MODAL INTERVAL BASED PACKAGE FOR ROBUST CONTROL

Inès Ferrer Mallorquí

DOCTORAL THESIS

MODAL INTERVAL BASED PACKAGE FOR ROBUST CONTROL

Inès Ferrer Mallorquí

2016

Universitat de Girona

DOCTORAL THESIS

MODAL INTERVAL BASED PACKAGE FOR ROBUST CONTROL

Inès Ferrer Mallorquí

2016

DOCTORAL PROGRAMME IN TECHNOLOGY

Supervised by: Dr. Josep Vehí Casellas

Presented in partial fulfilment of the requirements for a doctoral degree from the University of Girona

Universitat de Girona

Dr Josep Vehí, of University of Girona,

I DECLARE:

That the thesis titles *Modal interval based package for robust control*, presented by Inès Ferrer Mallorquí to obtain a doctoral degree, has been completed under my supervision.

For all intents and purposes, I hereby sign this document.

Signature

Girona, May 2016

Acknowledgments

I would like to thank my family: Jordi, my husband, for his encouragement and belief in my ability to complete this research work, sometimes more than my own; my two sons and daughter: Adrià, Joan, and Nuri for their patience and understanding of my preoccupation with this work and for their smiles in difficult moments; my parents for their support and providing me the opportunity of having a career; my brother for his unconditional support and assistance.

I also thank my supervisor, Dr. Josep Vehí, for his help and infinite patience.

Finally, thank you to all my colleagues of the University of Girona, and especially Dr. Lluís Pacheco for his help with the experimental part of this work.

Publications

The publications derived from this PhD thesis are the following:

Journals

- **Ferrer-Mallorquí, I.**; Vehí, J. *Combining symbolic tools with interval analysis. An application to solve robust control problems.*, American Journal of Computational Mathematics. 2014. vol(4) pp. 183-196.

- Pacheco, Ll.; Luo, N.; **Ferrer-Mallorquí, I.**; Cufí, X. *Interdisciplinary knowledge integration through an applied mobile robotics course.* International Journal of Engineering Education. 2009. vol(25) n.4. pp. 830-840.

Chapters of books

- **Ferrer-Mallorquí, I.**; Vehí, J. *Interval Robust Control framework for analysis and design of parametric uncertain systems.Advances on Control.* Dynamics and Monitoring. (Spain): 2011. ISBN 978-84-7653-539-4.

- Pacheco, Ll.; Luo, N.; **Ferrer–Mallorquí, I.** ; Cufí, X. ; Arbuse, R. *Gaining control knowledge trough an applied mobile robotics course.*Mobile robots. Currents trends. Zoran Gacovski (Croacia), 2011. pp. 69-86. ISBN: 978-953307-716-1.

- Vehí, J.; **Ferrer-Mallorquí, I.**; Sainz, M.A.; *Aplicaciones del análisis intervalar al control robusto.* El análisis de intervalos en España: desarrollos, herramientas

y aplicaciones. Ed: Universitat de Girona (Spain). Servei de publicacions. 2005, pp.121-141. ISBN: 84-934349-1-4.

- Vehí, J.; Sainz, M.A.; Armengol, J.; **Ferrer-Mallorquí, I.**;*Applications of modal interval analysis to systems and control.* Current trends in qualitative reasoning and applications. Ed. Universidad de Sevilla (Spain). Departamento de lenguajes y sistemas informáticos. 2000. pp.49-63. ISBN: 84-699-2786-8

International Conferences

- **Ferrer-Mallorquí, I.** , Vehí, J. *Combining symbolic tools with interval analysis. An application to solve robust control problems*, Applications of computer algebra ACA'10. **Vlore** (Albania), 2010.

- Pacheco, Ll.; Luo, N.; **Ferrer–Mallorquí, I.** , Cufí, X. *Control education within a multidisciplinary summer course on applied mobile robotics.*17th IFAC World Congress. Proceedings of IFAC World Congress. pp. 11660-11665 **Seul**. República de Corea. 2008.

- Pacheco, Ll.; Cufí, X. ; Luo, N.; **Ferrer-Mallorquí, I.** , . *Knowledge integration within an applied mobile robot summer course.* IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR 2008) THETA 16th edition. Proceedings of IEEE-TTTC International Conference on Automation. Quality & Testing, Robotics, pp. 310-315 **Cluj-Napoca**. Rumania. 2008.

- **Ferrer-Mallorquí, I.** . *Interval-Based tools for Robust Control.* Workshop and tutorial course on advanced computational tools for computer-Aided Design **Bremen** (Germany). 2001.

- **Ferrer-Mallorquí, I.**; , Vehí, J.; Armengol, J. *Interval-Based tools for Robust Control.* 9th GAMM - IMACS International Symposium on Scientific Computing,

Computer Arithmetic and Validated Numerics (SCAN 2000) - International Conference on Interval Methods in Science and Engineering (Interval 2000) **Karlsruhe** (Germany), 2000.

National conferences

- **Ferrer-Mallorquí, I.** , Vehí, J. *Interval Robust control framework for analysis and design of parametric uncertain systems*, Workshop on Control Dynamics. Monitoring and applications. Caldes de Montbui, 2011.

- Pacheco, Ll.; Luo, N.; Cufí, X.; **Ferrer-Mallorquí, I.** *A multidisciplinary knowledge integration experience through an applied robot mobile course.* International Technology, Education and Development Conference. INTED 2008. Valencia.

- Vehí, J. ; **Ferre-Mallorquí, I.**; Sainz, M.A. *A survey of applications of interval analysis to robust control*, 15th Triennial World Congress of the International Federation of Automatic Control. IFAC'02. Barcelona, 2002.

- Vehí, J. ; **Ferrer-Mallorquí, I.**; Armengol, J. *Using Interval and Symbolic Computations to Deal with Uncertain Dynamic Systems.* 1st Niconet workshop on numerical software in control engineering. Valencia, 1998.

List of abbreviations and symbols

ARX	AutoRegressive with eXternal input.
CACSD	Computer Aided Control System Design.
COEF	Matrix of Coeffifients.
CPSP	Parameter Space Points.
DES	Discrete Event System.
EXPO	Matrix of Exponents.
FSC	Find Specific Controller.
GS	Gain Scheduling.
IMPC	Internal Model Predictive Control.
INRIA	Institute National de Recherche en Informatique et en Automatique.
IRCAD	Interval Robust Control framework for Analysis and Design of parametric uncertain systems.
LTI	Linear Time Invariant.
MIA	Modal Interval Analisys.
MIMO	Multiple Input Multiple Output.

MPC	Model Predictive Control.
NFV	N-Functions Version.
NICONET	Numerics In Control Networks.
NP-hard	Non-deterministic Polynomial-time hard.
NPIA	Non-convex Polygon Interval Arithmetic.
OV	Overshoot.
PI	Proportional Integral.
PIA	Polygon Interval Arithmetic.
PID	Proportional Integral Derivative.
PMF	Polynomial Matrix Fraction.
PRBS	Pseudo-Random Binary Signal.
PRIM	Platform Robot Information Multimedia.
QC	Quadratic Constrain.
QE	Quantifier Elimination.
QFT	Quantitative Feedback Theory.
SCDS	Single Controller Design Specification.
SISO	Single Input Single Output.
SIVIA	Set Inversion Via Interval Analysis.
SSS	Single Square Stability.
VARI	Matrix of Variables.
WMR	Wheeled Mobile Robot.

List of Figures

List of Tables

Contents

Abstract

Complex systems are often subjected to uncertainties that make its model difficult, if not impossible to obtain. A quantitative model may be inadequate to represent the behavior of systems which require an explicit representation of imprecision and uncertainty. Assuming that the uncertainties are structured, these models can be handled with interval models in which the values of the parameters are allowed to vary within numeric intervals. Robust control uses such mathematical models to explicitly have uncertainty into account. Solving robust control problems, like finding the robust stability or designing a robust controller, involves hard symbolic and numeric computation. When interval models are used, it also involves interval computation. The main advantage using interval analysis is that it provides guaranteed solutions, but as drawback its use requires the interaction with multiple kinds of data making its implementation by control engineers complicated .

To deal with all these difficulties, in this thesis are presented two main contributions: a *methodology* integrated in a solver and a *framework*, IRCAD, that combines symbolic and numeric computation with modal interval analysis (MIA) to solve robust control problems. The methodology used reduces the task of checking the robustness of a controller to verifying the positivity of the range of a set of functions. The framework IRCAD offers an interactive environment that allows control engineers to solve robust control problems using interval analysis (MIA) in a transparent way achieving as main advantages: simplification in the computation of interval functions and semantic interpretation of the results.

As example of applicability, the contributions of this thesis have been implemented to solve the problem of local path following. An intervalar PID controller has been designed using the framework IRCAD and it has been implemented to a mobile robot.

Resum

Els sistemes complexes sovint contenen incerteses que fan que sigui difícil obtenir-ne el seu model i moltes vegades fins i tot ho fan impossible. Un model quantitatiu pot no ser el més adequat per representar el comportament de sistemes que requereixen una representació explícita de la seva imprecisió i incertesa. Assumint que les incerteses que volem tractar són estructurades, podem utilitzar models intervalars en els que els valors dels paràmetres varien dins intervals numèrics. En problemes de control robust és habitual utilitzar aquest tipus de models matemàtics per poder tenir en compte les incerteses. Un dels problemes en que ens podem trobar en la resolució de problemes de control robust com ara trobar l'estabilitat d'un sistema o dissenyar un controlador robust, és que aquesta resolució implica un alt cost computacional tan simbòlic com numèric. Si s'utilitzen models intervalars a més a més caldrà tractar amb computació intervalar. La utilització de l'anàlisi intervalar permet tractar amb aquests tipus de problemes obtenint solucions garantides. La desavantatge és que quan treballem amb anàlisi intervalar cal tractar amb múltiples tipus de dades complicant-se la seva implementació.per part de l'enginyer de control.

Per tal de tractar amb aquest tipus de problemes, en aquesta tesi es presenten dos tipus d'aportacions: una *metodologia* implementada en un solver i un *framework* anomenat IRCAD que combina computació simbòlica i numèrica amb l'anàlisi intervalar modal (MIA) per solucionar problemes de control robust. La metodologia implementada redueix la tasca de testejar la robustesa d'un controlador a la verificació de la positivitat d'un

conjunt de funcions. El farmework IRCAD ofereix un entorn als enginyers de control que els hi permet resoldre problemes de control robust utilitzant l'anàlisi intervalar (MIA) d'una forma transparent, aconseguint com a principals avantatges: una simplificació en el càlcul de funcions intervalars i una interpretació semàntica dels resultats.

Com exemple d'aplicació, les contribucions d'aquesta tesi han estat implementades a la resolució del problema de seguiment de trajectòries. S'ha utilitzat el framework IRCAD per dissenyar un controlador intervalar tipus PID i s'ha implementat el controlador trobat sobre un robot mòbil.

Resumen

Los sistemas complejos a menudo contienen incertidumbres que provocan que la obtención del modelo sea complicado e incluso muchas veces imposible. Un modelo cuantitativo puede no ser el más adecuado para representar el comportamiento de sistemas que requieren de una representación explícita de su imprecisión e incertidumbre. Asumiendo que las incertidumbres que queremos tratar son estructuradas, podemos utilizar modelos intervalares en los que los valores de los parámetros varían dentro de intervalos numéricos. En problemas de control robusto es habitual utilizar este tipo de modelos matamáticos para poder tener en cuenta las incertidumbres. Uno de los problemas en que nos podemos encontrar en la resolución de problemas de control robusto como encontrar la estabilidad de un sistema o diseñar un controlador robusto, es que esta resolución implica un alto coste computacional tanto simbólico como numérico. Si se utilizan modelos intervalares además será necesario trabajar con computación intervalar. La utilización del análisis intervalar permite tratar con este tipo de problemas obteniendo soluciones garantizadas. La desventaja es que cuando trabajamos con análisis intervalar hay que tratar con múltiples tipos de datos complicándose su implementación por parte del ingeniero de control.

Para tratar con este tipo de problemas, en esta tesis se presentan dos tipos de aportaciones: una *metodología* implementada sobre un solver y un *framework* llamado IRCAD que combina computación simbólica y numérica con el análisis intervalar modal (MIA) para solucionar problemas de control robusto. La metodología implementada reduce el problema de testear la robustez de un controlador a la verificación de la positividad de

un conjunto de funciones. El farmework IRCAD ofrece un entorno a los ingenieros de control que les permite resolver problemas de control robusto utilizando el análisis intervalar (MIA) de una forma transparente, consiguiendo como principales ventajas: una simplificación en el cálculo de funciones intervalares y una interpretación semántica de los resultados.

Como ejemplo de aplicación, las contribuciones de esta tesis han sido implementadas en la resolución del problema de seguimiento de trayectorias. Se ha utilizado el framework IRCAD para diseñar un controlador intervalar tipo PID y se ha implementado el controlador encontrado sobre un robot móvil.

Chapter 1

Introduction

The problem of parametric robust control is presented as background to the research work, the goals of the thesis are enumerated, and the outline of the work discussed.

1.1 Introduction

The main objective of a control system is to make the output of a dynamic process behave in a desirable way. The design procedure usually involves a specific or nominal mathematical model of the dynamic process. However, many details of real plant behaviour cannot be accurately captured within the model, leading to uncertainties.

Usually, high performance specifications are given in terms of the plant model. Thus, characterisation of model uncertainties should be incorporated in the design procedure to provide a reliable control system capable of dealing with the real process and assuring fulfillment of the performance requirements. The term robustness is used to denote the ability of a control system to cope with uncertainties.

Performance specifications are usually given for the regulation or tracking problem. The former manipulates the plant input to counteract the effect of output disturbances. The latter manipulates the input to keep controlled variables close to a given reference level.

The motivation for this thesis is how best to facilitate control methods and algorithms for robustness, design robust controllers, and obtain performance specifications for these models. These methods require the use of symbolic, numeric, and interval computations. Uncertainty in complex systems can be represented in many cases by interval models. To address this diversity, this work uses modal interval analysis (MIA), an extension of interval analysis that simplifies the computation of interval functions while also allowing semantic interpretation of the results.

The developed tools are integrated in a framework based on Matlab Simulink [54]. Symbolic computations are carried out using Maple routines [52], while numerical and interval computations are implemented in C++. The aim was to build an environment that would allow control engineers to address uncertain parametric systems by accessing, in a user-friendly way, the developed tool set, even for those engineers who have no knowledge about interval analysis. The framework includes analysis utilities to test stability and analyse frequency response using parametric Bode plots, design utilities to find a set of controllers that fulfill the design specifications, and tools to automatically select the optimal controller from a list of possible controllers. This selected controller can then be tested using the pot-design tools included within the framework.

1.2 The parametric robust control problem

Following the linear system of Fig. 1-1, where $\mathbf{k}$ is the parameter vector of the controller,

$$\mathbf{k} = [k_1 k_2 \cdots k_l]^T, \tag{1.1}$$

$\mathbf{q}$ is the parameter vector of the process,

$$\mathbf{q} = [q_1 q_2 \cdots q_m]^T; \tag{1.2}$$

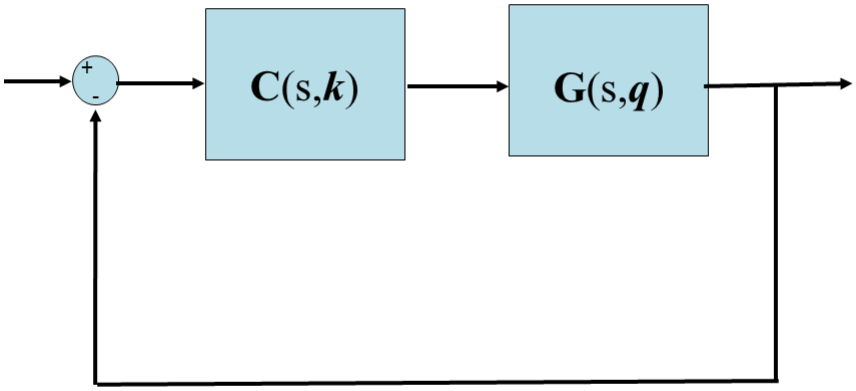


Figure 1-1: Uncertain feedback system

and G(s, $\mathbf{q}$) and C(s, $\mathbf{k}$) are the transfer functions of the process and the controller respectively. We define the uncertainty domains as the box

$$\mathbf{Q} = \{\mathbf{q} = [q_1 q_2 \cdots q_m]^T | q_i \in \left[\underline{q_i}, \overline{q_i}\right]\}. \tag{1.3}$$

$$\mathbf{K} = \{\mathbf{k} = [k_1 k_2 \cdots k_l]^T | k_l \in \left[\underline{q_i}, \overline{q_i}\right]\}. \tag{1.4}$$

This system $G(s, \mathbf{q})$, can be defined as extending to the discrete $G(z, \mathbf{q})$ where z is a complex variable defined for discrete systems, or in state space forms. For the latter case, state space form, a linear system with parametric uncertainty and static feedback control was considered [47]. Its form in this case is:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}(\mathbf{q})\mathbf{x}(t) + \mathbf{B}(\mathbf{q})\mathbf{u}(t) \\ \mathbf{u}(t) &= \mathbf{K}\mathbf{x}(t) \end{aligned} \tag{1.5}$$

where $\mathbf{A}(\mathbf{q})$ and $\mathbf{B}(\mathbf{q})$ are the space state matrices, $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector, $\mathbf{u}(t) \in \mathbb{R}^m$ is the input vector, $\mathbf{q}$ is an unknown parameter vector in a compact set $P_{\mathbf{q}} \subset \mathbb{R}^j$, and $\mathbf{K} \in \mathbb{R}^{m \times n}$ is the constant matrix.

This system has parametric uncertainty, i.e., its parameters are unknown. However, the parameters are bounded and these bounded values may be represented by intervals.

Giving this, design specifications may be formulated in terms of closed loop system stability and performances in the frequency domain. These specifications, such as bandwidth, resonance peak, control effort, etc..., can be described as a set of N inequalities of the type

$$f_i(\omega, \mathbf{q}, \mathbf{k}) > 0 \quad \omega \in \Omega, \mathbf{q} \in \mathbf{Q}, \mathbf{k} \in \mathbf{K}, i = 1, ..., N \,, \tag{1.6}$$

where ω is the frequency variable, Ω is a subset of $\mathbb{R}^+$ (usually an interval), and $\mathbf{K}$ and $\mathbf{Q}$ are nondegenerate sets.

Robust design may then be formulated as follows: given a controller structure, $C(s, \mathbf{k})$, the aim is to find the values of $\mathbf{k}$ which conform to the robust control specifications. Using this formulation some control problems can be proposed [88]:

1. *Performance checking.* Given the uncertain system, $G(s, \mathbf{q})$, and uncertain domain, $\boldsymbol{Q}$, the problem is to show the designed controller, $C(s, \mathbf{k}^0)$, achieves the robustness specifications from 1.6,

$$f_i(\omega, \mathbf{q}, \mathbf{k}^0) > 0 \tag{1.7}$$

2. *Performance margin computation.* Consider the plant, $G(s, \mathbf{q})$, which is described by a linear time invariant transfer function dependent on the parameter vector, $\mathbf{q}$. The value of $\mathbf{q}$ is known to belong to a box, Π, as a function of its radius, ρ [51],

$$\Pi(\rho) = \left\{ \mathbf{q} : \left\| \mathbf{q} - \mathbf{q}^0 \right\|_\infty^w \leq \rho \right\} , \tag{1.8}$$

where $\|\mathbf{q} - \mathbf{q}^0\|_\infty^w = max_i w_i^{-1} \left| q_i - q_i^0 \right|$ with q_i^0 are the nominal values of the parameters and $w_i > 0$ are known weights. The aim is to find the maximal set, $\Pi(\rho^*)$, so that the designed controller, $C(s, \mathbf{k}^0)$, achieves robust performances for all $\boldsymbol{\rho}$ belonging to $\Pi(\rho^*)$.

3. *Robust controller design.* Taking a particular structure for the controller and specifying the uncertain domain where the parameters of the system $\mathbf{Q}$ can vary, find a fixed structure controller, $C(s, \mathbf{k}^0)$, such that the controlled closed loop system achieves the robustness specifications

$$f_i(\omega, \mathbf{q}, \mathbf{k}^0) > 0 \tag{1.9}$$

4. *Obtain the set* $\mathbf{K}$*for the robust controller problem.* Given an uncertain plant, $G(s, \mathbf{q})$, the variation domain of the system parameters, $\mathbf{Q}$, and the controller structure, find the robust set $\mathbf{K}$ which allows $C(s, \mathbf{k})$ to achieve the robustness specifications

$$f_i(\omega, \mathbf{q}, \mathbf{k}) > 0 \tag{1.10}$$

5.*Estimate the stability region problem for a given* $\mathbf{k}^0$. Construct a set of all the regions, $\boldsymbol{\rho}$, giving closed loop stability given $\mathbf{k}^0$.

The framework allows us to deal with all these problems offering the control engineer a useful environment to work with intervals in a transparent way.

1.3 Goals

The main goal of this thesis is to build a framework for robust control analysis and design, to deal with systems that involve uncertain parametric models with the parameters modelled by intervals.

Obtaining a precise model for a physical system is difficult, and when available it is often too complicated to be useful in a control context. Thus, an interval parametric model is necessary to formulate such problems as robustness analysis, robust control design, interval simulation, and controller implementation.

The developed framework, the Interval Robust Control for Analysis and Design (IRCAD), addresses uncertain parametric systems and allows engineers user-friendly access to the control tools and methods. The goals of the thesis were achieved using interval

analysis in a transparent way, and applied it to systems that involve uncertain parametric models, with the parameters modelled by intervals. IRCAD provides a user-friendly environment to facilitate control engineers working with systems that involve intervals.

1.4 Outline of the Thesis

The thesis is structured as follows:

- Chapter 2: *Interval analysis approach to robust control.*

  Reviews the literature concerned with the application of interval analysis methods to problems of control and overviews the most representative works. It also describes frameworks that address robust control problems.

- Chapter 3: *Robust control via modal interval analysis.*

  A short introduction to MIA and its implementation to improve outcomes for robust control problems.

- Chapter 4: *Parametric solver based on modal interval analysis.*

  The first contribution of this thesis. Algorithms based on MIA are presented and the methods and tools generated to create a solver based on MIA that solves robust control problems are described.

- Chapter 5: *Parametric framework for robust control .*

  The second contribution of this thesis. The proposed parametric framework (IRCAD) to solve robust control problems is described. The framework uses MIA based algorithms in a transparent way to achieve improved results on complex robust control problems.

- Chapter 6: *Design example using IRCAD.*

  Selected robust control problems with high complexity in both analysis and design are addressed using the proposed framework.

- Chapter 7: *Applications of IRCAD to mobile robot control design.*

  Presents experimental work applying a control system developed with the IRCAD framework to a mobile robot (PRIM) to an application of local path following.

- Chapter 8: *Conclusions and further research.*

  Concludes the thesis, summarizing the work described, provides concluding remarks, and some proposals for further research.

Chapter 2

Interval analysis approach to robust control

We present an overview of approaches using interval analysis to treat the problem of parametric robust control, and approaches of computer aided control system design (CACSD) providing a comparative analysis of these.

2.1 Introduction

This chapter concentrates on the major developments of interval analysis to robust control problems. Robust control has generated a number of new research areas over the last few decades. Faedo [18] first proposed stability analysis for uncertain coefficient polynomials. However, his work is not considered an antecedent for the current work, because he did not consider interval arithmetic. Misra [56] presented the first study of robust analysis using interval analysis as a tool.

When setting the problem, it might be interesting to go back to Dorato [14], in which, although none of the works summarized there are based on parametric methods, Kharitonov polynomials are introduced. A definition of robust control can be extracted from this work as being the problem of analysing and designing precise control systems

given plants which contain significant uncertainties. Dorato and Yedavally [15] used parametric methods, following Kharitonov's results. Approaches in this field fall primarily into two methods: polynomial and parametric methods. are Many researchers (e.g. [7, 1, 9, 13]) have developed methods based on Kharitonov's polynomials.

Interval arithmetic extends computation on real numbers to intervals in a natural and intuitive way and is the natural tool to use when dealing with interval models. The basis for the tool can be found in Moore [57, 58], followed by Alefeld [3]. As a result of maturing interval analysis in the mathematical field, applications have been subsequently developed using this tool, e.g. Hansen[34] suggests global optimization.

Time interval analysis was a well-established tool by 1990, and has been applied to different control problems in all applied control fields. Thus, applications of interval analysis to robust control have arisen, and, following the same evolution as parametric methods, the approaches were mainly centered on analysis with some later works considering design.

This chapter discussed the most relevant methods used to apply interval analysis to robust control. The increasing number of studies across many control fields testifies to the importance of this field.

The last section of this chapter is centered on analysing the existing CACSD to solve robust control problems.

2.2 Applications of interval analysis to robust control

The approaches to interval analysis for robust control fall into four groups: three general control fields (parameter space, frequency, and discrete time methods), and "other applications", which includes those that do not fit in any of the other three groups.

2.3 Parameter space methods

The first approaches to robust stability under real parametric uncertainty, following the introduction of Routh Hurwitz's theorem, suggested that robust control problems for real parametric uncertainties could be approached without conservatism or overbounding. However, later works offered a more realistic vision of these approaches.

The main feature to consider when analysing parametric approaches is that the structure of the model to be treated should to be given. These structures can be expressed in terms of a difference or differential equation of fixed order, or in terms of vector space. Development of algorithms and tools to solve some classical control problems using parametric space approaches, implied a required preliminary task transforming the parameter space model to interval functions. The transformed problems could then be solved using parametric theory. Thus, we must consider the computation of the parametric stability margin over a given uncertainty set, which in turn is a measure of the robust performance of the system.

Another common control aim is to check the stability and its solution inside the parameter space. Some applications can also be found in the design field. Parameter spaces of the controller are explored, finding the regions which fulfill a control specification, to identify a controller that satisfies it.

The statement of the parametric robust control problem is introduced in section1.2. Studies in this field have classified the parametric robust control problem into four main categories: robustness analysis, robust design, state space, and the H_2/H_∞ approach. These categories are below.

2.3.1 Robustness analysis

The parametric approach to robust stability analysis has received a great deal of attention in the past few years. Piazzi et al. [75] had already conducted a survey based primarily on robust stability. Several other approaches illustrate these problems. Walter et al. [94],

proposed that characterization of the stability domain can be approached as a problem of set inversion which can be solved with interval analysis tools. Garloff et al. [25], proposed an algorithm which relies on the expansion of a multivariate polynomial into Bernstein polynomials and was based on the inspection of the value set of the family of polynomials on an imaginary axis. Jaulin et al. [39] used interval analysis to develop an algorithm that proved the feasible set is included in the stability domain. Malan et al. [51] provided an approach to finding global minima of multimodal optimization problems. They proposed a Bernstein branch and bound algorithm (B^3), which provided an efficient and simple way to check if the polynomial reached its minimum on one of the vertices of the domain. However, while many approaches to robustness analysis have been developed, none of them have been illustrated with examples.

To conclude this section, an approach proposed by Didrit et al. [12] is presented and their work will be used to illustrate the approaches presented above. The concept underlying this example is to show the performance and the limitations of branch and bound algorithms, including modifications based on modal intervals to improve their effectiveness in non-monotonic regions [89, 88].

Example 1 *Given the third-degree uncertain polynomial,*

$$p(s, \boldsymbol{q}) = 2 + r^2 + 6q_1 + 6q_2 + 2q_1q_2 + (2 + q_1 + q_2)s + (2 + q_1 + q_2)s^2 + s^3,$$

We classify the parameter space in unstable regions, as regions with stability between 0 and 0.1, and regions with stability greater than or equal to 0.1.

The condition of stability greater than a is obtained by substituting s with $s = -a + j\alpha, \quad \alpha \geq 0$ in the characteristic polynomial and applying the classic Hurwitz test,

$$\begin{aligned} F(\boldsymbol{q}) \;=\; & -14q_2a - 14q_1a - 32a^2 - r^2 + 8a^3 + 2q_2^2a + 4q_1aq_2 \\ & -8q_1 + 24a + 32 + q_1^2 + q_2^2 + 2q_1^2a + 8q_1a^2 + 8q_2a^2 - 8q_2 > 0. \end{aligned} \tag{2.1}$$

For $a = 0$ and $a = 0.1$, the stability conditions are

$$32 - r^2 - 8q_1 - 8q_2 + q_1^2 + q_2^2 > 0, \tag{2.2}$$

respectively, and

$$1.2\left(q_1^2+q_2^2\right)+34.08-9.32\left(q_1+q_2\right)-r^2+0.4q_1q_2>0. \tag{2.3}$$

To calculate the absolute stability region (stability 0) and the region which reaches stability 0.1, a branch and bound algorithm based on modal intervals is used, analysing two functions and generating four linked lists:

1. Unstable regions
2. Regions with stability between 0 and 0.1
3. Regions with stability greater than 0.1
4. Residual rectangles.

Figure 2-1 shows the results from an algorithm built on structures of algorithms based on modal intervals. In Fig. 2-1 (a), the outer region has stability greater than 0.1 and the inner region is unstable, whereas regions in the intermediate zone have stability between 0 and 0.1.

The particular case $r = 0$ is shown in Fig. 2-1 (b).The unstable region of parameter space is a point. Thus there is a single, unique, unstable polynomial.

Note that, contrary to other methods, modal interval algorithms do not omit the unstable point, as there is an undetermined region around it.

2.3.2 Robustness design

In terms of robustness design, the number of approaches is still very limited and they largely concentrate on controller design. To illustrate this problem, we present an example

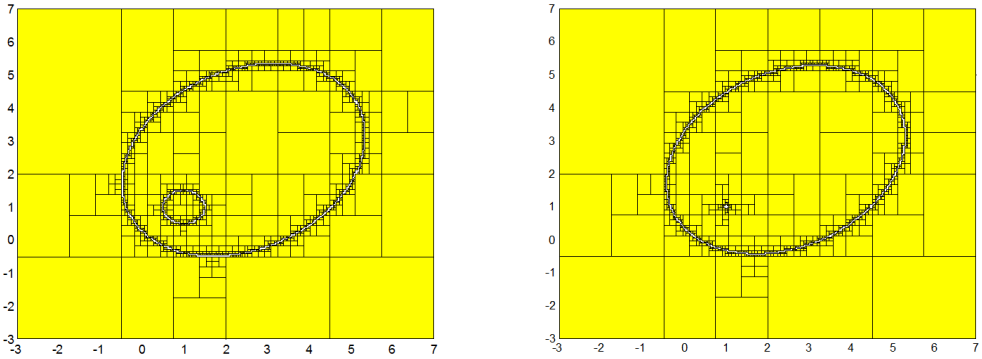


Figure 2-1: Stability regions. (a) $r = 0.5$. (b) $r = 0$

suggested by Fiorio et al. [21], and later studied by Malan et al. [50], with the aim of tuning a PI (Proportional Integral) controller for an interval plant. Fiorio proposed a method based on Bernstein's polynomial expansion for the problem of designing robust controllers of fixed structure dependent on some free design parameters. Modal intervals have also been applied to this example [88]

Example 2 *The plant is described by the transfer function*

$$G(s, \mathbf{q}) = \frac{q_1}{1 - \frac{s}{q_2}}, \tag{2.4}$$

where q_1 *and* q_2 *are the uncertain parameters remaining inside an interval* $q_i = [0.8,\ 1.25]$.

The PI controller is

$$C(s, \mathbf{k}) = \frac{k_1 \left(1 + \frac{s}{k_2}\right)}{s}, \tag{2.5}$$

where $\mathbf{k} = [k_1\ k_2]^T$ *is the design parameter vector.*

The design aim is to find the parameter set, K, of the controller that completes the performance specifications

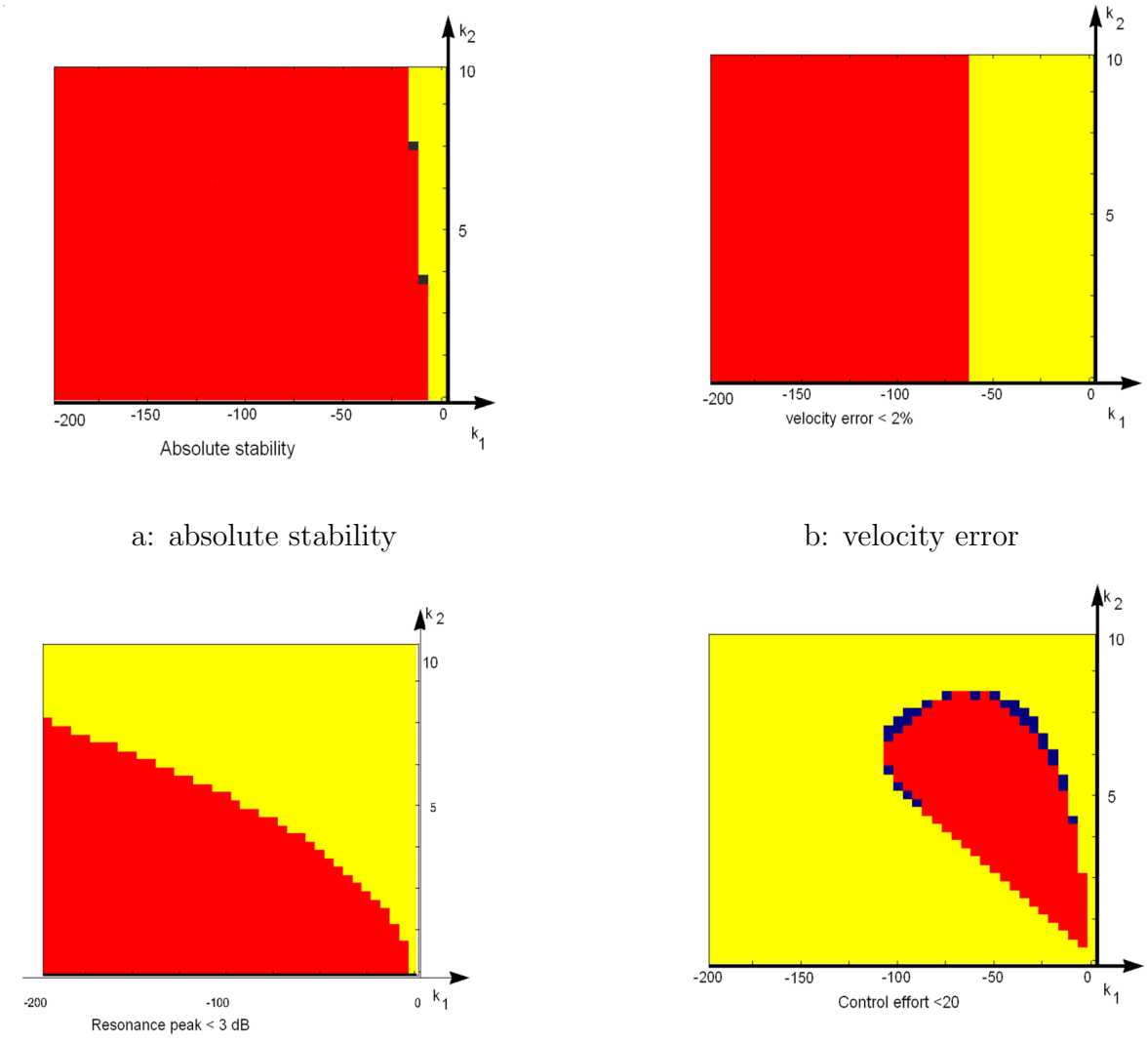
1. Closed loop stability.

2. Velocity error less than 2%.

3. Control effort less than 20.

4. Resonance peak of the closed loop transfer function less than 3 dB.

Given the initial range, $\mathbf{K}_{init}$, as a starting point, the algorithm computes the set of controllers, $\mathbf{K}$, which fulfill the performance specifications (Fig 2-2).



a: absolute stability

b: velocity error

c: resonance peak

d: control effort

Figure 2-2: Feasible regions

In this example, the regions of the feasible controllers for each of the specifications are computed separately.

The parameter space region of the feasible controllers is the intersection of the four regions determined by application of the four different specifications, as shown in Fig. 2-3.

The application of coercion theorems from MIA reduces computation time for the most complex case to less than half.

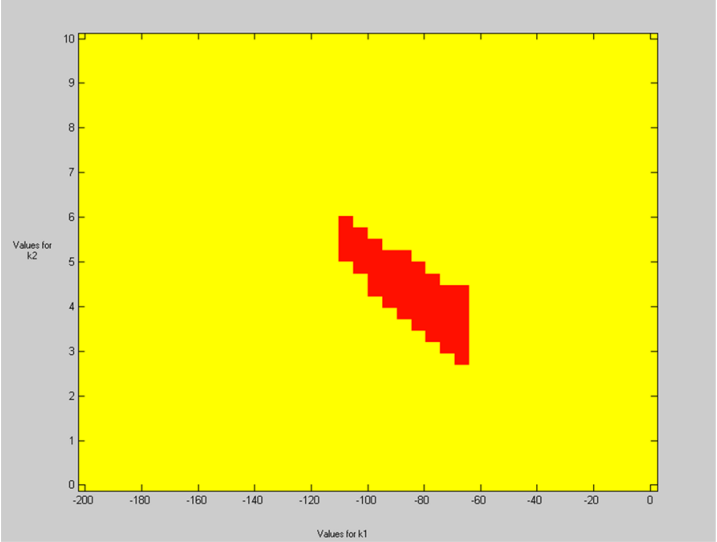


Figure 2-3: Region of feasible controllers

Malan et al. [50] provided and overview of some of the main interval mathematical algorithms to test their efficiency on several problems, such as robustness design. Fiorio and Malan[21] directly considered the presence of parametric perturbations, which has not been treated by many authors. Piazzi et al. [77] proposed a feedforward/feedback synthesis design with the aim of minimizing the worst case settling time relative to the transition. This method used feedback to reduce the sensitivity to parametric uncertainties, and then system inversion to obtain a reference input corresponding to an arbitrarily smooth output without oscillations. From there, it is only necessary to solve a simplified optimization problem to complete the design.

2.3.3 State space

Misra [56], is considered as the first use interval arithmetic in an explicit form to study stability analysis. He proposed the problem of finding the state feedback vector that achieved stability for an interval coefficient polynomial.

A SISO (Simple Input Simple Output) system, represented by a transfer function, $g(s) = n(s)/d(s)$, can be described as the controllable realization

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ -a_0 & -a_1 & \cdots & -a_{n-2} & -a_{n-1} \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(t), \qquad (2.6) \\
\mathbf{y}(t) &= \begin{bmatrix} b_0 & b_1 & \cdots & b_{n-2} & b_{n-1} \end{bmatrix} \mathbf{x}(t),
\end{aligned}$$

where a_i and b_i are the *i-th* order coefficients of the numerator and denominator, respectively. Because this system is controllable, a control $u(t) = -\mathbf{k}^T\mathbf{x}(t)$ will always exist, which gives the closed loop system the desired eigenvalues. Therefore, given an interval polynomial,

$$d(s) = [\underline{a_0}, \overline{a_0}] + [\underline{a_1}, \overline{a_1}]s + [\underline{a_2}, \overline{a_2}]s^2 + \cdots + s^n; \qquad (2.7)$$

and a state feedback polynomial, $\mathbf{k}^T = \begin{bmatrix} k_0 & k_1 & \cdots & k_{n-1} \end{bmatrix}$; the characteristic closed loop polynomial will be

$$\begin{aligned}
d_{CL}(s) &= [1,1]s^n + [\underline{a_{n-1}} + k_{n-1}, \overline{a_{n-1}} + k_{n-1}]s^{n-1} + \cdots \qquad (2.8) \\
&\quad ... + [\underline{a_0} + k_0, \overline{a_0} + k_0].
\end{aligned}$$

The solution proposed by Misra uses Routh's table for the feedback interval poly-

nomial and computes the elements of the first column via the evaluation of its natural extension. Computing the elements of Routh's table using interval arithmetic overestimates the exact range, so sufficient conditions for stability are obtained. To find the natural extension of the Routh table elements, Misra used symbolic computation. A sufficient condition for stability is that the lower bounds of the interval result for the first column be all larger than zero. This gives a nonlinear system of algebraic inequalities that, if they have a solution, yield $\mathbf{k}$. If the system has no solution, this does not necessarily mean that $\mathbf{k}$ does not exist, because of overestimation when computing the elements of Routh's table.

Thus Misra addresses a problem with many restrictions, but only considers the case of polynomials with interval coefficients and, hence, only obtains a partial solution. That is, it is not certain that a state feedback vector stabilizing the plant will be found, even if such a feedback vector exists. The problem can be solved more simply using the Kharitonov theorem. However, this is only a starting point based on an interval form from Faedo's proposition [18].

Another interesting approach, following the same approach on state space but more general than Misra's, was proposed by Kwon and Cain [47, 48]. They consider the problem of finding a state feedback vector for an uncertain system with two matrices, $\mathbf{A}$ and $\mathbf{B}$, which depend on an uncertain parameter vector.

They proposed a linear system with parametric uncertainty and a static state feedback control, given by

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}(p)\mathbf{x}(t) + \mathbf{B}(p)\mathbf{u}(t) \\ \mathbf{u}(t) &= \mathbf{K}\mathbf{x}(t), \end{aligned} \tag{2.9}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector, $\mathbf{u}(t) \in \mathbb{R}^m$ is the input vector, p is an unknown parameter vector and $\mathbf{K} \in \mathbb{R}^{m\times n}$ is the constant matrix. The purpose is to find a constant matrix which can stabilize the perturbed system against all parametric perturbations over

a variation domain, Q.

By letting $\mathbf{k} = vec(\mathbf{K})$, and calling $g(\mathbf{k}, \mathbf{q})$ the vector formed by the coefficients of the characteristic polynomial, Kwon and Cain show that the problem of robust pole location is equivalent to finding a vector k^*, such that $g(\mathbf{k},\mathbf{q}) > 0,\ \forall q \in Q$. The problem can then be formulated and solved as an NP-hard optimization. The algorithm used to implement this optimization was based on interval analysis and the theory of semi-infinite programming. Although the algorithm can be theoretically applied to any nonlinear dependency between the uncertain parameters and the coefficients of the plant, in the Kwon and Cain example, the uncertainty exists only in the A matrix and as an interval form.

2.3.4 $\mathbf{H_2/H_\infty}$

To treat problems in the context of a state space description of the process, researchers have proposed to cast them in an H_2/H_∞ optimization framework. Classical techniques are usually applied to solve these, such as the Ricatti equation. Searching for more innovative techniques to solve H_2/H_∞ fixed structure controller design problems leads to a design method based on a global optimization approach to a single input, single output (SISO) H_2/H_∞ problem. Another approach based on unity feedback systems and genetic procedures can also be usefully applied.

Optimal H_2/H_∞ controller design aims to find a k^* such that a feedback controller $C(s; k^*)$ internally stabilizes the plant while minimizing a nominal H_2 cost, subject to the robust closed loop stability constraint, (H_∞).

With the aim of determining a global minimizer, k^* , an effective option is a global optimization algorithm with hybrid features. This is based on a genetic algorithm at the upper level and an interval procedure at the lower level to handle semi-infinite constraints.

Another possibility is to use a technique based on the partially elitist genetic algorithm. Using this method, computation of some terms makes a special interval procedure necessary, i.e., a deterministic algorithm that uses concepts of interval analysis. An al-

gorithm of this kind was presented by Guarino et al. [26].

Considering the problem of calculating compensators, Weinhofer et al. [96] presented a new H_∞ approach that, after converting the original problem into a suboptimal Nehari problem, builds the compensator using interval arithmetic to avoid numerical problems. Thus, it obtains a precision bound on the results, i.e., each result lies within a known interval. This approach uses frequency domain methods for the design process which offer a minimal degree for the compensators. Consequently, a numerically stable design tool was presented [30]where the obtained interval could be adjusted dynamically, making it possible to calculate results with a predefined precision.

Haas et al. [32] proposed an approach for the case of design methods for H_2 compensators (two parameter compensators) for linear multivariable plants. The aim of these methods was to find an optimal fixed structure controller that minimizes a nominal H_2 cost. The proposed method had two main advantages over most published H_2 design methods: decoupling of the design of reference tracking and disturbance rejection, and optimization of a cost function with respect to non-square integrable deterministic signals. Interval arithmetic was introduced to obtain a bound on the precision of the results. This makes it possible to avoid numerical obstacles and obtain a numerically stable design tool.

Once approaches H_2 and H_∞ were proposed, a mixed study between H_2 and H_∞ was presented by Guarino et al. [29], where they proposed a solution to a H_2/H_∞ fixed-structure controller design problem via a global optimization approach. To implement this approach, a necessary preliminary step was to convert the H_∞ constraint into a semi-infinite inequality over a real bound interval. Thus, the optimization problem was reduced to a bound constrained problem. To solve this new problem, the genetic/interval algorithm [26] was proposed. In a previous study by these authors [27], interval analysis was also applied to a global optimization problem. In that case, via an *ad hoc* interval procedure incorporating concepts of interval analysis to obtain convergence with certainty within the specified numerical tolerance.

The approach by Haas et al.[30], presented a control toolbox which implemented the most common algorithms used in the frequency domain. Thus, its implementation is relevant to all the approaches presented above.

2.4 Frequency methods

Another approach to the robust control of parametric models consists of frequency methods. Classical frequency control has been largely focused on the frequency domain properties of control systems and design methods based on simple but powerful graphical tools such as the Nyquist plot, Bode plots, and the Nichols chart. For multivariable systems, it is necessary to extend these classical methods to families of polynomials.

The family of polynomials is

$$P(s, K) = \{p(s, k) \quad | \quad k \in K\}, \tag{2.10}$$

which in turn have been generated by an uncertain polynomial

$$p(s, k) = a_0(k) + a_1(k)s + a_2(k)s^2 + \cdots + a_n(k)s^n, \tag{2.11}$$

where $a_0(k), a_1(k), ..., a_n(k)$ are real and dependent on a constant but unknown real parameter vector, $k = [k_1 k_2 \cdots k_l]^T$. Independent bounds are in the form $k_i \in [\underline{k_i}, \bar{k}_i]$ for parameter. Consequently, the set of possible parameter vectors is

$$K = \left\{k = [k_1 k_2 \cdots k_l]^T \mid \quad k_i \in [\underline{k_i}, \bar{k}_i], \quad i = 1, 2, ..., l\right\}. \tag{2.12}$$

For robust analysis of multivariable systems in the frequency domain, the number of parameters increases, and new methods must be found. In these cases the complex value taken by a polynomial when it is evaluated at $s = jw$ will be called its *value* at frequency w. Computing and plotting the value of a polynomial for all non-negative frequencies

creates the polynomial frequency plot. The most representative methods in this sense are based on these parameterized sets and their aim is to determine the complex plane images of the sets.

Two methods are presented to generate the sets. In the first method the sets are called *value sets* and the proposed problem to be solved is value set computation. In the second method the sets are called *templates* and the objective is to solve the template generation problem and is studied as a QFT problem.

2.4.1 Value sets

Some gridding approaches have the drawback of the length of time it takes to compute the set of frequency plots. An alternative and faster technique [1], is to compute the frequency plot, $p(jw, \mathbf{k}), w \geq 0$, for each $\mathbf{k}$ on a grid of K. That is, to compute the value set for each w on a grid of frequencies from 0 to $+\infty$,

$$P(jw, K) = \{p(jw, \mathbf{k}) \in \mathbb{C} \mid \mathbf{k} \in K\}. \tag{2.13}$$

The value set problem is usually considered as a graphical control tool for analysis using frequency plots, and a analysis problem is checking stability. In the case of a family of polynomials, the stability test is based on repeating the construction of the value set for a grid of frequencies to give the set of all possible frequency plots. The collection of value sets would then indicate stability or instability depending on how the zero exclusion theorem is formulated.

Theorem 1 *(Zero exclusion). Given a polynomial family* $P(s, K) = \{p(s, k) \mid k \in K\}$. *This set is robustly stable if and only if*

1. *A stable polynomial* $p(s, \mathbf{k}) \in P(s, K)$ *exists and*

2. $0 \notin P(jw, K)$ *for all* $w \geq 0$.

Ackermann [1] presents three methods for robust stability analysis. The first checks stability with a root set construction, and can only be approximated by use of a sufficiently accurate parameter set grid, which is effective despite the drawback of a too high computational time. The second is an algebraic approach which provides a useful tool, but has the disadvantage of excessive complexity in computation which may occur even when the uncertainties are relatively simple in form. With the third method, the parameter space approach, the main difficulty is the need to deal with singular frequencies. Even though the graphical information concerning stability is very appealing and highly informative, its application is restricted to two uncertain parameters.

Some authors have focused their studies on finding methods which allow the value sets to be expressed in a form that made it possible to apply the zero exclusion theorem simply. Otha et al. [66] introduced polygon interval arithmetic (PIA). This powerful tool solves robust controller design (and robust stability analysis) more efficiently than other classical methods. Ohta et al. [67] subsequently proposed an improvement on this tool to reduce computing time. In the same year, Ohta proposed two design methods as applications of PIA. The first [71] was based on the gain phase shaping approach, with the main advantage of guaranteeing the worst case performance. The second [68] proposed a method of computing an almost exact gain margin. Another field of control where Ohta applied PIA was efficiently solving zero exclusion problems [69], where PIA was used to estimate the value sets of multi-linear functions. Ohta et al. [70] subsequently proposed an extension called non-convex polygon interval arithmetic (NPIA), where the arithmetic was defined within the set of all polygons in the complex plane.

It is very useful to compute estimates of value sets of transfer functions including uncertain physical parameters in a reasonable computing time. Ohta et al. [72], in a more applied paper, proposed a method to compute a region of parameters of a PID (Proportional Integral and Derivative) controller which guarantees robust stability and several robust performances for systems with uncertain parameters. Again in the field of PID controllers, Ohta et al. [65] address the design of a two degrees of freedom robust

PID controller by solving minimization problems. Since in this last approach the obtained regions are not convex, set operations are used to compute level sets of the minimization problems.

For nonlinear control systems with uncertain parameters and constant reference inputs (Lur'e systems), the possibility of a shift in the equilibrium state or loss of stability increases. Wada et. al [93] check conditions for parametric absolute stability for this case by computing value sets using the PIA algorithm, because it significantly simplifies computation.

Other approaches have been focused on more applied fields. For example Hedrich et al. [35] formulated a possible verification technique of linear analog circuits with parameter tolerances based on a curvature driven bound computation for value sets using interval arithmetic. The advantage of this tool compared to other studies that also compute performance characteristics from an actual circuit and compare this to specifications, is the capability of an exact test of the correctness of the design.

2.4.2 Quantitative feedback theory

While value sets are largely concerned with analysis, when the aim is finding a compensator to satisfy design specifications, quantitative feedback theory (QFT) is the most important frequency approach. This can be considered as a natural extension of classical frequency domain design approaches. One of the main objectives is to design a simple low order controller where the bandwidth of the feedback controller is as small as possible. For a fixed frequency, the plant's frequency response set is called a template. In the bound generation step of QFT design procedure, the plant template is used to translate the supplied robustness specifications in domains in the Nichols chart where the controller gain phase values are allowed to lie.

To perform frequency response analysis and design incorporating robustness with respect to parameter uncertainty, we need to be able to determine the complex plane images of various parameterized sets.

For QFT, the sets are called templates and the problem of computing the template at a given frequency is the template generation problem. Plant uncertainty is represented as a template of possible complex values of the plant transfer function, $G(jw)$, at a given frequency, w. The actual loop gain at any given frequency is a set of values given by the nominal gain plus the template of possible plants. The solution is then based on shaping the nominal loop gain so that the feedback system stays robustly stable and satisfies various design objectives, such as input-output accuracy and/or noise rejection, among others.

In several applications of QFT, templates of non-rational transfer functions must be numerically generated. Sardar et al. [80] proposed an algorithm to generate templates of uncertain non-rational transfer functions which avoids the requirement of rational transfer function approximation. Nataraj et al. [60] introduced an algorithm based on the Moore-Skelboe global optimization technique of interval mathematics to generate Bode plot envelopes for uncertain transfer functions. Subsequently, Nataraj et al. [61] proposed quadratic constraint (QC) algorithms for computing QFT bounds to achieve robust sensitivity reduction and gain phase margin specifications. These algorithms improve on existing algorithms where discrete controller phase values are used, and can generate bounds over intervals of controller phase values solving the difficulty that, at the non-selected phase, the bound values are not actually computed. Thus, Nataraj et al. [62] proposed algorithms which use interval analysis to build plant templates, achieving important improvements, including reducing safety problems associated with the phase discretization process in QFT bound generation, and improving the security of computed results. Moreover, they can generate bounds over intervals of controller phase values, as opposed to just the discrete controller phase values of existing algorithms.

2.5 Discrete time control

The processes are generally assumed to use continuous time. However, the ever increasing availability of computers has allowed breaking the processes into discrete time models for many domains. Numerical simulation of discrete time models is much simpler and quicker, which makes them well suited to real time process control. However, they may engender some loss of information on the behaviour of the underlying continuous time system.

It should be noted that since discrete time models are simpler to simulate numerically, it is possible to push the experimental study of their properties much further. On the other hand, they impose constraints on measurement times, which must correspond to the discrete times for which the model output is computed, and they may hide oscillations of the associated continuous time system. Furthermore, their parameters generally do not have any clear physical meaning. In particular, the parameter values of a model obtained by discretizing a continuous time model depend on the sampling period chosen.

Some known issues with discrete time models are:

- Model conversion.

  Conversion is needed in both senses, from a continuous time interval state space model to a discrete time interval model and vice versa.

- Schur stability test.

  The problem of checking the stability of a discrete- time system is reduced to determination of whether or not the roots of the characteristic polynomial of the system lie strictly within the unit disc, i.e., whether or not the characteristic polynomial is a Schur polynomial. If

$$P(z) = p_n z^n + p_{n-1} z^{n-1} + ... + p_1 z + p, \tag{2.14}$$

  where z_i are the n roots of $P(z)$, then, if $P(z)$ is Schur, all these roots are located

inside the unit circle, $|z| < 1$, so that when z varies along the unit circle, $z = e^{j\theta}$, the argument of $P(e^{j\theta})$ increases monotonically. For a Schur polynomial of degree n, $P(e^{j\theta})$ has a net increase of argument of $2n\pi$, and thus the plot of $P(e^{j\theta})$ encircles the origin n times. This can be used as a frequency domain test for Schur stability [8]

- Non-linear discrete-time control of uncertain systems.

  This problem can be formulated as: Find one c,

$$c \in S_c = \{c \in C \quad | \quad \forall p \in P, \quad f(c,p) > 0\}, \tag{2.15}$$

  where f is a vector function that can be evaluated using algorithms based on interval analysis. This problem is both quite complex because it involves a quantifier, and at the same time very simple because the only objective is to find a single feasible vector. This makes the consideration of a larger number of tuning parameters corresponding to the guaranteed tuning problem [42] possible. Thus it is possible to combine nonlinearity and structured uncertainty with guaranteed results.

- Robust analysis.

  Given a plant assumed to be described by the uncertain discrete time transfer function [91]

$$G(z^{-1}, q) = \frac{B(z^{-1}, q)}{A(z^{-1}, q)} = \frac{b_1(q)\, z^{-1} + b_{21}(q)\, z^{-2} + \cdots + b_m(q)\, z^{-m}}{1 - a_1(q)\, z^{-1} - \cdots - a_n(q)\, z^{-n}}, \tag{2.16}$$

  which depends on a structured perturbation characterized by (1.1)- (1.4) .

  MIA [23] can be introduced here, showing that incorporating it in the analysis of the robustness of predictive controllers allows us to convert the robust stability problem into a problem of checking the positivity of a rational function, i.e., a problem where the aim is to verify the positiveness of the range of a set of functions.

- Modelling uncertainties through interval values.

  Consider the plant to be controlled is described by the nonlinear time-varying state space model [10]

$$\begin{aligned} x(k) &= f(x(k-1), u(k)p(k)) \\ y(k) &= g(x(k)), \end{aligned} \tag{2.17}$$

  where $u(k)$ is a vector of inputs or manipulated variables, $x(k)$ is a vector of state variables, $p(k)$ is a vector of uncertainties, and $y(k)$ is a vector of controlled variables or outputs.

  The problem to be solved at each sampling time may be stated as

$$\min_{u} \; J(u(k), y(k), w(k), \theta(k)), \tag{2.18}$$

  subject to

$$C_1(u(k)), C_2(y(k)), C_3(\theta(k)), \tag{2.19}$$

  where J is an objective function over a finite control horizon, $w(k)$ is the set point sequence, and $C_i(k)$ are sets of nonlinear constraints.

  Thus, the problem to be solved is defining $u(k)$ in an interval mathematics form so that it minimizes J while satisfying the sets of constraints $C_i(k)$.

- Guaranteed characterization .

  The aim is to achieve guaranteed characterization [43] of the set of all possible solutions in a nonlinear and discrete time context. A control sequence, v, of length m is feasible if

$$g(v) \cong f\,(\,f\,(\cdots(f(x(0), u(0)), \cdots), u(m-2)), u(m-1)) \;\in\; X_t. \tag{2.20}$$

Thus, the set of all feasible input sequences of length m is

$$V = g^{-1}(X_t), \tag{2.21}$$

where g^{-1} is the reciprocal function of g in a set-theoretic sense. If f is polynomial in x and u, then g is polynomial in v. When X_t is a singleton solving (2.21) for V, the problem is to find all the solutions required for a set of polynomial equations with several unknowns.

Methods of model conversion, from a continuous time uncertain system to an equivalent discrete time interval model, have not yet been thoroughly established. Shieh et al. [82] proposed an approach converting from a continuous to an enclosing discrete time interval model which used an interval arithmetic tool and its inclusion theorem to avoid numerical problems. They also focused on the digital redesign process consisting of converting a designed continuous time controller into an equivalent discrete time controller. This is not a trivial problem because hybrid control specifications for sampled data uncertain systems are difficult to predetermine and the closed loop inter-sampling behaviour is difficult to control. However, with this new method, they achieved very good results because a digital controller may need redesigning to modify the sampling period to achieve robust stability of the system.

In a further study, Shieh et al. [83] proposed a method whose function is exactly the inverse of the previous one [82], i.e., a method to convert a discrete time into an equivalent continuous time uncertain model to ensure that well-developed analog robust control techniques can be applied to the converted analog model for indirect robust control of sampled data uncertain systems. Interval analysis was also used for the construction of the proposed procedure based on an interval geometric series method.

The robust Schur stability problem is introduced in Section 2.2 of this thesis, as a problem of discrete time domain. An approach was proposed by Garloff et al. [24], in which they consider the robust Schur stability of polynomials with coefficients depending

polynomially on parameters varying in given intervals. They proposed an algorithm based on Bernstein expansion of the symmetric and anti-symmetric parts for the polynomial family to obtain verification of stability.

Many design problems, including control and signal processing, can be formulated within the framework of guaranteed tuning. Jaulin and Walter [42] considered guaranteed tuning and proposed a prototype numerical algorithm based on interval analysis. This was a very useful tool in this case, because the algorithm must characterize sets defined by inequalities, and for this function interval analysis is the best tool. Thus, these problems can be solved in a guaranteed way.

Because interval analysis is a useful tool to solve problems of set characterization involving optimization, the same authors[43] computed all the sequences of controls driving a deterministic nonlinear discrete time state space system from a given initial state to a given desired set of terminal states. Interval analysis provides guaranteed characterization of the set of all possible solutions in a nonlinear discrete time context. This method takes into account nonlinearities such as saturations and thresholds.

Concerning robustness in the discrete time domain, Vehi et al. [91] applied interval techniques to the analysis of robustness of predictive controllers. The basic tool was MIA [23]. The authors based their approach on the affirmation that checking the robustness of a predictive controller is equivalent to verifying the positiveness of the range of a set of functions. As some studies show (referenced in [91]), robust stability and most robust performance problems can be stated in this form. Therefore, methods to study the positiveness of functions are essential tools for robustness analysis. The positivity conditions can be expressed using modal intervals, but the preliminary step of presenting the uncertain domain as a suitable set of modal intervals is required. Formulating the problem in this way simplifies the evaluation of interval functions by taking advantage of monotonicity.

Model Predictive Control (MPC) is one of the most popular control strategies. An improved MPC strategy was proposed by Bravo et al. [10], called interval model pre-

dictive control (IMPC). This approach appeared to complete the MPC strategy in the case where the process or the constraints were nonlinear or the cost function was not quadratic. To solve these special cases, global optimization algorithms, which use interval analysis, were introduced as the best tool. Giving a plant described in the discrete domain, IMPC strategy can be used in a large number of cases, including linear and nonlinear models, linear and nonlinear constraints, quadratic or non-quadratic objective function(s), and bounded uncertainties, to name a few.

2.6 Other applications of intervals

Not all control problems concerned with robust control have been treated. Some other problems have also been analysed in the literature, showing how interval analysis can be a very useful tool.

As stated in Section 2.2, there are a few other problems that do not fit with any of the control subjects discussed up to now, but also use interval analysis. Estimation of unknown parameters of a model, and gain scheduling problems are two important subjects in that context. Rather than just theoretical problems, there are a number of practical examples of applied control problems (robotics) which show how interval analysis can be a very useful tool when applied.

2.6.1 Estimation

Bounded error parametric estimation approaches have been promoted over the last decade, largely because

- these approaches can address deterministic structural errors not adequately described by random variables, and
- they are well suited to the guaranteed characterization of parameter uncertainty.

Some researchers have based their approach to these control problems on applying interval techniques to obtain improvements over classical methods. For example, Walter and Pronzato [95] propose the criteria:

> Given a model structure, one problem is the choice of the criterion to be optimized in order to find the best model in the class be defined. That is to say, the criterion for this selection is the optimization of a scalar cost function $j(\boldsymbol{\rho})$ with respect to the model parameters p.

Some classic approaches for comparing parameter values are based on criteria including

- Least squares.
- Least modulus.
- Maximum likelihood.
- Maximum a posteriori (minimum risk).

Using any of these methods provides an estimator. An estimator is said to be robust if its performance does not deteriorate significantly when the base hypotheses are not completely satisfied. Some approaches to obtain robust estimators have been studied in [95].

Following the same line of study, Jaulin and Walter [40, 41, 44] followed this approach and cast nonlinear bounded error estimation into the framework of set inversion, and proposed a solution algorithm incorporating interval analysis. They expressed the problem as the estimation of unknown parameters of a model from experimental data collected on a system under known experimental conditions. Solving this problem as a set inversion problem is relatively easy when the model output depends linearly on the parameters to be estimated. However, the nonlinear case is much more complicated, and the tools provided by interval analysis seem very promising because they provide guaranteed global

results. Another important feature of interval analysis for set inversion is the capability of rapidly eliminating large portions of the parameter space before concentrating on the indeterminate region. Thus, the method is interesting as an initial procedure before using more local approaches. Set inversion via interval analysis (SIVIA) [44] was proposed as an adapted algorithm from [40], and characterizes the set of all values of the parameter vector to be estimated which in turn are consistent with the hypotheses:

- The error between the model output and output data which should lie between some prior bounds.
- The factors characterizing the experiments already carried out are uncertain, and should also lie between prior bounds.

Using this approach, convergence analysis shows that in almost any situation, the set of feasible parameter vectors can be characterized with accuracy that is only limited by the effect of rounding.

Jaulin et al.[38] applied interval analysis to bounded error parametric estimation for discrete event systems (DES), whose behaviours are governed by occurrences of different types of events rather than by clock ticks. Problems involving DES systems are generally nonlinear, non-convex and non-differentiable, so classic methods often fail to give reliable results. The aim of this approach was to show interval analysis could address these problems in a guaranteed global way, producing more reliable outcomes than classic methods.

In the same line, Kieffer et al. [45] proposed an approach addressing recursive nonlinear state estimation in the context of bounded state perturbation and measurement noise. A new state estimator was obtained which evaluated a set estimate guaranteed to contain all the values of the state consistent with the available observations, given the noise bounds, and a set containing the initial value of the state. The generation of this estimator was based on interval analysis and the concept of set inversion. As in classic Kalman filtering, this state estimator alternates prediction and correction based on the

SIVIA algorithm.

Another algorithm for parameter estimation was proposed by Feng et al. [19]. A recursive algorithm was presented to calculate axis aligned orthotopes, or boxes, which bound the set of feasible parameters. They showed that interval mathematics could provide an efficient tool to calculate these orthotopic bounds for each iteration, providing very accurate estimates.

Markov et al. [53] considered the problems of interpolation and curve fitting in the presence of unknown but bounded errors in the output measurements using generalized polynomials under bounded measurement uncertainties. They considered the task of finding the interpolation interval function, incorporating two constrained linear optimization problems, and obtained a modeling function that interpolated a set of m data.

2.6.2 Robotics

Remaining with estimation approaches, but applied them to robot localization, Kieffer et al. [46] introduced a methodology to compute a set guaranteed to contain all values of the parameter vector, and ensure that an upper bound on the number of fault tests is available. The estimator used was obtained from an adaptation of the SIVIA algorithm [45].

A typical problem of robot control, where interval analysis can be applied, is minimum time trajectory planning. This is a critical problem for automated industrial environments and space robotics, but has been studied by a few researchers.

Guarino et al. [28] attempted to compute optimal robot trajectory planning. Their approach combined stochastic optimization using a genetic algorithm, and deterministic optimization using an interval algorithm to derive a feasibly certain estimation of the global solution. Later papers from Piazzi et al. [76, 78] also considered the same planning problem, but used a global deterministic approach based on a procedure incorporating interval analysis tools to build a solution. In their first study [76], the aim was to compute the total traveling time required to perform the robotic task, stated as an

optimal trajectory planning problem with a minimum time criterion.
On the other hand, the second study [78] was concerned with jerk constraints, since joint position errors increase as the jerk increases, and to limit excessive wear on the robot and the excitation of resonances, thereby extending robot life span. In general, the minimum jerk is desirable to provide greater similarity to human joint movements. The resulting minimum jerk trajectory planning was shown to be a global constrained minimax optimization problem solved with an algorithm incorporating interval analysis. Two key outcomes can be observed from the prior research. Minimizing the maximum jerk in joint space has a beneficial effect in reducing the actuator and mechanical strain. This is because, considering the manipulator dynamics, the derivative of vector torque depends on the typically dominant term of the inertia matrix multiplied by the vector joint jerk. In addition, using trigonometric splines provides the advantage of allowing easy alteration of the planned trajectory in mid-course, if necessary, e.g. to avoid unexpected obstacles in real time environments.

Another basic robot control problem is computing the actuating torques required to make the robot follow a desired trajectory. Vehi et al. [90] considered take a non-holonomic mobile robot as a prototype and obtained a velocity control based on an estimated model. They proposed a method to design and implement an interval model based on a PI controller using MIA [23], which provided tools to solve the interval equations that appear during the design process and compute control laws as interval functions.

2.6.3 Gain Scheduling

Finally, we should consider the case of Gain Scheduling (GS) controller approaches which incorporate interval analysis. Gain scheduling compensators are normally used in closed loops to achieve good performance in spite of large parameter variations. Fadali et al. [17] used a linear time invariant (LTI) system and proposed a robust design synthesis approach based on the solution of a Diophantine equation. Interval analysis extended the synthesis

procedure proposed by Fadali et al. to systems containing uncertain transfer function coefficients. Consequently, a satisfactory controller or family of controllers for bounded parameter uncertainty were obtained. Following the same line of controller design, Fadali et al. [16] proposed a methodology to control nonlinear systems with slowly varying dynamics using gain scheduling. GS is a common design method for slowly varying systems which places the system poles near a desired location by designing controllers at a set of linearized operating points and linearly interpolating controller values between those operating points. Again, interval analysis was used to extend this approach.

McNichols and Fadali [55] used an interval arithmetic tool to determine intervals that restricted the closed loop poles of the system to regions around ideal transfer function coefficients. The aim of their design was to determine a minimal set of design points that connected with GS. Consequently, they obtained the ideal controller coefficients, which in turn placed the closed loop poles at their nominal design locations. This approach was applied for each sampled value of the scheduling variable.

2.7 Computer aided control system design for Robust Control

2.7.1 Parametric computer aided control system design

Control theory is widely used to describe, control, and optimize industrial processes. A large number of theoretical results have led to a variety of computational approaches and numerical algorithms to solve system analysis and design problems. Subsequently, these approaches have resulted in several generic computer aided control system design (CACSD) software packages to solve practical control problems. All CACSD packages have the common feature of providing tools to manage numerical computations. Most control software commercially available from such as the Numerical Algorithms Group, Ltd. (NAG) [59], and The Mathworks, Inc. [54] to name but two. However, many

research outcomes developing new algorithms and associated robust numerical CACSD software are freely available electronically, such as Scilab [81] from the Institute National de Recherche en Informatique et en Automatique (INRIA) and the SLICOT libraries [63] from the Numerics In Control Network (NICONET), which are available via ftp. Some well-known and useful control applications are embedded inside these control packages, such as the robust control toolbox [11] by Chiang and Safonov, and the μ -analysis and synthesis toolbox [6] by Balas et al.

Considering generic frames, there are other frames for more specific control problems. The study is particularly interested in tools for solving robust parametric control problems of ever increasing importance in control engineering. This importance has resulted in the emergence of different tools and software to assist control engineers.

We present a review of the existing tools, and concentrate on the yynomial toolbox, Paradise, Frequency Domain toolbox and Robust Control Synthesis Toolboxes. Our aim is to provide an overview of each, focusing on their main features, the format of their input parameters, and the type of algorithms used, including, in some cases, the functions provided.

2.7.2 Polynomial toolbox for Matlab

The polynomial toolbox 2.0 (Fig. 2-4) developed by the Czech Technical University in Prague is a registered trademark of The MathWorks, Inc. [87]. This is a package for systems, signals, and control analysis/design based on advanced polynomial methods.

This package offers all the tools needed for systems described by polynomial matrix fraction (PMF), a format commonly adopted in control problems to express transfer matrices. In addition, it provides a wide range of macros to test various robustness measures for systems with parametric uncertainties, including single parameter stability margins, interval polynomials, and polytopic uncertainties.

A number of classical design methods in the frequency domain are also provided and many other design routines can be developed based upon the polynomial matrix macros.

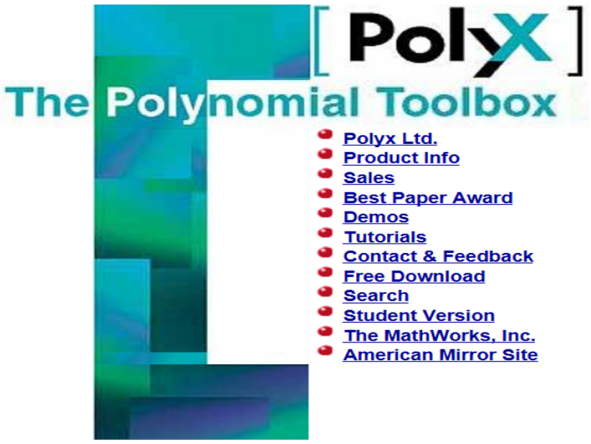


Figure 2-4: Polynomial toolbox [87]

The most important features of the toolbox are:

- Input, manipulation, and display of polynomials and polynomial matrices based on a polynomial matrix object.
- Solvers for numerous linear and quadratic matrix polynomial equations for overloaded operations and functions.
- Polynomial matrices with complex coefficients for applications in signal processing.
- Continuous time and discrete time systems and signal models based on polynomial matrix fractions.
- Classical and robustness analysis for LTI systems and filters.
- Conversion to and from LTI objects in the control system toolbox and polynomial objects defined in the symbolic math toolbox.
- A Simulink [54] block set for LTI systems described by polynomial matrix fractions.

The polynomial toolbox provides several routines to solve typical design tasks. Their modifications, as well as polynomial solutions for many other design problems, can be

built with the basic tools in the polynomial toolbox. It includes basic control routines to:

- Stabilize the plant and to parametrize all stabilizing controllers.
- Place closed loop poles by dynamic output feedback.
- Design deadbeat controllers for discrete-time systems.
- Implement H∞ optimization.
- Implement H2 optimization.
- Achieve robust control with parametric uncertainties: single parameter, interval polynomials and polytopic polynomials.
- Develop numerical methods for polynomial matrices. Some types of numerical techniques included in this toolbox can be mentioned here, such as: methods based on equating indeterminate coefficients, polynomial reduction based on elementary row and column operations, interpolation methods and state space methods.

2.7.3 Paradise toolbox for Matlab

The PArametric Robustness Analysis and Design Interactive Software Environment (Paradise) (Fig. 2-5) toolbox developed by the Institute of Robotics and Mechatronics of Germany[2, 84] is intended to assist the control engineer develop for robust parametric controls. The required input for the package is a Simulink model, which removes the requirement for a parametric plant model. From this graphical input, Paradise computes the symbolic closed loop model using the extended symbolic toolbox. This frees the control engineer from symbolic calculations, and offers him a graphical, user-friendly interface. Moreover, if Simulink is not available, the closed loop system equations (state space or transfer function representation) could be used as an input.

Three classes of parameters can be used in Paradise:

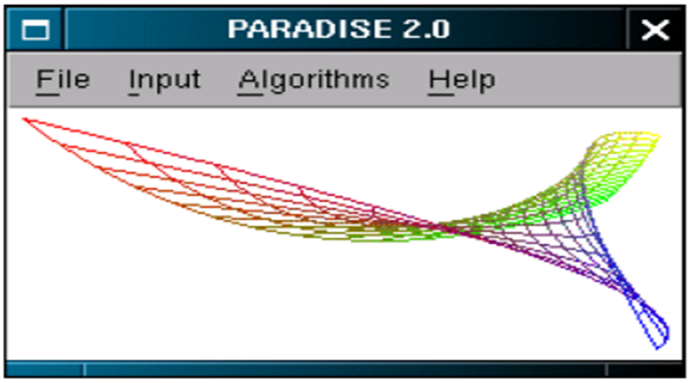


Figure 2-5: Paradise toolbox [2, 84]

1. Varying parameters. Plant parameters which are uncertain and assumed to vary within given intervals defined by upper and lower bounds.

2. Fixed parameters. Parameters assumed to be constant. Their initial value is zero.

3. Controller parameters. Parameters to be determined by the design process.

Some algorithms included in the Paradise toolbox are:

- Algorithms to determine the set of stabilizing parameters in a two-dimensional parameter space. These can be applied to different problems, such as the design of fixed gain or gain scheduled controllers, and robustness analysis.

- Algorithms for designing an invariant two-dimensional space. For a nominal plant, an $m-$dimensional cross-section in controller parameter space is determined so that only m eigenvalues are shifted while the remaining eigenvalues remain at their locations.

- Algorithms for constructing value sets and evaluation of tree structures. The use of a tree structure to build the value set allows one to handle systems with a large number of uncertain parameters, which can be analysed quickly.

- Algorithms for the construction of Popov sets. The Popov criterion then allows the engineer to check stability of some types of control loops.

- Algorithms for design by contraction of stability regions. In general, the set of $\Gamma-$stabilizing parameter vectors of a polynomial is bound by regions (hypersurfaces in the case of Hurwitz stability).

- Algorithms for calculating stability radius, defined as the smallest number, ρ, such that the polynomial, $p(s, q)$, is stable for all q with $\|q\| < \rho$.

Some additional features included in the toolbox are:

- A graphical editor for the construction of Γ-regions.

- A simplification of entries in case a term appears more than once in the system equations, specifying a substitute term in the Simulink model.

- Performance specifications for a specific control problem can be specified via the closed loop eigenvalue location.

2.7.4 Frequency Domain toolbox for Matlab

In contrast to the previous toolboxes, the frequency domain toolbox developed by the University of Linz (Austria) [31, 33] uses interval arithmetic for frequency domain analysis and design of control systems. Numerical algorithms are based on polynomial or transfer matrices, unlike other toolboxes in Matlab , which use numerical algorithms based on state space methods.

Numerical problems are avoided by using polynomial arithmetic based on interval arithmetic. To develop a numerical toolbox based on frequency domain methods, polynomials are a reasonable basic data type to be used. The toolbox deals with polynomials of the type $p = \sum_i a_i s^i, \quad a_i \in \mathbb{R}$ represented by intervals of type *rfloat*, the basic data type provided by interval arithmetic. Taking the data type *polynom* as the development base,

the toolbox includes a class structure implemented in C++ including rational transfer functions, polynomial and transfer matrices, with operators and overloading functions.

Polynomial arithmetic not only provides a numerical method for the implementation of frequency domain methods, it also increases accuracy by using a variable length of the mantissa of the bounds. This length can be changed dynamically, increasing accuracy and automatically verifying the results.

The most interesting algorithms included in this polynomial toolbox, on which most controller design algorithms in the frequency domain are based, are:

- Coprime factorization.

  Each transfer matrix, $P(s)$, can be expressed as the ratio, $P(s) = N(s)D^{-1}(s)$, of two right coprime polynomial matrices, $N(s), D(s)$. This can be solved using a canonical form of transfer matrices called the Smith-McMillan form.

- Spectral and inner-outer factorization.

  This can be solved using a canonical form of polynomial matrices, called the Smith form and an iterative algorithm for the spectral factorization of a polynomial.

- Solution of Diophantine equations.

  The task is to compute the solution, $X(s)$ and $Y(s)$, of polynomial equations such as $Z(s) = X(s)N(s) + Y(s)D(s)$, in which $D(s), N(s)$, and $Z(s)$ are given.

2.7.5 Robust Control Synthesis toolbox for Matlab

Parameter space approaches based on symbolic quantifier elimination (QE), have been proposed for these types of problems. QE based methods provide an organized approach to addressing the parameter space of fixed structure robust controller synthesis problems [37, 36, 79]. The QE-based approach can uniformly deal with a lot of important design specificatons such as stability margin(gain/phase), stabilty radius. The toolbox offers to control engineers a graphical interface to achieve multi-objective robust controller design

smoothly. The toolbox implements a method of parameter space deign for robust control synthesis which guarantees the real stability radius specification using QE.

2.8 Summary

Some key approaches concerned with robust control were surveyed to illuminate the advantages provided by interval analysis. Interval analysis is a very useful tool that allows avoidance of numerical problems. In addition, it can be very helpful for problems of set characterization involving optimization, nonlinear inequalities and quantifiers, due to its ability to produce guaranteed results even in a nonlinear context.

Interval analysis was also shown to be very powerful in bounding the ranges of functions efficiently while providing mathematically rigorous results. This capability is especially welcome in robust control since a variety of analysis and design problems can be cast in the evaluation of the range of functions over intervals. The rising importance of this tool is shown by the number of approaches being developed to achieve a perfect match with the corresponding problem, including interval analysis variants, such as PIA, NPIA, MIA, etc. We note the lack of approaches concerned with design. Most works are focused on the space state approach, which encourages this research approach, because there remains a lot of work to do.

Considering the exposed parametric CACSDs, all of them solve robust parametric control problems and have specific data structures, which are transparent for the user. The CACSDs all assist control engineers in a user-friendly manner, but the polynomial and Paradise toolboxes offer a more attractive input environment than the frequency domain toolbox. In spite, frequency domain toolbox is the only one which uses interval arithmetic to address the real numbers represented by intervals.

The useful and clever approach of Paradise is to allow direct Simulink input. In contrast, the engineer must obtain the model before using the other toolboxes, a task which not can always be performed easily due to the symbolic calculations required.

Chapter 3

Robust Control via Modal Interval Analysis

Modal Interval Analysis (MIA) is introduced as an extension of interval analysis, and its application to the field of robust control is discussed.

3.1 Introduction

In general, classical methods employed to deal with interval models use interval analysis. Techniques and methods of robust control, for example, have been implemented using interval analysis, as discussed in Chapter 2. MIA is an extension of interval analysis, and provides an essential tool to implement the methods and algorithms included in the proposed package. MIA provides an improvement over interval analysis because it simplifies interval function computation and allows semantic interpretation of the outcomes. Some applications of MIA to robustness analysis and to the design of robust controllers are presented, and the achieved improvements discussed.

3.2 Modal Interval Analysis

MIA [23, 22, 86] extends real numbers to intervals, identifying the intervals by the predicates the real numbers fulfill, whereas classical interval analysis identifies the intervals with the set of real numbers they contain.

Given the set of closed intervals of $\mathbb{R}$, $I(\mathbb{R}) = \{[a,b]' \mid a,b \in \mathbb{R}, \quad a \leq b\}$, and the set of logical existential and universal quantifiers, $\{E,U\}$, a modal interval is the pair

$$X := (X', QX), \tag{3.1}$$

where $X' \in I(\mathbb{R})$ is the extension and $QX \in \{E,U\}$ is the modality. The set of modal intervals are denoted by $I^*(\mathbb{R})$. A modal interval, $([a_1,a_2]',E)$, is called an existential or proper interval, and $([a_2,a_1]',U)$ is called a universal or improper interval.

The modal quantifier, Q connects every real predicate $P(.) \in \text{Pred}()$ with a unique interval predicate, i.e., for a variable $x \in \mathbb{R}$ and a modal interval $(A^{'}, QA) \in I^*(\mathbb{R})$,

$$Q(x,(A',QA)) := QA(x,A'). \tag{3.2}$$

The set of real predicates accepted by a modal interval is

$$\text{Pred}((A^{'},QA)) := \{P(.) \in \text{Pred}(\mathbb{R}) \mid Q(x,(A^{'},QA))P(x)\}, \tag{3.3}$$

and inclusion for modal intervals can be introduced in a similar way as for classic intervals: for $A,B \in I^*(\mathbb{R})$,

$$A \subseteq B \iff \text{Pred}(A) \subseteq \text{Pred}(B). \tag{3.4}$$

The modal interval can be stated in canonical notation as

$$[a,b] := \left\{ \begin{array}{lll} ([a,b]',E) & \text{if} & a \leq b \\ ([b,a]',U) & \text{if} & a \geq b \end{array} \right\}, \tag{3.5}$$

and the inclusion is then characterized by

$$[a_1, b_1] \subseteq [a_2, b_2] \Longleftrightarrow (\ a_1 \geq a_2,\ b_1 \leq b_2). \tag{3.6}$$

It is also possible to define the set of predicates rejected by a modal interval in an alternate, equivalent, manner as

$$\text{Copred}((A^{'}, QA)) := \{P(.) \in \text{Copred}(\mathbb{R}) \mid \quad \bar{Q}(x, (A^{'}, QA))P(x)\}. \tag{3.7}$$

There is complementarity between Pred and Copred through the duality operator,

$$\text{Dual}([a_1, a_2]) = [a_2, a_1], \tag{3.8}$$

and

$$A \subseteq B \Longleftrightarrow \text{Dual}(A) \supseteq \text{Dual}(B) \Longleftrightarrow \text{Copred}(A) \supseteq \text{Copred}(B). \tag{3.9}$$

The structure $(I^* (\mathbb{R}), \subseteq)$ is a lattice and the minimum and maximum for a family of modal intervals, $A(i) = \left[\underline{a\,(i)}, \overline{a\,(i)}\right],\ i \in I$, are called meet and join, respectively, where:

- Meet: $\wedge\,(i, I)\ A\,(i) = \left[\max\limits_{i \in I} \underline{a\,(i)}, \min\limits_{i \in I} \overline{a\,(i)}\right]$.
- Join: $\vee\,(i, I)\ A\,(i) = \left[\min\limits_{i \in I} \underline{a\,(i)}, \max\limits_{i \in I} \overline{a\,(i)}\right]$.

The dual formulation of the modal intervals allows the definition of two semantic interval functions, f^* and f^{**}, which have a very important role in the theory, because they are closely related to the modal interval extensions and provide meanings to the interval computations.

Definition 1 (* and ** semantic functions) *If f is an $\mathbb{R}^n$ to $\mathbb{R}$ continuous function*

and $A \in I^*(\mathbb{R}^n)$*, then*

$$\begin{aligned} f^*(A) &= \vee\left(a_p, A'_p\right) \wedge (a_i, A'_i)\left[f(a_p, a_i), f(a_p, a_i)\right] = \\ &= \left[\min_{a_p \in A'_p} \max_{a_i \in A'_i} f(a_p, a_i), \max_{a_p \in A'_p} \min_{a_i \in A'_i} f(a_p, a_i)\right], \end{aligned} \tag{3.10}$$

$$\begin{aligned} f^{**}(A) &= \wedge(a_i, A'_i) \vee \left(a_p, A'_p\right)\left[f(a_p, a_i), f(a_p, a_i)\right] = \\ &= \left[\max_{a_i \in A'_i} \min_{a_p \in A'_p} f(a_p, a_i), \min_{a_i \in A'_i} \max_{a_p \in, A'_p} f(a_p, a_i)\right], \end{aligned} \tag{3.11}$$

where $a = (a_p, a_i)$ *is the component split corresponding to* $A = (A_p, A_i)$*, with* A_p *a subvector containing the proper components of* A*; and* A_i *a subvector containing the improper components of* A*.*

The following theorem expresses the meaning of the interval results f^* and f^{**}.

Theorem 2 (f^* semantic theorem) *Given a modal interval vector* $A \in I^*(\mathbb{R}^n)$*, a function* $f : \mathbb{R}^n \to \mathbb{R}$ *continuous in* A'*, and a modal interval* $F(A) \in I^*(\mathbb{R})$*,*

$$f^*(A) \subseteq F(A) \Leftrightarrow U(a_p, A'_p) Q(z, F(A)) E(a_i, A'_i)$$

$$\text{so that} \quad z = f(a_p, a_i). \tag{3.12}$$

The following theorem establishes a dual semantic for proper and improper modal intervals.

Theorem 3 (f^{**} semantic theorem) *Given a modal interval vector* $A \in I^*(\mathbb{R}^n)$*, a function* $f : \mathbb{R}^n \to \mathbb{R}$ *continuous in* A'*, and a modal interval* $F(A) \in I^*(\mathbb{R})$*,*

$$f^{**}(A) \supseteq F(A) \Leftrightarrow U(a_i, A'_i) Q(z, \text{Dual}(F(A))) E(a_p, A'_p)$$

$$\text{so that} \quad z = f(a_p, a_i). \tag{3.13}$$

Generally, computing the semantic extensions f^* and f^{**} is a difficult challenge. When the continuous function f is rational, it can be operationally extended to a modal rational function using the syntactical tree of the expression of the function, where the real operators are transformed into their * or ** semantic extensions. Both semantic extensions are equal for any class of operators. In this case there is a modal rational function, $fR(A)$, associated with the syntactical tree of f where the real operators are transformed into their semantic extensions. However, $fR(A)$ is not interpretable. The interpretation problem for a modal rational function consists of relating it to the corresponding semantic functions which have standard meanings defined by the semantic theorems.

On the other hand the interpretable rational interval program, $fR(A)$, may nevertheless result in a loss of information far more important than that produced by numerical rounding. Thus, it is very important to determine criteria to characterize the rational interval functions. Fundamentally, $fR(A)$ with an ideal computation (infinite precision) has the property that

$$f^*(A) = fR(A) = f^{**}(A), \tag{3.14}$$

and in this case, $fR(.)$ is said to be optimal for A.

There are several coercion and partial coercion theorems which characterize the optimality of a modal rational function according to its monotonicity. These theorems are demonstrations of the construction shown in this section as well as other recent research on MIA related to rational functions [86, 91].

3.3 Robust control

3.3.1 Introduction

Effective control of practical time varying systems with parametric uncertainties and external disturbances is a main focus in the design of robust control systems.

The usual objective in robust control of interval processes is to achieve stability and robustness of the closed loop system based upon some suitable trade-off. In general, robust control of uncertain systems is achieved using fixed nonlinear feedback control functions which operate effectively over a specified wide range of class of system parameter variations (i.e., parametric intervals). This deterministic approach contrasts sharply with many other adaptive control schemes where online identification and global parameter convergence properties are required, and statistical information must be provided to yield the desired robust dynamic behaviour.

3.3.2 Robustness analysis

Checking the robustness of a controller is equivalent to verifying the positiveness of the range of a set of functions [5, 50]. Therefore, methods used to study the positiveness of functions are essential tools for robustness analysis [92, 91].

Definition 2 *$F(\boldsymbol{q})$ is positive over $\boldsymbol{Q}$ if $F(\boldsymbol{q}) > 0$ for all $\boldsymbol{q} \in \boldsymbol{Q}$.*

To formulate the positivity conditions using modal intervals, the uncertainty domain should be represented as a suitable set of modal intervals. An uncertainty domain means that every parameter q_i has an unknown value between $\underline{q_i}$ and $\overline{q_i}$. Thus, every uncertain parameter must be considered as a proper or existential modal interval,

$$q_i = ([\underline{q_i}, \overline{q_i}], E) \Rightarrow q_i = [\underline{q_i}, \overline{q_i}], \quad \underline{q_i} \leq \overline{q_i}. \tag{3.15}$$

From Definition 2, the semantic interpretations of f^* and f^{**} are very closely related to the concept of positivity of the range [88]. Thus, * and **semantics can be used for testing positivity non-positivity, respectively, and optimality implies necessary and sufficient positivity conditions. Outer approximations of f^* will yield sufficient positivity conditions and necessary conditions will be obtained by inner approximations of f^{**}.

We will also show that the application of Theorem 3 provides for faster and simpler computations.

A straightforward translation of Theorems 2 and 3 to the interval function by substituting proper modal intervals for the uncertain parameters provides the following theorem.

Theorem 4 *Consider the function $F(\boldsymbol{q})$, dependent on an uncertain parameter vector, $\boldsymbol{q}$, belonging to an uncertainty domain, $\boldsymbol{Q}$. The function $F(\boldsymbol{q})$ is positive over $\boldsymbol{Q}$ if, and only if, $F^*(\boldsymbol{Q}) > 0$.*

Since the * and ** extensions are not always calculable, some properties of modal intervals applied to rational functions must be employed in order to work with computable functions. Theorem 4 may be reformulated for modal rational extensions as follows.

Theorem 5 *Let $FR(\boldsymbol{Q})$ be an optimal modal rational extension of $F(\boldsymbol{q})$, then$F(\boldsymbol{q})$ is positive over $\boldsymbol{Q}$ if, and only if, $FR(\boldsymbol{Q}) > 0$.*

The key problem is finding such an optimal extension. As shown in section 3.2, optimality is closely related to monotonicity. When $F(\boldsymbol{q})$ is monotonic for all or some of its variables, MIA provides conditions of optimality, or at least interpretability, of $FR(\boldsymbol{Q})$.

If $F(\boldsymbol{q})$ is uniformly monotonic for each variable and for all its incidences, then the coercion theorem can be applied to obtain an optimal extension.

The following examples consider the robust stability of uncertain polynomials.

EXAMPLE 1:

Consider the family of polynomials $\mathcal{P}$ described by

$$
\begin{aligned}
p(s,q) &= a_0(q) + a_1(q)s + a_2(q)s^2 + a_3(q)s^3 + s^4, \\
a_0 &= 1 - 3q_1^2q_2^2, \\
a_1 &= 6 + 6q_1 - 8q_2, \\
a_2 &= 6 + 3q_1q_2 - 4q_2, \\
a_3 &= 5 + 0.2q_1q_2 + 0.1q_1 - 0.1q_2,
\end{aligned}
$$

with uncertainty bounds $0 \leq q_i \leq 0.5$ for $i = 1, 2$. The objective is to determine whether $\mathcal{P}$ is robustly stable.

The critical stability condition is obtained from the third Hurwitz determinant:

$$\begin{aligned} F(\boldsymbol{q}) = & -266.6q_2 - 3.2q_2^3 + 8.8q_1q_2^3 + .03q_1^4q_2^2 + 1.8q_1^3q_2 \\ & + 69.5q_1^2q_2^2 + 6.6q_1^3q_2^2 - 7.8q_1^2q_2^3 + 5.9q_1^3q_2^3 + .1q_1^4q_2^3 \\ & + .12q_1^4q_2^4 + 96.5q_1^2q_2 - 130.5q_1q_2^2 + 103.1q_2^2 + 119 \\ & + 110.6q_1 + 60.4q_1q_2 - 32.4q_1^2 - .1q_1^3q_2^4 + .03q_2^4q_1^2. \end{aligned}$$

Applying classical interval arithmetic, inclusions of the range of this function can be obtained, for example, by using the natural extension
$FR([0, 0.5], [0, 0.5]) = [-39.36, 232.59]$. As the range is overbound and contains zero, this computation does not provide a conclusion regarding positivity. Applying the coercion theorem, it is possible to obtain an optimal expression for $F(\boldsymbol{q})$, if it is monotonic with respect to each variable q_i. This may be checked by computing the partial derivatives of the Hurwitz determinant following Theorem 4,

$$\begin{aligned} \left(\frac{\partial F}{\partial q_1}\right)^*([0, 0.5],[0, 0.5]) &\subseteq [44.5693686, 210.066391], \\ \left(\frac{\partial F}{\partial q_2}\right)^*([0, 0.5],[0, 0.5]) &\subseteq [-335.74993, -86.74788]. \end{aligned}$$

From Theorem 2, these inclusions mean that for every q_1 and q_2 in $[0, 0.5]$, the values of both derivatives do not change their sign, so $F(\boldsymbol{q})$ is uniformly monotonic in both variables. Moreover, $F(\boldsymbol{q})$ is also monotonic for every incidence of q_1 and q_2 and it is possible to apply coercion theorem to evaluate the optimal modal rational extension of

$F(\boldsymbol{q})$,

$$\begin{aligned}
FR(\boldsymbol{Q}D) =& 0.03Q_1^4\text{Dual}(Q_2)^2+69.56Q_1^2\text{Dual}(Q_2)^2\\
&+1.8Q_1^3\text{Dual}(Q_2)+8.8Q_1\text{Dual}(Q_2)^3+110.6Q_1\\
&+0.12Q_1^4\text{Dual}(Q_2)^3+0.03\text{Dual}(Q_2)^4Q_1^2-3.2Q_2^3\\
&+96.56Q_1^2\text{Dual}(Q_2)-130.56\text{Dual}(Q_1)Q_2^2+119\\
&+60.42Q_1\text{Dual}(Q_2)-0.12\text{Dual}(Q_1)^3Q_2^4\\
&+5.9Q_1^3\text{Dual}(Q_2)^3-266.6Q_2+0.12Q_1^4\text{Dual}(Q_2)^4\\
&+6Q_1^3\text{Dual}(Q_2)^2-32\text{Dual}(Q_1)^2-7\text{Dual}(Q_1)^2Q_2^3\\
&+103.19\text{Dual}(Q_2)^2|_{Q_i=[0,0.5]}=[11.0975, 166.1975].
\end{aligned}$$

The resulting interval is the optimal solution for $FR(Q_i = [0, 0.5])$, and is positive. Hence, $F(\boldsymbol{q})$ is positive, and the polynomial family $\mathcal{P}$ is robustly stable.

Obviously, bounds of a monotonic function in all variables can be computed without using intervals. This example was provided to illustrate modal intervals, and the application of the coercion theorem for proper arguments.

Generally, without monotonicity it is not possible to compute the exact result. The usual case is a function which is totally monotonic for only some of its variables.

EXAMPLE 2:

Consider the family of polynomials $\mathcal{P}$ described by

$$\begin{aligned}
p(s, \boldsymbol{q}) &= 1 - 3q_1^2q_2^2 + (1 + 2q_3 - q_2)s\\
&\quad +(6 - 3q_1q_2 - 4q_2q_4^2)s^2 + s^3,
\end{aligned}$$

with uncertainty bounds $0 \leq q_i \leq 0.5$. for $i = 1, ..., 4$. The objective is to determine whether $\mathcal{P}$ is robustly stable.

The critical stability condition is given by the determinant

$$\begin{aligned} F(\boldsymbol{q}) = 5 &+ 12q_3 - 6q_2 - 3q_1q_2 - 8q_2q_4^2q_3 \\ &+3q_1^2q_2^2 - 6q_1q_2q_3 + 3q_1q_2^2 - 4q_2q_4^2 + 4q_2^2q_4^2. \end{aligned}$$

Computing the partial derivatives, Theorem 3 shows that

$$\begin{aligned} &\left(\frac{\partial F(\boldsymbol{q})}{\partial q_1}\right)^*([0{,}0.5],[0{,}0.5],[0{,}0.5],[0{,}0.5]) \subseteq [-3{,}1.5], \\ &\left(\frac{\partial F(\boldsymbol{q})}{\partial q_2}\right)^*([0{,}0.5],[0{,}0.5],[0{,}0.5],[0{,}0.5]) \subseteq [-11{,}-2.75], \\ &\left(\frac{\partial F(\boldsymbol{q})}{\partial q_3}\right)^*([0{,}0.5],[0{,}0.5],[0{,}0.5],[0{,}0.5]) \subseteq [9.5{,}12], \\ &\left(\frac{\partial F(\boldsymbol{q})}{\partial q_4}\right)^*([0{,}0.5],[0{,}0.5],[0{,}0.5],[0{,}0.5]) \subseteq [-4{,}1]. \end{aligned}$$

In this case, from Theorem 2 and the inclusions above, uniform monotonicity can only be assured in q_2 and q_3.

Applying the partial coercion theorem for q_2 and q_3, one gets a sub-optimal form of $FR(\boldsymbol{Q})$,

$$\begin{aligned} FR(\boldsymbol{Q}DT^*) = 5 &+ 12Q_3 - 6Q_2 - 8Q_2Q_4^2\text{Dual}(Q_3) \\ &-6Q_1Q_2\text{Dual}(Q_3) + 3Q_1[\text{Dual}(Q_2)]^2 - 4Q_2Q_4^2 \\ &+4[\text{Dual}(Q_2)]^2Q_4^2 + 3Q_1^2[\text{Dual}(Q_2)]^2 - 3Q_1Q_2. \end{aligned}$$

For $Q_i = [0, 0.5]$, $\text{Dual}(Q_2) = [0.5, 0]$ and $\text{Dual}(Q_3) = [0.5, 0]$, the computed range is $[0.75, 11]$, which is positive. Hence, $F(\boldsymbol{q})$ is positive and the polynomial family $\mathcal{P}$ is robustly stable.

Remark: The same bounds can be computed using FR_l and FR_u, obtained by replacing q_3 with $\underline{q_3}$ in FR_l and $\overline{q_3}$ in FR_u and vice versa for q_2 [34]. In this case, two interval functions must be computed. However, using the MIA formalism, only one

interval function is computed. This result is more relevant in the context of splitting domain algorithms so that when k monotonicities are found, partial coercion theorem converts the initial function of l variables to a single, simpler, interval function of the $l-k$ variables.

When $F(\boldsymbol{q})$ is not totally monotonic for any variable, Theorems 3 and 4 can be applied to obtain the following general conditions.

Theorem 6 *$F(\boldsymbol{q})$ is positive over $\boldsymbol{Q}$ if $FR(\boldsymbol{Q}T^{*}) > 0$.*

Theorem 6 provides only a sufficient positivity condition. To implement a suitable positivity testing algorithm, certain necessary conditions are needed, as provide in the following.

Theorem 7 *Consider the function $F(\boldsymbol{q})$ as described in Theorem 6. Let $FR^{k}(\boldsymbol{Q}T^{**})$, $k = 1, \ldots, N$ be the set of the N possible results obtained by transforming, for every multi-incident parameter, all incidences but one into its dual.*

The function $F(\boldsymbol{q})$ is non-positive over $\boldsymbol{Q}$ if the lower bound of the interval $\left[\underline{FR^{k}(\boldsymbol{Q}T^{**})}, \overline{FR^{k}(\boldsymbol{Q}T^{**})}\right]$ is not positive for some k.

To illustrate the importance of Theorem 7, we will analyse the interpretation provided by Theorem 3 applied to the result of Theorem 4, according to the modality of the result.

Suppose that $FR^{k}(\boldsymbol{Q}T^{**})$ is proper. Thus, for every value, c, in the interval $\left[\underline{FR^{k}(\boldsymbol{Q}T^{**})}, \overline{FR^{k}(\boldsymbol{Q}T^{**})}\right]$, there is a point $\boldsymbol{q} \in \boldsymbol{Q}$ such that $F^{k}(\boldsymbol{q}) = c$, i.e., every negative or zero value of the result corresponds to an unstable case.

Suppose now that $FR^{k}(\boldsymbol{Q}T^{**})$ is improper. The semantic interpretation is that there is a value $c \in \left[\underline{FR^{k}(\boldsymbol{Q}T^{**})}, \overline{FR^{k}(\boldsymbol{Q}T^{**})}\right]$ and a point $\boldsymbol{q} \in \boldsymbol{Q}$ such that $F^{k}(\boldsymbol{q}) = c$. Because $FR^{k}(\boldsymbol{Q}T^{**})$ is improper, $\underline{FR^{k}(\boldsymbol{Q}T^{**})} \leq 0$ implies $\overline{FR^{k}(\boldsymbol{Q}T^{**})} \leq 0$, so $c \leq 0$ and at least one unstable case exists.

3.3.3 Robust control design

The results obtained in Section 3.2 can be combined in an algorithm to check the performance for a given set, $\boldsymbol{Q}$. The implementation splits $\boldsymbol{Q}$ into sub-boxes, distinguishing three types of regions according to their monotonicity:

1. *Regions with uniform monotonicity for all variables*: none of the ranges for partial derivatives contain zero. Necessary and sufficient positivity conditions can be checked using Theorem 5.

2. *Regions with uniform monotonicity for k variables*: k partial derivatives do not contain zero. The partial coercion theorem is applied to the monotonic variables. The original function of l variables is converted into a function of $l - k$ variables.

3. *Regions without monotonic variables*: all of the ranges for partial derivatives contain zero. Sufficient and necessary conditions for positivity are provided by Theorem 6 and Theorem 7, respectively.

To analyse the monotonicity of each box, the range of every first order partial derivative is computed using classic interval methods, and positivity and non-positivity checked for each case.

Since most performance specifications can be stated as a set of inequalities to be satisfied, $F_i(q) > 0;\ i = 1, \ldots, l$, the following robustness problems, which are standard in the analysis and control design systems, can be addressed by means of this algorithm.

1. *Robustness checking*: Given a perturbed plant $G(z^{-1}, \boldsymbol{q})$, and an uncertainty set $\boldsymbol{Q}$, check whether the controlled system achieves robust performances:

$$F_i(\boldsymbol{q}) > 0;\ \forall \boldsymbol{q} \in \boldsymbol{Q};\ i = 1, \ldots, l.$$

2. *Robust margin assessment*: Given a nominal plant, $G(z^{-1}, \boldsymbol{q}^0)$, find the larger set

$\boldsymbol{Q}_m$ around $\boldsymbol{q}^0$ such that the designed controller achieves robust performances:

$$\boldsymbol{Q}_m = \max\{\boldsymbol{Q} | F_i(\boldsymbol{q}) > 0;\ \forall \boldsymbol{q} \in \boldsymbol{Q};\ i = 1, \ldots, l\}.$$

3. *Robust control design*: Assuming a controller structure, find a fixed controller $C(z^{-1}, \boldsymbol{k}^0)$ so the closed loop controlled system achieves robust performances:

$$F_i(\boldsymbol{q}, \boldsymbol{k}^0) > 0;\ \forall \boldsymbol{q} \in \boldsymbol{Q};\ i = 1, \ldots, l.$$

4. *Robust $\boldsymbol{K}$ set estimation*: Given a perturbed plant, a specification set and a controller structure, find the largest set $\boldsymbol{K}$ such that $C(z^{-1}, \boldsymbol{k})$ achieves robust performances:

$$F_i(\boldsymbol{q}, \boldsymbol{k}) > 0;\ \forall \boldsymbol{q} \in \boldsymbol{Q},\ \boldsymbol{k} \in \boldsymbol{K};\ i = 1, \ldots, l.$$

3.4 Summary

In this chapter we introduced MIA in the field of robust control. The set of the modal intervals is an extension of the set of classical intervals. An important characteristic of modal intervals is that intervals are not referred to a set of values but to the properties that the set fulfills. This provides association with one of two quantifiers: universal or existential, introducing a dual formulation and semantic interpretation to the outcomes.

Semantic theorems to continuous functions provide criteria to interpret computational outcomes, and determine how to round the results.

Finally, MIA for rational functions provides methods to compute the exact range of a function, or, in defect, external and internal rounding.

Chapter 4

Parametric solver based on modal interval analysis

An important contribution of the thesis is presented consisting of the development of a solver incorporating MIA. The solver is based on an algorithm that checks the positivity of a function as the criterion to classify the regions of the user defined parameter space.

4.1 Introduction

MIA provides a number of advantages, as discussed in Chapter 3, the most important of which is that it provides guaranteed solutions. In this chapter we propose a solver based on MIA as an original outcome the thesis.

To build the MIA solver, we employed a modular design structure based on the development of

- new methods,
- a set of algorithms, and
- a set of tools.

The resultant solver is reliable and allows the control engineer to find guaranteed results with no explicit knowledge of intervals or modal intervals.

4.2 Methods and algorithms

The first step to build the solver is to develop a set of MIA based methods [86] to manage robust control problems with parametric uncertainties. The starting point is the requirement of interval and extensive symbolic numeric computations. Existing methods summarized in Section 2.2 have several limitations. Most studies concerned with robustness tests are based on extreme point results. This is a problem because the main concept is to extract properties (for example stability) for a family of plants from a finite subset of polynomials, and the size of the input set is not sufficient in all the cases. Another problem is that existing methods are very conservative and are generally based on analytic results which provide sufficient but not necessary conditions for robust stability. To solve or improve on these problems amongst others, it is necessary to develop new methods. Our MIA based proposal (chapter 3) addresses the most common robust control problems. Its semantic interpretation, and existential ($\exists$) and universal ($\forall$) solutions offer a high guarantee of achieving a solid base to build the algorithms and tools of the new robust control toolbox. The goal is to build a set of functions using MIA as a numerical method that will compare favourably to the classical methods.

The methods detailed above and also in Chapter 3, involve building a set of algorithms also based on MIA. They also consider the operators ease of use issues for symbolic, numeric, and interval computations. Incorporating MIA produces simple, efficient, and fast algorithms. Simple algorithms allows them to be shorter and more versatile, and efficiency is essential in achieving a reliable framework. Speed is always desirable, because almost all problems to be solved will involve iterations to achieve the outcomes. For example, if the problem consists of checking a closed loop system for stability, it will

be necessary to execute the same algorithm along the whole range of variations of the uncertain parameters of the plant. Hence, fast algorithms are essential from the point of view of execution time when assessing parametric uncertain systems.

An assessment of monotonicity is part of MIA, and this will influences the overall computation speed to a large degree since it will allow directly discarding iterations without needing to completely analyse the case. The MIA based algorithm is significantly simplified since they incorporate many improvements to achieve the optimal result.

Another aspect to be considered in developing the algorithms is the stop condition. MIA solves this point by incorporating coercion theorems [86], which provide necessary and sufficient stop conditions for the algorithms. Moreover, lineal operations and the f^{**} function(theorem 3) also provide stop conditions for the algorithms.

The most important algorithms developed are summarized as follows.

- Single controller design specification (SCDS).

  In the case we have a single controller design specification, this algorithm graphically shows three types of regions in the parameter space: the region that fulfills the proposed specification; the region does not achieve the specification; and the undetermined region, where it is not possible to say anything. We have developed this algorithm based on an algorithm developed by J.Vehí [88]. The new algorithm is independent of C++, but still requires Maple [52] to input the functions (specifications) to be checked. Thus, at the moment the algorithm could be considered to be semi-automated.

- Single square stability (SSS).

  As already commented above, parametrically uncertain systems have a range of parameters. If all the possible values of the parameters are shown as little squares, this algorithm would take into account only one single square. This algorithm checks the stability of a single parameter over its full range.

- Find specified controller (FSC).

Given a single specification, the concept is to find the set of controllers that fulfills it. The algorithm has different aspects, for example, it may be applied to an individual parameter range (or ranges), rather than applying it to the full parameter space.

- n-functions version (NFV).

  The concept is to solve the robust control design problem using an improved strategy that checks not only an individual function (specification), but also checks the entire set of n functions (specifications).

- Controller parameter space points (CPSP).

  This centres the computations and checks a single specific point in the parameter space. During the controller design stage, this algorithm provides greater speed in solving the proposed robust control problems, since rather than considering the complete parameter space, it considers only an individual point inside each square, e.g. the center point, or a point in a specific, user defined, location, which corresponds to using a grid.

The preceding work of J.Vehí [88] has implemented some of the proposed algorithms. The original contribution of the developed solver is to automate the algorithms, which entails designing completely new algorithms with completely new data structures. Aspects this automation include integrating interval and extensive symbolic numeric computation, making the use of Maple and C++ transparent to the user, and conforming all the inputs (functions, controller parameters, system parameters, etc.) into the same format (using, for example, Matlab).

Table 4.1 summarizes the algorithms, showing the features:

- The number of functions (or specifications) the algorithm evaluates.
- The scope of application of the algorithm (the strategy), i.e., if and when the algorithm is applied to the full parameter space or over a more reduced area.
- The control action, i.e., the goal of the algorithm.

Algorithm	NoFs	Scope	Control action	Current state
SCDS	1	Full P.S.	Control design	Semi-automated
NFV	n	Full P.S.	Control design	full-automated
FSC	1	Full P.S.	Set control design	full-automated
SSS	1	Single square of P.S.	Stability check	full-automated
CPSP	1	Grid	Control design	full-automated

[1]P.S.: Parameter Space
[2]NoFs: Number of Functions

Table 4.1: Algorithms

- The current state, level of automation.

4.3 Parametric solver for single design specifications

The solver built can manage multiple design specifications, It incorporates sorting techniques to check the specifications from the easier computational cost to the higher. The specifications are tested as single specifications using this sorting in order to accelerate the computation. Therefore the solver developed is for single design specifications.

SCDS is proposed as the prototype solver. Its routines are largely dedicated to solving a specific control problem defined by the user. Depending on the control problem, the solver requires different specific operations. To accommodate this, the algorithms incorporate different programming strategies, i.e., the same problem can be solved following several strategies. In the SCDS algorithm, a single function is evaluated, whereas the NFV algorithm evaluates n functions at the same time. It is useful to offer different strategies and let the control engineer decide which algorithms to use depending on the specific type of problem.

The SCDS algorithm can be considered as the second generation to the original algorithm [88] which provided a useful tool to compute some specifications of an uncertain system, but had the drawback of requiring manual inputs for execution. Fully automated tools require a completely new design, structurally and functionally, of the algorithm, and

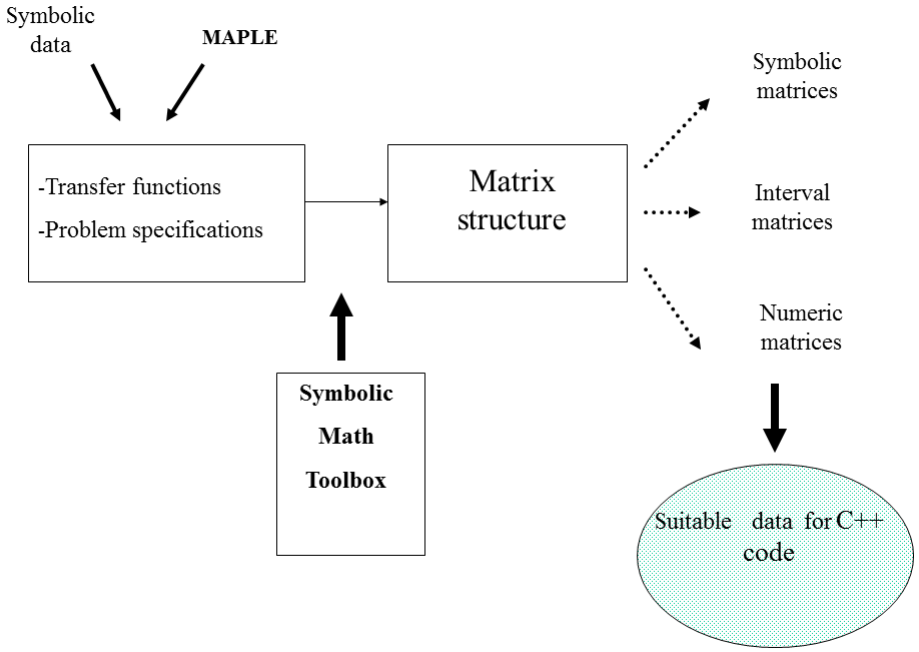


Figure 4-1: Symbolic methodology

the SCDS algorithm incorporates only the concept of building a set of tools to solve robust control problems using modal interval methods from the preceding algorithm.

Symbolic algorithms

The symbolic environment permits conversion of a control problem into a set of matrices suitable for use by Matlab. The global process is shown in Fig. 4-1.

An uncertain plant and a control design problem can usually be defined in terms of transfer functions and design specifications, respectively. Control specifications, such as absolute stability, velocity error, control effort, and resonance peak, can be expressed as functions of variables, with feasible values between lower and upper bounds. The main problem is how to pass the symbolic information contained in these functions to the C++ routine, SCDS, etc.; to check these conditions; and to provide information about the actual regions of the full parameter space. For the SCDS routine, these checks are

made over a single specification. This has some advantages, because the control engineer can obtain information about the role of each specification individually, and thus put stronger constrictions on the corresponding function, if necessary, to obtain a better controller. However, in further work, an extended version of the SCDS algorithm (NFV) will check all specifications at the same time over the full parameter space to achieve a quicker and more efficient algorithm. Two steps are needed to achieve the proposed aim:

1. Place the functions into the simplest and most suitable form.

   The math package Maple V [52] is used to obtain the functions. This provides many tools to obtain the functions in their most simplified form, as well as following a specific structure. The function must be expressed as a set of addends, each containing a product of its coefficient and a number of variables with its exponent. The final form of a specification might be (for example):

   $$f_i = a_1 X_0^2 X_1^3 X_2 X_3 + a_2 X_2 X_3^2 + a_3 X_1$$

   As commented above, all the variables of these functions are symbols, some of them have values only inside the C++ routine where they vary over the full parameter space, and others have a fixed interval value. To obtain a graphic control interpreted result, and thus help the control engineers in their design tasks, it is usual to limit the number of variables to two or three which vary over the full parameter space, and fix all the others. In the examples used in the SDCS algorithm, only two parameters were allowed to vary over the parameter space to obtain an easily interpretable result on a two-dimensional plane.

2. Obtain a set of matrices suitable to be passed to C++ routines.

   Once the functions are in simplified form, it remains to convert all the information to be passed to the C++ routines. The Matlab symbolic math toolbox, has its a function *sym* that provides a symbolic representation of the input scalar or matrix. Using this tool, all the data derived from the problem functions, including the

symbolic data, are transformed to a matrix structure which can then be passed to the C++ routine for checking. The resultant matrix structure, discussed in Section 4.3, includes three types of matrices: symbolic, interval and numeric, in a suitable form for C++ code.

Matrix structure

The symbolic matrix provided by the Matlab symbolic math toolbox [4] for each specification function contains all the information, and has the structure

$$\begin{bmatrix} & X_0 & X_1 & ... & .. & X_n \\ a_0 & e_{00} & e_{01} & ... & ... & e_{0n} \\ a_1 & e_{10} & e_{11} & ... & ... & e_{1n} \\ ... & ... & ... & ... & ... & ... \\ ... & ... & ... & ... & ... & ... \\ a_m & e_{m0} & e_{m1} & ... & ... & e_{mn} \end{bmatrix},$$

where X_i are the variables (symbols), a_j are the coefficient values (numeric) of the summand j, and e_{ij} are the exponent values (numeric) that the variable i has in the summand j. The number of variables is n and the number of summands m.

We employ Maple to convert the specification data into a matrix whose size depends on the number of variables of the function and the number of summands, i.e., an $m \times n$ matrix, and this matrix is used inside the C++ algorithms to reconstruct the original function and compute its value over the range of operation, and build the matrices corresponding to the first and second order derivatives.

For the SCDS algorithm, three smaller matrices are passed as input parameters rather than the larger matrix, avoid working with large matrices in C++, which is not explicitly devised to manage matrix elements (as is Matlab), and thereby make programming in C++ easier and clearer. Also, independent matrices for independent concepts produces simpler and more understandable code.

We employ Matlab to convert the large matrix into three matrices or vectors with structures:

- Vector of variables.

  A symbolic column vector of dimension n, consisting of the names (symbols) of the variables contained in the function to be checked, with structure

$$\begin{bmatrix} conj(X_1) \\ conj(X_2) \\ ... \\ ... \\ conj(X_n) \end{bmatrix},$$

  where $conj$ denotes that variables $X_1, X_2, ...$ are symbolic.

- Vector of coefficients.

  A column vector of dimension m, consisting of the numerical elements corresponding to the coefficients of each summand of the function to be evaluated, with structure

$$\begin{bmatrix} a_1 \\ a_2 \\ ... \\ ... \\ a_m \end{bmatrix}.$$

- Matrix of exponents.

  An $m \times n$ numeric matrix where the e_{ij} element is the exponent of variable i in the

addend position j, with structure

$$\begin{bmatrix} e_{00} & e_{01} & \dots & \dots & e_{0n} \\ e_{10} & e_{11} & \dots & \dots & e_{1n} \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ e_{m0} & e_{m1} & \dots & \dots & e_{mn} \end{bmatrix}$$

In appendix A is presented the essential routines of the algorithm SCDS as sample of the developed programs.

4.4 Development of new tools

The diversity of data (symbolic, numeric, and interval) results in complex algorithms and processes, as discussed above. To avoid discouraging control engineers from using the algorithms, we developed a set of tools which together form the framework IRCAD framework, an original outcome of this thesis.

The tools were created to make interval methods more convenient for control engineers. It is essential to create a package that can be used by engineers with little or no knowledge of intervals. These tools and the IRCAD framework are presented in Chapter 5.

4.5 Summary

A solver is presented in this chapter as the first contribution of this work . It is based on MIA and has been built integrating a diversity of routines though to solve specific robust control problems defined by the users. The building trade of this solver involves the generation of routines designed to integrate symbolic, interval and numerical data in a transparent way for the control engineer. Constraints related to these diversity of data,

results on the use of suitable software packages that fit with the aim of the routines (for example to have faster processes is used C++). In addition It has also been exposed how the interactions between the different kind of data are processed using matrices (Maple package is used in this case). To end this summary is significant to stand out the use of different strategies to solve robust control problems.

Chapter 5

Parametric framework for robust control

We present the second contribution of this thesis: the parametric framework interval robust control for analysis and design (IRCAD). This framework uses the proposed MIA solver to solve robust control, analysis and design, problems of intervalar systems.

5.1 Introduction

The analysis and design of robust controllers has been one of the major outcomes of linear control theory, as discussed above. A common feature of most advanced robust control algorithms is that the parameters are posed in terms of an uncertainty description. Solving such problems that involve uncertain parameters is difficult for control engineers.

In Chapter 2, we presented an overview of approaches for applications of interval analysis to robust control. In most cases, the studies focused on a single problem. However, in practice there are numerous problems that vary widely. Thus there are different sets of tools to solve each problem and the control engineer must be familiar with a large number of different techniques and methods to be able to apply the most adequate solution to the specific problem being considered. Using such a wide set of tools can be complicated

because the problem type has a significant impact on how to input the functions, input uncertain parameters, write control structures, as well as interpret the results.

To cope with this diversity, we propose the construction of a set of interval based tools [20], or a toolbox for robust control following the concepts of the parametric CACSD, described in Chapter 2.

The main contribution of this work is the development of a common methodology which will allow the control engineer to solve control problems based on MIA.

A framework has been developed to isolate interval theory from control theory. The development of the framework was broken into two steps: first to develop the set of new tools, and second integrate the tools into a user-friendly framework. This framework provides the user (usually a control engineer) an interface to solve robust control problems that involve interval parameters, i.e., problems where the solution requires interval analysis, where the interval analysis methods are transparent to the user.

5.2 General description of the framework

5.2.1 Menu composition

The main window of the toolbox is shown in Fig. 5-1.

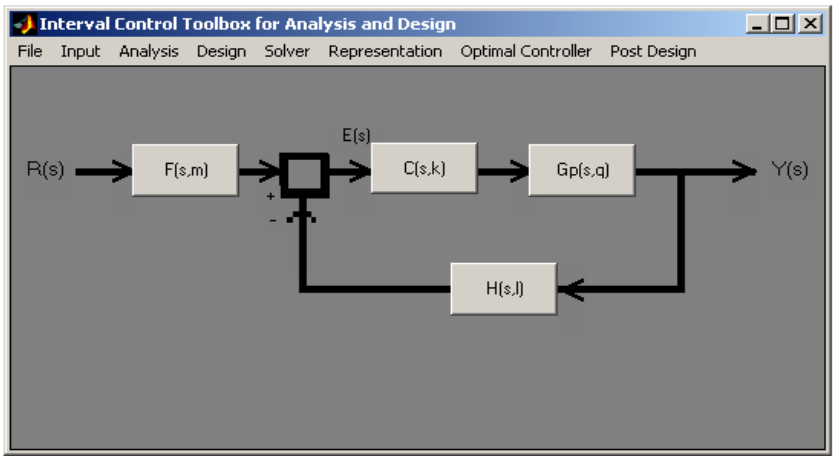


Figure 5-1: IRCAD framework

The framework application consists of a pop-up menu with eight menu options:

- File
- Input
- Analysis
- Design
- Solver
- Representation
- Optimal controller
- Post Design

Some of these options are focused on input of the data of a control system into the framework, others on representation of the framework outcomes, and there are specific tools to solve robust control problems. The IRCAD framework also includes a set of post design tools that allows the user to verify that the controller proposed by the framework fulfills the design specifications without needing to leave the framework.

5.2.2 Integration of numeric, interval and symbolic computation

In contrast to current parametric CACSDs, the proposed toolbox incorporates specific tools for robust control functions developed using MIA, which provides a number of advantages for the reliability of the result (Chapter 3) but has the drawback of needing to be able to deal with a diversity of data. A schematic of the toolbox is shown in Fig. 5-2, detailing the different tools included.

We employed C++ to develop procedures for numeric interval computation and incorporated modal interval arithmetic libraries [86] as required. C++ also provided tools

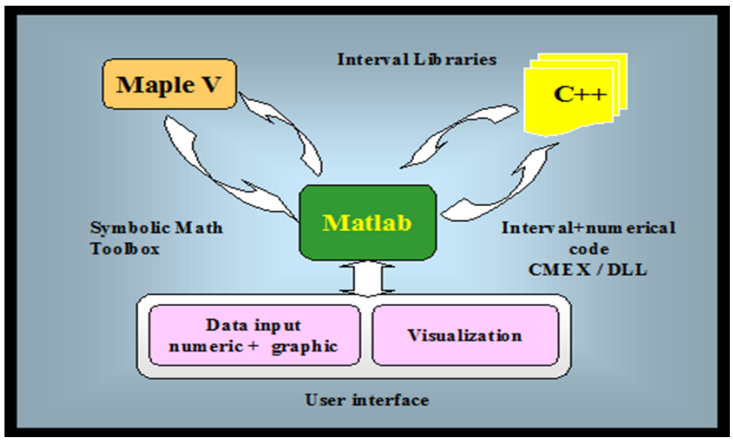


Figure 5-2: IRCAD. Data integration

to interface with Matlab toolboxes using DLL and MEX functions. The Maple package was employed to allow symbolic data input, such as transfer functions or problem specifications. Matlab allows these symbolic inputs through the Symbolic Math toolbox, and provides integration of numeric, interval, and symbolic data, transforming them into a suitable format for, and passing them to, C++ routines, and finally collecting the returned data. The last step, the user interface block in Fig. 5-2, incorporates showing the results in a way suitable for control applications using the GUI tools of Matlab.

Clearly, Matlab is the center of the proposed environment, and most were developed in that environment, due to Matlab has:

- high computational power for problems that involve matrix data;
- facility to deal with high level languages; and
- ease of connection with other commercial packages, such as Maple.

The IRCAD framework solves robust control problems. Thus, generally all problems will require a high level of computation. This aspect would be a bottleneck if we only solved them using Matlab scripts. However, for efficiency, computation routines were

implemented in C++. This allows inclusion of tools from interval analysis libraries [23] and provides procedures for numeric interval computation that execute significantly faster than the same outcomes can be achieved within Matlab. The high level C++ functions were compiled using Borland, obtaining a set of DLL functions, which may be accessed from within Matlab using CMEX within suitable Matlab commands and/or scripts.

5.2.3 Data input

The starting point to use the framework is to input the transfer function of the plant. Since we consider uncertain systems, the plant is defined as a family of transfer functions. To input them into the framework, the user defines a transfer function with undefined parameters and defines their uncertain values range (interval values).

This may be achieved by loading a previously saved control system, or can be defined directly as a transfer function for a family of plants, defining the system, $G(s, \mathbf{q})$ with the uncertain parameter $\mathbf{q} = [q_1, q_2]$.

The data input menus are:

- File: Allows studying any example using different formats (Fig. 5-3):
  - 'New input': Build a new problem in the framework.
  - 'Load workspace' Load a previous created problem.
  - 'Initialise workspace: Clear all the current variables located in the workspace.
  - 'Quit': Exit from the framework and optionally save the created example using the Matlab option 'save workspace'.
- Input menu: Enter data represented in the control block diagram: intervalar transfer functions, and their interval values (Fig. 5-4). The transfer functions that include the standard block diagram are:
  - Pre-filter F(s, $\mathbf{m}$),

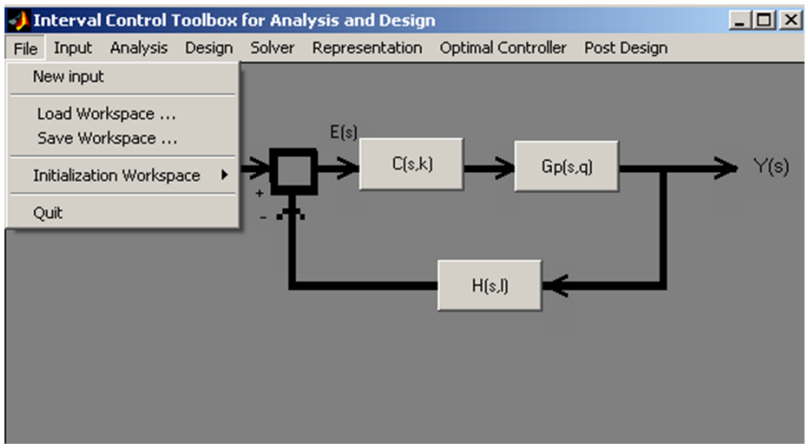


Figure 5-3: IRCAD. File menu.

– Controller C(s, **k**),

– Plant G(s, **q**), and

– Sensor H(s, **l**),

where **m**, **k**, **q**, **l** have interval values.

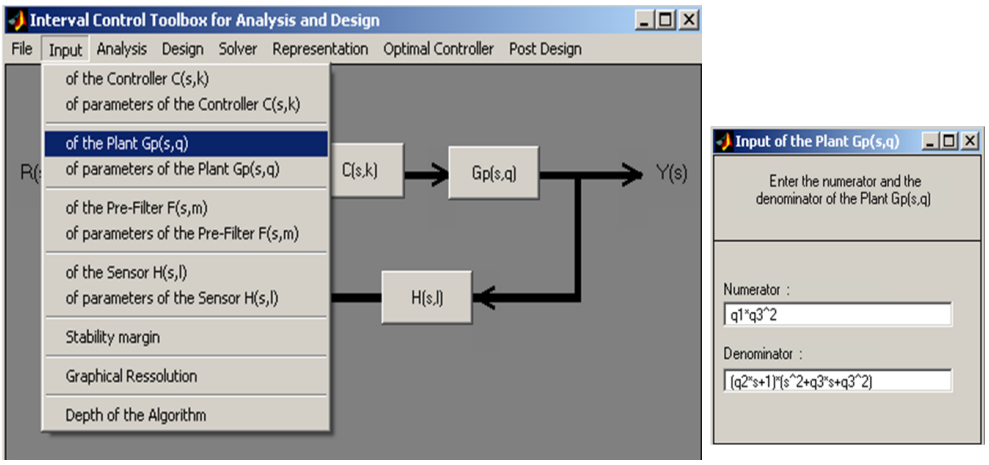


Figure 5-4: IRCAD. Data menu.

It is also possible to enter configuration parameters from this menu for the solver

and other algorithms included on the framework. These options are:

- The coefficient to compute the stability margin. This is the starting point of the computation when the option of computing the stability margin is selected.
- The desired graphical resolution for x and y axes to represent the feasible controllers on the parametric space.
- Depth of the algorithm to cut the branch of the solver tree and accelerate the computation (Chapter 4). If the depth is not limited, the solver may never finalise an undetermined region an require high computation to obtain a result in a reasonable period of time.

All the blocks of the control block diagram. Pre-filter, Sensor, and Plant are input in this manner. The Controller block can be similarly input or input directly by accepting the proposed controller from the optimal controller tool (Fig. 5-5)

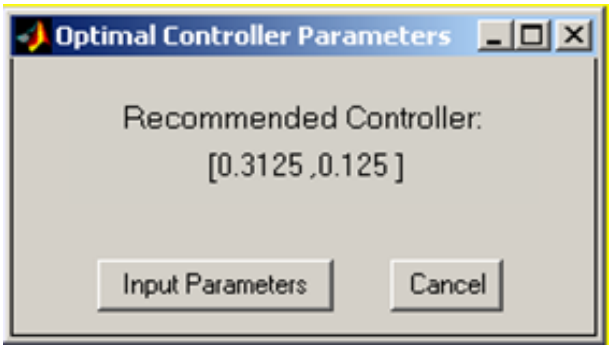


Figure 5-5: IRCAD. Input of the recommended controller

- Solver. Allows selection of the desired solver. The framework is built in a flexible manner, and allows the use of any solver chosen by the user (Fig. 5-6). In the examples presented for this thesis, we used the solver designed in Chapter 4.

5.2.4 Representation of the results

For any problem, it is essential to analyse the results obtained. The control design computation produces a classification of the parameter space in a set of intervalar controllers

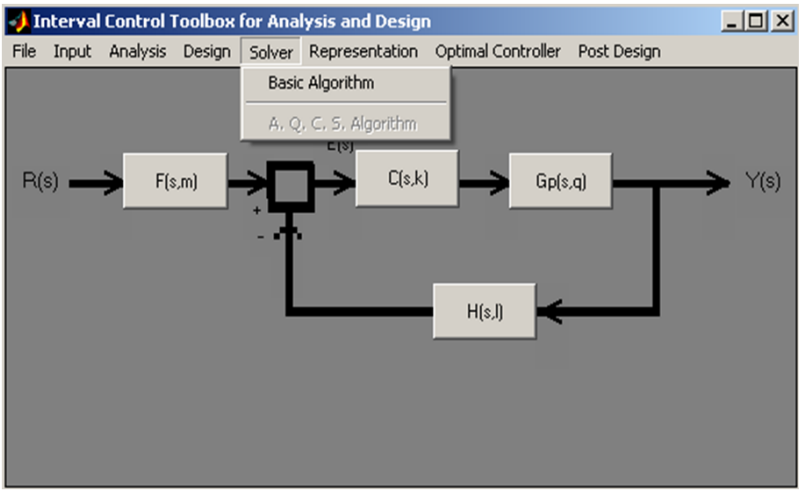


Figure 5-6: IRCAD. Solver menu.

that fulfill all the selected design specifications, another region of intervalar controllers that don't fulfill any or all the specifications, and a region classified as undetermined because the problem required high computation effort and the branch and bound algorithm was stopped before its state was determined. The framework provides to methods to visualize the controllers (Fig 5-7):

- Graphical (Colormap): A two-dimensional representation of the parameter space. Assuming a two parameter controller (as a PI controller), x axis shows one of the controller parameters and the y axis the other. Each (x,y) pair is represented as a colored square corresponding with the solver result, where the colors are chosen as follows:
  - Red denotes a region where the controller fulfills the specifications.
  - Yellow denotes a region where the controller does not fulfill the specifications
  - Blue denotes a region where the controller outcome is undetermined.
- Numeric: A set of intervals. The list of the intervalar pairs from the parameter

space classified by the solver as controllers that fulfills the specifications are reported in a table.

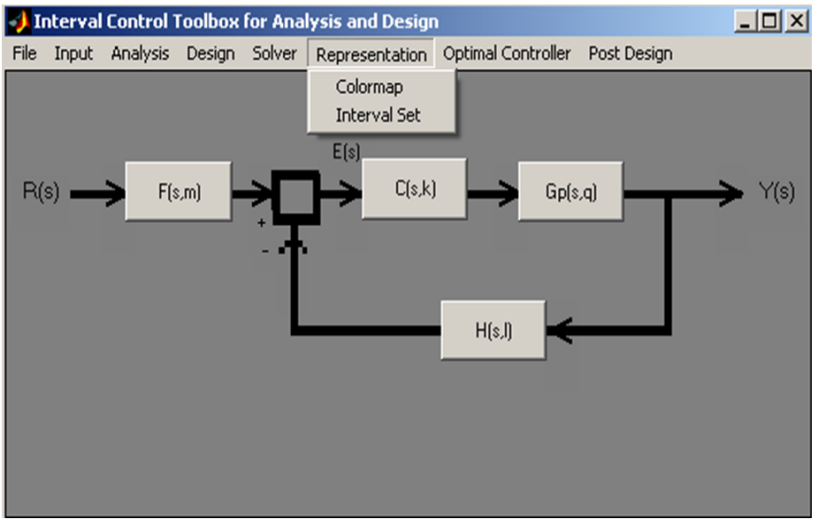


Figure 5-7: IRCAD. Representation of the results.

5.3 New tools

The robust control tools of the IRCAD framework can be categorised as three sets of problems.

1. Robust analysis problems. Problems such as testing the stability of a system to find the stability margin.

2. Robust design problems. The main tool is an application that allows selection of the design specifications the control system must achieve.

3. Post design tools. These allow a selected controller to be applied to the system without leaving the IRCAD framework.

5.3.1 Tools for robust analysis

The IRCAD framework includes a set of tools to solve robust analysis problems. The Analysis menu includes the following functions for an uncertain system with the transfer function defined as a family of plants q_i (Fig 5-8):

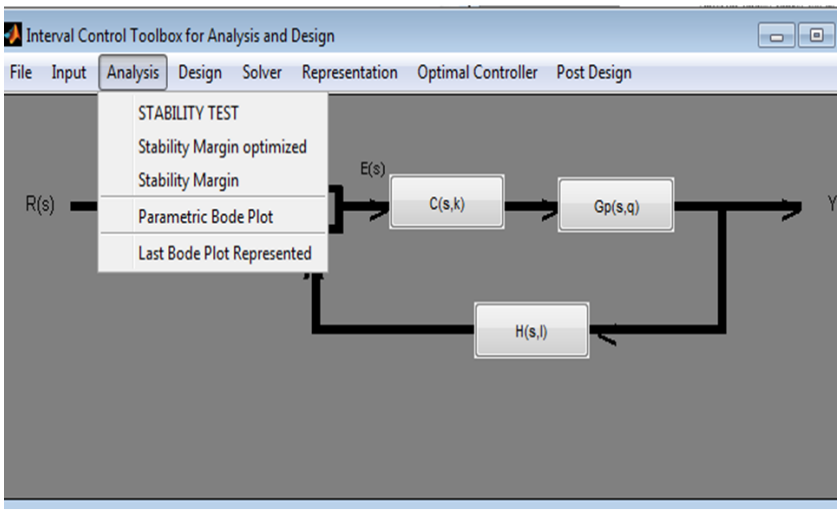


Figure 5-8: IRCAD. Analysis menu

- Stability test. To calculate the absolute stability region, we employ a MIA based branch and bound algorithm. The tool shows the stable and unstable regions. Regions where the depth of the algorithms was limited (to accelerate computation) can appear as undefined. If there are no computation time issue, e.g. the computer has high computation resources, it is possible to omit limiting the depth of search and then only stable and unstable regions result.

- Stability margin computation. The parametric stability margin is defined as the length of the smallest perturbation, Δq, which destabilizes the closed loop. This is a quantitative measure of the robustness of the closed loop system with respect to parametric uncertainty evaluated at the nominal point q^0.

In control problems, the characteristic polynomial coefficients generally do not perturb independently. At the first level of detail every feedback system is composed of at least two subsystems, controller and plant, connected in a feedback loop. The characteristic polynomial coefficients of such a system are functions of plant and controller parameters. Both parameters influence the coefficients, but their natures are quite different.

- The plant contains parameters that are subject to uncontrolled variations depending on the physical operating conditions, disturbances, modeling errors, etc.
- The controller parameters are often fixed during the operation of the system.

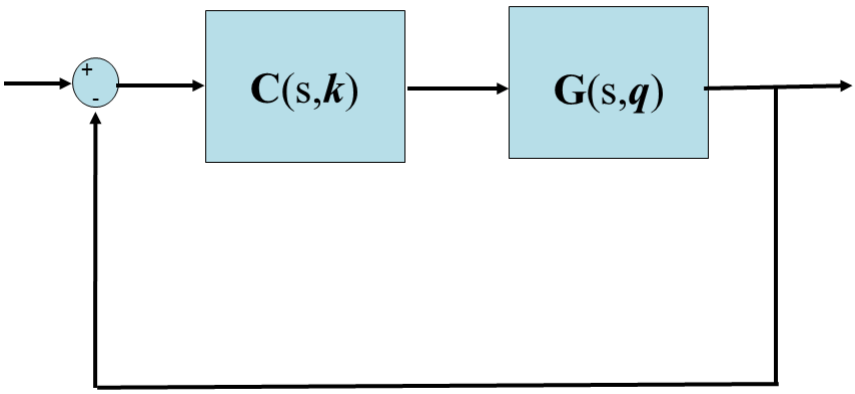


Figure 5-9: Standard feedback system

Consider a standard feedback system, as shown in Fig. 5-9, where the plant transfer function includes that the real parameter vector $\mathbf{q}$ belongs to the plant transfer function, and the controller is characterized by the real vector $\mathbf{k}$.

Problem statement Suppose that $\mathbf{q}^0 =$ is the nominal value of the plant parameter vector $\mathbf{q}$, $K^0$is a fixed controller, $\mathbf{K}^0$ is a parameter of controller K^0 that stabilizes the nominal plant, $G(s, q^0)$ and $\Delta p = p - p^0$ is a perturbation of the plant parameter vector from the nominal $\mathbf{q}^0$.

Then we define the *Parametric stability margin* as a bound on the size of Δq for guaranteed loop stability. It is useful as it provides us a region parameter space over which the parameters can freely vary without destroying closed loop stability. This is especially useful in controller design as a means of comparing the performance of proposed controllers, and is used as a quantitative measure of the robustness of the closed system with respect to parametric uncertainty evaluated at the nominal point q^0.

- Stability margin optimized. An improved computation of the stability margin that provides the minimum and maximum value rather than only a point value.

- Parametric Bode plots. From frequency domain analysis methods, one might conjecture that it is not necessary to check all frequencies. Thus, it is common to construct Bode plots using only a grid of w values. However, cases of singular frequencies occur, which require special attention in all frequency domain methods for robustness analysis.

  Problem statement. When an uncertain polynomial, p(s,$\mathbf{q}$), is evaluated at a frequency w, it equals a complex number
  $p(jw,\ \mathbf{q})$=$h(-w^2,\mathbf{q}) + jw.g(w^2,q)$. Separating the real and imaginary parts,

$$h(-w^2,\mathbf{q}) = a_0(\mathbf{q}) - a_2(\mathbf{q})w^2 + a_4(q)w^4 - ... \tag{5.1}$$

$$wg(-w^2,q) = a_1(q)w - a_3(q)w^3 + a_5(q)w^5 - ... \tag{5.2}$$

  Associated with these functions is a Jacobian matrix,

$$\mathrm{J(w,\mathbf{q})} = \begin{pmatrix} \frac{\delta h(-w^2,\mathbf{q})}{\delta q_1} & \frac{\delta h(-w^2,\mathbf{q})}{\delta q_2} & \cdots & \frac{\delta h(-w^2,\mathbf{q})}{\delta q_l} \\ \frac{\delta wg(-w^2,\mathbf{q})}{\delta q_1} & \frac{\delta wg(-w^2,\mathbf{q})}{\delta q_2} & \cdots & \frac{\delta wg(-w^2,\mathbf{q})}{\delta q_l} \end{pmatrix}.$$

  The Jacobian and the real and imaginary parts are employed to provide the algebraic definition of singular frequencies.

Definition. The nonnegative frequency, w_s is a singular frequency of the uncertain polynomial, $p(s, \mathbf{q})$, if there exists a $\mathbf{q}^0 \in \Re^l$ such that the three following conditions are simultaneously satisfied.

$$h(-w_s^2, \mathbf{q}^0) = 0, \tag{5.3}$$

$$w_s g(-w_s^2, \mathbf{q}^0) = 0, \tag{5.4}$$

and

$$rank[\mathbf{J}(w_s, q^0)] < 2. \tag{5.5}$$

5.3.2 Tools for robust design

The IRCAD framework has a powerful set of tools to design controllers for parametric control systems. The menu allows selection of one or more than one specification (Fig. 5-10) for an uncertain system with transfer function defined as a family of plants q_i.

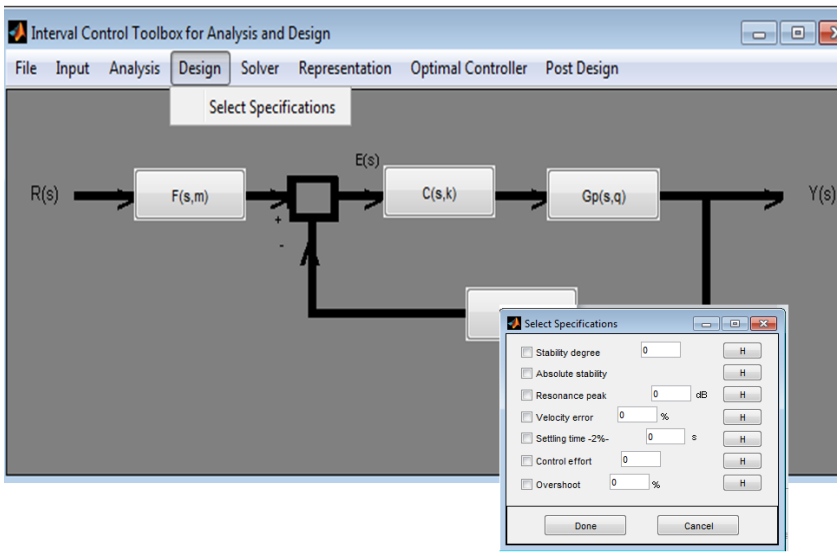


Figure 5-10: IRCAD. Design menu

The specifications incorporate the temporal and frequency domains and include: stability degree, absolute stability, resonance peak, velocity error, settling time, control

effort, and overshoot.

Once the specifications have been selected, they are translated to an interval function, converted into matrix structures and passed to the (C++) solver. The outcomes are returned in matrix format as a set of controllers that fulfill the selected specification or set of specifications. The controllers are a set of interval values. Using Matlab GUI tools, the resultant matrix, which contains all the information about the valid controllers, is passed to the user.

The framework allows the user to choose the format to show the set of valid controllers. It is possible to visualise them graphically or numerically. The decision often depends on the number of parameters selected for the controller. If the controller contains two or less parameters, graphical format might be the most appropriate, and the parameter space will be ranged by the parameters of the controller. Valid controllers will be shown as a red region (Fig. 5-11), unfeasible controllers as a yellow region, and undefined regions as blue. If the user chose the numeric format, then the framework produces a table containing the set of intervals for $k1, k2, \ldots, ki$ that allows the system to fulfill all the specifications.

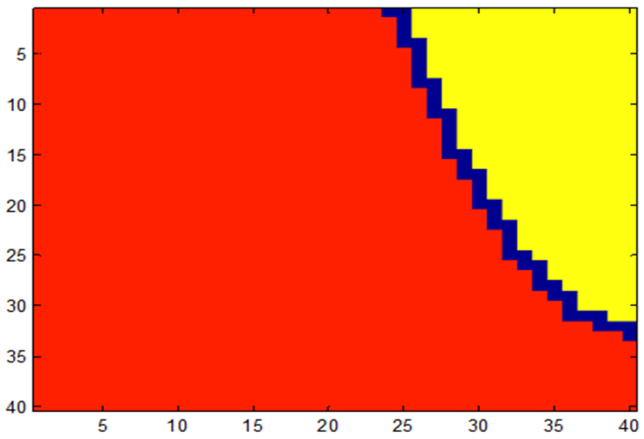


Figure 5-11: Example of a parameter space classified by the IRCAD design tool

- Overshoot

The Matlab routine **Oversho.m** computes the function (or constraint) to be tested by the solver.

Given the transfer function for the plant, $G(s, \mathbf{q})$, the aim is to design a controller, $C(s, \mathbf{k})$, that fulfills a step response with an overshoot equal or less than an specific value (OV).

1. Compute the characteristic polynomial $1 + G(s, \mathbf{q})C(s, \mathbf{k})$.
2. Compute the damping ratio ξ,

$$\xi = \sqrt{\frac{\ln(\frac{OV}{100})^2}{\ln(\frac{OV}{100})^2 + \pi^2}}.$$

3. Find the lines of constant damping. A feasible parametrization by the parameter α was proposed by Ackermann [1] in example 9.1, page 235,

$$s = \alpha + j\frac{\alpha\sqrt{1-\xi^2}}{\xi}.$$

4. Substitute the variable **s** of the characteristic polynomial with the above expression of s.
5. Putting the resultant equation in complex format (Re + j Im), the condition to be tested is

$$p_1(\alpha, q, k) + j \quad p_2(\alpha, q, k) \neq 0.$$

   CONDITION: This will be true if $p_1$and p_2 have no common roots.

6. TEST:
   (a) Remove α in the equation $p_2(\alpha, q, k)$.
   (b) Compute the determinant of the resultant matrix using $p_1(\alpha, q.k)$ and the new p_2.
   (c) If the determinant of the resultant matrix is different from zero then $p_1$and p_2 have no common roots, and so the specification of overshoot is fulfilled

and the determinant of the resultant matrix is assigned to the function to be returned.

(d) Return the function to be tested.

- Absolute stability in the feedback loop

  The Matlab routine **EstaAbs.m** computes the function (or constraint) to be tested by the solver.

  Given the transfer function for the plant, $G(s, \mathbf{q})$, the aim is to design a controller, $C(s, \mathbf{k})$, that obtains a step response that fulfills the specification of absolute stability.

  1. Compute the feedback transfer function,

  $$M(s) = \frac{G(s)C(s)}{1 + G(s)C(s)}.$$

  2. Extract the characteristic polynomial, $1 + G(s, \mathbf{q})C(s, \mathbf{k})$.
  3. Compute the determinant of the characteristic polynomial.
  4. Return the expression of the determinant as the function to be tested.

- Stability degree

  The Matlab routine **StaDegr.m** computes the function (or constraint) to be tested by the solver.

  Given the transfer function for the plant, $G(s, \mathbf{q})$, the aim is to design a controller, $C(s, \mathbf{k})$, that obtains a step response that fulfills the specification of relative stability selected on a specific value (**Degr**).

  1. Compute the characteristic polynomial with s.
  2. Substitute $s = Degr + j * w$ for the characteristic polynomial and build the limits of stability (a translation of the zero vertical axis).

3. Expressing the obtained equation in a complex format (Re + j Im),

$$p_1(w, q, k) + j \quad p_2(w, q, k) \neq 0.$$

   (a) Remove w in the equation $p_2(w, q, k)$.

   (b) Compute the determinant of the resultant matrix using $p_1(w, q, k)$ and the new p_2.

   (c) The function to be returned is the determinant of the resultant matrix.

   (d) Return the function to be tested.

- Resonance peak. In the frequency domain, the user can find the regions of the parameter space where a controller fulfill a maximum decibel for the control system.

  The Matlab routine **RessoPeak.m** which computes the frequency response from the feedback system.

  Given the transfer function for the plant, $G(s, \mathbf{q})$, the aim is to design a controller, $C(s, \mathbf{k})$, that obtains a step response that fulfils the specification for the resonance peak.

  1. Compute the characteristic polynomial with s.
  2. Substitute $s = j * w$, corresponding to the system stable response for a sinusoidal input.
  3. Compute the polynomial. The function (or constraint) to be tested is obtained by requiring that the polynomial be less than Res_in.
  4. Return the function to be tested.

- Velocity error. The user enters the maximum velocity error allowed for the control system. The framework returns the set of all the possible intervalar controllers that fulfill this constraint.

The Matlab routine **ErrVelo.m** computes the function (or constraint) to be tested by the solver.

1. The expression for the velocity error is

$$\mid e_v \mid \quad = \frac{1}{\mid K_v \mid} = \frac{1}{\mid \lim_{s->0} \quad sG(s,\mathbf{q})C(s,\mathbf{k}) \mid}.$$

2. Compute the open loop system$G(s,\mathbf{q})C(s,\mathbf{k})$.
3. Compute the expression for e_v imposing that it be less than an specific value (**ErrVelo**). The resultant function is the constraint to be passed to the C++ routines.
4. Return the function to be tested.

- Settling time ($T_{S2\%}$). The user introduces the maximum time until the response settles into a band of 2% of its final response. The framework gives the intervalar controller set from the specified parameter space that fulfills it.

  The Matlab routine **SetTime.m** calculates the settling time , and requires computing

$$R(s,\mathbf{q},\mathbf{k}) = \frac{C(s,\mathbf{k})G(s,\mathbf{q})}{1 + C(s,\mathbf{k})G(s,\mathbf{q})}$$

  1. Compute the characteristic polynomial with s.
  2. Considering $T_{S2\%}=\frac{4}{a} \leq SetTin$ (where $SetTin$ is the specific settling time chosen by the user), then set $a = \frac{4}{SetTin}$ and substitute ($s = -a + j * w$) into the characteristic polynomial to build the limits of stability (a translation of the zero vertical axis).
  3. The resultant equation in complex format is

$$p_1(w,\mathbf{q},\mathbf{k}) + j \quad p_2(w,\mathbf{q},\mathbf{k}) \neq 0.$$

  4. Remove w in the equation $p_2(w,q,k)$.

5. Compute the determinant of the resultant matrix using $p_1(w, q, k)$ and the new p_2.
6. The function to be returned is the determinant of the resultant matrix.
7. Return the function to be tested.

- Control effort. If the user selects this specification, he must introduce a value (*con_in*) so the control system will not pass this limit when computing the control signals.

  The Matlab routine **ContEff.m** calculates the module and requires computing the frequency response of

$$R(s, \mathbf{q}, \mathbf{k}) = \frac{C(s, \mathbf{k})}{1 + C(s, \mathbf{k})G(s, \mathbf{q})}.$$

  1. Compute the characteristic polynomial.
  2. Substitute $s = j * w$.
  3. Imposing that the polynomial be less than *con_in* produces the function (or constraint) to be tested.
  4. Return the function to be tested.

5.3.3 Post design tools

To assist control engineers, IRCAD offers a powerful set of post design functions. These functions allow the user to complete the design by selecting an interval controller from the feasible controller set. They also allow verification that the selected controller achieves the specifications, without exiting from the framework.

Optimal controller

The outcomes of the robust control problem design are returned as a set of controllers that fulfill the specification(s) in the design process. To choose the optimal controller

among this set is sometimes a difficult task. To offer a more reliable framework, the package allows choosing a criterion to select the optimal controller (Fig. 5-12).

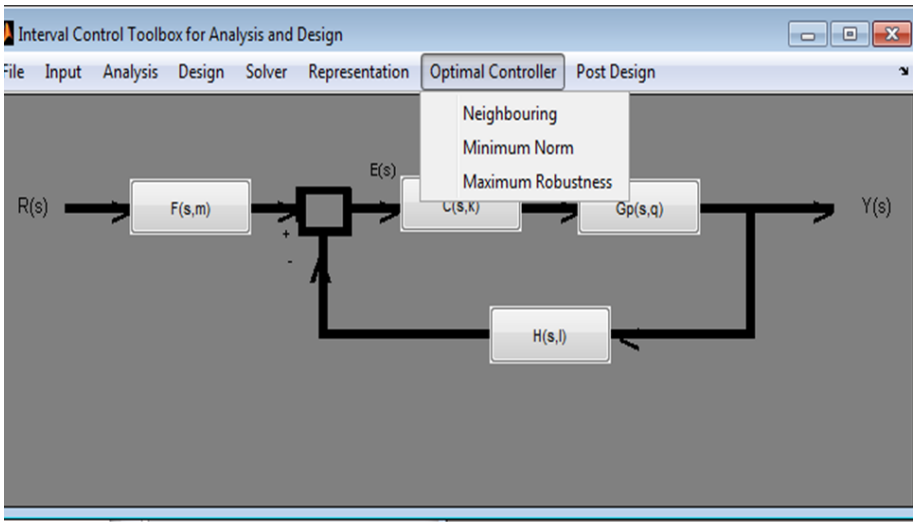


Figure 5-12: IRCAD. Post design tools

This process comprises two steps:

- Choose the criterion to compute the optimal controller. The user can select three criteria to choose the optimal controller from all the possible controllers in the set of interval controllers that fulfill the design specifications:
  - Minimum norm
  - Maximum robustness
  - Neighboring

  Selecting one of these techniques has two outcomes. A graphic is opened corresponding to the last design selection executed (with the parameter space classified by stable, unstable, and undetermined regions). The optimal controller, computed following the selected criteria, is marked as a bold color in the stable region. A

numeric list where the selected interval optimal controller is presented in a text box attached to the graphical representation (Fig. 5-13).

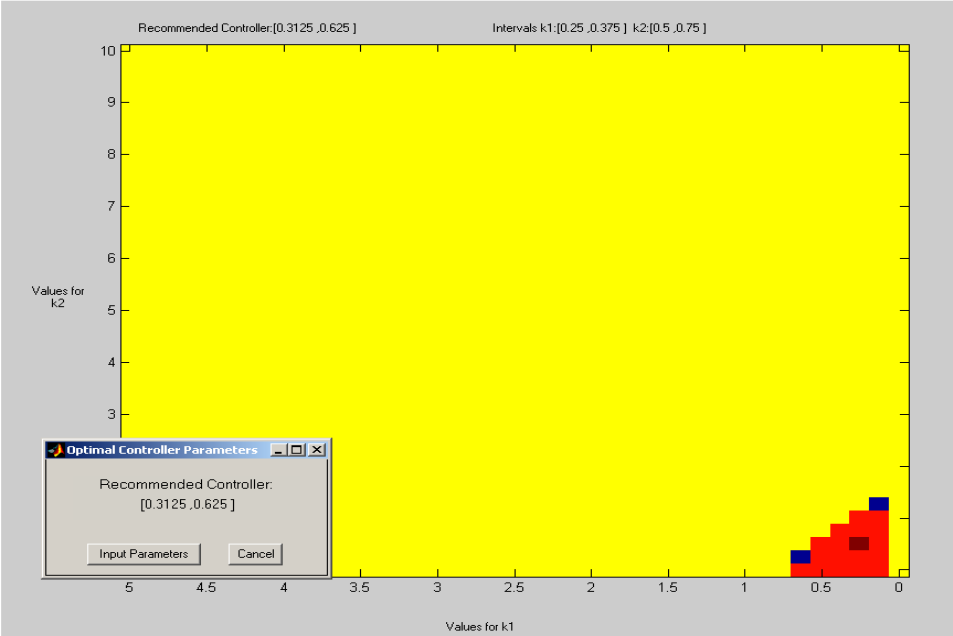


Figure 5-13: IRCAD. Choosing the optimal controller using neighboring criterion

The criteria are:

- Minimum norm: Given a graphical region of feasible controllers in the parameter space delimited by the parameters of the controller k1 and k2, this criterion selects the intervalar controller (k1,k2) that fulfilling the specifications with the minimum norm. The norm is calculated from the parameter space origin to the square defined by the interval (k1,k2). The selected controller is presented graphically, marked in bold red, and numerically as shown in Fig. 5-14.
- Maximum robustness: The framework computes the stability margin over each square, and selects the square with the maximum value as the controller. The selected controller is presented graphically, marked in bold red, and numerically as shown in Fig. 5-15.

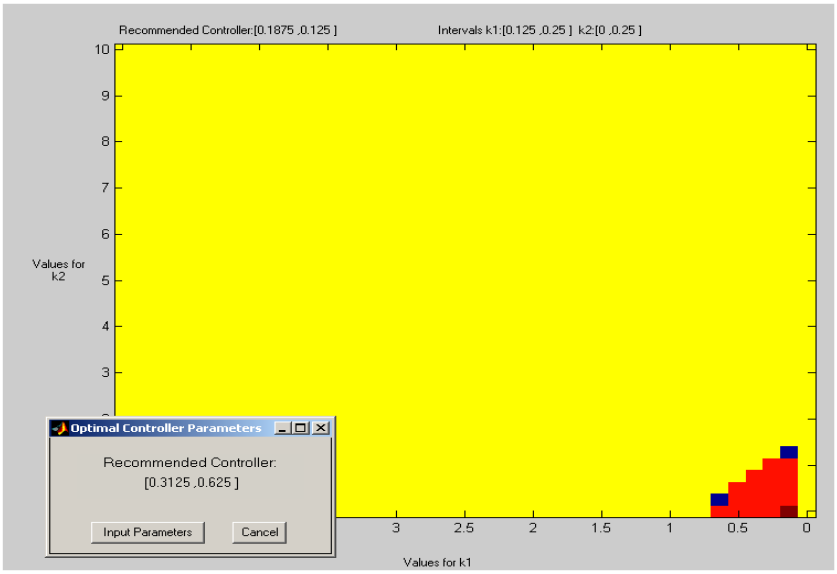


Figure 5-14: IRCAD. Choosing the optimal controller using minimum norm criterion

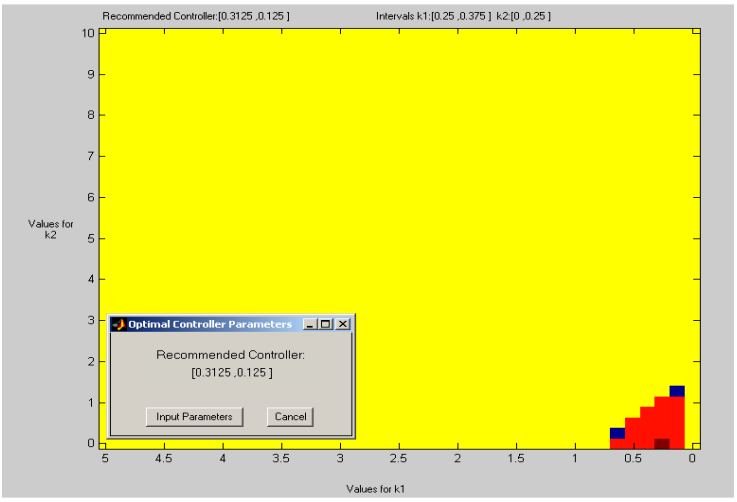


Figure 5-15: IRCAD. Choosing the optimal controller using maximum robustness criterion

– Neighboring: The framework computes the number of neighbors for each feasible intervalar controller. The square with the maximum number of neighbors is selected as shown in Fig. 5-13.

- Input optimal controller. The recommended value for the controller can be introduced as an entry using the input menu, or it can be introduced automatically using the post design tools as shown in Fig. 5-16.

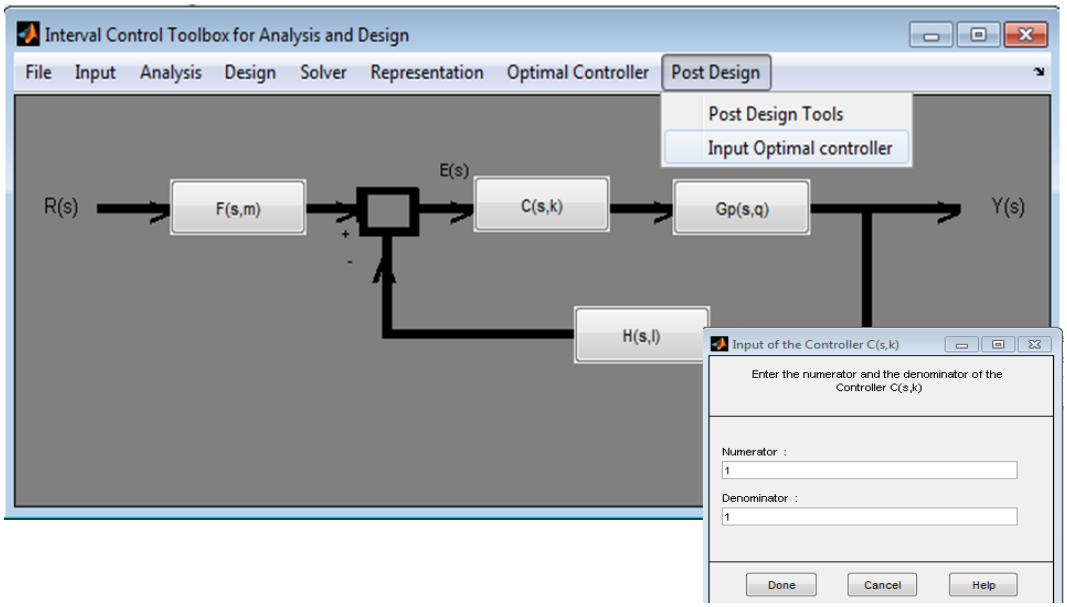


Figure 5-16: IRCAD. Input optimal controller

The framework allows the parameters of the recommended controller, obtained using one of the techniques discussed above, can be directly input. This way to insert controller parameters, allows the execution of post design tasks without exiting from the framework.

Post design tools

The final IRCAD menu is the post design tool that allows the user to verify the specification (or set of specifications) selected for the design.

It is not necessary to move to a simulation environment to check the results of the design process, because the selected controller can be simulated using IRCAD directly (Fig. 5-17).

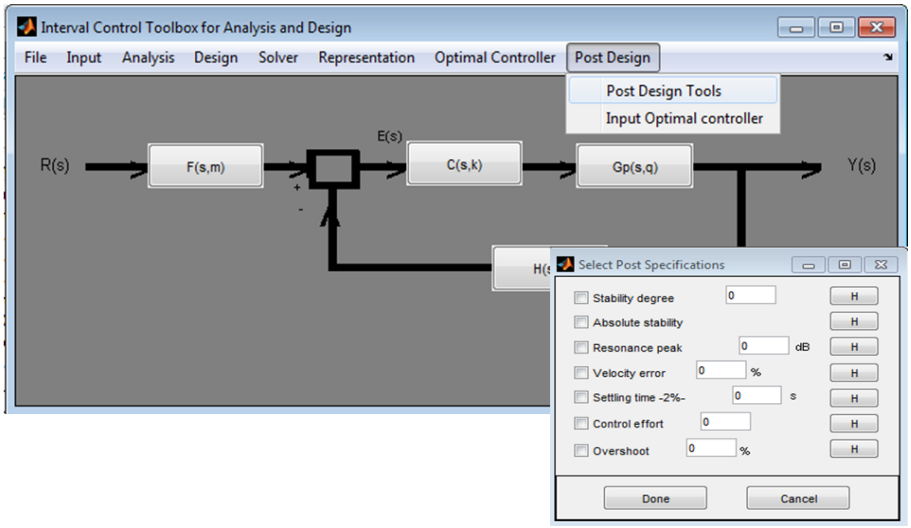


Figure 5-17: IRCAD. Post design tools

The tools of this block provide facilities to check that a designed controller fulfill a set of specifications among the following: Stability degree, Absolute stability, Resonance peak, Velocity error, Settling time 2%, Control effort and Overshoot.

These provide a user-friendly environment to develop robust control problems, with the security that the outputs are guaranteed solutions.

As an example, suppose the user selects the design specification of absolute stability. Then the post design tool available by IRCAD to verify the controller calculated by the framework, fulfills the specifications is a graphical tool. The result is a zero

pole map for all the parameters of the plant (Fig. 5-18).

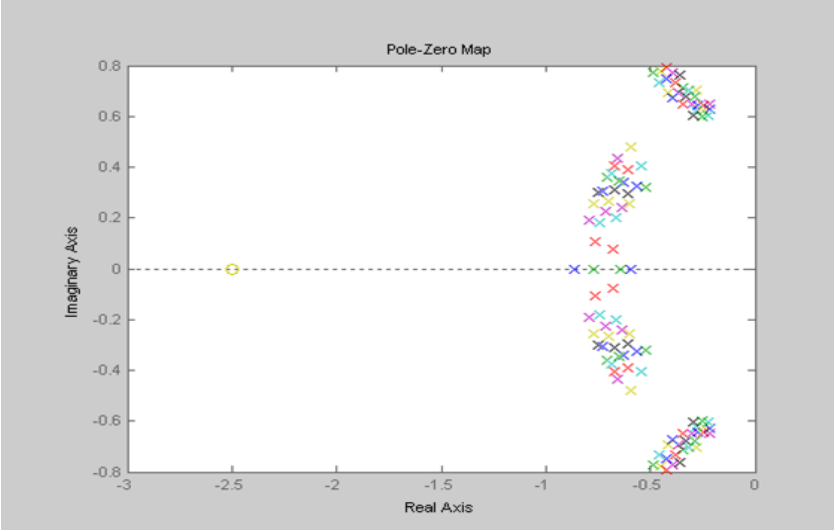


Figure 5-18: IRCAD. Zero pole map for absolute stability in the post design tool.

5.4 Summary

MIA [85] provides necessary and sufficient conditions of stability that can be implemented via algorithms which verify stability for a given uncertain domain. This chapter showed the integration of MIA based algorithms developed to solve problems of robust control into a user-friendly framework called IRCAD. The aim of this framework is to offer to the control engineer automated tools to study robust control analysis and design of a system on the parameter space.

Chapter 6

Design example using IRCAD

We present an application of the IRCAD framework extracted from literature to illustrate how a complete example of design is implemented.

6.1 Introduction

We chose the most representative case from the set of examples implemented during the thesis development. This example was proposed by Fioro et al [21] and corresponds to a simple but complete example in the area of robust control design.

6.2 Statement of the general problem

To illustrate how the solver builds and manages the data, we used the example presented in Section 2.3.2, as suggested by Fiorio et al. [21] and later studied by Malan et al. [50]. The aim of the example is tuning a PI controller for an interval plant.

The transfer function of the plant is

$$G(s, \mathbf{q}) = \frac{q_1}{1 - \frac{s}{q_2}}, \tag{6.1}$$

where $\mathbf{q}$ is the vector of uncertain parameters, $\mathbf{q} = [q_1 \quad q_2]$; $q_1 = q_2 = [0.8, 1.25]$. The

PI controller is

$$C(s, \mathbf{k}) = \frac{k_1 \left(1 + \frac{s}{k_2}\right)}{s}, \tag{6.2}$$

where $\mathbf{k}$ is the vector of the design parameters, $\mathbf{k} = [k_1 \quad k_2]$.

The design aim is to find the set $\mathbf{K}$ of controller parameters that fulfill the performance specifications:

1. Absolute stability.
2. Velocity error lower than 2%.
3. Resonance peak lower than 3 dB.
4. Control effort lower than 20.

6.3 Problem design using IRCAD

The aim is to tuning the PI controller using IRCAD framework, i.e., we want to compute the two parameters for the intervalar controller. To start the designing process by IRCAD, we need to select the design specifications to be fulfilled, as shown in Fig. 6-1

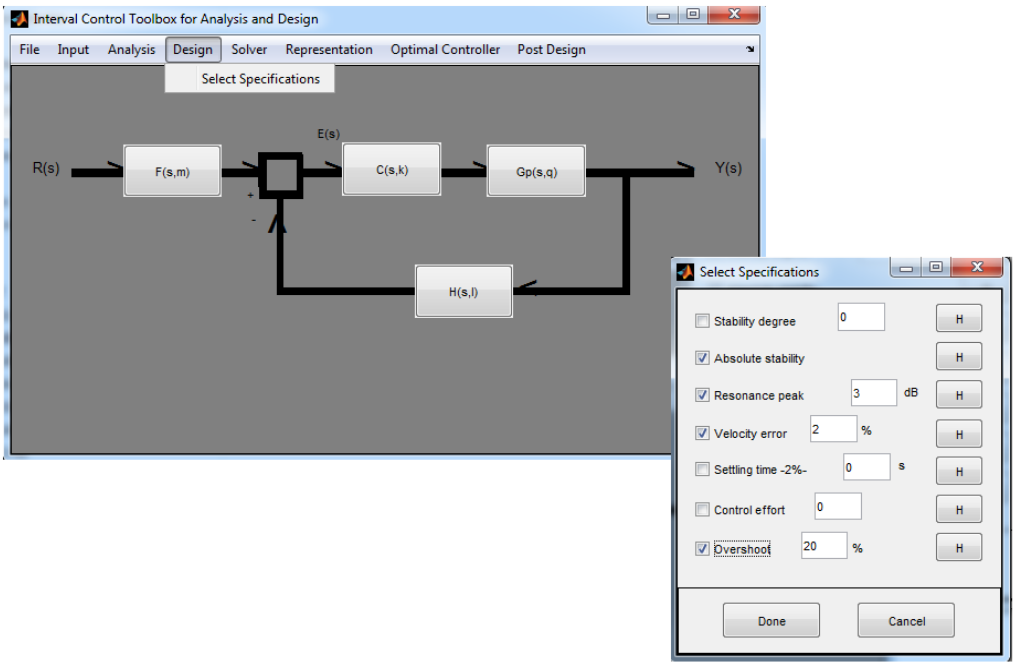


Figure 6-1: IRCAD. Selection of design specifications

The framework manages each one of the specifications, transforming them via the Symbolic Toolbox into a set of polynomial interval functions (Eqs. (6.3)–(6.6)). Matlab transforms these to matrix structure and passes the functions to the solver. This transformation is performed within IRCAD, and is transparent to the user. The solver tests the positivity of the functions over the parameter space, as determined by the controller parameters. In this case the parameter space is determined by k_1 and k_2. IRCAD allows presentation of the results graphically, or as a set of intervals.

For this example, the parameter space to be checked is limited by providing initial interval values $k_1 = [-200, 0]$ and $k_2 = [0, 10]$. To work with positive intervals, $\overline{k_1} = -k_1$ has been substituted.

As explained in Section 3.3.2, to test if one specification or a set of specifications are fulfilled using MIA it is enough to test if a function or a set of functions are positive. In

our example it is enough to test the positivity of the following functions:

1. For the specification of absolute stability, the resultant function is

$$f1 = -k_2 + q_1\overline{k_1}. \tag{6.3}$$

2. For the specification of velocity error less than 2%, the resultant function is

$$f2 = q_1\overline{k_1} - 50. \tag{6.4}$$

3. For the specification of resonance peak lower than 3 dB, the resultant function is

$$\begin{aligned} f3 &= 2k_2^2w_1^4 - 4k_2^2w_1^2\overline{k_1}q_1q_2 + \overline{k_1}^2q_1^2q_2^2k_2^2 + 2k_2^2w_1^2q_2^2 - 4k_2w_1^2q_2^2\overline{k_1}q_1 \\ &+ \overline{k_1}^2q_1^2q_2^2w_1^2. \end{aligned} \tag{6.5}$$

4. Finally, for control effort lower than 20, the resultant function is

$$\begin{aligned} f4 &= 400k_2^2w_1^4 - 800k_2^2w_1^2\overline{k_1}q_1q_2 + 400\overline{k_1}^2q_1^2q_2^2k_2^2 + 400k_2^2w_1^2q_2^2 \\ &+ 400\overline{k_1}^2q_1^2q_2^2w_1^2 - \overline{k_1}^2k_2^2q_2^2 - \overline{k_1}^2w_1^4 - \overline{k_1}^2k_2^2w_1^2 - \overline{k_1}^2w_1^2q_2^2 \\ &- 800k_2w_1^2q_2^2\overline{k_1}q_1. \end{aligned} \tag{6.6}$$

The specification of absolute stability (6.3) depends only on q_1, k_1 and k_2, and for the velocity error specification (6.4) only depends on q_1 and k_1. Thus an analytical solution is possible.

The other two other expressions, (6.5) and (6.6) have a more complicated computation due their dependence on five parameters: two plant parameters, two PI controller parameters, and the frequency. The theoretical range of variation is $w = [0, \infty]$, but for practical purposes it is enough to consider $w = [0, 100]$ since the bandwidth will be of the order of 15 rad/s. Expressions for f_3 and f_4 may be simplified by substituting $\alpha = w^2$

Parameter	value
q_1	$[0.8, 1.25]$
q_2	$[0.8, 1.25]$
k_1	$[0, 200]$
k_2	$[0, 10]$
α	$[0, 10000]$
$N(divk_1)$	40
$N(divk_2)$	40
$Maxlist$	$variable$

Table 6.1: IRCAD: Parameters for the example

With these considerations, using the parameters shown in Table 6.1, the regions of fulfilling controllers are computed for each one of the four specifications. The resolution is 5 unities for parameter q_1 and 0.25 unities for parameter q_2.

The resultant feasible regions for each specifications using IRCAD are shown in Fig. 2-2. Specifications f3 and particularly f4 have high computation cost due the high number of summand components and also the large number of plant and controller variables. However, the computational cost is significantly reduced for f1 and f2, because some regions are discarded by the application of least cost..

The parameter space area of the fulfilling controllers is the intersection of the four regions of Fig. 2-2, which is shown in detail in Fig. 2-3

6.3.1 Computation of the set of controllers that fulfill design specifications

To simplify the problem. The computation will be made using three of the four specifications.

- Absolute stability.
- Resonance peak lower than 3 dB
- Control effort lower than 20.

This computation can be made following different strategies:

- SCDS algorithm.
- NFV algorithm.

$k_1 = [-70, -65]$	$k_2 = [2.75, 3]$
$k_1 = [-75, -65]$	$k_2 = [3, 3.25]$
$k_1 = [-80, -65]$	$k_2 = [3.25, 3.5]$
$k_1 = [-85, -65]$	$k_2 = [3.5, 3.75]$
$k_1 = [-90, -65]$	$k_2 = [3.755, 4]$
$k_1 = [-95, -65]$	$k_2 = [4, 4.25]$
$k_1 = [-100, -65]$	$k_2 = [4.25, 4.5]$
$k_1 = [-100, -75]$	$k_2 = [4.5, 4.75]$
$k_1 = [-105, -80]$	$k_2 = [4.75, 5]$
$k_1 = [-110, -85]$	$k_2 = [5, 5.25]$
$k_1 = [-110, -95]$	$k_2 = [5.25, 5.5]$
$k_1 = [-110, -100]$	$k_2 = [5.5, 5.75]$
$k_1 = [-110, -105]$	$k_2 = [5.75, 6]$

Table 6.2: Set of feasible controllers

Design example using the single controller design specification(SCDS) algorithm

This example has been reduced to three specifications, so the process below must be repeated three times, one for each specification. Only the first specification is reported in detail.

Specification

The condition of absolute stability obtained for this specification must be expressed in its most simplified format, in our case obtained with the help of Maple.

substituting $\tilde{k}_1 = -k_1$, the transfer function of the closed loop system is

$$M(s, \mathbf{q}, \mathbf{k}) = \frac{\tilde{k}_1(k_2 + s) \quad q_1 q_2}{k_2 s^2 + (-k_2 q_2 + \tilde{k}_1 q_1 q_2)\, s + \tilde{k}_1 q_1 q_2 k_2},$$

and the characteristic polynomial is

$$pc(s, \mathbf{q}, \mathbf{k}) = k_2 s^2 + (-k_2 q_2 + \tilde{k}_1 q_1 q_2)\, s + \tilde{k}_1 q_1 q_2 k_2.$$

The absolute stability condition is obtained from the coefficient of the s term,

$$f_1(\mathbf{q}, \mathbf{k}) = -k_2 + \tilde{k}_1 q_1 > 0.$$

Constrain

Substituting $X_1 = k_2, \quad X_0 = \tilde{k}_1, \quad X_2 = q_1$, the constraint is

$$f_1 = -X_1 + X_0 X_2.$$

Matrix structure

Using the Matlab routine included in Appendix A, the constraint is transformed into

$$\begin{bmatrix} 0 & X0 & X1 & X2 \\ -1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix},$$

which is split into three smaller matrices

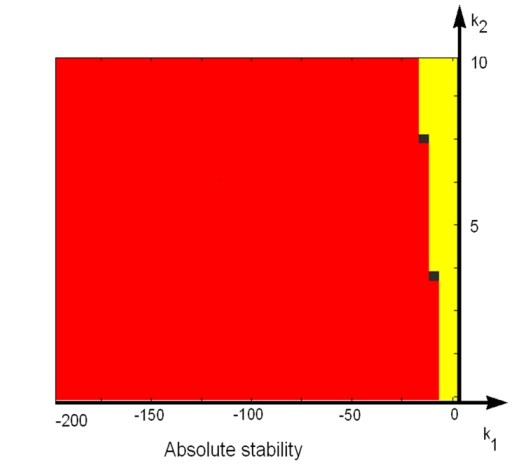
- matrix (vector) of variables:
$$\begin{bmatrix} X0 \\ X1 \\ X2 \end{bmatrix}$$
- matrix of coefficients:
$$\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

- matrix of exponents:

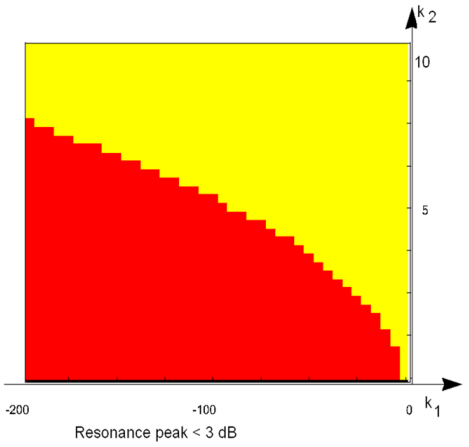
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Results

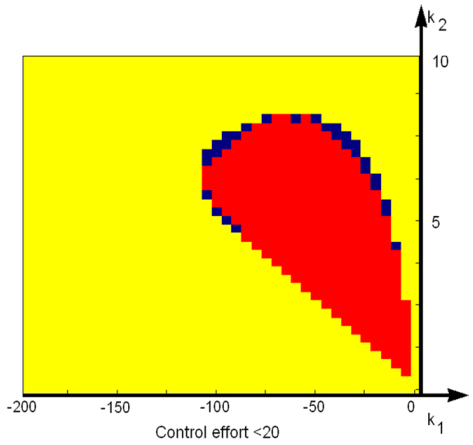
This algorithm individually tests each one of the specifications. Graphical representations of the outcomes for each specification are shown in Fig. 6-2.



a: absolute stability



b: resonance peak



c: control effort

Figure 6-2: Feasible regions

The colors for each box are:

- Yellow: Regions which are not feasible.
- Red: Regions which are feasible.
- Blue: Undetermined regions.

Execution of the three specifications produces three matrices of dimension 40×40 that can be represented within, and also saved as part of the Matlab workspace.

Example using the multiple specification algorithm NFV

The algorithm includes the same three specifications, but here only one execution is required to achieve the regions which fulfill the three specifications simultaneously.

Specifications

Following the same process as in the single specification algorithm, the specifications obtained are in Eqs (6.3), (6.5) and (6.6).

Constraints

Substituting $X_1 = k_2, \quad X_0 = \tilde{k}_1, \quad X_2 = q_1$, the three constraints are

$$f_1 = -X_1 + X_0X_2, \tag{6.7}$$

$$f_3 := 2X_1^2X_4^2 + 4X_1^2X_4X_0X_2X_3 + 2X_1^2X_4X_3^2 + 4X_4X_1X_3^2X_0X_2 + X_4X_3^2X_2^2X_0^2, \tag{6.8}$$

$$f_4 := 400X_1^2X_4^2 - 800X_1^2X_4X_0X_2X_3 + 400X_0^2X_2^2X_3^2X_1^2 + 400\,X_1^2X_4X_3^2,$$

$$-800X_1X_4X_3^2X_0X_2 + 400X_0^2X_2^2X_3^2X_4 - X_0^2X_1^2X_3^2 - X_0^2X_4^2 - X_0^2X_1^2X_4 - X_0^2X_4X_3^2; \tag{6.9}$$

Matrix structure

For our example of three specifications, the sorting criterion followed to evaluate the constrains is presented in the Matlab script of Appendix A. The justification for this sorting method is summarized in Table 6.3 using the following three functions:

1. First function ==> The function of absolute stability regions(2 addends - 3 variables - 2 controller parameters).

2. Second function =>The function of regions with resonance peak less than 3 dB (6 addends - 5 variables - 2 controller parameters).

3. Third function ==>The function of regions with control effort less than 20 (10 addends - 5 variables - 2 controller parameters).

Function	Addends	Variables	Controller parameters
absolute stability regions	2	3	2
resonance peak $<$ 3 dB	6	5	2
control effort $<$ 20	10	5	2
...	...	...	...

Table 6.3: IRCAD: Parameters for the example

The matrix structure is composed by three matrices:

- COEF: Matrix that contains the coefficients of the constraints 6.7, 6.8, 6.9.

- VARI: Matrix that contains the variables of the constraints 6.7, 6.8, 6.9.

- EXPO: Matrix that contains the exponents of the variables of the constraints 6.7, 6.8, 6.9.

The COEF matrix is formed by the juxtaposition of $N + 1$ vectors where N is the number of functions (or specifications). In our example with three specifications, there

are four columns with empty elements filled by zeros,

$$\left[\begin{array}{cccc} 2 & -1 & 2 & 400 \\ 3 & 1 & -4 & -800 \\ 6 & \mathit{0} & 1 & 400 \\ 5 & \mathit{0} & 2 & 400 \\ 10 & \mathit{0} & -4 & -800 \\ 5 & \mathit{0} & 1 & 400 \\ \mathit{0} & \mathit{0} & \mathit{0} & -1 \\ \mathit{0} & \mathit{0} & \mathit{0} & -1 \\ \mathit{0} & \mathit{0} & \mathit{0} & -1 \\ \mathit{0} & \mathit{0} & \mathit{0} & -1 \end{array}\right]$$

In column 1, positions 1 & 2 correspond to the number of rows (number of addends) and number of columns (number of variables), respectively, of the first function (specification), positions 3 & 4 correspond to the number of addends and number of variables for the second function, respectively, and positions 5 & 6 correspond to the number of addends and variables for the third function. Column 2, 3, and 4 are the coefficients of functions 1, 2, and 3, respectively.

The EXPO matrix is formed by the juxtaposition of N matrices, each one with their own dimension. In our example:

- Matrix of exponents for the first specification:

$$\left[\begin{array}{ccc} 0 & 1 & 0 \\ 1 & 0 & 1 \end{array}\right]$$

- Matrix of exponents for the second specification:

$$\begin{bmatrix} 0 & 2 & 0 & 0 & 2 \\ 1 & 2 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 0 \\ 0 & 2 & 0 & 2 & 1 \\ 1 & 1 & 1 & 2 & 1 \\ 2 & 0 & 2 & 2 & 1 \end{bmatrix}$$

- Matrix of exponents for the third specification:

$$\begin{bmatrix} 0 & 2 & 0 & 0 & 2 \\ 1 & 2 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 0 \\ 0 & 2 & 0 & 2 & 1 \\ 1 & 1 & 1 & 2 & 1 \\ 2 & 0 & 2 & 2 & 1 \\ 2 & 2 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 2 \\ 2 & 2 & 0 & 0 & 1 \\ 2 & 0 & 0 & 2 & 1 \end{bmatrix}$$

- The global matrix EXPO is the juxtaposition of the three matrices with zeros to

fill the empty positions:

$$\begin{bmatrix}
0 & 1 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 2 & 0 & 0 & 2 \\
1 & 0 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 \\
\mathit{0} & \mathit{0} & \mathit{0} & 2 & 2 & 2 & 2 & 0 & 2 & 2 & 2 & 2 & 0 \\
\mathit{0} & \mathit{0} & \mathit{0} & 0 & 2 & 0 & 2 & 1 & 0 & 2 & 0 & 2 & 1 \\
\mathit{0} & \mathit{0} & \mathit{0} & 1 & 1 & 1 & 2 & 1 & 1 & 1 & 1 & 2 & 1 \\
\mathit{0} & \mathit{0} & \mathit{0} & 2 & 0 & 2 & 2 & 1 & 2 & 0 & 2 & 2 & 1 \\
\mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & 2 & 2 & 0 & 2 & 0 \\
\mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & 2 & 0 & 0 & 0 & 2 \\
\mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & 2 & 2 & 0 & 0 & 1 \\
\mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & \mathit{0} & 2 & 0 & 0 & 2 & 1
\end{bmatrix}$$

Similarly, the VARI matrix is formed by the juxtaposition of three matrices:

- The matrix of variables for the first specification (which has three variables) is:

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \end{bmatrix}$$

- The matrix of variables for the second specification (which has five variables) is:

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

- The matrix of variables for the third specification (which has five variables) is:

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

- Thus, VARI is the juxtaposition:

$$\begin{bmatrix} X_0 & X_0 & X_0 \\ X_1 & X_1 & X_1 \\ X_2 & X_2 & X_2 \\ 0 & X_3 & X_3 \\ 0 & X_4 & X_4 \end{bmatrix}$$

Results

We obtain the set of regions of the parameter space from which the controller can be chosen. It corresponds to the intersections of the red (fulfilled) three parameter areas. The result, shown in Fig. 6-3, is obtained by intersecting the three matrices.

6.3.2 Selection of one controller using post design tools

IRCAD offers a set of post design tools that allows the user to choose the optimal controller from the set of the automatically proposed controllers. The framework chooses the controller following the criteria the user has selected among:(Section 5.3.3).

- Minimum norm
- Neighboring
- Maximum robustness

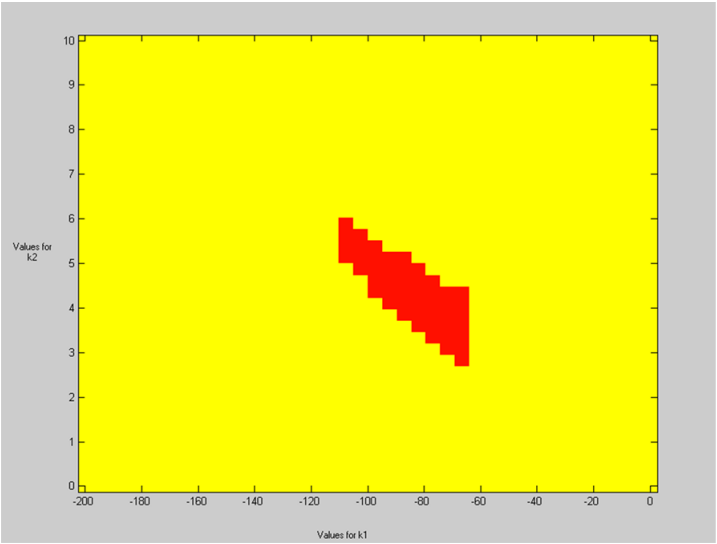


Figure 6-3: Region of feasible controllers for the three specifications defined

In this example, we suppose the user select the criterion of minimum norm. The controller suggested by IRCAD is $[-67.5, 2.875]$, as shown in Fig. 6-4.

6.3.3 Validation of the proposed controller using post design tools from IRCAD

To validate the controller, the user must input the suggested controller in the controller transfer function. This action can be performed in two ways:

- Manually: The user can type in the optimal parameters of the controller, the controller obtained by IRCAD.
- Automatically: When the controller is suggested by IRCAD, there is an option that allows it to be inserted on the closed loop in an automatic way.

Whatever is the way selected, the effect is the same. The PI controller obtained by IRCAD, $k_1 = -67.5$ and $k_2 = 2.875$ is introduced to the transfer function of the controller C(s) for the closed loop system (Fig. 6-5) . Its expression is:

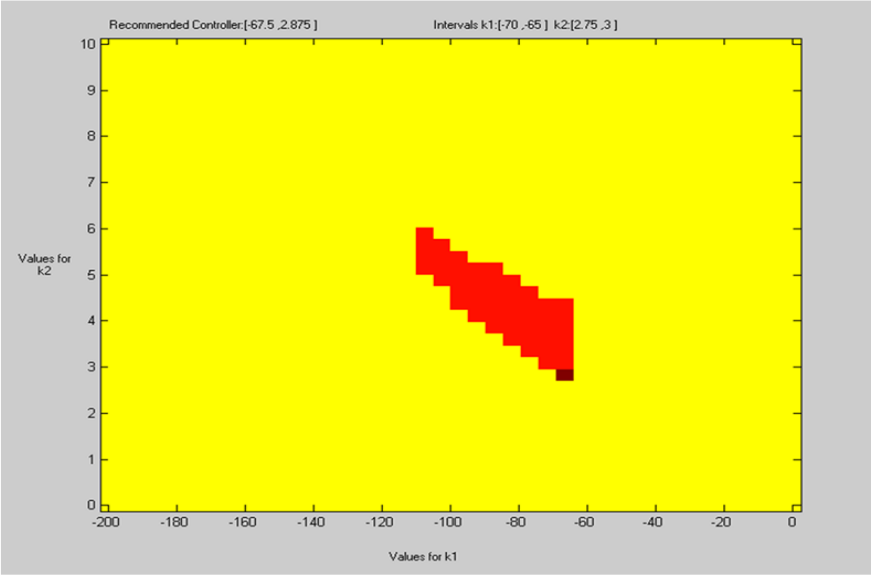


Figure 6-4: Minimum norm criteria

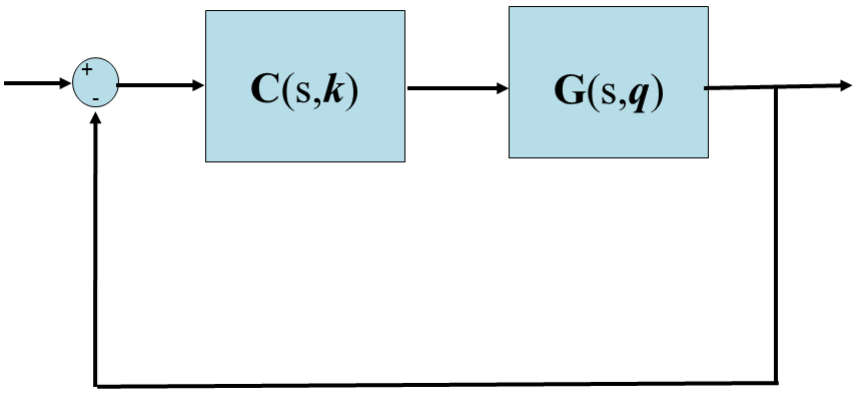


Figure 6-5: Closed loop system with the controller suggested by IRCAD

$$C(s) = \frac{-67.5\left(1 + \frac{s}{2.875}\right)}{s} \tag{6.10}$$

IRCAD also offers a set of tools that allows validation of the suggested controller without exiting from the framework (see Fig. 6-6):

- Apply an input to the control system to check if the specifications are actually fulfilled.
- Draw the temporal response of the control system.
- Check design specifications. In our example this option is taken to check the three design specifications: absolute stability, resonance peak and control effort.

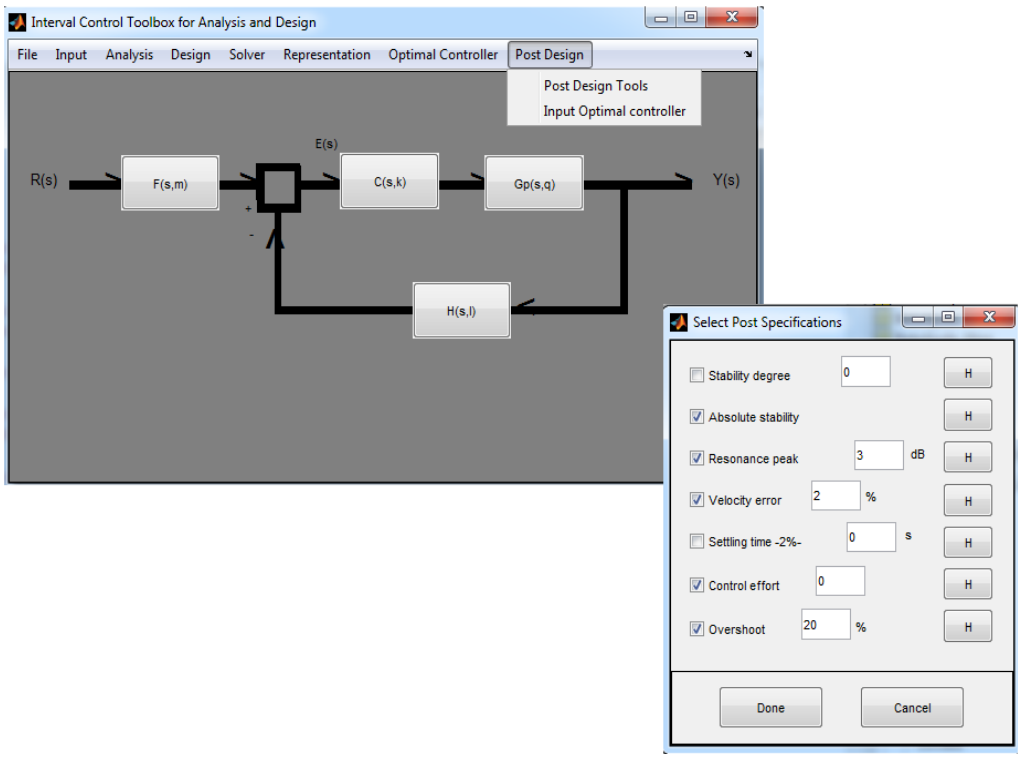


Figure 6-6: Post design tool to check specifications in IRCAD

The result is in Figure . 5-18.

6.4 Summary

This chapter presents an example of design with a high computation level and its resolution using IRCAD . Three specifications of robust control design are taken into account

to illustrate the whole process. IRCAD take the specifications, transform them into constraint problems. The solver transform these constraints into matrices, computes its positivity and classify the parameter space into regions that fulfill the specifications, regions that don't fulfill the specifications and into undetermined regions. From the region that fulfills, IRCAD offers tools to select the optimal controller, thus the problem of robust control design has been solved.

Chapter 7

Application of IRCAD to mobile robot control design

In this chapter is presented the experimental application of a control law, developed using IRCAD, to an educational and research mobile robot.

7.1 Introduction

Path following is an important aspect of robotics. Computation of control laws required for a robot to perform this task is reported in many works. In this chapter we show how these laws may be computed using the IRCAD design tools.

We first detail the robot that the derived control laws are implemented upon. The robot has three models corresponding to three different velocities at which it can operate (low, medium and high). In a classical control design this structure corresponds to the computation of three control laws: one each for low, medium, and high velocities, which in turn requires a commuted controller.

We propose an alternative solution to the control of the robot, consisting of designing a single intervalar controller to covers the three velocities. The design of this controller was accomplished using the IRCAD framework.

Path following outcomes using a classical commuted PI controller are compared to the single intervalar controller for the robot following the same path.

7.2 The robot PRIM

PRIM is a mobile robot built at the University of Girona to provide the platform as an open educational tool, as well as an available research platform. Educational outcomes cover electronics, control and modeling, sensor fusion, and computer science, among other areas. Research activities from the mutual interaction and integration between subjects, and high level control strategies have also been developed. The use of open platforms allows development of understanding in a multidisciplinary context.

7.2.1 Mechanical description

The robot structure is made from aluminum, with parts distributed on different levels. On the first level are two differential driven wheels, controlled by two DC motors, and a third omni-directional wheel providing the third contact point with the floor. On the second level is the PC computer, and on the third level the specific hardware and sonar sensors. The forth level could be used, extending the flexibility of the system, to place machine vision system and/or multimedia set ups depending on the platform application. Table 7.1 summarizes the basic mechanical description of PRIM.

7.2.2 System architecture

This section introduces the wheeled mobile robot (WMR) system architecture used in PRIM. The main decision system resides within the PC that controls the hardware. Data gathering and control frequency is 100 ms. The platform allows connection to other PCs through a LAN, which facilitates multimedia point of information and machine vision, as an advanced sensor system (Fig. 7-1).

Features of	robot PRIM
Wide	$580mm$
Large	$400mm$
Height	$1200mm$
Distance between wheels	$560mm$
Diameter of the wheels	$160mm$
Weight	$20Kg$
Maximal speed	$0.48m/s$
Motor max. cont. torque	$131mNm$
Gear reduction	$86:1$
Total robot force	141N

Table 7.1: Robot PRIM mechanical properties

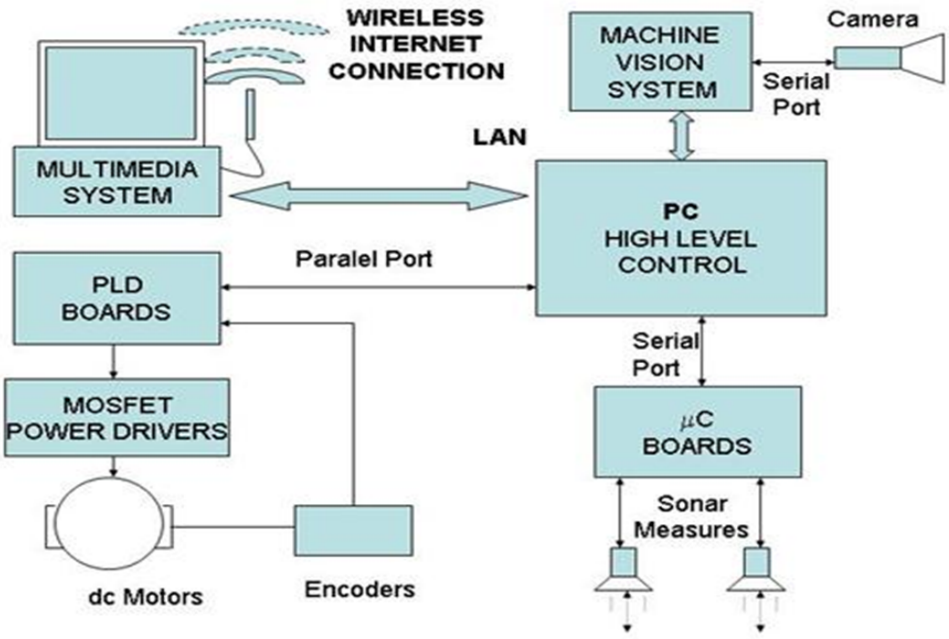


Figure 7-1: Decision architecture for PRIM

The multimedia system is composed of a PC with a tactile screen that allows interaction with humans. The computer is configured with the software that the various user applications demand. A wireless internet connection allows communication with the whole world, and provides multiple possibilities.

The machine vision system is composed of a remote camera with motorized focus, iris, and zoom control by a serial port, two stepper motors that control the pan and tilt position of the camera, and specific hardware boards running on a PC exclusively used by the machine vision system. The system is connected to the main control system through a LAN.

The control laws, computed in any method, to control PRIM, must be implemented as a part of the code consisting of the generation of straight line trajectory tracking in C under Windows XP on the high level PC system.

7.3 Mobile robot kinematic and dynamic systems

PRIM is a differentially driven WMR with a free rotating wheel, as shown in Fig. 7-2, designed for indoor navigation.

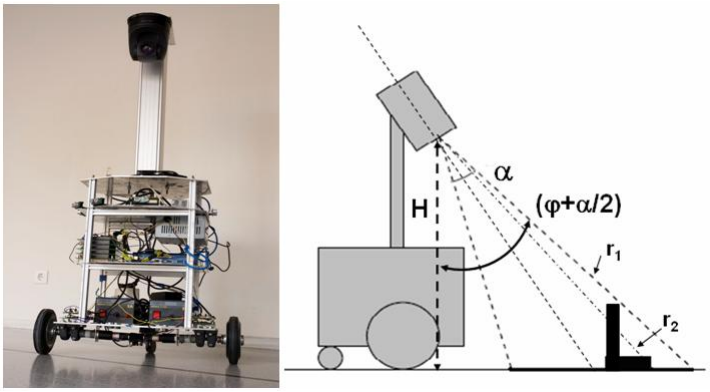


Figure 7-2: Wheeled mobile robot architecture for PRIM

Kinematic system

The WMR is a rigid body, and consequently non-deforming wheels are required. The vehicle is assumed to move without slipping on a plane, so that there is pure rolling contact between the wheels and the ground.

Denoting the position and orientation coordinates by (x, y, θ), and the velocity vector by $u = [v, w]$, where v and w are the tangential and angular velocities respectively, the kinematic model of the WMR can be stated as

$$dx = vcos\theta,$$

$$dy = vsin\theta,$$

and

$$\theta = w,$$

Using a discrete time representation (with T being the sampling period and k the time instant) and Euler's approximation, the following discrete time model can be obtained for the robot dynamics

$$x(k+1) = x(k) + v(k)cos\theta(k)T,$$

$$y(k+1) = y(k) + v(k)sin\theta(k)T,$$

and

$$\theta(k+1) = \theta(k) + w(k).$$

The WMR platform uses incremental encoders to obtain the position and orientation coordinates. We can describe the positioning of the robot as a function of the radius of the left and right wheels (equations 7.1, 7.2), (R_e and R_d), respectively, and the angular incremental positioning (θ_e, θ_d), where E is the distance between the two wheels and dS is the incremental displacement of the robot. The position and angular incremental

displacements can be expressed as:

$$dS = \frac{R_d d\theta_d + R_e d\theta_e}{2}, \tag{7.1}$$

and

$$d\theta = \frac{R_d d\theta_d - R_e d\theta_e}{E}. \tag{7.2}$$

The (x, y, θ) coordinates can be expressed as:

$$x(k+1) = x(k) + dscos(\theta(k) + d\theta),$$

$$y(k+1) = y(k) + dssin(\theta(k) + d\theta),$$

and

$$\theta(k+1) = \theta(k) + d(k).$$

Thus, the incremental position of the robot can be obtained using the odometer and the encoder information.

7.4 System identification for PRIM: Transfer functions

The aim of this section is to present a set of dynamic models for high (0.9 m/s), medium (0.6 m/s), and slow (0.3 m/s) velocities suitable for controlling the velocity of each wheel using PID controllers. The models are obtained using a set of linear transfer functions which represent the nonlinearities of the whole system.

The parametric identification process is based on black box models ([49], [64]). The nonholonomic system of PRIM is considered initially to be a multiple input multiple output (MIMO) system, as shown in Fig. 7-3, because of the dynamic influence between the two DC motors. This MIMO system is composed of a set of single input single output

(SISO) subsystems with coupled connections.

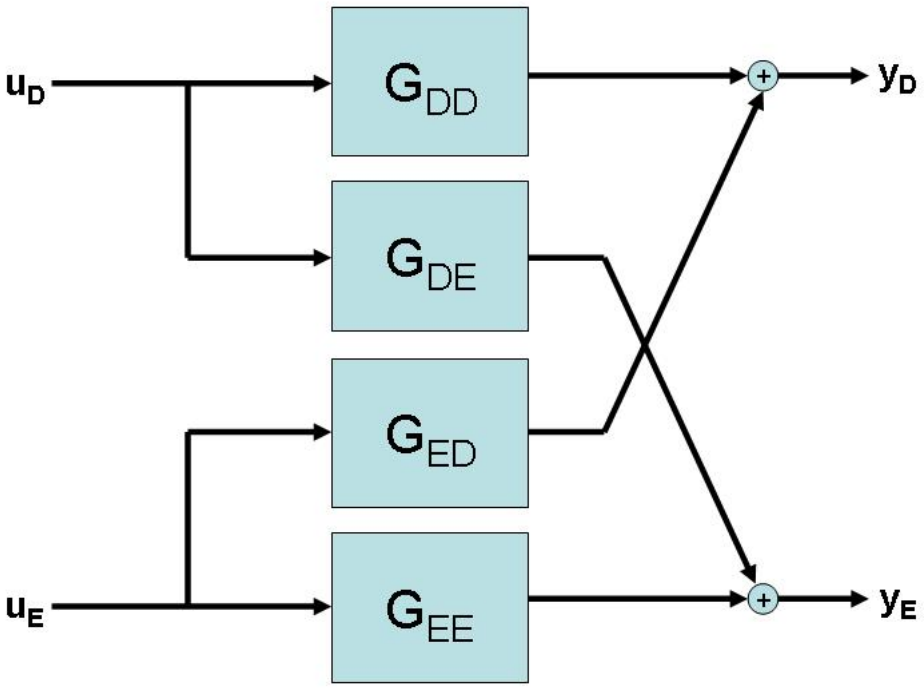


Figure 7-3: Structure of a multiple input multiple output system

Parameter estimation is performed using a pseudo-random binary signal (PRBS), such as an excitation input signal. This guarantees the correct excitation of all dynamic sensible modes of the system along the whole spectral range, resulting in accurate high precision parameter estimation. The experiments be performed consisted of exciting the two DC motors at various speed ranges (low, medium, and high).

An autoregressive with external input (ARX) structure was employed to identify the system parameters. The problem consists of finding a model that minimizes the error between the real and estimated data. Expressing the ARX equation as a lineal regression, the estimated output is

$$\hat{y} = \lambda\phi, \tag{7.3}$$

where $\hat{y}$ is the estimated output vector, λ is the vector of estimated parameters, and ϕ is

the vector of measured input variables. Using the coupled system structure, the transfer function of the robot can be expressed as

$$\begin{pmatrix} Y_D \\ Y_E \end{pmatrix} = \begin{pmatrix} G_{DD} & G_{ED} \\ G_{DE} & G_{EE} \end{pmatrix} . \begin{pmatrix} U_D \\ U_E \end{pmatrix}$$

,

where Y_D and Y_E represent the speed of the right and left wheels, and U_D and U_E the corresponding speed commands, respectively.

To obtain information about the coupled system, the transfer function matrix must be identified. The process of identification was developed by Pacheco, Ll. [73], and comprise four transfer functions corresponding to the four possible interrelations between the wheels. G_{DD} is the right wheel transfer function, G_{EE} is the left wheel transfer function, and G_{DE} and G_{ED} are the dynamic influence between the left and right wheel DC motors. The system is identified using the Identification toolbox of Matlab ('ident' command), employing the PRBS signals as inputs. Table 7.2 shows the continuous transfer functions obtained for the three different lineal speed models.

Linear trans.function	High velocities	Medium velocities	Low velocities
G_{DD}	$\frac{0.20s^2-3.15s+9.42}{s^2+6.55s+9.88}$	$\frac{0.20s^2+3.10s+8.44}{s^2+6.17s+9.14}$	$\frac{0.16s^2+2.26s+5.42}{s^2+5.21s+6.57}$
G_{ED}	$\frac{-0.04s^2-0.60s-0.32}{s^2+6.55s+9.88}$	$\frac{-0.02s^2-0.31s-0.03}{s^2+6.17s+9.14}$	$\frac{-0.02s^2-0.20s+0.41}{s^2+5.21s+6.57}$
G_{DE}	$\frac{-0.01s^2-0.08s-0.36}{s^2+6.55s+9.88}$	$\frac{0.01s^2+0.13s+0.20}{s^2+6.17s+9.14}$	$\frac{-0.01s^2-0.08s-0.17}{s^2+5.21s+6.57}$
G_{EE}	$\frac{0.031s^2+4.47s+8.97}{s^2+6.55s+9.88}$	$\frac{0.29s^2+4.11s+8.40}{s^2+6.17s+9.14}$	$\frac{0.25s^2+3.5s+6.31}{s^2+5.21s+6.57}$

Table 7.2: Second order wheeled mobile robot model

As shown in Fig. 7-3, the MIMO system is composed of a set of SISO subsystems with coupled connections. A set of reduced order dynamic transfer functions can be obtained if the coupling effects due to the influence between the two DC motors of the wheels can be described. Table 7.2 shows that the dynamics of the DC motors are different and the steady gains of the coupling terms are relatively small (less than 20% of the gains of main diagonal terms). Thus, it seems reasonable to neglect the coupling dynamics to obtain a simplified mode, and this has been verified experimentally by Pacheco, Ll.[73]. The functions considered for each velocity are summarized in Table 7.3, and following this assumption, we only consider the transfer functions of the left and right wheels directly, G_{EE} and G_{DD}.

Linear trans.function	High velocities	Medium velocities	Low velocities
G_{DD}	$\frac{0.95}{0.42s+1}$	$\frac{0.92}{0.41s+1}$	$\frac{0.82}{0.46s+1}$
G_{EE}	$\frac{0.91}{0.24s+1}$	$\frac{0.92}{0.27s+1}$	$\frac{0.96}{0.33s+1}$

Table 7.3: The reduced wheeled mobile robot model

7.5 Statement of the experiment

The aim is to design an intervalar PID controller using the IRCAD framework and implement this controller on a mobile robot (PRIM) to solve the problem of local path following.

The same experiment is also performed applying a classical PID controller. For PRIM, the classical controller must be considered a commuted controller, because there are three different models depending on the velocity (Section 7.4), i.e., for each model (speed) a different controller is applied.

The path-following algorithm

Path following is accomplished by tracking a sequence of fixed points consisting of positions and orientations. Feedback of orientations and cartesian coordinates reduces the path deviation. The required motion to follow the desired path is composed by a sequence of straight and turning actions. Figure 7-4 shows a path defined by three waypoints where the path between two consecutive waypoints is a straight line. To perform a straight path following action, both wheels must have the same velocity, and the heading angle must remain constant.

Right and left turning actions are performed by the following algorithms:

$$U_R(k+1) = U_R(k) + K_R(\theta_d - \theta(k)) \pm \Delta U_{PD}, \tag{7.4}$$

$$U_L(k+1) = U_L(k) + K_L(\theta(k) - \theta_d) \mp \Delta U_{PD}, \tag{7.5}$$

where U_R and U_L denote the required speed for the right and left wheels, respectively; θ_d denotes the desired orientation; θ is the WMR's heading (Fig. 7-4); and K_R and K_L are tuning factors for right and left turning of the wheels, respectively.

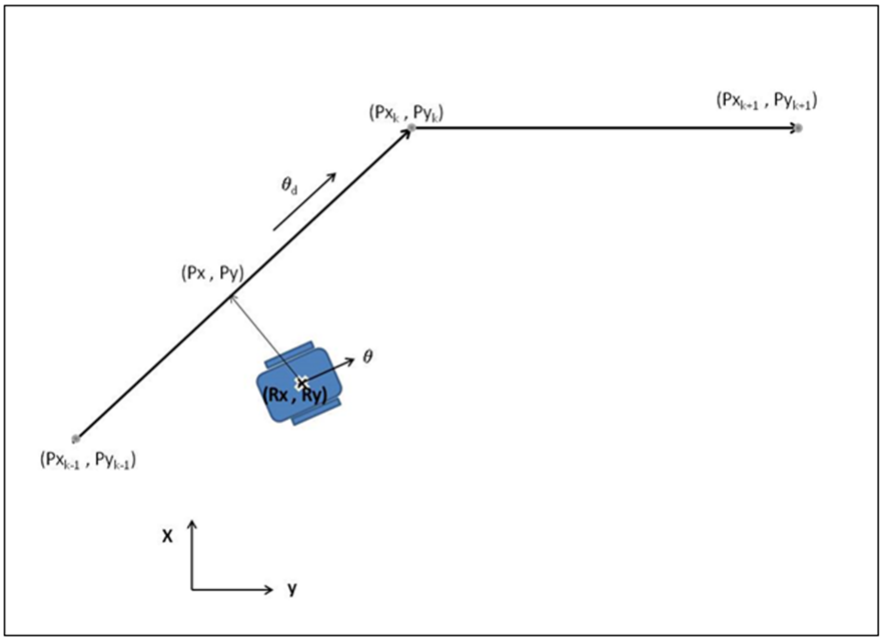


Figure 7-4: Example of a path described by a set of waypoints $(P_{x_{k-1}}, P_{y_{k-1}})$ (P_{x_k}, P_{y_k}) and $(P_{x_{k+1}}, P_{y_{k+1}})$

Thus, when a straight path is followed, the difference between θ_d and θ is zero, and straight path following is accomplished by commanding the same velocity to each wheel. When left turning is required, θ_d is larger than θ and Eqs. 7.4 and 7.5 increase U_R while U_L is decreased. Conversely, when right turning is required, θ is larger than θ_d and Eqs. 7.4 and 7.5 increase U_L while U_R is decreased. ΔU_{PD} is a turning parameter that reduces the path deviation when cartesian coordinates are considered. It is applied with different sign to both wheels with the aim of creating a difference of velocities. Path deviation is given by the Euclidean distance between the WMR coordinates, (Rx, Ry), and the straight line defined by the consecutive waypoints that define the path to be followed, (Fig. 7-4). ΔU_{PD} is considered an heuristic parameter:

- $\Delta U_{PD} = 0$ when the path deviation is very small.

- ΔU_{PD} produces a slight turn when the path deviation is small.
- ΔU_{PD} produces a turn when the path deviation is significant.

Euclidean path deviations are reduced by using turning actions that return the vehicle to the path. Thus, turning actions depend on the segment being followed and the relative WMR position.

The proposed algorithms consider five motion types to perform local path following: (straight, wide left turn, slight left turn, wide right turn, and slight right turn).

PRIM path following experiment

The path following experiment for PRIM was based on a path defined by a set of twelve points with a shape similar to a dodecagon, as shown in Fig. 7-5.

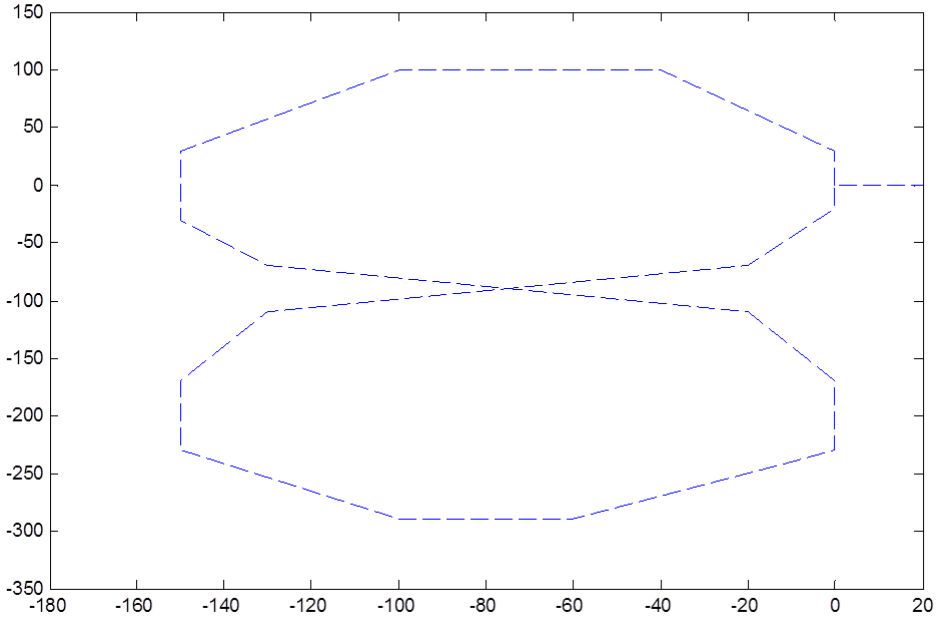


Figure 7-5: Clockwise dodecagon path

This path allows us to test both right and left turns with PRIM at different velocities. Thus, it is necessary to use different controllers during the experiment. To analyse path deviation, two consecutive points are joined by a straight line. When the WMR is close to a required point, a reduction in velocity is produced and the next point of the sequence is produced as the subsequent objective.

In following sections, we report on the experimental outcomes for a classical commuted PID controller and a controller based on intervalar techniques, developed using the proposed IRCAD framework.

The steps in designing the controller set using IRCAD were:

- Select the design specifications.

  A set of specifications was devised that the model must fulfill:

  - Setting time $\leqslant 2s$
  - Overshoot $< 10\%$

- Select a controller structure to achieve the specifications.

  To achieve the specifications, we chose a PI structure with a proportional and integral control actions:

  $$C(s) = K_P + \frac{K_I}{s}. \tag{7.6}$$

- Tune the controllers to obtain a continuous PI controller.

  The tuning process consists of computing the parameters, K_P and K_I, of the PI structure. This process could be executed in different ways, and the set of controllers (in the case of the desired commuted controller) allows the system to achieve the specifications.

- Convert the continuous controller to a discrete controller to be implemented on PRIM.

Discrete transfer functions were obtained by applying a transformation based on a zero order holder (zoh) method to the controllers transfer functions. These discrete functions were then ready to be applied to PRIM.

7.6 Classical commuted PI controller for PRIM

In Section 7.4, three different models were obtained for PRIM, for high, medium, and low velocities. Since each model requires a different controller, this implies a commuted controller for implementation.

Tuning the controllers requires computing the three PI continuous controllers then transforming them to discrete controllers to be implemented on PRIM. Pacheco, Ll. and Luo, N. [74] explain the processing detail, and the results are summarized in Tables 7.4 and 7.5 for the continuous and discrete controllers, respectively.

	High velocities	Medium velocities	Low velocities
Right wheel	$\frac{0.59s+3.36}{s}$	$\frac{0.7s+3.65}{s}$	$\frac{1.02s+4.59}{s}$
Left wheel	$\frac{0.04s+2.16}{s}$	$\frac{0.33s+2.4}{s}$	$\frac{0.33s+2.81}{s}$

Table 7.4: Continuous PI controller transfer functions

	High velocities	Medium velocities	Low velocities
Right wheel	$\frac{0.59-0.25z^{-1}}{1-z^{-1}}$	$\frac{0.7-0.54z^{-1}}{1-z^{-1}}$	$\frac{1.-0.54z^{-1}}{1-z^{-1}}$
Left wheel	$\frac{-0.04+0.54z^{-1}}{1-z^{-1}}$	$\frac{0.33s+2.4}{s}$	$\frac{0.33s+2.81}{s}$

Table 7.5: Discrete PI controller transfer functions

Considering PRIM to be modeled by the three transfer functions of the reduced WMR model, corresponding to the three velocities (Table 7.3), a commuted PI controller also

comprising three transfer functions (Table 7.5) is required to control the robot. In the high level code there is a switch structure with three options for the three velocities. Thus, the controller employed at any moment changes depending on the velocity.

7.6.1 Clockwise dodecagon path following for a commuted PID

The six control laws (three for each wheel), were input in the control code for clockwise dodecagon path following. The controller is commuted, depending on the robot velocity. PRIM was then activated to follow the path of the clockwise dodecagon, with the result as shown in Fig. 7-6, where the dashed trajectory is the set points, and the green trajectory corresponds to PRIMs actual path using the commuted PI controller.

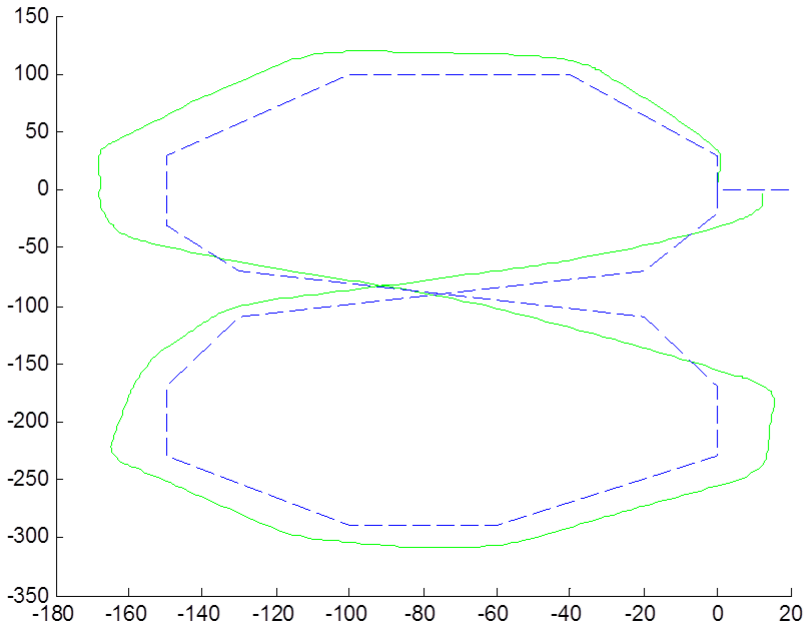


Figure 7-6: Clockwise dodecagon trajectory tracking using a PID commuted controller

7.7 Clockwise dodecagon path following for the proposed Intervalar based PI controller

IRCAD allows the control engineer to obtain a single controller rather than three commuted classical PID controllers (see Section 7.6).

This section shows how the intervals based controller is obtained and its path following performance tested. These results are compared with those obtained using a commuted PI controller following the same path in Section 7.8.

7.7.1 Obtaining the intervalar controller using IRCAD

The process to compute the intervalar controller is summarized in this section. An extended tutorial detailing the computation of the controller using IRCAD is presented in Appendix B.

Translation of the transfer functions to interval format

The first order transfer functions shown in Table 7.3 that represent the three reduced WMR models for each wheel are translated to a format suitable for IRCAD input, i.e., to interval format, as shown in Table 7.6.

	Left wheel	Right wheel
Interval transfer function	$G_{EE}(s,\mathbf{q}) = \frac{q_1}{q_2 s+1}$	$G_{DD}(s,\mathbf{q}) = \frac{q_1}{q_2 s+1}$
Interval parameter q_1	$q_1 = [0.91, 0.96]$	$q_1 = [0.82, 0.95]$
Interval parameter q_2	$q_2 = [0.24, 0.33]$	$q_1 = [0.41, 0.46]$

Table 7.6: Interval models for PRIM

Finding a set of feasible controllers for PRIM using IRCAD

To design the intervalar controller, we must provide the IRCAD framework with the following data (see Appendix B for details of how this data was obtained):

- Transfer functions of the model in interval format (Table 7.6).
- Interval values for the parameters of the intervalar transfer functions (Table 7.6).
- The specifications that must be fulfilled by the system when the controller is incorporated (Section 7.5).
- The structure of the controller. A PI controller structure was selected (Table 7.7, K_1 and K_2 have interval values).
- The parameter space of the controller parameters. The parameter space was $K_1 = [0, 100]$ and $K_2 = [0, 100]$, which corresponds to the region IRCAD searches for the set of feasible controllers.
- Selection of the solver. For this experiment we employed the solver developed in chapter 4.

	IRCAD controller
Right wheel	$\frac{K_1(1+\frac{s}{K_2})}{s}$
Left wheel	$\frac{K_1(1+\frac{s}{K_2})}{s}$

Table 7.7: Structure for the PI intervalar controller

Results from the intervalar controllers computed by IRCAD

The IRCAD design tools produce a sequence of algorithms following the process detailed in Appendix B. IRCAD provides a set of feasible controllers that fulfill the design spec-

ification in graphical (Figs. 7-7 and 7-8 for the right and left wheels, respectively), and numerical (Table 7.8) format .

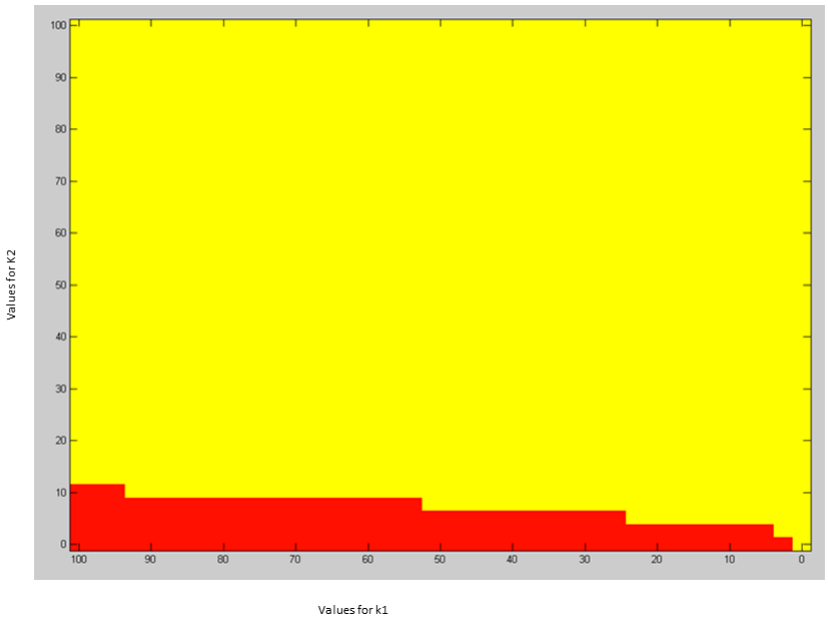


Figure 7-7: Set of feasible controllers of right wheel using IRCAD

The question remains, then, which is the best controller among all the possibles. This depends in the system to be implemented. IRCAD includes tools to assist the control engineer to make this selection. Three possible criteria may be employed in IRCAD (Section 5.3.3): neighboring, minimum norm, and maximum robustness. The proposed controller will be different depending on the criterion used, because each criterion emphasizes different aspects when choosing the best controller. It is important to note that the results of these criteria are not intervalar. The parameters of the recommended PI controller, returned by these criteria, are point real numbers, non-interval parameters. However, it is this format that the control engineer needs to obtain the information. The selected intervalar controller is summarized in Table 7.9.

	Set of feasible intervalar controllers
Right wheel	$K_1 = [2.5975, 100], K_2 = [0, 2.5975]$
	$K_1 = [5.095, 100], K_2 = [2.5975, 5.095]$
	$K_1 = [25.075, 100], K_2 = [5.095, 7.5925]$
	$K_1 = [52.5475, 100], K_2 = [7.5925, 10.09]$
	$K_1 = [92.5075, 100], K_2 = [10.09, 12.5875]$
Left wheel	$K_1 = [2.5975, 100], K_2 = [0, 5.095]$
	$K_1 = [0, 100], K_2 = [5.095, 7.5925]$
	$K_1 = [25.075, 100], K_2 = [7.5925, 10.09]$
	$K_1 = [50.05, 100], K_2 = [10.09, 12.5875]$
	$K_1 = [80.02, 100], K_2 = [12.5875, 15.085]$

Table 7.8: Set of feasible controllers

Criterion	PI controller for right wheel	PI controller for left wheel
Minimum norm	$K_1 = 3.8563, K_2 = 1.8488$	$K_1 = 3.8563, K_2 = 1.8488$
Neighboring	$K_1 = 61.2888, K_2 = 6.3438$	$K_1 = 53.7963, K_2 = 8.8412$
Maxim.robustness	$K_1 = 3.8563, K_2 = 1.8488$	$K_1 = 3.8563, K_2 = 1.8488$

Table 7.9: PI controllers proposed by IRCAD from the feasible set depending on the criterion selected

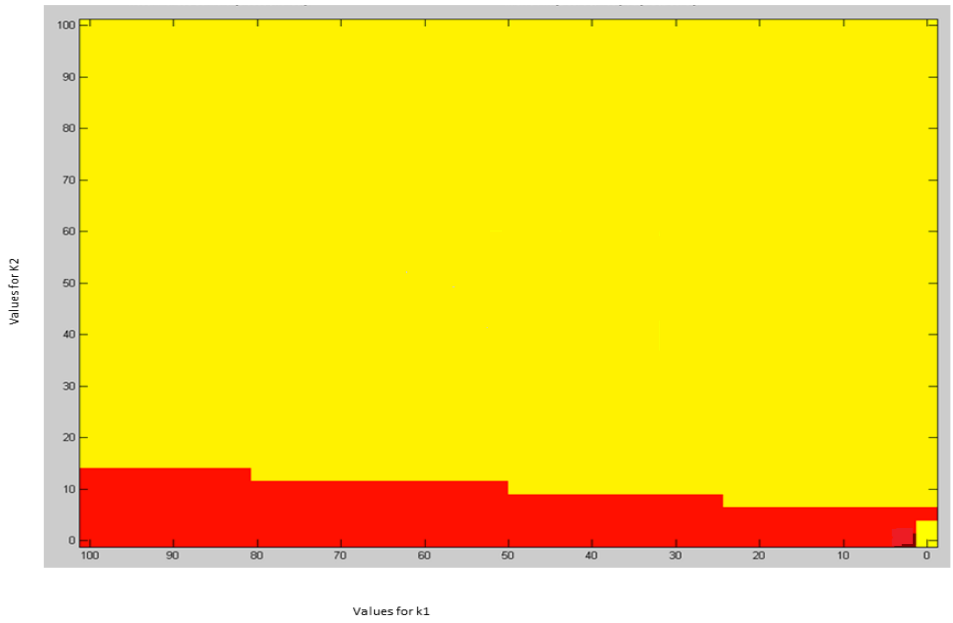


Figure 7-8: Set of feasible controllers for the left wheel using IRCAD

Validation of the controllers computed by IRCAD

IRCAD includes a set of tools that allow simulations of the system to verify the controllers. The simulation results for the right wheel are shown in Figs. 7-9 and 7-10 for the neighboring and minimum norm criteria, respectively. Figures 7-11 and 7-12 show the same results, respectively, for the left wheel.

The simulation responses are better with for controllers proposed using the neighboring criterion. Thus these controllers were implemented on PRIM to execute the experiments reported here.

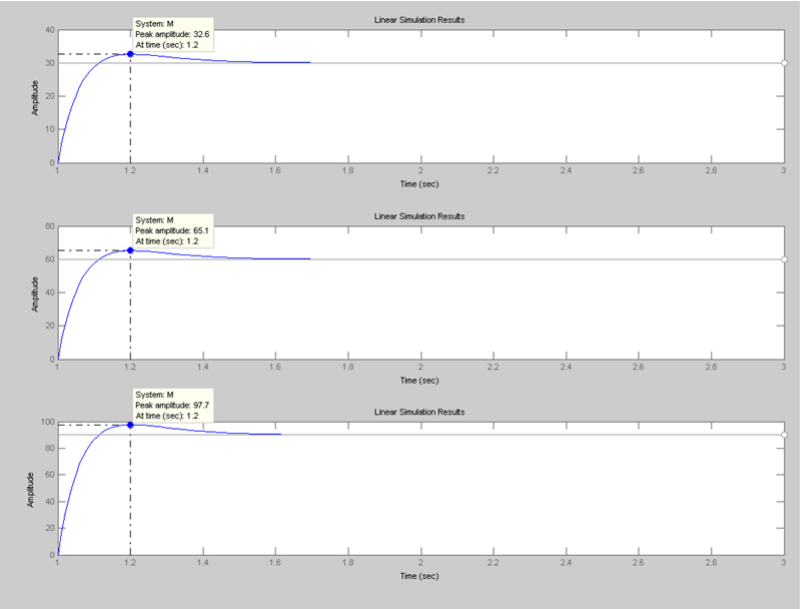


Figure 7-9: System response of the right wheel using the neighboring criterion

7.7.2 Clockwise dodecagon path following using the IRCAD PI controller

We implemented the controllers recommend by IRCAD onto PRIM to perform the path following task presented in Section 7.5.

To achieve this, we built the control law (the controller) and transformed it to C code to implement on PRIM. In contrast to the implementation of Section 7.6, which included three different controllers (one for each velocity), this case there is a single controller covering all the velocities (see Appendix B for details)). Table 7.10 shows the selected controllers, computed using the neighboring criterion for each wheel. Implementing the controller structure of Table 7.7, the final controllers for each wheel are shown in Table 7.11, and the discrete format controller is shown in Table 7.12

Thus, the control laws for each wheel as function of the error, $e(z)$ are as follows.

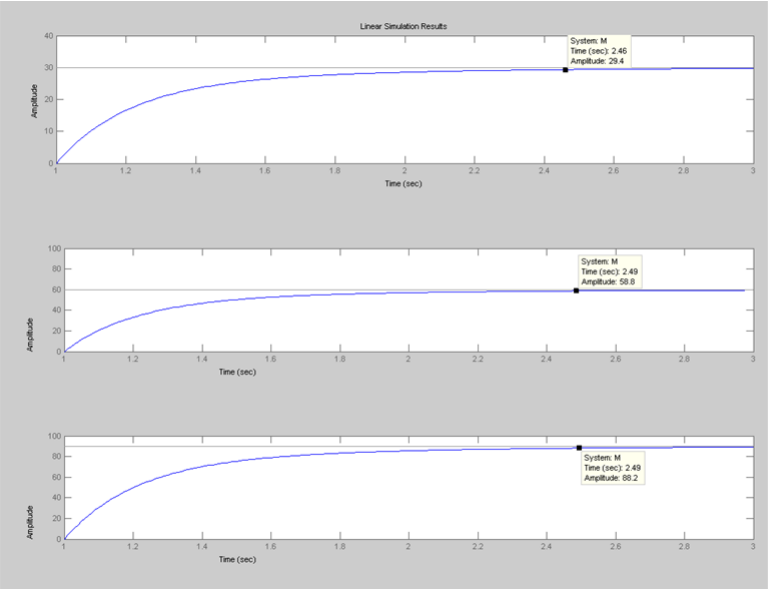


Figure 7-10: System response of the right wheel using the minimum norm criterion

Criterion	PI controller for right wheel	PI controller for left wheel
Neighboring	$K_1 = 61.2888, K_2 = 6.3438$	$K_1 = 53.7963, K_2 = 8.8412$

Table 7.10: Selected PI controllers ("NEIGHBORING" criterion)

For the right wheel

$$u(z) = u(z-1) + 9.67e(z) - 3.54e(z-1), \tag{7.7}$$

and for the left wheel,

$$u(z) = u(z-1) + 6.09e(z) - 0.71e(z-1). \tag{7.8}$$

These two control laws were input in the C code built for clockwise dodecagon path following, with results as shown in Fig. 7-13.

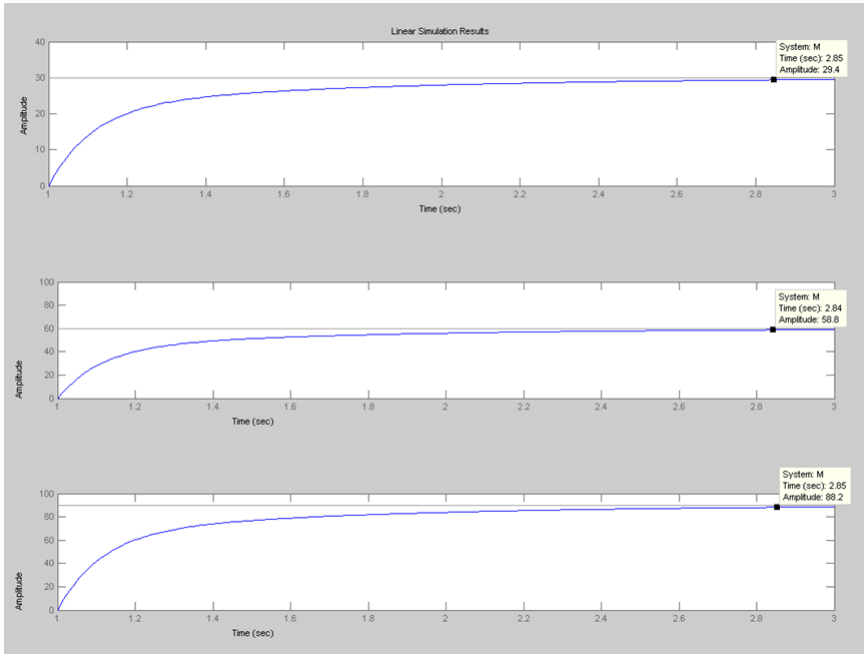


Figure 7-11: System response of the left wheel using the neighboring criterion

7.8 Comparing the commuted and intervalar PI controllers

The proposed IRCAD controller implementation produces a resultant path (Fig. 7-13) very similar to that of the commuted controller (Fig. 7-6).

Figure 7-14 directly compares these paths to allow a more accurate assessment. The dashed green line shows the path when using the commuted PI controller, the dashed

	IRCAD controller
Right wheel	$\frac{9.67s+61.3}{s}$
Left wheel	$\frac{6.09s+53.8}{s}$

Table 7.11: Continuous PI controller obtained from IRCAD

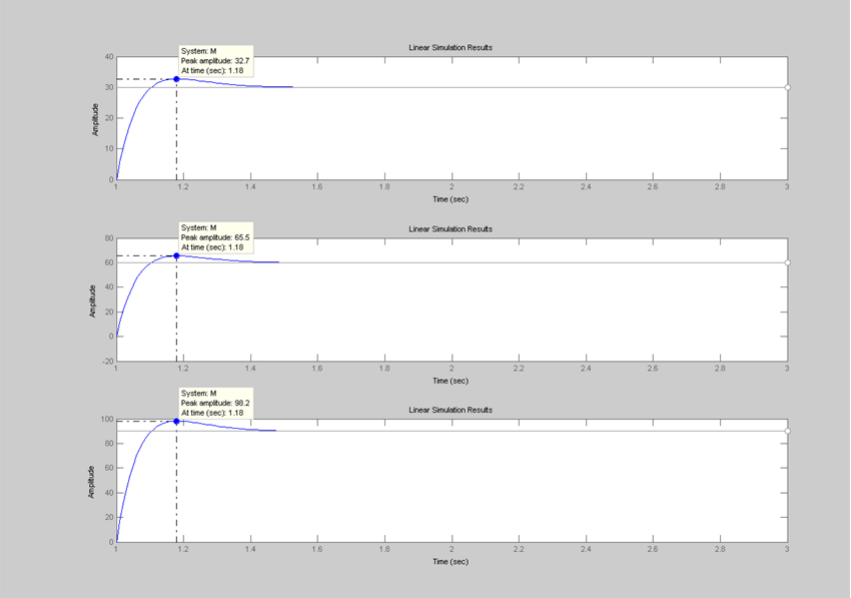


Figure 7-12: System response of the left wheel using the minimum norm criterion

red line is that when using the IRCAD PI controller. Path tracking accuracy is slightly improved using the commuted controller.

	IRCAD controller
Right wheel	$\frac{9.67-3.54z^{-1}}{1-z^{-1}}$
Left wheel	$\frac{6.09-0.71z^{-1}}{1-z^{-1}}$

Table 7.12: Discrete PI controller obtained from IRCAD

Figure 7-15 concentrates on a single segment of the path the robot must to follow corresponding to a right turn, and the specific responses of the left and right wheels are shown in Figs. 7-16 and 7-17, respectively. Control of the left wheel is less abrupt using

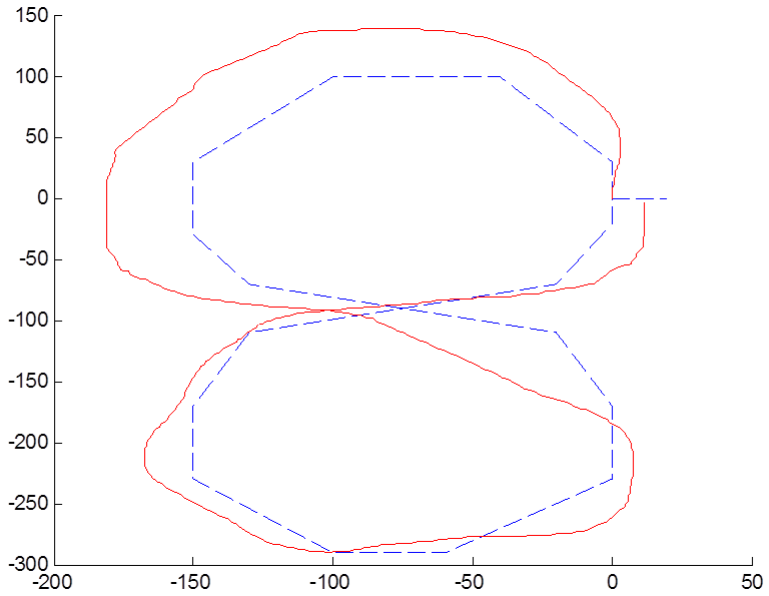


Figure 7-13: Clockwise dodecagon path following using a PID controller based on intervalar techniques

the intervalar based controller, whereas that of the right wheel is less abrupt using the commuted controller. However, PRIM follows the set point with good accuracy regardless of the controller used.

Figure 7-18 shows the path error from the two controllers. Overall, the path error resulting from the intervalar controller (red) is superior to that from the commuted controller (green), especially in the case of the left wheel.

Figure 7-19 shows the control signal generated for PRIM as it executed the path following experiment. The signal is smoother for the intervalar controller (red) than the commuted controller (green), especially for the left wheel.

Finally, the time required for PRIM to complete the dodecagon path was similar (approximately 70 s) for both control strategies.

In addition to the quantitative improvements using the proposed IRCAD controller (improved error and smoother control), there are several qualitative improvements:

- A clear gain in simplicity and ease of design, due to the IRCAD framework.

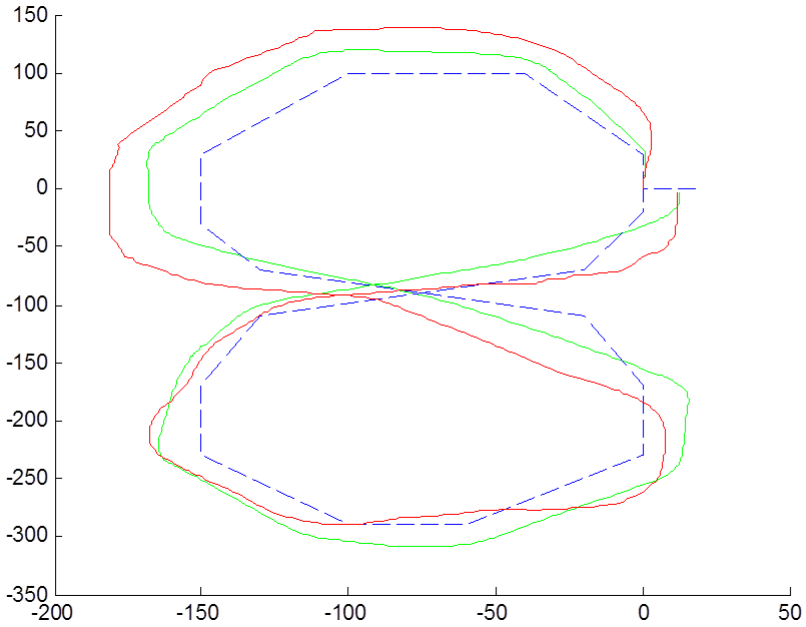


Figure 7-14: Clockwise dodecagon path following using a commuted controller (dashed green line) and an intervalar controller (dashed red line)

- Easier to validation of candidate controllers, because a single controller is implemented rather than one controller for each velocity.
- Easier to implement into high level code and more time effective.

The proposed IRCAD framework facilitates control of a complex system, such as the robot PRIM, with a single controller whatever PRIMs velocity. In contrast, the commuted PI controller is necessarily more complex because the control law changes with PRIMs velocity.

Thus, the interval controller is a valid implementation for PRIM to solve the path following problem, and is an improvement over the (classical) commuted PI controller.

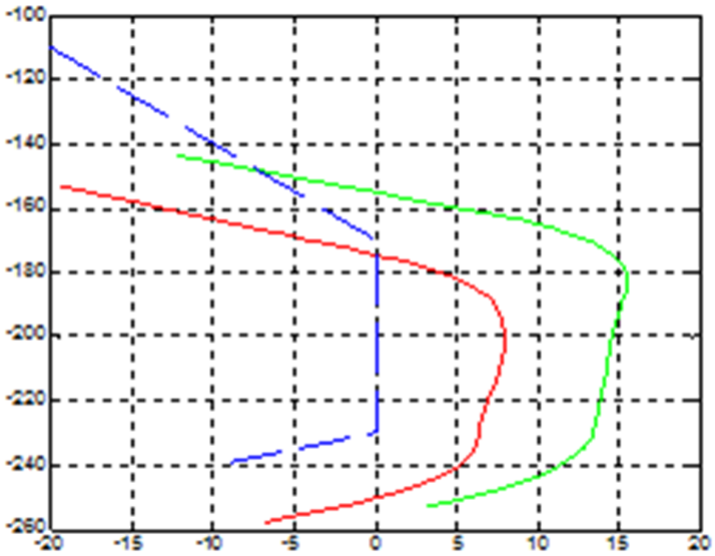


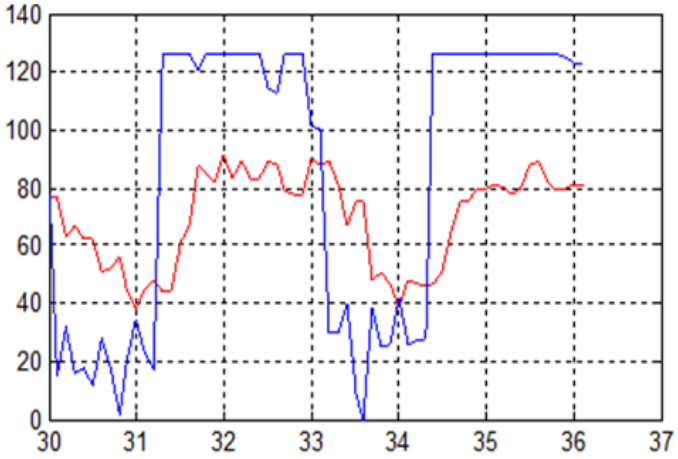
Figure 7-15: Trajectory of a right turn using two control strategies

7.9 Summary

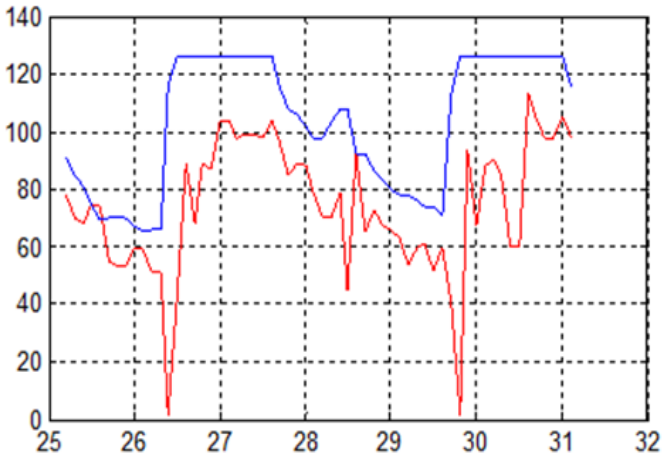
We have shown the application of controllers obtained using the design tool of the framework IRCAD to a real system, the robot PRIM. Two controllers were obtained, for the left and right wheels.

An experiment consisting of clockwise path following around a dodecagon form was performed and analysed. This served to illustrate the process to design a controller using the IRCAD framework and also how to manage the interval information during the design process.

The control laws obtained from IRCAD were verified in simulation then implemented on PRIM and path following performance was compared with a classical commuted controller. Path following was a little more accurate using the commuted PID. However, overall the simplicity of the IRCAD single controller and its smoother control signal is a significant improvement over the commuted controller.

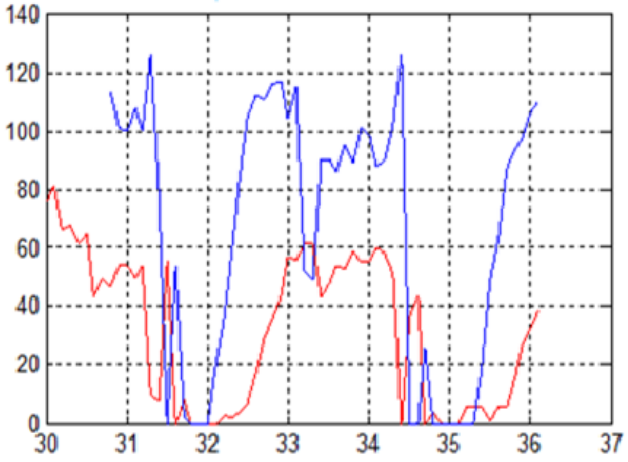


(a) Response using the controller obtained from intervalar techniques

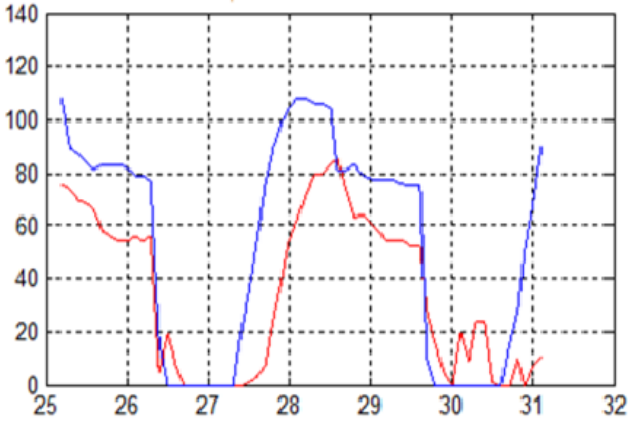


(b) Response using the commuted controller

Figure 7-16: Left wheel response during a right turn



(a) Response using the controller obtained from intervalar techniques



(b) Response using the commuted controller

Figure 7-17: Right wheel response during a right turn

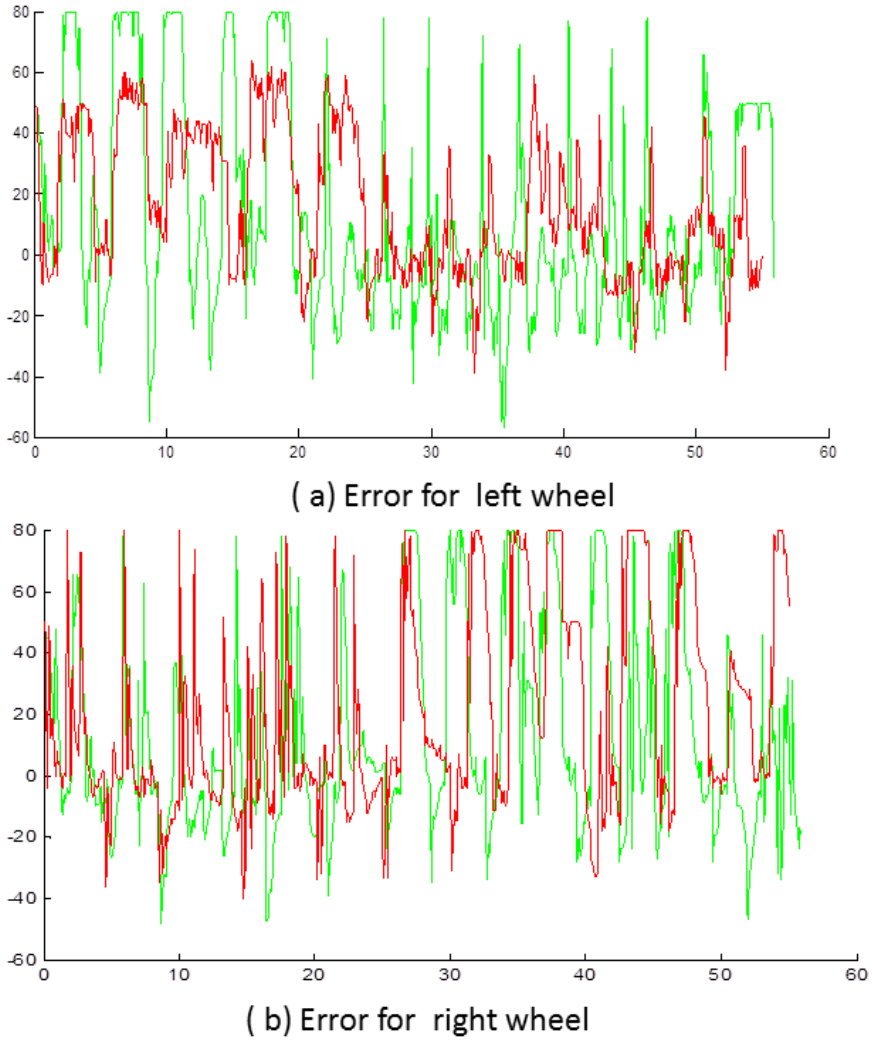


Figure 7-18: Error for clockwise dodecagon path following using the PID commuted and IRCAD controllers

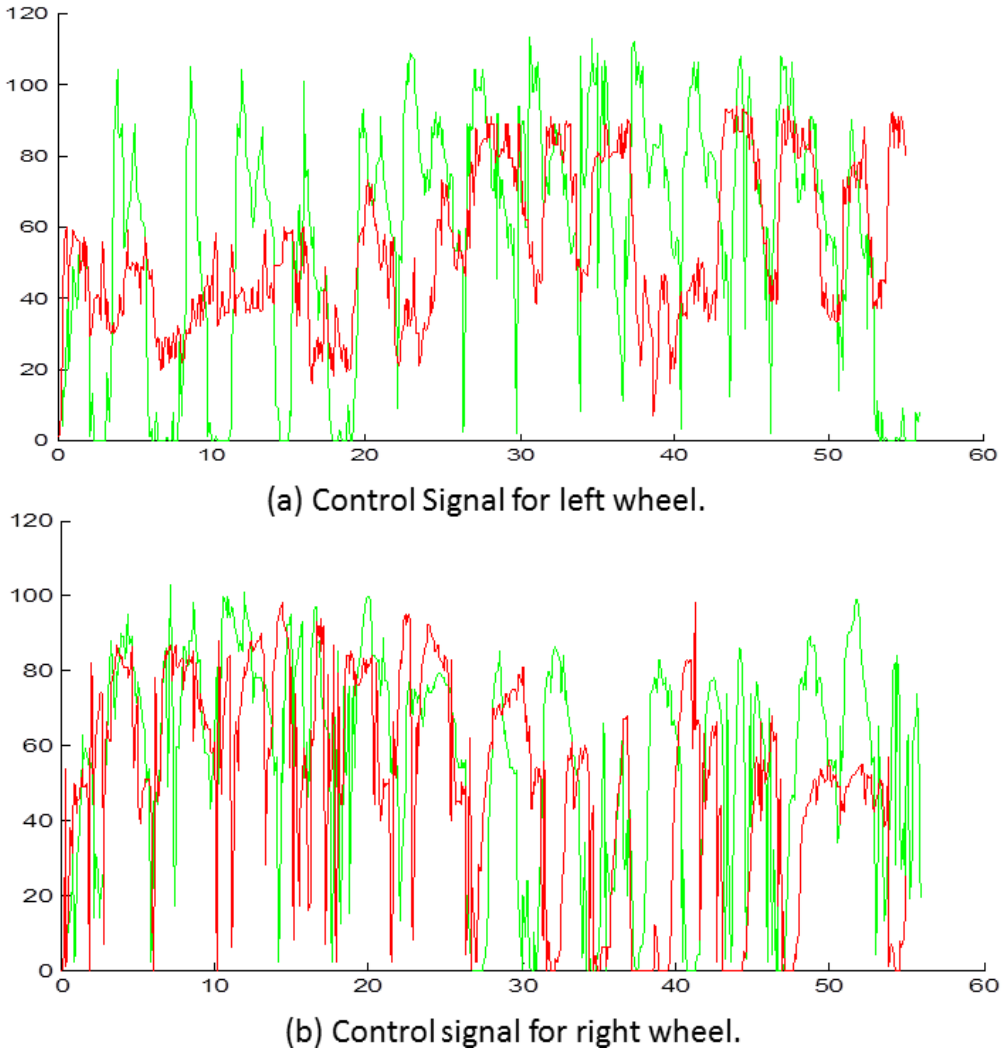


Figure 7-19: Control signal for clockwise dodecagon path following using the PID commuted and IRCAD controllers

Chapter 8

Conclusions and further research

We present a general summary of the outcomes and a specific statement of the original contributions of this thesis. Some directions for future research on robust control are also discussed.

8.1 Summary

This work proposes a solution to solve robust control problems using interval parametric models. Complex systems are often subject to uncertainties that make modeling difficult, if not impossible. A quantitative model may be inadequate to represent the behavior of systems which require an explicit representation of imprecision and uncertainty. Assuming the uncertainties are structured, models can be constructed using with modal interval methods where the parameters are allowed to vary within numeric intervals. Robust control uses such mathematical models to explicitly incorporate uncertainty. Solving robust control problems, such as finding the robust stability or designing a robust controller, involves difficult symbolic and numeric computation. However, if interval models are employed, this allows interval computations. The main advantage of using modal interval analysis is that it provides guaranteed solutions, but as drawback its requires interaction with multiple data types. We propose a methodology and framework that combines sym-

bolic and numeric computation with interval analysis to solve robust control problems.

8.2 Original contributions of this work

The original contributions of this thesis work are:

- A control toolbox for systems with parametric uncertainties.

  A set of tools based on interval methods to solve robust control problems, in the case of parametric uncertain systems. These methods are based on Modal Interval Analysis.

- Integration of symbolic, interval, and numeric data.

  - The IRCAD toolbox, developed using the GUI tools of Matlab, allows user-friendly input of the system transfer functions in a symbolic manner, with easy input of the numerator and the denominator from their respective polynomials.
  - The IRCAD toolbox allows input of interval values for the uncertain parameters of the transfer functions.
  - Development of a matrix structure to pass the set of constraints (symbolic data) obtained from Matlab to the solver (a C++ executable program) that requires numeric data.
  - The solver returns interval and numeric data that are converted to symbolic to be displayed in the Matlab environment.

- Build uncertain constraints to solve robust control problems.

  Given a robust control problem, the IRCAD toolbox transforms it into a function (or constraint) with interval parameters. Thus, the toolbox reduces a set of different problems (shown in the following list) to a unique constraint.

  - Analysis. Build the constraint corresponding to testing for:

    - open loop stability, and
    - Closed loop stability.
  - Design. Find the set of possible controllers that fulfill the defined specifications, and build the constraint for that specification:
    - Stability degree.
    - Absolute stability.
    - Resonance peak.
    - Velocity error.
    - Settling time 2%.
    - Control effort.
    - Overshoot.
  - Stability margin. Build the algorithm corresponding to the computation of the stability margin.
  - Parametric bode plot. Build the constraints corresponding to the phase and magnitude interval functions.

  Once these constraints are obtained, the problems are reduced to check positivity.

- Application of sorting techniques to check multiple uncertain constraints (specifications) at the same time.
- Use of Modal Interval Analysis methods to build algorithms used by the IRCAD framework.
  - Development of a solver. This algorithm is used to check the positivity of a function.
  - Stability conditions.
  - Parametric Bode plot representation. Use the f* algorithm to solve the parametric Bode representation.

- Capability of the toolbox to incorporate existing solvers based on parametric uncertainties (section 5.2.3). This allows the toolbox to be used as an open framework.

8.3 General Conclusions

The main goal of this thesis has been achieved: to build a framework for robust control analysis and design to deal with systems that involve uncertain parametric models with the parameters modeled by intervals. The robust control tools included in the proposed IRCAD framework provide and improvement over other interval based robust control frameworks through the incorporation of state of the art modal interval analysis. Many researchers working with parametric models use interval models to represent the uncertainty, but use classical interval analysis to build their models. This has the drawback of producing solutions that cannot be guaranteed valid for the problem, whereas model interval analysis applied to parametric interval models provides guaranteed valid solutions.

8.4 Further work

This section illuminates research areas that remain open or have been instigated by this thesis.

- 3-Dimensional parameter space. The IRCAD framework is currently limited to controllers with 2 parameters. Extending the framework dimensionality, and providing an appropriate graphical method to represent the feasible controllers on parameter spaces of the extended dimensionality, would facilitate the application to many real-world problems

- Extension to discrete systems. Discrete systems deserve a full and thorough study.

- Application to other problems of control engineering. The IRCAD framework could

include tools for predictive calculation horizon for uncertain systems, controllability, observability, etc.

- Intervalar simulation. Giving a family of intervalar systems, the IRCAD framework could incorporate a tool to compute MIA based envelopes of their response.

- Application of contraction techniques to reduce the initial controller parameter space. To obtain the set of controllers that fulfill some specifications, it is necessary to input the initial interval values of the parameter space to start the search. Frequently, these data are difficult to ascertain. Thus, applying contraction techniques localizes and reduces the initial parameter space.

Appendix A

Routines for the developed solver. SCDS algorithm

Description of the appendix

In this appendix are presented the essential components of the SCDS algorithm, showing the principal functions of the main procedures which comprise the algorithm in a schematic pseudocode format. Stability conditions included in the procedures are discussed, emphasizing their importance in the overall toolbox building process.

Routines

There are three C++ routines in the SCDS structure, **Main, Ivalunix** and **Interes2_2**, as shown in Fig. A-1, which must be compiled to obtain the final executable.

1. **Main**

    - *Definition*: This is the main procedure.
    - *Function*: Collect the input data from the interval, numeric, or symbolic matrices and pass them as a parameters to **Interes2_2**. Collect the outcomes into a matrix accessible from Matlab.

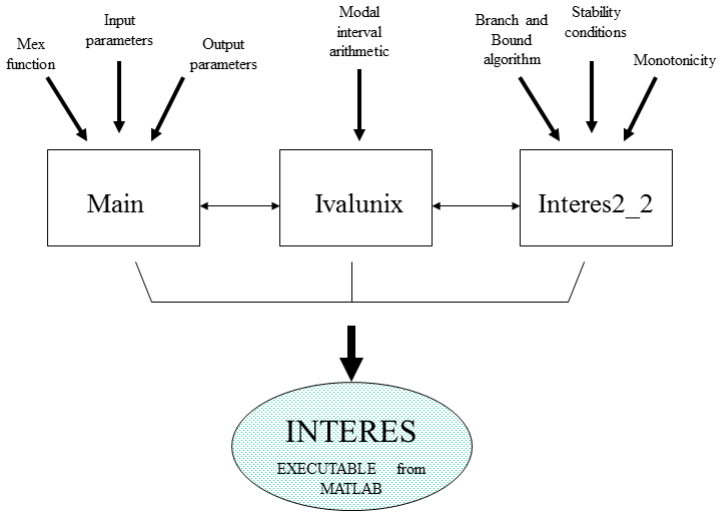


Figure A-1: Interval algorithms

- *Structure*: The body of this routine is located inside a mexFunction structure with the format:

  void mexFunction

  (a) Assign the input parameter matrices obtained from the Matlab environment to local variables.

  (b) Use the **Interes2_2** routine without any symbolic data.

  (c) Pass the output parameter matrix returned by **Interes2_2** to the Matlab environment.

  end mexFunction

2. **Ivalunix**

- Definition: This is the procedure that contains the modal interval arithmetic.
- Function: Because we are working with uncertain models where the parameters are usually defined by intervals, it is necessary to have a tool to perform interval operations.
- Data Structure: All the parameters declared by Ival <FL> are intervals. They

are composed of a lower bound (Infim) and an upper bound (Suprem), which can be accessed in two forms: for the lower bound (interval.ii or Inf(interval)) and for the upper bound (interval.ss or Sup(interval))

- Structure: Each interval function is defined as a template. The relationships between the most important functions included in this interval arithmetic are as follows:
  - Unary operators.

    Ival<FL> operator-(const Ival <FL>);

    Ival<FL> Inf(const Ival<FL > a);

    Ival<FL> Sup(const Ival<FL > a);

    Ival<FL> Du(const Ival<FL >);

    Ival<FL> Pro(const Ival<FL >);

    Ival<FL> Impr(const Ival<FL >);
  - Relational.

    operator<=(const Ival<FL>, const Ival<FL>);

    operator>=(const Ival<FL>, const Ival<FL>);

    operator<(const Ival<FL>, const Ival<FL>);

    operator>(const Ival<FL>, const Ival<FL>);

    operator<<(const Ival<FL >, const Ival<FL>);

    operator>>(const Ival<FL >, const Ival<FL>);

    operator==(const Ival<FL>, const Ival <FL>);
  - Binary inner operators.

    Ival<FL> operator+(const Ival <FL>, const Ival<FL>);

    Ival<FL> operator-(const Ival <FL>, const Ival<FL>);

    Ival<FL> operator(const Ival <FL>, const Ival<FL>);*

    Ival<FL> operator/(const Ival <FL>, const Ival<FL>);

    *Ival<FL> operator&&(const Ival<FL>, const Ival<FL>); /**Meet**/*

    *Ival<FL> operator|| (const Ival<FL>, const Ival <FL>); /**Join**/*

Ival<FL> operator^(const Ival <FL>, int);

- Lineal operations for the sum and the product. Constructed overloading standard C++ operators:
  Lineal sum ==> +=
  Lineal product ==> *=

- Interes2_2
  - Definition: This is the procedure containing the body of the control actions: stability check and so forth.
  - Function: In summarizing the function of this routine, it can be said that it takes the transfer function, corresponding with problem specifications, and checks it over the whole range of interval variation input parameters - both transfer function and interval parameters are passed as input parameters of the routine.
    To check all parameter space, an algorithm of the branch-and-bound family is used, implemented in the 'splitty' function. In addition, monotonicity and stability conditions are considered to help the checking. As a result, the routine gives a matrix with the full parameter space classified in three regions: stable regions, unstable regions and undefined regions.
  - Data structure: Each element obtained in the branch-and-bound process is defined as a structure with format:
    * Ivfloat variab[...] /* Interval defined by its lower and upper bounds */
    * int nvar[...] /* number of variables */
    * int monot[...] /*monotonicity of the first order derivatives */
    * int monot2[...][...] /*monotonicity of the second order derivatives */

    The result obtained by checking each box over the parameter space is defined as a structure with format:
    * Ivfloat x[...] /* first interval of the box */

* Ivfloat y[...] /* second interval of the box */
* int type[...] /* type of result: positive, negative or undefined */

– Structure: This routine is composed of a set of functions. Among them, **interes** is the function called by the **Main** routine which, in turn calls on all the others. The routines and their functions are:

(a) void **interes** (...parameters...)

i. Compute first differentiation matrices. /* The result is a set of matrices */.

ii. Compute second differentiation matrices. /* The result is another set of matrices */.

iii. *.for (x=0; x<lim1; x += in1)*
for (y=0; y< lim2; y+=in2)
{
(Give the values of the parameter variation range to the symbolic variables to check each region, then put each one inside an array of nItem elements).
}

iv. Call on the recursive function **calcula** to check full parameter space. /* The result returned by this function is a vector containing the checking result for each box of the full parameter space */

v. Convert the result vector in a vector of colors(**color**). /* The vector is the return parameter of this routine, ready to be interpreted and plotted by Matlab */

end **interes**

(b) void **calcula** (...parameters...)
for (i=0; i < nItems-1; i++)/* for each member of the array */

i. Initialization of the node to be checked.

ii. Initialization of the monotonicities of the first order derivatives.

iii. Initialization of the monotonicities of the second order derivatives.

iv. items[i].type =**buclePrincipal** (...parameters...). /* Call on the recursive function buclePrincipal to check the stability. */

end **calcula**

(c) void **buclePrincipal** (...parameters...)

for (i=0; i < nItems-1; i++) /*for each member of the array*/

i. ypmeani = **CalculaDerivades** (... parameters...)/* compute function in the center of the interval */

ii. yp = (**avalua1** (... parameters...&& ypmeani) /* Meet between the value of the function and the value of the function on the center of the interval */

iii. yl = **avalua_lin** (... parameters...) /* compute function with lineal operations */

iv. If it is possible to determine the stability with this information,

est = ***assignaEstabilitats*** *(val, yp, yl, Sup(ypmeani))* /*it is assigned to the variable est*/

return ***est*** /* return **est** and end this routine */

v. Else /* Check the possibility of iterating another time */

{

Call the function **splitty** /* to divide the current interval in two */

Call the function ***bucle Principal*** *recursively with the first interval obtained by splitty*

Call the function ***bucle Principal*** *recursively with the second interval obtained by splitty*

If the results in both cases check positive

return **1** /* return positive and end this routine */

end **buclePrincipal**

(d) void **calculaDerivades** (...parameters...)

i. Compute the second order derivatives.

ii. Assign monotonicities brought about by second order differentiation.

iii. Compute the function taking the values of the center of the interval ypmeani.

end **calculaDerivades**

(e) void **assignaEstabilitats** (...yp, yl, ycentre...) /* Three input parameters are needed to check the stability yp=interval result to compute the function //, yl=interval result to compute the function with lineal operations //, ycentre=interval result to compute the function using the interval center and the centered form */

i. *If (yp.ss <=0 | | (yp.ii <=0 && num_mons == numvar) || (ycentre<=0 || (yl.ii <= 0 || yl.ss <=0))*

ii. return -1; /* unstability cases */

iii. *If (yp.ii >0 && yp.ss > 0)*

iv. return 1; /* stability cases */

v. else Return 0; /* undefined cases */

end **assignaEstabilitats**

(f) void **splitty** (...parameters...) /* A type of branch and bound algorithm */

i. Compute the largest interval in the set of interval variables.

ii. Split this interval down the center into two intervals without touching the others.

iii. These two smaller squares of parameter space are returned to be checked recursively.

end **splitty**

Stability conditions

A good definition of stability and instability conditions for the full parameter space is considered a goal and also a bottleneck in the algorithm. The more accurately they are defined, the earlier a stability condition can be decided. When a box composed of interval variables that bound variates throughout the full parameter space is evaluated in a first iteration for the SCDS algorithm, the result can be:

- Stable. —> The evaluation of the box is finished.
- Unstable. —> The evaluation of the box is finished.
- Undefined. —> The stability conditions checked by the algorithm (SCDS or other) are not sufficient to decide if the full box is stable or unstable, so the algorithm continues to divide the box into two parts with the **splitty** algorithm and recursively check the stability of both parts.

The undefined case usually generates a numerous set of sub-boxes, each of which need their stability to be examined, which rapidly increases the depth of recursion. To solve this bottleneck, the algorithm includes a depth limit to avoid infinite recursion. When there is a reduced number of variables, the results are not very important, but when the size of the problem increases (five or more variables) it is essential to decide the stability condition as quickly as possible. To achieve this, the algorithm must include:

- Speed: The number of iterations needed to define a condition is reduced. In this sense, the most expensive computation time is caused by the undefined case.

- Efficiency: If stability is not determined by the recursion depth limit, the algorithm returns undefined, whereas the ideal situation is to have stability defined across the full parameter space.

A first approach to the algorithm considered the result of computing a function (corresponding to one specification problem) whose modalities had previously been changed (cases in which it would be necessary), and computing the monotonicity. The conditions obtained were:

- Unstable cases

    1. (yp.ss <= 0) /* When Suprem of the specification function is <= 0, without checking the Infim */
    2. (yp.ii <= 0 && num_mons == nvar) /* In the case that all the variables are monotonous and the Infim of the function specification is <=0 */

- Stable cases

    1. (yp.ii > 0 && yp.ss > 0) /* When both Infim and Suprem of the specification function are > 0 */

Considering only these cases, the computation time for five variables is very long and the number of undefined boxes at the end is too large. To improve the algorithm, it appears necessary to compute the original function using the centered form interval tool. This form gives an interval which is added to the interval center obtaining an interval called, in the SCDS algorithm, *ypmeani* . The conditions added to these new results mean a refined find of unstable boxes.

- Unstable cases

    1. ypmeani.ss <= 0)/* When the Suprem of the interval computed by the centered form is <= 0 */

2. To further improve efficiency, the algorithm computes the original function (corresponding to a specification problem) using lineal rather than interval operations. This computation produces an interval, yl, which adds the following constraint to instability cases:

   (yl.ii <= 0 || yl.ss <= 0) /* When one or both functions with lineal operations are <= 0 */

Comparing the initial case, where only the first two conditions of instability are used, with the final version of the SCDS algorithm, where the four conditions are used, over an example of five variables, the computation time was reduced by almost 40%. However, efficiency depends not only on stability conditions, but also on the right choice of recursion depth limit to accomplish the stability test of the full parameter range with the minimum number of undefined boxes.

Appendix B

Application of IRCAD tools to PRIM

Description of the appendix

In this appendix chapter are presented the steps to follow in order to use IRCAD Design tools. It wants to be used as manual of execution, guiding control engineers to use design tools of IRCAD. The example illustrated is robot PRIM.

Translation of the transfer functions to interval format

The first-order transfer functions shown in Table 7.3, representing the three reduced WMR models for each wheel, must be translated to a format suitable for the IRCAD framework.

The parameters of the left wheel transfer function (G_{EE}) are assumed to be uncertain variables, q_1 and q_2, and the interval transfer function is then

$$G_{EE}(s, \mathbf{q}) = \frac{q_1}{q_2 s + 1},$$

where q_1 and q_2 are uncertain parameters with values that include high, medium, and low velocities (table 7.3), i.e.,

$q_1 = [0.91, 0.96]$

$q_2 = [0.24, 0.33]$

Similarly, the right wheel interval transfer function is

$$G_{DD}(s, \mathbf{q}) = \frac{q_1}{q_2 s + 1},$$

where in this case

$q_1 = [0.82, 0.95]$

$q_2 = [0.41, 0.46]$

Table B.1 summarizes these format changes for PRIM.

	Left wheel	Right wheel
Interval transfer function	$G_{EE}(s, \mathbf{q}) = \frac{q_1}{q_2 s+1}$	$G_{DD}(s, \mathbf{q}) = \frac{q_1}{q_2 s+1}$
Interval parameter q_1	$q_1 = [0.91, 0.96]$	$q_1 = [0.82, 0.95]$
Interval parameter q_2	$q_2 = [0.24, 0.33]$	$q_1 = [0.41, 0.46]$

Table B.1: Interval models for PRIM

Finding a set of feasible controllers for PRIM using IRCAD

One of the most important utilities of the IRCAD framework is to assist control engineers to design a controller that satisfies the predetermined system specifications. To design the controller, the following data must be input to the framework:

- Transfer function of the model in interval format
- Interval values for the parameters of the intervalar transfer function.
- Definition of the system specifications that must be fulfilled.
- A structure for the controller.
- The region of parameter space for the controller parameters.

Transfer function in interval form.

The previous sections prepared the parametric models for PRIM in a suitable format to be used by IRCAD. These transfer functions on the IRCAD framework are:

$$G_{EE}(s, \mathbf{q}) = \frac{q_1}{1 - \frac{s}{q_2}}, \tag{B.1}$$

and

$$G_{DD}(s, \mathbf{q}) = \frac{q_1}{1 - \frac{s}{q_2}}. \tag{B.2}$$

Interval values for the parameters of the interval transfer function.

The interval values, q_1 and q_2, of the two transfer functions are defined in Table B.1. The values were computed by merging the set of transfer functions from Table 7.3.

In the case of the left wheel,

- High velocity: $q_1 = 0.91$
- Medium velocity: $q_1 = 0.92$
- Low velocity: $q_1 = 0.96$

Thus, $q_1 = [0.91, 0.96]$.

For q_2,

- High velocity: $q_1 = 0.24$

- Medium velocity: $q_1 = 0.27$
- Low velocity: $q_1 = 0.33$

Thus, $q_2 = [0.24, 0.33]$.
In the case of right wheel,

- High velocity: $q_1 = 0.95$
- Medium velocity: $q_1 = 0.92$
- Low velocity: $q_1 = 0.82$

Thus, $q_1 = [0.82, 0.95]$.
For q_2

- High velocity: $q_1 = 0.42$
- Medium velocity: $q_1 = 0.41$
- Low velocity: $q_1 = 0.46$

Thus, $q_2 = [0.41, 0.46]$.

Specifications that the system must fulfil

In any problem of design is necessary to define a set of specifications that the system, robot PRIM in this example, must fulfill when the controller is incorporated. As is detailed in section 7.5, the selected specifications are

- Overshoot $\leq 10\%$, and
- Settling time within $2\% \leq 2sec.$

These specifications are introduced to the framework, as shown in Fig. B-1.

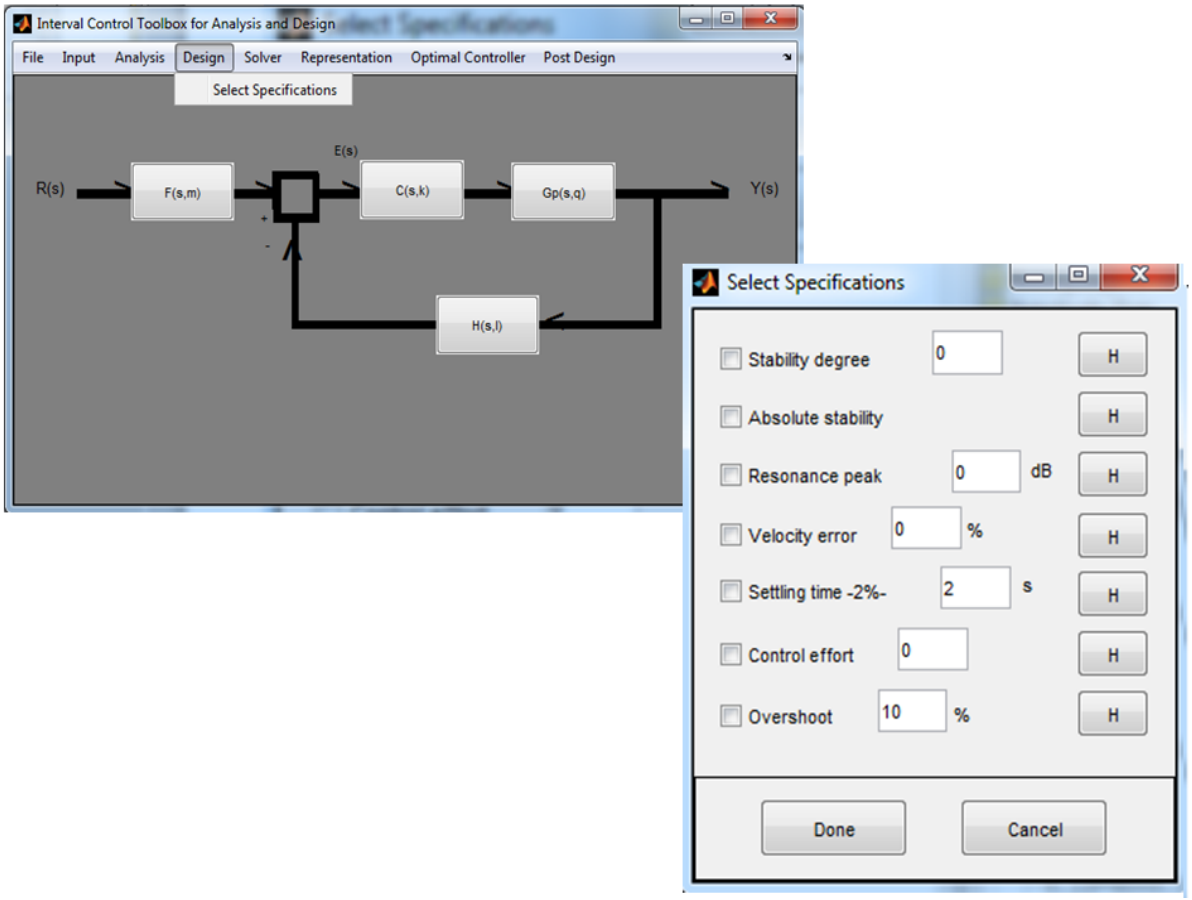


Figure B-1: Selection of the specifications using IRCAD

A structure for the controller

The IRCAD design tool requires a PID structure to be chosen, and allows the user to define the structure of an intervalar controller in an easy manner. In the robot PRIM example, the controller structure includes two specifications: a proportional action and an integral action, i.e., a PI controller. The same controller was also used for the commuted controller (Section 7.6), which was used as the comparator.

The transfer function for the two controllers was input to IRCAD (Fig. B-2) in the

format :

$$C(s, \mathbf{k}) = \frac{k_1 \left(1 + \frac{s}{k_2}\right)}{s}, \tag{B.3}$$

where $\mathbf{k} = [k_1 \ k_2]^T$ is the design parameter vector.

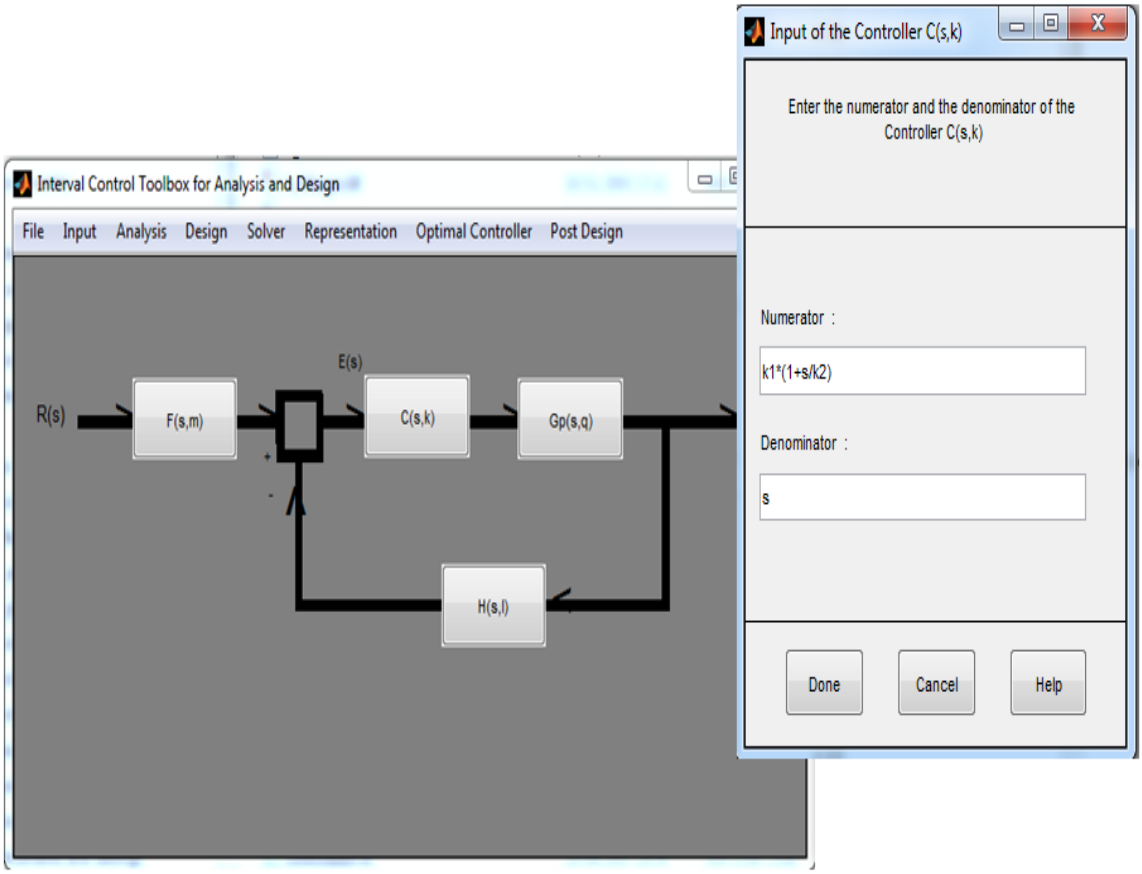


Figure B-2: Definition of the PI structure for IRCAD

Since there are two models, one for each wheel (right and left), two controllers are required.

Definition of the parameter space for the controller

IRCAD requires the definition of a parameter space region within which to start searching for a suitable controller (Fig. B-3) . This is input using the parameter vector $\mathbf{k} = [k_1\ k_2]^T$, i.e., the parameters space to search for a controller that will fulfill the required specifications.

The initial parameter space for the controller for our example was

$$k_1 = [0, 100]$$

$$k_2 = [0, 100]$$

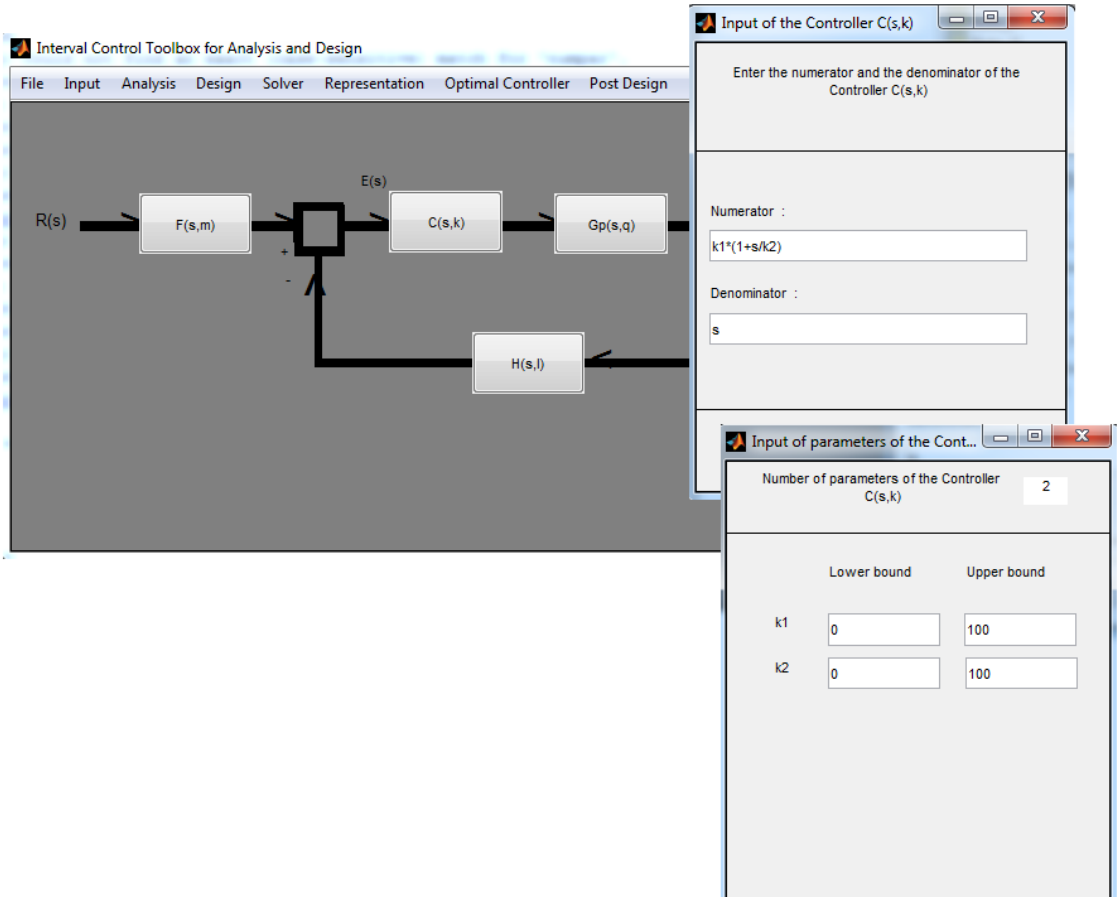


Figure B-3: Defining the parameter space on IRCAD

Definition of the solver

IRCAD allows the user to select a solver (Fig. B-4). In our case, we selected a basic solver is based on a branch and bound algorithm.

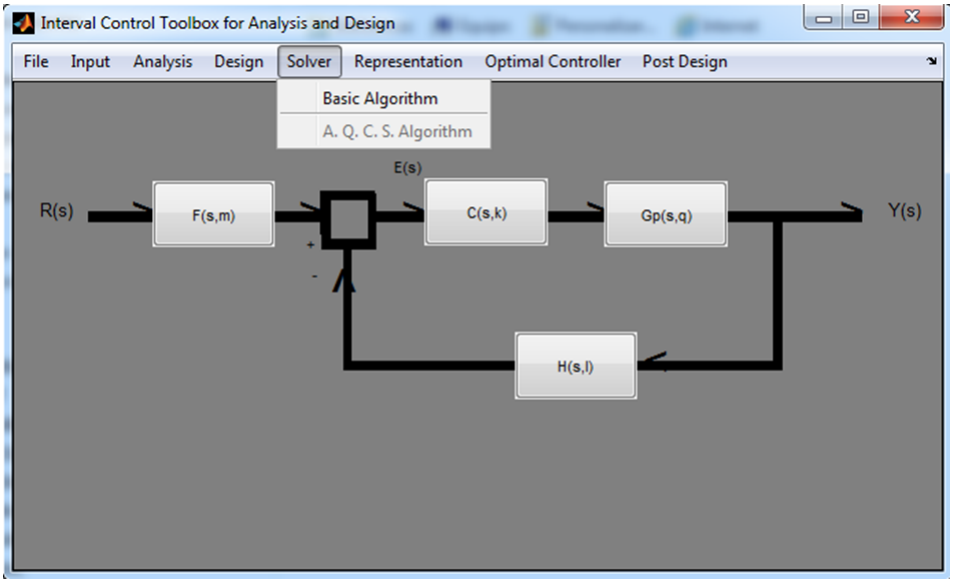


Figure B-4: Selection of the solver on IRCAD

Results obtained in the case of right wheel

Once all the inputs to IRCAD are completed, it only remains to run the design tool. When the user starts this tool, the framework executes a set of algorithms included in the selected solver that divide the parameter space into rectangles. The number of rectangles the parameter space is divided into is configurable using IRCAD options. In this example the resolution was chosen to be 40 $\times$40. Each of these intervals (rectangles) are scanned by the algorithm until it has been classified as one of either unfeasible (yellow), or feasible (red) region. The outcomes for our example are shown in Fig. B-5. Most of the parameter space is classified as unfeasible (yellow), with only a few regions classified as feasible (red).

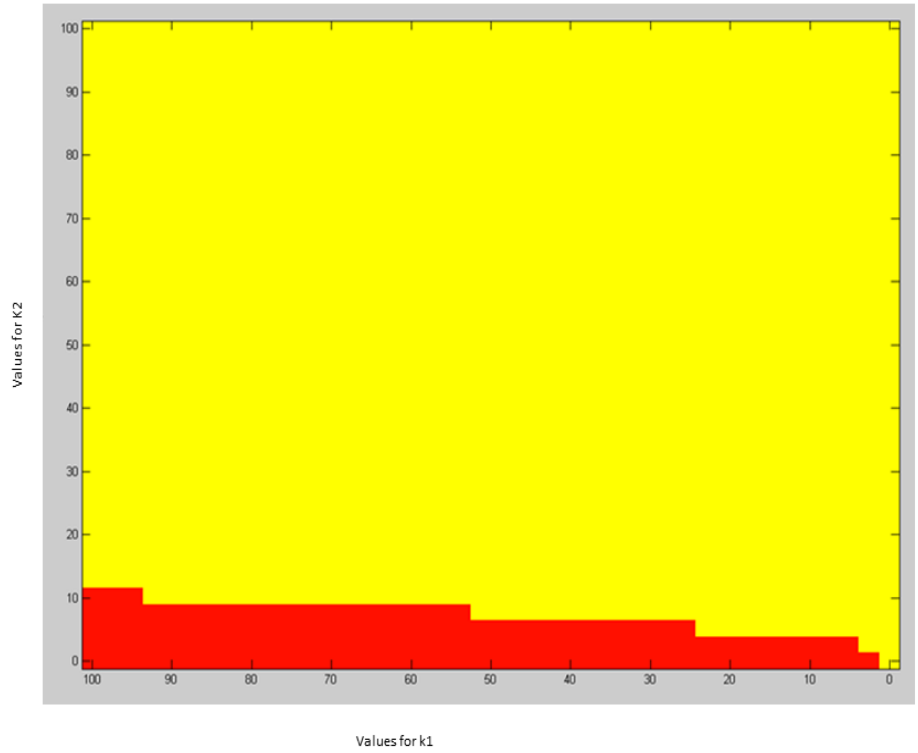


Figure B-5: Set of feasible controllers for the right wheel

Any of the controllers included on the red area could be chosen as the intervalar controller, because they all allow the system to achieve the design specifications. The list of feasible controllers (Fig. B-5) is a set of intervals, and the feasible values are:

$K_1 = [2.5975, 100], K_2 = [0, 2.5975]$

$K_1 = [5.095, 100], K_2 = [2.5975, 5.095]$

$K_1 = [25.075, 100], K_2 = [5.095, 7.5925]$

$K_1 = [52.5475, 100], K_2 = [7.5925, 10.09]$

$K_1 = [92.5075, 100], K_2 = [10.09, 12.5875]$

Although all candidate controllers are valid, is necessary to select a single one, and it is important to select the best one to achieve a good control for the robot. There is no

unique criterion to select the best controller from a set, and so IRCAD offers a utility to help the user select the best controller from a set of feasible controllers, using one of the neighboring, minimum norm, or maximum robustness criteria.

The outcomes for our example for PRIM are:

1. Minimum norm criterion. The PI controller proposed by IRCAD using the minimum norm criterion is $K_1 = 3.8563$ and $K_2 = 1.8488$, as shown in Fig. B-6. IRCAD presents the proposed values in graphical form, putting in bold red color in the proposed square on the feasible region of the parameter space ; and also in numerical form, showing the result in a pop-up window that appears in the center of Fig. B-6 containing the pair of real numbers.

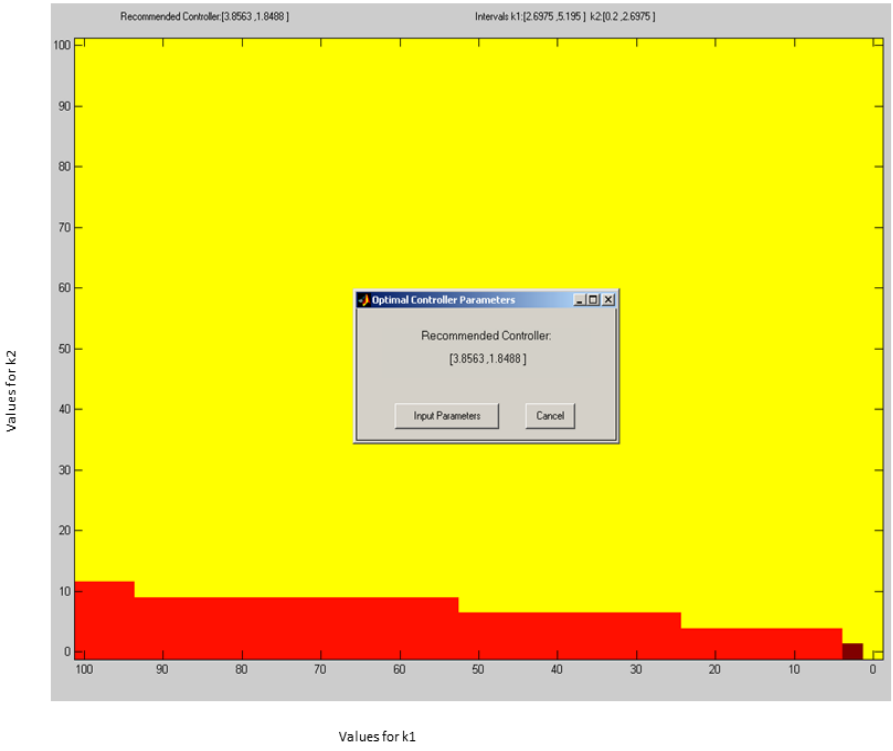


Figure B-6: Minimum norm criterion: recommended controller for the right wheel

2. Neighboring criterion. The PI controller proposed by IRCAD for the right wheel using the neighboring criterion is $K_1 = 61.2888$ and $K_2 = 6.3438$, as shown in Fig. B-7. The proposed values for the controller are presented in a graphical form, putting a bold red color in the proposed square on the feasible region of the parameter space; and numerical form, in a window that appears in the center of Fig. B-7 containing the pair of real numbers.

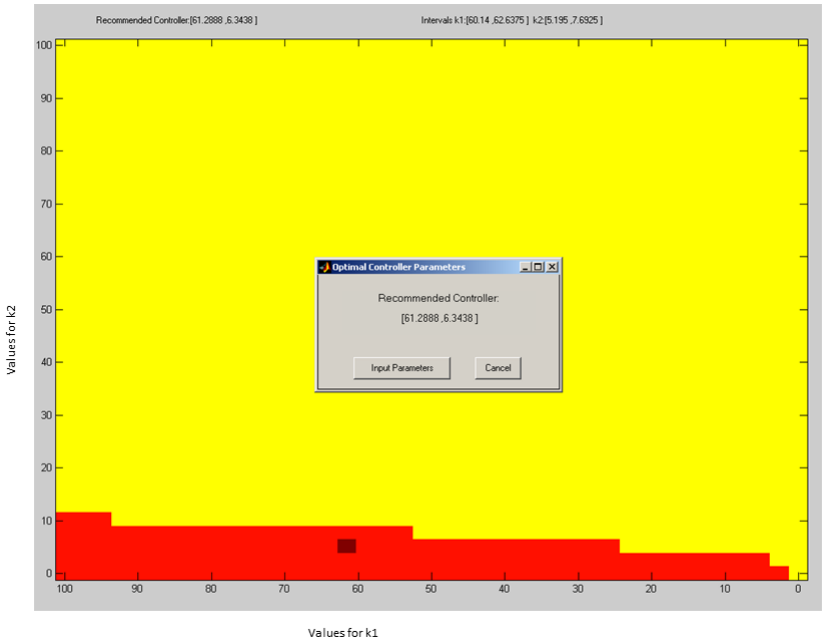


Figure B-7: Neighboring criterion: recommended controller for the right wheel

3. Maximum robustness criterion. The PI controller proposed by IRCAD using the maximum robustness criterion is the same for the Minimum Norm criterion, $K_1 = 3.8563$ and $K_2 = 1.8488$. This is because, in this example, the conditions that are satisfied are the same.

Results obtained in the case of left wheel

The process to obtain the controller for the left wheel is identical to that for the right wheel, and the outcomes are shown in Fig. B-8.

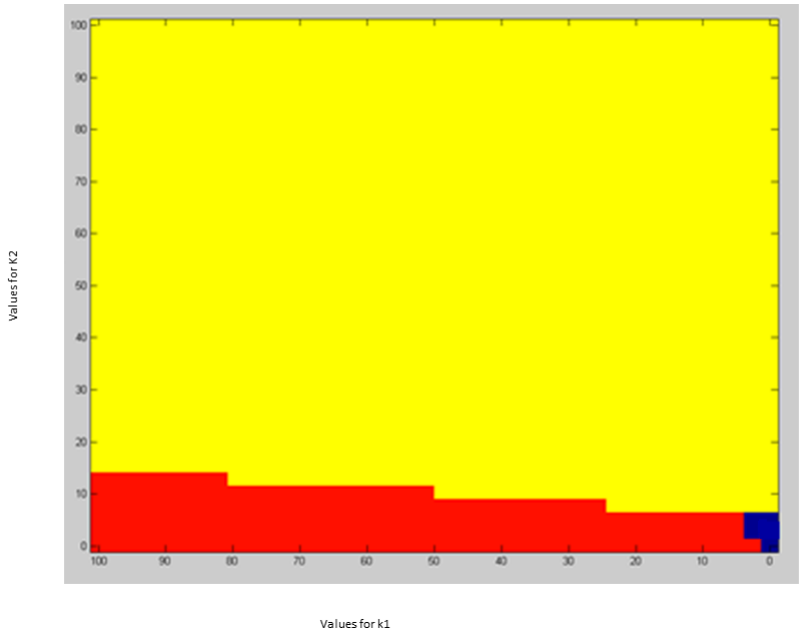


Figure B-8: Set of feasible controllers for the left wheel

Similar to the case for the right wheel, most of the parameter space is classified as unfeasible (yellow), with only a few regions classified as feasible (red). The blue region, is undetermined, where. the branch and bound algorithm was unable to determine the feasibility of the parameters.

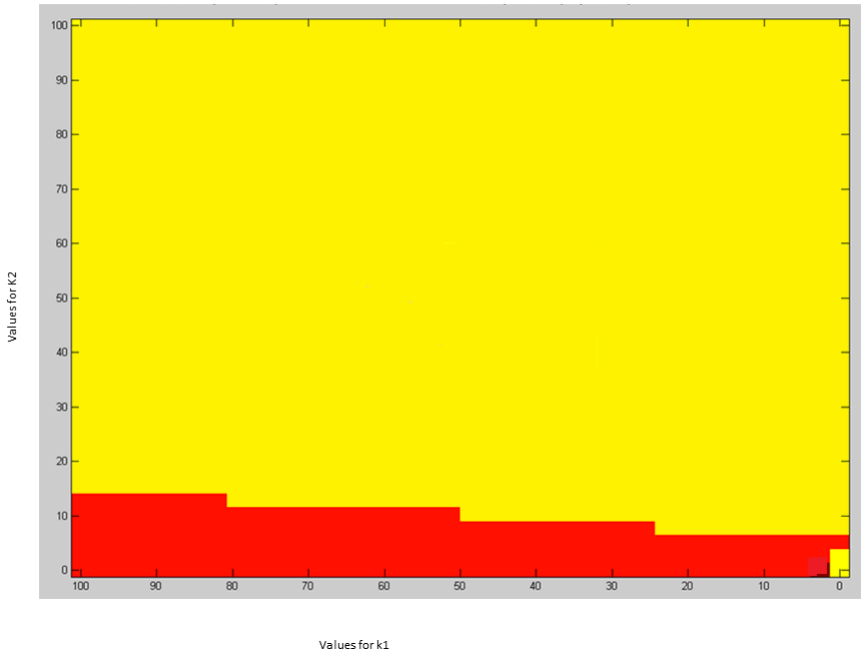


Figure B-9: Set of feasible controller for the left wheel, with blue regions omitted

Eliminating the undetermined zones will increase the reliability of the design. Figure B-9 shows the same execution as for Fig. B-8, but the depth of checking was increased so that all the regions could be successfully classified. Just as for the right wheel, these feasible controllers are a set of intervals, and provide feasible values for the PI controller

$$K_1 = [2.5975, 100], K_2 = [0, 5.095]$$

$$K_1 = [0, 100], K_2 = [5.095, 7.5925]$$

$$K_1 = [25.075, 100], K_2 = [7.5925, 10.09]$$

$$K_1 = [50.05, 100], K_2 = [10.09, 12.5875]$$

$$K_1 = [80.02, 100], K_2 = [12.5875, 15.085]$$

Similarly, the outcome for the three criteria for selecting the best controller are

1. Minimum norm criterion. The PI controller proposed by IRCAD using the minimum norm criterion is $K_1 = 3.8563$ and $K_2 = 1.8488$, as shown in Fig. B-10. The proposed values for the controller are presented in graphical form, putting a bold red color in the proposed square on the feasible region of the parameter space; and in numerical form, showing the result in a pop-up window that appear on the center of Fig. B-10 containing the pair of real numbers.

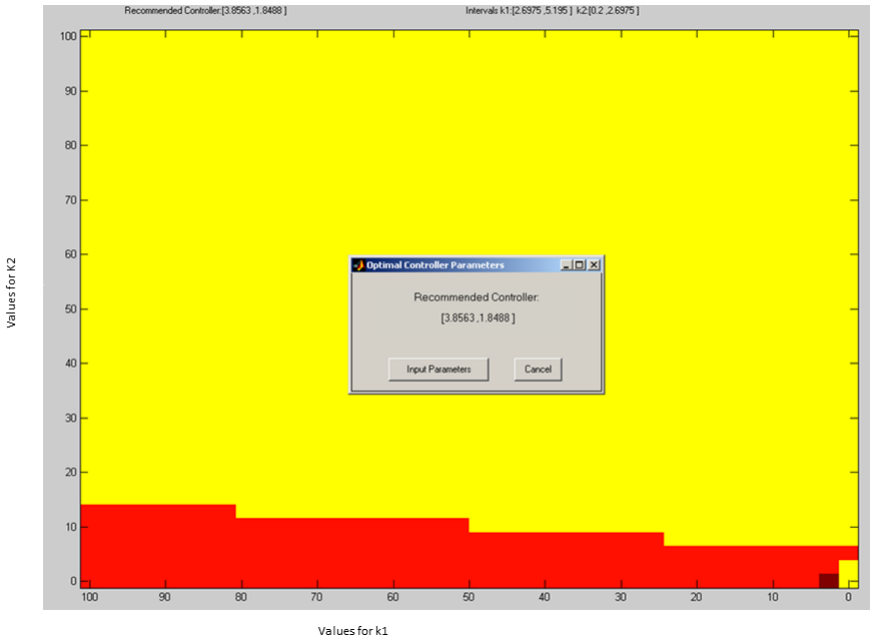


Figure B-10: Minimum norm criterion: recommended controller for the left wheel

2. Neighboring criterion The PI controller proposed by IRCAD for the left wheel using the neighboring criterion is $K_1 = 53.7963$ and $K_2 = 8.8412$, as shown in Fig. B-11. As usual, the proposed values are presented in graphical form, a bold red color in the proposed square of the feasible region of the parameter space; and numerical form as the pair of real numbers.

3. Maximum robustness criterion. The PI controller proposed by IRCAD for the left wheel using the maximum robustness criterion is the same as for the minimum

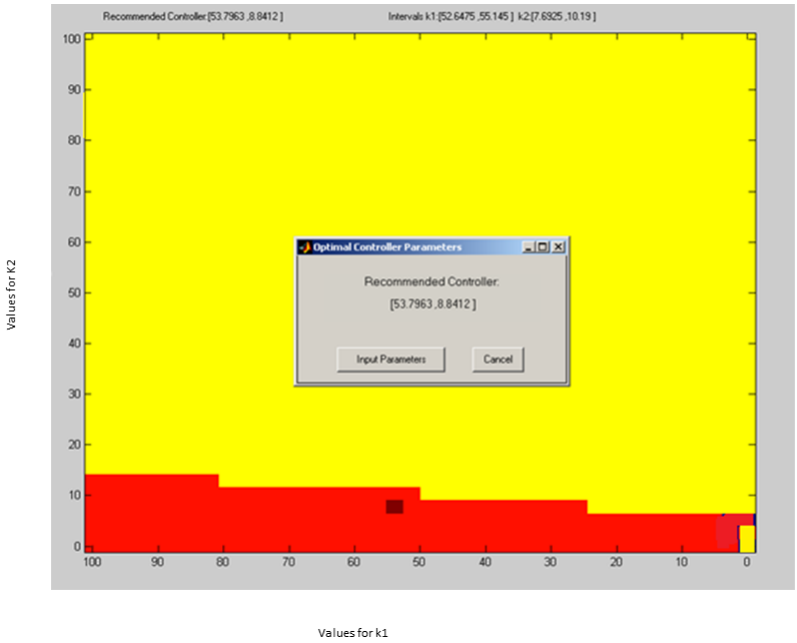


Figure B-11: Neighboring criterion: recommended controller for the left wheel

norm criterion, $K_1 = 3.8563$ and $K_2 = 1.8488$.

Validation of the controllers proposed by IRCAD

Once the controllers have been designed, IRCAD provides a tool to test them, in simulation, to establish if the resultant system fulfills the design specifications.

The right wheel

For the right wheel, selecting the neighboring criterion, the PI controller is

$$C(s) = \frac{61.3\left(1 + \frac{s}{6.34}\right)}{s}. \tag{B.4}$$

Three inputs corresponding to the different velocities (low (0.3 m/s), medium (0.6 m/s), and high (0.9 m/s) were applied and the system control simulated to analyse the system response, as shown in Fig. B-12.

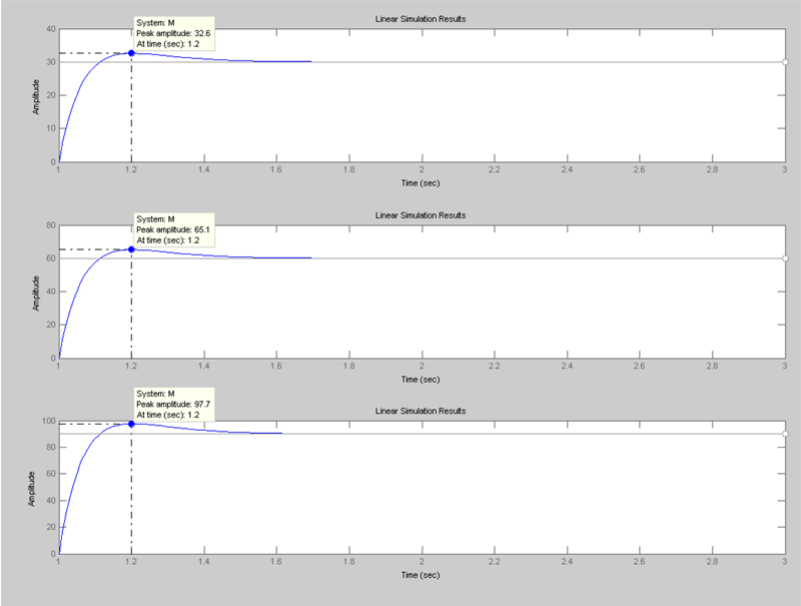


Figure B-12: System response of the right wheel using the neighboring criterion

This choice of criterion produces a system where:

- Low velocity: Overshoot = 8.67%, settling time < 2%.
- Medium velocity: Overshoot = 8.5%, settling time < 2%
- High velocity: Overshoot = 8.56%, settling time < 2%

Thus, all the design specification are fulfilled, and the controller defined in (B.4) is valid.

In the case of the minimum norm criterion, the PI controller is

$$C(s) = \frac{3.84\left(1 + \frac{s}{1.85}\right)}{s}. \tag{B.5}$$

We apply the same inputs as the previous criterion case, as shown in Fig. B-13. This controller is also valid because all the specifications are also fulfilled. However, the

overshoot and settling time, while within the design specifications, are slightly inferior to those achieved using the neighboring criterion (Fig. B-12).

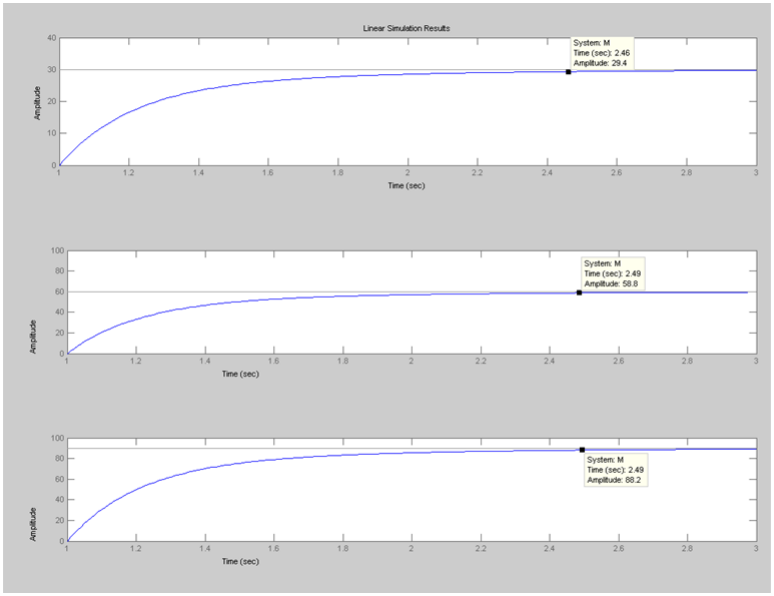


Figure B-13: System response of the right wheel using the minimum norm criterion

The left wheel

For the left wheel, selecting the best controller from the set of controllers that fulfill the specifications using the neighboring criterion, the PI controller is the same as that for the right wheel,

$$C(s) = \frac{53.8\left(1 + \frac{s}{8.84}\right)}{s}. \tag{B.6}$$

Applying the same inputs as the right wheel case, the results are shown in Fig. B-14.:

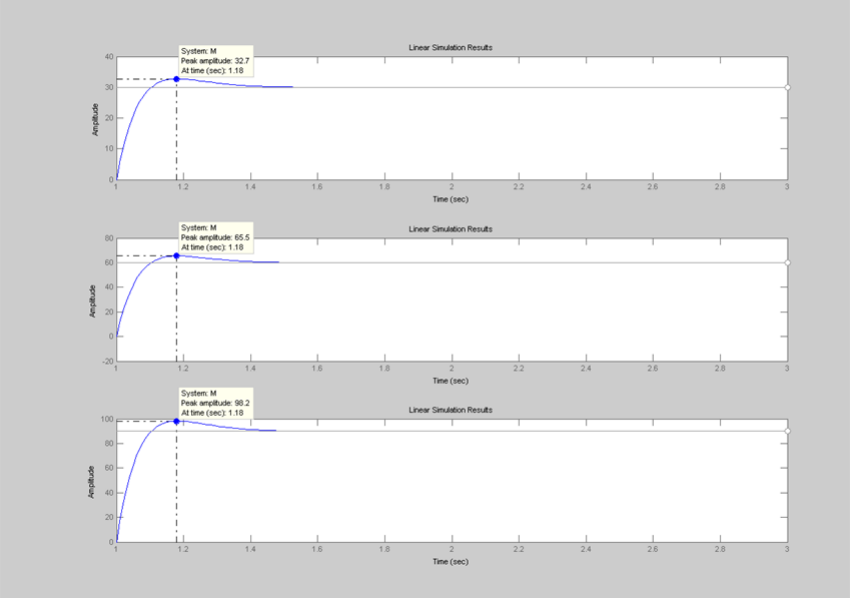


Figure B-14: System response of the left wheel using the neighboring criterion

The resulting system has:

- Low velocity: Overshoot = 9%, settling time < 2%.
- Medium velocity: Overshoot = 9.17%, settling time < 2%
- High velocity: Overshoot = 9.11%, settling time < 2%

Thus, the controller defined in (B.6) is valid, because it fulfills all the design specifications.

Selecting the minimum norm criterion, the PI controller is

$$C(s) = \frac{3.84\left(1 + \frac{s}{1.85}\right)}{s}. \tag{B.7}$$

Applying the same inputs as previously, we have the outcomes as shown in Fig. B-15.

Even though the controller is the same as for the right wheel, the simulation is not identical because the transfer function (model) is different for the wheels.

The resulting system has:

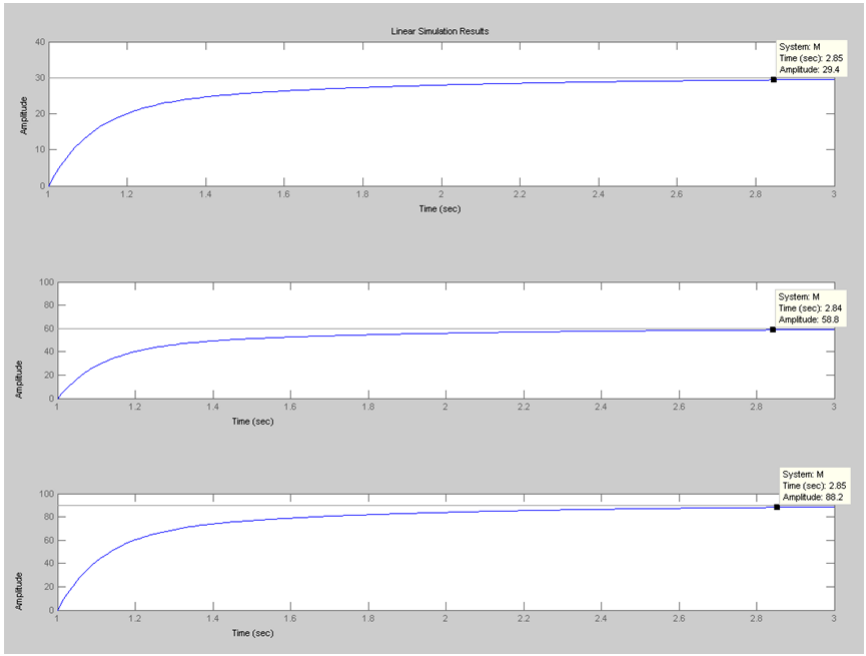


Figure B-15: System response of the left wheel using the minimum norm criterion

- Low velocity: Overshoot= 9.8%, settling time $< 2\%$
- Medium velocity: Overshoot $= 9.2\%$, settling time $< 2\%$
- High velocity: Overshoot= 9.6%, settling time $< 2\%$

Thus, for the left wheel, the simulation is superior for controllers using the neighboring criterion. Accordingly, we have chosen controllers obtained using this criterion for both the left and the right wheels.

Bibliography

[1] J. Ackermann. *Robust control: systems with uncertain physical parameters.* Communications and control engineering series. Springer-Verlag, 1993.

[2] J. Ackermann. PARADISE - parametric robustness analysis and design interactive software environment. Technical report, Institute of Robotics and Mechatronics, 2000. Web: www.op.dlr.de/FF-DR-RR/paradise.

[3] G. Alefeld and J. Herzberger. *Introduction to Interval Computations.* Academic Press, 1983.

[4] J. Armengol. *Application of Modal Interval Analysis to the Simulation of the Behaviour of Dynamic Systems with Uncertain Parameters.* PhD thesis, Universitat De Girona, April 2000.

[5] V. Balakrishnan and S. Boyd. Global optimization in control systems analysis and design. In C. Leondes, editor, *Control and Dynamic Systems: Advances in Theory and Applications*, volume 53, 1992.

[6] G. J. Balas, J. C. Doyle, K. Glover, A. Packard, and R. Smith. *μ-Analysis and Synthesis Toolbox.* The mathworks Inc., The MathWorks Inc.

[7] B. R. Barmish. *New Tools for Robustness of Linear Systems.* MacMillan Publishing Company, 1994.

[8] C. H. Battacharyya S.P and K. L.H. *Robust Control: The Parametric Approach*. Information and system Science Serie. Englewood Cliffs, N.J. Prentice Hall., 1995.

[9] S. P. Bhattacharyya, H. Chapellat, and L. H. Keel. *Robust Control: The Parametric Approach*. Information and System Science Series. Englewood Cliffs, N.J. Prentice Hall, 1995.

[10] J. Bravo, C. Varet, and E. Camacho. Interval model predictive control. *IFAC Congress , Mallorca*, 5 2000.

[11] R. Y. Chiang and M. G. Safonov. *Robust Control Toolbox*. The Mathworks Inc., The Mathworks Inc.

[12] O. Didrit, L. Jaulin, and E. Walter. Guaranteed analysis and optimisation of parametric systems with application to their stability degree. *European Journal of Control*, (3):68–80, 1997.

[13] T. Djaferis. Robust performance design via simultaneous polynomial stabilization. In *Proc. of the American Control Conference*, pages 1008–1013, 1994.

[14] P. Dorato. *Robust control*. IEEE Press selected reprint series. IEEE, 1987.

[15] P. Dorato and R. K. Yedavalli. *Recent Advances in Robust Control*. Collection of selected IEEE Conf. papers. IEEE, 1990.

[16] M. Fadali and K. H. McNichols. Controller design for slowly varying interval plants using gain scheduling and interval mathematics. In *Proceedings of the American Control Conference Chicago, Illinois*, june 2000.

[17] S. Fadali and M. Bebis. Control system design for LTI systems using interval analysis. In *International Conference on Computers and Applications*, pages 413–416, 1998.

[18] S. Faedo. Un nuovo problema di stabilità per le equazioni algebriche a coefficienti reali. *Annali della Scuola Normale Superiore di Pisa*, 7:53–63, Gennaio-Giugno 1953.

[19] X. Feng, G. F. Corliss, and R. W. Kelnhofer. A new interval bounding algorithm for parameter estimation from bounded-error data. In *Proceedings of 14th Triennal IFAC Worl Congress, Beijing P.R. China*, 1999.

[20] I. Ferrer, J. Vehi, and J. Armengol. Interval-based tools for robust control. In *Book of Abstracts*, page 88. Scan2000 / Interval 2000, 2000.

[21] G. Fiorio, S. Malan, M. Milanese, and M. Taragna. Robust performance design of fixed structure controllers for systems with uncertain parameters. In *Proc of the 32nd Conference on Decision and Control*, pages 3029–3031, 1993.

[22] E. Gardeñes and H. Mielgo. Modal intervals: functions. In *Proc. Of the Polish Symposium on Interval and Fuzzy Mathematics*. Zamenkhov's University of Pozna, 1986.

[23] E. Gardeñes, H. Mielgo, and A. Trepat. Modal intervals: reasons and ground semantics. In *Interval Mathematics 1985*, Berlin: Springer, 1985.

[24] J. Garloff and B. Graft. Robust schur stability of polynomials with polynomial parameter dependency. *Multidimensional Systems and Signal Processing.*, 10:189–199, 1999.

[25] J. Garloff, B. Graft, and M. Zetler. Speeding up an algorithm for checking robust stability of polynomials. *Proc. of the IFAC Symposium on Robust Control Design, Elsevier Science, Oxford*, pages 183–188, 1998.

[26] C. Guarino Lo Bianco and A. Piazzi. A hybrid genetic/interval algorithm for semi infinite optimization. In *Proceedings of the 35th Conference on Decision and Control*, volume 2, pages 2136–2138, Kobe, Japan, december 1996.

[27] C. Guarino Lo Bianco and A. Piazzi. Mixed h_2/h_∞ fixed-structure control via semi-infinite optimization. In *Proceedings of the 7th IFAC Symposium on Computer Aided Control Systems Design*, pages 329–334, Gent, Belgium, 1997.

[28] C. Guarino Lo Bianco and A. Piazzi. A genetic/interval approach to optimal trajectory planning of industrial robots under torque constraints. In *Proceedings of European Control Conference. ECC'99*, pages (CD–Rom), 09 1999.

[29] C. Guarino Lo Bianco and A. Piazzi. A global optimization approach to scalar h_2/h_∞ control. In *Proceedings of the European Control Conference, Karlsruhe (Germany) (CD-Rom)*, 09 1999.

[30] W. Haas. A control toolbox using polynomial matrices and interval arithmetic. In *Proceedings of Interval'98. Nanking, China*, 1998.

[31] W. Haas and K. Schlacher. A frequency domain toolbox using interval arithmetic. In *International Conference on Control'98, Swansea, GB*, pages 1060–1065, 1998.

[32] W. Haas and J. Weinhofer. Multivariable compensator design using polynomial matrices and interval arithmetic. In *Proceedings of the 1996 IEEE International Symposium on Computer-Aided Control System Design*, 1996.

[33] W. Haas and J. Weinhofer. A frequency domain toolbox with interval arithmetic. Technical report, University of Linz (Austria), 1998. Web: http://regpro.mechatronik.uni-linz.ac.at/.

[34] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 1992.

[35] L. Hedrich and E. Barke. A formal approach to verification of linear analog circuits with parameter tolerances. *"Entwicklung von Analogschaltungen mit CAE-Methoden"*, Feb. 1999.

[36] N. Hyodo, M. Hong, H. Yanami, S. Hara, and H. Anai. Solving and visualizing nonlinear parametric constraints in control based on quatifier elimination. a matlab toolbox for parametric control system design. *Applicable algebra in engineering, communication and computing.*, 18(6):497–512, December 2007.

[37] N. Hyodo, H. M., H. Yanami, H.and Anai, and S. Hara. Development of a matlab toolbox for parametric robust control -new algorithms and functions-. In *SICE-ICASE International Joint Conference 2006*, pages 2856–2861, Bexco, Busan, Korea, october 2006.

[38] L. Jaulin, J. Boimond, and L. Hardouin. Estimation of discrete-event systems using interval computation. *Reliable computing*, 5(2):165–173, 1999.

[39] L. Jaulin and J. Burger. Proving set inclusion via intervals: Application to parametric robust stability. *Automatica*, 35:627–632, 1999.

[40] L. Jaulin and E. Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29:1053–1064, 1993.

[41] L. Jaulin and E. Walter. Guaranteed nonlinear set estimation via interval analysis. In *Bounding Approaches to System Identification*, pages 363–382. M. Milanese, J. Norton, H. Piet-Lahanier and É.Walter (Eds.), Plenum, New York., 1996.

[42] L. Jaulin and E. Walter. Guaranteed tuning, with application to robust control and motion planning. *Automatica*, 32(8):1217–1221, 1996.

[43] L. Jaulin and E. Walter. Global numerical approach to nonlinear discrete-time control. *IEEE Trans. on Automatic Control*, 42(6):872–875, 1997.

[44] L. Jaulin and E. Walter. Guaranteed parameter bounding for nonlinear models with uncertain experimental factors. *Automatica*, 35(5):849–856, 1999.

[45] M. Kieffer, L. Jaulin, and E. Walter. Guaranteed recursive nonlinear state estimation using interval analysis. In *Proceedings on the 37th IEEE Conference on Decision and Control. Tampa. Florida.*, 12 1998.

[46] M. Kieffer, L. Jaulin, E. Walter, and D. Meizel. *Robustness in Identification and Control (Lecture Notes in Control and Information Sciences 245)*, chapter Nonlin-

ear Identification Based on Unreliable Priors and Data, with Application to Robot Localization, pages 190–203. A. Garulli, A. Tesi and A.Vicino (Eds), 1999.

[47] S. Kwon and J. Cain. A numerical algorithm for robust stabilization of a linear parametric uncertain system. In *Proceedings of the 34th Conference on Decision and Control*, pages 132–135, Dec. 1995.

[48] S.-I. Kwon and J. T. Cain. Interval analysis application for stability margin computation of linear uncertain systems. In *Proc. Of the Twenty-Eigth Southeastern Symposium on System Theory*, 1996.

[49] L. Ljung. *System Identification: Theory for the user*. Prentice Hall, 1989.

[50] S. Malan, M. Milanese, and M. Taragna. Robust analysis and design of control systems using interval arithmetic. *Automatica*, 33(7):1363–1372, 1997.

[51] S. Malan, M. Milanese, M. Taragna, and J. Garloff. B3 algorithm for robust performance analysis in presence of mixed partametric and dynamic perturbations. In *Proc. 31st IEEE CDC*, pages 128–133, 1992.

[52] Maple V, Waterloo Maple Inc., mailto:info@maplesoft.com. *http://www.maplesoft.com/*.

[53] S. Markov and E. Popova. Linear interpolation and estimation using interval analysis. In *Bounding Approaches to System Identification.*, pages 139–157. In: Milanese, M.; Norton, J. P.; P.-Lahanier H.; Water, E. (Eds.), Plenum Press, London, N. Y., 1996.

[54] Matlab- Simulink, The MathWorks Inc, mailto:info@mathworks.com. *http://www.mathworks.com/*.

[55] K. McNichols and M. Fadali. Selecting operating points for discrete-time gain scheduling. *Journal of Computers and Electrical Engineering*, page Under review., 2001.

[56] P. Misra. On stabilization of systems with uncertain parameters: An interval arithmetic approach. In *Proc. Of the American Control Conference*, 1989.

[57] R. E. Moore. *Interval Analysis*. Prentice-Hall series in automation comp. Englewood Cliffs, N.J., Prentice-Hall, 1966.

[58] R. E. Moore. *Methods and applications of interval analysis*. SIAM studies in applied mathematics ; 2. Philadelphia : Siam,1979, 1979.

[59] L. NAG. Numerical algorithms group. http://www.nag.co.uk.

[60] P. Nataraj and G. Sardar. Construction of bode plot envelopes using the moore-skelboe optimization algorithm. In *Conference on Information Technology, Bhubaneshwar, 1999*, pages 224–229, 12 1999.

[61] P. Nataraj and G. Sardar. Computation of QFT bounds for robust sensitivity and gain-phase margins specifications. *Journal of Dynanic Systems Measurement and Control - Transactions of ASME-.*, 2000.

[62] P. Nataraj and G. Sardar. Template generation for continuous transfer functions using interval analysis. *Automatica*, 36:111–119, february 2000.

[63] NICONET. *SLICOT*. ftp://wgs.esat.kuleuven.ac.be/pub/WGS/SLICOT, ftp://wgs.esat.kuleuven.ac.be/pub/WGS/SLICOT.

[64] J. Norton. *An introduction to identification*. Academic Press, 1986.

[65] Y. Ohta. Design of 2 degree of freedom robust PID controller using set operation. In *Proceedings of the Eighteenth IASTED International Conference -Modelling, Identification and Control -Austria-*, 02 1999.

[66] Y. Ohta, L. Gong, and H. Haneda. Polygon interval arithmetic and design of robust control systems. In *Proceedings of the 29th Conference on Decision and Control, Honolulu Hawaii*, Honolulu Hawaii, 12 1990.

[67] Y. Ohta, L. Gong, and H. Haneda. Polygon interval arithmetic and interval evaluation of value sets of transfer functions. *IEICE Trans. Fundamentals*, E77-A(6):1033–1042, June 1994.

[68] Y. Ohta, A. Isogai, K. Tagawa, and H. Haneda. Computation of almost exact gain margin by using polygon interval arithmetic. In *Proceedings of the Asian Control Conference Tokyo.*, 07 1994.

[69] Y. Ohta, A. Isogai, K. Tagawa, and H. Haneda. Fast solving of 0-exclusion problems of value sets of multi-linear functions via PIA. In *Proceedings of 3rd European Control Conference - Rome, Italy*, 09 1995.

[70] Y. Ohta, H. Kakiuchi, M. Watabiki, K. Tagawa, and H. Haneda. NPIA for fast computation of almost correct estimate of value sets of transfer functions. In *Proceedings of the 35th Conference on Decision and Control, Kobe - Japan*, 12 1996.

[71] Y. Ohta, F. Kawamura, A. Isogai, K. Tagawa, and H. Haneda. Robust servosystem design by using PIA. In *Proceedings of IEEE 1994 Industrial Electronics Conference*, pages 1876–1881, 1994.

[72] Y. Ohta, J. Li, K. Tagawa, and H. Haneda. Robust PID controller design. In *International Symposium on Nonlinear Theory and its Applications, (NOLTA'97) Honolulu, USA*, 12 1997.

[73] L. Pacheco. *Control for Navigation of a Mobile Robot using Monocular Data*. Phd thesis, Universitat de Girona. Spain, 2009.

[74] L. Pacheco and N. Luo. Mobile robot local trajectory tracking with dynamic model predictive control techniques. *International Journal of Innovative Computing Information and Control*, 7:3457–3483, 2011.

[75] A. Piazzi and G. Marro. Robust stability using interval analysis. *International Journal of Systems Science*, 27(12):1381–1390, 1996.

[76] A. Piazzi and A. Visioli. Global minimum-time trajectory planning of mechanical manipulators using interval analysis. *International Journal of Control*, 71(4):631–652, 1998.

[77] A. Piazzi and A. Visioli. A system inversion approach to robust set-point regulation. In *Proceedings of the 37th IEEE Conference on Decision and Control Tampa, Florida*, 12 1998.

[78] A. Piazzi and A. Visioli. Global minimum-jerk trajectory planning of robot manipulators. *IEEE Transactions on Industrial Electronics*, 47(1), 2 2000.

[79] K. Sakabe, H. Yanami, H. Anai, and S. Hara. A matlab toolbox for robust control synthesis by symbolic computation. *SICE Annual Conference in Sapporo. Hokkaido Institute of Tecnology, Japan.*, pages 1968–1973, August, 2004.

[80] G. Sardar and P. Nataraj. A template generation algorithm for non-rational transfer functions in QFT designs. *Proceedings of 36th IEEE Conference on Decision and Control*, pages 2684–2689, 2000.

[81] Scilab, http://www-rocq.inria.fr/scilab/scilab.html. *Institute National de Recherche En Informatique et En Automatique (INRIA).*

[82] L.-S. Shieh, X. Zou, and N. P. Coleman. Digital interval modelling and hybrid control of uncertain systems. *IEEE Trans. on Industrial Electronics*, 43(1):173–183, February 1996.

[83] L.-S. Shieh, X. Zou, and N. P. Coleman. Model conversion of discrete-time uncertain systems via the interval geometric-series method. In *Proceedings of 14th Triennial World Congress Beijing, P.R. China IFAC*, 1999.

[84] W. Sienel, B. Tilman, and J. Ackermann. Paradise: Parametric robust analysis and design interactive software environment: A matlab-based robust control toolbox. In

IEEE Int. Symposium on Computer-Aided Control System Design, pages 380–385, September 1996.

[85] SIGLA/X. Aritmetica intervalar en el entorno c++. *Report IMA 97-06-RR*, 1997.

[86] SIGLA/X. Modal intervals. In J. Vehi and M. Sainz, editors, *Application of Interval Analysis to Systems and Control*, pages 157–221, Girona, Spain, 1999.

[87] I. The MathWorks. The polynomial toolbox for matlab, v.2.0. Technical report, 1999. Web: www.polyx.com.

[88] J. Vehí. *Analysis and Design of Robust Controllers by Means of Modal Intervals.* Phd thesis, Universitat de Girona. Spain, 1998. (in Catalan).

[89] J. Vehí, J. Armengol, J. Rodellar, and M. Sainz. Using interval methods for control systems design in the parameter space. In *7th IFAC Symposium on Computer Aided Control Systems Design*, pages 371–375, Gent-Belgium,1997, 1997.

[90] J. Vehí, A. Figueras, and N. Luo. Interval pi velocity control of a non-holonomic mobile robot. *Proc. IFAC Workshop on Digital Control*, 2000.

[91] J. Vehí, J. Rodellar, M. Sainz, and J. Armengol. Analysis of the robustness of predictive controllers via modal intervals. *Reliable Computing*, 6(3):281–301, August 2000.

[92] J. Vehí and M. Sainz. Necessary and sufficient conditions for robust stability using modal intervals. In *42nd IEEE Midwest Symposium on Circuits and Systems, Las Cruces. New Mexico.*, volume 2, pages 673–676, August, 1999.

[93] T. Wada, M. Ikeda, Y. Ohta, and D. D. Siljak. Parametric absolute stability of lur'e systems. *IEEE Transactions on Automatic Control*, 43(11):1649–1653, 11 1998.

[94] E. Walter and L. Jaulin. Guaranteed characterization of stability domains via set inversion. *IEEE Trans. on Automatic Control*, AC-35:835–841, 1994.

[95] E. Walter and L. Pronzato. *Identification of Parametric Models from Experimental Data*. Springer-Verlag, 1997.

[96] J. K. Weinhofer and W. C. Haas. H-inf.-control using polynomial matrices and interval arithmetic. *Reliable computing*, 3:229–237, 1997.