take a snapshot and compute the intrinisic calibration.

10.3.2 Practical aspects

The main drawback of using headless modules mounted at inaccessible locations, e.g. a ceiling, poses a thread to the practical applicability of the proposed system. Although remote access such as an ssh connection provides full control of the module, the absence of a graphical interface is tedious, especially in the context of image processing. To overcome this burden, a graphical user interface (GUI) was developed. The GUI initially displays the list of modules that were added to the system earlier (Figure 10.3). This window allows the user to add a new module to the list by specifying its ip address and credentials. Once listed, the user is able to select the module and press calibrate. This results in a new window with a live feed of the camera image, shown in Figure 10.4. When the chessboard pattern is detected, its corners are highlighted to give the user feedback about the usability of the picture. The snapshot button enables the user to gather the necessary pictures to estimate the intrinsic

calibration. Ideally, this set consists of a mixture of images with the chessboard flat on the ground and images where the chessboard is not parallel to the ground. The intrinsic calibration requires this mix of images to reliably estimate the focal length (Zhang, 2000). The images with the chessboard flat on the ground are used to estimate the ground plane γ, which is done automatically when calibrating the module. A RANSAC algorithm (Fischler and Bolles, 1981) is used to robustly select the images with a chessboard on the ground, which avoids the extra human intervention to classify the images. The extrinsic calibration happens in a similar fashion to the intrinsic calibration. This time, two modules are selected from the list: one with an external calibration available, the other without. A window with two video feeds pops up and pairs of snapshots are taken while the chessboard is moved within the zone of overlap between the two fields of view. Via the calibrate button, the extrinsic calibration of the module is determined and the new module is set up.

10.4 Deployment

This section explains how the system determines the global position of the resources on the work floor. The localization is based on a dedicated pattern which is introduced first, together with the corresponding extraction algorithm. The second section spends a few words on how the measurements are made available to other systems.

10.4.1 Pattern recognition

To obtain a reliable localization at a sufficient rate, a distinct marker is installed on each resource for which position information is desired. The marker, depicted in Figure 10.5, consists of three disks in a isosceles triangle configuration, since the overall positioning accuracy of a series of small disks is found to outperform one bigger disk (Bruce and Veloso, 2003). Along its centerline, a grid of black and white patches is added. This grid forms a 4-bit code that uniquely identifies each marker. Using only black and white colors renders the system more robust to illumination changes without intricate correction schemes as proposed by Klančar et al. (2004).

An OpenCV-based (Bradski, 2000) software implementation to localize this type of marker is available from Van Parys and Verbandt (2018). The algorithm starts with a background subtraction procedure followed by thresholding the resulting image. An erosion-dilation operation takes care of noise suppression. This yields a series of regions corresponding to unidentified objects traversing the workspace, making the search for markers more efficient. A blob detection algorithm is used to detect the distinct black disks of the marker. All possible combinations of blobs are tested against the properties of the isosceles triangle, rejecting false-positive blobs and returning

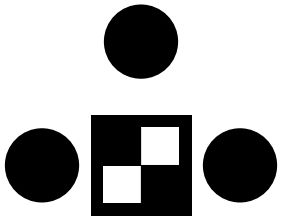


Figure 10.5 · Example marker for the localization system.

the marker's position. Since the disks are not arranged in an equilateral triangle configuration, it is possible to determine which blobs correspond to the left, right and top disks. This allows blindly sampling the identification patches to determine the marker's ID.

To transform the blob's pixel coordinates to world coordinates, depth information is required. It is assumed that each marker moves in a specific plane, $\{{}_{w}p \in \mathbb{R}^3 \mid k^T \, {}_{w}p = l\}$, which is typically parallel to the ground plane. Given the pixel coordinates ${}_{p}p_b = [u_b, v_b, 1]^T$ of a blob, its corresponding coordinates ${}_{w}p_b$ in $\{w\}$ are determined by solving the system of equations (10.7) for λ and ${}_{w}p_b$.

$$\begin{cases} 0 = K \, {}^{c}_{w}T \, {}_{w}p_b - \lambda \, {}_{p}p_b \\ l = k^T \, {}_{w}p \end{cases} \tag{10.7}$$

From the global coordinates of the three blobs, the marker's position and orientation are derived.

10.4.2 Measurement broadcasting

The detection algorithm described in 10.4.1 is executed continuously while the system is running. To make the resulting measurements available to others, ProjectEagle relies on Zyre (Hintjens and Boccassi, 2018). This project enables the individual modules to form a group of agents, e.g. eagle. Each individual module is able to broadcast its measurements within this group. Similarly to the camera modules, other systems are able to register as a member of this group, which allows them to receive these measurements.

10.5 Quantitative and qualitative assessment

The calibration procedure was carried out on the two camera modules located at KU Leuven's Robotics Lab. The two modules are first calibrated intrinsically and yield an

rms fitting error of 0.127 px and 0.134 px. This means that pinhole camera model is able to accurately relate the 2-dimensional image coordinates to the 3-dimensional camera coordinate system. Together with the intrinsic calibration, the ground plane is estimated using the same set of images. The RANSAC algorithm is able to correctly classify 40 of the 41 images having the chessboard lying on the ground. The entire image set contains 57 images. The RANSAC algorithm is indeed able to robustly identify the images corresponding to the ground plane. This leads to a transformation matrix with nearly zero Euler angles and a translation vector $[0.030,\ 0.042,\ 3.047]^T$ m. Although a quantitative assessment of these results is not possible due to the lack of a ground truth, the calibration is line with the expectations because the camera is mounted nearly parallel with the ground at a height of about 3.05 m. The same conclusion holds for the extrinsic calibration of the second module: the nearly zero Euler angles and the translation vector $[-0.234,\ 1.162,\ 3.059]^T$ m correspond to the manually measured distances. Based on the overlap between the fields of view, the rms calibration error is estimated to be 0.009 m. An accuracy of 1 cm is thus achieved, but this accuracy drops with every extra link introduced in the system.

The absence of a ground truth prevents a quantitative analysis of the ultimate positioning accuracy. However, Van Parys et al. (2018) proved that the accuracy of the proposed system suffices to obtain reliable motion planning. This work fused the global measurements with local encoder measurements. The global measurements serve as a long term correction on the encoder measurements that provide accurate position information on a short term.

10.6 Conclusion

This chapter presented ProjectEagle, an open-source project that provides the software to calibrate and deploy a multi-camera localization system. The system consists of several smart camera modules that are capable of localizing a specific marker within an image. By providing overlap between the different fields of view, the modules are calibrated with respect to the same reference, which leads to consistent measurements. A graphical user interface facilitates the interaction with the inaccessible headless modules: it establishes the network connection, provides a live feed of the camera and runs a semi-automatic calibration routine. By means of the Zyre project, the modules are capable of broadcasting the measurements to their peers.

11

An integrated demo

This chapter describes the development of a demonstration that integrates the work and systems presented in chapters 9 and 10. The main goal is to demonstrate how these systems are readily brought together to come to a functional whole. More specifically, this chapter presents a system in which a mobile robot executes a controlled point-to-point motion. The mobile robot is chosen to be the ball-balancing robot of section 9.4.1, which is readily interfaced thanks to the MECO-CSI toolchain, described in section 9.2. The necessary position measurements are provided by ProjectEagle, which is the subject of chapter 10. The remainder of this chapter is structured as follows. The first section describes the physical composition of the system, drawing attention to the different components involved. The next section focuses on the developments needed to integrate the components that are already available. An experimental validation of the system is presented in the last section as proof of a successful integration.

11.1 Overview of the integrated system

The system considered in this chapter is depicted schematically in Figure 11.1. At the center of the application sits the ball-balancing robot (1) presented in section 9.4.1, taking on the role of mobile robot. The robot is velocity-controlled and receives its setpoints from the central coordinator (2) via QRoboticsCenter, which is introduced in section 9.2. The latter thus acts as an intermediary to transfer velocity commands. The position of the robot is being tracked using ProjectEagle (3), which was the subject of chapter 10. By capturing these position measurements, the central coordinator is able to run a control algorithm that enables the mobile robot to track a desired trajectory.

11.2 The central coordinator

The sole ingredient that is needed to complete the desired system and has not been discussed earlier, is the central coordinator. A schematic of its inner workings is provided in Figure 11.2. The central coordinator's main responsibility is to run a

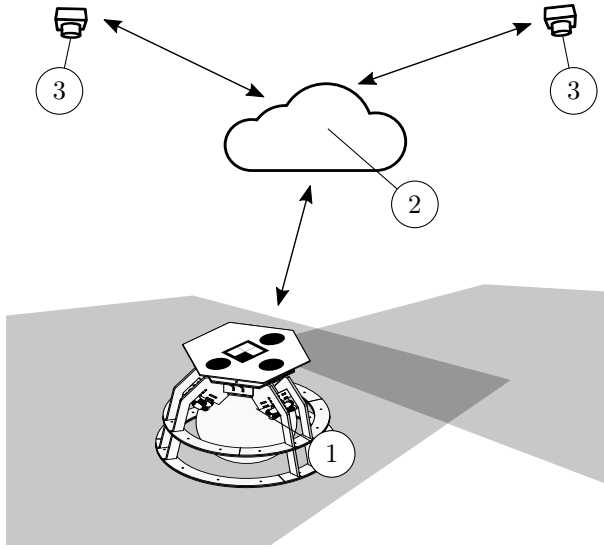


Figure 11.1 · Schematic representation of the integrated demo. The mobile robot (1) receives velocity commands from the central coordinator (2). The latter runs a control algorithm based on the position measurements coming from the positioning system (3).

control algorithm that positions the mobile robot (the yaw angle is not controlled). To this end, a Kalman estimator and a controller are running continuously at a rate of 10 Hz. The resulting velocity commands are passed to the qrc transmitter which in turn passes the setpoints on to QRoboticsCenter. Raw pose measurements coming from ProjectEagle are captured via a component referred to as the eagle receiver. Since these measurements are provided in an asynchronous manner, the eagle receiver runs in a separate thread indicated by the rounded rectangle. Position targets are provided by the user via the user interface which also runs asynchronously in a separate thread. The next paragraphs give some more details about the different components within the central coordinator.

11.2.1 Eagle receiver

The eagle receiver interfaces ProjectEagle to provide the central coordinator with pose measurements $\vec{p}_m$. To this end, the ProjectEagle protocol was implemented on

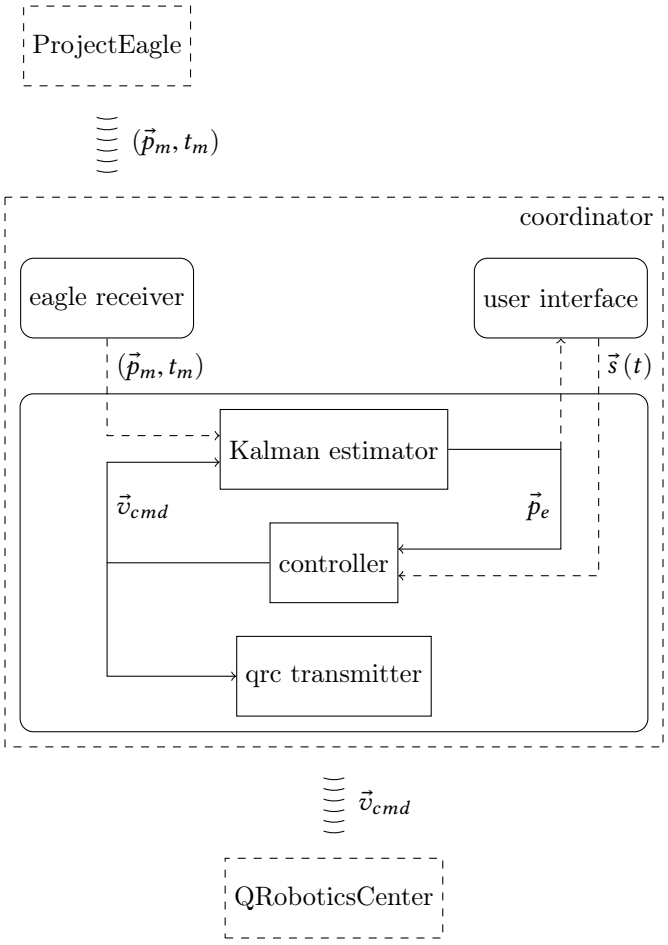


Figure 11.2 · Schematic overview of the central coordinator. The central control loop consists of a Kalman estimator and a controller. Peripherals are needed to transfer measurements and setpoints through the system. The interaction with ProjectEagle and QRoboticsCenter is indicated as well.

top of a python implementation of the zyre project (Loonstra and Kierkels, 2018). Important to note is that pose measurements do not come at regular intervals and are delayed due to the image processing. Therefore, the measurement is accompanied by the time t_m at which the image was captured.

11.2.2 User interface

A simple graphical user interface is created in order to enter a position setpoint. A simple mouse click on the map sets a target position and triggers the computation of the corresponding time-dependent trajectory $\vec{s}(t)$. Since trajectory generation is not considered within the scope of this demonstration, a smooth straight path that is constrained by a maximum velocity and acceleration is generated.

11.2.3 Kalman filter

To cope with the asynchronous and delayed nature of the position measurements, a Kalman estimator (De Schutter et al., 1999) is introduced in the feedback loop. The mobile robot's position is modeled as a combination of a low-pass filtered local velocity command and a standard kinematic model (11.5). The low-pass filter mimics the nature of the underlying velocity control loop and therefore has a cut-off frequency f_c of 0.4 Hz.

$$\begin{cases} x_1^{k+1} = (1-\alpha)\, x_1^k + \alpha \left(u_1^k \cos\left(x_6^k\right) - u_2^k \sin\left(x_6^k\right) \right) \\ x_2^{k+1} = (1-\alpha)\, x_2^k + \alpha \left(u_1^k \sin\left(x_6^k\right) + u_2^k \cos\left(x_6^k\right) \right) \\ x_3^{k+1} = (1-\alpha)\, x_3^k + \alpha u_3^k \\ x_4^{k+1} = x_4^k + x_1^k \Delta t \\ x_5^{k+1} = x_5^k + x_2^k \Delta t \\ x_6^{k+1} = x_6^k + x_3^k \Delta t \end{cases} \tag{11.1}$$

$$\alpha = \frac{2\pi \Delta t f_c}{2\pi \Delta t f_c + 1} \tag{11.2}$$

States x_4 and x_5 correspond to the x- and y-component of the position vector and x_6 models the robot's yaw angle so that the robot's pose vector is given by

$$\vec{p}_e = \begin{bmatrix} x_4 & x_5 & x_6 \end{bmatrix}^T. \tag{11.3}$$

Contrary to a standard implementation, the estimator stores the series of model inputs $\vec{u} = \vec{v}_{cmd}$ and their respective timestamps. When a new, delayed measurement

$(\vec{p}_m, t_m)$ becomes available, the estimator combines the series of previous inputs and the new measurement to come to an optimal estimate of the state $\vec{x}_e$ at time t_m. This ensures that at any time, the most recent measurement is incorporated in the state estimate. At any future time instant, the robot model is employed to optimally predict the state $\vec{x}_e$ and position $\vec{p}_e$.

11.2.4 Controller

To control the position of the mobile robot, a simple feedback controller has been installed. Based on the evaluation $\vec{s}\,(t_k)$ of the reference trajectory and an estimate of the pose $\vec{p}_e$ at time instant t_k, two decoupled hand-tuned PID control laws determine the x and y components of the velocity command $\vec{v}^{\,g}_{fb}$ in the global coordinate frame. Because the time-derivative of the reference trajectory is readily available, it is supplied as a feedforward $\vec{v}^{\,g}_{ff}$. The total commanded velocity in the global coordinate frame $\vec{v}^{\,g}_{cmd}$ thus becomes

$$\vec{v}^{\,g}_{cmd} = \vec{v}^{\,g}_{fb} + \vec{v}^{\,g}_{ff}. \tag{11.4}$$

Note that the yaw angle is not controlled so that the yaw velocity command is kept zero. Because the robot expects velocity commands to be expressed in the robot coordinate frame, which is fixed to the robot's body, the velocity command is transformed to the robot coordinate frame using the estimate of the yaw angle $x_{e,6}$:

$$\vec{v}_{cmd} = \begin{bmatrix} \cos\left(x_{e,6}\right) & -\sin\left(x_{e,6}\right) & 0 \\ \sin\left(x_{e,6}\right) & \cos\left(x_{e,6}\right) & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{v}^{\,g}_{cmd}. \tag{11.5}$$

11.2.5 Qrc transmitter

The qrc transmitter provides an interface to QRoboticsCenter, so that the velocity commands $\vec{v}_{cmd}$ generated by the controller are ultimately executed by the mobile robot. To this end, the advanced message transfer of QRoboticsCenter, discussed in section 9.2.1, is used. Via a UDP port, the QRC transmitter pipes the velocity commands to QRoboticsCenter which in turn transmits the velocity commands to the mobile robot.

11.3 Experimental validation

To experimentally validate the system, the mobile robot is instructed to subsequently move from one target position to the next. Starting from the initial position $\vec{p}_1$, two

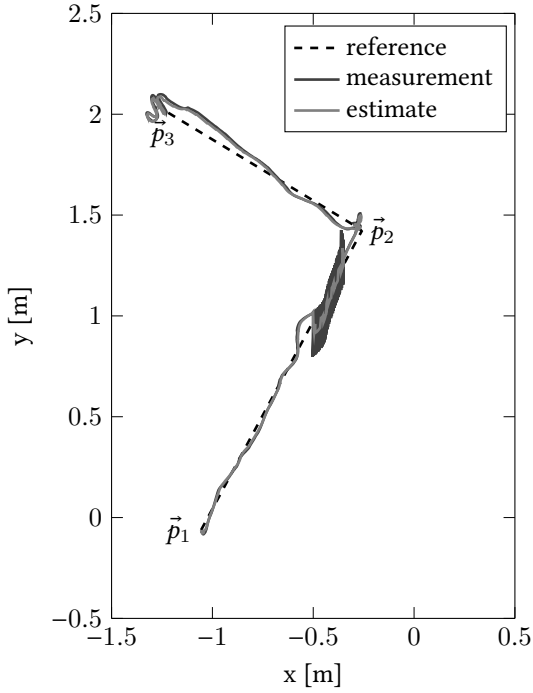


Figure 11.3 · Position of the mobile robot in the xy-plane during the experiment. The two dashed line segments indicate the subsequent reference trajectories. The dark gray line represents the raw position measurements whereas the light gray line shows the position provided by the Kalman estimator.

subsequent targets $\vec{p}_2$ and $\vec{p}_3$ were given as indicated in Figure 11.3. This figure suggests that the path is tracked fairly well from a spacial point of view. Interesting to note is the oscillation in the y-direction of the pose measurement, which is happening for $y \in [0.8, 1.5]$. This occurs because the marker that is attached to the robot and is detected by ProjectEagle, does not reside within the programmed marker plane. ProjectEagle believes a marker to be 0.07 m above the ground whereas the actual location of the marker is about 0.4 m above the ground. As a consequence, an error is introduced when transforming the marker's image coordinates to the global coordinate system. This error is especially visible when the robot is within the field of view of both camera's which then contradict each other resulting in an oscillatory measurement. Although it is possible to change the programmed marker plane, it was deliberately left unchanged to expose this weakness of ProjectEagle. On the other hand, decent filtering is able to resolve this issue as is shown by this experiment.

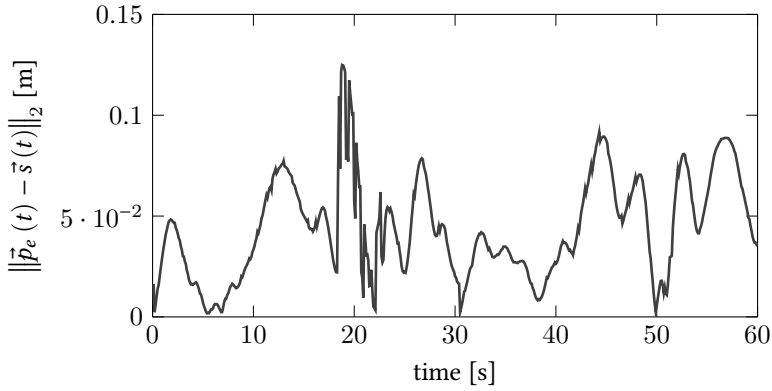


Figure 11.4 · Position error of the mobile robot with respect to the reference trajectory as a function of time.

To assess the the tracking accuracy as a function of time, the two-norm of the positioning error is shown in Figure 11.4. The error stays well below 0.15 m, which is considered sufficient given the fact that the system has a limited bandwidth (see chapter 6). The adequate tracking of the provided path also proves the successful integration of ProjectEagle and QRoboticsCenter as a robotic interface, which was the main intent of this experiment.

11.4 Conclusion

This chapter showed how the MECO-CSI project, discussed in chapter 9, and ProjectEagle, the subject of chapter 10, are seamlessly integrated in a motion planning demonstration. The missing link was found to be a central coordinator that glues these pieces together. Therefore, the development of this central coordinator was discussed in more detail. Attention was drawn to its two main tasks: interfacing the aforementioned systems and executing a control algorithm. The resulting system was experimentally validated and found to behave as expected.

12

Conclusion

The experimental validation of new methodologies often requires new technological developments. Two projects that confirm this statement are discussed in the previous chapters. The MECO-CSI toolchain was designed to facilitate the development of embedded software. ProjectEagle was born as a consequence of the MECO group's research in the area of motion planning, which requires accurate position measurements in an indoor environment. These systems are in second place brought together in an integrated demo. The remainder of this chapter summarizes the content of these chapters and gives a sketch of what the future might hold for these projects.

12.1 Summary

Chapter 9 presents the MECO-CSI toolchain. The main objective of this toolchain is to streamline the development of embedded software. This is accomplished by the interplay between MicroOS and QRoboticsCenter. MicroOS is a software library written on top of the Arduino API that allows the developer to make variables and parameters accessible to the outside world via a serial port such as USB or Bluetooth. The desktop application QRoboticsCenter is designed to receive this serial communication so that the developer ultimately has access to the embedded application's internals. Transferring information from the desktop to the embedded application is also supported, allowing swift interaction with the platform. QRoboticsCenter is also able to share incoming variables with and accept variables from other programs via a UDP port, which allows another program, possibly running on another pc within the network, to interact with the embedded platform. To conclude, the chapter presents a number of mechatronic systems that have been developed using the MECO-CSI toolchain.

Chapter 10 is dedicated to the development of ProjectEagle, a distributed localization system capable of detecting a specific pattern (marker) in an image and extracting its position in a global reference frame. The system consists of several smart camera modules that are typically mounted on the ceiling. Depending on the size of the floor, multiple cameras are used to cover the entire area. To obtain global

position information, the camera modules need to be calibrated with respect to one another. To streamline this process, ProjectEagle provides a semi-automatic calibration routine, packed in a user-friendly graphical user interface. At run-time, each module continuously runs image processing software that extracts a specific marker from the camera's video stream. Given the calibration of the system, the marker's position within the image is transformed to the global reference. These measurements are then broadcast on the network so that they can be used by other devices. Because the calibration of each camera is stored on the module itself, the system is truly distributed and keeps on functioning even if one or more modules fail.

The final piece of this thesis is introduced in chapter 11. It demonstrates the smooth integration of the mechatronic systems described in chapters 9 and 10 by means of an integrated demo. The purpose of the demo is to control the position of a ball-balancing robot and execute point-to-point motions. It is shown that the only missing component was the application-specific central coordinator that is mainly responsible for executing the control law. By adding this last piece to the puzzle, the integrated demo was born and successfully executed.

12.2 Suggestions

Under the slogan "no project is truly finished unless you are missing something", the following sections present some suggestions to improve the projects that were the subject of section 12.1.

12.2.1 QRoboticsCenter interfacing

One of the purposes of QRoboticsCenter is to act as a bridge between the embedded application and some other program such as a python script. This use is for instance illustrated in chapter 11. To share the information, a basic UDP port is currently used. However much more powerful networking protocols are emerging, mainly driven by the internet-of-things hype. Depending on the protocol, it is for instance possible to automatically detect peers in the network, to share information with multiple peers or to detect failures. Therefore it might be interesting to include such a protocol to QRoboticsCenter.

12.2.2 3D pose estimation

The current implementation of ProjectEagle assumes that markers move within a predefined plane. On the one hand, this assumption turned out to be acceptable in

a setting where several identical mobile robots were driving around. On the other hand, this assumption excludes other interesting uses of the system such as having mobile robots of different sizes or tracking objects in the 3D space such as a robotic arm or a quadrotor. Whereas the first example would already benefit from a plane-per-marker approach, the 3D tracking needs to step away from this assumption. A possible approach would be to estimate the 3D pose directly from the image data. This problem is referred to as the perspective-n-point problem (Lepetit et al., 2009) and requires at least four point correspondences with a reference geometry to obtain a solution. Because the marker is currently localized on the basis of three black disks, the number of points is insufficient. To overcome this issue, either an additional blob should be added to the marker or the image processing should be improved to extract more information from the marker.

12.2.3 Tracking instead of localization

ProjectEagle's image processing software has been written from a static point of view. Each image is treated separately, i.e. the image processing algorithm does not have any memory. However, taking into account the fact that the images are coming from a video stream has the potential to speed up the image processing. Prior information reduces the uncertainty about a marker's locations and hence allows a more targeted search for the marker when a new image is available. Speeding up the image processing is not only beneficial for the frequency at which measurements become available, but also decreases the delay on the measurements, which is of the utmost importance for a control system.

12.2.4 Contradictory measurements

In order to calibrate the system, the field of view of the smart camera modules within ProjectEagle should slightly overlap. If a marker happens to be positioned in this region, the amount of position measurements doubles. However, the measurements from the different cameras contradict each other to some extent due to miscalibration or camera model error. To overcome this issue, it might be beneficial to have an internal communication layer that allows the modules to communicate to each other. By doing so, the modules are able to exchange information and come to a consensus on the actual marker position, reducing the noise level.

12.2.5 Improve the demo

The developments in chapter 11 focus mainly on integration rather than demonstration. Although a fully functioning system is obtained, a nice next step would be to

bump the demonstration level. One way of doing so might be to add obstacles and replace the fairly rudimental motion planner by a motion planner worthy of the name. Inspiration in this direction can be drawn from Mercy et al. (2017) and Van Parys and Pipeleers (2017).

A

Experimental details

This appendix provides the details regarding the identification experiments performed on the ball-balancing robot described in sections 6.1.4 and 6.1.5.

A.1 Multisine attitude identification

As the robust method with random multisines has proven its value, this method is employed to estimate the transfer functions $\mathcal{D} : d_a \rightarrow y_a$ and $\mathcal{S} : d_a \rightarrow (d_a + u_a)$, which eventually yield an estimate of $\mathcal{G}_a$. The experimental settings are found in Table A.1. From the time domain data, the frequency response functions $\mathcal{D}_a$ and $\mathcal{S}_a$ are extracted using the robust detection algorithm for nonlinearities, i.e. `Robust_NL_Anal` in the Linear Control Toolbox. These are combined to obtain a nonparametric unbiased estimate of $\mathcal{G}_a$. The latter is used to identify a parametric transfer function, using a standard nonlinear least-squares approach. To this end, the Linear Control Toolbox's function `param_ident` is invoked with the method `MIMO_NLS`. The settings are listed in Table A.2. A differentiator was enforced based on physical grounds. The number of poles and zeros of the transfer function was chosen to obtain a decent fit whilst maintaining a relatively low model order. The result is shown in Figure 6.3.

number of realizations	5
period	10 s
amplitude spectrum	flat
amplitude	0.8 V
lowest excited frequency	0.2 Hz
highest excited frequency	35 Hz
sample frequency	100 Hz

Table A.1 · Experimental settings for the multisine identification experiment of the attitude.

total order of numerator	5
order of denominator	7
number of differentiators	1
number of integrators	0
frequency weighting	relative

Table A.2 · Experimental settings for the parametric attitude model estimation after a multisine identification.

number of periods	10	10	10
amplitude	5.0 V	5.5 V	3.0 V
lowest excited frequency	0.1 Hz	0.5 Hz	1.0 Hz
highest excited frequency	0.5 Hz	1.0 Hz	2.0 Hz
frequency spacing	0.1 Hz	0.05 Hz	0.1 Hz
sample frequency	100 Hz	100 Hz	100 Hz

Table A.3 · Experimental settings for the three frequency intervals during the stepped sine identification experiment of the attitude.

A.2 Stepped sine attitude identification

Because the multisine identification proposed in A.1 turned out not to be able to estimate the low-frequency dynamics with sufficient confidence, a stepped sine identification is carried out to estimate the dynamics in the region of $[0.1,\ 2.0]$ Hz.

To cope with the frequency dependent amplification, the excited frequency domain is further divided into 3 parts that are combined later on. The experimental details are provided in Table A.3. By combining the nonparametric frequency response function with the earlier frequency response obtained via a multisine identification, a better estimate of the transfer function is made. Once again, the Linear Control Toolbox's function `param_ident` is invoked with the method `MIMO_NLS` and the settings that are listed in Table A.4. Again, a differentiator is enforced but a relatively high-frequency pair of poles and zeros is dropped without an obvious increase in fitting error compared to the fitting in section A.1. The result is shown in Figure 6.4.

A.3 Velocity identification details

Similarly to the attitude identification proposed in section A.1, a closed-loop multisine identification is performed to obtain a velocity model. To overcome the problem that the energy transfer depends on where the excitation enters the loop, two

total order of numerator	3
order of denominator	5
number of differentiators	1
number of integrators	0
frequency weighting	relative

Table A.4 · Experimental settings for the parametric attitude model estimation after a stepped sine identification.

period	10 s
amplitude spectrum	flat
lowest excited frequency	0.2 Hz
highest excited frequency	35 Hz
sample frequency	100 Hz

Table A.5 · Experimental settings for the multisine identification experiment of the velocity.

nonparametric frequency response functions were identified. One is obtained by disturbing the control signal with random multisines with an amplitude of 0.01 rad. Four such experiments were carried out, yielding a nonparametric model $\mathcal{G}_h$ that is reliable in the high-frequency region. Five random multisines with an amplitude of $0.6\,\mathrm{m\,s^{-1}}$ were used as a disturbance on the reference. The resulting model $\mathcal{G}_l$ is reliable in the low-frequency region. The other settings are listed in Table A.5. The overall nonparametric model $\mathcal{G}_o$ is composed of the high-frequency part of $\mathcal{G}_h$ and the low-frequency part of $\mathcal{G}_l$, using 2 Hz as a cut-off frequency. Next, the Linear Control Toolbox's function `param_ident` is invoked with the method `MIMO_NLS` to obtain a parametric estimate of the system. The settings that are listed in Table A.6. An integrator is enforced because this was suggested by the linearized model discussed in section 6.1.3. The number of poles and zeros was found to be a good compromise between simplicity and fitting accuracy. The result is shown in Figure 6.6

total order of numerator	2
order of denominator	5
number of differentiators	0
number of integrators	1
frequency weighting	absolute

Table A.6 · Experimental settings for the parametric velocity model estimation.

Bibliography

Andersson, J. A. E., J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl (2018). "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation*.

Apkarian, P. and R. J. Adams (1998). "Advanced gain-scheduling techniques for uncertain systems". In: *IEEE Transactions on Control Systems Technology* 6.1, pp. 21–32.

Apkarian, P. and D. Noll (2006). "Nonsmooth H_∞ Synthesis". In: *IEEE Transactions on Automatic Control* 51.1, pp. 71–86.

Apkarian, P. and D. Noll (2017). "The H_∞ control problem is solved". In: *AerospaceLab Journal* 13.

Arnold, W. F. and A. J. Laub (1984). "Generalized Eigenproblem Algorithms and Software for Algebraic Riccati Equations". In: *Proceedings of the IEEE* 72.12, pp. 1746–1754.

Arzelier, D., D. Georgia, S. Gumussoy, and D. Henrion (2010). *H_2 for HIFOO*. Tech. rep. arXiv: 1010.1442. URL: http://arxiv.org/abs/1010.1442.

Beinschob, P. and C. Reinke (2015). "Graph SLAM based Mapping for AGV Localization in Large-Scale Warehouses". In: *International Conference on Intelligent Computer Communication and Processing*, pp. 245–248.

Benner, P., T. Mitchell, and M. L. Overton (2018). *Low-Order Control Design using a Reduced-Order Model with a Stability Constraint on the Full-Order Model*. Tech. rep., pp. 1–10. arXiv: arXiv:1803.06549v1.

Bernstein, D. S. and W. M. Haddad (1989). "LQG control with an H_∞ performance bound: A Riccati equation approach". In: *IEEE Transactions on Automatic Control* 34.3, pp. 293–305.

Blonk, K. van der (2014). *Modeling and Control of a Ball-Balancing Robot*. Tech. rep., pp. 1–139.

Blumrich, J. (1974). "Omnidirectional wheel". Patent US 3789947A.

Boyd, S., L. El Ghaoui, E. Feron, and V. Balakrishnan (1994). *Linear Matrix Inequalities in System and Control Theory*. Vol. 15, pp. 1–187. arXiv: arXiv:1011.1669v3.

Boyd, S. and L. Vandenberghe (2004). *Convex Optimization*. arXiv: 1111.6189v1. URL: https://web.stanford.edu/%7B~%7Dboyd/cvxbook/bv%7B%5C_%7Dcvxbook.pdf.

Bradski, G. (2000). "The OpenCV Library". In: *Dr. Dobb's Journal: Software Tools for the Professional Programmer* 25.11, pp. 120–124. arXiv: 1308.2414.

Brezak, M., I. Petrović, and E. Ivanjko (2008). "Robust and accurate global vision system for real time tracking of multiple mobile robots". In: *Robotics and Autonomous Systems* 56.3, pp. 213–230.

Bruce, J. and M. Veloso (2003). "Fast and accurate vision-based pattern detection and identification". In: *International Conference on Robotics and Automation*, pp. 1277–1282. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1241768.

Burke, J. V., D. Henrion, A. S. Lewis, and M. L. Overton (2006). "HIFOO – a MATLAB package for fixed-order controller design and H infinity optimization". In: *IFAC Symposium on Robust Control Design*. URL: http://cs1.cs.nyu.edu/cs/faculty/overton/papers/pdffiles/hifoo-rocond.pdf.

Chen, W. and F. Tu (1995). "The Strict Bounded Real Lemma for Linear Time-varying Systems". In: *Conference on Decision and Control*. December, pp. 675–680.

Chilali, M. and P. Gahinet (1996). "H_∞ design with pole placement constraints: an LMI approach". In: *IEEE Transactions on Automatic Control* 41.3, pp. 358–367. URL: http://ieeexplore.ieee.org/document/486637/.

Curtis, F. E., T. Mitchell, and M. L. Overton (2017). "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles". In: *Optimization Methods & Software* 32.1, pp. 148–181.

De Boor, C. (2001). "A practical guide to splines, revised Edition". In: *Applied Mathematical Sciences* 27.

De Caigny, J., J. F. Camino, R. C. Oliveira, P. L. D. Peres, and J. Swevers (2009). "Gain-Scheduled H_∞-Control for Discrete-Time Polytopic LPV Systems Using Homogeneous Polynomially Parameter-Dependent Lyapunov Functions". In: *Proc. 6th IFAC Symp. Robust Control Design*, pp. 19–24.

De Caigny, J., J. F. Camino, and J. Swevers (2011). "Interpolation-Based Modeling of MIMO LPV Systems". In: *Transactions on Control Systems Technology* 19.1, pp. 46–63.

De Schutter, J., J. De Geeter, T. Lefebvre, and H. Bruyninckx (1999). "Kalman Filters: A Tutorial". In: *Journal A* 40.4, pp. 538–546.

Di Sirio, G. (2018). *ChibiOS free embedded RTOS*. URL: http://www.chibios.org/dokuwiki/doku.php.

Doyle, J., A. Packard, and K. Zhou (1991). "Review of LFTs, LMIs, and μ". In: *Conference on Decision and Control*. Vol. 30, pp. 1227–1232.

Doyle, J. C., K. Glover, P. P. Khargonekar, and B. a. Francis (1989). "State-space solutions to standard H_2 and H_∞ control problems". In: *IEEE Transactions on Automatic Control* 34.8, pp. 831–847.

Fankhauser, P. and C. Gwerder (2010). *Modeling and Control of a Ballbot*. Tech. rep. arXiv: 1101.2157.

Feng, Y. and Z. Su (2014). "H_∞ Control with Output Weights For Descriptor Systems: An LMI Approach". In: *Chinese Control Conference*, pp. 4395–4400.

Fischler, M. A. and R. C. Bolles (1981). "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6, pp. 381–395. arXiv: 3629719.

Gahinet, P. and P. Apkarian (1994). "A linear matrix inequality approach to H_∞ control". In: *International Journal of Robust and Nonlinear Control* 4.4, pp. 421–448.

Gahinet, P. and P. Apkarian (2013). "Automated Tuning of Gain-Scheduled Control Systems". In: *Conference on Decision and Control*. 52, pp. 2740–2745.

Gahinet, P., A. Nemirovski, A. J. Laub, and M. Chilali (1995). "LMI Control Toolbox". In: *LMI Control Toolbox*. Chap. 7, pp. 10–14.

Gopal, M. (2002). *Control Systems: Principles and Design, 2nd Ed.*

Grigoriadis, K. M. and R. E. Skelton (1996). "Low-order control design for LMI problems using alternating projection methods". In: *Automatica* 32.8, pp. 1117–1125.

Hara, S. and H. Katori (1985). "On constrained H_∞ optimization problem for SISO systems". In: *Conference on Decision and Control*, pp. 892–893.

Henrion, D., J. Löfberg, M. Kočvara, and M. Stingl (2005). "Solving polynomial static output feedback problems with PENBMI". In: *Conference on Decision and Control and the European Control Conference*. Vol. 2005, pp. 7581–7586.

Hilhorst, G., G. Pipeleers, W. Michiels, R. C. L. F. Oliveira, P. L. D. Peres, and J. Swevers (2016). "Fixed-order linear parameter-varying feedback control of a lab-scale overhead crane". In: *IEEE Transactions on Control Systems Technology* 24.5, pp. 1899–1907.

Hilhorst, G., E. Lambrechts, and G. Pipeleers (2016). "Control of linear parameter-varying systems using B-splines". In: *Conference on Decision and Control*. Vol. 55, pp. 3246–3251.

Hilhorst, G., G. Pipeleers, W. Michiels, and J. Swevers (2015). "Sufficient LMI conditions for reduced-order multi-objective H_2/H_∞ control of LTI systems". In: *European Journal of Control* 23, pp. 17–25. URL: http://dx.doi.org/10.1016/j.ejcon.2015.01.007.

Hintjens, P. and L. Boccassi (2018). *Zyre - an open-source framework for proximity-based peer-to-peer applications*. URL: https://github.com/zeromq/zyre.

Hjartarson, A., P. Seiler, and A. Packard (2015). "LPVTools: A toolbox for modeling, analysis, and synthesis of parameter varying control systems". In: *IFAC-PapersOnLine* 48.26, pp. 139–145.

Hoffmann, C. and H. Werner (2015). "A survey of linear parameter-varying control applications validated by experiments or high-fidelity simulations". In: *IEEE Transactions on Control Systems Technology* 23.2, pp. 416–433.

Ilon, B. E. (1975). "Wheels for a course stable selfpropelling vehicle movable in any desired direction on the ground or some other base". Patent US 3876255A.

Iwasaki, T. and R. E. Skelton (1994). "All Controllers for the General H_∞ Control Problem - LMI Existance Conditions and State Space Formulas". In: *Automatica* 30.8, pp. 1307–1317.

Jacob, B. and G. Guennebaud (2018). *Eigen*. URL: `http://eigen.tuxfamily.org/index.php?title=Main%7B%5C_%7DPage`.

Klančar, G., M. Kristan, S. Kovačič, and O. Orqueda (2004). "Robust and efficient vision system for group of cooperating mobile robots with application to soccer robots". In: *ISA transactions* 43.3, pp. 329–342.

Köroğlu, H. (2013). "H_∞ Synthesis with Unstable Weighting Filters: An LMI Solution". In: *Conference on Decision and Control*. 2, pp. 2429–2434.

Kumagai, M. and T. Ochiai (2008). "Development of a robot balancing on a ball". In: *International Conference on Control, Automation and Systems*, pp. 433–438.

Kwakernaak, H. (1983). "Robustness optimization of linear feedback systems". In: *Conference on Decision and Control*. Vol. 22, pp. 618–624.

Kwakernaak, H. (1993). "Robust Control and H_∞-Optimization - Tutorial Paper". In: *Automatica* 29.2, pp. 255–273.

Lambrechts, E. and J. Gillis (2018). *OptiSpline*. URL: `https://github.com/meco-group/optispline`.

Lambrechts, E. and G. Pipeleers (2017). "B-Spline based certificates of positivity as alternative for Pólya relaxations". In: *Benelux Meeting on Systems and Control*.

Lasserre, J. B. (2009). "Chapter 2". In: *Moments, Positive Polynomials and Their Applications*. Imperial College Press. Chap. 2.

Lauwers, T., G. Kantor, and R. Hollis (2005). "One is enough!" In: *Robotics Research*, pp. 1–10.

LeafLabs (2018). *Maple | LeafLabs*. URL: `https://www.leaflabs.com/maple/`.

Lepetit, V., F. Moreno-Noguer, and P. Fua (2009). "EPnP: An Accurate O(n) Solution to the PnP Problem". In: *International Journal of Computer Vision* 81.2, pp. 155–166.

Löfberg, J. (2012). "Automatic robust convex programming". In: *Optimization methods and software* 27.1, pp. 115–129.

Loonstra, A. and R. Kierkels (2018). *Pyre - an open-source framework for proximity-based peer-to-peer applications*. URL: https://github.com/zeromq/pyre.

Ltd., Real Time Engineers (2018). *FreeRTOS - Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions*. URL: https://www.freertos.org/.

Luenberger, D. G. (1971). "An Introduction to Observers". In: *IEEE Transactions on Automatic Control* 16.6, pp. 596–602. arXiv: 0104167v5 [arXiv:cond-mat].

Masubuchi, I. (2007). "Output feedback controller synthesis for descriptor systems satisfying closed-loop dissipativity". In: *Automatica* 43.2, pp. 339–345.

Mathworks (2018). *Robust Control Toolbox*. URL: https://nl.mathworks.com/products/robust.html.

Megretski, A. (2006). "Well Posedness of LTI Feedback Design". In: *Dynamic Systems and Control, Lecture notes*. Chap. 21, pp. 9–10.

Meier, L. (2018). *Introduction - MAVLink Developer Guide*. URL: https://mavlink.io/en/.

Meinsma, G. (1995). "Unstable and nonproper weights in $\mathcal{H}_\infty$ control". In: *Automatica* 31.11, pp. 1655–1658.

Meinsma, G. (2016). *Design Methods for Control Systems, lecture slides*. Tech. rep., pp. 1–178.

Mercy, T., R. Van Parys, and G. Pipeleers (2017). "Spline-Based Motion Planning for Autonomous Guided Vehicles in a Dynamic Environment". In: *IEEE Transactions on Control Systems Technology*.

Orsi, R., U. Helmke, and J. B. Moore (2006). "A Newton-like method for solving rank constrained linear matrix inequalities". In: *Automatica* 42.11, pp. 1875–1882.

Peters, A. A., D. A. Oyarzun, E. I. Silva, and M. E. Salgado (2009). "An Analytic Characterization of a Stabilizing Feedback for LTI plants". In: *European Control Conference*. August, pp. 231–235.

Pintelon, R. and J. Schoukens (2012a). *System Identification: A Frequency Domain Approach. Second Edition*. Wiley.

Pintelon, R. and J. Schoukens (2012b). URL: http://wiley.mpstechnologies.com/wiley/BOBContent/downloadBobProjectFile.do?bpfId=1029&bpfFileType=.zip&bpfFileName=FreqDomBox.zip.

PJRC (2018). *Teensy USB Development Board*. URL: https://www.pjrc.com/teensy/.

Polya, G. (1928). "Über positive Darstellung von Polynomen". In: *Vierteljahrsschrift der naturforschenden Gesellschaft* 73.1, pp. 141–145.

Putinar, M. (1993). "Positive Polynomials on Compact Semi-algebraic Sets". In: *Indiana University Mathematics Journal* 42.3, pp. 969–984.

Roque, W. L. and D. Doering (2005). "Trajectory planning for lab robots based on global vision and Voronoi roadmaps". In: *Robotica* 23.4, pp. 467–477.

Rugh, W. J. and J. S. Shamma (2000). "Research on gain scheduling". In: *Automatica* 36.10, pp. 1401–1425.

Rzadkowski, R. and J. Sokolowski (2004). "Natural Frequencies and Modes Shapes of Two Rigid Bladed Discs on the Shaft". In: *TASK Quarterly* 8.1, pp. 51–69.

Safonov, M. G. and D. J. N. Limebeer (1988). "Simplifying the H_∞ theory via loop shifting". In: *Conference on Decision and Control*, pp. 1399–1404.

Scherer, C., P. Gahinet, and M. Chilali (1997). "Multiobjective output-feedback control via LMI optimization". In: *IEEE Transactions on Automatic Control* 42.7, pp. 896–911.

Scherer, W. and J. Hol (2006). "Matrix Sum-of-Squares Relaxations for Robust Semi-Definite Programs". In: 211, pp. 189–211.

Schweitzer, G. (2009). *Magnetic Bearings - Theory, Design, and Application to Rotating Machinery*. Springer.

Skogestad, S. and I. Postlethwaite (2001). "Uncertainty and Robustness for SISO systems". In: *Multivariable Feedback Control Analysis and Design*, pp. 253–291.

Sorkine, O. and M. Rabinovich (2016). *Least-Squares Rigid Motion Using SVD*. Tech. rep., pp. 1–6.

Steinhauser, A., M. Verbandt, N. van Duijkeren, R. Van Parys, L. Jacobs, J. Swevers, and G. Pipeleers (2017). "Low-cost Carry-home Mobile Platforms for Project-based Evaluation of Control Theory". In: *IFAC-PapersOnLine*. Vol. 50. 1, pp. 9138–9143.

Stojanović, D. H. and N. M. Stojanović (2014). "Indoor Localization and Tracking: Methods, Technologies and Research Challenges". In: *Facta Universitatis, Series: Automatic Control and Robotics* 13.1, pp. 57–72.

Stoorvogel, A. A. (1993). "The robust H_2 control problem: a worst-case design". In: *IEEE Transactions on Automatic Control* 38.9, pp. 1358–1371.

Swevers, J., B. De Moor, and H. Van Brussel (1992). "Stepped sine system identification, errors-in-variables and the quotient singular value decomposition". In: *Mechanical Systems and Signal Processing* 6.2, pp. 121–134.

The QT Company (2018). *Qt for developers by developers | Cross-platform development*. URL: `https://www.qt.io/developers/`.

Tran Dinh, Q., S. Gumussoy, W. Michiels, and M. Diehl (2012). "Combining convex-concave decompositions and linearization approaches for solving BMIs, with application to static output feedback". In: *IEEE Transactions on Automatic Control* 57.6, pp. 1377–1390. arXiv: 1109.3320.

Trofino, A. (2009). "Sufficient LMI conditions for the design of static and reduced order controllers". In: *Proceedings of the IEEE Conference on Decision and Control*. 2, pp. 6668–6673.

Van Loock, W., E. Lambrechts, G. Hilhorst, and G. Pipeleers (2016). "Approximate parametric cone programming with applications in control". In: *European Control Conference*, pp. 178–183.

Van Parys, R. and G. Pipeleers (2017). "Distributed MPC for multi-vehicle systems moving in formation". In: *Robotics and Autonomous Systems*.

Van Parys, R. and G. Pipeleers (2018). "Real-time proximal gradient method for linear MPC". In: IEEE, pp. 1142–1147.

Van Parys, R. and M. Verbandt (2018). *ProjectEagle: Robot detection with topview camera*. URL: https://github.com/meco-group/ProjectEagle.

Van Parys, R., M. Verbandt, M. Kotz, P. Coppens, J. Swevers, H. Bruyninckx, J. Philips, and G. Pipeleers (2018). "Distributed Coordination , Transportation & Localisation in Industry 4.0". In: *International Conference on Indoor Positioning and Indoor Navigation*. September.

Velasco, M., E. Aranda, H. Rodríguez, and J. González (2012). "Trajectory tracking for a wheeled mobile robot using a vision based positioning system and an attitude observer". In: *European Journal of Control* 18.4, pp. 348–355.

Verbandt, M., L. Jacobs, D. Turk, T. Singh, J. Swevers, and G. Pipeleers (2018). "Linear Control Toolbox - supporting B-splines in LPV control". In: *Journal of Mechatronics* 52, pp. 78–89.

Verbandt, M. (2018). *MECO Control Setup Interface*. URL: https://github.com/meco-group/MECO-CSI.

Verbandt, M., L. Jacobs, T. Singh, D. Turk, J. Swevers, and G. Pipeleers (2018). *LC Toolbox - An open-source linear control toolbox for MATLAB*. URL: https://github.com/meco-group/lc_toolbox.

Verbandt, M., J. Swevers, and G. Pipeleers (2016). "An LTI control toolbox - simplifying optimal feedback controller design". In: *European Control Conference*, pp. 2005–2010.

Yang, D., E. Sihite, J. M. Friesen, and T. Bewley (2015). "Design and control of a micro ball-balancing robot (MBBR) with orthogonal midlatitude omniwheel placement". In: *International Conference on Intelligent Robots and Systems*, pp. 4098–4104.

Zaccarian, L. (1998). *DC motors: dynamic model and control techniques*. Tech. rep., pp. 1–23.

Zames, G. (1981). "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses". In: *IEEE Transactions on automatic control* 26.2, pp. 301–320.

Zhang, Z. (2000). "A Flexible New Technique for Camera Calibration (Technical Report)". In: *Transactions on Pattern Analysis and Machine Intelligence* 22.11, pp. 1330–1334. arXiv: `arXiv:1011.1669v3`.

Ziegler, J. G. and N. B. Nichols (1993). "Optimum settings for automatic controllers". In: *Journal of Dynamic Systems, Measurement, and Control* 115.2B, pp. 759–765.

List of publications

Articles in international peer reviewed journals

Verbandt, M., L. Jacobs, D. Turk, T. Singh, J. Swevers, and G. Pipeleers (2018). "Linear Control Toolbox - supporting B-splines in LPV control". In: *Journal of Mechatronics* 52, pp. 78–89.

Articles in international conference proceedings

Ertveldt, J., J. Stoev, M. Verbandt, J. Swevers, S. Vanlanduit, and R. Pintelon (2016). "Study of the nonlinear aeroelastic response of a cantilever wing from wind tunnel tests on the active aeroelastic test bench". In: *International Conference on Noise and Vibration Engineering and International Conference on Uncertainty in Structural Dynamics*, pp. 431–444.

Verbandt, M., J. Swevers, and G. Pipeleers (2016). "An LTI control toolbox - simplifying optimal feedback controller design". In: *European Control Conference*, pp. 2005–2010.

Steinhauser, A., M. Verbandt, N. van Duijkeren, R. Van Parys, L. Jacobs, J. Swevers, and G. Pipeleers (2017). "Low-cost Carry-home Mobile Platforms for Project-based Evaluation of Control Theory". In: *IFAC-PapersOnLine*. Vol. 50. 1, pp. 9138–9143.

Jacobs, L., M. Verbandt, A. D. Preter, J. Anthonis, J. Swevers, and G. Pipeleers (2018). "A toolbox for robust control design : an illustrative case study". In: *International Workshop on Advanced Motion Control*, pp. 29–34.

Van Parys, R., M. Verbandt, M. Kotz, P. Coppens, J. Swevers, H. Bruyninckx, J. Philips, and G. Pipeleers (2018). "Distributed Coordination , Transportation & Localisation in Industry 4.0". In: *International Conference on Indoor Positioning and Indoor Navigation*. September.

Abstracts in international conference proceedings

Verbandt, M., G. Pipeleers, J. Swevers, K. U. Leuven, and B.-. Heverlee (2015). "Bringing optimal feedback controller design to practice". In: *Benelux Meeting on Systems and Control*, p. 87.

Verbandt, M., G. Pipeleers, and J. Swevers (2016). "An LTI control toolbox - simplifying optimal feedback controller design". In: p. 153.

Verbandt, M., J. Swevers, and G. Pipeleers (2017). "A Linear Control Toolbox - towards simple LPV control". In: *Benelux Meeting on Systems and Control*, p. 120.

Verbandt, M., R. V. Parys, M. Kotz, J. Swevers, J. Philips, and G. Pipeleers (2018). "Development and implementation of a reconfigurable assembly cell". In: *Benelux Meeting on Systems and Control*.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF MECHANICAL ENGINEERING
MECO
Celestijnenlaan 300C
B-3001 Leuven
maarten.verbandt@kuleuven.be
https://www.mech.kuleuven.be/en/pma/research/meco