Global minmax optimization for robust H∞ control

Dominique Monnet

THESE DE DOCTORAT DE

L'ECOLE NATIONALE SUPERIEURE
DE TECHNIQUES AVANCEES BRETAGNE
COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies
de l'Information et de la Communication
Spécialité : *Automatique, productique et robotique*

Par

Dominique MONNET

Global minmax optimization for robust H_∞ control

Thèse présentée et soutenue à Brest le 19/11/2018
Unité de recherche : LabSTICC – UMR 6285

Rapporteurs avant soutenance :

Tibor CSENDES — Professor, University of Szeged
Gérard SCORLETTI — Professor, Ecole Central de Lyon, Ampère Laboratory

Composition du Jury :

Président :	Sorin OLARU	Professor, Central Supélec, L2S
Examinateurs :	Sophie TARBOURIECH	Senior Researcher, LAAS-CNRS
	Alexandre GOLDSZTEJN	Research Associate, LS2N
Dir. de thèse :	Benoît CLEMENT	Associate Professor, Lab-STICC
Co-dir. de thèse :	Jordan NININ	Associate Professor, Lab-STICC

Invité(s)
Véronique SERFATY — Head of Information Engineering and Robotics Scientific Area, DGA

Acknowledgements

Puisque ce manuscrit est désormais dans sa forme définitive, j'adresse ici mes remerciements aux personnes qui m'ont accompagnées jusqu'au terme de ces trois longues années de thèse, pêle-mêle: encadrants, famille, collègues, co-bureau, amis, compagnons de bar. Les personnes qui seront mentionnées ci-dessous peuvent se retrouver dans plusieurs des catérogies mentionnées ci-dessus, c'est selon, et je laisse à chacun d'eux la liberté de s'inclure dans celles qui leur sembleront pertinentes.

En premier lieu, je tiens à remercier les deux personnes qui ont pariées sur moi en me confiant le sujet de thèse qui a donné lieu au présent document. Ces deux personnes sont bien sûr Benoît et Jordan, qui ont su m'encadrer tout en me laissant une grande liberté dans mes travaux de recherche. Plus particulièrement, je remercie Benoît pour les conseils avisés qu'il m'a dispensé, notament lors de longs et parfois un peu frais séminaires en rade de Brest. Je remercie Jordan pour la patience, voir l'abnégation, dont il a fait preuve pour l'amélioration des codes d'optimisation (mais en avons nous fini avec le code ?), et les quelques heures passées ensemble à répondre à la question: "et là, comment on contracte ?", question qui a désormais réponse dans les pages de ce manuscrit. Enfin, je leur suis redevable à tout les deux de m'avoir envoyé en conférence de-ci de-là, de Berlin à Braga en passant pas les casinos de Las Vegas et les plages de Santa Catarina.

Je remercie mes colocataires du bureau M111, qui lui ont donné une dimension résoluement internationale : Gaspard, Juan, Lorenzo, Nisha et Federico. Nous avons passé ensemble de nombreuses heures à mener nos recherche côte-à-côte, et partagé des moments allant de la rigolage au désespoir quasi-catatonique. Mes pensées vont en particulier à Juan, avec qui j'ai passé une excellente année pendant laquelle nous avons travailler ensemble. Je ne le remercierai jamais assez de s'être occupé des manips. Et bien sûr, je n'oublie pas Gaspard, avec qui j'ai passé la majorité de la thèse. Nous nous sommes bien souvent posés la question de notre présence dans ce bureau lors de légers moments de doute quant à nos thèses. Je crois qu'aucun de nous n'a jamais vraiment trouvé de réponse définitive, mais maintenant que plus aucun de nous deux n'est dans ce bureau disons que la question n'a plus plus lieu d'être.

L'équipe au grand complet de la zone de réflexion scientifique, ou plus vulgairement salle café, à toute sa place parmi ces lignes. Les nombreuses discussions qui y ont eu lieu, souvent de hautes volées, ont été une source d'inspriration continue. Je remercie en particulier Michel et Christophe pour leurs anecdotes qu'ils ont toujours eu à cœur de rendre plus savoureuses avec soupçon de mauvais esprit ou de cynisme. J'espère m'être acquitté honorablement de ma tâche de responsable café pendant mes deux dernières années de thèse, bien que mes réflexes étudiants m'aient plus poussés vers les promotions que les considérations gustatives.

Je remercie chaudement Annick et Michèle, qui m'ont grandement aidées dans mes diverses démarches administratives. Leurs facilités à gérer des situations dans lesquels un néophyte tel que moi avait toutes les chances de s'empêtrer m'ont toujours stupéfié.

Je me dois aussi de remercier Alexandre, et plus récement Rui, qui donnent du sens à mes

travaux en les applicant sur des problèmes réels. J'espère ne pas avoir trop dégoûté Alexandre de la commande robuste, car bien souvent il a joué le rôle de bêta-testeur pour les codes que j'ai développé et il a du composer avec une interface, sinon austère, tout du moins rebutante au premier abord. C'est, entre autre, grâce à ses encouragements et son enthousiasme pour mes travaux que j'ai pu les mener jusqu'à leurs termes. Je tiens, en outre, à remercier Pierre Simon pour son aide indispensable lors des manips avec le Ciscrea.

Agradezco al grupo de investigación de la Universidad Nacional de La Plata por la excelente estada realizada. En particular quiero agradecerles a Fabricio, Hernán y Juan por su afectuoso recibimiento y el tiempo que me han dedicado. Tengo la esperanza que nuestra colaboración pueda continuar en el tiempo. Por muchos mas baratones juntos!

Je n'oublie pas les doctorants et ingénieurs de recherche du bâtiment M, bande de joyeux drilles retrouvée autour d'un tas de gauffres ou de crèpes ou lors de pique-nique sur les plages bretonnes : Auguste, Charles, Gaspard, Guillherme, Irène, Joris, Romain, Simon, Thomas. Je souhaite bon courage aux doctorants en rédaction, et une bonne continuation à tous !

Ma famille mérite bien plus que ces quelques lignes pour le soutien qu'elle m'a apportée durant cette thèse. Mes parents et mes soeurs m'ont toujours offert une place où m'échapper et me ressourcer, un ailleurs pour prendre du recul sur la thèse, et tout le reste.

Je dédie mes dernières pensées à mes grand-parents, qui s'en sont allés au cours de ces trois années de thèse, ainsi qu'à ma tante et mon ami, partis trop tôt.

Dominique.

"So we shall let the reader answer this question for himself: who is the happier man, he who has braved the storm of life and lived or he who has stayed securely on shore and merely existed ?"

Hunter S. Thompson.

Notations

Intervals

$\emptyset$	:	empty set.
$\mathbf{x}$	:	vector of interval (or box)
$\mathbb{IR}$	:	set of real intervals
$\underline{\mathbf{x}}$	:	lower bound of $\mathbf{x}$
$\overline{\mathbf{x}}$	:	upper bound of $\mathbf{x}$
$\mathbf{f}$	:	inclusion function of f

Optimization

x	:	minimization variable
f	:	objective function, $\mathbb{R}^n \mapsto \mathbb{R}$.
x^*	:	minimizer (resp. maximizer) of a minimization problem (resp. maximization problem)
$\tilde{x}$	:	best feasible point found
f^*	:	global minimum (resp. maximum) of a minimization problem (resp. maximization problem)
$\mathcal{C}$	:	contractor
$\mathcal{C}_{\tilde{f}}$	:	contractor based on the best feasible point
$\mathcal{C}_J$	:	monotonicity contractor

Minmax Optimization

y	:	maximization variable
f_x	:	$f_x(y) = f(x, y)$
$\mathbb{Y}_x$	:	feasible domain of y for the maximization problem at x
$\mathbb{Y}_{\mathbf{x}}$	:	$\bigcup_{x \in \mathbf{x}} \mathbb{Y}_x$
f_{sup}	:	objective function of the minmax problem, $f_{sup}(x) = \sup_{y \in \mathbb{Y}_x} f(x, y)$
y_x^*	:	maximizer of f_x over $\mathbb{Y}_x$
$f_{\mathbf{x}}$	:	set of functions f_x, $f_{\mathbf{x}} = \{f_x \,\vert\, x \in \mathbf{x}\}$
$f_{\mathbf{x}}^-$	:	lower bound of the hull of $f_{\mathbf{x}}$, $f_{\mathbf{x}}^-(y) = \min_{x \in \mathbf{x}} f(x, y)$
$f_{\mathbf{x}}^-$	:	upper bound of the hull of $f_{\mathbf{x}}$, $f_{\mathbf{x}}^+(y) = \sup_{x \in \mathbf{x}} f(x, y)$
$\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$	:	contractor for $\mathbb{Y}_{\mathbf{x}}$
$\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$	:	extension of $\mathcal{C}_{\tilde{f}}$ to a set of functions
$\mathcal{C}_{J_y}$	:	extension of $\mathcal{C}_J$ to a set of functions
$\mathcal{C}_l$	:	adaptation of $\mathcal{C}_{\tilde{f}}$ for the minmax problem

Control

s	:	Laplace's variable
i	:	imaginary unit
ω	:	pulsation
k	:	vector of controller parameters
θ	:	vector of uncertain parameters
Θ	:	set of uncertain parameters
$T_{w \to z}$	:	transfer from w to z
$\tilde{w}$	:	weighted counterpart of the signal w
$\tilde{z}$	:	(non weighted) performance output

Acronyms

Optimization

B&B	:	Branch and Bound
CSP	:	Constraint Satisfaction Problem
IBBA	:	Interval Branch and Bound Algorithm
SIBBA	:	Set Interval Branch and Bound Algorithm
MMIBBA	:	MinMax Interval Branch and Bound Algorithm
SIC	:	Semi Infinite Constraint
SIP	:	Semi Infinite Program

Control

SISO	:	Single Input Single Output
MISO	:	Multiple Input Single Output
SIMO	:	Single Input Single Output
MIMO	:	Multiple Input Multiple Output
LMI	:	Linear Matrix Inequality
SS	:	Structured Synthesis
IOSS	:	Independent Outputs Structured Synthesis
RA	:	Robustness Analysis
SSV	:	Structured Singular Value
RSA	:	Robust Stability Analysis
RHA	:	Robust H_∞ Analysis
RSS	:	Robust Structured Synthesis

Contents

Introduction

Control theory consists in controlling dynamical systems such as they behave as desired. This is achieved by searching for a control law such that criteria, translating what is the desired behavior, are met. Finding this control law is the *synthesis problem*: compute the best control law with respect to the synthesis criteria. Formulating such a problem under a mathematical form often requires a model of the dynamical system. This model may not describe exactly the real system due to approximations, and as a consequence suffers from uncertainties. The *robustness analysis problem* consists in verifying that the control law is robust to model uncertainties, meaning that the synthesis criteria are met for all possible values of uncertainties. Finally, the *robust synthesis problem* resides in synthesizing a control law taking the model uncertainties into account, and in a sense merges both the synthesis and the analysis problems.

Those three problems are generally framed as optimization programs in order to be solved, deriving from the synthesis criteria an objective function to minimize and constraints that must be satisfied. Both the objective function and the constraints are in general not convex. To this extent, several optimization methods/approaches can be considered.

- The initial non-convex problem is reformulated into a convex one. In this way, the new problem has only one local minimum which is also the global minimum, and can be solved easily. Such an approach is said to be *conservative*, since the problem solved is different than the initial problem. Three kind of convex reformulation can be performed:
  - A convex subset provides an upper bound of the objective over the search domain.
  - A convex relaxation provides a lower bound on the objective function over the search domain.
  - A convex approximation has no guarantee to be lower or greater than the objective function, their relative positions is not known over the search domain.

  An enclosure of the global optimum of the initial problem can be derived from convex subset and relaxation, but this enclosure can be pessimistic.
- Local optimization methods can be used to solve the initial non-convex problem. These methods converge locally, and the solution provided depends in the starting point chosen in the search domain. To avoid this drawback, a multi-start strategy is generally adopted, starting the local optimization methods at several random points. Alternatively to multi-start, meta-heuristics methods such as Tabu search provide strategies to avoid falling in the same local minima. However, there is no guarantee that the global optimum is found.

- Global optimization methods are guaranteed to converge to the global minimum, independently from the initial condition of the method, and do not need starting point. These methods are very interesting when one desire to find the global optimum with a given accuracy. For non-convex problems, the probably most popular method for global optimization is the branch and bound algorithm.

This thesis focuses in particular on H_∞ control, and the three associated control problems, namely the synthesis, the robustness analysis and the robust synthesis. Despite of the advantage that global optimization presents over local optimization and convex reformulation, very few attention has been given to this approach to solve H_∞ problems. The aim of this thesis is to investigate how these problems can be solved in a global way. The global optimization method chosen is Branch and Bound algorithms based on interval analysis. This method require explicit functions of the objective and the constraints, and it will appear that the problems have, under certain conditions, an explicit minmax formulation and involve semi infinite constraints. However, if the global resolution of this particular class of problems has received some attention in the literature, no solver is available. As a consequence, this thesis is dedicated to both the explicit formulation of the H_∞ problems and the development of algorithms to solve them in a global way. To do so, this thesis is organized as it follows.

Chapter 1 presents the principle of H_∞ control and the related synthesis, robustness analysis and robust synthesis problems. First, the necessary mathematical tools of control theory are introduced. Then, the H_∞ control problems are presented, and a brief overview of the existing methods to solve them is given. The chapter concludes by indicating the approach chosen in this thesis to solve them in a global way.

Chapter 2 introduces the classical global optimization methods. More precisely, this chapter focuses on Branch and Bound algorithms based on interval analysis. The interval methods that enable to compute guaranteed bounds on a non-convex function as well as constraint propagation techniques are presented. It is explained how these methods are integrated in a Branch and Bound framework to compute the global optimum of a program or to characterize the feasible set of a constraint satisfaction problem.

Chapter 3 is dedicated to the global resolution of continuous minmax optimization problems subject to semi infinite constraints, and strongly relies on the tools introduced in Chapter 2. The chapter begins with a brief state of the art on continuous minmax and semi infinite programs, and we propose a framework to deal with both the max objective function and the semi-infinite constraints the same way. A special Branch and Bound algorithm is introduced to compute bounds on the maximums of sets of functions, adapting the interval tools presented in Chapter 2. This algorithm is embedded in a main Branch and Bound algorithm to solve the minmax problem subject to semi infinite constraints. The performance of this double Branch and Bound strategy is illustrated with minmax and semi infinite programs benchmark examples.

Chapter 4 continues Chapter 1 by formulating the H_∞ control problems in a new way such that it can be dealt with the global optimization algorithms developed in Chapters 2 and 3. The synthesis is formulated as a minmax problem, the analysis as a maximization

problem, and the robust synthesis as a minmax problem subject to semi infinite constraints. Several examples are solved, and the results are discussed.

Chapter 5 presents an application of H_∞ control. The regulation of the yaw angle of an autonomous underwater vehicle subject to external disturbances is studied. A methodology is proposed for the synthesis and the analysis of a control law from H_∞ criteria. This control law is validated through simulations and experiments, and compared with two other classical empirical solutions.

Finally, Chapter 6 lists the contributions of this thesis in both optimization and control fields. Some prospects for future works are also proposed.

Publications

International peer-reviewed journals

- Dominique Monnet, Jordan Ninin, and Luc Jaulin. Computing an inner and an outer approximation of the viability kernel. *Reliable Computing*, 22:138–148, 2016

International peer-reviewed conferences with proceedings

- Dominique Monnet, Jordan Ninin, and Benoît Clement. Global optimization of H_∞ problems: Application to robust control synthesis under structural constraints. In *International Conference on Mathematical Aspects of Computer and Information Sciences*, pages 550–554. Springer, 2015

- Dominique Monnet, Jordan Ninin, and Benoît Clement. A global optimization approach to structured regulation design under H_∞ constraints. In *Proceedings of the IEEE 55th Conference on Decision and Control*, pages 658–663, 2016

- Dominique Monnet, Jordan Ninin, and Benoît Clement. A global optimization approach to H_∞ synthesis with parametric uncertainties applied to auv control. *Proceedings of 20th IFAC World Congress, IFAC-PapersOnLine*, 50(1):3953–3958, 2017

- Juan Luis Rosendo, Dominique Monnet, Benoît Clement, Fabricio Garelli, and Jordan Ninin. Control of an autonomous underwater vehicle subject to robustness constraints. In *Accepted to 9th IFAC Symposium on Robust Control Design ROCOND*. Elsevier, September 2018

- Dominique Monnet, Juan Luis Rosendo, Hernán De Battista, Benoît Clement, Fabricio Garelli, and Jordan. Ninin. A global optimization approach for non-linear sliding mode control analysis and design. In *Accepted to 9th IFAC Symposium on Robust Control Design ROCOND*. Elsevier, September 2018

Other communications

- Juan Luis Rosendo, Dominique Monnet, Benoît Clement, Fabricio Garelli, Irvin Probst and Jordan Ninin. Control of an Autonomous Underwater Vehicule under Robustness Constraints. *Small Workshop on Interval Analysis.* 2016

- Dominique Monnet, Jordan Ninin and Benoît Clement. Commande structurée approchée par l'optimisation globale et l'analyse intervalle. *Séminaire GT MOSAR*. 16 Mars 2016.

- Dominique Monnet, Jordan Ninin and Benoît Clement. Global optimization of continuous minimax problem. *XIII Global Optimization Workshop GOW16 4-8 September 2016*, 16:175–178.

- Dominique Monnet, Jordan Ninin and Benoît Clement. Optimisation globale de problèmes Min-Max: Application à la synthèse de loi de commande robuste. *ROADEF* , 21-23 février 2018.

- Dominique Monent, Jordan Ninin and Benoît Clement. Interval Branch-and-Bound Algorithm for semi-infinite programming. *International Symposium on Mathematical Programming (ISMP)*. July 2018.

Chapter 1

H_∞ control problems

This chapter presents the H_∞ synthesis principle, and the underlying optimization problem. This problem has received a lot of attention since it was introduced at the beginning of the 80's. An additional difficulty occurs when systems depending on uncertain parameters are considered, since the robustness of the control law must be taken into account. This rises two other problems: the robustness analysis and the robust synthesis problems.
This chapter is organized as follows. Sections 1.1 to 1.3 provide the basics in control theory needed to introduce the H_∞ problems, and will also be useful in Chapter 4. These three sections are mainly based on [133] and [123]. However, the reader can refer to numerous books dedicated to control theory for further insights [134, 23, 76, 53]. Sections 1.4 to 1.8 introduce the H_∞ problems that are addressed in this thesis, and the existing methods to solve them. Finally, Section 1.9 describes how we propose to approach these problems to solve them in a global way.

1.1 Dynamical systems

A dynamical system is described by two *state equations*: the *evolution equation* and the *observation equation*.

$$\begin{cases} \dfrac{dx(t)}{dt} = f(x(t), u(t), t) & \text{evolution equation,} \\ y(t) = g(x(t), u(t), t) & \text{observation equation,} \end{cases}$$

where

- $x(t) \in \mathbb{R}^{n_x}$ is the state vector,
- $u(t)$, $u : \mathbb{R}^+ \mapsto \mathbb{R}^{n_u}$ is the vector of input signals,
- $y(t)$, $y : \mathbb{R}^+ \mapsto \mathbb{R}^{n_y}$ is the vector of output signals.

The evolution equation describes the dynamics of the system as an ordinary differential equation, generally derived from the physical laws that govern the system (Newton's laws, electrical laws, etc). The observation equation provides the signal y in function of the states and the input signals. A system is said to be

- *single input single output* (SISO) if $n_u = n_y = 1$,
- *multiple input single output* (MISO) if $n_u > 1$ and $n_y = 1$,
- *single input multiple output* (SIMO) if $n_u = 1$ and $n_y >$,
- *multiple input multiple output* (MIMO) if $n_u > 1$ and $n_y > 1$.

A system is said to be time invariant (TI) if its state equations do not vary over the time explicitly.

$$\begin{cases} \dot{x} = f(x(t), u(t)) \\ y(t) = g(x(t), u(t)) \end{cases}$$

1.2 Linear Time invariant systems

Linear Time Invariant (LTI) systems represent a particular class of dynamical systems. Both their evolution and observation functions depend linearly on the states and the inputs,

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du, \end{cases}$$

where $A \in \mathbb{R}^{n_x \times n_x}$, $B \in \mathbb{R}^{n_x \times n_u}$, $C \in \mathbb{R}^{n_y \times n_x}$ and $D \in \mathbb{R}^{n_y \times n_u}$ are real matrices.
The properties of observability and controllability are in general highly desirable. The controllability ensures that all the states of the system can be controlled by a state feedback controller. On the other hand, the observability ensures that, knowing the input signal u, it is possible to retrieve the states of the system.

Definition. A LTI system is said to be controllable if, for any initial state $x(0) = x_0$, any time $t_1 > 0$ and any final state x_1, there exists an input $u(t)$ such that $x(t_1) = x_1$. Otherwise, the system is said to be uncontrollable.

Definition. An LTI system is said to be observable if, for any time $t_1 > 0$, the initial state $x(0) = x_0$ can be determined from the time history of the input $u(t)$ and the output $y(t)$ in the interval $[0, t_1]$. Otherwise, the system is said to be unobservable.

LTI systems are of particular interest since they can be interpreted from a frequency point of view thanks to the Laplace transform. The frequency representation of linear systems relies on the *transfer functions* which are ratio of polynomials in $s = i\omega$ with i the imaginary unit and ω a real variable which corresponds to the *pulsation* in rad/s. The transfer from $u(s)$ the Laplace transform of $u(t)$ to $y(s)$ with zero initial condition ($x_0 = 0$) is given by

$$u(s) = G(s)y(s), \quad G(s) = C(sI - A)^{-1}B + D,$$

where $G(s)$ is a matrix of transfer functions of dimensions $n_y \times n_u$.
Conversely, a *state space realization* (A, B, C, D) can be retrieved from a transfer matrix. The realization is said to be minimal if the dimension of A is the smallest possible one. The eigenvalues of A are called the *modes* of G. If G is SISO, then $G(s)$ is a transfer function, and the poles correspond to the roots of its denominator. The zeros of G correspond to the roots of the numerator of $G(s)$.

1.2.1 Stability of LTI systems

A major property of systems is the stability, which ensures that the states of the system do not diverge to infinity over the time. The main objective of a control law is to ensure stability of the system. The definition of stability as well as several ways to prove the stability of a system are given here.

Definition. A LTI system is said asymptotically stable if its states converge to 0 provided that the input is null at any time.

$$u(t) = 0, \forall t \geq 0 \implies \lim_{t \to +\infty} x(t) = 0.$$

If $u(t) = 0$ at any time, then $x(t)$ the solution to the state equation is

$$x(t) = e^{At} x_0,$$

where $x_0 = x(0)$ is the initial condition. As a consequence, it follows that $x(t)$ tends to zero for any initial conditions x_0 if and only if the eigenvalues of A have strictly negative real parts. The stability of a LTI system can be proved in multiple ways, the following theorem provides the commonly used ones:

Theorem 1.2.1. *Let G be a LTI system. The four following statements are equivalent:*

1. *G is asymptotically stable.*
2. *The eigenvalue of A have negative real parts, in this case A is said to be* Hurwitz*.*
3. *The roots of the characteristic polynomial of A have negative real part.*
4. *There exists X positive definite such as $A^T X + XA$ is definite negative [23]. X is called the* Lyapunov matrix*.*

The link between statements 2 and 3 is straightforward since the eigenvalues of A are the roots of its characteristic polynomial, given by $Q(s) = \det(sI - A)$. The last condition comes from Lyapunov's stability theory. Considering the third statement, the *Hurwitz criterion* given by Theorem 1.2.2 allows to verify if the roots of the characteristic polynomial have negative real parts.

Theorem 1.2.2. *[48]*
Consider the polynomial $Q(s) = \sum_{i=0}^{n} a_i s^i$ with $a_n > 0$, and its associated Hurwitz matrix H.

$$H = (h_{ij}) = \begin{pmatrix} a_{n-1} & a_{n-3} & a_{n-5} & \dots & \dots & \dots & 0 & 0 & 0 \\ a_n & a_{n-2} & a_{n-4} & & & & \vdots & \vdots & \vdots \\ 0 & a_{n-1} & a_{n-3} & & & & \vdots & \vdots & \vdots \\ \vdots & a_n & a_{n-2} & \ddots & & & 0 & \vdots & \vdots \\ \vdots & 0 & a_{n-1} & & \ddots & & a_0 & \vdots & \vdots \\ \vdots & \vdots & a_0 & & & \ddots & a_1 & 0 & \vdots \\ \vdots & \vdots & 0 & & & & a_2 & a_n & \vdots \\ \vdots & \vdots & \vdots & & & & a_3 & a_1 & 0 \\ 0 & 0 & 0 & \dots & \dots & \dots & a_4 & a_2 & a_0 \end{pmatrix}.$$

The real parts of the roots of $Q(s)$ are strictly negative if all the leading principal minors Δ_i of H are positive.

$$\Delta_i = \det \begin{pmatrix} h_{11} & \dots & h_{1i} \\ \vdots & & \vdots \\ h_{n1} & \dots & h_{ni} \end{pmatrix}$$

The condition $a_n > 0$ is not constraining. If this condition is not ensured it suffices to apply the Hurwitz criterion to the polynomial $Q'(s)$ having the same roots as $Q(s)$,

$$Q'(s) = \frac{1}{a_n} \sum_{i=0}^{n} a_i s^i.$$

The Hurwitz criterion permits to express the stability condition of an LTI system as n_x polynomial inequalities, and will be of particular importance in Chapter 4.

Weaker notions of controllability and observability are the stabilizability and detectability.

Definition. A system is said to be stabilizable if there exists F such that $A + BF$ is stable.

The stabilizability of a system ensures that, even if not all the states can be controlled, the unstable states can be stabilized.

Definition. A system is said to be detectable if there exists H such that $A + HC$ is stable.

The detectability of a system ensures that at least the unstable states can be observed. In this case, the system has no hidden unstable mode. In this thesis, the systems considered are stabilizable and detectable.

1.2.2 Systems and signals norms

The norms of LTI systems are of particular interest, since they characterize properties of the system in terms of input-output signal relationship. To this extend, they can be used as synthesis criteria for control laws.

The $\mathcal{L}_2$ space is the space of *Lebesgue measurable* signals, which have a finite energy given by the 2-norm,

$$||u||_2 \triangleq \left(\int_0^\infty ||x(t)||_2 dt \right)^{1/2}.$$

A linear system is in fact an application from the space of signals into this same space. The H_∞ norm of an LTI system is defined as the maximal frequency response of the system,

$$||G||_\infty \triangleq \sup_{Re(s)>0} \sigma_{max}(G(s)^H G(s)),$$

where $G(s)^H$ is the hermitian transpose of $G(s)$ and σ_{max} is the greatest singular value. If G is stable, then

$$||G||_\infty = \sup_\omega \sigma_{max}(G(j\omega)^H G(j\omega)),$$

and the H_∞ norm corresponds to the worst-case ratio between the energy of the output and input signals,

$$||G||_\infty = \sup_{u \in \mathcal{L}_2, ||u||_2 > 0} \frac{||y||_2}{||u||_2}.$$

If G is SISO, then the H_∞ norm is simply the maximum of the modulus of its transfer function over the pulsations. To this extend, the maximal singular value can be viewed as an extension of the gain to MIMO systems.

Theorem 1.2.3. *Let G be a stable LTI system with input u and output y,*

$$||G||_\infty = \sup_{u \in \mathcal{L}_2, ||u||_2 > 0} \frac{||y||_2}{||u||_2}.$$

For further details about systems and signals spaces and norms, the reader can refer to Chapter 4 of [133].

As an example, consider a second order system G with $\omega_n = 1$ and $\xi = 0.1$.

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$

The H_∞ norm of G is equal to 5, obtained at the pulsation $\omega_{max} = 0.9899$, and can be retrieved either from the time response of G to an input sine wave $u(t) = \sin(\omega_{max} t)$ in Figure 1.1b, or from the Bode diagram of G in Figure 1.1a.

1.2.3 Interconnection of systems and internal stability

The purpose of feedback control is to compute a controller, denoted K, which sends the correct control input signal u from the measured output signal y such that the stability, and possibly performance criteria, are achieved. That is, K is interconnected with the system to control, and provides a third system.

The *Linear Fractional Transform* (LFT) [133] is a mathematical tool that enables to represent the system resulting of the interconnection from two systems. Consider P and K two LTI systems as represented on Figure 1.2, $P(s)$ being partitioned as,

$$\begin{pmatrix} z(s) \\ y(s) \end{pmatrix} = P(s) \begin{pmatrix} w(s) \\ u(s) \end{pmatrix} = \begin{pmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{pmatrix} \begin{pmatrix} w(s) \\ u(s) \end{pmatrix}.$$

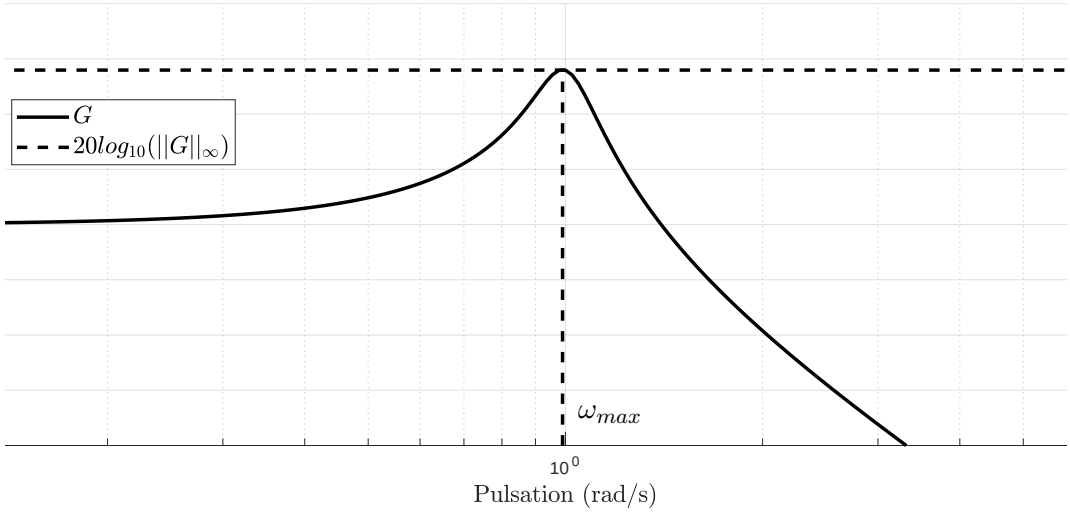
The LFT of $P(s)$ with $K(s)$ is given by

$$F(P(s), K(s)) = P_{11}(s) + P_{12}(s) G_l(s) (I - P_{22}(s) K(s))^{-1} P_{21}(s)$$

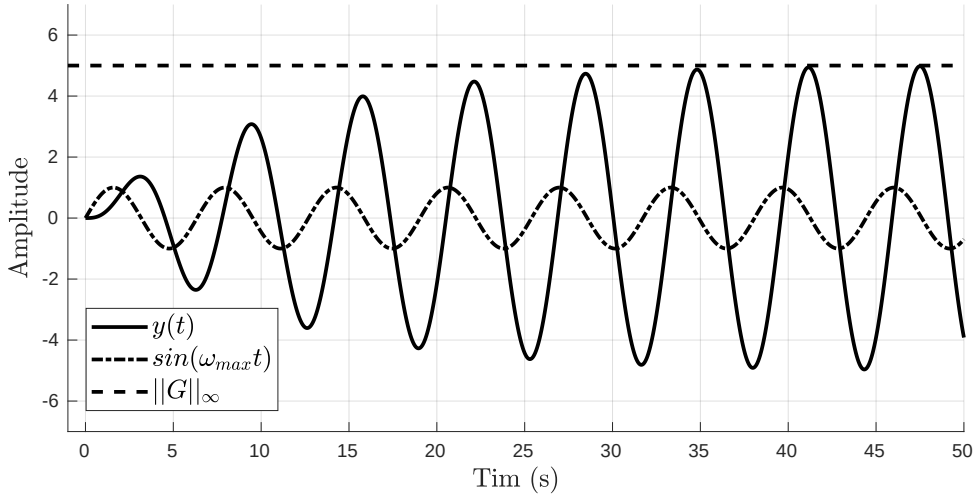
provided that $P(s)$ and $K(s)$ have compatible dimensions and $(I - P_{22}(s)K(s))$ is invertible.

Consider now that P and K are given by their minimal state space representations,

$$P = \left(\begin{array}{c|cc} A_P & B_w & B_u \\ \hline C_z & D_{wz} & D_{uz} \\ C_y & D_{wy} & D_{uy} \end{array} \right), \quad K = \left(\begin{array}{c|c} A_K & B_K \\ \hline C_K & D_K \end{array} \right)$$



(a) Bode diagram of G.



(b) Time response of G to a sine wave input.

Figure 1.1: Interpretation of the H_∞ norm in the frequency and time domains.

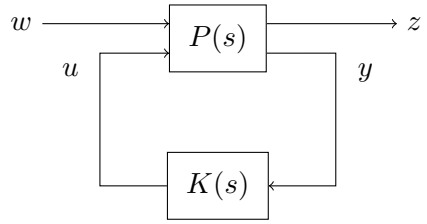


Figure 1.2: Linear fractional transform of two systems.

The system resulting from the interconnection of P with K has a (not necessarily minimal) state space representation given by

$$\left(\begin{array}{c|c} A_{cl} & B_{cl} \\ \hline C_{cl} & D_{cl} \end{array}\right),$$

with

$$A_{cl} = \begin{pmatrix} A_P + B_w(I - D_K D_{uy})^{-1} D_K C_y, & B_u(I - D_K D_{uy})^{-1} C_K \\ B_K(I - D_{uy} D_K)^{-1} C_z, & A_K + B_K(I - D_{uy} D_K)^{-1} D_{uy} C_K \end{pmatrix}$$

$$B_{cl} = \begin{pmatrix} B_w + B_u(I - D_K D_{uy}^{-}1) D_K D_{wy} \\ B_K(I - D_{uy} D_K)^{-1} D_{wy} \end{pmatrix}$$

$$C_{cl} = \begin{pmatrix} C_z + Duz(I - D_K duy)^{-1} D_K C_y, & D_{uz}(I - D_K D_{uy})^{-1} C_K \end{pmatrix}$$

$$D_{cl} = \begin{pmatrix} D_{wz} + D_{uz}(I - D_K D_{uy})^{-1} D_K D_{wy} \end{pmatrix}$$

When two systems are interconnected, an important notion is the internal stability. From the transfer matrix representing the input-output transfers, only the Bounded Input Bounded Output (BIBO) stability can be inferred. That is, the energy of output signal is bounded if the energy of input signal is bounded. However, BIBO stability cannot ensure that the internal signals are energy bounded. The internal stability can be defined as proposed in [123].

Definition. A system is internally stable if none of its components contain hidden unstable modes and the injection of bounded external signals at any place in the system results in bounded output signals measured anywhere in the system.

Following this definition, the internal stability of two interconnected systems can be verified via the generic interconnection scheme shown in Figure 1.3. The system is internally stable if the transfer matrix

$$\begin{pmatrix} (I - K(s)G(s))^{-1} & K(s)(I - G(s)K(s))^{-1} \\ G(s)(I - G(s)K(s))^{-1} & (I - G(s)K(s))^{-1} \end{pmatrix}, \tag{1.1}$$

from (w_1, w_2) to (z_1, z_2) is stable [133, p. 68], *i.e.* the poles of every transfers have negative real parts.

Remark. In practice, it often suffices to check that the denominators of the elements of the transfer matrix $(I - K(s)G(s))^{-1}$ have roots with negative real parts. However, due to

possible unstable poles-zeros cancellation in the transfers, it is necessary to consider all the transfers of the four blocks given by Equation (1.1). An illustrative example can be found in [123, p. 144].

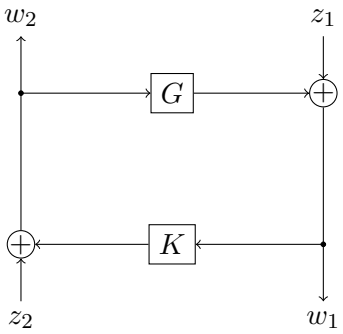


Figure 1.3: Generic regulation scheme for ensuring internal stability.

Alternatively, the *small gain theorem*, given by Theorem 1.2.4, also ensures internal stability from the infinity norm of K and G.

Theorem 1.2.4. *Suppose that G and K are stable. The interconnected system is internally stable if $||G||_\infty \cdot ||K||_\infty < 1$.*

Remark. The internal stability can directly be inferred from the stability of the state space realization of the interconnected system, but implies to compute this realization if the systems are given under their transfer matrix forms.

1.3 Systems with parametric uncertainties

A particular class of LTI systems considered in this thesis are the systems subject to parametric uncertainties. Such systems depends on parameters which values are not known precisely, but are within certain ranges. For example, the weight of a car varies in function of the number of passengers, or the aerodynamic drag coefficient of the vehicle is not known precisely. An uncertain system G depends on the vector of its uncertain parameters, denoted as $\theta \in \mathbb{R}^{n_\theta}$.

$$\text{Uncertain LTI system} \begin{cases} \dot{x} = A(\theta)x + B(\theta)u \\ y(t) = C(\theta)x + D(\theta)y \end{cases}$$

The set of admissible values of θ is denoted $\Theta \subset \mathbb{R}^{n_\theta}$. It will be assumed in the sequel that Θ is bounded. Uncertain systems are frequently met and are difficult to deal with since properties, such as stability, must be proved for any value of θ, *i.e.* for an infinite number of systems. This point is discussed in Section 1.7 dedicated to robustness analysis. Note that the dependency on θ is not necessarily linear or even convex.

Remark. It is important to note that the uncertain parameters do not vary over the time, they have a fixed value which is unknown. Systems which parameters vary over the time are called Linear Parameter Varying (LPV) systems, and their related control problems represent a particular field of control theory [118]. Only uncertain parameters are considered in this thesis.

1.4 General H_∞ synthesis problem

The H_∞ synthesis problem can be formulated as first proposed by Zames [131]. Given an LTI system P and a set $\mathcal{K}$ of LTI systems,

$$\begin{cases} \underset{K\in\mathcal{K}}{\text{minimize}} & ||F(P,K)||_\infty, \\ \text{subject to} & K \text{ ensures internal stability }, \end{cases} \tag{1.2}$$

where $F(P,K)$ is the LFT that describes the closed loop system represented in Figure 1.4, and $||F(P,K)||_\infty$ is its H_∞ norm.

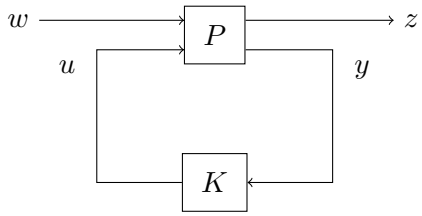


Figure 1.4: General H_∞ synthesis scheme.

In Figure 1.4, $w(t) \in \mathbb{R}^{n_w}$ represents the external input signals, $z(t) \in \mathbb{R}^{n_z}$ the error or performance output signals, u and y are the control and the measure outputs.
In the following and for the sake of clarity, $F(P,K)$ the transfer matrix from w to z is denoted $T_{w\to z}$. The transfer from the i^{th} input w_i, being the i^{th} element of the vector w, to the j^{th} output z_j is denoted $T_{w_i\to z_j}$.

$$T_{w\to z} = \begin{pmatrix} T_{w_1\to z_1} & \dots & T_{w_{n_w}\to z_1} \\ \vdots & \ddots & \vdots \\ T_{w_1\to z_{n_z}} & \dots & T_{w_{n_w}\to z_{n_z}} \end{pmatrix}$$

In addition, we also define the transfer matrix $T_{w\to z_j}$ from w to the particular output z_j.

$$T_{w\to z_j} = (T_{w_1\to z_j}, ..., T_{w_n\to z_j}).$$

1.5 Example of H_∞ synthesis for tracking problem

Consider for example a LTI system G and a controller K in a tracking error regulation loop as shown in Figure 1.5, where e represents the tracking error signal, $e(t) = r(t) - y(t)$, and d and n are two signals that model external perturbations. In this case, we have $w = (r, d)$ and $z = (z_1, z_2, z_3)$.

The principle of H_∞ synthesis is to minimize the infinity norm of objective channels $T_{w\to z_j}$. In Figure 1.5, the signals e, u and y are weighted by transfer functions Z_1, Z_2 and Z_3, such as non-desired behaviors of the closed loop system are penalized. For example, a non-desired behavior may be an important tracking error signal e at low frequencies (steady state). In this case, a high gain of Z_1 at low frequencies penalizes the sensitivity function $T_{w\to e}$ as desired.

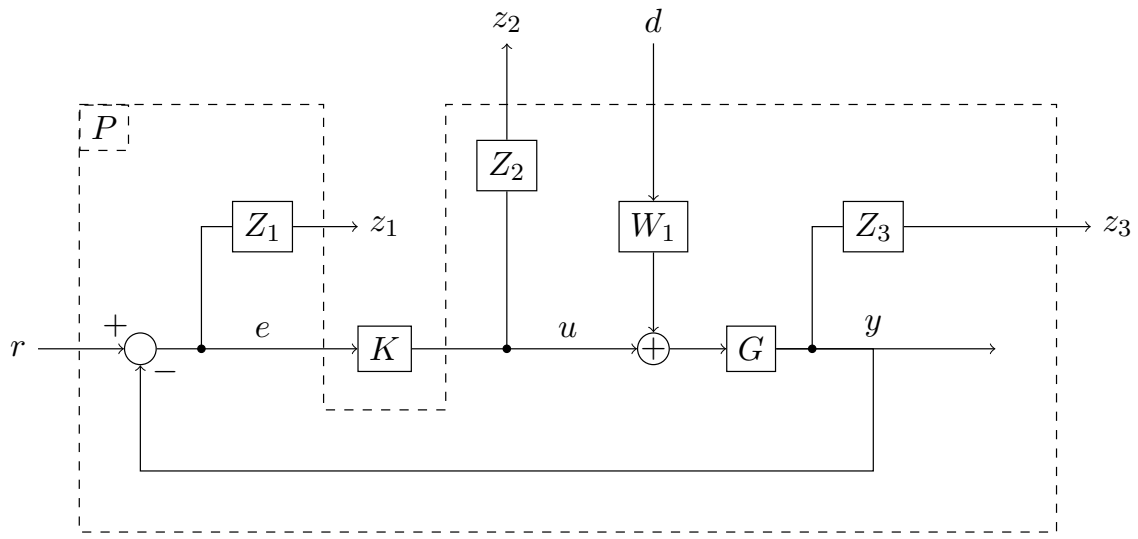


Figure 1.5: H_∞ sensitivity approach.

In that respect, z is the weighted counterpart of the non weighted signal $\tilde{z} = (e, u, y)$.
When formulating the automatic control problem as an H_∞ problem, design objectives are expressed as bounds on the gain of objective channels. These bounds are frequency templates designed from user specifications, and correspond to the inverse of weighting functions Z_j. An H_∞ constraint is expressed as $||T_{w\to z_j}||_\infty \leq 1$.

In the SISO case ($n_w = n_z = 1$), we have that $\sigma_{max}(T_{w\to z})(s) = |T_{w\to z}(s)|$ and an H_∞ constraint can be interpreted as a frequency dependent upper bound on the gain,

$$\begin{aligned} ||T_{w\to z}||_\infty \leq 1 &\iff \forall\omega,\ |T_{w\to z}(i\omega)| &&\leq 1 \\ &\iff \forall\omega,\ |Z_j(i\omega)T_{w\to\tilde{z}}(i\omega)| &&\leq 1 \\ &\iff \forall\omega,\ |Z_j(i\omega)||T_{w\to\tilde{z}}(i\omega)| &&\leq 1 \\ &\iff \forall\omega,\ |T_{w\to\tilde{z}}(i\omega)| &&\leq \frac{1}{|Z_j(i\omega)|} \\ &\iff \forall\omega,\ |T_{w\to\tilde{z}}(i\omega)| &&\leq |Z_j^{-1}(i\omega)| \end{aligned} \tag{1.3}$$

In addition, the inputs w can be also weighted with respect to their spectral content. In conclusion, the weighting functions Z_j express the design objectives and the W_i characterize the spectral contents of the input signals.

Once weighting functions Z_i and W_j are defined, the goal is to find K that minimizes every objectives $||T_{w\to z_j}||_\infty$ all at once, so that the closed loop system offers the desired behavior. In addition, K must internally stabilize the closed loop system. As a consequence the H_∞ problem can be formulated as proposed by the constraint satisfaction problem (CSP) 1.4.

$$\begin{cases} ||T_{w\to z_j}||_\infty \leq 1,\ \forall i \in \{1, \ldots, n_z\} \\ \text{s. t. } K\text{ensures internal stability} \\ K \in \mathcal{K} \end{cases} \tag{1.4}$$

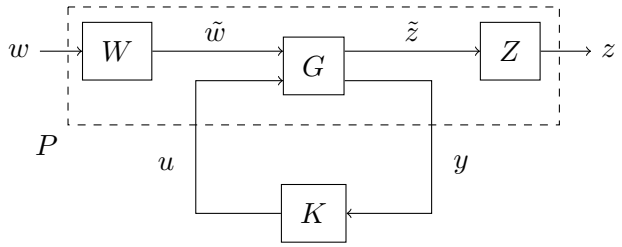


Figure 1.6: Interconnected systems

Problem 1.2 can be retrieved from CSP 1.4 thanks to inequality (1.5).

$$||T_{w\to z_j}||_\infty \leq ||T_{w\to z}||_\infty, \forall j \in \{1,\dots,n_z\}. \tag{1.5}$$

If the solution of Problem 1.2 is lower than 1, then it is also a solution to CSP 1.4. Problem 1.2 presents the main advantage to frame the n_z H_∞ constraints of CSP 1.4 as a single objective, and it is the classical way to present the H_∞ synthesis problem.
External disturbances can be taken into account and their influence on the closed loop system limited. This robustness w.r.t disturbances must not be misunderstood with the robustness w.r.t system uncertainties, addressed in Section 1.7. The H_∞ synthesis design process is introduced in this section because the global optimization approach, presented in Chapter 4, requires the understanding of its principle.

1.6 Resolution of H_∞ synthesis problem

The first method developed to solve Problem 1.2 was proposed by Doyle, Glover, Khargonekar and Francis, and relies on the resolution of an Algebraic Ricatti Equation (ARE) [42], and is generally referred to as the DGKF method form the initials of its authors. Later, Gahinet and Apkarian have shown in [47] that Problem 1.2 can be formulated as a Semi Definite Problem (SDP), and thus can be solved efficiently with dedicated optimization tools such as the interior point method [97]. However, both [42] and [47] impose that the order of K is equal to the one of P in order to make the problem convex, and therefore computationally tractable with the optimization methods and computer power available at this time. Therefore, this two methods generally provide high order controller, referred to as *full order* or *unstructured* controllers, since their structure cannot be chosen. In practice, small order controller are preferred since they are easier to implement and their dynamic is simpler for interpretation. These points have been the motivation of many publications about reduced order controller.

Remark. Structured controllers are also highly desirable to control LPV systems. Such systems have their parameters varying over the time, and the classical approach to control them is *gain scheduling* [46]. The principle lies in computing several controllers for possible values of the parameters, and to extrapolate a parameter varying controller. In this way, the final controller has its parameters varying along with the parameters of the LPV system, ensuring the stability and design criteria. Extrapolating a parameter varying controller from a set of controllers is much easier to achieve if they are structured.

In order to compute a low order controller, or *structured* controller, several approaches can be considered as explained in Figure 1.7.

Model order reduction can be used either to reduce the order of P or to reduce the order

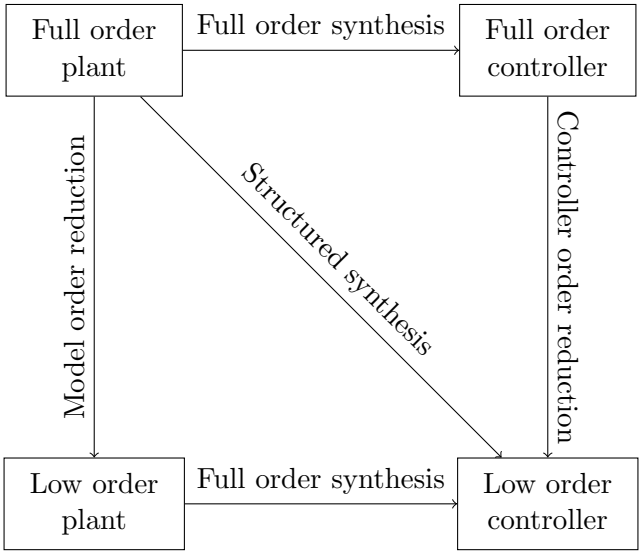


Figure 1.7: Low order controller synthesis process.

of the full order controller, resulting in a low order controller [132, 49]. However, it is often more interesting to synthesize directly a low order controller, especially when the difference between the orders of the controller and the system is large. A survey of model order reduction is provided in [58], but this topic is beyond the scope of this thesis, which focuses on the structured synthesis problem, given by Problem 1.6.1.

Problem 1.6.1. *Structured Synthesis (SS) problem:*

$$\begin{cases} \underset{K \in \mathcal{K}_s}{minimize} \; ||F(P,K)||_\infty \\ s.\ t.\ K \text{ ensures internal stability} \end{cases}$$

where $\mathcal{K}_s$ denotes the set of structured controllers. Problem 1.6.1 is not convex and not even smooth over $\mathcal{K}_s$.

Remark. Most of the existing synthesis methods compute fixed-order controllers, *i.e.* provides the state space matrices of K. A structured controller is slightly different since it has not only a fixed order but also a particular structure, and cannot necessarily be retrieved from a fixed order controller.

It is important to note that convex reformulations with LMIs have been very popular in the control field since the interior points method was developed, as evidenced by the number of citations of [23] (20,972 citations[1]) and [25] (40,937 citations [1]), and therefore that a

[1] according to google scholar the 18^{st} of July, 2018

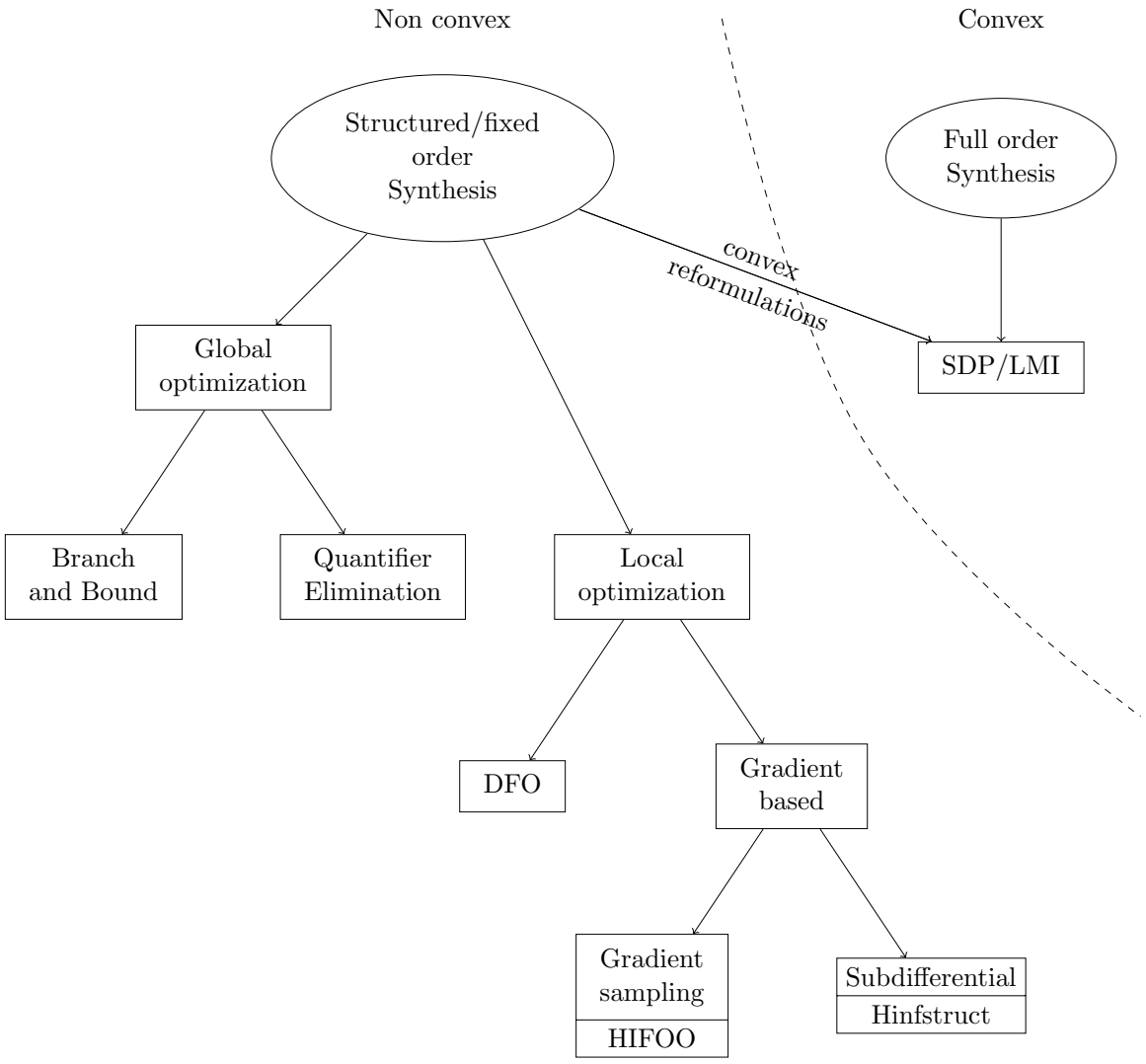


Figure 1.8: Chart of optimization approaches to the H_∞ synthesis problem.

large number of control problems, including Problem 1.6.1, has been approached with LMI. In that respect, several convex reformulation for fixed order [15, 57, 63] or structured synthesis [24, 115, 62] have been proposed until recently.

On the other hand, the fixed order/structured synthesis problem has also been addressed under its initial non-convex form with several local optimization methods. A general survey of those methods can be found in [6], to which can be added a recent work based on Derivative Free Optimization [121]. The (probably) two most famous approaches to the structured synthesis problem rely on line search methods [100], generalized to non-smooth problems:

- Burke et al.'s approach relies on a gradient sampling algorithm [28]. The idea is to compute the search direction not only from the gradient at the current iterate but also at nearby points. The HIFOO Matlab package [27] implements this algorithm for the

design of fixed order controller.

- Apkarian and Noll's approach is based on the Clarke sub-differential [32], which is a generalization of the gradient to non-smooth functions. The formula of the sub-differential of $||F(P,K)||_\infty$ has been proposed in [9]. The Hinfstruct Matlab package, and more recently Systune package, are based on this result and enable to compute structured controllers.

The few approaches based on global optimization to H_∞ synthesis mainly focus on SISO systems subject to parametric uncertainties [5, 85, 21], or does not consider the structured synthesis directly but the sub-optimal problem given by Problem 1.4 [60]. Further details about those method are provided in Section 1.8 dedicated to the robust structured synthesis. Figure 1.8 summarizes the different approaches to the SS problem.

1.7 Uncertainty representation and robustness analysis

When designing a control law, as models derive from approximations, it is necessary to take uncertainties into account. The uncertainties can come from the uncertain parameters of the model, or dynamics that has been neglected during the modelization process. In this thesis, only the first type of uncertainty are considered. If $G(\theta)$ is the uncertain system to control, then the augmented system $P(\theta)$ also depends on θ since it is built from G. The robustness analysis problem consists in verifying that, given a controller K, the H_∞ norm of the closed loop system is lower than one and the stability is achieved for any admissible value of θ. The first part of the problem is called the robust performance analysis and the second part the robust stability analysis.

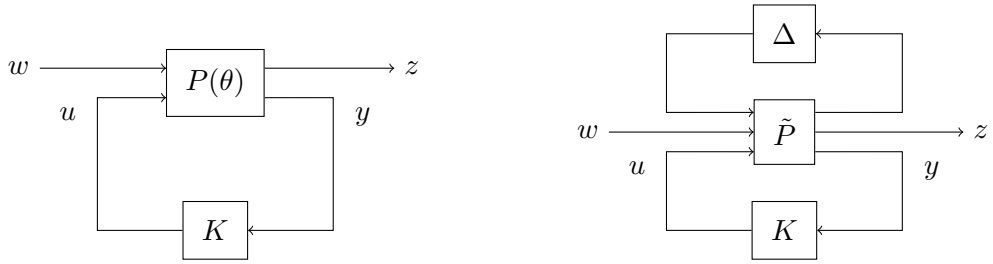
Problem 1.7.1. *Robustness analysis problem:*

$$\textit{Given } K\textit{, verify if} \begin{cases} ||F(P(\theta),K||_\infty \leq 1,\ \forall \theta \in \Theta \\ K \textit{ ensures internal stability }, \forall \theta \in \Theta \end{cases}$$

The robustness analysis problem is not convex in the general case, since $||T_{w\to z}(\theta)||_\infty$ is not convex over Θ and the subset of stable systems over Θ is not convex either. The most efficient tool for robustness analysis is the μ-analysis.

1.7.1 μ analysis approach

The μ-analysis is a robustness analysis method that was mainly developed during the 80's and the 90's [41, 130, 129], and is probably the most commonly used one. This method is based on the Δ-structure representation of the uncertainty. The initial closed loop system being the interconnection of the uncertain system $P(\theta)$ with K, given on Figure 1.9a, is rearranged as the interconnection of three LTI systems as shown on Figure 1.9a. $P(\theta)$ is reformulated as the interconnection of a nominal LTI plant $\tilde{P}$ with another structured block diagonal matrix $\Delta(s)$, such that $||\Delta||_\infty$ is bounded by one. That is, the initial set of plant $\{P(\theta)\,|\,\theta \in \Theta\}$ is reformulated as $\{F(\tilde{P},\Delta)\,|\,||\Delta||_\infty \leq 1\}$. This set of system is equivalent to the set $\{P(\theta)\,|\,\theta \in \Theta\}$ if Θ is an hyper-rectangle, but is an over-approximation otherwise.



(a) Initial problem for the robustness analysis. (b) Δ-structure reformulation.

Figure 1.9: Δ-structure representation of uncertainty.

Such an over-approximation may corresponds to disk centered on the nominal plant $\tilde{P}$ at a given pulsation in the Nyquist plan, as shown on Figure 1.10.

Let M be the LFT of $\tilde{P}$ and K, $M = F(\tilde{P}, K)$. The small gain theorem provides a sufficient condition for the internal stability of the interconnection of M with Δ. In addition the Structured Singular Value (SSV), generally denoted μ_Δ, generalizes the singular value σ for $F(M, \Delta)$. Consequently, the robust performance is achieved if

$$\frac{1}{k_{max}} = \sup_{\omega \geq 0} \mu_\Delta(M(i\omega)) \leq 1,$$

where k_{max} is called the *robustness margin*. Computing the structured singular value at a given pulsation is known to be NP-Hard [26]. However, an upper bound can be obtained by solving an LMI. Moreover, computing the SSV requires to compute μ_Δ at an infinite number of pulsation. As a consequence, the maximal SSV is generally approximated by

$$\max_{j \in \{1,\ldots,N\}} \mu_\Delta(M(i\omega_j))\}.$$

Such an approximation may lead to falsely conclude that the uncertain system achieves robust performance, but a solution has been proposed in [43] to overcome this problem. However, the Δ-structure is a conservative approximation of the initial system if Θ is not an hyper-rectangle.

1.7.2 Polytopic uncertain systems

Another approach to robustness analysis is to consider matrix polytopes to describe an uncertain system. A polytope $\mathcal{A}$ of $\mathbb{R}^{n \times n}$ with N vertices $A_i \in \mathbb{R}^n$ is defined as

$$\mathcal{A} = \left\{ A \in \mathbb{R}^{n \times n} | A = \sum_{i=1}^{N} \alpha_i A_i,\ \alpha_i \geq 0,\ \sum_{i=1}^{N} \alpha_i = 1 \right\}.$$

A *polytopic uncertain system* is described by a state space realization composed of four polytopes $(\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D})$. Such a representation of uncertain systems provides a framework to relax the stability and H_∞ performance analysis as LMIs [104] by studying only the vertices of the polytopes, that is a finite number of systems. One of the first method developed to analyze the stability of such systems was proposed in [18], by using a unique Lyapunov matrix for

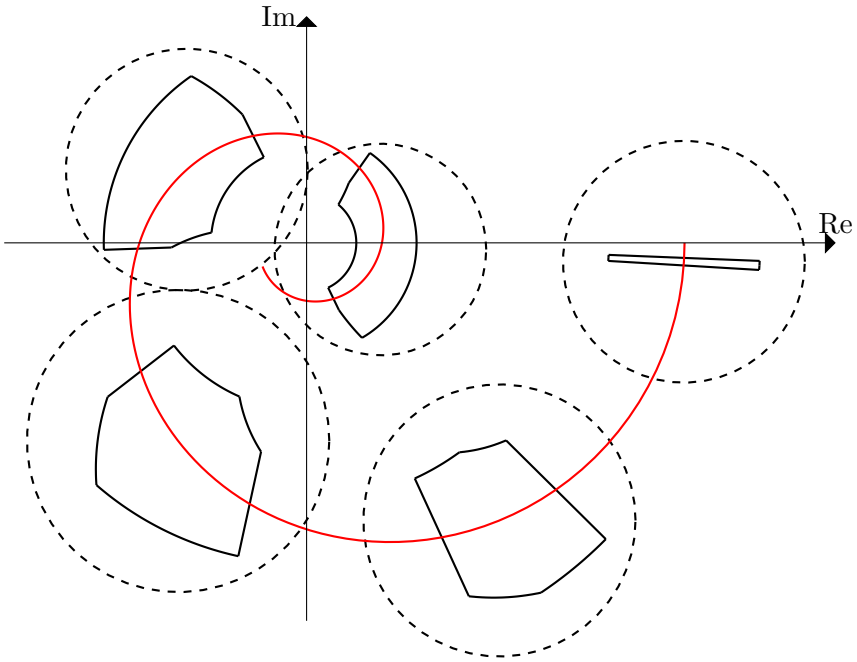


Figure 1.10: The exact regions of the uncertainty are delimited by hard line, and the convex approximations are represented by dashed circle, plotted at 5 pulsations in the Nyquist plan. The red curve represent the nominal system $\tilde{P}$.

all the matrices in $\mathcal{A}$. A less conservative approach relies on parameter-dependent Lyapunov matrices [105, 106], In addition, a guaranteed upper bound on the H_∞ norm of any systems in the polytopic set is provided by solving an LMI problem [104], which enables to prove the robust performance. However, the initial problem is relaxed to be formulated as an LMI one. Consequently, a polytopic uncertain system can be robustly stable but it may not be possible to prove it, and the H_∞ norm upper bound is not necessarily a close approximation of the exact value. Moreover, the interconnection of K with $P(\theta)$ cannot be described exactly by a polytopic uncertain system in the general case.

Another way to represent and analyze uncertain systems, but limited to the SISO case, rely on the frequency representation of the interconnection of $P(\theta)$ with K. This representation is not discussed here but in subsection 1.8.3, since it is used for robust synthesis but rarely for robustness analysis.

1.8 Robust structured synthesis

The structured H_∞ synthesis problem can be extended to the uncertain plant case, to provide Problem 1.6.1.

Problem 1.8.1. *Robust Structured Synthesis (RSS) problem:*

$$\begin{cases} \textit{Find } K \in \mathcal{K}_s \textit{ such that } ||F(P(\theta), K)||_\infty \leq 1 \\ \textit{subject to } K \textit{ ensures internal stability }, \forall \theta \in \Theta \end{cases}$$

Several approaches has been proposed to find a solution to the RSS problem, relying on the Δ-structure or the polytopic representation of the uncertainties.

1.8.1 Convex methods

The first approach to robust synthesis was the $D-K$ iteration [103], also called *μ-synthesis*, based on the minimization of the maximal SSV but provides full order controllers. Since then, a significant number of papers have been dedicated to the synthesis of static output feedback controllers (zero order controller) based either on the Δ-structure or on polytopic uncertainties. A well documented survey of those methods is provided in [113]. However, the general case where K has a fixed structured has received few attention due to the complexity of the RSS problem. In effect, this problem gather the structured synthesis and the robustness analysis problem. Several convex reformulations has been proposed in the SISO case with the Δ-structure uncertainty representation. For example, convex relaxations of the problem from geometrical interpretations of the stability and H_∞ norm in the Nyquist plan are proposed in [72, 71, 62].

1.8.2 Worst-case minimization approaches

Several non convex approaches relies on the minimization of the worst-case H_∞ norm. Under the Δ-structure of the uncertainty, the worst-case corresponds to

$$\max_{\Delta\in\mathbf{\Delta}} ||T_{wz}(\Delta)||_\infty,$$

where $\mathbf{\Delta}$ is the set of admissible uncertainty. Consequently, the RSS problem is reformulated as a minmax problem, which is not convex.

$$\begin{cases} \min\limits_{K\in\mathcal{K}_s} \max\limits_{\Delta\in\mathbf{\Delta}} ||T_{wz}(\Delta)||_\infty, \\ \text{s. t. } K \text{ ensures internal stability } \forall\Delta\in\mathbf{\Delta} \end{cases}$$

In particular, Apkarian and Noll have adapted local optimization algorithms, initially dedicated to the SS problem, to solve this minmax problem [36, 10, 7].

1.8.3 Parametric approaches to robust synthesis

The parametric approaches to the RSS problem rely on the frequency representation and are limited to SISO systems [40, 21]. The uncertain system $G(\theta)$ to control is given by a transfer function which coefficients of the polynomials are uncertain,

$$G(\theta,s)=\frac{\alpha_n s^n+\cdots+\alpha_1 s+\alpha_0}{\beta_n s^m+\cdots+\beta_1 s+\beta_0}, \tag{1.6}$$

with $\theta=(\alpha_n,\ldots,\alpha_0,\beta_m,\ldots,\beta_0)$. Not all coefficients are necessarily uncertain. Such a representation of the uncertainty cannot describe systems not depending linearly on the uncertain parameters, but is convenient for robust synthesis purpose as it will be explained in the sequel.

The controller K is given by its transfer function, parametrized by tunable real parameters. For example a PID controller has three parameters,

$$K(k,s) = k_p + \frac{k_i}{s} + k_d s,$$

with $k = (k_p, k_i, k_d)^T \in \mathbb{R}^{n_k}$. The stability of the sensitivity function S, corresponding to the transfer from r to e for the system depicted in Figure 1.5, is generally used as stability criterion.

$$S(k,\theta,s) = \frac{1}{1 + G(\theta,s)K(k,s)}.$$

Due to the representation of $G(\theta)$, both the numerator and the denominator of $S(k,\theta,s)$ are multivariate polynomials in s, k and θ. That is, the stability of the closed loop is ensured if the poles of $S(k,\theta,s)$ have negative real parts. Taking advantage of the Hurwitz criterion (see Theorem 1.2.2), the stability can be expressed as multivariate polynomials inequalities in k and θ,

$$R(k,\theta) \leq 0,\ \forall \theta \in \Theta.$$

Concerning the H_∞ criteria, they are expressed as objectives on SISO transfers $T_{w_i \to z_j}$ of the closed loop system, depending on k and θ. In this particular case, dealing with the H_∞ norm does not require to compute the maximal singular value but only the modulus of the transfer (see Equation 1.3). Hence, the H_∞ constraints can also be expressed as multivariate polynomial inequalities,

$$\begin{aligned} |T_{w_i \to z_j}(k,\theta,i\omega)| \leq 1,\ \forall \theta \in \Theta,\ \forall \omega \geq 0 \quad &\iff \frac{|N_{ij}(k,\theta,i\omega)|}{|D_{ij}(k,\theta,i\omega)|} \leq 1,\ \forall \theta \in \Theta,\ \forall \omega \geq 0 \\ &\iff |N_{ij}(k,\theta,i\omega)| - |D_{ij}(k,\theta,i\omega)| \leq 0, \\ &\qquad\qquad \forall \theta \in \Theta,\ \forall \omega \geq 0. \end{aligned}$$

In the end, the RSS problem, given by Problem 1.8.2, is formulated as multivariate polynomial inequalities that must be satisfied $\forall \theta \in \Theta$ and $\forall \omega \geq 0$, which are quantified constraints.

Problem 1.8.2.
Find k such that:

$$\begin{cases} |N_{ij}(k,\theta,i\omega)| - |D_{ij}(k,\theta,i\omega)| \leq 0, & \forall \theta \in \Theta,\ \forall \omega \geq 0. \\ R(k,\theta) \leq 0, & \forall \theta \in \theta. \end{cases}$$

Under this form, the parametric methods characterize $\mathbb{K}_r$ the subset of $\mathbb{R}^{n_k}$ such as $K(k)$ is solution to Problem 1.8.2. This approach differs from the worst case minimization formulation which provide an optimal (at least locally) controller w.r.t. the worst case. The parametric methods are global in the sense that the whole set $\mathbb{K}_r$ is characterized instead of providing only one feasible controller. To this extent, they are generally referred to as *set-membership* methods. In [40], Dorato compares several of these methods.
In particular, Anai et al. proposes to use quantifier elimination techniques [3, 4, 5] to deal

with the $\forall$ quantifier. Doing so, $\mathbb{K}_r$ is described exactly by a set of quantifier-free constraints. Another approach proposed by Malan et al. relies on Bernstein Branch and Bound algorithm [84, 85]. This algorithm has a classical Branch and Bound structure, and the Bernstein polynomials are used to compute the range of the polynomial constraints over continuous sets. This algorithm characterizes $\mathbb{K}_r$ by a set of interval vectors.
It is important to note that these methods, in order to be computationally tractable, require the RSS problem to be formulated as inequalities on multivariate polynomials, such that tools dedicated to polynomial analysis can be used [52, 96, 86]. In this way, the parametric methods are limited to uncertain system as given by Equation 1.6.

1.9 Proposed approach to H_∞ problems

In this chapter the three main problems related to H_∞ control has been presented, namely the structured synthesis, robustness analysis and robust structured synthesis problems, that are recalled here.

Structured synthesis problem

$$\begin{cases} \underset{K\in\mathcal{K}_s}{\text{minimize}} & ||F(P,K)||_\infty \\ \text{subject to} & K(k) \text{ ensures internal stability} \end{cases} \tag{1.7}$$

Robustness analysis problem

$$\text{Given } K\text{, verify if} \begin{cases} ||F(P(\theta),K||_\infty \leq 1,\ \forall\theta\in\Theta \\ K \text{ ensures internal stability },\ \forall\theta\in\Theta \end{cases}$$

Robust structured synthesis

$$\begin{cases} \text{Find } K \text{ such that } ||F(P(\theta),K)||_\infty \leq 1 \\ \text{subject to } K \text{ ensures internal stability },\ \forall\theta\in\Theta \end{cases}$$

One of the purpose of this thesis is to propose a unified approach of the three problems and global optimization tools to solve them under their original non convex formulation. This is done by following the parametric approach, on which are based the only known works relying on global optimization methods, since it does not required approximation of the uncertainty with the Δ-structure. However, we propose some improvements with respect to the parametric methods:

- Our approach is not limited to the SISO case, but can deal with the MIMO case.
- The multivariate polynomial formulation is not needed, and therefore a larger class of uncertain system can be dealt with.
- The RSS problem is solved under its worst case minimization.

In the sequel, Chapter 4 shows how the SS and the RSS problems can be formulated as minmax problems, and solved in a global way using the algorithms developed in Chapter 3, and how the robustness analysis problem can be expressed as a maximization problem that can be solved with the methods introduced in Chapter 2.

Chapter 2

Interval Analysis Tools and Branch and Bound algorithms

The purpose of this chapter is to introduce global optimization methods, namely branch and bound algorithms based on interval analysis. These algorithms are able to solve globally continuous non convex optimization problems, in the sense that they converge to the global optimizer. In addition, they provide an enclosure of the global optimum. If the local optimization methods are efficient in providing rapidly a good solution, they are unable to provide such an enclosure, and have no guarantee to converge to the global optimizer. However, in certain cases it is mandatory to have an enclosure of the global solution, as it will be shown in Chapter 4 when addressing the robustness analysis problem.
The present chapter is organized as follows. Section 2.1 presents the basics of interval analysis, and how reliable bounds can be computed on a functions over continuous set. Section 2.2 introduces advanced tools to deal with constraints. Finally, Section 2.3.2 gives two branch and bounds algorithms, based on the tools provided in the two previous sections, to solve constrained optimization problems in a global way. The purpose of this chapter is not only to introduce global optimization algorithms, but also to provide the tools that will be necessary in the next chapter dedicated to the global resolution of minmax problems.

2.1 Interval arithmetic

Interval arithmetic was first developed to deal with floating point numbers rounding errors. Since it is impossible to represent all the real numbers in computers, numerical computations necessarily lead to approximated numerical results. During the 50s and the 60s, methods to bound the approximation errors were found by using intervals. The key idea is to represent a real number by a couple of floating point numbers which enclosed it, and run the numerical computations with intervals instead of scalars in order to get a guaranteed enclosure of the result. In particular, Moore published *Interval Analysis* [95] in 1966 after his doctorate about the use of intervals for numerical errors in computer, which can be considered as the reference book on interval analysis.
An important property is that computations based on intervals provide guaranteed enclosure of the numerical results not only for the non representable real of interest, but for all the

reals in the interval. That is, operations on continuous sets are performed. This is of major interest since guaranteed bounds of non-convex functions can be computed over continuous sets, that can be taken advantage of for global optimization purposes.

2.1.1 Intervals

An Interval is a closed connected subset of $\mathbb{R}$. Intervals are denoted using boldface letters $\mathbf{x}$. A non-empty interval $\mathbf{x}$ is represented by its endpoints:

$$\mathbf{x} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}] = \{x \,|\, \underline{\mathbf{x}} \leq x \leq \overline{\mathbf{x}}\},$$

with $\underline{\mathbf{x}} \in \mathbb{R} \cup \{-\infty\}$, $\overline{\mathbf{x}} \in \mathbb{R} \cup \{+\infty\}$ and $\underline{\mathbf{x}} \leq \overline{\mathbf{x}}$. $\underline{\mathbf{x}}$ is the lower bound of the interval $\mathbf{x}$ and $\overline{\mathbf{x}}$ is its upper bound. If $\underline{\mathbf{x}} = \overline{\mathbf{x}}$, $\mathbf{x}$ is said to be degenerate. The set of real intervals is denoted $\mathbb{IR}$, $\mathbf{x} \subset \mathbb{IR}$.
The *width* of a non-empty interval $\mathbf{x}$ is

$$w(\mathbf{x}) \triangleq \overline{\mathbf{x}} - \underline{\mathbf{x}}.$$

In the case where $\underline{\mathbf{x}} = -\infty$ or $\overline{\mathbf{x}} = +\infty$, $w(\mathbf{x}) = \infty$.
The *midpoint* of a bounded interval ($\underline{\mathbf{x}} \neq -\infty$ and $\overline{\mathbf{x}} \neq +\infty$) is

$$mid(\mathbf{x}) \triangleq \frac{\overline{\mathbf{x}} - \underline{\mathbf{x}}}{2}.$$

Since an interval is a set, the classical set operations can be defined for intervals. The union of two intervals $\mathbf{x}$ and $\mathbf{y}$ is defined by

$$\mathbf{x} \cup \mathbf{y} \triangleq \{z |\, z \in \mathbf{x} \text{ or } z \in \mathbf{y}\}.$$

The intersection of $\mathbf{x}$ and $\mathbf{y}$ is

$$\mathbf{x} \cap \mathbf{y} \triangleq \{z |\, z \in \mathbf{x} \text{ and } z \in \mathbf{y}\}.$$

$\mathbf{x}$ deprived of $\mathbf{y}$ is

$$\mathbf{x} \setminus \mathbf{y} \triangleq \{z,\, z \in \mathbf{x} \text{ and } z \notin \mathbf{y}\}.$$

Note that the intersection of two intervals is an interval (possibly empty), but generally the union and the set difference are not. In addition, we define the *interval hull* of two intervals $\mathbf{x}$ and $\mathbf{y}$ as the smallest interval containing their union,

$$\mathbf{x} \sqcup \mathbf{y} \triangleq [\min(\underline{\mathbf{x}}, \underline{\mathbf{y}}), \max(\overline{\mathbf{x}}, \overline{\mathbf{y}})].$$

A *box* $\mathbf{x}$ is the Cartesian product, denoted by $\times$, of intervals,

$$\mathbf{x} \triangleq \mathbf{x}_1 \times \cdots \times \mathbf{x}_n,\ \mathbf{x}_i \in \mathbb{IR}.$$

Alternatively, a box is a vector of intervals, that is a subset of $\mathbb{IR}^n$

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{pmatrix}.$$

For convenience, intervals and boxes are denoted using boldface since an interval is a box of dimension one. The set of n-dimensional boxes is denoted by $\mathbb{IR}^n$.

The definition of midpoint can be extended to boxes. The midpoint of a bounded box $\mathbf{x} \in \mathbb{IR}^n$ is the vector

$$mid(\mathbf{x}) \triangleq \begin{pmatrix} mid(\mathbf{x}_1) \\ \vdots \\ mid(\mathbf{x}_n) \end{pmatrix},$$

and the width of a bounded non empty box $\mathbf{x}$ is defined by

$$w(\mathbf{x}) \triangleq \max_{1 \le i \le n} w(\mathbf{x}_i).$$

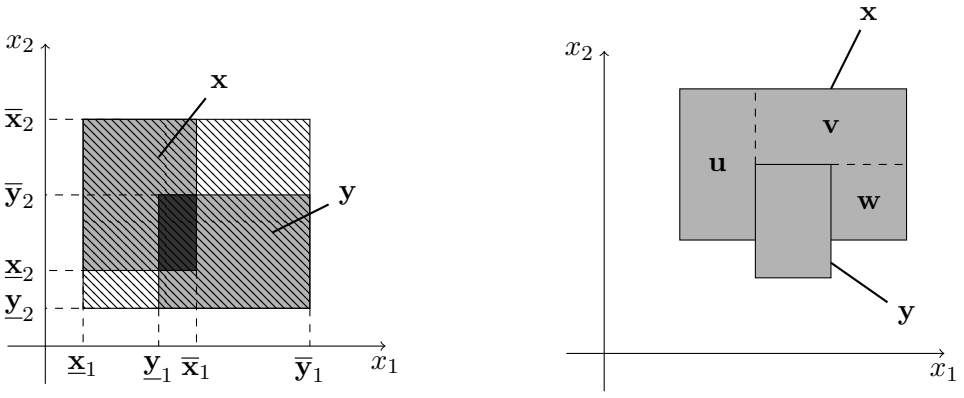
Also, the intersection of two boxes $\mathbf{x}$ and $\mathbf{y}$ is a box and is given by

$$\mathbf{x} \cap \mathbf{y} \triangleq \{\mathbf{x}_1 \cap \mathbf{y}_1 \times ... \times \mathbf{x}_n \cap \mathbf{y}_n\}.$$

As for intervals, the union of two boxes $\mathbf{x}$ and $\mathbf{y}$ is not a box in the general case. The interval hull operator is extended to boxes as the *box hull* operator, also denoted $\sqcup$. $\mathbf{x} \sqcup \mathbf{y}$ is the smallest box containing the union of $\mathbf{x}$ and $\mathbf{y}$,

$$\mathbf{x} \sqcup \mathbf{y} \triangleq \{\mathbf{x}_1 \sqcup \mathbf{y}_1 \times ... \times \mathbf{x}_n \sqcup \mathbf{y}_n\}.$$

Figure 2.1a illustrates set operations on two boxes. The box hull is represented by the hatched area and the intersection by the dark gray area.
The set difference of two boxes can be represented as the union of boxes as shown on Figure 2.1b.



(a) Intersection and box hull of two boxes

(b) Set difference of two boxes: $\mathbf{x} \setminus \mathbf{y} = \mathbf{u} \cup \mathbf{v} \cup \mathbf{w}$.

Figure 2.1: Set operations on boxes.

2.1.2 Interval Arithmetic

Interval arithmetic extends the unary and binary operators from operation on real numbers to the intervals, such that the resulting interval is the smallest one containing all the images of the initial intervals.
Consider two intervals $\mathbf{x}$ and $\mathbf{y}$, and a binary operator $\diamond$ being $+,-,\cdot$, or $/$. The operator $\diamond$ is extended to the intervals as it follows:

$$\begin{aligned}&\mathbf{x} \diamond \mathbf{y} \triangleq \{x \diamond y,\, x \in \mathbf{x},\, y \in \mathbf{y}\} \subseteq \mathbb{IR},\\ &\diamond : \mathbb{IR} \times \mathbb{IR} \mapsto \mathbb{IR}.\end{aligned}$$

The four basic binary operators are extended according to the following formulas,

$$\begin{aligned}\mathbf{x}+\mathbf{y} &= [\underline{\mathbf{x}}+\underline{\mathbf{y}}, \overline{\mathbf{x}}+\overline{\mathbf{y}}],\\ \mathbf{x}-\mathbf{y} &= [\underline{\mathbf{x}}-\overline{\mathbf{y}}, \overline{\mathbf{x}}-\underline{\mathbf{y}}],\\ \mathbf{x}\cdot\mathbf{y} &= [\min(\overline{\mathbf{x}}\cdot\overline{\mathbf{y}}, \underline{\mathbf{x}}\cdot\overline{\mathbf{y}}, \overline{\mathbf{x}}\cdot\underline{\mathbf{y}}, \underline{\mathbf{x}}\cdot\underline{\mathbf{y}}), \max(\overline{\mathbf{x}}\cdot\overline{\mathbf{y}}, \underline{\mathbf{x}}\cdot\overline{\mathbf{y}}, \overline{\mathbf{x}}\cdot\underline{\mathbf{y}}, \underline{\mathbf{x}}\cdot\underline{\mathbf{y}})],\\ \mathbf{x}/\mathbf{y} &= \mathbf{x}\cdot[\frac{1}{\overline{\mathbf{y}}}, \frac{1}{\underline{\mathbf{y}}}] \text{ if } 0 \notin \mathbf{y}.\end{aligned}$$

The extension of monotonic unary operators to interval is straightforward, since the order of the elements in the interval is either preserved or reversed. Here are some examples of monotonic unary functions.

$$\begin{aligned}\exp(\mathbf{x}) &= [\exp(\underline{\mathbf{x}}), \exp(\overline{\mathbf{x}})],\\ \log(\mathbf{x}) &= [\log(\underline{\mathbf{x}}), \log(\overline{\mathbf{x}})],\\ \mathbf{x}^{-1} &= [\overline{\mathbf{x}}^{-1}, \underline{\mathbf{x}}^{-1}] \text{ if } 0 \notin \mathbf{x}.\end{aligned} \tag{2.1}$$

More generally, it is possible to extend the common operators, such as trigonometric ones, to intervals. If the operators are not monotonic, the local monotonicity and known bounds can be used. For example, the formula for the square function is

$$\operatorname{sqr}(\mathbf{x}) = \begin{cases} [\min(\operatorname{sqr}(\underline{\mathbf{x}}), \operatorname{sqr}(\overline{\mathbf{x}})), \max(\operatorname{sqr}(\underline{\mathbf{x}}), \operatorname{sqr}(\overline{\mathbf{x}}))] & \text{if } 0 \notin \mathbf{x},\\ [0, \max(\operatorname{sqr}(\underline{\mathbf{x}}), \operatorname{sqr}(\overline{\mathbf{x}}))] & \text{else.}\end{cases}$$

The interval arithmetic has been implemented in several numerical libraries, such as INTLAB [111] (in Matlab), PROFIL/BIAS in C [78], GAOL [54] and Filib++ [82] in C++. All these libraries implement the *rounded interval arithmetic*. The initial interval is approximated by the smallest one which contains it and is represented by floating point numbers. In this way, the computations can be performed by a computer and the results are guaranteed. The IEEE 1788 standard is dedicated to interval computation and rounded interval arithmetic [2]. In this thesis, IBEX [30] library is used because it implements advanced tools such as contractors, that will be presented in Section 2.2.1.

2.1.3 Inclusion functions

Let f be a function mapping $\mathbb{R}^n$ into $\mathbb{R}^m$, and $\mathbb{X}$ a subset of $\mathbb{R}^n$. The *direct image* of $\mathbb{X}$ by f is defined by

$$f(\mathbb{X}) = \{f(x) |\, x \in \mathbb{X}\}\,.$$

An inclusion function $\mathbf{f}$ of f maps $\mathbb{IR}^n$ into $\mathbb{IR}^m$ and respects the property

$$\forall \mathbf{x} \in \mathbb{IR}^n,\ f(\mathbf{x}) \subseteq \mathbf{f}(\mathbf{x}).$$

Figure 2.2 illustrates the concept of inclusion function. $\mathbf{f}$ is said to be *minimal* if $\forall \mathbf{x} \in \mathbb{IR}^n$, $\mathbf{f}(\mathbf{x})$

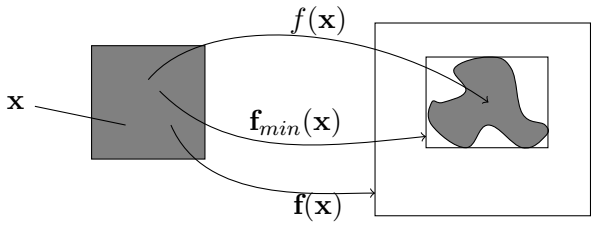


Figure 2.2: Inclusion function $\mathbf{f}$ and minimal inclusion function f_{min}

is the smallest box which contains $f(\mathbf{x})$. In Figure 2.2, $\mathbf{f}_{min}$ illustrates a minimal inclusion function of f. An inclusion function is not necessarily minimal, in that case the inclusion function is said to be *pessimistic*. The difference between $\mathbf{f}(\mathbf{x})$ and $\mathbf{f}_{min}(\mathbf{x})$ is referred to as the *pessimism* of the evaluation, and is not trivial to evaluate if any minimal function is known [29]. $\mathbf{f}$ is *inclusion monotonic* if

$$\mathbf{x} \subset \mathbf{y} \implies \mathbf{f}(\mathbf{x}) \subset \mathbf{f}(\mathbf{y}),$$

and $\mathbf{f}$ is *thin* if for any vector x of $\mathbb{R}^n$, $\mathbf{f}(x) = f(x)$.
In order to introduce another property of inclusion functions, consider $\mathbf{x}_l$ a sequence of boxes. $\mathbf{f}$ is *convergent* if

$$\lim_{l\to\infty} w(\mathbf{x}_l) = 0 \implies \lim_{l\to\infty} w(\mathbf{f}(\mathbf{x}_l)) = 0.$$

2.1.4 Natural inclusion functions

Now that the concept of inclusion function is defined, the question is how to build one. One way is to use interval arithmetic introduced in Subsection 2.1.2.

Definition. Consider the function

$$\begin{aligned} f:\ & \mathbb{R}^n \to \mathbb{R} \\ & (x_1, \ldots, x_n) \mapsto f(x_1, \ldots, x_n), \end{aligned}$$

expressed as a finite composition of the operators $+, -, ., /$ and elementary functions (sin, cos, log, exp, ...). Such a function will be referred to as *explicit* function in the sequel. The function

$$\mathbf{f} : \mathbb{IR}^n \to \mathbb{IR}$$

obtained by replacing each real variable x_i by an interval variable $\mathbf{x}_i$ and each operator or function by its interval counterpart is the *natural inclusion function* of f.

Theorem 2.1.1. *[69]*
$\mathbf{f}$ *is an inclusion function for* f*. Moreover,* $\mathbf{f}$ *is thin and monotonic.*

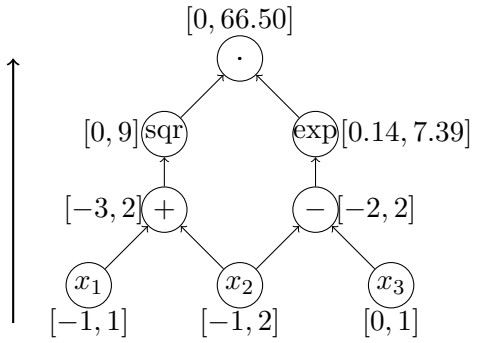


Figure 2.3: Interval computation done by the natural inclusion function of f.

Proposition 2.1.1. *[69]*
If f involves only continuous operators and continuous elementary functions, then $\mathbf{f}$ *is convergent. If, moreover, each of the variables $(x_1, \ldots, x_n)$ occurs at most once in the formal expression of f, then* $\mathbf{f}$ *is minimal.*

In order to illustrate natural inclusion function, consider the function

$$\begin{aligned} f: \quad & \mathbb{R}^3 \to \mathbb{R} \\ & (x_1, x_2, x_3) \mapsto \mathrm{sqr}(x_1 + x_2) \cdot \exp(x_2 - x_3). \end{aligned}$$

f can be represented by its *expression tree* which decomposes it in nodes as shown of Figure 2.3. Each node is either a variable or an operator. The bottom nodes, which correspond to the variables, are called the *leaves* and the top node is called the *root*. In order to perform an interval computation, the interval value of the leaves are propagated to the root by calling repetitively the interval counterpart of the operators. This propagation is illustrated on Figure 2.3 with $\mathbf{x}_1 = [-1, 1]$, $\mathbf{x}_2 = [-1, 2]$, and $\mathbf{x}_3 = [0, 1]$. The result of the interval computation is $\mathbf{f}([-1, 1], [-1, 2], [0, 1]) = [0, 32.64]$, with $\mathbf{f}$ the natural inclusion function of f.
The natural inclusion functions are not minimal in general, because the dependency between the occurrences of the same variable in the expression is lost. When the interval computation is done, if the same interval variable appears n times in the function, it will be replaced by n intervals which are independent. For example, consider the function

$$f(x) = \mathrm{sqr}(x).$$

We have that $\mathbf{f}([-1, 1]) = [0, 1]$. However, if f is rewritten as

$$f(x) = x \cdot x,$$

we obtain $\mathbf{f}([-1, 1]) = [-1, 1] \cdot [-1, 1] = [-1, 1]$. The result remains a reliable enclosure but $\mathbf{f}$ is not minimal anymore.
When using natural inclusion functions, it is generally preferable to have as less variable repetition as possible in order to limit the pessimism. Ideally, an expression where each variable occurs only once is searched, for in this case the natural inclusion function is minimal

as stated in Theorem 2.1.1.
The natural inclusion functions can also be defined for a vector valued function

$$\begin{aligned} f: \quad & \mathbb{R}^n \to \mathbb{R}^m \\ & x \mapsto f(x) = (f_1(x), \dots, f_m(x))^T \end{aligned}$$

by

$$\begin{aligned} \mathbf{f}: \quad & \mathbb{R}^n \to \mathbb{R}^m \\ & \mathbf{x} \mapsto \mathbf{f}(\mathbf{x}) = (\mathbf{f}_1(\mathbf{x}), \dots, \mathbf{f}_m(\mathbf{x}))^T \end{aligned}$$

where $\mathbf{f}_i$ is the natural inclusion function of f_i. One important point to mention is that the image of a box by a vector valued function f is generally not a box, due to the dependencies of its coordinate functions f_i to the same variable x, as illustrated on Figure 2.2. However, inclusion functions can only compute a box which contains the exact image since they are valued in $\mathbb{IR}^m$ by definition. This approximation is known as the *wrapping effect*.
Although natural inclusion functions suffer from overestimation and wrapping effect, they remain an efficient tools to perform set-membership computations. The main inconvenience of the natural inclusion functions and interval arithmetic is that the considered sets are limited to boxes, but it is also an advantage since boxes are easy to represent in computer and set operations can be easily done (see Subsection 2.1.1). In this thesis, only natural inclusion functions will be used. However, there exist other inclusion functions based on the gradient such as the *centered form* or the *Baumann form* [19], or using different arithmetics such as affine arithmetic [37].

2.2 Constraint Satisfaction Problem and contractors

A *constraint satisfaction problem* (CSP) consists in finding solutions which are consistent with respect to a set of rules or facts, called *constraints*. A CSP is composed of:

- a finite sequence of variables $x = (x_1, \dots, x_n)$,
- an initial domain $\mathbb{X}$,
- a list of constraints $\{C_1, \dots, C_p\}$.

A vector $x \in \mathbb{X}$ is said to be consistent, or *feasible*, if it satisfies all the constraints $\mathcal{C}_i$. On the contrary, if at least one of the constraints is not satisfied, x is said to be inconsistent or *infeasible*. The *feasible set* of a CSP is the set of all feasible elements of the initial domain. By extension, a box will be said feasible if all its elements are feasible.
In this thesis, the constraints considered are inequalities or equalities on functions for which natural inclusion functions can be defined. That is, a constraint C_i will be expressed as $g_i(x) \leq 0$ or $h_j(x) = 0$. Due to the rounding error inherent to the implementation of interval arithmetic on computers, the equality constraints are practically almost impossible to prove. That is why they are generally reformulated as "almost" equalities with a small tolerance error $\epsilon > 0$, leading to two inequalities.

$$h(x) \in [-\epsilon, \epsilon] \iff \begin{cases} h(x) \leq \epsilon, \\ h(x) \geq -\epsilon. \end{cases}$$

In the following, this reformulation is used and henceforth only inequality constraints are considered.
Inclusion functions are particularly interesting to deal with CSP. By enclosing the exact range of a function over a continuous set, inclusion functions enable to prove whether a constraint is respected or not over a box, in a guaranteed way. Consider first a unique constraint $g(x) \leq 0$. Three cases are possible:

- $\overline{\mathbf{g}(\mathbf{x})} \leq 0 \implies \forall x \in \mathbf{x},\ g(x) \leq 0$, which proves that $\mathbf{x}$ is feasible,
- $\underline{\mathbf{g}(\mathbf{x})} > 0 \implies \forall x \in \mathbf{x},\ g(x) > 0$, which proves that $\mathbf{x}$ is non-feasible,
- $0 \in \mathbf{g}(\mathbf{x})$, nothing can be concluded.

By extension, a box can be proved to be feasible with respect to a CSP if it is proved to be feasible for all the constraints composing the CSP. On the contrary, the box is infeasible if it is infeasible for at least one constraint. In the other cases, nothing can be concluded. In these last cases, constraint propagation techniques can be used to reduce a box to its feasible subset.

2.2.1 Constraint propagation: Forward Backward

When considering the initial domain of a CSP, *constraint propagation* resides in rejecting any elements of this set that are not feasible with respect to the constraints. For example, if you discard the Manhattan cocktail when choosing a drink in a bar because you think mixing whisky with any other liquid is sacrilegious, you perform a constraint propagation. In particular, interval arithmetic enables us to perform constraint propagation over a box. In order to illustrate constraint propagation based on interval arithmetic, consider the CSP given by

$$\begin{cases} x_1 + x_2 \ \leq 0, \\ x_1 \in [-1.5, 1],\ x_2 \in [0, 2]. \end{cases}$$

Performing interval computation with the initial domains of x_1 and x_2, we obtain $x_1 + x_2 \in [-1.5, 3]$. However, we search (x_1, x_2) such that $x_1 + x_2 \in [-\infty, 0]$. As a consequence, the feasible values of x_1 and x_2 are such that $x_1 + x_2 \in [-0.5, 3] \cap [-\infty, 0] = [-0.5, 0]$, which implies that x_1 belongs to the domain $[-0.5, 0] - x_2$ for any value of x_2 in $[0, 2]$. That is, $x_1 \in [-0.5, 0] - [0, 2] = [-2.5, 0]$ reducing its domain to $[-0.5, 1] \cap [-2.5, 0] = [-0.5, 0]$. The same can be done with the domain of x_2, $x_2 \in [0, 2] \cap ([-0.5, 0] - [-1, 0]) = [0, 1.5]$.
This simple example shows how interval arithmetic allows to propagate constraints involving binary operators. This constraint propagation technique can be extended to any constraint involving a function which expression tree is available [87]. Consider the CSP

$$\begin{cases} \exp(x_1 + x_2) - x_3 \cdot x_4 \leq 0, \\ (x_1, x_2, x_3, x_4) \in [-2, 3] \times [0, 2] \times [-1, 2] \times [1, 3]. \end{cases}$$

The propagation procedure can be applied to the whole tree, as illustrated in Figure 2.4. The first phase of the procedure is the *forward* evaluation, already presented in Section 2.1.4. Then, the interval result is reduced with respect to the constraint. The second phase is the

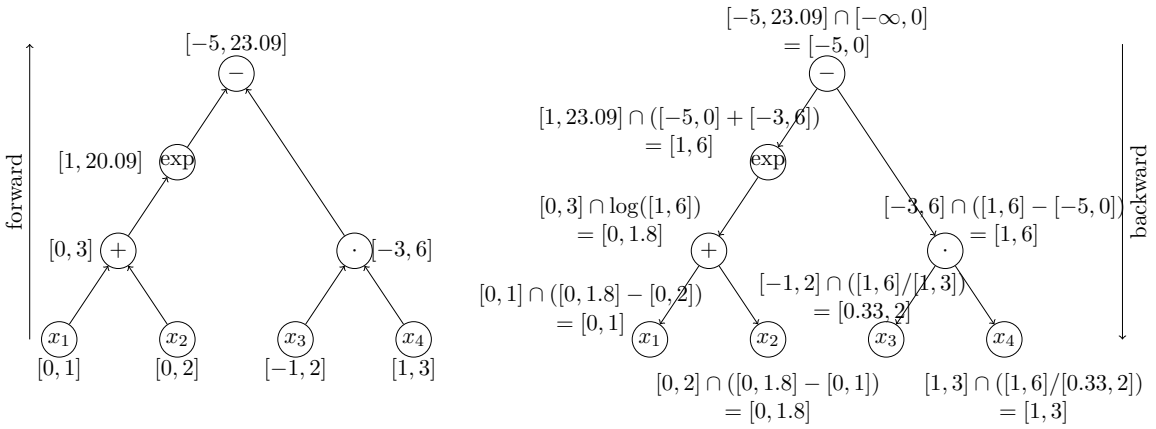


Figure 2.4: Constraint propagation procedure.

backward propagation, which propagates the reduced interval from the root to the leaves. At each node of the tree, the constraint propagation procedure explained for the unary and binary operator in the first CSP example is applied. The constraint is finally propagated to the initial domains at the leaves, reducing them. The procedure can be called successively until a fixed point is reached. This constraint propagation technique will be referred to as the *forward-backward* propagation.
Other constraint propagation techniques based on interval exist, most of them being based on the resolution of linear programs in the interval case [98, 61]. More generally, such constraint propagation procedures are called *contractors* or *filtering algorithms*.

2.2.2 Contractors

Contractors are mathematical functions that propose a formalism for constraint propagation, providing a unified framework for different constraint propagation techniques. Hence, contractors aim to reduce, or *contract*, an initial domain to its feasible subset. A contractor is formally defined as follows.

Definition. Consider $\mathcal{H}$ a CSP and $\mathbb{S}$ the feasible set of $\mathcal{H}$. The function

$$\begin{aligned} \mathcal{C}: \quad & \mathbb{IR}^n \to \mathbb{IR}^n \\ & \mathbf{x} \mapsto \mathcal{C}(\mathbf{x}) \end{aligned}$$

is a contractor for $\mathcal{H}$ if

$$\begin{cases} \mathbb{S} \subseteq \mathcal{C}(\mathbf{x}) & (\textit{consistency}), \\ \mathcal{C}(\mathbf{x}) \subseteq \mathbf{x} & (\textit{contraction}). \end{cases}$$

The first property ensure that no solution of $\mathcal{H}$ is lost, and the second that non feasible solutions are discarded or at least that the initial box does not grow. We will say equivalently that $\mathcal{C}$ is a contractor for the CSP $\mathcal{H}$ or its feasible set $\mathbb{S}$.

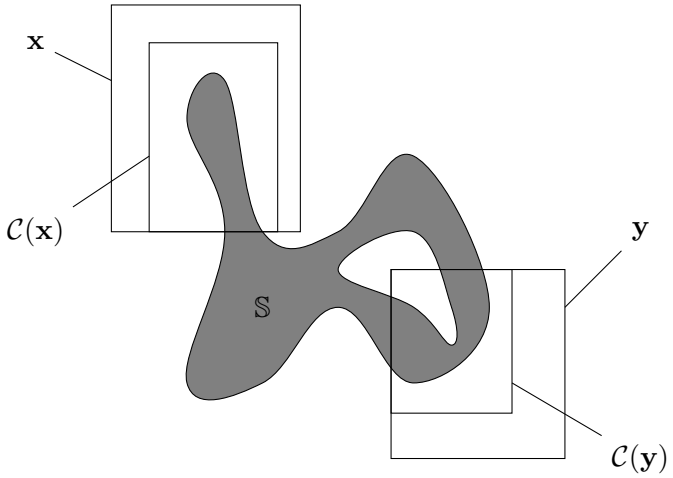


Figure 2.5: Contractor.

Analogue properties of inclusion functions can be defined for contractors.
A contractor $\mathcal{C}$ is *monotonic* if

$$\mathbf{x} \subseteq \mathbf{y} \implies \mathcal{C}(\mathbf{x}) \subseteq \mathcal{C}(\mathbf{y}).$$

A contractor is *minimal* if it reduces the initial domain to the smallest box containing the solutions. Figure 2.5 illustrates how a contractor performs. From the definition of contractor, the forward-backward propagation can be defined as a contractor, which will be called the *forward-backward contractor*, since both the consistency and contraction properties are respected [69]. In this thesis, only this contractor will be used due to its weak computational cost.

2.2.3 Contractors programming

When dealing with a CSP composed of several constraints, it is convenient to define a unique contractor for this CSP. To do so, the forward backward contractors defined for each particular constraints of the CSP can be combined to provide a unique contractor for the feasible set of the CSP. This concept of combining contractors is known as *contractor programming*, and has been formalized in [30]. Consider two contractors $\mathcal{C}_1$ and $\mathcal{C}_2$ mapping $\mathbb{IR}^n$ into $\mathbb{IR}^n$. $\mathcal{C}_1$ and $\mathcal{C}_2$ can be combined as it follows:

$$\begin{array}{ll} (\mathcal{C}_1 \cap \mathcal{C}_2)(\mathbf{x}) = \mathcal{C}_1(\mathbf{x}) \cap \mathcal{C}_2(\mathbf{x}) & (intersection), \\ (\mathcal{C}_1 \cup \mathcal{C}_2)(\mathbf{x}) = \mathcal{C}_1(\mathbf{x}) \sqcup \mathcal{C}_2(\mathbf{x}) & (union), \\ (\mathcal{C}_1 \circ \mathcal{C}_2)(\mathbf{x}) = \mathcal{C}_1(\mathcal{C}_2(\mathbf{x})) & (composition). \end{array}$$

In the left hand side of the equations, the contractors must be understood (abusively) as subsets of $\mathbb{IR}^n$, and define a third contractor.

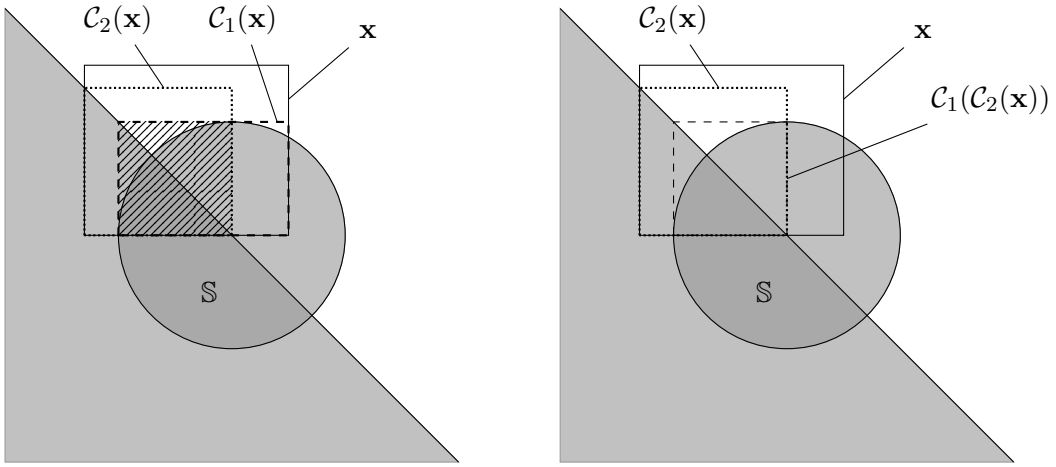
Property 2.2.1. *[30]*
Let $\mathcal{C}_1$ be a contractor for $\mathbb{S}_1$ and $\mathcal{C}_2$ be a contractor for $\mathbb{S}_2$. Then,

- $(\mathcal{C}_1 \cap \mathcal{C}_2)$ *is a contractor for* $\mathbb{S}_1 \cap \mathbb{S}_2$.
- $(\mathcal{C}_1 \cup \mathcal{C}_2)$ *is a contractor for* $\mathbb{S}_1 \cup \mathbb{S}_2$.
- $(\mathcal{C}_1 \circ \mathcal{C}_2)$ *is a contractor for* $\mathbb{S}_1 \cap \mathbb{S}_2$.

As an illustration, consider the CSP composed of two constraints:

$$\begin{cases} x_1^2 + x_2^2 - 1 \leq 0 \\ x_1 + x_2 \leq 0 \end{cases}$$

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two contractors for the first and the second constraint respectively. The feasible set $\mathbb{S}$ is represented by the dark gray set. In Figure 2.6a, the hatched set corresponds to the contraction of $\mathbf{x}$ by the contractor $(\mathcal{C}_1 \cap \mathcal{C}_2)$. The same box is obtained in Figure 2.6b using the contractor defined as $(\mathcal{C}_1 \circ \mathcal{C}_2)$.



(a) Intersection of $\mathcal{C}_1$ and $\mathcal{C}_2$.

(b) Composition of $\mathcal{C}_1$ and $\mathcal{C}_2$.

Figure 2.6: Combination of two contractors.

By extension, a unique contractor can be defined for a CSP composed of several (more than two) constraints by combining iteratively the primitive forward backward contractors.

2.2.4 Proving feasibility and non-feasibility with contractors

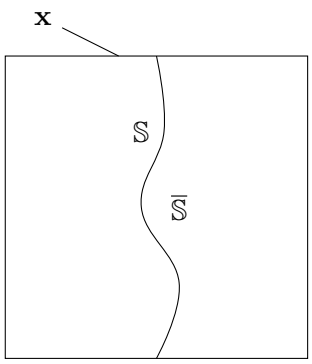
Consider a CSP $\mathcal{H}$ composed of p constraints $\{C_1, \ldots, C_p\}$. A unique contractor $\mathcal{C}$ for the feasible set $\mathbb{S}$ of $\mathcal{H}$ can be defined. When $\mathcal{C}$ contracts a box, the subset of $\mathbf{x}$ discarded is proved to be non-feasible due to the consistency property of the contractor. Considering it the other way around, if $\mathcal{C}_{inv}$ is a contractor for $\overline{\mathbb{S}}$ the complementary of $\mathbb{S}$, then the subset discarded by $\mathcal{C}_{inv}$ is proved to be feasible. If the whole box is discarded, then it is proved to be entirely feasible. Property 2.2.1 recap the above mathematically, and is illustrated on Figure 2.7.

Proposition 2.2.1.

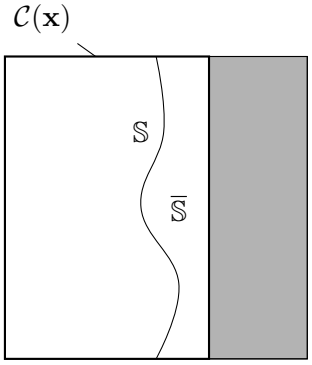
$\mathbf{x} \setminus \mathcal{C}(\mathbf{x}) \subseteq \overline{\mathbb{S}}$,
$\mathbf{x} \setminus \mathcal{C}_{inv}(\mathbf{x}) \subseteq \mathbb{S}$,
$\mathcal{C}(\mathbf{x}) = \emptyset \implies \mathbf{x} \subseteq \overline{\mathbb{S}}$,
$\mathcal{C}_{inv}(\mathbf{x}) = \emptyset \implies \mathbf{x} \subseteq \mathbb{S}$.

Since $\mathbb{S} = \underset{i}{\cap}\mathbb{S}_i$ with $\mathbb{S}_i$ the feasible set related to the constraint C_i, its complementary is $\overline{\mathbb{S}} = \underset{i}{\cup}\overline{\mathbb{S}_i}$ with $\overline{\mathbb{S}_i}$ the complementary of $\mathbb{S}_i$. Consequently, $\mathcal{C}_{inv}$ can be constructed as the union of the forward backward contractors for the $\overline{\mathbb{S}_i}$.

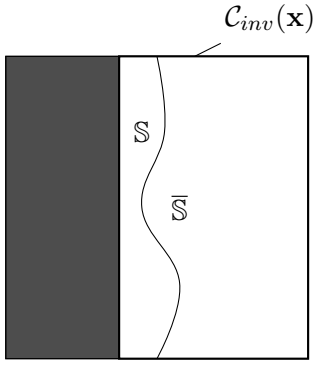
Remark. Since $\mathcal{C}(\mathbf{x})$ is a box, $\mathbf{x} \setminus \mathcal{C}(\mathbf{x})$ can be represented as a union of boxes (see Section 2.1.1).



(a) Initial box $\mathbf{x}$.



(b) Contraction done by $\mathcal{C}$, the set proved to be infeasible is in gray.



(c) Contraction done by $\mathcal{C}_{inv}$, the dark gray set is proved to be feasible.

Figure 2.7: Contractions performed by $\mathcal{C}$ and $\mathcal{C}_{inv}$.

2.3 Branch and Bound algorithms based on Intervals

When searching for the global optimum of a non convex problem, an exhaustive search must be performed. To do so, Branch and Bound (B&B) algorithms can be used. These algorithms

were first developed to solve discrete problems [80], to avoid the exhaustive enumeration of candidate solutions. The principle relies on splitting, or *branching*, repetitively the search domain of solutions into smaller disjoint subsets. Bounds on the objective over these subsets are computed, and enable to state whether the global solution is possibly contained in a subset or not. In the last case, the subset is discarded, reducing the domain over which the global solution is searched. One of the advantages of such an algorithm is that an enclosure of the global optimum is provided, permitting to determine how far is a solution from the global optimum.

2.3.1 Notations

The optimization problems addressed in this thesis are expressed in the general way as

$$\begin{cases} \min\limits_{x\in\mathbb{X}} & f(x), \\ \text{subject to} & g_i(x) \leq 0, \quad \forall i \in \{1,\ldots,m\}. \end{cases}$$

The *objective function* is denoted f, the constraints are restricted to inequalities (see Section 2.2 for equality constraints). We assume that the functions f and g_i have explicit expressions, and the initial domain $\mathbb{X}$ is a box. The feasible set defined by the constraints is denoted $\mathbb{S}$. The *global optimizer* is denoted x^*, and is defined as the feasible point where f is minimal.

$$\begin{cases} x^* \in \mathbb{S}, \\ \forall x \in \mathbb{S}, \quad f^* = f(x^*) \leq f(x). \end{cases}$$

The *global minimum* is denoted f^*.

Remark. x^* is also called the minimum argument, and is defined as

$$x^* \triangleq \underset{x\in\mathbb{S}}{\operatorname{argmin}} f(x).$$

Note that the global optimizer of an optimization problem is not necessarily unique. Figure 2.8 illustrates an optimization problem and the notations adopted. $\mathbb{S}$ is displayed in bold on the x axis.

Remark. Considering a minimization problem, the optimizer is called the *minimizer* and the optimum value the *mimimum*. Considering a maximization problem, the optimizer is called the *maximizer* and the optimum value the *maximum* or the *supremum*.

An important property that will be needed in Chapter 3 is that the minimum of a function over a set $\mathbb{X}_1$ is greater or equal to its minimum over a superset $\mathbb{X}_2$ of $\mathbb{X}_1$.

Proposition 2.3.1.

- $\mathbb{X}_1 \subseteq \mathbb{X}_2 \implies \min\limits_{x\in\mathbb{X}_1} f(x) \geq \min\limits_{x\in\mathbb{X}_2} f(x).$
- $\mathbb{X}_1 \subseteq \mathbb{X}_2 \implies \max\limits_{x\in\mathbb{X}_1} f(x) \leq \max\limits_{x\in\mathbb{X}_2} f(x).$

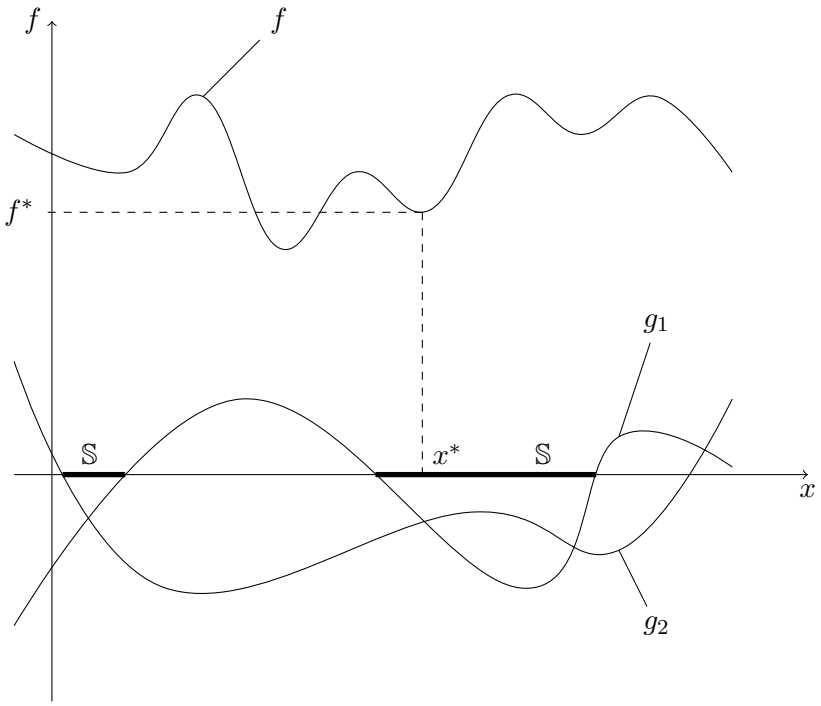


Figure 2.8: Minimization of f subject to the constraints $g_1(x) \leq 0$ and $g_2(x) \leq 0$.

2.3.2 Interval Branch and Bound Algorithm

The B&B algorithms based on interval emerged during the 70's [122], but were limited to simple optimization problems due to the low computing power available at that time. Since then, several works have been proposed to improve the convergence of interval B&B algorithms [61, 73, 67, 75]. This section introduces the Interval Branch and Bound Algorithm (IBBA), using the concept of inclusion functions and contractors. The IBBA structure given by Algorithm 1, as well as the acceleration techniques, are presented in [108, 99].

IBBA is designed to solve constrained optimization problems globally as defined in the previous section. It takes as input an optimization problem (objective function, constraints and initial domain), and returns a feasible solution $\tilde{x}$, a set of boxes containing the global optimum x^* and an enclosure $[\underline{f}, \overline{f}]$ of the minimum f^*. IBBA can be decomposed in several steps: extraction, contraction, bounds computation, solution search, bisection and insertion. The algorithm is first described, and further details about its implementation are provided in the dedicated paragraphs below.

IBBA is based on two data structures $\mathcal{L}$ and $\mathcal{L}_s$ in which boxes are stored. Initially, $\mathcal{L}$ contains the initial domain $\mathbb{X}$. At each iteration of the algorithm, a box is extracted from $\mathcal{L}$ and contracted to discard subsets proved not to contain the global optimum x^*. Then, bounds on the objective function are computed over the contracted box, and a feasible solution is searched over x. This solution becomes the new best solution $\tilde{x}$ if it provides a lower value of f than the current solution. Finally, the box is bisected into several non overlapping boxes that are inserted either in $\mathcal{L}$ or in $\mathcal{L}_s$ whether their width is smaller than a minimal bisection criterion ϵ. $\mathcal{L}_s$ is used as a memory structure. At any time, the union of boxes contained

in $\mathcal{L}$ and $\mathcal{L}_s$ corresponds to the initial domain $\mathbb{X}$ deprived from the subsets discarded in the contraction step. As a consequence, x^* is contained in this set and a lower bound of f^* is given by

$$\underline{f} = \min_{\mathbf{x}_i \in \mathcal{L} \cup \mathcal{L}_s} \underline{\mathbf{f}(\mathbf{x}_i)}.$$

An upper bound is provided by the value of f at a feasible solution $\tilde{x}$,

$$\overline{f} = \tilde{f}.$$

IBBA terminates either when a stopping criterion is reached, or when $\mathcal{L}$ is empty. The classical stopping criterion is the reach of a given precision on the width of the enclosure of f^*. The criterion chosen in this thesis is the relative precision ϵ_r, that is IBBA terminates if

$$\overline{f} - \underline{f} \leq \epsilon_r \max(1, |\overline{f}|).$$

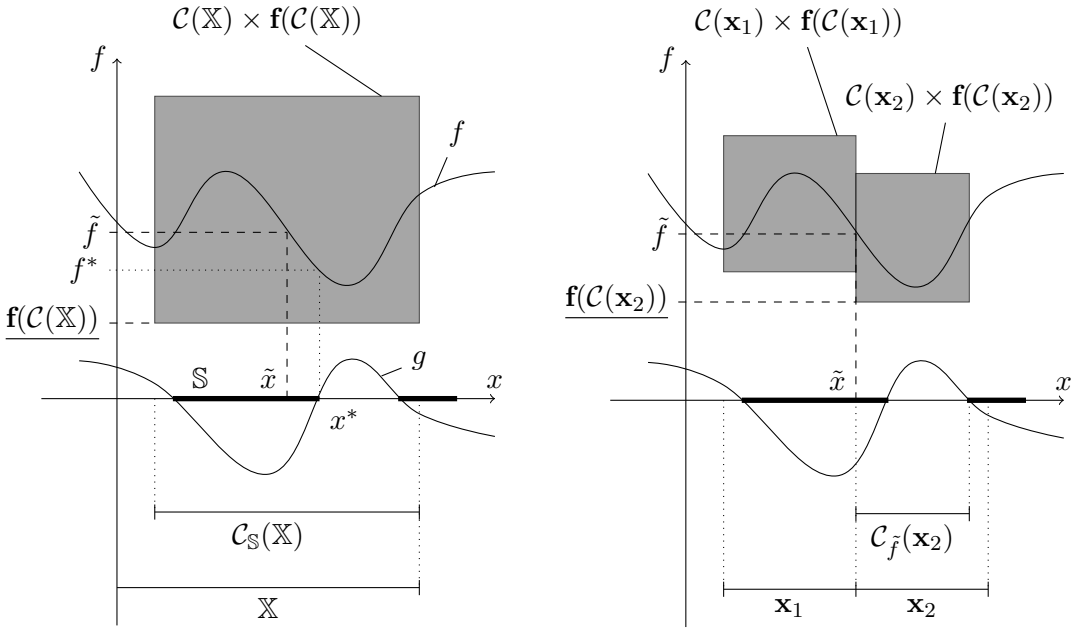
If both $\mathcal{L}$ and $\mathcal{L}_s$ are empty when IBBA terminates, it means that the problem is infeasible over the initial domain.

Algorithm 1 Interval Branch and Bound algorithm: IBBA.

```
Set L = {X}, L_s = ∅.
while Stop criterion not reached do
    Extract a box x from L.                                  ▷ extraction
    Contract x with acceleration techniques.                 ▷ contraction
    if x ≠ ∅ then
        Compute f(x).                                        ▷ bounds computation
        Search for a feasible vector in x,
         and update the best current solution x̃.             ▷ solution search
        Bisect x into p boxes x_i.                           ▷ bisection
        for i ∈ {1, ..., p}. do
            if w(x_i) > ε then                               ▷ insertion
                Insert x_i in L.
            else
                Insert x_i in L_s
            end if
        end for
    end if
end while
```

Extraction/Insertion. When boxes are inserted in $\mathcal{L}$ they are sorted with respect to a priority criterion, deciding which box will be extracted first. This criterion can be the maximal width of the box, the width of $\mathbf{f}(\mathbf{x})$, etc. A good order criterion is the lower bound of $\mathbf{f}(\mathbf{x})$. That is, the box $\mathbf{x}$ extracted from $\mathcal{L}$ is such that

$$\underline{\mathbf{f}(\mathbf{x})} = \min_{\mathbf{x}_i \in \mathcal{L}} \underline{\mathbf{f}(\mathbf{x}_i)}.$$



(a) First iteration. The box $\mathbb{X}$ is contracted by $\mathcal{C}_{\mathbb{S}}$. $\underline{\mathbf{f}(\mathcal{C}(\mathbb{X}))}$ provides a lower bound on f^*, the midpoint of $\mathcal{C}(\mathbb{X})$ is feasible and provides an upper bound $\tilde{f}$ on f^*. $\mathcal{C}(\mathbb{X})$ is bisected into $\mathbf{x}_1$ and $\mathbf{x}_2$.

(b) Second and third iterations. $\mathbf{x}_1$ is not contracted by any contractor, $\mathcal{C}(\mathbf{x}_1) = \mathbf{x}_1$. $\mathbf{x}_2$ is contracted by $\mathcal{C}_{\tilde{f}}$. The midpoint of $\mathbf{x}_1$ is not better than $\tilde{x}$, and the midpoint of $\mathcal{C}(\mathbf{x}_2)$ is not feasible. Therefore $\tilde{x}$ and $\tilde{f}$ are not updated. The lower bound on f^* becomes $\underline{\mathbf{f}(\mathcal{C}(\mathbf{x}_2)}$.

Figure 2.9: First steps of Interval Branch and Bound Algorithm.

Such a criterion allows to focus on increasing the lower bound $\underline{f}$, and also enable to know at any time the value of the lower bound on f^*. This criterion is the one that will be used in the sequel.
The data structure $\mathcal{L}$ is generally a list. However, a sorting algorithm must be used when inserting a box in $\mathcal{L}$, and the complexity of such algorithms is at best linear with respect to the number of elements in the list. We propose to use a binary heap structure instead, since the complexity of the insertion is $\log_2(n)$. $\mathcal{L}_s$ is also a binary heap with the same sorting criterion. Doing so the first element of $\mathcal{L}_s$ provides the lowest lower bounds on f over the boxes it contains. The lower bound $\underline{f}$ is given by the minimum between $\underline{\mathbf{f}(\mathbf{x}_1)}$ and $\underline{\mathbf{f}(\mathbf{x}_{s,1})}$, where $\mathbf{x}_1$ is the first box in $\mathcal{L}$ and $\mathbf{x}_{s,1}$ the first box in $\mathcal{L}_s$.

Contraction. The contraction step aims to contract the box $\mathbf{x}$ so that subsets of $\mathbf{x}$ that do not contain the global minimum are removed. The definition of the minimum provides two constraints that a vector x must be satisfied to be the minimizer.

- The first one is that x must be feasible. For this constraint, a contractor $\mathcal{C}_{\mathbb{S}}$ for $\mathbb{S}$ can be built as explained in Subsection 2.2.2
- The second constraint stands that f is minimal at x over the feasible set $\mathbb{S}$. That is, it is

possible to define the additional fictive constraint $f(x) < \tilde{f} = f(\tilde{x})$, with $\tilde{x}$ any feasible vector. In the IBBA, $\tilde{x}$ is computed at the step **solution search**. If no $\tilde{x}$ is available, $\tilde{f} = +\infty$. For this constraint, another contractor $\mathcal{C}_{\tilde{f}}$ is defined. If $\mathcal{C}_{\tilde{f}}(\mathbf{x}) = \emptyset$, it implies that $\forall x \in \mathbf{x}$, feasible or not, $f(x) > \tilde{f}$ and therefore $f(x) > f^*$, implying that $x^* \notin \mathbf{x}$.

- A last contractor $\mathcal{C}_J$ based on the monotonicity test of f, described in Subsection 2.3.3, is also used to contract $\mathbf{x}$.

A unique contractor $\mathcal{C}$ being the composition of the three contractors $\mathcal{C}_{\mathbb{S}}$, $\mathcal{C}_{\tilde{f}}$, and $\mathcal{C}_J$ is defined.

$$\mathcal{C} = \mathcal{C}_{\mathbb{S}} \circ \mathcal{C}_{\tilde{f}} \circ \mathcal{C}_J.$$

If $\mathcal{C}$ contracts $\mathbf{x}$ into $\emptyset$, $\mathbf{x}$ is discarded since it is proved that it does not contains x^*. These three contractors compose the acceleration techniques.

Bounds computation. Once $\mathbf{x}$ is contracted, bounds on $f(\mathbf{x})$ are computed. This is done using the natural inclusion function $\mathbf{f}$ of f. This step is of major interest since it enables to compute the lower bound $\underline{f}$ on f^*.

Solution search. In this step, a feasible vector x is searched such that it is better, with respect to the value of f, than the current best feasible vector $\tilde{x}$. The most straightforward way is to chose $x = mid(\mathbf{x})$. If x is feasible and $f(x) < \tilde{f}$, then x is assigned to $\tilde{x}$. This step enables both to lower $\tilde{f}$ which is an upper bound of f^*, and to get better contraction from $\mathcal{C}_{\tilde{f}}$. In the following, the candidate solution is chosen as the midpoint of $\mathbf{x}$. When a new solution is found, the elements of $\mathcal{L}$ and $\mathcal{L}_s$ having a lower bound strictly greater than $\tilde{f}$ are discarded since they do not contain x^*.

Bisection. The bisection of $\mathbf{x}$ consists in splitting it into several non overlapping boxes $\mathbf{x}_i$, $i \in \{1, \ldots, p\}$.

$$\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_p.$$

In this way, any vector of $\mathbf{x}$ is lost in the bisection process. One of the most economical way to perform a bisection in term of computational time is to bisect $\mathbf{x}$ along its largest dimension, providing two sub-boxes. This is the strategy chosen in this thesis. However, other bisection methods exist, relying on heuristics that enable to chose along which dimension $\mathbf{x}$ should be bisected based on the local characteristics of the objective function and the constraints [55, 12, 35]. These methods aim for a bisection so that the contractors to discard quickly the resulting sub-boxes, but may be computationally expensive.

2.3.3 Monotonicity test

The monotonicity test permits to contract a box by studying the monotonicity of the objective function. The principle lies in contracting each dimensions of a box $\mathbf{x}$ to the coordinate of the minimizer of f over this box. The coordinates are deduced from the monotonicity of f.

Consider the differentiable objective function $f : \mathbb{R}^n \to \mathbb{R}$ and a feasible box $\mathbf{x} = \mathbf{x}_1 \times \cdots \times \mathbf{x}_n$ in $\mathbb{IR}^n$. The Jacobian of f is

$$J \triangleq (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})^T = (J_1, \ldots, J_n)^T.$$

Suppose that an inclusion function $\mathbf{J}$ of J is available, and $\underline{\mathbf{J}_i(\mathbf{x})} \geq 0$. We have that the projection of f on the i^{th} axis is monotonous increasing,

$$\underline{\mathbf{J}_i(\mathbf{x})} \geq 0 \implies \forall x \in \mathbf{x},\ f(x) \geq f(x_1, \ldots, \underline{\mathbf{x}_i}, \ldots, x_n).$$

As a consequence, the i^{th} coordinate of the minimizer of f over $\mathbf{x}$ is $\underline{\mathbf{x}_i}$ and $\mathbf{x}_i$ can be contracted to its lower bound $\underline{\mathbf{x}_i}$. Following the same reasoning, if $\overline{\mathbf{J}_i(\mathbf{x})} \leq 0$, then $\mathbf{x}_i$ is contracted to $\overline{\mathbf{x}_i}$. Such contraction can be performed only if $\mathbf{x}$ is feasible. Indeed, $\mathbf{x}$ is contracted to the coordinate of the minimizer of f over $\mathbf{x}$ but not to the minimizer of f over the feasible subset of $\mathbf{x}$, being $\mathbf{x} \cap \mathbb{S}$. For example, consider the unconstrained minimization problem

$$\left\{ \min_{x_1,x_2} f(x_1, x_2) = x_1 \cdot x_2^2. \right.$$

over the box $[-1, 1] \times [-1, 1]$. The Jacobian of f is $J(x_1, x_2) = (x_2^2, 2x_1x_2)^T$, for which the natural inclusion function provides $\mathbf{J}([-1, 1], [-1, 1]) = ([0, 1], [-2, 2])^T$. Hence, f is monotonic over the x_1 axis as illustrated in Figure 2.10a by the projection of f on the (x_1, f) plan for several values of x_2 . As a consequence the initial box can be contracted to $[-1, -1] \times [-1, 1]$. However the function is not monotonous over x_2 and the box cannot be contracted over its second dimension.
Consider now an additional constraint $x_1 \geq -0.5$ as represented on Figure 2.10b. The problem becomes

$$\left\{ \begin{array}{ll} \min\limits_{(x_1,x_2)} & f(x_1, x_2) = x_1 \cdot x_2^2, \\ \text{s. t.} & x_1 \geq -0.5. \end{array} \right.$$

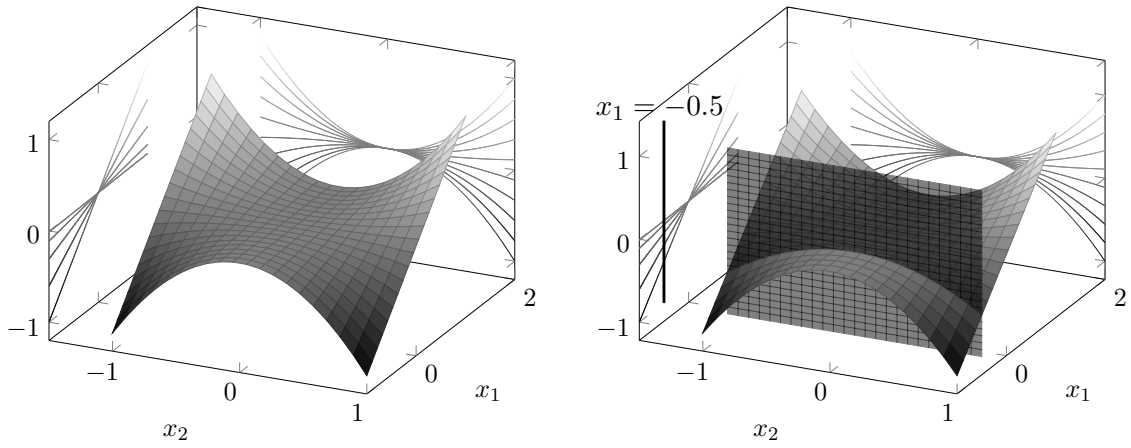
The first coordinate of the minimizer is now -0.5. The contraction cannot be performed because the contracted box $[-1, -1] \times [-1, 1]$ does not contain the minimizer.

Algorithm 2 framed the monotonicity test as the contractor $\mathcal{C}_J$.

Algorithm 2 Monotonicity contractor $\mathcal{C}_J$.

Input: $\mathbf{x}$.
Output: Contracted $\mathbf{x}$.

```
1: if x is feasible. then
2:     Compute J(x).
3:     for i ∈ {1, ..., n} do
4:         if J_i(x) ≥ 0 then
5:             x_i = x_i.
6:         else if J_i(x) ≤ 0 then
7:             x_i = x_i.
8:         end if
9:     end for
10: end if
```



(a) The box is feasible, the initial box can be contracted to -1 over the x-axis.

(b) The box is not entirely feasible, the monotonicity test cannot be used.

Figure 2.10: Contraction based on monotonicity test.

At the first line of Algorithm 2, the feasibility of $\mathbf{x}$ is verified either using the inclusion function of the constraints functions or using a contractor on the complementary of the feasible set (see Section 2.2 and 2.2.2, respectively). If $\mathbf{x}$ is feasible, the monotonicity test is performed over the n dimensions of $\mathbf{x}$.
$\mathcal{C}_J$ is a contractor for the minimizer of f over $\mathbf{x}$ since it respects the consistency and contraction properties. By extension, $\mathcal{C}_J$ is also a contractor for the global minimizer of f over the initial domain $\mathbb{X}$. Indeed, if x^* is contained in $\mathbf{x}$ then it is also the minimizer of f over $\mathbf{x} \cap \mathbb{S}$.

2.3.4 Feasible set characterization

When solving an optimization problem, it may be interesting to compute not only a good feasible solution, but also a sub-optimal region where the objective function is low enough to be satisfying. To this extend, the optimization problem is reformulated by the CSP $\mathcal{S}_\alpha$ by transforming the objective function into a constraint.

$$\mathcal{S}_\alpha \begin{cases} f(x) \leq \alpha \\ g_i(x) \leq 0, \quad \forall i \in \{1, \ldots, m\} \\ x \in \mathbb{X}. \end{cases}$$

The feasible set $\mathbb{S}_\alpha$ of $\mathcal{S}_\alpha$ corresponds to a sub-optimal region, and can be characterized by a *sub-paving* of $\mathbb{X}$ by adapting the IBBA. A sub-paving of $\mathbb{X}$ is a union of non-overlapping sub-boxes of $\mathbb{X}$ [69]. This sub-paving is computed by the Feasible Set Characterization Algorithm (FSCA), given by Algorithm 3.

Remark. A sub-optimal region can also be characterized by the *near optimal* set, composed of the feasible points whose objective values are near the optimum. However, the optimum, or at least an approximation, must be known to define such a set. Characterizing this set is

thus more difficult than characterizing the feasible set of $\mathcal{S}_\alpha$. In this thesis, only sub-optimal regions as defined by $\mathcal{S}_\alpha$ are considered.

FSCA takes as input the CSP $\mathcal{S}_\alpha$ and a minimal width ϵ. It returns a sub-paving of $\mathbb{X}$ contained in three lists:

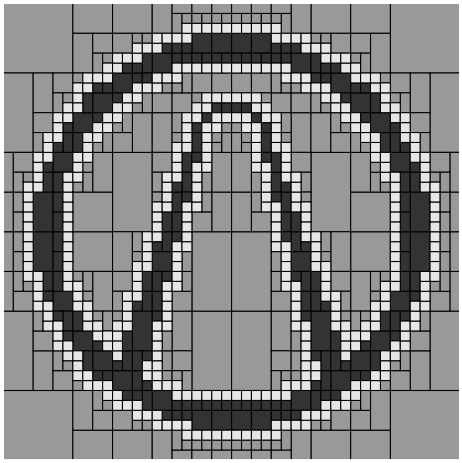
- $\mathcal{L}_{in}$ contains feasible boxes.
- $\mathcal{L}_{out}$ contains infeasible boxes.
- $\mathcal{L}_{mb}$ contains boxes not proved to be feasible or infeasible.

Two contractors $\mathcal{C}_\alpha$ and $\mathcal{C}_{inv}$ are defined for $\mathbb{S}_\alpha$ and its complementary $\overline{\mathbb{S}_\alpha}$ respectively. FSCA works as follows. A box $\mathbf{x}$ is extracted from $\mathcal{L}$ and contracted with $\mathcal{C}_\alpha$ and $\mathcal{C}_{inv}$. The subset discarded by the first contraction is proved to be infeasible and is stored in $\mathcal{L}_{out}$. The second contraction discards a subset proved to be feasible, which is stored in $\mathcal{L}_{in}$. The box resulting from these two contractions is not proved to be feasible or infeasible. It is hence bisected and added to $\mathcal{L}$ if its width is greater than ϵ. On the contrary, the box is stored in $\mathcal{L}_{mb}$. FSCA terminates when $\mathcal{L}$ is empty. The initial domain $\mathbb{X}$ is then completely characterized by the boxes contained in the lists $\mathcal{L}_{in}$, $\mathcal{L}_{out}$, and $\mathcal{L}_{mb}$. The bisection step of FSCA is the same as the one of IBBA. However, the boxes contained in $\mathcal{L}$ does not need to be sorted, simplifying the extraction and insertion steps.

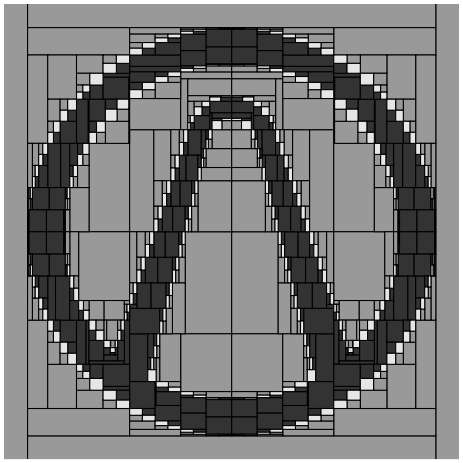
Algorithm 3 Feasible Set Characterization Algorithm: FSCA.

```
1:  Set L = {X}, L_in = ∅, L_out = ∅, L_mb = ∅.
2:  while L is not empty. do
3:      Extract a box x from L.                          ▷ extraction
4:      Insert x \ C_α(x) in L_out                       ▷ Feasible contraction
5:      Insert x \ C_inv(x) in L_in                      ▷ Non feasible contraction
6:      Bisect x_B into p boxes x_i.                     ▷ bisection
7:      for i ∈ {1, ..., p}. do
8:          if w(x_i) > ε then                           ▷ Insertion
9:              Insert x_i in L.
10:         else
11:             Insert x_i in L_mb
12:         end if
13:     end for
14: end while
```

Figure 2.11 illustrates a sub-paving obtained with FSCA. The first sub-paving in Figure 2.11a is obtained using only the feasibility test based on the inclusion function. The second sub-paving in Figure 2.11b is computed using the contractors $\mathcal{C}_\alpha$ and $\mathcal{C}_{inv}$. Obtaining he second sub-paving requires three time less iterations to terminate than the first one, and illustrates the effectiveness of the contractors.



(a) Subpaving obtained with FSCA without the contraction. 1811 iterations are needed for FSCA to terminate.



(b) Subpaving obtained with FSCA with the contraction step. 551 iterations are needed for FSCA to terminate.

Figure 2.11: Subpaving obtained with FSCA with the contraction step.

2.4 Conclusion

This chapter has introduced interval analysis tools, and how they can be embedded in B&B algorithms either to solve an optimization problem or to characterize the feasible set of a CSP in a global way. The main point of the IBBA and FSCA algorithms is the reliability of the results they provide, ensured by the guaranteed computations done with interval arithmetic. The efficiency of those two algorithms rely on the acceleration techniques, framed as contractors, permitting to discard subsets of the search domain and hence improve the convergence. The optimization problems considered in this chapter are minimization problems subject to constraints. The next chapter is dedicated to the global resolution of continuous minmax problems subject to semi infinite constraints, which is more difficult to handle. However, the framework provided by the B&B and the contractors can be extended to minmax problems to solve them globally. As a consequence, the next chapter strongly relies on the tools presented in this chapter.

Chapter 3

Minmax optimization and semi infinite programming

3.1 Introduction

This chapter is dedicated to the global resolution of minmax problem subject to semi infinite constraints. The IBBA introduced in Chapter 2 is extended to solve this class of problems. It will be shown in Chapter 4 that the structured synthesis, and the structured robust synthesis problems can be formulated as minmax problems. As a consequence, this chapter introduced the necessary tools to solve these two control problems in a global way.

3.1.1 MinMax problems

The minmax problem initially emerges from game theory and is stated as follows

$$\left\{ \begin{array}{ll} \min\limits_{x\in\mathbb{X}} & \sup\limits_{y\in\mathbb{Y}} f(x,y). \end{array} \right. \tag{3.1}$$

Consider two players P1 and P2. The payoff of P1 is given by $-f$ and the one of P2 by f. P2 choses y to maximize its payoff while P1 aims to minimize the best payoff P2 could obtain. The game is said to have an equilibrium point, called the Nash equilibrium from the mathematician John Nash, if

$$\min_x \sup_y f(x,y) = \sup_y \min_x f(x,y).$$

This equilibrium point may not exists and only the following inequality holds in the general case [14]:

$$\min_x \sup_y f(x,y) \leq \sup_y \min_x f(x,y).$$

In addition, constraints on the problem can be considered representing rules that the players must respect. The general minmax problem is stated as $\mathcal{M}_{mm}$.

$$\mathcal{M}_{mm} : \left\{ \begin{array}{ll} \min\limits_{x\in\mathbb{X}} & \sup\limits_{y\in\mathbb{Y}} f(x,y), \\ \text{s. t.} & g_i(x,y) \leq 0,\ \forall i \in \{1,\dots,m\}. \end{array} \right.$$

In the following, we denote $\mathbb{S} \subseteq \mathbb{X} \times \mathbb{Y}$ the feasible set defined by the constraints g_i, and $\mathbb{Y}_x = \{y \in \mathbb{Y} \,|\, (x, y) \in \mathbb{S}\}$ the feasible subset of $\mathbb{Y}$ given x.

In game theory, the minmax problem has been studied mostly in the unconstrained convex-concave case, i.e. f is convex with respect to x and concave with respect to y. Many papers are dedicated to the resolution of the unconstrained minmax problem using evolutionary approaches such as genetic algorithms [16, 17, 34] or particle swarm optimization [120, 79], but few of them consider the constrained case [83, 119]. Those methods do not allow to solve the minmax problem in a global way since evolutionary algorithms converge locally but not globally. In this thesis, the minmax problem is aimed to be solved in a global way. However, very few attention has been given to solve this problem globally. In [136], Zuhe proposes a branch and bound algorithm based on interval analysis for the unconstrained case, and Sainz approaches the constrained minmax problem with modal intervals [117].

3.1.2 Semi infinite programs

A Semi Infinite Program (SIP) is an optimization problem involving Semi Infinite Constraints (SIC) [64], generally expressed as

$$SIP : \begin{cases} \min\limits_{x \in \mathbb{X}} & f(x), \\ \text{s. t.} & q(x, z) \leq 0, \, \forall z \in \mathbb{F}. \end{cases}$$

This constraint is called semi infinite because it involves a finite number of variable but an infinite number of constraint.

Remark. $\mathbb{F}$ can be any subset of $\mathbb{R}^{n_z}$, possibly characterized by constraints. In the litterature, the distinction is made whether $\mathbb{F}$ is a hyperrectangle or is defined by constraints. In the second case, the problem is called a Generalized SIP (GSIP). In this section, we consider GSIP, but simply called them SIP.

A SIP problem is closely related to minmax problems as pointed out in [22]. Indeed, a SIC can be dealt with by considering a related maximization problem due to the following equivalence,

$$\forall z \in \mathbb{F}, q(x, z) \leq 0 \iff \sup_{z \in \mathbb{F}} q(x, z) \leq 0.$$

The main difficulty with SIP is that q is not convex in the general case, meaning that the maximization problem related to the SIC must be solved in a global way to ensure the satisfaction of the SIC. To this extend, [20] and [44] propose a convex reformulation of the SIC, and [101] takes advantages of the reliability of interval analysis tools. In [22], Blankenship proposes an algorithm which approximates the infinite set of constraint by a finite one, by replacing $\mathbb{F}$ by the solutions to the maximization problem for different values of x. In [89, 38], a guaranteed version of the algorithm relying on the global resolution of the maximization problem is proposed.

Quantified constraints. More generally, a SIC is a quantified constraints involving the quantifier $\forall$. The first approach of quantified constraints relies on the Cylindrical Algebraic Decomposition (CAD), introduced in [33]. This method enables to reformulate quantified constraints as quantifier-free constraints, and is implemented in several solvers. As pointed out in [107] CAD methods are generally slow to compute solutions, due to the doubly exponential behavior of the underlying algorithm. In addition the resulting constraints, although being quantifier-free, may not be used straightforwardly and need further processing. On the other hand, branch and bound algorithms based on interval methods emerged during the 1990s to solve problems subject to quantified constraints [70, 85]. During the 2000s, the branch and bound methods gained in efficiency by integrating propagation methods [107, 51, 50].

In this thesis, we consider the case where $\mathbb{F}$ is defined by a set of constraints depending both in x and z. That is, $\mathbb{F} = \{z \in \mathbb{Z} \,|\, h_i(x,z) \leq 0,\ \forall i\}$. For the sake of clarity, the notation $\mathbb{Z}_x$ instead of $\mathbb{F}$ is used in the following. Hence, a SIC is formulated as

$$q(x,z) \leq 0,\ \forall z \in \mathbb{Z}_x.$$

The feasible subset of $\mathbb{X}$ defined by the SIC is denoted $\mathbb{Q}$.

3.1.3 Problem of interest

Chapter 4 addresses the formulation of H_∞ control problems as optimization ones. Those problems have various formulations, involving SIC or not, having objectives expressed as maxima or not, etc. In order to solve them in a global way with the same method, independently of their objectives/constraints, we consider the very general case given by problem $\mathcal{M}$. For the sake of clarity, only one SIC is considered, Section 3.4 explains how multiple SICs can be taken into account.

$$\mathcal{M}: \begin{cases} \min\limits_{x \in \mathbb{X}} & \sup\limits_{y \in \mathbb{Y}_x} f(x,y), \\ \text{s. t.} & p_j(x) \leq 0,\ \forall j, \\ & q(x,z) \leq 0,\ \forall z \in \mathbb{Z}_x. \end{cases}$$

We assume that:

- The initial domains $\mathbb{X}$, $\mathbb{Y}$ and $\mathbb{Z}$ for the variable x, y, and z are subsets of $\mathbb{IR}^{n_x}$, $\mathbb{IR}^{n_y}$, and $\mathbb{IR}^{n_z}$, respectively,
- all the functions have an explicit expression.

We also recall that the sets $\mathbb{Y}_x$ and $\mathbb{Z}_x$ are defined by constraints.

- $\mathbb{Y}_x = \{y \in \mathbb{Y} \,|\, g_i(x,z) \leq 0,\ \forall i\}$
- $\mathbb{Z}_x = \{z \in \mathbb{Z} \,|\, h_j(x,z) \leq 0,\ \forall j\}$

Problem $\mathcal{M}$ is hence a minmax problem subject to regular constraints p_j and a SIC.

3.2 General approach

The approach we propose to solve $\mathcal{M}$ relies on branch and bound algorithms and interval tools presented in Chapter 2. This choice is motivated by the reliability of interval tools, and the fact that such approaches have already proved their efficiency for dealing with the minmax problem [136, 117], quantified constraints [51, 50], and SIP [101].
First of all, taking advantage of the reformulation of a SIC as a maximization problem, problem $\mathcal{M}$ can be expressed equivalently as

$$\mathcal{M}: \begin{cases} \min\limits_{x\in\mathbb{X}} & \sup\limits_{y\in\mathbb{Y}_x} f(x,y), \\ \text{s. t.} & p_j(x) \leq 0,\ \forall j, \\ & \sup\limits_{z\in\mathbb{Z}_x} q(x,z) \leq 0. \end{cases}$$

The last Problem $\mathcal{M}$ is therefore composed of two maximization problems sharing the same formulation.
Consider the maximization problem of f. We denote

$$\begin{array}{rl} f_x: & \mathbb{R}^{n_y} \mapsto \mathbb{R}, \\ & y \to f(x,y). \end{array}$$

The maximization of f at x is given by problem $\mathcal{M}_x$,

$$\mathcal{M}_x: \begin{cases} \sup\limits_{y\in\mathbb{Y}_x} & f_x(y). \end{cases}$$

The maximizer of $\mathcal{M}_x$, depending on x, is denoted y_x^* and the maximum is denoted f_x^*. We defined the function f_{sup} as

$$\begin{array}{rl} f_{sup}: & \mathbb{R}^{n_x} \to \mathbb{R}, \\ & x \mapsto f_x^*. \end{array}$$

Given a box $\mathbf{x}$, $f_{sup}(\mathbf{x})$ is the set of maxima of the problems $\mathcal{M}_x$,

$$f_{sup}(\mathbf{x}) = \{f_x^* \,|\, x \in \mathbf{x}\} \subset \mathbb{R}.$$

Using the same notations for the maximization problem related to the SIC, we define

$$\begin{array}{rl} q_x: & \mathbb{R}^{n_z} \mapsto \mathbb{R}, \\ & z \to q(x,z). \end{array}$$

and

$$\begin{array}{rl} q_{sup}: & \mathbb{R}^{n_x} \to \mathbb{R}, \\ & x \mapsto q_x^*. \end{array}$$

With these notations, problem $\mathcal{M}$ can be reformulated as,

$$\mathcal{M}: \begin{cases} \min\limits_{x\in\mathbb{X}} & f_{sup}(x), \\ \text{s. t.} & p_j(x) \leq 0,\ \forall j, \\ & q_{sup}(x) \leq 0. \end{cases}$$

The minimizer of $\mathcal{M}$ is denoted x^* and the minimum is $f^*_{sup} = f_{sup}(x^*)$. Under this form $\mathcal{M}$ is a "regular" minimization problem, and the IBBA structure can be used to solve it in a global way. However, the contraction and bound computation steps need to be modified in order to deal with the maximization problem $\mathcal{M}_x$. Indeed, bounds on f_{sup} and q_{sup} must be computable over a box $\mathbf{x}$, implicitly meaning to solve two maximization problems for sets of functions.

The rest of this chapter is organized as follows. Section 3.3 introduces a branch and bound algorithm to compute guaranteed bounds on f_{sup} and q_{sup}. Section 3.4 presents an algorithm to solve $\mathcal{M}$, and Section 3.5 illustrates the performance of this algorithm on numerical examples. Section 3.6 discusses the performance and novelty of our approach, identify possible improvements, and shed a light on other optimization problems to which our approach could be used.

3.3 Branch and bound algorithm for a set of functions

This section is dedicated to the computation of bounds on f_{sup} and q_{sup} over a box $\mathbf{x}$. The function f_{sup} is considered in this section, but all the results can be transposed to q_{sup}.
This section is organized as follows: Section 3.3.1 introduces theorems to compute guaranteed bounds on a set of maxima using inclusion functions. Section 3.3.2 proposes a way to prove the feasibility of a box $\mathbf{y} \subset \mathbb{Y}$ with respect to an infinite number of constraints. Sections 3.3.3 and 3.3.4 extend the contractors already introduced in Chapter 2 to the maximization of a set of functions. Finally, Section 3.3.5 introduces the Set Interval Branch and Bound (SIBBA), an algorithm to compute efficiently reliable bounds on f_{sup} and Section 3.3.6 proposes a numerical example.

3.3.1 Computing bounds of a set of maxima

Given a box $\mathbf{x}$, we aim to compute bounds on f_{sup} over $\mathbf{x}$. That is, a lower bound and an upper bound on the set $f_{sup}(\mathbf{x})$ are searched. The set of functions $f_{\mathbf{x}}$ is defined as,

$$f_{\mathbf{x}} = \{f_x \,|\, x \in \mathbf{x}\}.$$

The proposed approach to compute a reliable enclosure of f_{sup} relies on the hull of the set of function $f_{\mathbf{x}}$, which is composed of two functions $f^+_{\mathbf{x}}$ and $f^-_{\mathbf{x}}$ expressed as

$$\begin{aligned} f^+_{\mathbf{x}} :&\ \mathbb{R}^{n_y} \mapsto \mathbb{R}, \\ &\ y \to \sup_{x \in \mathbf{x}} f(x,y). \\ f^-_{\mathbf{x}} :&\ \mathbb{R}^{n_y} \mapsto \mathbb{R}, \\ &\ y \to \min_{x \in \mathbf{x}} f(x,y). \end{aligned}$$

For any $y \in \mathbb{Y}$, the functions $f^+_{\mathbf{x}}$ and $f^-_{\mathbf{x}}$ provides an enclosure of $f_{\mathbf{x}}$ as shown on Figure 3.1. Those functions are not known in the general case, and their expression will not be searched for, but they propose a framework to make the connection between the set of maxima $f_{sup}(\mathbf{x})$ and a practical way to compute bounds using inclusion functions.

Since the maximization problem $\mathcal{M}_x$ is considered for a set of functions, the feasible set $\mathbb{Y}_x$ must also be extended to the case where x is a set. The set being the union of all feasible sets related to the vectors in $\mathbf{x}$ is denoted $\mathbb{Y}_{\mathbf{x}}$,

$$\mathbb{Y}_{\mathbf{x}} = \underset{x\in\mathbf{x}}{\cup} \mathbb{Y}_x = \{y \in \mathbb{Y} \,|\, \exists x \in \mathbf{x},\, y \in \mathbb{Y}_x\}.$$

Hence, $\forall x \in \mathbf{x}$, $\mathbb{Y}_x \subseteq \mathbb{Y}_{\mathbf{x}}$. Theorem 3.3.1 proposes a first result to enclose $f_{sup}(\mathbf{x})$ using the hull of $f_{\mathbf{x}}$.

Theorem 3.3.1.

$$f_{sup}(\mathbf{x}) \subseteq [\min_{y\in\mathbb{Y}_{\mathbf{x}}} f_{\mathbf{x}}^-(y),\, \sup_{y\in\mathbb{Y}_{\mathbf{x}}} f_{\mathbf{x}}^+(y)].$$

Moreover, if $\forall x \in \mathbf{x}, \mathbb{Y}_x = \mathbb{Y}$*, then*

$$f_{sup}(\mathbf{x}) \subseteq [\sup_{y\in\mathbb{Y}} f_{\mathbf{x}}^-(y), \sup_{y\in\mathbb{Y}} f_{\mathbf{x}}^+(y)].$$

Proof. Consider $x \in \mathbf{x}$,

$$f_{sup}(x) = \sup_{y\in\mathbb{Y}_x} f_x(y) \leq \sup_{y\in\mathbb{Y}_{\mathbf{x}}} f_x(y), \tag{3.2}$$

since $\mathbb{Y}_x \subseteq \mathbb{Y}_{\mathbf{x}}$.
In addition, we have by definition

$$\begin{array}{l} \forall y \in \mathbb{Y}, f_x(y) \leq f_{\mathbf{x}}^+(y) \\ \implies \sup\limits_{y\in\mathbb{Y}_{\mathbf{x}}} f_x(y) = f_x(\operatorname{argmax}\limits_{y\in\mathbb{Y}_{\mathbf{x}}} f_x(y)) \leq f_{\mathbf{x}}^+(\operatorname{argmax}\limits_{y\in\mathbb{Y}_{\mathbf{x}}} f_x(y)) \\ \implies \sup\limits_{y\in\mathbb{Y}_{\mathbf{x}}} f_x(y) \leq \sup\limits_{y\in\mathbb{Y}_{\mathbf{x}}} f_{\mathbf{x}}^+(y). \end{array} \tag{3.3}$$

From Equations 3.2 and 3.3, it follows that

$$f_{sup}(x) \leq \sup_{y\in\mathbb{Y}_{\mathbf{x}}} f_{\mathbf{x}}^+(y). \tag{3.4}$$

From the definition of $f_{\mathbf{x}}^-$, we have

$$\forall y \in \mathbb{Y}, f_x(y) \geq f_{\mathbf{x}}^-(y) \implies \forall y \in \mathbb{Y}_x, f_x(y) \geq f_{\mathbf{x}}^-(y), \tag{3.5}$$

and since $\mathbb{Y}_x \subseteq \mathbb{Y}_{\mathbf{x}}$,

$$\min_{y\in\mathbb{Y}_x} f_{\mathbf{x}}^-(y) \geq \min_{y\in\mathbb{Y}_{\mathbf{x}}} f_{\mathbf{x}}^-(y). \tag{3.6}$$

From Equation 3.5 and 3.6, it follows that

$$f_{sup}(x) = f_x(y_x^*) \geq f_{\mathbf{x}}^-(y_x^*) \geq \min_{y\in\mathbb{Y}_x} f_{\mathbf{x}}^-(y) \geq \min_{y\in\mathbb{Y}_{\mathbf{x}}} f_{\mathbf{x}}^-(y). \tag{3.7}$$

Ultimately, From Equation 3.4 and 3.7, we obtain

$$\min_{y\in\mathbb{Y}_{\mathbf{x}}} f_{\mathbf{x}}^-(y) \leq f_{sup}(x) \leq \sup_{y\in\mathbb{Y}_{\mathbf{x}}} f_{\mathbf{x}}^+(y)$$

which proves the first statement of Theorem 3.3.1.
The second statement of Theorem 3.3.1 corresponds to the unconstrained case. Its proof is straightforward from the definition of $f_{\mathbf{x}}^-$ and $f_{\mathbf{x}}^+$. ■

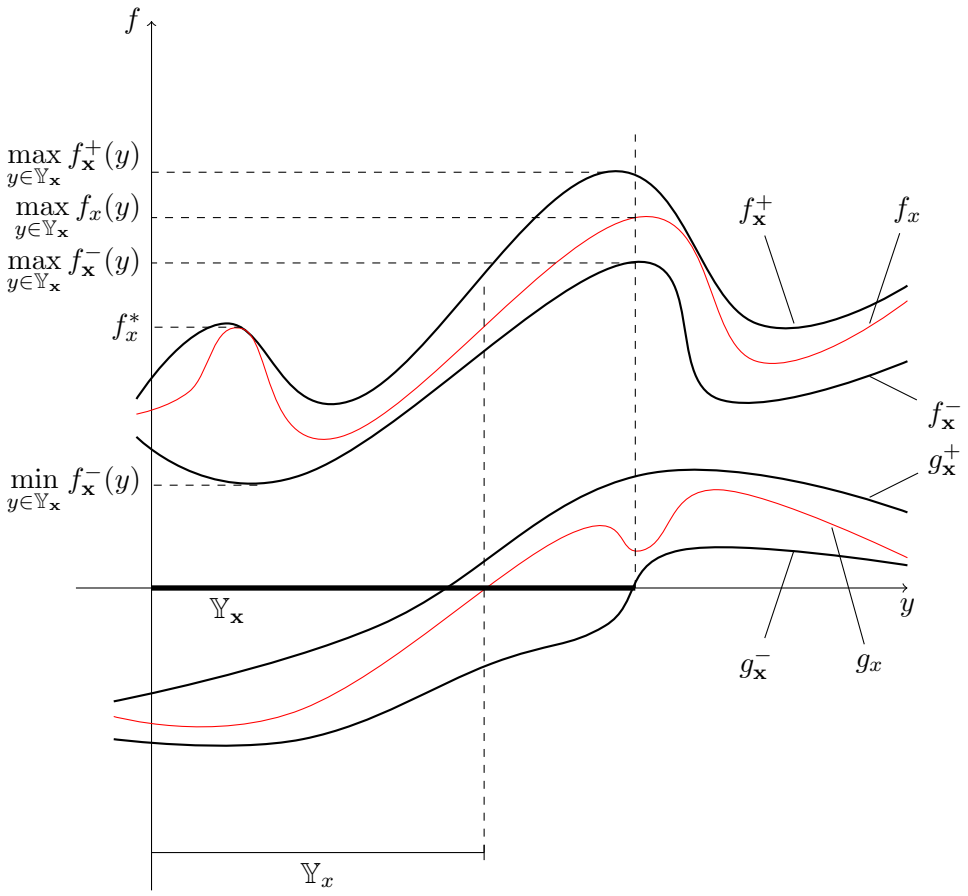


Figure 3.1: Optimization of a family of function with IBBA

The first statement of Theorem 3.3.1 corresponds to the general case where $\mathcal{M}_x$ is subject to constraints. In this case, $\sup_{y\in\mathbb{Y}_x} f^-_{\mathbf{x}}$ is not a reliable lower bound of $f_{sup}(\mathbf{x})$. In effect, given a vector $x \in \mathbf{x}$, since $\mathbb{Y}_x \subseteq \mathbb{Y}_{\mathbf{x}}$, there possibly exists an infeasible vector $y \in \mathbb{Y}_{\mathbf{x}} \setminus \mathbb{Y}_x$ such that $f^-_{\mathbf{x}}(y) > f^*_x$. Figure 3.1 illustrates this case, the set $\mathbb{Y}_{\mathbf{x}}$ is defined by a unique constraint and is represented in bold on the y axis. The second part of Theorem 3.3.1 is the unconstrained case, for which $\sup_{y\in\mathbb{Y}_{\mathbf{x}}} f^-_{\mathbf{x}}(y)$ can be used as a reliable lower bound of $f_{sup}(\mathbf{x})$.
Theorem 3.3.2 generalizes Theorem 3.3.1 by replacing $\mathbb{Y}_{\mathbf{x}}$ by any set $\mathbb{H}$ which is guaranteed to contain the set of maximizers $\{y^*_x \,|\, x \in \mathbf{x}\}$. This result will be needed further.

Theorem 3.3.2. *Consider a set $\mathbb{H} \subseteq \mathbb{Y}$ and a box $\mathbf{x} \in \mathbb{X}$. If $\forall x \in \mathbf{x}, y^*_x \in \mathbb{H}$ then*

$$f_{sup}(\mathbf{x}) \subseteq [\min_{y\in\mathbb{H}} f^-_{\mathbf{x}}(y), \sup_{y\in\mathbb{H}} f^+_{\mathbf{x}}(y)].$$

Moreover, if $\forall x \in \mathbf{x}, \mathbb{Y}_x = \mathbb{Y}$, then

$$f_{sup}(\mathbf{x}) \subseteq [\sup_{y\in\mathbb{H}} f^-_{\mathbf{x}}(y), \sup_{y\in\mathbb{H}} f^+_{\mathbf{x}}(y)].$$

Proof. The proof of Theorem 3.3.1 relies on the fact that $\forall x \in \mathbf{x}, y_x^* \in \mathbb{Y}_{\mathbf{x}}$ and that the feasible set $\mathbb{Y}_x$, which contains y_x^*, is a subset of $\mathbb{Y}_{\mathbf{x}}$.
In this theorem, the condition provides $\forall x \in \mathbf{x}, y_x^* \in \mathbb{H}$. Consequently, $y_x^* \in \mathbb{H} \cap \mathbb{Y}_x$. In addition, we have that $\mathbb{H} \cap \mathbb{Y}_x \subseteq \mathbb{H}$.
Therefore, the proof of Theorem 3.3.2 is the same as the proof of Theorem 3.3.1 replacing $\mathbb{Y}_{\mathbf{x}}$ by $\mathbb{H}$ and $\mathbb{Y}_x$ by $\mathbb{H} \cap \mathbb{Y}_x$. ■

Corollary 3.3.2.1 provides a practical way, using inclusion functions, to compute guaranteed bounds of $f_{sup}(\mathbf{x})$. It relies on the fact that an inclusion function enables to compute bounds enclosing the hull of $f_{\mathbf{x}}$ over a box $\mathbf{y}$.

Corollary 3.3.2.1. *Consider $\mathbb{H} = \bigcup_i \mathbf{y}_i$ a subset of $\mathbb{Y}$, $\mathbf{f}$ an inclusion function of f and a box $\mathbf{x} \subseteq \mathbb{X}$. If $\forall x \in \mathbf{x}, y_x^* \in \mathbb{H}$, then*

$$f_{sup}(\mathbf{x}) \subseteq [\min_i \underline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)}, \max_i \overline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)}].$$

Moreover, if $\forall x \in \mathbf{x}, \mathbb{Y}_x = \mathbb{Y}$, then

$$f_{sup}(\mathbf{x}) \subseteq [\max_i \underline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)}, \max_i \overline{\mathbf{f}(\mathbf{x}, \mathbf{Y}_i)}].$$

Proof. In order to prove Corollary 3.3.2.1, considering the statements of Theorem 3.3.2 it suffices to prove the following three statements:

1. $\min_i \underline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)} \leq \min_{y \in \mathbb{H}} f_{\mathbf{x}}^-(y)$,
2. $\max_i \overline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)} \geq \sup_{y \in \mathbb{H}} f_{\mathbf{x}}^+(y)$,
3. $\max_i \underline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)} \leq \sup_{y \in \mathbb{H}} f_{\mathbf{x}}^-(y)$.

Proof of 2: Let $y_s = \operatorname{argmax}_{y \in \mathbb{H}} f_{\mathbf{x}}^+(y)$. $\exists j$ such as $y_s \in \mathbf{y}_j$. From the definition of inclusion function, we have

$$\begin{aligned} \forall x \in \mathbf{x}, \forall y \in \mathbf{y}_j, \overline{\mathbf{f}(\mathbf{x}, \mathbf{y}_j)} \geq f(x, y) \quad &\Longrightarrow \forall x \in \mathbf{x}, \overline{\mathbf{f}(x, \mathbf{y}_j)} \geq f(x, y_s) \quad (\text{for } y_s \in \mathbf{y}_j), \\ &\Longrightarrow \overline{\mathbf{f}(\mathbf{x}, \mathbf{y}_j)} \geq f_{\mathbf{x}}^+(y_s), \\ &\Longrightarrow \max_i \overline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)} \geq f_{\mathbf{x}}^+(y_s). \end{aligned}$$

Proof of 3: Let $m = \operatorname{argmax}_i \underline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)}$ be the index of the box which provides the greatest lower bound. From the definition of inclusion function, we have

$$\begin{aligned} \forall x \in \mathbf{x}, \forall y \in \mathbf{y}_m, f(x, y) \geq \underline{\mathbf{f}(\mathbf{x}, \mathbf{y}_m)} \quad &\Longrightarrow \forall y \in \mathbf{y}_m, f_{\mathbf{x}}^-(y) \geq \underline{\mathbf{f}(\mathbf{x}, \mathbf{y}_m)}, \\ &\Longrightarrow \sup_{y \in \mathbb{H}} f_{\mathbf{x}}^-(y) \geq \underline{\mathbf{f}(\mathbf{x}, \mathbf{y}_m)}. \end{aligned}$$

Proof of 1: From 3, we have

$$\max_i \underline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)} \leq \sup_{y \in \mathbb{H}} f_{\mathbf{x}}^-(y) \quad \Longrightarrow \min_i \underline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)} \leq \sup_{y \in \mathbb{H}} f_{\mathbf{x}}^-(y).$$

■

Corollary 3.3.2.1 makes the connection with branch and bound algorithms. The union of boxes $\mathbf{y}_i$ guaranteed to contains the set of solutions corresponds to the boxes contained in the data structures $\mathcal{L}$ and $\mathcal{L}_s$ in the IBBA. Figure 3.2 illustrates Theorem 3.3.2 and its corollary. The set $\mathbb{H}$ corresponds to the projection of the gray boxes on the y axis.

Theorem 3.3.3 proposes a second formula for the lower bound of $f_{sup}(\mathbf{x})$, and its corollary offers a practical way to compute it using inclusion functions.

Theorem 3.3.3. *Consider $\tilde{y} \in \mathbb{Y}$ and $\mathbf{x} \subseteq \mathbb{X}$. If $\forall x \in \mathbf{x}, \tilde{y} \in \mathbb{Y}_x$ then $f^-_{\mathbf{x}}(\tilde{y})$ is a lower bound of $f_{sup}(\mathbf{x})$.*

Proof.

$$\begin{array}{lll} \forall x \in \mathbf{x}, \tilde{y} \in \mathbb{Y}_x & \implies \forall x \in \mathbf{x}, f_x(\tilde{y}) \leq f_{sup}(x) & \text{(def. of maximum)} \\ & \implies \forall x \in \mathbf{x}, f^-_{\mathbf{x}}(\tilde{y}) \leq f_{sup}(x) & \text{(def. of } f^-_{\mathbf{x}}\text{)}. \end{array}$$

■

Corollary 3.3.3.1. *Consider $\tilde{y} \in \mathbb{Y}$, $\mathbf{x} \subseteq \mathbb{X}$ and $\mathbf{f}$ an inclusion function of f. If $\forall x \in \mathbf{x}, \tilde{y} \in \mathbb{Y}_x$ then $\underline{\mathbf{f}(\mathbf{x}, \tilde{y})}$ is a lower bound of $f_{sup}(\mathbf{x})$.*

Proof. It suffices to prove that $\underline{\mathbf{f}(\mathbf{x}, \tilde{y})} \leq f^-_{\mathbf{x}}(\tilde{y})$. This inequality arises directly from the definition of $f^-_{\mathbf{x}}$ and the property of inclusion functions.

■

Corollary 3.3.3.1 is in fact the extension of the solution search step of IBBA to the case of a set of functions.

In the end, this section provides practical tools to compute reliable bounds of f_{sup} over a box that can be directly embedded in a branch and bound algorithm, which is SIBBA describe in Section 3.3.5.

3.3.2 Consistency w.r.t infinite number of constraints

In order to use Corollary 3.3.3.1, a vector $\tilde{y}$ must be proved to belong to $\mathbb{Y}_x$, $\forall x \in \mathbf{x}$. Considering $x \in \mathbf{x}$, $\mathbb{Y}_x$ is the feasible set of CSP $\mathcal{S}_x$ composed of n_g constraints.

$$\mathcal{S}_x \begin{cases} g_{x,i}(y) \leq 0 & \forall i \in \{1, \ldots, n_g\} \\ y \in \mathbb{Y} \end{cases}$$

We recall that $\mathbb{S} \subset \mathbb{X} \times \mathbb{Y}$ is the feasible set defined by the constraints g_i.
Proving that $\forall x \in \mathbf{x}$, $\tilde{y} \in \mathbb{Y}_x$ means to solve a CSP composed of an infinite number of constraints. This can be done thanks to the consistency property of contractors. Theorem 3.3.4 goes a step further by considering a box $\mathbf{y}$ instead of a vector.

Theorem 3.3.4. *Consider $\mathbf{x}$ and $\mathbf{y}$ two boxes subset of $\mathbb{X}$ and $\mathbb{Y}$ respectively, and $\mathcal{C}_{inv}$ a contractor for the complementary set $\overline{\mathbb{S}}$ of $\mathbb{S}$. If $\mathcal{C}_{inv}(\mathbf{x}, \mathbf{y}) = \emptyset$ then $\forall x \in \mathbf{x}$, $\mathbf{y} \subseteq \mathbb{Y}_x$.*

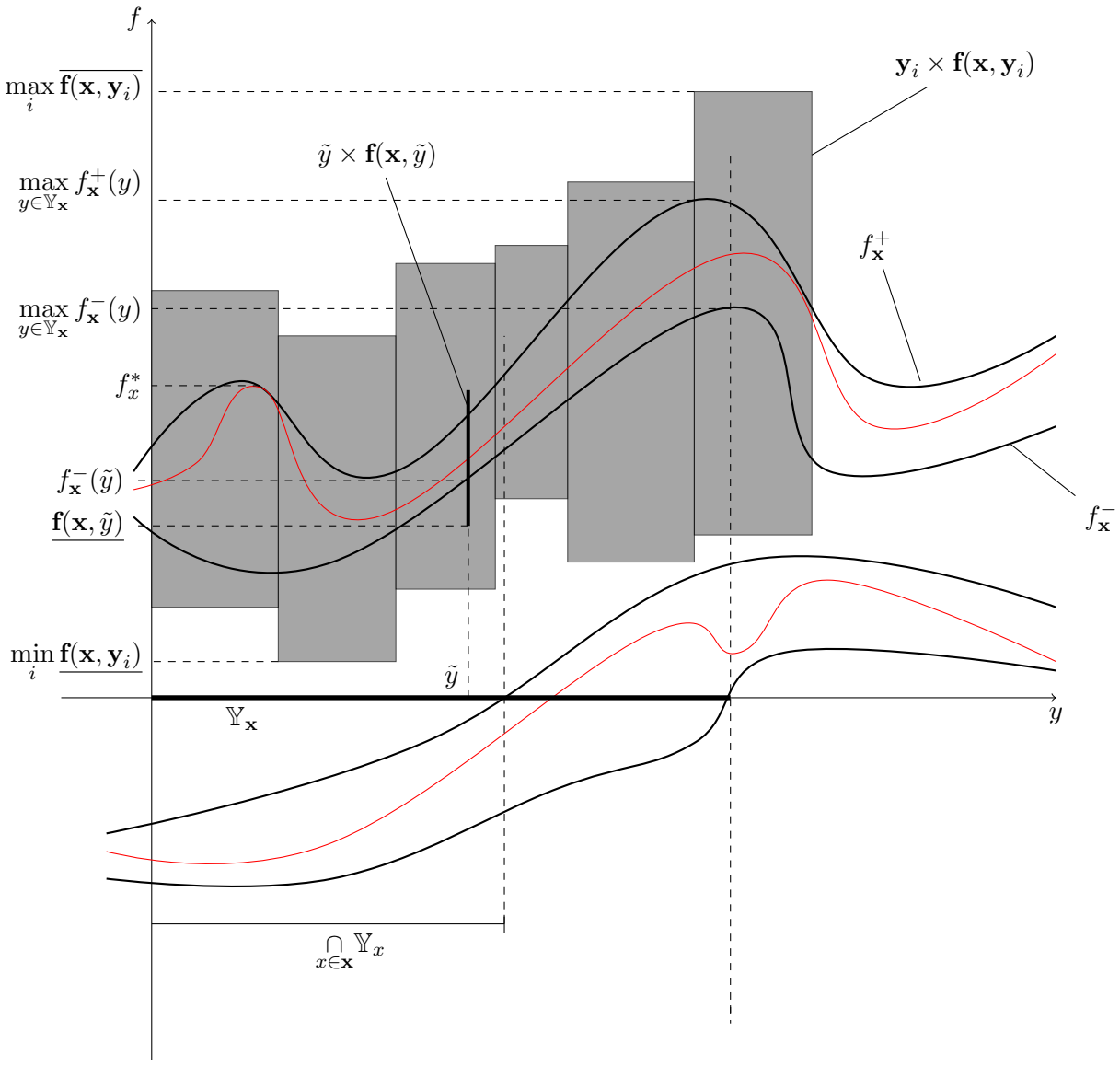


Figure 3.2: Optimization of a family of function with IBBA

Proof. Consider a vector $y \in \mathbf{y}$. Suppose $\exists x \in \mathbf{x}$ such as $y \notin \mathbb{Y}_x$.

$$\begin{aligned} y \notin \mathbb{Y}_x &\implies (x, y) \notin \mathbb{S} \\ &\implies (x, y) \in \overline{\mathbb{S}}. \end{aligned}$$

This is false due to the consistency property of $\mathcal{C}_{inv}$. As a consequence, $\forall y \in \mathbf{y}$, it holds that $\forall x \in \mathbf{x}$, $y \in \mathbb{Y}_x$ which proves Theorem 3.3.4 ■

Remark. The set defined by $\{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x},\ y \in \mathbb{Y}_x\}$ is equal to the intersection of all the sets $\mathbb{Y}_x$ for $x \in \mathbf{x}$,

$$\{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x},\, y \in \mathbb{Y}_x\} = \bigcap_{x \in \mathbf{x}} \mathbb{Y}_x.$$

It is important to note that even if $\mathbf{y} \subset \bigcap_{x \in \mathbf{x}} \mathbb{Y}_x$, $\mathcal{C}_{inv}$ does not necessarily contract $\mathbf{x} \times \mathbf{y}$ to the empty set since it is not minimal and because of the wrapping effect. In the following, a box $\mathbf{y}$ or a vector y will be said to be feasible if it is subset of $\bigcap_{x \in \mathbf{x}} \mathbb{Y}_x$.

3.3.3 Extension of contractors for the maximization of a set of functions

In Chapter 2, IBBA was introduced to solve minimization problems. Its efficiency relies on the acceleration techniques, framed as contractors. This section proposes the extension of those contractors to the case of the maximization of a set of functions. For every contractors introduced in this section, a particular care is being taken to prove that no maximizers y_x^* are discarded.

Contractor for the feasible set

Theorem 3.3.5 is the extension of the contractor $\mathcal{C}_{\mathbb{S}}$ introduced in Chapter 2. It enables to contract a box $\mathbf{y}$ with respect to the set $\mathbb{Y}_{\mathbf{x}}$.

Theorem 3.3.5. *Let $\mathcal{C}$ be a contractor for $\mathbb{S}$, and $\mathbf{x}_c$ and $\mathbf{y}_c$ the boxes resulting from the contraction by $\mathcal{C}$, $\mathcal{C}(\mathbf{x}_c, \mathbf{y}_c) = \mathbf{x}_c \times \mathbf{y}_c$. The function $\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$ defined by*

$$\begin{aligned} \mathcal{C}_{\mathbb{Y}_{\mathbf{x}}} :\ & \mathbb{IR}^m \to \mathbb{IR}^m \\ & \mathbf{y} \mapsto \mathbf{y}_c, \end{aligned}$$

is a contractor for $\mathbb{Y}_{\mathbf{x}}$. In addition, $\{y_x^* \,|\, x \in \mathbf{x}\} \cap \mathbf{y} \subseteq \mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}(\mathbf{y})$.

Proof. Suppose $\exists y \in \mathbb{Y}_{\mathbf{x}}$ such as $y \in \mathbf{y} \setminus \mathbf{y}_c$.

$$\begin{aligned} y \in \mathbb{Y}_{\mathbf{x}} &\implies \exists x \in \mathbf{x}, (x, y) \in \mathbb{S} && \text{(def. of } \mathbb{Y}_{\mathbf{x}}) \\ &\implies \exists (x, y) \notin \mathbf{x}_c \times \mathbf{y}_c, (x, y) \in \mathbb{S}. && \text{(since } y \notin \mathbf{y}_c) \end{aligned}$$

This statement is false due to the consistency property of $\mathcal{C}$. As a consequence, $y \in \mathbf{y}_c$ which proves the consistency property. The contraction property of $\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$ arises from the one of $\mathcal{C}$. As a consequence, $\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$ is a contractor for $\mathbb{Y}_{\mathbf{x}}$.
Consider $x \in \mathbf{x}$ such as $y_x^* \in \mathbf{y}$. From the definition of the optimizer, we have

$$\begin{aligned} y_x^* \in \mathbb{Y}_x \cap \mathbf{y} &\implies y_x^* \in \mathbb{Y}_{\mathbf{x}} \cap \mathbf{y} && \text{(since } \mathbb{Y}_x \subseteq \mathbb{Y}_{\mathbf{x}}) \\ &\implies y_x^* \in \mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}(\mathbf{y}) && \text{(consistency property of } \mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}) \end{aligned}$$

■

It is important to note that $\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$ does not contract $\mathbf{x}$. In effect, $(x, y) \notin \mathbb{S}$ does not imply that $x^* \neq x$ but only proves that x is not feasible at y. Figure 3.3a illustrates $\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$. The boxes $\mathbf{y}$ and $\mathbf{y}'$ are contracted around $\mathbb{Y}_{\mathbf{x}}$ displayed in bold.

Monotonicity contractor

The next contractor to be extended to the maximization of a set of functions is the monotonicity contractor. Note that this contractor is introduced for a minimization problem in Chapter 2, but is here considered for a maximization problem. However, this difference does not change the underlying principle of the contractor. Let J_y be the derivative of f with respect to the variable $y = (y_1, \ldots, y_m)$.

$$\begin{aligned} J_y : \quad & \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m, \\ & (x, y) \mapsto (\tfrac{\partial f}{\partial y_1}(x, y), \ldots, \tfrac{\partial f}{\partial y_m}(x, y))^T = (J_{y,1}(x, y), \ldots, J_{y,m}(x, y))^T. \end{aligned}$$

Algorithm 4 describes $\mathcal{C}_{J_y}$ the monotonicity contractor for the maximization of a set of functions, and Theorem 3.3.6 ensures that no maximizers y_x^* are discarded.

Algorithm 4 $\mathcal{C}_{J_y}$ Monotonicity contractor for set of function.

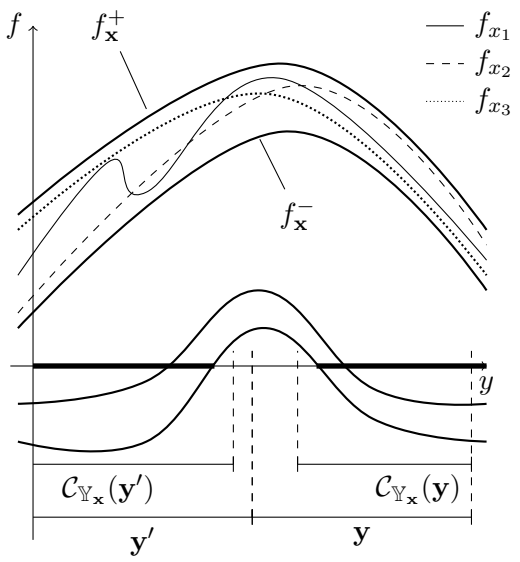
Input: $\mathbf{x}$, $\mathbf{y}$.
Output: Contracted $\mathbf{y}$.

```
if ∀x ∈ x, y ⊆ 𝕐_x. then
    Compute J_y(x, y).
    for i ∈ {1, ..., m}. do
        if J_{y,i}(x, y) ≥ 0. then
            y_i = ȳ_i.
        else if J_{y,i}(x, y) ≤ 0. then
            y_i = y_i.
        end if
    end for
end if
```

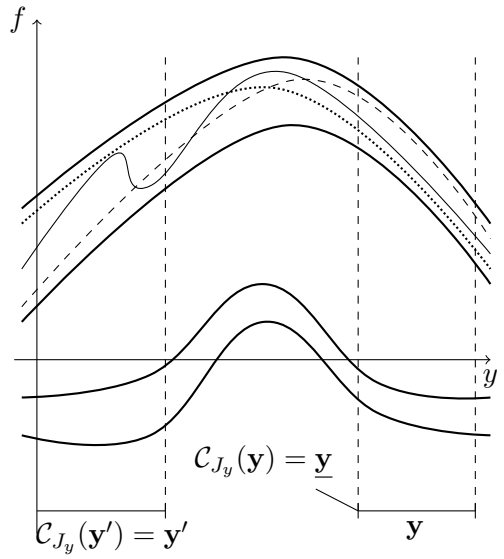
Theorem 3.3.6. *Consider two boxes* $\mathbf{x}$ *and* $\mathbf{y}$. $\mathcal{C}_{J_y}$ *is a contractor for the set* $\{\operatorname{argmax}_{y \in \mathbf{y}} f_x(y) \mid x \in \mathbf{x}\}$. *In addition* $\mathbf{y} \cap \{y_x^* \mid x \in \mathbf{x}\} \subseteq \mathcal{C}_{J_y}(\mathbf{y})$.

Proof. Consider $i \in \{1, \ldots, m\}$.

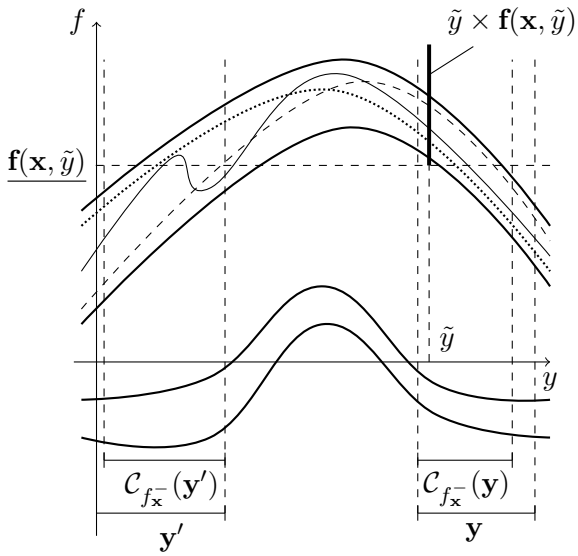
$$\begin{aligned} \underline{\mathbf{J}_{y,i}(\mathbf{x}, \mathbf{y})} \geq 0 \quad & \implies \forall x \in \mathbf{x}, \underline{\mathbf{J}_{y,i}(x, \mathbf{y})} \geq 0, \\ & \implies \forall x \in \mathbf{x}, \overline{\forall y \in \mathbf{y}, f_x(y)} \leq f_x(y_1, \ldots, \overline{\mathbf{y}_i}, \ldots, y_m), \\ & \implies \forall x \in \mathbf{x}, (\operatorname*{argmax}_{y \in \mathbf{y}} f_x(y))_i = \overline{\mathbf{y}_i}, \\ & \implies \forall x \in \mathbf{x}, \operatorname*{argmax}_{y \in \mathbf{y}} f_x(y) \subseteq \mathbf{y}_1 \times \cdots \times \overline{\mathbf{y}_i} \times \cdots \times \mathbf{y}_m. \end{aligned}$$



(a) Contraction done by $\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$. $\mathbb{Y}_{\mathbf{x}}$ is represented in bold over the y axis. Both y and y' are contracted.



(b) Contraction done by the monotonicity contractor $\mathcal{C}_{J_y}$. $\mathbf{y}$ is contracted into its lower bound since all functions are monotonically decreasing. $\mathbf{y}'$ is not contracted since f_{x_1} is not monotonous over this box.



(c) Contraction done by $\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$. Subsets of $\mathbf{y}$ and $\mathbf{y}'$ are discarded for they do not maximize f for any $x \in \mathbf{x}$.

Figure 3.3: Illustration of the contraction done by the contractors for the maximization of a set of functions under constraint. The solid, dotted and dashed curves represent three particular functions taken in $f_{\mathbf{x}}$ at x_1, x_2 and x_3 respectively.

The same reasoning leads to

$$\overline{\mathbf{J}_{y,i}(\mathbf{x},\mathbf{y})} \leq 0 \implies \forall x \in \mathbf{x}, \operatorname*{argmax}_{y\in\mathbf{y}} f_x(y) \subseteq \mathbf{y}_1 \times \cdots \times \underline{\mathbf{y}_i} \times \cdots \times \mathbf{y}_m.$$

Let $\mathbf{y}_{c,i}$ be the box resulting from the contraction of $\mathbf{y}$ along its i^{th} dimension, that is:

$$\mathbf{y}_{c,i} = \begin{cases} \mathbf{y}_1 \times \cdots \times \overline{\mathbf{y}_i} \times \cdots \times \mathbf{y}_m, & \text{if } \underline{\mathbf{J}_{y,i}(\mathbf{x},\mathbf{y})} \geq 0, \\ \mathbf{y}_1 \times \cdots \times \underline{\mathbf{y}_i} \times \cdots \times \mathbf{y}_m, & \text{if } \overline{\mathbf{J}_{y,i}(\mathbf{x},\mathbf{y})} \leq 0, \\ \mathbf{y} & \text{else.} \end{cases} \tag{3.8}$$

Since $\operatorname*{argmax}_{y\in\mathbf{y}} f_x(y) \subseteq \mathbf{y}_{c,i}, \forall i \in \{1,\ldots,m\}$, we have

$$\forall x \in \mathbf{x}, \operatorname*{argmax}_{y\in\mathbf{y}} f_x(y) \subseteq \mathbf{y}_{c,i} \subseteq \bigcap_{i\in\{1,\ldots,m\}} \mathbf{y}_{c,i} = \mathcal{C}_{J_y}(\mathbf{y}),$$

which proves the consistency property. The contraction property is straightforward. As a consequence, $\mathcal{C}_{J_y}$ is a contractor for $\{\operatorname*{argmax}_{y\in\mathbf{y}} f_x(y) \,|\, x \in \mathbf{x}\}$.
In order to prove the second statement of Theorem 3.3.6, consider $x \in \mathbf{x}$ such as $y_x^* \in \mathbf{y}$. Since $\mathbf{y} \subseteq \mathbb{Y}_x$ (theorem hypothesis) we have from the definition of the maximizer

$$\begin{aligned} y_x^* = \operatorname*{argmax}_{y\in\mathbf{y}} f_x(y) &\implies y_x^* \subseteq \mathcal{C}_{J_y}(\mathbf{y}) \qquad \text{(consistency of } \mathcal{C}_{J_y}) \\ &\implies \mathbf{y} \cap y_x^* \subseteq \mathcal{C}_{J_y}(\mathbf{y}). \end{aligned}$$

Therefore, $\forall x \in \mathbf{x}$, $\mathbf{y} \cap y_x^* \subseteq \mathcal{C}_{J_y}(\mathbf{y})$, which proves the second statement. ■

As for the monotonicity contractor $\mathcal{C}_J$, $\mathcal{C}_{J_y}$ can only be used if $\mathbf{y}$ is feasible. Otherwise, the maximizer of f_x over $\mathbf{y}$ is not proved to be feasible, and the contraction could lead to the deletion of y_x^*. Note that the monotonicity test cannot be used to contract $\mathbf{x}$. Indeed, if f is monotonically increasing with respect to x over $\mathbf{y}$, it does not prove that it is the case at $y \notin \mathbf{y}$. This is illustrated Figure 3.3b which depicts the contraction performed by $\mathcal{C}_{J_y}$. The order of f_{x_1}, f_{x_2}, and f_{x_3} changes over $\mathbb{Y}$. Since f_{x_1} is not monotonous over $\mathbf{y}'$, this box is not contracted.

Contractor based on feasible points

Theorem 3.3.7 introduces the contractor $\mathcal{C}_{\tilde{f}_\mathbf{x}}$, the extension of the contractor $\mathcal{C}_{\tilde{f}}$ introduced in Chapter 2. It is based on the hull of $f_\mathbf{x}$, and its corollary provides a practical way to use it by the mean of inclusion functions.

Theorem 3.3.7. *Consider* $\mathbf{x}$ *a subset of* $\mathbb{X}$ *and* $\mathbf{y}$ *a subset of* $\mathbb{Y}$*. Let* $\tilde{y} \in \mathbb{Y}$ *be a feasible vector. Consider* $\mathcal{C}$ *a contractor for the set defined by the inequality* $f(x,y) \geq f_\mathbf{x}^-(\tilde{y})$*,* $\mathcal{C}(\mathbf{x}\times\mathbf{y}) = \mathbf{x}_c \times \mathbf{y}_c$*.*
The function

$$\begin{aligned} \mathcal{C}_{\tilde{f}_\mathbf{x}} : \; & \mathbb{IR}^m \to \mathbb{IR}^m \\ & \mathbf{y} \mapsto \mathbf{y}_c \end{aligned}$$

is a contractor for the set

$$\{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x},\, f_x(y) \geq f_{\mathbf{x}}^-(\tilde{y})\}.$$

In addition,

$$\mathbf{y} \cap \{y_x^* \,|\, x \in \mathbf{x}\} \subseteq \mathcal{C}_{\tilde{f}_{\mathbf{x}}}(\mathbf{y}).$$

Proof. Suppose $\{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x},\ f_x(y) \geq f_{\mathbf{x}}^-(\tilde{y})\} \not\subseteq \mathbf{y}_c$, as a consequence

$$\exists x \in \mathbf{x},\ \exists y \notin \mathbf{y}_c,\ f_x(y) \geq f_{\mathbf{x}}^-(\tilde{y}) \implies \exists (x,y) \notin \mathbf{x}_c \times \mathbf{y}_c, f_x(y) \geq f_{\mathbf{x}}^-(\tilde{y}).$$

This statement is false due to the consistency property of $\mathcal{C}$. Therefore,

$$\{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x},\ f_x(y) \geq f_{\mathbf{x}}^-(\tilde{y})\} \subseteq \mathbf{y}_c,$$

which proves the consistency of $\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$. The contraction property of $\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$ arises from the one of $\mathcal{C}$. Consider $x \in \mathbf{x}$ such that $y_x^* \in \mathbf{y}$. Since $\tilde{y} \in \mathbb{Y}_x$, from the definition of maximum we have

$$f_x(y_x^*) \geq f_x(\tilde{y}) \implies f_x(y_x^*) \geq f_{\mathbf{x}}^-(\tilde{y}).$$

As a consequence $y_x^* \in \mathcal{C}_{\tilde{f}_{\mathbf{x}}}(\mathbf{y})$ due to the consistency property of $\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$. Finally,

$$\mathbf{y} \cap \{y_x^* \,|\, x \in \mathbf{x}\} \subseteq \mathcal{C}_{\tilde{f}_{\mathbf{x}}}(\mathbf{y}).$$

■

Corollary 3.3.7.1. *Let* $\mathbf{f}$ *be an inclusion function of* f*. If* $\mathcal{C}$ *is a contractor for*

$$\{(x,y) \in \mathbf{x} \times \mathbf{y} \,|\, f(x,y) \geq \underline{\mathbf{f}(\mathbf{x}, \tilde{y})}\},$$

then $\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$ *is a contractor for the set*

$$\{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x},\, f_x(y) \geq f_{\mathbf{x}}^-(\tilde{y})\}.$$

In addition,

$$\mathbf{y} \cap \{y_x^* \,|\, x \in \mathbf{x}\} \subseteq \mathcal{C}_{\tilde{f}_{\mathbf{x}}}(\mathbf{y}).$$

Proof. Replacing $f_{\mathbf{x}}^-(\mathbf{y})$ by $\underline{\mathbf{f}(\mathbf{x}, \tilde{y})}$ in the proof of Theorem 3.3.7, we obtain that $\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$ is a contractor for the set

$$\{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x},\, f_x(y) \geq \underline{\mathbf{f}(\mathbf{x}, \tilde{y})}\}.$$

From the property of inclusion function, we have that

$$f_{\mathbf{x}}^-(\mathbf{y}) \geq \underline{\mathbf{f}(\mathbf{x}, \tilde{y})} \implies \{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x},\, f_x(y) \geq f_{\mathbf{x}}^-(\mathbf{y})\} \subseteq \{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x},\, f_x(y) \geq \underline{\mathbf{f}(\mathbf{x}, \tilde{y})}\}.$$

This proves the consistency of $\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$. The contraction property arises from the one of $\mathcal{C}$.
The proof of the second statement is straightforward from the one of Theorem 3.3.7. ■

$\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$ does not allow to contract $\mathbf{x}$. It simply enables to discard subset of $\mathbf{y}$ which are proved not to maximize f_x over $\mathbb{Y}$ for any $x \in \mathbf{x}$, as shown on Figure 3.3c. Note that $\mathbf{y}$ does not need to be feasible. A vector y, feasible or not, is not a maximizer if another feasible vector $\tilde{y}$ provides a greater value of f.

3.3.4 Contractor for subsets of x

The contractor introduced here does not contract $\mathbf{y}$ but $\mathbf{x}$, contrary to the other contractors presented in this section.

Theorem 3.3.8. *Consider two boxes $\mathbf{x} \in \mathbb{X}$ and $\mathbf{y} \in \bigcap_{x \in \mathbf{x}} \mathbb{Y}_x$, and $b \in \mathbb{R}$. Let $\mathcal{C}$ be a contractor for the inequality $f(x, y) \leq b$, and $\mathbf{x}_c$, $\mathbf{y}_c$ be the boxes resulting from the contraction, $\mathcal{C}(\mathbf{x}, \mathbf{y}) = \mathbf{x}_c \times \mathbf{y}_c$.*
Let $\mathcal{C}_l$ be a function defined by

$$\begin{array}{rl} \mathcal{C}_l : & \mathbb{IR}^n \to \mathbb{IR}^n \\ & \mathbf{x} \mapsto \begin{cases} \emptyset & \text{if } \mathbf{y}_c \subset \mathbf{y} \text{ (strict inclusion)} \\ \mathbf{x}_c & \text{else.} \end{cases} \end{array}$$

$\mathcal{C}_l$ is a contractor for the set $\{x \in \mathbf{x} \,|\, f_x^* \leq b\}$. In addition, $\{x^*\} \cap \mathbf{x} \subseteq \mathcal{C}_l(\mathbf{x})$.

Proof. Consider first the case where $\mathbf{y}_c \subset \mathbf{y}$. Due to the strict inclusion, $\mathbf{y} \setminus \mathbf{y}_c \neq \emptyset$. Let y_o be a vector in $\mathbf{y} \setminus \mathbf{y}_c$. Since $\mathcal{C}$ is a contractor, from the consistency property we have

$$\begin{array}{lll} y_o \notin \mathbf{y}_c & \implies \forall x \in \mathbf{x}, (x, y_o) \notin \mathbf{x}_c \times \mathbf{y}_c, & \\ & \implies \forall x \in \mathbf{x}, f(x, y_o) > b, & \text{(consistency of } \mathcal{C}) \\ & \implies \forall x \in \mathbf{x}, f_x^* > b. & \text{(def. of maximum)} \end{array}$$

As a consequence,

$$\{x \in \mathbf{x} \,|\, f_x^* \leq b\} = \emptyset \subseteq \mathcal{C}_l(\mathbf{x}).$$

Consider the second case where $\mathbf{y}_c = \mathbf{y}$. Suppose that $\exists x \in \mathbf{x}_c$ such as $f_x^* > b$. It implies that

$$\exists x \notin \mathbf{x}_c, f(x, y_x^*) \geq b \implies \exists (x, y) \notin \mathbf{x}_c \times \mathbf{y}_c, f(x, y) \geq b.$$

This statement is false due to the consistency property of $\mathcal{C}$. As a consequence $\{x \in \mathbf{x} \,|\, f_x^* \leq b\} \subseteq \mathcal{C}_l(\mathbf{x})$, which proves the consistency property $\mathcal{C}_l$. The contraction property of $\mathcal{C}_l$ arises from the one of $\mathcal{C}$.

■

This contractor will play different roles, whether the objective f_{sup} or the SIC function q_{sup} is considered.

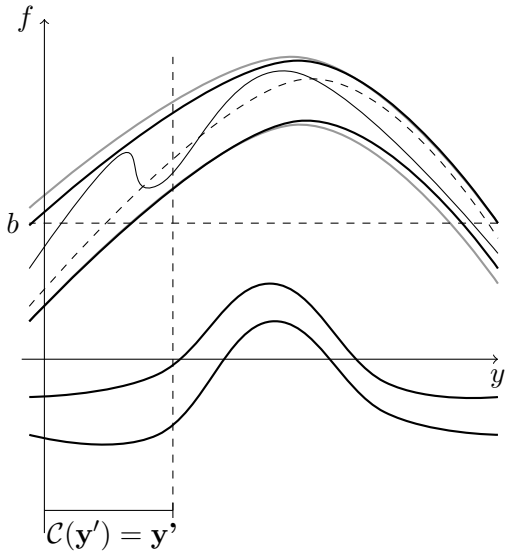
- If f_{sup} is considered. If b is an upper bound on the minimum f_{sup}^* of problem $\mathcal{M}$, then $\mathcal{C}_l$ is a contractor for $\{x^*\}$.

$$\begin{array}{ll} x \in \mathbf{x} \setminus \mathbf{x}_c & \implies f_{sup}(x) > b \geq f_{sup}^*, \\ & \implies x \neq x^*. \end{array}$$

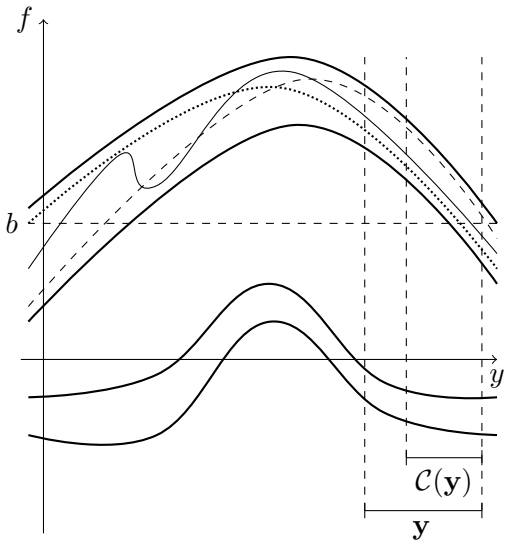
- If q_{sup} is considered. In this case, choosing $b = 0$ provides a contractor for $\mathbb{Q}$ the subset of $\mathbb{X}$ defined by the SIC.

$$\begin{array}{ll} x \in \mathbf{x} \setminus \mathbf{x}_c & \implies q_{sup}(x) > 0, \\ & \implies \exists z \in \mathbb{Z}_x,\ q(x, z) > 0, \\ & \implies x \notin \mathbb{Q}. \end{array}$$

  By extension $\mathcal{C}_l$ is also a contractor for $\{x^*\}$ in this case, since $x^* \in \mathbb{Q}$.



(a) The forward backward contraction of $\mathbf{x} \times \mathbf{y}'$ is performed with respect to the constraint $f(x, y) \leq b$. $\mathbf{x}$ is contracted to $\mathbf{x}_c$, resulting in a smaller set of functions represented by the new hull in bold. x_3 has been discarded in the contraction process since f_{x_3} is strictly greater than $f_{sup}(\tilde{x})$ over $\mathbf{y}'$. $\mathbf{y}'$ is not contracted.



(b) $\mathbf{y}$ is contracted by $\mathcal{C}$, proving that $x^* \notin x$. As a consequence, $\mathcal{C}_l$ contracts $\mathbf{x}$ to $\emptyset$.

Figure 3.4: Illustration of the contraction of $\mathbf{x}$ done by $\mathcal{C}_l$.

$\mathcal{C}_l$ does not improve the convergence of the resolution of the maximization problem, but instead improve the one of the minimization problem $\mathcal{M}$. However, $\mathcal{C}_l$ is introduced here because it will be used repeatedly during the resolution of the maximization problem done by SIBBA. Figure 3.4 illustrates $\mathcal{C}_l$ with f_{sup}.

3.3.5 SIBBA algorithm

In Section 3.3.1 it has been shown how bounds on the set of maxima can be computed, and in Section 3.3.3 several contractors have been proposed to discard subsets proved not to contains the maximizers y_x^*. The Set Interval Branch and Bound Algorithm (SIBBA) integrates those results in a branch and bound framework.
SIBBA takes as inputs:

- a maximization problem, given either by f_{sup} or q_{sup},
- a box $\mathbf{x}$ defining the set of problems,
- an initial domain, either $\mathbb{Y}$ or $\mathbb{Z}$,
- a minimum width ϵ_y for the bisection,

- the number of iterations `iter`,

and returns

- $\mathcal{L}_{\mathbf{x}}$ a heap containing the set of solutions $\{y_x^* \,|\, x \in \mathbf{x}\}$ or $\{z_x^* \,|\, x \in \mathbf{x}\}$,
- an enclosure $[c^-, c^+]$ of the set of maxima $f_{sup}(\mathbf{x})$ or $q_{sup}(\mathbf{x})$,
- the box $\mathbf{x}$ contracted by $\mathcal{C}_l$.

The extraction and insertion steps of SIBBA are similar to the ones of IBBA presented in Chapter 2. Both $\mathcal{L}_{\mathbf{x}}$ and $\mathcal{L}_s$ are binary heaps, the boxes $\mathbf{y}$ are sorted such that the first box $\mathbf{y}$ has the greatest upper bound,

$$\mathbf{y} = \max_{\mathbf{y}_i \in \mathcal{L}_{\mathbf{x}}} \overline{\mathbf{f}(\mathbf{x}, \mathbf{y}_i)}.$$

The bisection step is also the same as the one of IBBA, the boxes are stored in the memory heap $\mathcal{L}_s$ when their widths is lower than ϵ_y.
Once a box $\mathbf{y}$ is extracted from $\mathcal{L}_{\mathbf{x}}$, it is first contracted by $\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$ to discard non feasible subsets at line 3, and also by $\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$ if a feasible solution $\tilde{y}$ has already been found. Then, $\mathbf{y}$ is verified to be feasible or not as explained in Section 3.3.2. If $\mathbf{y}$ is feasible, then $\mathbf{x}$ is contracted by $\mathcal{C}_l$, and $\mathbf{y}$ is contracted by the monotonicity contractor $\mathcal{C}_{J_y}$. At line 10, a feasible vector y is searched in $\mathbf{y}$. If such a vector is found, by virtue of Corollary 3.3.3.1 the lower bound of $\mathbf{f}(\mathbf{x}, y)$ is a lower bound of $f_{sup}(\mathbf{x})$. If this bound is greater than c^-, this last takes this new value. At line 11, if $\mathbf{x}$ has been contracted to the empty set by $\mathcal{C}_l$, it means that $x^* \notin \mathbf{x}$. In this case, SIBBA terminates and returns the empty set. Otherwise, SIBBA terminates when $\mathcal{L}_{\mathbf{x}}$ is empty or the maximum number of iteration `iter` is reached. The elements of $\mathcal{L}_s$ are inserted in $\mathcal{L}_{\mathbf{x}}$, such that the set of solutions $\{y_x^* \,|\, x \in \mathbf{x}\}$ is contained in $\mathcal{L}_{\mathbf{x}}$. If $\mathcal{L}_{\mathbf{x}}$ is empty, then $\mathbb{Y}_{\mathbf{x}}$ is empty and $x^* \notin \mathbf{x}$. In this case, SIBBA returns the empty set. Finally, the upper and lower bounds on $f_{sup}(\mathbf{x})$ are computed at line 30 and line 32, based on the Corollaries 3.3.2.1 and 3.3.3.1. Note that a feasible solution is not necessarily found.

In the end, SIBBA enables to:

- contract $\mathbf{x}$ with respect to $\{x^*\}$,
- compute an enclosure $[c^-, c^+]$ of $f_{sup}(\mathbf{x})$ or $q_{sup}(\mathbf{x})$,
- compute a superset of $\{y_x^* \,|\, x \in \mathbf{x}\}$ or $\{z_x^* | x \in \mathbf{x}\}$, contained in $\mathcal{L}_{\mathbf{x}}$.

Reliability Thanks to the consistency of the contractors $\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$, $\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$, and $\mathcal{C}_{J_y}$, it is guaranteed that no solutions y_x^* are discarded during the execution of SIBBA. Consequently, the union of the boxes contained in $\mathcal{L}_{\mathbf{x}}$ is proved to contain $\{y_x^* \,|\, x \in \mathbf{x}\}$. The union of boxes contained in $\mathcal{L}_{\mathbf{x}}$ corresponds to the set $\mathbb{H}$ of Corollaries 3.3.3.1 and 3.3.2.1. Those two corollaries ensure the reliability of $[c^-, c^+]$.

Remark. If $\mathbf{x}$ is contracted to $\mathbf{x}_c$ by $\mathcal{C}_l$, the contraction of a box $\mathbf{y}$ done by the other contractors remains reliable.

Algorithm 5 Set Interval Branch and Bound algorithm: SIBBA.

Input: $\mathcal{M}_x$, $\mathbf{x}$, b, $\mathbb{Y}$, ϵ_y, `iter`
Output: $\mathcal{L}_{\mathbf{x}}$, $[c^-, c^+]$, contracted $\mathbf{x}$

while `i` $\leq$ `iter` and $\mathcal{L}_{\mathbf{x}}$ not empty. **do**
  Extract a box $\mathbf{y}$ from $\mathcal{L}_{\mathbf{x}}$. ▷ extraction
  Contract $\mathbf{y}$ with $\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$ ▷ contraction any box
  Contract $\mathbf{y}$ with $\mathcal{C}_{\tilde{f}_{\mathbf{x}}}$
  if $\mathbf{y}$ is feasible **then** ▷ contraction feasible box
    Contract $\mathbf{x}$ with $\mathcal{C}_l$ at $\mathbf{y}$
    Contract $\mathbf{y}$ with $\mathcal{C}_{J_y}$
  end if
  Search a feasible solution y in $\mathbf{y}$,
  update c^-. ▷ solution search
  if $\mathbf{x} = \emptyset$ **then**
    return $\emptyset$.
  else if $y \neq \emptyset$ **then**
    Compute $\mathbf{f}(\mathbf{x}, \mathbf{y})$. ▷ bounds computation
    Bisect $\mathbf{y}$ into $\mathbf{x}_1$ and $\mathbf{x}_2$. ▷ bisection
    for $i \in \{1, 2\}$. **do**
      if $w(\mathbf{y}_i) \geq \epsilon_y$ **then** ▷ insertion
        Insert $\mathbf{y}_i$ in $\mathcal{L}_{\mathbf{x}}$.
      else
        Insert $\mathbf{y}_i$ in $\mathcal{L}_s$
      end if
    end for
  end if
  `i++`
end while
Insert element of $\mathcal{L}_s$ in $\mathcal{L}_{\mathbf{x}}$
if $\mathcal{L}_{\mathbf{x}}$ is empty **then**
  Return $\emptyset$
else
  $c^+ := \max_{\mathbf{y} \in \mathcal{L}_{\mathbf{x}}} \overline{\mathbf{f}(\mathbf{y})}$
  if No feasible $\tilde{y}$ has been found **then**
    $c^- := \min_{\mathbf{y} \in \mathcal{L}_{\mathbf{x}}} \underline{\mathbf{f}(\mathbf{y})}$
  end if
end if

- $\mathbb{Y}_{\mathbf{x}_c} \subseteq \mathbb{Y}_{\mathbf{x}} \subseteq \mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}(\mathbf{y})$,
- $\{\underset{y\in\mathbf{y}}{\operatorname{argmax}} f_x(y) \,|\, x \in \mathbf{x}_c\} \subseteq \{\underset{y\in\mathbf{y}}{\operatorname{argmax}} f_x(y) \,|\, x \in \mathbf{x}\} \subseteq \mathcal{C}_{J_y}(\mathbf{y})$,
- $\{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x}_c,\, f_x(y) \geq f^-_{\mathbf{x}_c}(\tilde{y})\} \subseteq \{y \in \mathbf{y} \,|\, \forall x \in \mathbf{x},\, f_x(y) \geq f^-_{\mathbf{x}}(\tilde{y})\} \subseteq \mathcal{C}_{\tilde{f}_{\mathbf{x}}}(\mathbf{y})$.

$[c^-, c^+]$ is still a reliable enclosure for $f_{sup}(\mathbf{x}_c)$ since $f_{sup}(\mathbf{x}_c) \subseteq f_{sup}(\mathbf{x})$.

Convergence of the bounds. The values to which c^+ and c^- converge are studied. First of all, it must be noticed that the hull of $f_{\mathbf{x}}$ defined by $f^-_{\mathbf{x}}$ and $f^+_{\mathbf{x}}$ cannot provides an exact enclosure of $f_{sup}(\mathbf{x})$ in the general case, as stated by Theorem 3.3.9.

Theorem 3.3.9. *Consider* $\mathbf{x} \in \mathbb{X}$.

- *Unconstrained case: If* $\forall x \in \mathbf{x}$, $\mathbb{Y}_x = \mathbb{Y}$, *then*
  - $\overline{f_{sup}(\mathbf{x})} = \sup_{y\in\mathbb{Y}} f^+_{\mathbf{x}}(y)$,
  - $\underline{f_{sup}(\mathbf{x})} \geq \sup_{y\in\mathbb{Y}} f^-_{\mathbf{x}}(y)$.
- *Constrained case:*
  - $\overline{f_{sup}(\mathbf{x})} \leq \sup_{y\in\mathbb{Y}_{\mathbf{x}}} f^+_{\mathbf{x}}(y)$,
  - $\underline{f_{sup}(\mathbf{x})} \geq \sup_{y\in\bigcap_{x\in\mathbf{x}}\mathbb{Y}_x} f^-_{\mathbf{x}}(y)$.

Proof.

- Unconstrained case:
  - $$\begin{aligned}\overline{f_{sup}(\mathbf{x})} &= \sup_{x\in\mathbf{x}} f^*_x \\ &= \sup_{x\in\mathbf{x}} \sup_{y\in\mathbb{Y}} f(x,y) \\ &= \sup_{y\in\mathbb{Y}} \sup_{x\in\mathbf{x}} f(x,y) \\ &= \sup_{y\in\mathbb{Y}} f^+_{\mathbf{x}}(y) \qquad (\text{ def. of } f^+_{\mathbf{x}}).\end{aligned}$$
  - $$\begin{aligned}\underline{f_{sup}(\mathbf{x})} &= \min_{x\in\mathbf{x}} f^*_x \\ &= \min_{x\in\mathbf{x}} \sup_{y\in\mathbb{Y}} f(x,y) \\ &\leq \sup_{y\in\mathbb{Y}} \min_{x\in\mathbf{x}} f(x,y) \\ &= \sup_{y\in\mathbb{Y}} f^-_{\mathbf{x}}(y) \qquad (\text{ def. of } f^-_{\mathbf{x}}).\end{aligned}$$
- Constrained case:

- $\overline{f_{sup}(\mathbf{x})} = \sup_{x\in\mathbf{x}} \sup_{y\in\mathbb{Y}_{\mathbf{x}}} f(x,y),$
  $\geq \sup_{x\in\mathbf{x}} \sup_{y\in\mathbb{Y}_x} f(x,y), \quad (\text{ because } \mathbb{Y}_x \subseteq \mathbb{Y}_{\mathbf{x}})$
  $= \sup_{y\in\mathbb{Y}_{\mathbf{x}}} \sup_{x\in\mathbf{x}} f(x,y),$
  $= \sup_{y\in\mathbb{Y}_{\mathbf{x}}} f^+_{\mathbf{x}}(y) \qquad (\text{ def. of } f^+_{\mathbf{x}}).$
- $\underline{f_{sup}(\mathbf{x})} = \min_{x\in\mathbf{x}} \sup_{y\in\mathbb{Y}_{\mathbf{x}}} f(x,y),$
  $\leq \min_{x\in\mathbf{x}} \sup_{y\in\bigcap_{x\in\mathbf{x}}\mathbb{Y}_x} f(x,y), \quad (\text{ because } \bigcap_{x\in\mathbf{x}}\mathbb{Y}_x \subseteq \mathbb{Y}_x)$
  $\leq \sup_{y\in\bigcap_{x\in\mathbf{x}}\mathbb{Y}_x} \min_{x\in\mathbf{x}} f(x,y),$
  $= \sup_{y\in\bigcap_{x\in\mathbf{x}}\mathbb{Y}_x} f^-_{\mathbf{x}}(y). \qquad (\text{ def. of } f^-_{\mathbf{x}}).$

■

Only the upper bound provided by $f^+_{\mathbf{x}}$ is exact for the unconstrained case. The other bounds are exact only in particular cases. In addition, even if $\mathbf{f}$ is thin, it is not minimal and suffers from pessimism. Consequently, given $y \in \mathbb{Y}$, $f^-_{\mathbf{x}}(y) \geq \underline{\mathbf{f}(\mathbf{x},y)}$ because $\mathbf{x}$ is a box. In the end, c^- and c^+ converge to the following values as the width of the boxes $\mathbf{y}$ tends to 0.

- Unconstrained case:

$$\begin{aligned}\lim_{w(\mathbf{y})\to 0} c^+ &= \sup_{y\in\mathbb{Y}} \overline{\mathbf{f}(\mathbf{x},y)},\\ \lim_{w(\mathbf{y})\to 0} c^- &= \sup_{y\in\mathbb{Y}} \underline{\mathbf{f}(\mathbf{x},y)}.\end{aligned}$$

- Constrained case: Let $\mathbb{N} \subset \mathbb{Y}$ be the set discarded by the contractors, and $\mathbb{P}_{\mathbf{x}} \subseteq \bigcap_{x\in\mathbf{x}}\mathbb{Y}_x$ be the set that can be proved to be feasible.

$$\begin{aligned}\lim_{w(\mathbf{y})\to 0} c^+ &= \sup_{y\in\mathbb{Y}\setminus\mathbb{N}} \overline{\mathbf{f}(\mathbf{x},y)},\\ \lim_{w(\mathbf{y})\to 0} c^- &= \begin{cases} \sup_{y\in\mathbb{P}_{\mathbf{x}}\setminus\mathbb{N}} \underline{\mathbf{f}(\mathbf{x},y)} & \text{if } \mathbb{P}_{\mathbf{x}} \neq \emptyset \\ \min_{y\in\mathbb{Y}\setminus\mathbb{N}} \underline{\mathbf{f}(\mathbf{x},y)} & \text{else.}\end{cases}\end{aligned}$$

In conclusion, SIBBA converges at best to the hull of $f_{\mathbf{x}}$ over the subset $\mathbb{Y}\setminus\mathbb{N}$, and the bounds obtained from the hull are not equal to the one of $f_{sup}(\mathbf{x})$. The precision of the enclosure $[c^-, c^+]$ depends both on the expression of f and the width of $\mathbf{x}$.

Remark. If $w(\mathbf{x}) = 0$, then SIBBA performs the same as IBBA and $[c^-, c^+]$ converges to $f_{sup}(\mathbf{x})$.

3.3.6 Numerical example

Consider the constrained minmax problem taken from [117],

$$\begin{cases} \min\limits_{x\in\mathbb{X}} \sup\limits_{y\in\mathbb{Y}} & f(x,y) = (\cos y + \cos(2y+x))^2, \\ \text{s. t.} & g_1(x,y) = y - x(x+6.28) \leq 0, \\ & g_2(x,y) = y - x(x-6.28) \leq 0. \end{cases}$$

with $\mathbb{X} = \mathbb{Y} = [-3.14, 3.14]$. SIBBA is run with the box $\mathbf{x} = [-0.5, -0.2]$. Figure 3.5 shows the boxes $\mathbf{y}_i$ contained in $\mathcal{L}_{\mathbf{x}}$ and the bounds $[c^-, c^+]$ at different iterations of SIBBA. $\mathcal{C}_l$ is not used in this example and ϵ_y is set to 0. The contractor $\mathcal{C}_{\mathbb{Y}_{\mathbf{x}}}$ discards completely the non feasible set at the first iteration. The monotonicity contractor $\mathcal{C}_{J_y}$ is not efficient in this case since the feasible set is very small (located on the left).The lower bound c^- almost equal to zero, since the feasible solutions provide low lower bound. It is represented by a cross on the 100^{th} iteration. One can remark that c^+ converges rapidly to a constant value, but does not decrease after the 30^{th} iteration.

In this section, SIBBA is introduced to compute bounds on f_{sup} and q_{sup}. In addition, it enables to contract a box $\mathbf{x}$ for the solution x^* of the problem $\mathcal{M}$. Therefore, SIBBA can be embedded into a classical branch and bound to solve minmax problems subject to SIC, which is the topic of the next section.

3.4 MinMax problems subject to SIC

This section extends IBBA presented in Section 2 for the resolution of the problem $\mathcal{M}$. The problem is first considered without SIC in subsection 3.4.1 which proposes an adaptation of IBBA integrating SIBBA for bounds computation and contraction. An acceleration technique is proposed in subsection 3.4.2. The bisection and storage strategy in heap of IBBA are preserved.

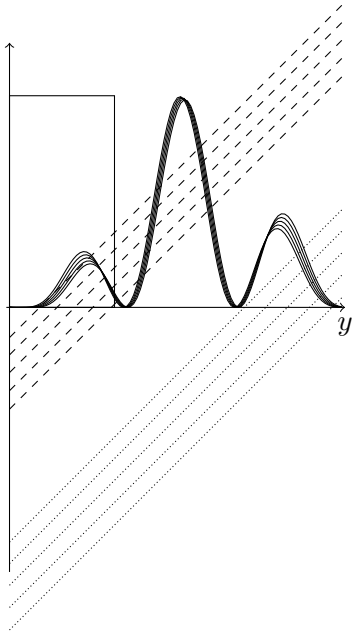
3.4.1 Algorithm for minmax problems

Consider the minmax problem without SIC given by $\mathcal{M}_{mm}$,

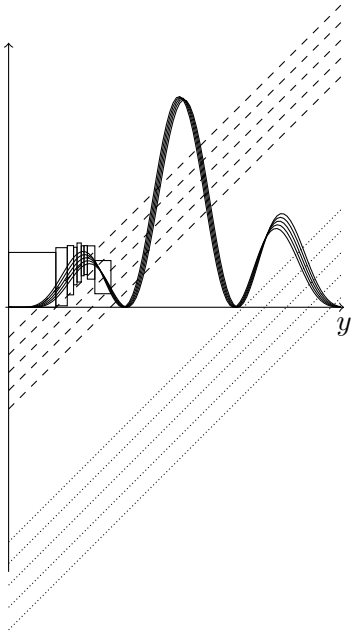
$$\mathcal{M}_{mm} : \begin{cases} \min\limits_{x\in\mathbb{X}} & f_{sup}(x), \\ \text{s. t.} & p_j(x) \leq 0,\ \forall j \in \{1, \ldots, n_p\} \end{cases}$$

In Section 3.2, a strategy to solve $\mathcal{M}$ is proposed consisting in using two branch and bounds algorithm. The main one is used to solve $\mathcal{M}_{mm}$ and the second one is SIBBA which enables to compute bounds of the objective function f_{sup}. The Min Max Interval Branch and Bound Algorithm (MMIBBA), described by Algorithm 6, is the main algorithm. The feasible set of $\mathcal{M}_{mm}$ defined by the constraints p_j is denoted $\mathbb{P}$ and $\mathcal{C}_{\mathbb{P}}$ is the forward backward contractor for $\mathbb{P}$.
MMIBBA takes as inputs:

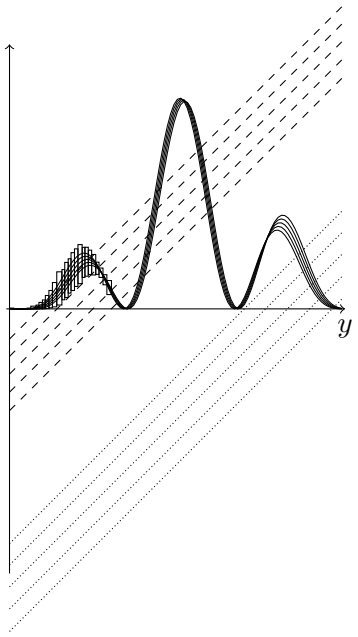
- the problem $\mathcal{M}_{mm}$,



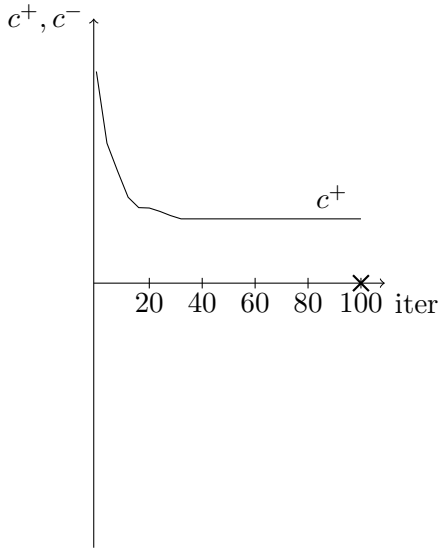
(a) First iteration of SIBBA.



(b) 10^{th} iteration of SIBBA.



(c) 100^{th} iteration of SIBBA.



(d) Evolution of c^+ and c^- over the iterations of SIBBA.

Figure 3.5: Evolution of $\mathcal{L}_{\mathbf{x}}$ and the bounds of $f_{sup}(\mathbf{x})$ over the iterations of SIBBA. f is represented by the solid curves at several values of x, g_1 by the dashed curves and g_2 by the dotted curves.

- the initial domains $\mathbb{X}$ and $\mathbb{Y}$,
- the bisection precision ϵ_x,
- the relative precision ϵ_r,

and returns

- an enclosure $[b^-, b^+]$ of f^*_{sup},
- the heaps $\mathcal{L}$ and $\mathcal{L}_s$ containing x^*,
- the best feasible solution found $\tilde{x}$.

Algorithm 6 Branch and Bound algorithm for Minmax problems.

Input: $\mathbb{X}$, $\mathbb{Y}$, $\mathcal{M}_{mm}$, ϵ_x
Output: Enclosure of the minimum, heps $\mathcal{L}$ and $\mathcal{L}_s$, best known solution $\tilde{x}$

while $b^+ - b^- \leq \epsilon_r |b^+|$ **do**
  Extract a box $\mathbf{x}$ from $\mathcal{L}$. ▷ Extraction
  Contract $\mathbf{x}$ with $\mathcal{C}_{\mathbb{P}}$. ▷ Contraction
  if $\mathbf{x} \neq \emptyset$ **then**
    Contract $\mathbf{x}$ and
     compute bounds of $f_{sup}(\mathbf{x})$ with SIBBA. ▷ Bound computation
    Search for a feasible vector $x \in \mathbf{x}$,
     compute an upper bound c^+ of $f_{sup}(x)$ using SIBBA,
     update the best current solution $\tilde{x}$ and b^+. ▷ Solution search
    Bisect $\mathbf{x}$ into p boxes $\mathbf{x}_i$. ▷ Bisection
    for $i \in \{1, \ldots, p\}$. **do**
      if $w(\mathbf{x}_i) \geq \epsilon_x$ **then** ▷ Insertion
        Insert $\mathbf{x}_i$ in $\mathcal{L}$.
      else
        Insert $\mathbf{x}_i$ in $\mathcal{L}_s$
      end if
    end for
  end if
end while

MMIBBA is very similar to the IBBA. The only steps that change are the bound computation and the solution search steps. The computation of the bounds of the objective f_{sup} is performed by SIBBA over $\mathbb{Y}$. At this step, SIBBA also performed a contraction of $\mathbf{x}$ with respect to the set $\{x \in \mathbf{x} \,|\, f_{sup}(x) \leq b^+\}$ with b^+ the upper bound on f^*_{sup}, as explained in Section 3.3.4. This upper bound b^+ is computed at the solution search step. If a feasible vector x is found, SIBBA is run once again to compute an enclosure of $f_{sup}(x)$. If the upper bound computed on $f_{sup}(x)$ is lower than the current best minimum b^+, this last is updated as well as the best solution $\tilde{x}$. When b^+ is updated, the elements $\mathbf{x}$ of $\mathcal{L}$ such as

$$\underline{f_{sup}(\mathbf{x})} > b^+,$$

are discarded. When MMIBBA terminates, $\mathcal{L}$ and $\mathcal{L}_s$ contain x^*, and $[b^-, b^+]$ is a reliable enclosure of f^*_{sup}. As for IBBA, b^- is the lowest lower bound on the objective given by the boxes in $\mathcal{L}$ and $\mathcal{L}_s$,

$$b^- = \min_{\mathbf{x} \in \mathcal{L} \cap \mathcal{L}_s} \underline{f_{sup}}(\mathbf{x}).$$

SIBBA parameters. SIBBA parameters ϵ_y and `iter` must be chosen at every execution. Two cases are considered.

- If SIBBA is executed at the solution search step, that is for a vector x, `iter` is set to ∞ and ϵ_y is chosen sufficiently low. In this way, SIBBA terminates when all the boxes $\mathbf{y}$ have reached the minimum width providing a small enclosure of $f_{sup}(x)$.
- The choice of SIBBA parameters when executed at the bound computation step, that is for a box $\mathbf{x}$, is discussed in the next subsection after an acceleration technique is introduced.

3.4.2 Inheritance strategy

The acceleration strategy presented here is sometime referred to as *propagation* in the literature, and is a known acceleration technique employed in [51, 136] in particular. We have chosen to named it the *inheritance* strategy in order to avoid confusion with constraint propagation.
In MMIBBA, each boxes $\mathbf{x}$ are processed independently. SIBBA is applied to $\mathbf{x}$, the list $\mathcal{L}_\mathbf{x}$ containing initially the search domain $\mathbb{Y}$. As a consequence, the solutions y^*_x are searched over the whole initial domain $\mathbb{Y}$. However, if two boxes $\mathbf{x}$ and $\mathbf{x}'$ such as $\mathbf{x}' \subseteq \mathbf{x}$ are considered, maximizing the set of functions $f_\mathbf{x}$ and maximizing $f_{\mathbf{x}'}$ are two closely dependent problems. In effect, the problem of maximizing $f_{\mathbf{x}'}$ is somehow contained in the problem of maximizing $f_\mathbf{x}$. Consequently, the results given by SIBBA when executed to maximize the set of functions $f_\mathbf{x}$ are also consistent for $f_{\mathbf{x}'}$, as stated by Theorem 3.4.1.

Theorem 3.4.1. *Consider a box $\mathbf{x}$, $\mathcal{L}_\mathbf{x}$ the data structure returned by SIBBA when run for $\mathbf{x}$, and $[c^-, c^+]$ the enclosure of $f_{sup}(\mathbf{x})$. If a box $\mathbf{x}'$ is such that $\mathbf{x}' \subset \mathbf{x}$, then*

$$\{y^*_x \,|\, x \in \mathbf{x}'\} \subseteq \bigcup_{y \in \mathcal{L}_\mathbf{x}} \mathbf{y},$$

and

$$f_{sup}(\mathbf{x}') \subseteq [c^-, c^+].$$

Proof. From Section 3.3, we have that

$$\{y^*_x \,|\, x \in \mathbf{x}\} \subseteq \bigcup_{\mathbf{y} \in \mathcal{L}_\mathbf{x}} \mathbf{y},$$

and

$$f_{sup}(\mathbf{x}) \subseteq [c^-, c^+].$$

Moreover, since $\mathbf{x}' \subseteq \mathbf{x}$,

$$\{y_x^* \,|\, x \in \mathbf{x}'\} \subseteq \{y_x^*, x \in \mathbf{x}\},$$

and

$$f_{sup}(\mathbf{x}') \subseteq f_{sup}(\mathbf{x}).$$

As a consequence,

$$\{y_x^* \,|\, x \in \mathbf{x}'\} \subseteq \bigcup_{\mathbf{y} \in \mathcal{L}_\mathbf{x}} \mathbf{y},$$

and

$$f_{sup}(\mathbf{x}') \subseteq [c^-, c^+].$$

■

At this point, two major aspects must be considered together:

- The first one arises directly from Theorem 3.4.1. In MMIBBA, the data structure $\mathcal{L}_\mathbf{x}$ computed by SIBBA with a box $\mathbf{x}$ can be reused as SIBBA's input for boxes $\mathbf{x}_i$ stemming from the bisection of $\mathbf{x}$. This way, when SIBBA is run for a box $\mathbf{x}_i$, the solutions are not searched over the whole search domain $\mathbb{Y}$ but only over the subset being the union of boxes in $\mathcal{L}_\mathbf{x}$. Concretely, when SIBBA is run for $\mathbf{x}_i$, it can be resumed where it stops when run for $\mathbf{x}$.

- The second aspect comes from Section 3.3. The numerical example shows that during the execution of SIBBA, the distance between the upper and lower bounds decreases significantly during the first iterations, but converges slowly thereafter. This phenomenon is in fact usual for most of branch and bound algorithms.

Considering these two points, the following strategy is adopted for MMIBBA. At the bound computation step, only few iterations of SIBBA are run. When $\mathbf{x}$ is bisected into $\mathbf{x}_i$, $\mathcal{L}_\mathbf{x}$ is duplicated and associated to each $\mathbf{x}_i$. When MMIBBA processes one of the $\mathbf{x}_i$, $\mathcal{L}_{\mathbf{x}_i}$ is initialized with $\mathcal{L}_\mathbf{x}$ in SIBBA. That is, the boxes processed in MIBBA inherit the data structure computed for the box they stemmed from. Consequently, the maximizations problems are not solved by SIBBA from the beginning at each iteration of MMIBBA, but instead are solved along with the minimization problem, solved by MMIBBA. This is the inheritance strategy, depicted in Figure 3.6. For the sake of visibility, only the objective function f is plotted over the y axis in the representation of SIBBA, no constraint is considered. The hulls of the sets of functions in MMIBBA converge whilst $\mathbb{X}$ is bisected in smaller boxes, and SIBBA converges simultaneously.

We propose to only choose the number of iterations of SIBBA at the bound computation step. As a consequence, the bisection precision ϵ_y is chosen sufficiently small to ensure a thin enclosure of f_{sup} at the solution search step. A reasonable choice is to set $\epsilon_y \leq \epsilon_x$ in the general case. Section 3.5 compares numerical results obtained for different values of `iter`, and emphasized the efficiency of the inheritance strategy.

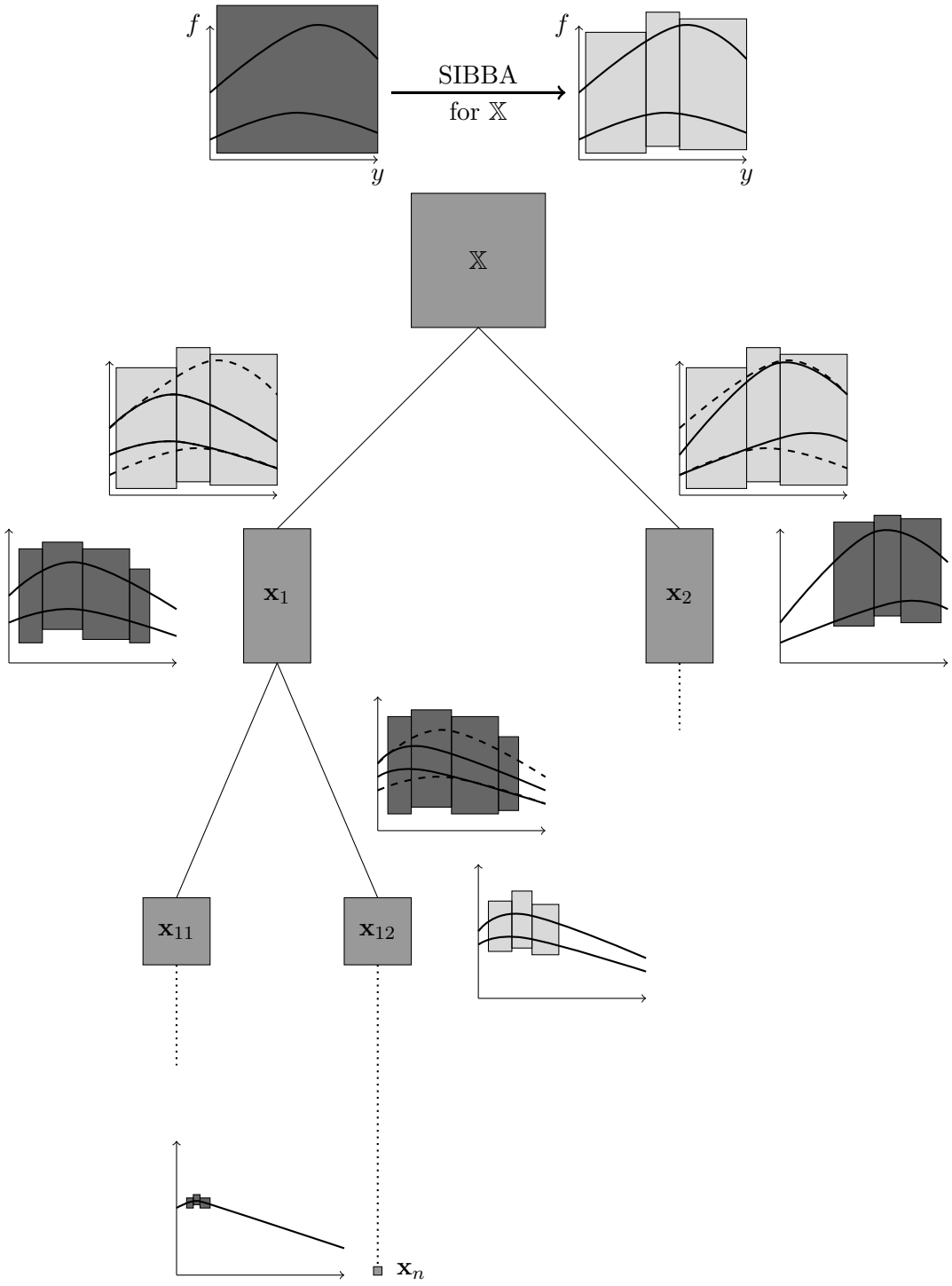


Figure 3.6: Illustration of inheritance strategy in MMIBBA.

3.4.3 Taking semi infinite constraints into account

The minmax problem subject to SIC, $\mathcal{M}$, is now considered.

$$\mathcal{M}: \begin{cases} \min\limits_{x\in\mathbb{X}} & f_{sup}(x) \\ \text{s. t.} & p_j(x) \leq 0,\ \forall j \\ & q_{sup}(x) \leq 0. \end{cases}$$

We recall that the set defined by the SIC is denoted $\mathbb{Q}$,

$$\mathbb{Q} = \{x \in \mathbb{X} \,|\, q_{sup}(x) \leq 0\} = \{x \in \mathbb{X} \,|\, q(x,z) \leq 0,\ \forall z \in \mathbb{Z}_x\}.$$

In order to prove that a box $\mathbf{x}$ is feasible with respect to the SIC, it suffices to prove that an upper bound of $q_{sup}(\mathbf{x})$ is lower than 0. On the contrary, $\mathbf{x}$ is proved to be infeasible if a lower bound of $q_{sup}(\mathbf{x})$ is strictly greater than 0. These bounds are computed by SIBBA, which also enables to contract $\mathbf{x}$ with respect to $\mathbb{Q}$ by setting the parameter b of SIBBA to 0 (see Section 3.3.4).
Eventually, Algorithm 7 extends MMIBBA to take SIC into account and implements the inheritance strategy. MMIBBA is therefore suited to solve problem $\mathcal{M}$. To this extent, MMIBBA takes as additional inputs `iter` and ϵ_z, and returns

- an enclosure $[b^-, b^+]$ of f^*_{sup}.
- the heaps $\mathcal{L}$ and $\mathcal{L}_s$ containing x^*,
- the best feasible solution found $\tilde{x}$.

For the sake of simplicity, the same number of iterations is performed whether SIBBA is executed for f_{sup} or q_{sup}.

The contraction steps at lines 3 and 4 enable to contract a box $\mathbf{x}$ with respect to the feasible set of $\mathcal{M}$ being $\mathbb{P} \cap \mathbb{Q}$. At the solution search step, a vector x is proved to be feasible if $x \in \mathbb{P}$ and $x \in \mathbb{Q}$. The first condition is easy to verify with $\mathcal{C}_{\mathbb{P}}$, and the second one is verified using SIBBA.
The inheritance strategy is implemented at line 11. The heaps $\mathcal{L}_{\mathbf{x}}$ resulting form the execution of SIBBA at the bound computation step, and $\mathcal{L}^{\forall}_{\mathbf{x}}$ resulting from the execution of SIBBA for q_{sup} at Line 4, are inserted in $\mathcal{L}$ with $\mathbf{x}$.
When MMIBBA terminates, $\mathcal{L}$ and $\mathcal{L}_s$ contain x^*, and $[b^-, b^+]$ is a reliable enclosure of f^*_{sup} thanks to the viable enclosure of f_{sup} provided by SIBBA.

Taking multiple SICs into account. Suppose that $\mathcal{M}$ is subject to n_q SICs

$$C^{\forall}_i : q_i(x, z_i) \leq 0,\ \forall z_i \in \mathbb{Z}_{x,i}.$$

If the SICs share the same variables, $z = z_i\,, \forall i$, and the sets $\mathbb{Z}_{x,i}$ are equal, then they can be reformulated as a single SIC being

$$C^{\forall} : \max_{i\in\{1,\ldots,n_q\}} q_i(x,z),\ \forall z \in \mathbb{Z}.$$

If it is not the case, SIBBA can be used for every SIC and the inheritance strategy applied. However, doing so is memory greedy since each of the SIC will provide a heap that will be stored.

Algorithm 7 Minmax Interval Branch and Bound algorithm implementing inheritance: MMIBBA.

Input: $\mathcal{M}$,ϵ_r,$\mathbb{X}$,$\mathbb{Y}$,$\mathbb{Z}$,ϵ_x,ϵ_y,ϵ_z,`iter`.
Output: Enclosure of the optimum, heaps $\mathcal{L}$ and $\mathcal{L}_s$, best known solution $\tilde{x}$

1: **while** $b^+ - b^- \leq \epsilon_r \max(1,' |b^+|)$ **do**
2: Extract a box $\mathbf{x}$ from $\mathcal{L}$. ▷ Extraction
3: Contract $\mathbf{x}$ with $\mathcal{C}_{\mathbb{P}}$. ▷ Contraction
4: Contract $\mathbf{x}$ with SIBBA w.r.t $\mathbb{Q}$.
5: **if** $\mathbf{x} \neq \emptyset$ **then**
6: Contract $\mathbf{x}$ and compute bounds of $f_{sup}(\mathbf{x})$ with SIBBA. ▷ Bound computation
7: Search for a feasible vector $x \in \mathbf{x}$, compute an upper bound c^+ of $f_{sup}(x)$ using SIBBA, update the best current solution $\tilde{x}$ and b^+. ▷ Solution search
8: Bisect $\mathbf{x}$ into p boxes $\mathbf{x}_i$. ▷ Bisection
9: **for** $i \in \{1, \ldots, p\}$ **do**
10: **if** $\mathbf{x}_i$ respects insertion criterion **then** ▷ Insertion
11: Insert $(\mathbf{x}_i, \mathcal{L}_{\mathbf{x}}, \mathcal{L}_{\mathbf{x}}^{\forall})$ in $\mathcal{L}$.
12: **else**
13: Insert $\mathbf{x}_i$ in $\mathcal{L}_s$.
14: **end if**
15: **end for**
16: **end if**
17: **end while**

Conclusion. MMIBBA is a direct adaptation of the IBBA. Only the methods to compute bounds on the objective function and the contraction with respect to the feasible set are different, but this is transparent for the algorithm. As a consequence, MMIBBA can solve any problems having a maximum or a regular objective, being subject to SIC or not. It suffices to use SIBBA when needed.

3.4.4 Sub-optimal set characterization

The sub-optimal set characterization of problem $\mathcal{M}$ is addressed. That is, the feasible set of the Semi Infinite Constraint Satisfaction Problem (SICSP) $\mathcal{M}_\alpha$ is aimed to be characterized.

$$\mathcal{M}_\alpha : \begin{cases} f(x,y) \leq \alpha, \forall y \in \mathbb{Y}_x, \\ p_j(x) \leq 0, \\ q(x,y) \leq 0, \forall z \in \mathbb{Z}_x, \\ x \in \mathbb{X}. \end{cases}$$

The feasible set of $\mathcal{M}_\alpha$ is denoted by $\mathbb{M}_\alpha$. As emphasized in Chapter 2, using contractors for the feasible set and its complementary improves the convergence speed of the FSCA.
Let $\mathbb{F}_\alpha = \{f(x,y) \leq \alpha \,|\, \forall y \in \mathbb{Y}_x\}$ be the set defined by the first SIC of $\mathcal{M}_\alpha$. SIBBA can be used as a contractor for $\mathbb{Q}$ and $\mathbb{F}_\alpha$, since $\mathbb{F}_\alpha$ is defined by a SIC. As a consequence, SIBBA can be used as two contractors $\mathcal{C}_\mathbb{Q}$ and $\mathcal{C}_{\mathbb{F}_\alpha}$ for these two sets. In addition, $\mathcal{C}_\mathbb{P}$ is a contractor for $\mathbb{P}$. In conclusion, $\mathcal{C}_{in} = \mathcal{C}_\mathbb{Q} \circ \mathcal{C}_{\mathbb{F}_\alpha} \circ \mathcal{C}_\mathbb{P}$ is a contractor for $\mathbb{M}_\alpha$.
We now focus on defining a contractor of the complementary of $\mathbb{M}_\alpha$. To do so, two contractors for the complementary of $\mathbb{Q}$ and $\mathbb{F}_\alpha$ are needed. To this extent, Theorem 3.4.2 provides a contractor for $\overline{\mathbb{F}_\alpha} = \{x \in \mathbf{x} \,|\, \exists y \in \mathbb{Y}_x,\ f(x,y) > \alpha\}$ the complementary of $\mathbb{F}_\alpha$. By extension, Theorem 3.4.2 provides contractors for any complementary of a set defined by a SIC.

Theorem 3.4.2. *Consider $\mathbb{H} = \bigcup_i \mathbf{y}_i$, $\mathbf{y}_i \subseteq \mathbb{IR}^m$, a subset of $\mathbb{Y}$. Let $\mathbf{x}$ be a box in $\mathbb{X}$, and $\mathcal{C}$ be a contractor for $\{(x,y) \in \mathbb{X} \times \mathbb{Y} \,|\, f(x,y) > \alpha\}$ such that $\mathcal{C}(\mathbf{x}, \mathbf{y}_i) = \mathbf{x}_{c,i} \times \mathbf{y}_{c,i}$. If $\forall x \in \mathbf{x}$, $y_x^* \in \mathbb{H}$, then the function*

$$\mathcal{C}_\exists : \quad \mathbb{IR}^{n_x} \to \mathbb{IR}^{n_x} \mathbf{x} \mapsto \bigsqcup_i \mathbf{x}_{c,i}$$

is a contractor for $\{x \in \mathbf{x} \,|\, \exists y \in \mathbb{Y}_x,\ f(x,y) > \alpha\}$.

Proof. Consider $x \in \bigcup_i \mathbf{x}_{c,i}$.

$$\exists i, x \in \mathbf{x}_{c,i}$$

Suppose $x \notin \{x \in \mathbf{x} \,|\, \exists y \in \mathbb{H},\ f(x,y) > \alpha\}$,

$$\begin{aligned} x \notin \{x \in \mathbf{x} \,|\, \exists y \in \mathbb{H},\ f(x,y) > \alpha\} \quad &\implies \forall y \in \mathbb{H}, f(x,y) \leq \alpha, \\ &\implies \exists y \in \mathbf{y}_{c,i}, f(x,y) \leq \alpha, \\ &\implies \exists (x,y) \in \mathbf{x}_i \times \mathbf{y}_{c,i}, f(x,y) \leq \alpha. \end{aligned}$$

This statement is false due to the consistency property of $\mathcal{C}$. As a consequence,

$$\begin{aligned} \{x \in \mathbf{x} \,|\, \exists y \in \mathbb{H},\ f(x,y) > \alpha\} \quad &\subseteq \bigcup_i \mathbf{x}_{c,i}, \\ &\subseteq \bigsqcup_i \mathbf{x}_{c,i}. \end{aligned}$$

$\mathcal{C}_\exists$ is therefore a contractor for $\{x \in \mathbf{x} \,|\, \exists y \in \mathbb{H},\ f(x,y) > \alpha\}$. In addition

$$\begin{array}{rll} x \in \{x \in \mathbf{x} \,|\, \exists y \in \mathbb{Y}_x,\ f(x,y) > \alpha\} & \Longrightarrow & \exists y \in \mathbb{Y}_x,\ f(x,y) > \alpha, \\ & \Longrightarrow & f(x, y_x^*) > \alpha, \qquad \text{(def. of maximum)} \\ & \Longrightarrow & \exists y \in \mathbb{H} f(x,y) > \alpha, \qquad (y_x^* \in \mathbb{H} \text{ by hyp.}) \\ & \Longrightarrow & x \in \{x \in \mathbf{x} \,|\, \exists y \in \mathbb{H},\ f(x,y) > \alpha\}, \\ & \Longrightarrow & \{x \in \mathbf{x} \,|\, \exists y \in \mathbb{Y}_x,\ f(x,y) > \alpha\}, \\ & & \quad \subseteq \{x \in \mathbf{x} \,|\, \exists y \in \mathbb{H},\ f(x,y) > \alpha\}. \end{array}$$

As a consequence, $\mathcal{C}_\exists$ is also a contractor for $\{x \in \mathbf{x} \,|\, \exists y \in \mathbb{Y}_x,\ f(x,y) > \alpha\}$. ■

Thanks to Theorem 3.4.2, two contractors $\mathcal{C}_{\overline{\mathbb{Q}}}$ and $\mathcal{C}_{\overline{\mathbb{F}_\alpha}}$ for $\overline{\mathbb{Q}}$ and $\overline{\mathbb{F}_\alpha}$ can be defined. In addition, a contractor $\mathcal{C}_{\overline{\mathbb{P}}}$ for $\overline{\mathbb{P}}$ the complementary of $\mathbb{P}$ can be obtained easily. In the end, $\mathcal{C}_{out} = \mathcal{C}_{\overline{\mathbb{Q}}} \cup \mathcal{C}_{\overline{\mathbb{F}_\alpha}} \cup \mathcal{C}_{\overline{\mathbb{P}}}$ is a contractor for $\overline{\mathbb{M}_\alpha}$.
Since two contractors $\mathcal{C}_{in}$ and $\mathcal{C}_{out}$ for $\mathbb{M}_\alpha$ and its complementary are defined, the FSCA introduced in Chapter 2 can be used directly to characterize $\mathbb{M}_\alpha$ by a sub-paving.

Remark.

- If the upper bound on $f_{sup}(\mathbf{x})$ computed by SIBBA when $\mathcal{C}_{\mathbb{F}_\alpha}$ is used is lower than 0, then $\mathbf{x} \subseteq \mathbb{F}_\alpha$ and there is no need to use $\mathcal{C}_{\overline{\mathbb{F}_\alpha}}$. The same remark holds for $q_{sup}(\mathbf{x})$.
- When $\mathcal{C}_{\mathbb{F}_\alpha}$ is used to contract $\mathbf{x}$, SIBBA is executed and returns the heap $\mathcal{L}_\mathbf{x}$. The boxes contained in $\mathcal{L}_\mathbf{x}$ are then used as input for the contractor $\mathcal{C}_{\overline{\mathbb{F}_\alpha}}$. The same remark goes for $\mathcal{C}_{\overline{\mathbb{Q}}}$.
- The inheritance strategy can also be used in the FSCA.

3.5 Numerical Results

We propose to test MMIBBA over several examples taken from the literature. Unfortunately, no benchmark problems has been found for minmax problems subject to quantified constraints. But, SIP and minmax problems can be found. MMIBBA is tested on 12 SIP problems taken from [88], and 20 minmax problems taken from [112] and [119]. These problems are given in Appendix A.
MMIBBA is executed with the following parameters for all the problems:

- Bisection precision for x: $\epsilon_x = 1e^{-4}$,
- Bisection precision for y (minmax problems): $\epsilon_y = 1e^{-5}$,
- Bisection precision for z (SIP problems): $\epsilon_z = 1e^{-5}$,
- Relative precision on the objective: $\epsilon = 1e^{-1}$.

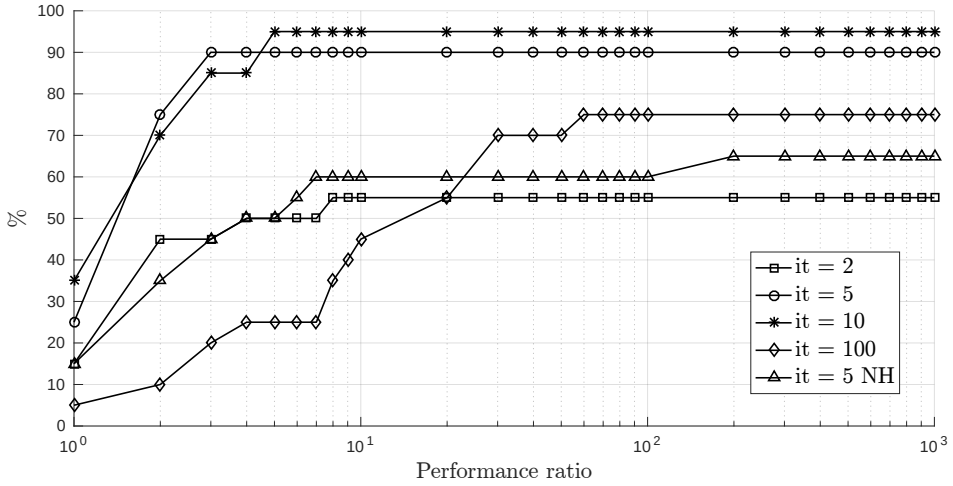


Figure 3.7: Performance profiles of MMIBBA on the minmax problems.

In order to study the inheritance strategy, MMIBBA is executed for different numbers of iterations `iter` of SIBBA : `iter` = 2,5,10,100. In addition, MMIBBA is executed without the inheritance strategy. In this case, the number of iterations performed by SIBBA is incremented by 5 between a box $\mathbf{x}$ and a sub-box $\mathbf{x}_i$ resulting from the bisection of $\mathbf{x}$. This way, SIBBA performed the same number of iterations as if `iter` = 5 with the inheritance strategy. This configuration is named 5NH.
The performance profiles obtained with the 5 different strategies for MMIBBA on the minmax and SIP problems are depicted on Figure 3.7 and 3.8, respectively. A performance profile is a graphical tools that represents the relative performances between different solvers on a set of benchmark problems [39]. Given a problem and a solver, the performance ratio is the ratio between the time taken by the solver to solve this problem and the time taken by the fastest solver to solve the same problem. The performance profiles show the percentage of problems solved within a given performance ratio.
The time limit for the resolution of the problems is set to 900 s. MMIBBA is stopped if it uses more than 4 GB of Random Access Memory (RAM).

From the performance profiles, it can be inferred that the inheritance strategy improves the convergence of MMIBBA. As expected from the numerical results of SIBBA presented in Section 3.3.6, performing too many iterations in SIBBA is not a good strategy. The configuration `iter` =2 provides good results on the SIP problems, but does not perform very well on the minmax problems. This can be explained by the fact that $\mathbb{X}$ is very large compared to $\mathbb{Z}$ for the SIP.
It seems that choosing a small value for `iter` is a reasonable choice. When MMIBBA is used the next chapters, `iter` is set to 5.

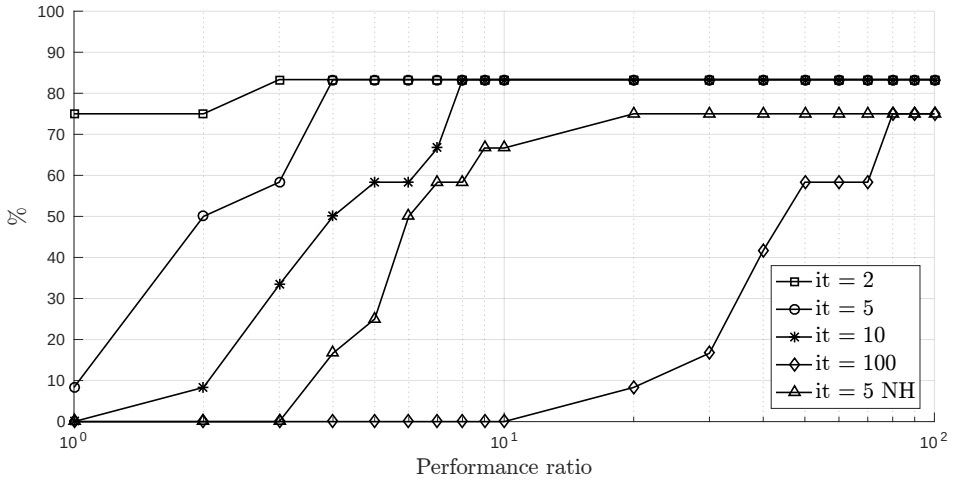


Figure 3.8: Performance profiles of MMIBBA on the SIP problems.

3.6 Conclusion and outlooks

Conclusion and contributions. In this Chapter an interval based algorithm, MMIBBA, has been proposed to solve the general problem $\mathcal{M}$. More generally, the proposed approach allows to solve any optimization problem being a minimization problem or a minmax problem, subject to SIC or not. It suffices to adapt the bound computation and contraction steps: if a minmax problem is considered, SIBBA is used to compute the bounds of the objective, if SICs are involved SIBBA is used for contraction. Thanks to SIBBA, MMIBBA is a direct extension of IBBA implementing the inheritance strategy and does not change the inherent structure of IBBA composed of the main steps: extraction, contraction, bound computation, solution search, bisection and insertion.
The three main contributions of this chapter are:

- an algorithm to solve a set of maximization problems subject to constraint: SIBBA,
- the formalization of the inheritance strategy, and the proof of its effectiveness,
- an algorithm to solve minmax problems subject to SIC: MMIBBA.

Strictly speaking, this chapter does not offer innovative solutions for the resolution of SIP or minmax problems since both have already been investigated with interval based branch and bound algorithms as mentioned in the introduction [51, 136]. What is innovative is the unified approach of those two problems that enables to deal with them at the same time by solving a minmax problem subject to SICs.

Improvement prospects. By now, MMIBBA implements the acceleration techniques based on constraint propagation, framed as contractors, and the inheritance strategy. Other approaches based on branch and bound propose solve SIPs by the convex reformulation of the

SICs [44, 20]. Those convexification methods could easily be added to our approach which is also based on a branch and bound algorithm. In addition bisection heuristics [35, 12, 55], already mentioned in Chapter 2, can be used to bisect boxes in a "smart" way. However the difficulty when having several tools at disposal, that may be computationally expensive or memory greedy, is to know which one should be used and in which case.
Another way to improve the convergence would be to tune dynamically the parameters `iter`, ϵ_y and ϵ_z during the execution of MMIBBA, or to choose others stop criterion for SIBBA. Finding heuristics would require an extensive study of the influence of SIBBA's parameters on the convergence rate.
Although the inheritance strategy makes it possible to improve the convergence speed of MMIBBA, it requires large amount of RAM. This is an issue especially on large scale problems, which can be addressed by limiting the size of the heap $\mathcal{L}_{\mathbf{x}}$ for example.

Some outlooks. A minmax problem is a particular case of *bi-level* problems. A bi-level problem is expressed as:

$$\mathcal{M}_{bi}: \begin{cases} \min\limits_{x\in\mathbb{X}} & F(x, y_x^*), \\ & \text{s. t. } G(x, y_x^*) \leq 0, \\ & y_x^* = \operatorname{argmin}\limits_{y\in\mathbb{Y}} f(x, y), \\ & \text{s. t.} \qquad\qquad\qquad g(x, y) \leq 0. \end{cases}$$

From a game theory point of view, P1 has an objective F different to the objective f of P2, as well as different constraints limit their play given by the constraints g and G. The bi-level problem is a minmax problem in the particular case where $F(x, y) = -f(x, y)$, and $G(x, y) = g(x, y)$. The bi-level denomination refers to the two levels of the problem. The upper level problem corresponds to the play of P1, that is the minimization of F subject to the G constraint. The lower level problem corresponds to the play of P2, that is the minimization of f subject to the g constraint.
One can remark that the lower level problem is expressed the same way as the maximization problems related to f_{sup} and q_{sup}. As a consequence, given a set $\mathbf{x}$, SIBBA can be used to compute a heap of boxes $\mathcal{L}_{\mathbf{x}}$ containing the set of minimizers $\{y_x^* \,|\, x \in \mathbf{x}\}$, corresponding to the possible plays of P2. If an enclosure of F over $\mathbf{x}$ can be computed from $\mathcal{L}_{\mathbf{x}}$, then the bi-level problem can be solved using IBBA, and the inheritance strategy can be used.
Of course, further investigations must be led, but it seems that the results proposed in this section could be extended to the resolution of bi-level problems and propose new ways to solve them [59, 77].

Chapter 4

Global optimization approach to H_∞ synthesis and analysis

This chapter proposes to solve the H_∞ problems introduced in Chapter 1 with the global optimization algorithms presented in Chapter 3. These problems are the structured synthesis, the robustness analysis and the robust structured synthesis. They are expressed in a general way as non convex problems, in contrast with numerous works dedicated to convex reformulations. Following the works based on the parametric approaches [84, 5, 21], the three problems are formulated as optimization ones having explicit objective functions and constraints. In this way, they are suited to be solved in a global way with the algorithms developed in Chapters 2 and 3.

4.1 Global solution to structured H_∞ synthesis by performance output independence

The structured H_∞ synthesis problem is recalled here.

Problem 4.1.1. *Structured Synthesis (SS)*

$$\begin{cases} \min\limits_{K \in \mathcal{K}_s} & ||F(P,K)||_\infty, \\ s.\ t. & K \text{ ensures internal stability}, \end{cases}$$

where K is the controller, P the augmented system, and $\mathcal{K}_s$ the set of structured controllers. It is supposed, without loss of generality, that the structured controllers are parametrized by $k \in \mathbb{R}^{n_k}$. In order to solve the SS problem globally, using the algorithms of Chapter 2 and 3, explicit formula for the H_∞ norm of $F(P, K(k))$ and the stability are required.

4.1.1 Explicit formula of the H_∞ norm of MISO systems

In the following, the frequency representation of the interconnection of P with K is considered, $F(P,K) = T_{w\to z}$. Since the controller depends on the tunable parameters k, $T_{w\to z}(k)$ is a matrix of transfer functions that also depends on k. More precisely, the coefficients of the polynomials in s being the numerators and the denominators of the transfers depend on k.

From the definition of the H_∞ norm, which is the maximum singular value of $T_{w\to z}$ over the frequencies, no explicit formula is available. However, if each of the performance outputs z_j are considered separately, it is possible to express the H_∞ norm of $T_{w\to z_j}$ as the maximization over the pulsations of a fraction of polynomials in ω, which coefficients depend on k.

Theorem 4.1.1. *The infinity norm of a MISO system is the maximum over the pulsations of the square root of a rational function.*

Proof. Consider the MISO system,

$$M(s) = (M_1(s), \dots, M_n(s)). \tag{4.1}$$

From the definition of the H_∞ norm,

$$\begin{aligned}
||M||_\infty &= \sup_{\omega\geq 0} \sigma_{max}(M(i\omega)) \\
&= \sup_{\omega\geq 0} \left(\lambda_{max}(M(i\omega)M(i\omega)^H)\right)^{1/2} \\
&= \sup_{\omega\geq 0} \left(\lambda_{max}\left((M_1(i\omega), \dots, M_n(i\omega))\begin{pmatrix}\overline{M_1(i\omega)}\\ \vdots \\ \overline{M_n(i\omega)}\end{pmatrix}\right)\right)^{1/2} \\
&= \sup_{\omega\geq 0} \left(\sum_{j=1}^n M_j(i\omega)\overline{M_j(i\omega)}\right)^{1/2} \\
&= \sup_{\omega\geq 0} \left(\sum_{j=1}^n |M_j(i\omega)|^2\right)^{1/2}.
\end{aligned}$$

Since $M_j(i\omega)$ is a fraction of two polynomials in ω, its modulus is also a fraction of two polynomials. By extension, $\sum_{j=1}^n |M_j(i\omega)|^2$ is also a fraction of polynomials. ■

Thanks to Theorem 4.1.1, $||T_{w\to z_j}(k)||_\infty$ is expressed as the maximum over the pulsations of the square root of a fraction of two polynomials in ω which coefficients depend on k, that is the maximization of an explicit function.

We define f_j as

$$f_j(k,\omega) = \sqrt{\sum_{i=1}^n |T_{w_i\to z_j}(k, i\omega)|^2},$$

and as a consequence

$$||T_{w\to z_j}||_\infty = \sup_{\omega\geq 0} f_j(k,\omega).$$

Note that f_j is not convex in the general case.

Recalling that $T_{w\to z_j}$ is the weighted counterpart of $T_{w\to \tilde{z_j}}$, $T_{w\to z_j} = T_{w\to \tilde{z_j}} Z_j$ (see Chapter 1), Equation 4.2 provides an interpretation of the inequality $||T_{w\to z_j}||_\infty \leq 1$ as a bound

on the response of $T_{w\to\tilde{z_j}}$ given by the maximal singular value.

$$
\begin{aligned}
||T_{w\to z_j}(k)||_\infty \leq 1 &\iff \sup_\omega \sqrt{\textstyle\sum_{i=1}^n |T_{w_i\to z_j}(k,i\omega)|^2} && \leq 1 \\
&\iff \forall\omega,\ \sqrt{\textstyle\sum_{i=1}^n |T_{w_i\to \tilde{z_j}}(k,i\omega)Z_j(i\omega)|^2} && \leq 1 \\
&\iff \forall\omega,\ \sqrt{\textstyle\sum_{i=1}^n |T_{w_i\to \tilde{z_j}}(k,i\omega)|^2|Z_j(i\omega)|^2} && \leq 1 \\
&\iff \forall\omega,\ \sqrt{|Z_j(i\omega)|^2\textstyle\sum_{i=1}^n |T_{w_i\to \tilde{z_j}}(k,i\omega)|^2} && \leq 1 \\
&\iff \forall\omega,\ \sqrt{\textstyle\sum_{i=1}^n |T_{w_i\to \tilde{z_j}}(k,i\omega)|^2} && \leq \frac{1}{|Z_j(i\omega)|} \\
&\iff \forall\omega,\ \sqrt{\textstyle\sum_{i=1}^n |T_{w_i\to \tilde{z_j}}(k,i\omega)|^2} && \leq |Z_j^{-1}(i\omega)| \\
&\iff \forall\omega,\ \sigma_{max}(T_{w\to\tilde{z_j}}(k,i\omega)) && \leq |Z_j^{-1}(i\omega)|.
\end{aligned}
\tag{4.2}
$$

In Chapter 1, Equation (1.3) shows how an H_∞ constraint is equivalent to a bound on the gain of a SISO systems. Equation 4.2 extends this interpretation to the case of MISO systems. Following this idea, the SS problem can be interpreted as a CSP involving semi infinite constraints:

Problem 4.1.2.

$$
\left\{\begin{array}{ll}
||T_{w\to z_j}(k)||_\infty \leq 1 & \forall j \in \{1,\dots,n_z\}, \\
s.\ t. & K(k) \textit{ ensures internal stability }, \\
k \in \mathbb{K}. &
\end{array}\right.
$$

$$
\Longleftrightarrow
$$

$$
\left\{\begin{array}{ll}
\sigma_{max}(T_{w\to\tilde{z_j}}(k,i\omega)) \leq |Z_j^{-1}(i\omega)|, & \forall j \in \{1,\dots,n_z\},\ \forall\omega \geq 0, \\
s.\ t. & K(k) \textit{ ensures internal stability }, \\
k \in \mathbb{K}. &
\end{array}\right.
$$

Actually Problem 4.1.2 can be considered as a closer formulation of the design objectives than the one given by Problem 4.1.1. Indeed, when designing the frequency templates $|Z_j|$ the user reasons in terms of bounds on the frequency response, that is having the equivalence of Equation (4.2) in mind. As a consequence, Problem 4.1.2 can be viewed as the direct expression of the user specifications as a CSP, whereas Problem 4.1.1 re-framed those n_z frequency constraints as a single objective being $||T_{w\to z}||_\infty \leq 1$, and introduced conservatism due to the inequality given by Equation (4.3),

$$
||T_{w\to z_j}||_\infty \leq ||T_{w\to z}||_\infty,\ \forall i \in \{1,\dots,n_z\}. \tag{4.3}
$$

In the end, the formulation of the H_∞ objectives considered in this thesis is given by CSP 4.1.2. This formulation is closer to the design objectives, and provides an explicit formula for the H_∞ norms. CSP 4.1.2 can be also formulated as a minimization problem having as objective function the maxima of the $||T_{w\to z_j}||_\infty$,

Problem 4.1.3.

$$
\left\{\begin{array}{ll}
\min\limits_{k\in\mathbb{K}} & \max\limits_j\,(||T_{w\to z_j}(k)||_\infty), \\
s.\ t. & K(k) \textit{ ensures internal stability.}
\end{array}\right.
$$

Denoting $f(k,\omega) = \max_j f_j(k,\omega)$, we have

$$\begin{aligned} \max_j \left(||T_{w\to z_j(k)}||_\infty\right) &= \max_j \left(\sup_{\omega\geq 0} f_j(k,\omega)\right) \\ &= \sup_{\omega\geq 0} \left(\max_j f_j(k,\omega)\right) \\ &= \sup_{\omega\geq 0} f(k,\omega). \end{aligned} \tag{4.4}$$

Thanks to Equation (4.4), Problem 4.1.3 is equivalent to Problem 4.1.4.

Problem 4.1.4.

$$\begin{cases} \min\limits_{k\in\mathbb{K}} & \sup\limits_{\omega\geq 0} f(k,i\omega), \\ s.\ t. & K(k) \textit{ ensures internal stability.} \end{cases}$$

f is an explicit function, for which an inclusion function can be defined. Using SIBBA introduced in Chapter 3, it is possible to compute an enclosure of $\sup\limits_{\omega\geq 0} f(k,i\omega)$ over a set of controllers $\{K(k),\, k\in\mathbf{k}\}$ with $\mathbf{k}\subset\mathbb{IR}^{n_k}$, and by extension bounds on the H_∞ norm of a MISO system. Finally, the H_∞ objective is formulated as a minimax problem.

Remark. In this Subsection, the initial mono-objective SS Problem 4.1.1 is reframed as an explicit multi-objective problem, by considering every performance channel $T_{w\to z_j}$ independently. This explicit formulation enables to use the global optimization algorithm developed in Chapter 4. It is also possible to consider each channel $T_{w_i\to z_j}$ as objective, which is SISO system, and has explicit formulation of its H_∞ norm.

4.1.2 Internal stability

This Subsection is based on the sections of Chapter 1 dedicated to the stability and internal stability. The internal stability can be expressed as explicit inequality constraints thanks to the Hurwitz stability criterion. We considered two cases that will be met in the sequel:

- either the general regulation scheme being the interconnection of P with $K(k)$ is consider, and P is given by its state space realization,
- or the system to control G and the weighting functions W and Z are given as transfer functions.

In the first case, P is given by its state space realization,

$$P = \left(\begin{array}{c|cc} A_P & B_w & B_u \\ \hline C_z & D_{wz} & D_{uz} \\ C_y & D_{wy} & D_{uy} \end{array}\right).$$

Then, the state space realization of $K(k)$ is also considered,

$$K = \left(\begin{array}{c|c} A_K(k) & B_K(k) \\ \hline C_K(k) & D_K(k) \end{array}\right).$$

As a consequence, the interconnection of P with K is given by

$$\left(\begin{array}{c|c} A_{cl}(k) & B_{cl}(k) \\ \hline C_{cl}(k) & D_{cl}(k) \end{array}\right),$$

where

$$A_{cl}(k) = \begin{pmatrix} A_P + B_w(I - D_K(k)D_{uy})^{-1}D_K(k)C_y, & B_u(I - D_K(k)D_{uy})^{-1}C_K(k) \\ B_K(k)(I - D_{uy}D_K(k))^{-1}C_z, & A_K(k) + B_K(k)(I - D_{uy}D_K(k))^{-1}D_{uy}C_K(k) \end{pmatrix}. \tag{4.5}$$

Hence, the coefficients of $A_{cl}(k)$ depend on k.

Remark. Computing $A_{cl}(k)$ requires to inverse a symbolic matrix if $D_K(k)$ and D_{uy} are not null. This point and the related issues are discussed in detail in the conclusion. For the examples considered further, the matrices are computed with the Matlab symbolic toolbox.

Let n_l be the dimension of A_{cl}, and $Q(k,s) = \sum_{i=0}^{n_l} \alpha_i(k)s^i$ be the characteristic polynomial of A_{cl}, the coefficients of which depends on k. The Hurwitz criterion applied to $Q(k,s)$ enables to express the stability of A_{cl} as n_l inequalities $R_l(k) \leq 0$, with R_l multivariate polynomials in the $\alpha_i(k)$.

Remark. In order to ensure the internal stability, it suffices to prove that the interconnection of K with P deprived from the weighting functions is stable. Indeed, the weighting functions W and Z are virtual and are needed only to express the H_∞ objectives. However, in the benchmark examples considered in the sequel, only P is provided.

Consider now the second case where G and the weighting functions W and Z are given by transfer matrices. Suppose that the interconnection of K with G can be rearranged as the internal stability scheme represented in Figure 4.1. The system is internally stable if the

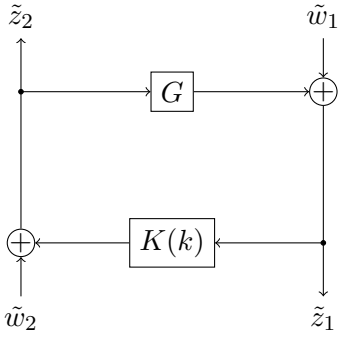


Figure 4.1: Internal stability generic scheme.

transfers from $(\tilde{w}_1, \tilde{w}_2)^T$ to $(\tilde{z}_1, \tilde{z}_2)^T$ are all stable. These transfers are the elements of the transfer matrix

$$\begin{pmatrix} (I - K(k,s)G(s))^{-1} & K(k,s)(I - G(s)K(k,s))^{-1} \\ G(s)(I - G(s)K(k,s))^{-1} & (I - G(s)K(k,s))^{-1} \end{pmatrix}. \tag{4.6}$$

Using the Hurwitz criterion on the denominator of each transfers allows to express the internal stability as a set of polynomial inequalities which coefficients depend on k.

Remark. Obtaining the polynomial inequalities can be automatized with any symbolic toolbox, since it implies to compute the determinants of symbolic matrices. The computation of the expression of the determinants is very fast, but simplifying the expressions can take some times according to the size of the expression. Simplifying the expressions is not necessary but helps reducing the number of occurrence of the variables, and hence reduces evaluation pessimism of inclusion functions (see Chapter 2). This may help to improve the convergence time of MMIBBA.

4.1.3 Explicit formulation of the structured synthesis problem

Considering each performance output independently, the SS problem is reformulated as the Independent Outputs Structured Synthesis (IOSS) problem,

Problem 4.1.5. *IOSS:*

$$\begin{cases} \min\limits_{K} & \max\limits_{j} ||T_{w\to z_j}||_\infty, \\ s.\ t. & K \textit{ ensures internal stability }. \end{cases}$$

The IOSS problem is equivalent to Problem 4.1.6, thanks to the Hurwitz criterion and Equation (4.4).

Problem 4.1.6.

$$\begin{cases} \min\limits_{k\in\mathbb{K}} & \sup\limits_{\omega\geq 0} f(k, i\omega) \\ s.\ t. & R_l(k) \leq 0 \,\forall l \in \{1, \ldots, n_l\}. \end{cases}$$

Problem 4.1.6 is a minmax problem as introduced in Chapter 3. Both the objective f and the constraints have explicit formulas, and therefore the IOSS problem can be solved with MMIBBA, the branch and bound algorithm developed in Chapter 3.

Remark. The range of ω, which should be $[0, \infty[$, must be limited to a bounded set in order to run MMIBBA. This approximation is generally sufficient to compute the H_∞ norm, provided that the bounded domain of pulsations is large enough. This domain is denoted by Ω, and will be systematically indicated in the sequel.

4.1.4 Illustrative example

In order to illustrate how the structured synthesis problem can be reformulated as in the form of Problem 4.1.6, consider the system given on Figure 4.2. The problem consists in finding the controller K which ensures the internal stability, such that the three transfers from the external input r to the performance outputs z_1, z_2 and z_3 have their H_∞ norms lower than one. In this case, $w = \tilde{w} = r$, $\tilde{z}_1 = e$, $\tilde{z}_2 = u$, and $\tilde{z}_3 = y$. The system G is given by its transfer function

$$G(s) = \frac{1}{s^2 + 1.4s + 1}, \tag{4.7}$$

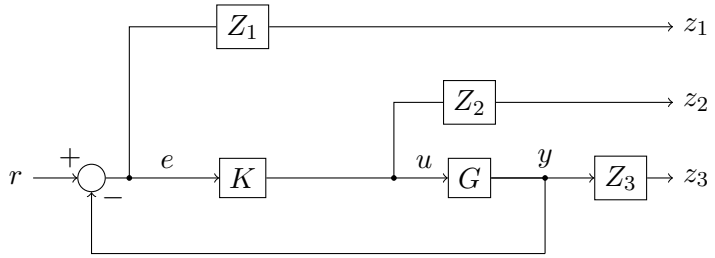


Figure 4.2: Mixed sensitivity problem.

and the weighting functions are

$$Z_1(s) = \tfrac{10s+100}{1000s+1}, \quad Z_2(s) = \tfrac{10s+1}{s+10}, \text{ and } \quad Z_3(s) = \tfrac{100s+1}{s+10}.$$

Their inverses appear as dotted curves in Figure 4.3.
The order of the augmented system P is equal to 5, which is the sum of the orders of G, W_1, W_2, and W_3. The controller K that is searched is a PID,

$$K(k,s) = k_p + \frac{k_i}{s} + \frac{k_d s}{1+s}.$$

The tuning variable is therefore $k = (k_p, k_i, k_d)^T$.

Remark. In the following, k is called the tuning variable and k_p, k_i and k_d are called the tuning parameters.

H_∞ objectives. Let $S(k,s) = (1+G(s)K(k,s))^{-1}$ be the sensitivity function. The transfers from r to z_1, z_2, and z_3 are given by

$$\begin{aligned}
T_{r\to z_1}(k,s) &= Z_1(s)S(k,s)\\
T_{r\to z_2}(k,s) &= Z_2(s)K(k,s)S(k,s)\\
T_{r\to z_3}(k,s) &= Z_3(s)K(k,s)S(k,s)G(s)
\end{aligned}$$

and their H_∞ norm is the maximum of their modulus since they are SISO,

$$\begin{aligned}
||T_{r\to z_1}(k)||_\infty &= \sup_\omega |Z_1(i\omega)S(k,i\omega)| &&= \sup_\omega f_1(k,\omega)\\
||T_{r\to z_2}(k)||_\infty &= \sup_\omega |Z_2(i\omega)K(k,i\omega)S(k,i\omega)| &&= \sup_\omega f_2(k,\omega)\\
||T_{r\to z_3}(k)||_\infty &= \sup_\omega |Z_3(i\omega)K(k,i\omega)S(k,i\omega)G(i\omega)| &&= \sup_\omega f_3(k,\omega),
\end{aligned}$$

with f_1, f_2, and f_3 having the explicit expressions

$$f_1(k,\omega) = \big(100\omega^2(25\omega^8 + 2524\omega^6 + 2424\omega^4 + 2425\omega^2 + 2500)\big) / \big((1000000\omega^2+1)(25k_d^2\omega^4 - 50k_dk_i\omega^2 + 50k_dk_p\omega^4 - 50k_d\omega^6 + 120k_d\omega^4 + 25k_i^2\omega^2 + 25k_i^2 - 70k_i\omega^4 - 70k_i\omega^2 + 25k_p^2\omega^4 + 25k_p^2\omega^2 - 50k_p\omega^6 + 50k_p\omega^2 + 25\omega^8 + 24\omega^6 + 24\omega^4 + 25\omega^2)\big)$$

$$f_2(k,\omega) = \big((2500\omega^6 - 75\omega^4 + 2499\omega^2 + 25)(k_d^2\omega^4 - 2k_dk_i\omega^2 + 2k_dk_p\omega^4 + k_i^2\omega^2 + k_i^2 + k_p^2\omega^4 + k_p^2\omega^2)\big) / \big((\omega^2+100)(25k_d^2\omega^4 - 50k_dk_i\omega^2 + 50k_dk_p\omega^4 - 50k_d\omega^6 + 120k_d\omega^4 + 25k_i^2\omega^2 + 25k_i^2 - 70k_i\omega^4 - 70k_i\omega^2 + 25k_p^2\omega^4 + 25k_p^2\omega^2 - 50k_p\omega^6 + 50k_p\omega^2 + 25\omega^8 + 24\omega^6 + 24\omega^4 + 25\omega^2)\big)$$

$$f_3(k,\omega) = \big(25(10000\omega^2+1)(k_d^2\omega^4 - 2k_dk_i\omega^2 + 2k_dk_p\omega^4 + k_i^2\omega^2 + k_i^2 + k_p^2\omega^4 + k_p^2\omega^2)\big) / \big((\omega^2+100)(25k_d^2\omega^4 - 50k_dk_i\omega^2 + 50k_dk_p\omega^4 - 50k_d\omega^6 + 120k_d\omega^4 + 25k_i^2\omega^2 + 25k_i^2 - 70k_i\omega^4 - 70k_i\omega^2 + 25k_p^2\omega^4 + 25k_p^2\omega^2 - 50k_p\omega^6 + 50k_p\omega^2 + 25\omega^8 + 24\omega^6 + 24\omega^4 + 25\omega^2)\big).$$

With our approach, we consider the outputs separately, that is the maxima of the three H_∞ norms is minimized. However the classical approach consists in minimizing the H_∞ norm of $F(P,K)$, given by the transfer matrix

$$T_{w\to z}(k,s) = \begin{pmatrix} Z_1(s)S(k,s) \\ Z_2(s)K(k,s)S(k,s) \\ Z_3(s)K(k,s)S(k,s)G(s) \end{pmatrix}.$$

Internal Stability The internal stability is ensured if the four transfers

$$\begin{pmatrix} (I+K(k,s)G(s))^{-1} & -K(k,s)(I+G(s)K(k,s))^{-1} \\ G(s)(I+G(s)K(k,s))^{-1} & (I+G(s)K(k,s))^{-1} \end{pmatrix}, \tag{4.8}$$

are stable. Since G and K are SISO, $(I+K(k,s)G(s))^{-1} = (I+G(s)K(k,s))^{-1}$, reducing the number of transfer which stability has to be verified to three. Note that the sign of K is different between Equation (4.8) and Equation (4.6), since a positive feedback action is considered for the generic regulation scheme to ensure internal stability, depicted in Figure 4.1, and in this example a negative feedback action is considered. The three transfers have the same denominator being

$$Q(k,s) = 5s^4 + 12s^3 + (5k_d + 5k_p + 12)s^2 + (5k_i + 5k_p + 5)s + 5k_i.$$

Since the leading coefficient of $Q(k,s)$ is strictly positive, the Hurwitz criterion is directly applied to $Q(k,s)$ (see Chapter 1), and it provides the set of stability constraints:

$$\begin{aligned} &12 && > 0 \\ &R_1(k) = 60k_d - 25k_i + 35k_p + 119 && > 0 \\ &R_2(k) = 300k_d - 250k_i + 770k_p + 300k_dk_i + 300k_dk_p + 50k_ik_p - 125k_i^2 + 175k_p^2 + 595 && > 0 \\ &R_3(k) = 25k_i(60k_d - 50k_i + 154k_p + 60k_dk_i + 60k_dk_p + 10k_ik_p - 25k_i^2 + 35k_p^2 + 119) && > 0. \end{aligned}$$

The first constraint is ignored in the following.

In the end, with the approach proposed in this chapter, the mixed sensitivity problem is formulated as

Problem 4.1.7.

$$\begin{cases} \min\limits_{k} & \sup\limits_{\omega} \left(\max\left(f_1(k,\omega), f_2(k,\omega), f_3(k,\omega)\right)\right) \\ s.\ t. & R_1(k) > 0 \\ & R_2(k) > 0 \\ & R_3(k) > 0. \end{cases}$$

Results and comparisons.
Problem 4.1.7 is solved using MMIBBA, introduced in Chapter 3, with the following parameters:

- The search domain of k is $\mathbb{K} = [-10, 10]^3$,
- The search domain of ω is limited to the bounded set $\Omega = [10^{-3}, 10^3]$,
- The bisection precision on k is 10^{-4},
- The bisection precision on ω is 10^{-5},
- The number of bisection done by SIBBA to compute bounds on the objective function is 5,
- The relative precision of the enclosure is set to 10^{-2}.

MMIBBA converges in 125 seconds, and provides $[0.978, 0.988]$ as an enclosure of the minimum of Problem 4.1.7. The best solution found is $k_{go} = (0.0348, 0.0993, 0.0625)$, and by extension the PID controller

$$K_{go}(s) = 0.0348 + \frac{0.0993}{s} + \frac{0.0625s}{1+s}.$$

Consequently, we have $\sup\limits_{\omega\in\Omega} \max\limits_{i} \left(f_i(k_{go}, \omega)\right) = 0.988$. Using Matlab's `norm` function, we obtain $\max\limits_{i}(||T_{w\to z_j}(k_{go})||_\infty) = 0.9982$, which is greater than the initial result provided by MMIBBA, but still lower than one ensuring that the H_∞ constraints $||T_{w\to z_j}(k_{go})||_\infty \leq 1$ are respected.

We propose to compare k_{go} with the solutions provided by the classical full order and the Matlab Hinfstruct toolbox to the classical formulation of the synthesis problem given by Problem 4.1.1, having $||T_{w\to z}||_\infty$ has objective. The Matlab function `hinfsyn` [47] is used to compute the full order controller K_{full} solution to Problem 4.1.1 in its convex form, using the LMI formulation. The order of K_{full}, equal to the one of P, is 5. The local optimization algorithm [9] of Hinfstruct toolbox is run 700 times to compute a PID, with different initial points in $[-10, 10]^3$ so that the computation time is equal to that of MMIBBA. These two methods are denoted by LMI full and Hinfstruct in Table 4.1 that presents the results. In addition, we also solve Problem 4.1.7 with the Matlab Systune toolbox based on local optimization. As a consequence, Systune and MMIBBA solve the same problem.

The H_∞ norm of $T_{w\to z}$ appears in the third column of Table 4.1, and the value of the maximum of the H_∞ norms of the three objectives are listed in the last column. The optimization objectives of the three synthesis methods are indicated by the gray cells. Hinfstruct

Method	CPU (s)	$\|\|T_{w\to z}\|\|_\infty$	$\max_i(\|\|T_{w\to z_j}\|\|_\infty)$
LMI full	2	1.0202	1.0161
Hinfstruct	118	1.0411	1.0411
Systune	122	1.0986	0.9912
MMIBBA	125	1.0801	[0.978,0.9982]

Table 4.1: Results obtained with the three different synthesis methods.

provides the solution $k_{struct} = (0.0305, 0.0969, 0.0736)$, and the PID controller is

$$K_{struct}(s) = 0.0305 + \frac{0.0969}{s} + \frac{0.0736s}{1+s}.$$

The LMI Full method gives a result rapidly due to the convexity of the problem. Considering the $||T_{w\to z}||_\infty$ criterion, LMI Full provides a better result than Hinfstruct probably because K_{full} has a higher order (K_{full} is order 5 and K_{sys} is order 2). MMIBBA and Systune show the worst results, but these methods minimize $\max_i(||T_{w\to z_j}||_\infty)$ instead of $||T_{w\to z}||_\infty$, on the contrary to the other two methods. However, considering the $\max_i(||T_{w\to z_j}||_\infty)$ objective, one can remark that MMIBBA and Systune are able to compute a solution lower than 1, *i.e.* to provide a controller that respects all the three H_∞ constraints $||T_{w\to z_i}(k)||_\infty \leq 1, i \in \{1,2,3\}$. This results is particularly interesting, since it emphasizes the fact that minimizing $\max_i(||T_{w\to z_j}||_\infty)$ and $||T_{w\to z}||_\infty$ are two different problems as discussed in subsection 4.1.1.
The bode diagrams of the three objective (non weighted) channels, as well as the frequency templates Z_i^{-1} are displayed on Figure 4.3. As expected from the H_∞ norm values of the weighted channels, that are close to one, the bode plots are also close to the Z_i^{-1}'s templates.

Remark. Obtaining the symbolic expression of the H_∞ norm of the three objective channel and the Hurwitz polynomials is done in less than one second with the Matlab's symbolic toolbox.

4.1.5 Benchmark examples

MMIBBA is now tested on several problems taken from COMPl$_e$ib library [81]. Table 4.2 lists the four examples we propose to solve. n_x denotes the number of states of the system, n_w the number of external inputs, n_u the number of control inputs, n_z the number of performance outputs and n_y the number of measurement outputs. For each example, the state space realization of P is provided. We propose to synthesize static output feedback controllers. Therefore, the controllers $K(k)$ have as state space realizations empty matrices A_K, B_K and C_K, and D_K is parametrized by k,

$$K(k) = D_K(k) = \begin{pmatrix} k_{11} & \dots & k_{1n_y} \\ \vdots & \ddots & \vdots \\ k_{n_u 1} & \dots & k_{n_u n_y} \end{pmatrix}.$$

The four examples have been chosen such that the order of P is small, limiting the size of the expressions of the Routh coefficients and the H_∞ norms, and also such that the product n_y

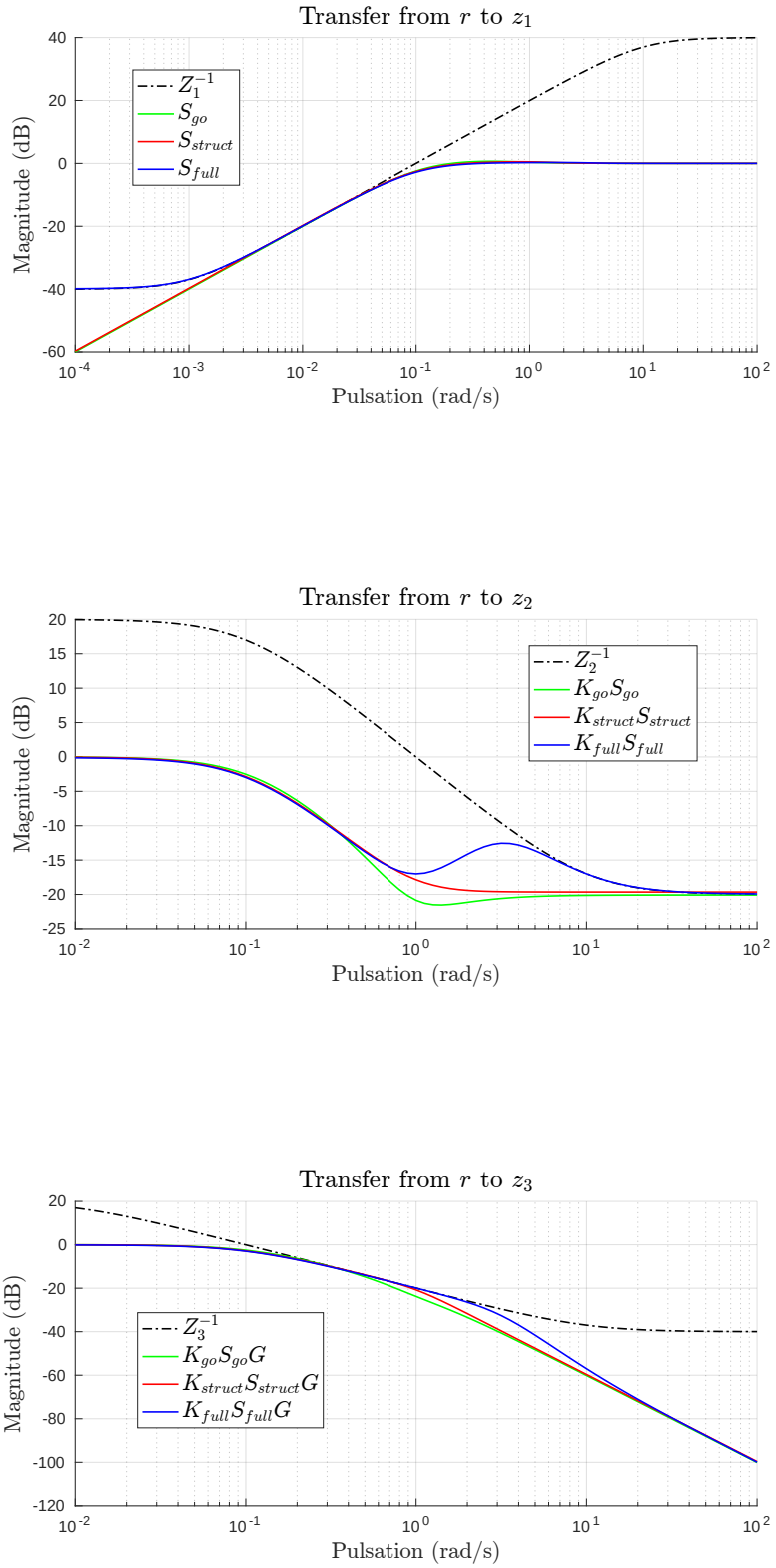


Figure 4.3: Frequency specifications, displayed in dotted, and closed loop objective transfers for the three controllers.

times n_u is small, limiting the number of controllers parameters.
For every problem, we set MMIBBA parameters as follows.

- The search domain of k is $\mathbb{K} = [-10, 10]^{n_u \times n_y}$,
- the search domain of ω is limited to the bounded set $\Omega = [10^{-3}, 10^3]$,
- the bisection precision on k is 10^{-4},
- the bisection precision on ω is 10^{-5},
- the number of bisection done by SIBBA (`iter` parameter) to compute bounds on the objective function is 5,
- the relative precision of the enclosure is set to 10^{-2}.

Problem	n_x	n_w	n_u	n_z	n_y	K	Ω
AC4	4	2	1	2	2	$[-10, 10]^2$	$[10^{-3}, 10^3]$
AC7	9	4	1	1	2	$[-10, 10]^2$	$[10^{-3}, 10^3]$
AC17	4	4	1	4	2	$[-10, 10]^2$	$[10^{-3}, 10^3]$
HE1	4	2	2	2	1	$[-10, 10]^2$	$[10^{-3}, 10^3]$

Table 4.2: COMPl$_e$ib problems.

The results obtained with MMIBBA are compared with those obtained with the Systune Matlab Toolbox [9] in the Table 4.3. Both MMIBBA and Systune solve the IOSS problem, and therefore we have the same objective function being $\max_i ||T_{w \to z_j}||_\infty$. The second column indicates the minimum obtained with Systune, and the third column the enclosure of the minimum obtained with MMIBBA. The best solution found by MMIBBA is the upper bound of the enclosure.

Results

Although the Systune solver is based on local optimization, it succeeds in finding a close approximation of the global minimum. On those four examples, MMIBBA does not provide a better solution than Systune, but it gives a lower bound on the minimum. This lower bound provides two results. First, it proves that the best solution found by MMIBBA cannot be lowered by more than ϵ, and indicates the quality of the solution found. Secondly, the lower bound can prove that CSP (4.1.8) is not feasible over $\mathbb{K}$. For example, it is proved that AC17 is not feasible over $\mathbb{K}$ since the lower bound on the minimum is greater than one.

Feasible set characterization

Beside solving the minimization problem, we also propose to characterize the sub-optimal feasible set for those four problems,*i.e.* to characterize by a sub-paving the feasible set of the CSP given by

Problem	$\max_{i\in\{1,\dots,n_z\}} \|\|T_{w\to z_j}\|\|_\infty$		ϵ	α	cpu time (s)
	Systune	MMIBBA			
AC4	0.914	[0.905,0.916]	1e-2	1e-4	26
AC7	0.065	[0.056,0.066]	1e-2	1e-4	11
AC17	5.35	[5.23,5.35]	1e-2	1e-4	17
HE1	0.152	[0.142,0.152]	1e-2	1e-4	185

Table 4.3: MMIBBA benchmark results.

Problem 4.1.8.

$$\begin{cases} \max_j \|\|T_{w\to z_j}\|\|_\infty \leq 1, \\ K \textit{ ensures internal stability.} \end{cases}$$

This CSP is reformulated as

$$\begin{cases} f_j(k,\omega) \leq 1,\ \forall \omega \in \Omega,\ \forall j \in \{1,\dots,n_z\} \\ R_l(k) > 0,\ \forall \{1,\dots,n_z\}, \end{cases}$$

so that it is proper to be solved with FSCA. Note that this CSP involves both quantified constraints and regular constraints.
FSCA is run on the four problems to solve CSP (4.1.8), with the following parameters :

- the search domain of k is $\mathbb{K} = [-10,10]^{n_u \times n_y}$,
- the search domain of ω is limited to the bounded set $\Omega = [10^{-3}, 10^3]$,
- the bisection precision on k is 10^{-1},
- the bisection precision on ω is 10^{-2},
- the number of bisection done by SIBBA (`iter` parameter) to compute bounds on the objective function is 5.

Table 4.4 indicates the computation time of FSCA to solve the problems. The sub-paving of problems AC7 and HE1 are represented on Figures 4.4 and 4.5 respectively. The infeasible boxes are displayed in gray, the feasible boxes in dark gray and the boxes not proved to be feasible or infeasible in light gray.

FSCA results

One can remark that the computation time of FSCA for the problem AC17 is lower than the one of MMIBBA. This is due to the fact that the minimum of problem AC17 is greater than one, making easy to prove that CSP (4.1.8) has no feasible solution. Concerning problem AC4, FSCA is not able to prove any box to belong to $\mathbb{S}_{true}$ for the given bisection precisions on k and ω, but converge rapidly to two boxes $[-0.36,-0.313]\times[-0.085,-0.07]$ and $[-0.313,-0.256]\times[-0.085,-0.07]$ for which nothing can be proved. On the contrary, the computation time of FSCA is greater than MMIBBA on problems AC7 and HE1. This is due to the fact that

Problem	cpu time (s)
AC4	22
AC7	702
AC17	< 1
HE1	3212

Table 4.4: FSCA benchmark time results.

FSCA has to characterize $\mathbb{K}$ entirely between the three sub-pavings instead of focusing only on the regions of $\mathbb{K}$ that can contain the solution to the minimization problem as MMIBBA does.
On Figures 4.4 and 4.5, one can see that the set of unproved boxes can be large. This is known as the *clustering effect*, and illustrates the pessimism of the enclosure provided by SIBBA, discussed in Chapter 3. From Figures 4.4 and 4.5, it can be inferred that the feasible set of CSP 4.1.8 is a convex set for problems AC17 and HE1. However, nothing could indicate a priori that the sets of feasible points are convex.

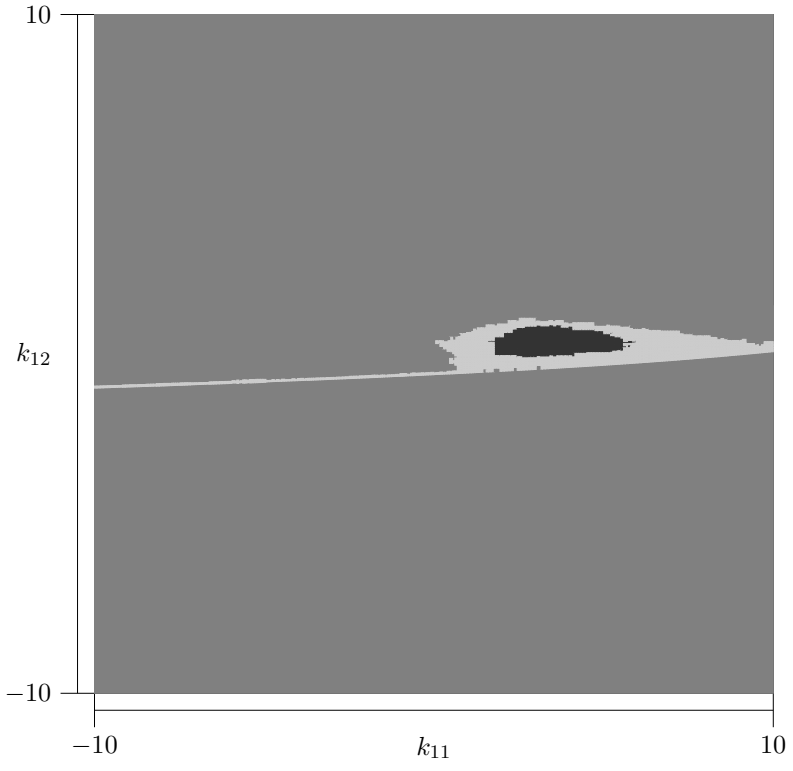


Figure 4.4: Characterization of feasible tunable parameters for the AC7 problem.

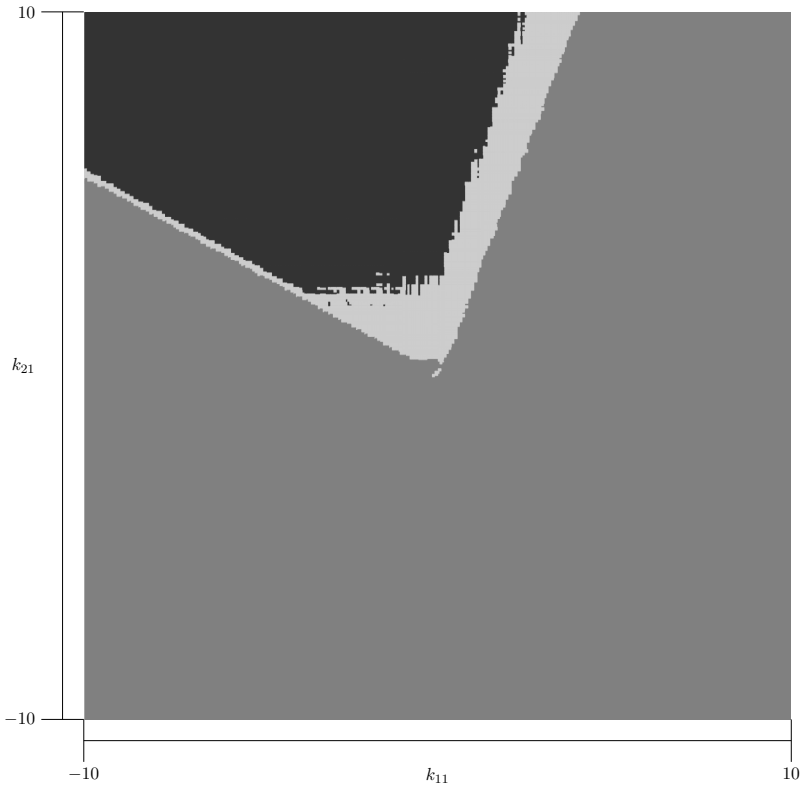


Figure 4.5: Characterization of feasible tunable parameters for the HE1 problem.

4.1.6 Discussion

This section presents how the structured H_∞ synthesis problem, by considering each performance outputs z_j separately, can be made in as an explicit minmax problem thanks to Theorem 4.1.1. If this reformulation is not the classical way to consider the H_∞ synthesis, it is close to the design objectives as shown with the illustrative example. It can be remarked that, with our problem formulation, the frequency templates can be any function of ω. Indeed, $f_j(\omega) \leq b(\omega)$, $\forall \omega \in \Omega$ with b a scalar valued function being the frequency template, is equivalent to $f_j(\omega) \cdot \dfrac{1}{b(\omega)} \leq 1$, $\forall \omega \in \Omega$ which can be set as an H_∞ constraint, since it has an explicit expression. This constraint ensures that $\sigma_{max}(T_{w \to z_j}(i\omega)) \leq b(\omega)$. To this extent, the global optimization approach makes it possible to define non conventional frequency templates, contrary to the existing approaches that require Z_j to be an LTI system, and by extension requires $|Z_j|$ to be a fraction of two polynomials in ω.

4.2 Solution to robustness analysis

Consider an uncertain LTI system G depending on an uncertain parameter $\theta \in \Theta$, Θ being the set of possible values. Suppose that a controller K has been computed for a nominal system

G from H_∞ criteria. The question is to decide if the design requirements, in our case the stability of the close loop and respect of H_∞ constraints, are met for all the possible values of θ. This is equivalent to verify the satisfaction of semi-infinite constraints, which can be reformulated as a maximization problem as shown in Chapter 3. The maximum corresponds to the *worst case* over the uncertainties.
The robustness analysis problem is divided in two parts :

- the Robust Stability Analysis (RSA), and
- the Robust H_∞ Analysis (RHA).

It will be shown that both problems are straightforward to express under explicit programs from Subsections 4.1.1 and 4.1.2.

4.2.1 Robust stability analysis

The robust stability analysis problem is expressed in a general form as the semi infinite constraint :

Problem 4.2.1. *(RSA problem):*

decide if K ensures internal stability , $\forall \theta \in \Theta$.

Like in Subsection 4.1.2, we shall consider two cases. $G(\theta)$ has a state space expression, and its state matrix $A_G(\theta)$ also depends on the uncertain variable, or $G(\theta)$ is represented by a transfer matrix, the coefficients of the polynomials of the transfers depending in θ. It can be noticed that those two cases correspond exactly to the ones considered in Subsection 4.1.2, with the only difference that now $G(\theta)$ is parametrized by θ instead of K being parametrized by k. Consequently, all the reasonings that have already been followed can be extended to the present case, and in the end the internal stability can be expressed as polynomial inequalities $R_l(\theta) \leq 0$ which coefficients depend on θ. Consequently, the RSA problem is equivalent to Problem 4.2.2

Problem 4.2.2.

$\left\{ \right.$ *decide if* $R_l(\theta) \leq 0$, $\forall \theta \in \Theta$, $\forall l \in \{1, \ldots, n_l\}$.

This problem can be handled by solving a maximization problem, the objective being the maximum of the R_l polynomials:

Problem 4.2.3.

$$\left\{ \sup_{\theta \in \Theta} \max_{l} (R_l(\theta)). \right.$$

If the maximum of Problem 4.2.3 is strictly greater than 0, it proves that there exists at least one polynomial R_l and one θ such that $R_l(\theta) > 0$. On the contrary, if the maximum is lower than 0, it proves that L is stable for any value of θ and thus robustly stable. The main point is that the global maximum must be computed, and Problem 4.2.3 is non convex. To this extent, IBBA presented in Chapter 2 can be used to compute an enclosure of the maximum. If the stability analysis concludes that the system is not stable, one can consider using FSCA in order to characterize the subset of Θ such that the system is stable in order to provide further insights. This set corresponds to the feasible set of CSP 4.2.4.

Problem 4.2.4.

$$\begin{cases} \max_l (R_l(\theta)) \leq 0, \\ \theta \in \Theta. \end{cases}$$

Example.

As an example, consider the problem taken from [68] consisting in studying the stability of the uncertain LTI system L depending on $\theta = (\theta_1, \theta_2)^T$, $\theta \in [-10, 10]^2$, which characteristic polynomial is

$$Q(\theta, s) = s^3 + \sin(\theta_1\theta_2)s^2 + \theta_1^2 s + \theta_1\theta_2.$$

Using IBBA to solve the maximization Problem 4.2.3, the enclosure provided is $[100, 100]$ meaning that L is not stable for all $\theta \in [-10, 10]^2$. In order to know which values of θ lead to the instability of L, FSCA is run and providse the sub-paving shown in Figure 4.6.

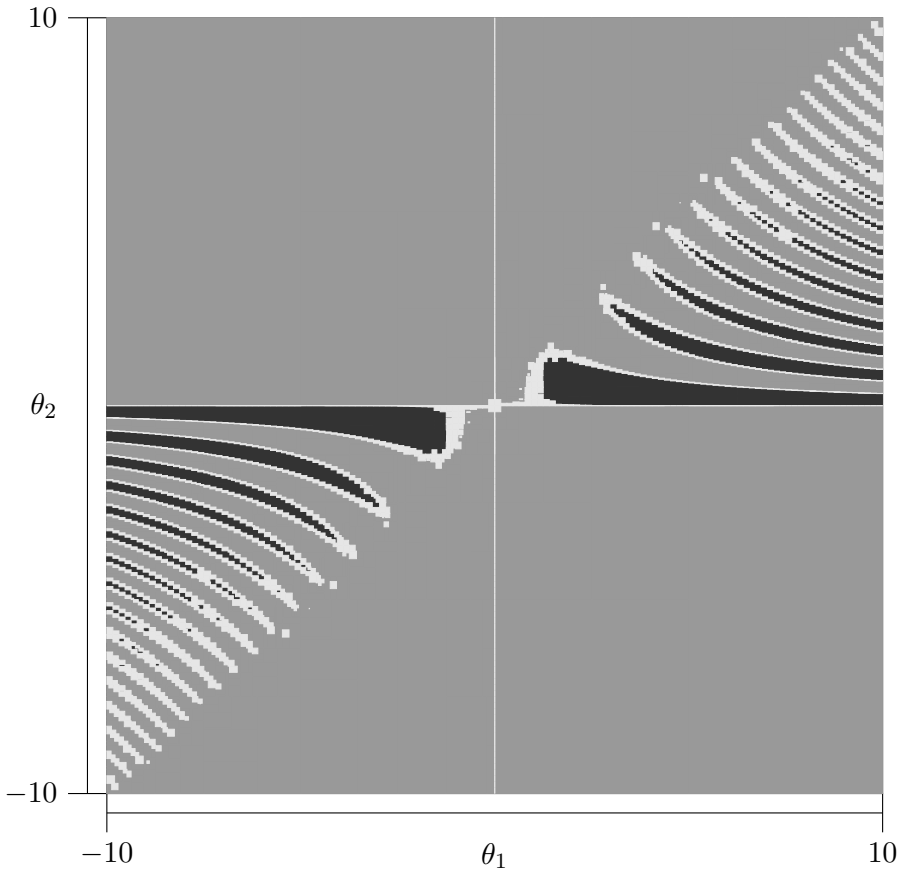


Figure 4.6: Characterization of the stable set of the uncertain parameters.

The stable set is displayed in dark gray, and the set of unstable parameters appears in gray. Nothing is proved for the light gray set. Note that the stable set is clearly not convex, not even connected, and emphasized the relevance of a non convex approach of the RSA problem.

4.2.2 Robust H_∞ analysis

In order to study the robust performance, we consider the transfer matrix $T_{w\to z}$ from the external inputs to the performance outputs. Following the outputs independence approach of the H_∞ criteria, the RHA consists in verifying that the H_∞ norms of the transfers from w to the outputs z_j are lower than one for all possible values of θ. Since $G(\theta)$ depends in θ, we also have that $T_{w\to z}(\theta)$ depends in this variable. The RHA is formulated as the satisfaction of n_z constraints

Problem 4.2.5. *RHA problem:*

$$\text{decide if } ||T_{w\to z_j}(\theta)||_\infty \leq 0, \forall j \in \{1,\dots,n_z\},\ \forall \theta \in \Theta.$$

As for the stability analysis, the related maximization problem having the maximum of the H_∞ norm as objective can be considered to decide the RHA problem. To this extent, the function f which here depends on the parameter θ instead of k can be used as the objective function of a maximization problem to ensure the robust performance. Indeed, we have

$$\begin{aligned} ||T_{w\to z_j}(\theta)||_\infty \leq 0, \forall j \in \{1,\dots,n_z\},\ \forall \theta \in \Theta \quad &\iff \sup_{\omega\geq 0} f(\theta,\omega) \leq 0,\ \forall \theta \in \Theta \\ &\iff \sup_{\theta\in\Theta}\sup_{\omega\geq 0} f(\theta,\omega) \leq 0 \\ &\iff \sup_{\theta\in\Theta,\omega\geq 0} f(\theta,\omega) \leq 0. \end{aligned} \tag{4.9}$$

In the end, the maximization problem is given by Problem 4.2.6.

Problem 4.2.6.

$$\begin{cases} \sup\limits_{\theta\in\Theta,\omega\geq 0} & f(\theta,\omega). \end{cases}$$

If the global maximum of Problem 4.2.6 is lower than one, then the system is robust to uncertainties with respect to the H_∞ objectives. On the contrary, if the maximum is strictly greater than one, then L is not robust and in this case FSCA can be used to characterize the set of uncertainty such that $L(\theta)$ respects the H_∞ constraints $||T_{w\to z_j}(\theta)||_\infty \leq 1$. This set corresponds to the feasible set of CSP 4.2.7, which has one semi infinite constraint.

Problem 4.2.7.

$$\begin{cases} f(\theta,\omega) \leq 1,\ \forall \omega \geq 0, \\ \theta \in \Theta. \end{cases}$$

The maximum of f corresponds actually to the worst case among the f_j due to its max formulation, the worst case over the uncertainties and the worst case over the pulsations. The fact that the H_∞ norm is a maximum over the pulsations, that is a worst case, enables to easily adopt this paradigm for the other parameters of the problem being the uncertain parameters and j the number of the output channels. From a mathematical point of view, this worst case formulation implies to commute the priority oder of the variables when computing

the wort-case, and provides interpretations in term of H_∞ norms.

$$\begin{array}{ll} \sup_\theta \sup_\omega \max_j f_j(\theta,\omega) & = \max_j \sup_\theta \sup_\omega f_j(\theta,\omega) \\ & = \sup_\theta \max_j \sup_\omega f_j(\theta,\omega) \\ & \iff \\ \sup_\theta \sup_\omega \max_j f_j(\theta,\omega) & = \max_j \sup_\theta ||T_{w\to z_j}(\theta)||_\infty \\ & = \sup_\theta \max_j ||T_{w\to z_j}(\theta)||_\infty \end{array}$$

In conclusion, the maximum of Problem 4.2.6 is the worst-case among the H_∞ objectives and over Θ, but the function f is used to adopt the formalism introduced in the optimization part. Note that the RHA can be performed for every H_∞ objectives independently by calculating

$$\sup_{\theta\in\Theta} ||T_{w\to z_j}(\theta)||_\infty.$$

This analysis provides more details since the wort-case value of the uncertainty may be different with respect to the H_∞ objective considered.

Example

In order to illustrate the RHA, consider the uncertain MISO system having as inputs (w_1, w_2) and one output z_1, composed of two second order systems which damping ratio ξ_1 and ξ_2 depend on an uncertain parameter.

$$\begin{array}{l} T_{w\to z_1}(s) = \begin{pmatrix} T_{w_1\to z_1}(s) & T_{w_2\to z_1}(s) \end{pmatrix} \\ T_{w_1\to z_1}(s) = 0.5\dfrac{\omega_1^2}{s^2+2\xi_1\omega_1 s+\omega_1^2} \qquad T_{w_2\to z_1}(s) = 0.5\dfrac{\omega_2^2}{s^2+2\xi_2\omega_2 s+\omega_2^2} \end{array}$$

with

$$\begin{array}{ll} \xi_1 = 0.2+\sin(\theta_1\theta_2)^2 & \xi_2 = 0.2+\theta_1^2+\theta_2^2 \\ \omega_1 = 1 & \omega_2 = 10 \end{array}$$

The objective function f of Problem 4.2.6 for this example simply corresponds to f_1,

$$f_1(\theta,\omega)\sqrt{|T_{w_1\to z_1}(s)|^2+|T_{w_2\to z_1}(s)|^2}.$$

Using IBBA to solve Problem 4.2.6 over the domain $[-4,4]^2\times[10^-2,10^2]$, the best solution obtained is

$$(\tilde{\theta_1},\tilde{\theta_2},\tilde{w}) = (-0.006, 0.002, 0.960)$$

and the enclosure of the global maximum is $[1.8040, 1.8315]$. IBBA takes 5 seconds to terminate on this example. The worst case value of the uncertainty computed is $\tilde{\theta} = (-0.006, 0.002)$. In order to determine the values of θ so that $||T_{w\to z_1}||_\infty$ is lower than one, FSCA is run to characterize the feasible set of QCSP 4.2.7. The initial domain is the same as previously, the minimum bisection precision on θ is set to 0.1. Figure 4.7 is obtained in 85 seconds.

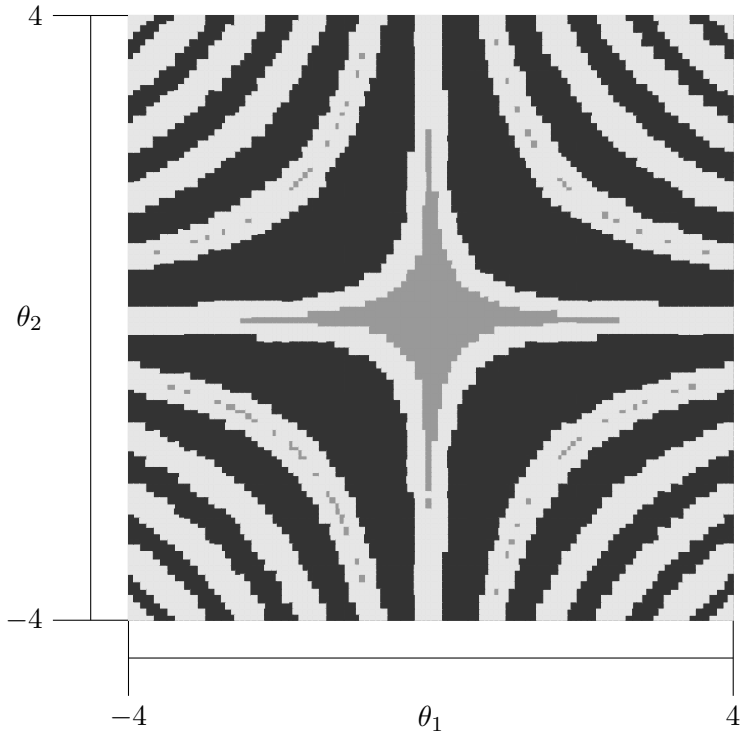


Figure 4.7: Set of parameters for which $||T_{w\to z_1}||_\infty$ is lower than 1, in dark gray, and strictly greater than 1, in gray. Nothing is proved for the light gray set.

4.2.3 Discussion

This section shows how the robustness analysis can be formulated as two explicit problems, being the stability and H_∞ analysis. Both can be formulated as maximization problems that must be solved globally in order to prove the robustness in a guaranteed way. These problems are actually reformulations of SICs involving the quantified variable θ. Solving the maximization problems with IBBA corresponds to perform a *worst case* analysis, the best solution returned by IBBA corresponding to the worst case. Concerning the stability analysis, this worst case among uncertainty is not very relevant since the Hurwitz polynomials provide conditions for stability but it is not a relevant metric for stability. Indeed, there is no direct link between the values of Hurwitz polynomials and the distance to instability, that is the distance between the real part of the poles and the imaginary axis. On the contrary, considering the H_∞ criterion, having the worst case is relevant since, contrary to the stability criterion, the computation of the worst-case H_∞ norm is performed directly and not reformulated as another equivalent problem.
The two examples proposed in this section illustrate that a global optimization approach enable to perform robustness analysis for systems that have non convex dependencies in the uncertain variable, without the need of approximating the initial set of uncertain systems by uncertain polytopic systems or the Δ-structure interconnection to perform the analysis. In order to illustrate the conservatism of the Δ-structure representation of the uncertainty,

consider the example taken from [133, p. 133]. The uncertain LTI system L depends on uncertain parameters $\theta \in [-1,1]^2$

$$L(\theta,s) = \frac{10((2+0.2\theta_1)s^2 + (2+0.3\theta_1+0.4\theta_2)s + (1+0.2\theta_2))}{(s^2+0.5s+1)(x^2+2s+3)(s^2+3s+6)}.$$

The set of uncertain systems is approximated as the nominal system $\tilde{L}$ plus a weighted additive uncertainty Δ,

$$L(\theta,s) \in L_\Delta = \{\tilde{L}(s) + W(s)\Delta(s) \,|\, ||\Delta|| \leq 1\}$$

with

$$\begin{aligned} \tilde{L}(s) &= L(0,s) \\ W(s) &= P(s,1,1) - P(s,0,0) \\ &= \frac{10(0.2s^2+0.7s+0.2)}{(s^2+0.5s+1)(s^2+2s+3)(s^2+3s+6)} \end{aligned}$$

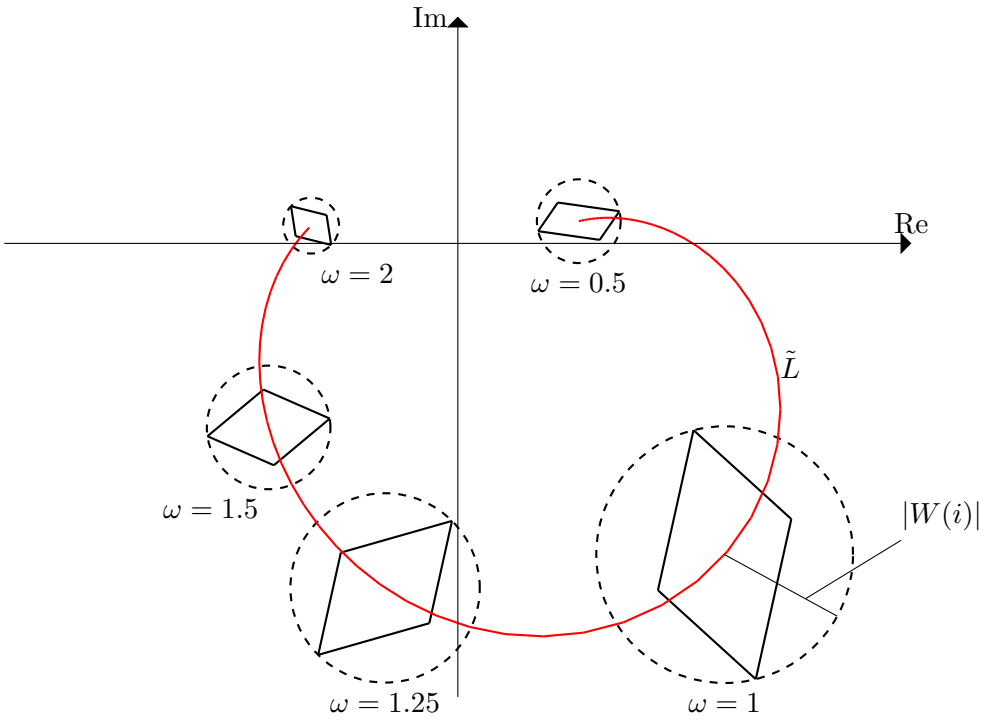
The set L_Δ is an over approximation of the exact set $L([-1,1]^2) = \{L(\theta)|\theta \in [-1,1]^2\}$, as shown on Figure 4.8a representing $L([-1,1]^2)$ and L_Δ at several pulsation in the Nyquist plan. The exact sets correspond to the polygons, and the L_Δ approximation appears as circles of radius $|W(s)|$ centered on $\tilde{L}$ represented by the red curve.

On the contrary, defining a natural inclusion function for the real and imaginary parts of L, the exact set of $L([-1,1]^2)$ can be approximated by computing the image of $[-1,1]^2$. In order to reduce the wrapping effect, the initial box can be divided into smaller boxes and the union of their image by the inclusion function provides a closer approximation of the exact set. Figure 4.8b shows the approximation of L obtained by dividing the initial domain in boxes of diameter 0.1. The gray sets are the union of the images of these boxes by the inclusion functions. Thanks to the convergence property of the inclusion function, these approximations converge to the exact sets as the diameter of the boxes tends to zero. Figures 4.8a and 4.8b emphasize the fact that an approach based on interval analysis is not conservative.

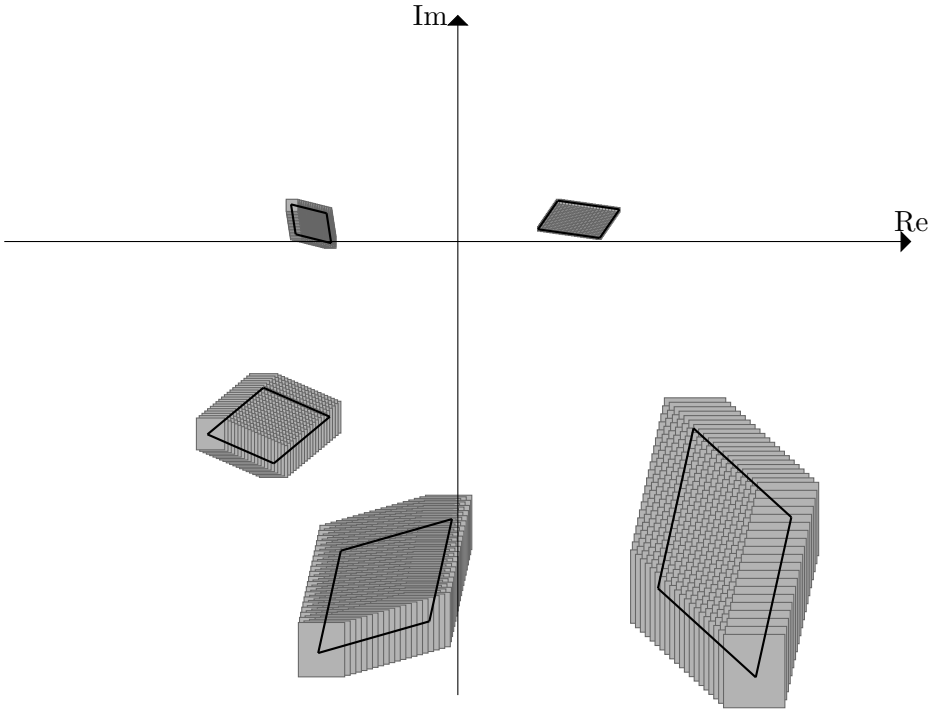
Remark. The problem of characterizing the feasible set of uncertain parameters with respect to both stability and H_∞ constraints is equivalent to Problem 4.1.8 introduced in the previous section. In the first case the a set of uncertain parameters is searched, in the second case a set of controllers is searched.

4.3 Robust structured H_∞ synthesis

Section 4.1 has proposed a formulation of the SS problem and Section 4.2 provided a way to analyze the robust stability and to compute the worst case among uncertainties with respect to the H_∞ objectives. The present section merges these two sections by proposing a formulation of the Robust Structured H_∞ Synthesis (RSS) problem expressed in a general way by Problem 4.3.1.



(a) Conservative approximation of the set of the uncertain system L by L_Δ



(b) Approximation of the set of systems using the natural inclusion function extension of the real and imaginary part of L at several pulsations.

Figure 4.8: *Delta*-structure and interval approximation of uncertain systems.

Problem 4.3.1. *Structured Robust H_∞ Synthesis*
Given P an uncertain LTI system depending on $\theta \in \Theta$, find the structured controller K such that

$$\begin{cases} ||F(P,K)||_\infty \leq 1, \forall \theta \in \Theta \\ K \text{ ensures internal stability } \forall \theta \in \Theta. \end{cases}$$

We suppose that P is built from an uncertain LTI system G depending on θ and weighting functions W_i and Z_j as shown on Figure 4.9.

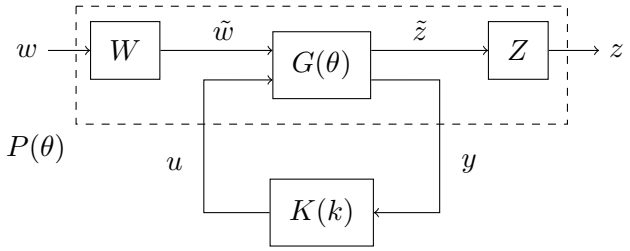


Figure 4.9: General regulation scheme for robust H_∞ synthesis.

4.3.1 Worst case minimization formulation

As proposed in Section 4.1, the structured controller is expressed as a parametric system depending on the tunable variable k. In addition, P depends on the uncertain variable θ. As a consequence, the interconnection of P with K depends both on the tunable and the uncertain variable, and the transfer matrix $T_{w\to z}$ of $F(P(\theta), K(k))$ has its transfers being rational functions depending on k, θ, and s. Considering each performance output z_j independently, we have that the function f also depends on these three variables.

$$\begin{array}{rll} f: & \mathbb{R}^{n_k} \times \mathbb{R}^{n_\theta} \times \mathbb{R} & \mapsto \mathbb{R} \\ & (k, \theta, \omega) & \to f(k, \theta, \omega) \end{array}$$

One possible approach to solve the RSS problem is to minimize over the domain of k the wort-case of the H_∞ objectives over Θ, *i.e.* to minimize the objective

$$\sup_{\omega \geq 0, \theta \in \Theta} f(k, \theta, i\omega),$$

which is equivalent to minimize

$$\sup_{\theta \in \Theta} \max_j ||T_{w\to z_j}(k, \theta)||_\infty,$$

as discussed in the section dedicated to robustness analysis.
Concerning the stability criterion, it suffices to extend the calculus of Subsection 4.1.2 considering the dependency of G on θ. Hence, it follows that using the Hurwitz criterion the internal stability can be expressed as polynomial inequalities $R_l(k, \theta) \leq 0$. The worst case minimization problem is given by Problem 4.3.2, which is the extension of Problem 4.1.6 taking the uncertain parameter into account.

Problem 4.3.2.

$$\begin{cases} \min\limits_{k} & \sup\limits_{\theta\in\Theta,\omega\geq 0} f(k,\theta,\omega) \\ s.\ t. & R_l(k,\theta) \leq 0,\ \forall\theta\in\Theta. \end{cases}$$

The objective becomes the worst case of f over both ω and θ, and the stability constraints become semi infinite constraints. In the end, Problem 4.3.2 is a minmax problem under semi infinite constraints, that can be solved with MMIBBA that has been developed for this purpose. If the best solution $\tilde{k}$ returned by MMIBBA is lower than one, then it means that $K(\tilde{k})$ ensures both robust stability and robust H_∞ performances with respect to the uncertainties. Indeed, MMIBBA repeatedly try to compute a good solution at particular points $\tilde{k}$ by verifying the constraints and evaluating the objective, that actually corresponds to perform both the RSA and the RHA at this point. On the contrary, if the lower bound on the minimum is strictly greater than one, then it means that the RSS problem has no solution.

4.3.2 Example

As an example, we consider the problem of designed a filtered PID controller for an autonomous robot taken from [128]. The robot is model by an uncertain system G depending on two uncertain parameters.

$$G(\theta,s) = \frac{1}{\theta_1 s^2 + \theta_2 s}$$

The range of the uncertain parameters is $[0.3, 0.69] \times [1.26, 2.34]$. The controller K has three tunable parameters, the filter coefficient being fixed to 1.

$$K(k,s) = k_p + \frac{k_i}{s} + \frac{k_d s}{1+s}$$

The H_∞ objectives focus on the limitation of the tracking error and avoidance of actuator saturation. Let $S(k,\theta)$ be the sensitivity function of the close loop. The two H_∞ objectives are given by:

$$||Z_e S(k,\theta)||_\infty, \quad ||Z_u K(k) S(k,\theta)||_\infty,$$

with Z_e and Z_u the weighting functions for the error and the control signal respectively, chosen as

$$Z_e(s) = 0.5\frac{s+0.92}{s+0.0046} \quad Z_u(s) = 0.01.$$

The resolution of the RSS problem with MMIBBA over the search domain $[0,5]^3$ provides $[0.57, 0.84]$ as enclosure of the minimum in 30 minutes. The best solution found is $\tilde{k} = (1.40885, 0.182807, 0.645285)$.

4.3.3 Discussion

As it can be noticed, the computation time is important on this example. This is principally due to the fact that every time MMIBBA evaluate a particular point $\tilde{k}$ both the RSA and the RHA are performed, which takes non negligible time. In comparison, MMIBBA converges to the lower bound in 50s if no feasible point is searched. However, as far as we know, no global solution has been proposed to solve globally the worst-case minimization problem until now. The main difference between our solution and previous work based on parametric approaches [84, 5] lies in the facts that these methods characterize the sub-optimal feasible set of Problem 4.3.2, which corresponds to the feasible set of the CSP

$$\begin{cases} f_j(k,\theta) \leq 1, \forall \theta \in \Theta, \forall \omega > 0 \\ R_l(k,\omega) \leq 0, \forall \theta \in \Theta. \end{cases}$$

The reasons is that the f_j are fractions of two multivariate polynomials N and D,

$$f_j(k,\theta,\omega) = \frac{N(k,\theta,\omega)}{D(k,\theta,\omega)}.$$

As a consequence an inequality constraints on f_j can be reformulated as a polynomial inequality:

$$f(k,\theta,\omega) \leq \gamma \iff N(k,\theta,\omega) - \gamma D(k,\theta,\omega) \leq 0.$$

Having a polynomial constraint is more convenient than having a fraction of polynomial as an objective function to minimize, since methods dedicated to polynomial analysis can be used [52, 96, 86]. This may explain why the majority of works based on the parametric approach are limited to the characterization of the sub-optimal set, but does not solve the worst case minimization problem.
Another point to be mentioned is the relevance of the worst case H_∞ norm, over Θ, as the objective. The worst case over Θ, beside depending on the controller parameters, may not correspond to the nominal case $\tilde{\theta}$ of the system. The worst case minimization proposed by Problem 4.3.2 is convenient since, if a solution lower than one is found, this solution ensures the robust performances of the system. However, one can consider to minimize the H_∞ norm of a nominal system $G(\tilde{\theta})$ and to change the objective of Problem 4.3.2 as a constraint, providing the new Problem 4.3.3.

Problem 4.3.3.

$$\begin{cases} \min\limits_{k} & \sup\limits_{\omega \geq 0} f(k,\tilde{\theta},\omega), \\ s.\ t. & f(k,\theta,\omega) \leq 0, \forall \theta \in \Theta, \\ & R_l(k,\theta) \leq 0, \forall \theta \in \Theta. \end{cases}$$

In this way, the structured synthesis is performed for the nominal system and the H_∞ robustness is guaranteed. Problem 4.3.3 is still a minmax problem subject to semi infinite constraints.

4.4 Conclusion

In this chapter a new formulation of three H_∞ related problems, namely the structured synthesis, the robustness analysis, and the robust structure synthesis has been proposed. This is possible thanks to the formulation of the H_∞ norm of MISO systems as an explicit maximization problem, by considering each performance outputs independently. This result is the first contribution of this chapter, and is of major interest since it enables to extend the existing parametric approaches [84, 5, 21], so far limited to SISO systems, to MISO systems. Furthermore, we show that this formulation is closer to the design objectives than minimizing the H_∞ norm of the full MIMO system with the illustrative example of Section 4.1.
The second contribution is the formulations of the SS, RA, and RSS problems themselves as explicit optimization and CSP problems. It has been shown over several examples that the algorithms developed in Chapter 3 converges in reasonable amounts of time. However, on the RSS problem, MMIBBA shows its limits.
Another point to mention is the explicit formulation of the stability and H_∞ criteria. These explicit formulas, obtained from LFT or the state space realization of interconnected systems, require to inverse symbolic matrices depending on k and θ (see Equation 4.5). Those matrices are in most cases sufficiently sparse to make their inverses easily computable by symbolic software, but the symbolic expressions of the criteria, especially the stability inequalities, may be huge. This can lead to badly conditioned problems, but the reliability of the numerical results are guaranteed by the interval computation that takes rounding error into account (see Chapter 2). In the end, the approach proposed in this thesis is suited for small scale problems, but avoid the conservatism due to uncertainty reformulation. The approach is not suited for large scale problem since:

- The expressions of the Routh's coefficients and the H_∞ norms grow with the order of the closed loop system, and increase the pessimism of the bounds provided by the inclusion functions. Our approach is limited to closed loop system of order 10 or less (order of the controller plus order of the augmented system).

- The complexity of MMIBBA grows exponentially with respect to the dimension of k and θ. Our approach is limited to problems where the dimension of k and θ are both lower or equal than 3.

However, our approach is relevant since the structured controllers are generally wanted of small order, and often with few tunable parameters.

Improvement prospects As explained in the discussion part of the section dedicated to the RSS problem, the convergence issue of MMIBBA lies in the amount of time needed to evaluate the objective function at a particular point. MMIBBA does not implement a "cleaver" method to find a good solution, since it simply evaluates the objective function at the center of the current box. A possible solution to overcome this problem could be to consider derivative free or black box optimization methods [13]. These methods are adapted to solve problems involving objective function requiring non negligible amount of time to be evaluated and that are not known explicitly, *i.e.* the gradient is not available. In our case, the function f is explicitly known but the objective function of the RSS problem is the maximum of f

over Θ and Ω, which is not known and must be computed by solving the RHA maximization problem.
Another way to improve the convergence lies in the expressions of f and the stability polynomials R_l. As the order of the system grows, these expressions become larger and the number of occurrences of the variables increase. This is inconvenient due to the pessimism of inclusion functions, generally increasing with respect to the number of occurrences. However, a large number of common subexpressions appear in the expressions, which can be exploited to improve the efficiency of constraints propagation techniques [11, 56, 125].

Outlooks For the SS problem and the RSS problem, we showed how the synthesis problems can be considered either as minmax optimization problem or CSP problems involving SICs by considering the objective as a constraint. However, any problem having a regular or a max objective function subject to regular constraints and/or SICs is suited to be solved. As a consequence the user can chose specifically which H_∞ criterion is a constraint and which is an objective function.
In Chapter 3, the general minmax problem is considered. This problem is subject to constraints involving both the minimization and the maximization variable, namely k and θ in this chapter. Such constraints can be used to represent dependencies between the controller and the uncertainty, or simply between the uncertain parameters of the system [114]. That is, the domain of uncertain parameters can be any non convex set instead of the box Θ. In the end, the tools developed in this thesis permit a large degree of freedom with respect to the expressions of the problem. This enables to limit the difference between the design objectives and their formulation as an optimization problem.
The performance outputs independence is used to obtain an explicit formulation of the problem, and use the global optimization tools developed in the two previous chapters. However, the branch and bound structure of MMIBBA can be employed even if no explicit formula is available, it suffices that reliable bounds on the objective function can be computed. In this way, works dedicated to the enclosure of the eigenvalues of interval matrices [65, 66] could be used to provide bounds on the singular values of $T_{w\to z}(k, \theta, s)$ over boxes of k, θ, and ω. In this way, $||T_{w\to z}||_\infty$ could be considered as an objective function, but MMIBBA would very probably run slower since the accelerations techniques, relying on explicit functions, could not be used anymore. This could be the topic of a future work.
The approach proposed in this chapter could be employed for the design of gain scheduling controllers for LPV systems. The structured synthesis method developed in Section 4.1 permits to obtain several controllers for different values of the varying parameters of the LPV system to control. Suppose that a regression function r of the relation between the controller parameters k and the varying parameters θ can be obtained, $k = r(\theta)$. The robustness analysis tools can be used to verify that the closed loop system, being the interconnection of $G(\theta)$ and $K(r(\theta))$, is stable and respects the H_∞ constraints, for any value of θ.

Chapter 5

Case study: robust control for underwater robot

A practical example of H_∞ control application is presented in this chapter. The control of an Autonomous Underwater Vehicle (AUV) named Ciscrea is considered, and the whole process from design objectives specification to experiments is performed. The first objective of this chapter is to show how the analysis procedure presented in Chapter 4 can be used on a real case in order to provide a detailed analysis of the system's behavior. The second objective is to emphasized the fact that H_∞ synthesis is an adapted tool to compute control laws for autonomous robots. Indeed autonomous robots generally evolve in unchecked environments and may be subjected to external disturbances, and make robust H_∞ synthesis a suitable approach to control such systems for two reasons :

- several objectives can be taken into account to design the control law, for example tracking error limitation and disturbance rejection,
- robots often suffer from model uncertainties that can be dealt with.

In particular, this chapter focuses on the control of the yaw angle of the Ciscrea subject to external perturbations. In that respect, two objectives are considered :

- ensure a small tracking error between the reference yaw and the Ciscrea's yaw,
- ensure perturbation rejection.

In addition, a simple structure of the control law is imposed. Doing so limits the computation time of the control signal, since the controller, being a dynamical system, must be simulated to obtained it. In addition, a low order structure of the controller ease the interpolation of controllers for different operating point, if desired. For this application the controller is a PID.
This chapter is organized as follows : Section 5.1 provides a non linear model of the Ciscrea, and proposes a linear approximation of this model in order to use H_∞ synthesis. Section 5.2 derives H_∞ criteria from the control objectives and the synthesis of a controller is performed. The performances of this controller are analyzed with the tools developed in Chapter 4. In

Figure 5.1: Ciscrea AUV.

addition, two other controllers are computed from empirical tuning rules taken from the literature. The performances of the three controllers are compared through simulations and experiments in Section 5.3.

This work was performed in cooperation with the National University of La Plata (UNLP), Argentina, with a scholarship granted by Université Bretagne Loire (UBL), and has led to a publication [110].

5.1 AUV model and perturbations

The Ciscrea's dynamical model is taken from [45] and the numerical values have been obtained in [127] which proposes a rigorous modeling of the robot. Based on [45], two coordinate systems are introduced: a NED-frame (North East Down) and a B-frame (Body fixed reference) for the localization as it is described in Fig. 5.2. In this model, all distances are expressed in meters, angles in radians and positive clockwise. The position vector η, velocity vector ν, and force vector τ are defined as is follows,

$$\begin{aligned} \eta &= (x, y, z, \phi, \theta, \psi)^T \\ \nu &= (u, v, w, p, q, r)^T \\ \tau &= (X, Y, Z, K, M, N)^T. \end{aligned} \tag{5.1}$$

According to [45], rigid-body hydrodynamic forces and moments can be linearly superimposed. Furthermore, the overall non-linear underwater model is characterized by two parts, the rigid-

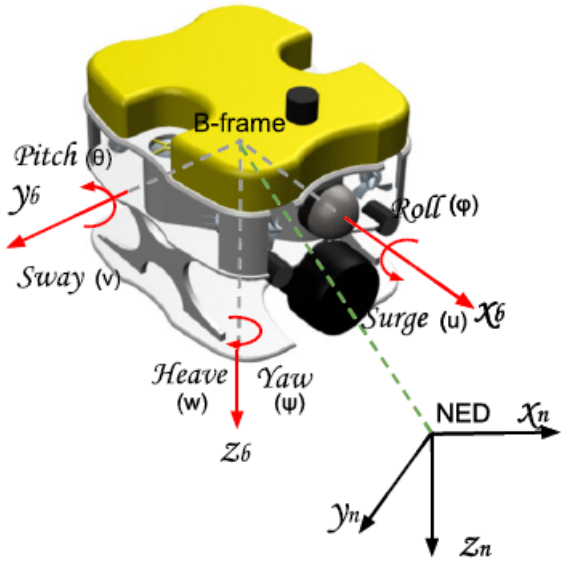


Figure 5.2: NED-frame and B-frame.

body dynamic (see Equation (5.2)) and hydrodynamic formulations included hydrostatics (see Equations (5.3)). Parameter definitions are given in Table 5.1.

$$M_{RB}\dot{\nu} + C_{RB}(\nu)\nu = \tau_{\mathbf{env}} + \tau_{\mathbf{hydro}} + \tau_{\mathbf{pro}} \tag{5.2}$$

$$\tau_{\mathbf{hydro}} = -M_A\dot{\nu} - C_A(\nu)\nu - D(|\nu|)\nu - \mathbf{g}(\eta) \tag{5.3}$$

Table 5.1: Nomenclature of AUV Model

Parameter	Description
M_{RB}	AUV rigid-body mass and inertia matrix
M_A	Added mass matrix
C_{RB}	Rigid-body induced coriolis-centripetal matrix
C_A	Added mass induced coriolis-centripetal matrix
η	Position vector
ν	Velocity vector
$D(\|\nu\|)$	Damping matrix
$\mathbf{g}(\eta)$	Restoring forces and moments vector
$\tau_{\mathbf{env}}$	Environmental disturbances (wind,waves and currents)
$\tau_{\mathbf{hydro}}$	Vector of hydrodynamic forces and moments
$\tau_{\mathbf{pro}}$	Propeller forces and moments vector

In the present application M_{RB} is obtained from [127]. In addition, since the vehicle speed is low, C_{RB} and C_A are neglected, then $C(\nu) \approx 0$. The restoring forces and moments vector

$\mathbf{g}(\eta)$ are composed of the forces and torque produced by the weight and the buoyancy forces. The assumption that both the buoyancy and gravity center are located at the geometrical center of the robot is made.
The marine disturbances, such as wind, waves, currents contribute to $\tau_{\mathbf{env}}$. Two hydrodynamic parameters deserve a greater explanation:

- $M_A \in \mathcal{M}_6(\mathbb{R})$: added mass, is a virtual concept representing the hydrodynamic forces and moments. Any accelerating emerged-object would encounter this M_A due to the inertia of the fluid.
- $D(|\nu|) \in \mathcal{M}_6(\mathbb{R})$: damping in the fluid, this parameter consists of four additive parts: Potential damping, wave drift damping, skin friction, and vortex shedding damping. The first two are dismissed in this application, and the others could be approximated by a linear and a quadratic matrix, D_L and D_N, respectively, as it is shown in (5.4) ([127], [45]).

$$D(|\nu|) = D_L + D_N|\nu|. \tag{5.4}$$

In the present work, we focus on the yaw direction to control. Due to the low coupling in the model directions, it is possible to consider that there is no dependency between the yaw dynamic and dynamics along the other axis. This last observation allows us to get the non-linear model for the yaw dynamic:

$$(I_{YRB} + I_{YA})\ddot{\psi} + (D_{YL} + D_{YN}|\dot{\psi}|)\dot{\psi} = \tau + d, \tag{5.5}$$

where the parameters are listed in Table 5.2. Since the yaw speed is mostly between 0 and 4 rad/s to dismiss the Coriolis effect in 5.5 is a reasonable approximation.

Table 5.2: Model parameters for Yaw dynamic

Parameter	Description
$I_{YRB} = 0.2862$	AUV inertia
$I_{YA} = 0.1104$	Added inertia
$D_{YL} = 0.0945$	Linear damping coefficient
$D_{YN} = 1.4676$	Non-linear damping coefficient
d	Disturbances
τ	Resulting torque produced for all propellers
ψ	Yaw position

To conclude the modeling of the Ciscrea robot, two other assumptions must be addressed: (a) a delay in the compass sensor, which is estimated experimentally around 0.3 seconds. And (b) the non-linear behaviour between the digital command torque T_d and the real torque T_a, in Newton, developed by the Ciscrea's thrusters expressed by the following equation:

$$T_a = \begin{cases} 4.7 & if \quad T_d \geq 127 \\ 3.2 \max\left(\frac{T_d}{203.874}, \frac{T_d - 30.3781}{65.6756}\right) & if \quad 0 \leq T_d < 127 \\ 4.3 \min\left(\frac{T_d}{203.874}, \frac{T_d + 30.3781}{65.6756}\right) & if \quad -127 < T_d < 0 \\ -6.32 & if \quad T_d \leq -127 \end{cases} \tag{5.6}$$

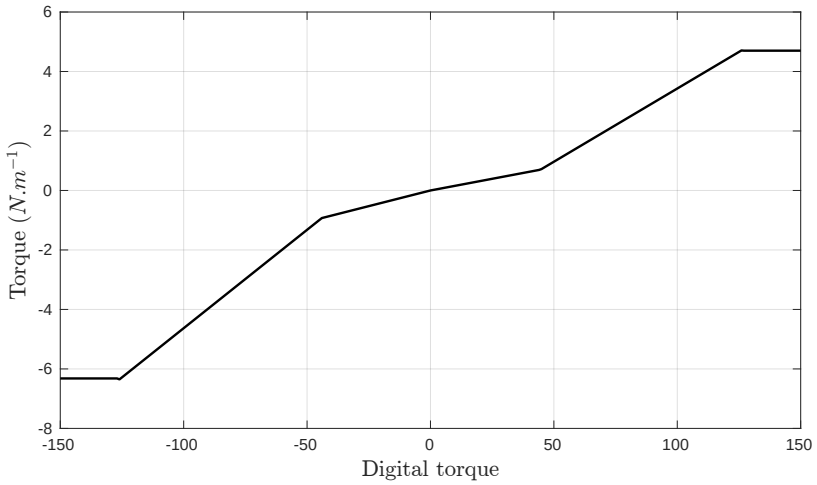


Figure 5.3: Generated torque in Newtons per meters versus digital control.

Figure 5.3 shows the torque over the digital control value. Further details about this model and its validation can be found in [109]. This model has been implemented in Matlab.

5.1.1 Linear model

A non linear model of the Ciscrea has been obtained as well as the model of the perturbations, considered as additional signal d on the control input u. In order to use H_∞ control, a linear model of the Ciscrea must be available. The two main sources of the non linear behavior of the Ciscrea are due to:

- the dynamical model given by Equation (5.5) and the torque's non linear behavior, and
- the compass delay.

The Ciscrea is approximated by a linear system by considering these two items separately.

Dynamical model and torque

We first consider the non linear differential equation of the Ciscrea and the torque control. In order to obtain a linear model from a non linear model, one possible way to proceed is to linearize the system around an operating point. That is, a system described by its evolution function f, its observation g and its state vector x can be approximated by state space matrices A, B, C and D.

$$\begin{aligned} A &= \frac{\partial f(\tilde{x})}{\partial x} \quad B = \frac{\partial f(\tilde{u})}{\partial u} \\ C &= \frac{\partial g(\tilde{x})}{\partial x} \quad D = \frac{\partial g(\tilde{u})}{\partial u} \end{aligned} \tag{5.7}$$

The operating point is given by $\tilde{x}$ and $\tilde{u}$. In our case, the operating point simply corresponds to a particular angular velocity which is the cause of non linearity in Equation (5.5). From experiments [126], it is known that this angular velocity can vary between 0 and 4 rad/s. Instead of linearizing around a single operating point, we propose to consider the operating angular velocity as a time invariant uncertain parameter $\theta \in [0, 4]$. Recalling that a Matlab model implementing both the dynamical equation of the Ciscrea and the non linear behavior of the torque is available, we directly linearized this model using Matlab's `linearize` function at several operating angular velocity. By extrapolation, we get an uncertain LTI system L.

$$L(\theta, s) = \frac{0.3931}{s^2 + 2.08\theta s}$$

Remark. L is only an approximation of the dynamics of the Ciscrea, an uncertain LTI system cannot describe the evolution of a non linear system. However, this approximation is generally sufficient to compute efficient control laws, as it will be shown in Section 5.3.

Compass delay

A delay is modeled in the Laplace space by a transfer $e^{-\tau s}$, with τ the delay time in seconds. The Padé approximation for the exponential function can also be used to get a rational function depending in s, *i.e.* a LTI system. This approximation is given by a series expansion, and we consider here only the first order approximation,

$$e^{-\tau s} = \frac{1 - s\frac{\tau}{2}}{1 + s\frac{\tau}{2}}. \tag{5.8}$$

Ultimately, the linear approximation of the whole system is given by the third order uncertain LTI system G.

$$G(\theta, s) = \frac{0.3931}{s^2 + 2.08\theta s} \frac{1 - s\frac{\tau}{2}}{1 + s\frac{\tau}{2}}$$

Figure 5.4 recaps the linearization process.

Remark. Other linearization processes can be employed to obtain a linear model. For example, a feedback linearization coupled with a predictor to compensate the delay is proposed in [128]. Further discussions about this topic can be found in [126].

Remark. The saturation of the propellers is not taken into account in the linear approximation. It will appear that such a phenomenon is not met in the experiments, presented in Section 5.3. However, actuator saturation is an issue often met, and might be critical in certain applications [124].

5.2 Design specifications, synthesis, and analysis

The design objectives consist in

- Achieve a small tracking error between the reference angle, denoted r, and the yaw angle of the robot.

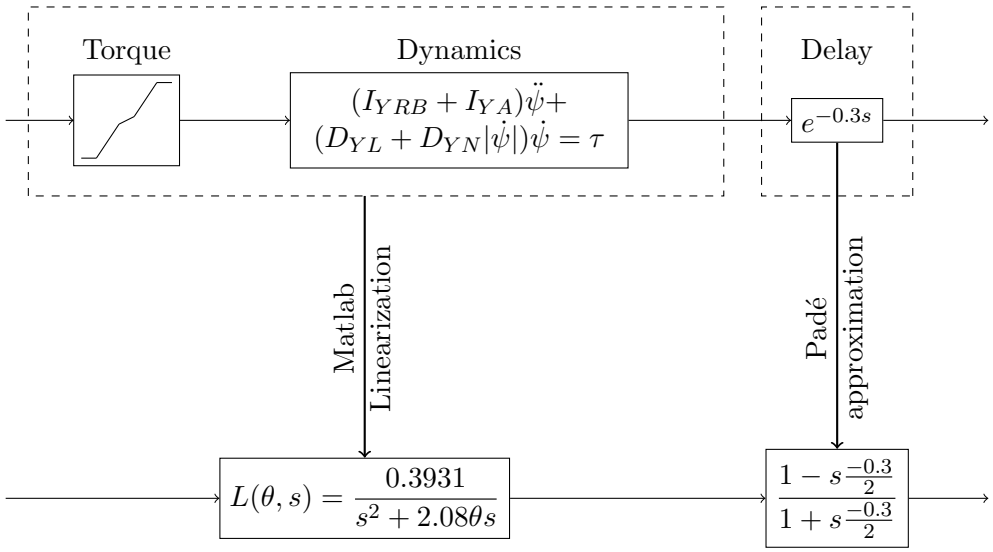


Figure 5.4: Linearization process of the Ciscrea, represented as three non linear blocks.

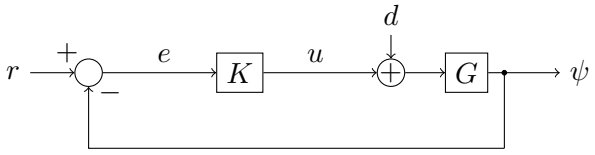


Figure 5.5: Regulation scheme for Ciscrea yaw angle control.

- Disturbance rejection.

The tracking error regulation scheme shown on Figure 5.5 is considered to achieve these objectives. From Equation (5.5), the disturbances are described as an additional input d on the control signal. In the end, Figure 5.5 shows a suitable configuration of the regulation problem for the H_∞ synthesis. The two objectives must now be translated to H_∞ objectives.

5.2.1 Specifications

Tracking error objective

The channel of interest for this objective is the transfer from the reference to the error signal, $T_{r\to e}$, corresponding to the sensitivity function.

$$T_{r\to e}(k, \theta, s) = \frac{1}{1 + G(\theta, s)K(k, s)}$$

In most cases, the frequency template Z_e^{-1} chosen to ensure a small tracking error is a high-pass filter. In this way, $T_{r\to e}$ has low gain at low frequencies, and error is tolerated at high frequencies. What is a low or a high frequency is determined by the cutoff frequency of Z_e^{-1}. A reasonable choice for this cutoff frequency is to have it around the cutoff frequency of G,

possibly lower. If it is higher, the relative H_∞ constraint specifies that the system must track the reference at high frequencies where G has less gain. The perverse effect of such a constraint is that the H_∞ synthesis may provide a controller which has a high gain at high frequency to compensate the low gain of G, which can lead to actuator saturation issues. Figure 5.6 displays the bode diagram of G for several values of θ in [0,4]. The cutoff frequency ω_c of G depends on θ, and is around 1 rad/s. In the following, we assume that $\omega_c = 1$. Following the

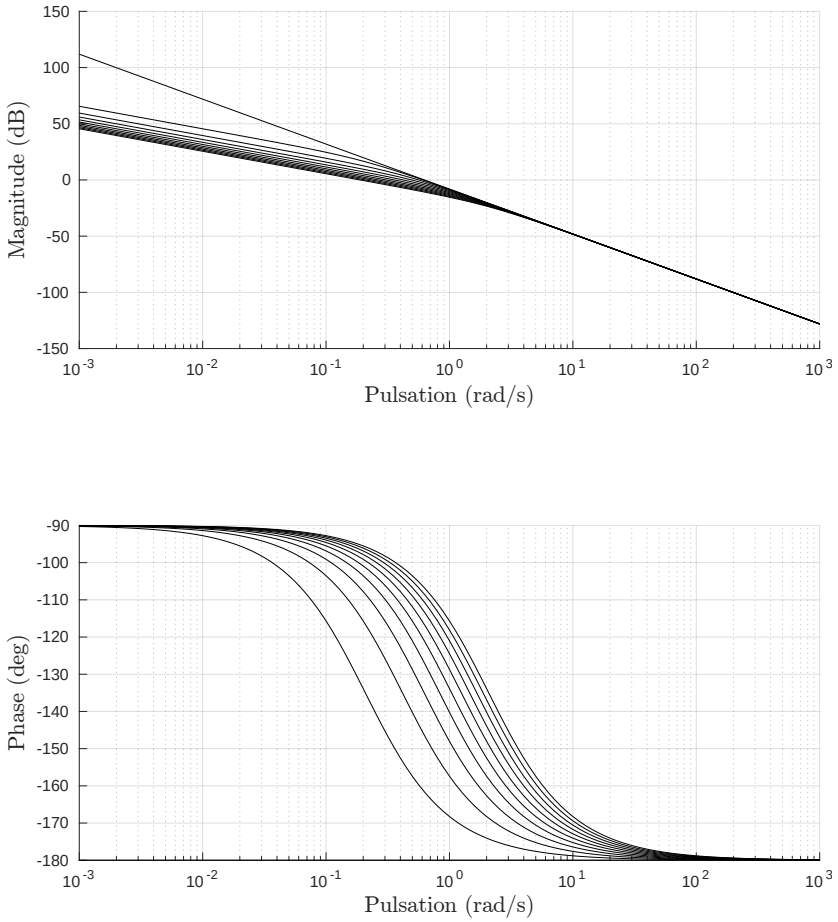


Figure 5.6: Bode diagram of G.

above, the weighting function Z_e (being the inverse of the frequency specification) is chosen as a low pass. The low frequency gain is set to 20 dB and the high frequency gain to -20 dB, with a cutoff frequency at 1 rad/s.

$$Z_e(s) = \frac{0.1s + 0.6283}{s + 0.06283}$$

Z_e^{-1} is displayed on Figure 5.7 with G at $\theta = 0.5$. Ultimately, the specification on the tracking error is expressed as the H_∞ objective $||W_e T_{r\to e}||_\infty$ that must be minimized.

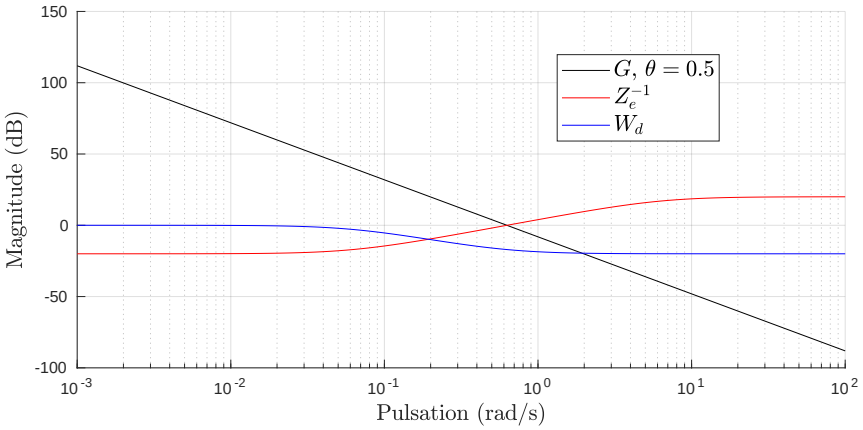


Figure 5.7: Gain of the frequency templates Z_e^{-1} in red, disturbance weighting function W_d in blue, and G in black at $\theta = 0.5$.

Disturbance rejection

The rejection of external disturbances, modeled by d, can be translated as the non sensitivity of e to d. The channel of interest is therefore $T_{d\to e}$. From the tracking error specification, e is already weighted by Z_e. That is, the objective $||T_{d\to e}W_e||_\infty$ ensure the rejection of the perturbation at low frequencies. However, this objective can be improved by taking into account the frequency content of d. The assumption is made that disturbances occur at low frequencies, below 1 rad/s. It follows that d is weighted by a low pass filter W_d, displayed on Figure 5.7 is

$$W_d(s) = \frac{0.1s + 0.06283}{s + 0.06283}.$$

In the end, the disturbance rejection specification is translated as the H_∞ objective:

$$||Z_e T_{d\to e} W_d||_\infty.$$

5.2.2 Synthesis

The controller K is structured as a filtered PID.

$$K(k, s) = k_p + \frac{k_i}{s} + \frac{k_p s}{T_f s + 1}$$

The tunable variable is $k = (k_p, k_i, k_d, T_f)$. This choice is motivated by the fact that a PID can be implemented easily, and its behavior can be understood directly from the value of its tunable parameters. In addition, it will allow to compare the H_∞ synthesis with other PID tuning methods presented further.
The H_∞ synthesis of the PID consists to solve the robust structured synthesis problem, introduced in Chapter 4, since G is uncertain. The problem to solve is thus

Problem 5.2.1.

$$\begin{cases} \min\limits_{k\in\mathbb{K}} & \sup\limits_{\theta\in[0,4]} \max(||Z_e T_{r\to e}||_\infty, ||Z_e T_{d\to e} W_d||_\infty, ||T_{r\to u}||_\infty Z_u) \\ s.\ t. & K(k) \textit{Ensure internal stability.} \end{cases}$$

Unfortunately, MMIBBA cannot find a good solution in a reasonable amount of time, for no heuristic is implemented to find rapidly a good solution (see Chapter 4). Instead, the Matlab's systune toolbox is used to compute a controller. Systune proceeds as follows: a controller is computed by solving Problem 5.2.1 at a nominal value of θ, corresponding to solve a structured synthesis problem. Then, a robustification procedure is run from this solution to find a controller that minimizes the worst case H_∞ norm while ensuring stability over the domain of uncertain parameters. For detail, see [8, 1]. Since Systune relies on local search optimization methods, solutions are fast to compute but are not guaranteed with respect to the stability and H_∞ criteria. That is why the strategy adopted is to perform the synthesis with Systune and to perform a robustness analysis with the tools developed in Chapter 4. The solution to Problem 5.2.1 provided by Systune is

$$\tilde{k} = (4.68, 0.71, 4.68, 0.11)$$

The robustness analysis is performed over the pulsation range [0.01,ω_c]. The following results are obtained:

$$\sup_{\theta\in[0,4]} ||Z_e T_{r\to e}||_\infty \in [3.16, 3.22],$$
$$\sup_{\theta\in[0,4]} ||Z_e T_{d\to e} W_d||_\infty \in [0.42, 0.57].$$

These results are confirmed by Figure 5.8 and 5.9 showing the Bode diagrams of $T_{r\to e}$ and $T_{d\to e}W_d$ respectively, for ten values of θ. $T_{r\to e}$ overpass the frequency template given by Z_e^{-1}, and $T_{d\to e}W_d$ remains below Z_e.

In addition, the robust stability analysis proves that the stability of the close loop for any value of θ in [0,4]. As a consequence, the solution provided by Systune respects the robust stability condition, which is the most critical condition, the disturbance rejection objective is met but not the tracking error objective. At this point, two options can be considered:

- Changing the H_∞ objectives to less demanding ones, for example by lowering the cutoff frequency of Z_e, then run the synthesis and the analysis. This is repeated until a controller that satisfies the H_∞ constraints is found,
- or perform a detailed analysis in order to know for which values of θ the objectives are not met and draw the conclusion.

As we developed the necessary analysis tools, we chose the second option. First, FSCA is run in order to obtain a sub-paving informing the subset of the uncertainty domain $[0, 4]$ such that the H_∞ constraints are respected. This sub-paving is displayed on Figure 5.10.

The results provided show that the H_∞ constraints are respected over the subset $[0, 1.21]$. In other words, the specifications are met for low angular velocities, and the closed loop system is stable at any angular velocities. We can conclude from this analysis, that the controller

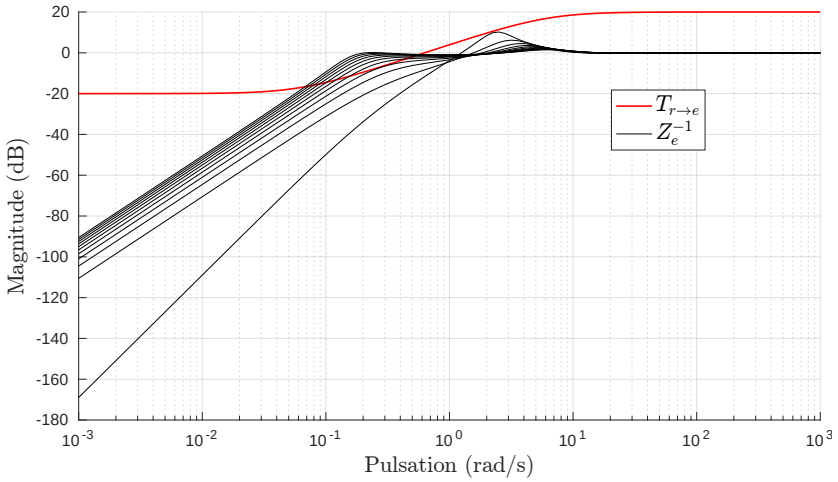


Figure 5.8: Diagram of $T_{r\to e}$ and Z_e^{-1}.

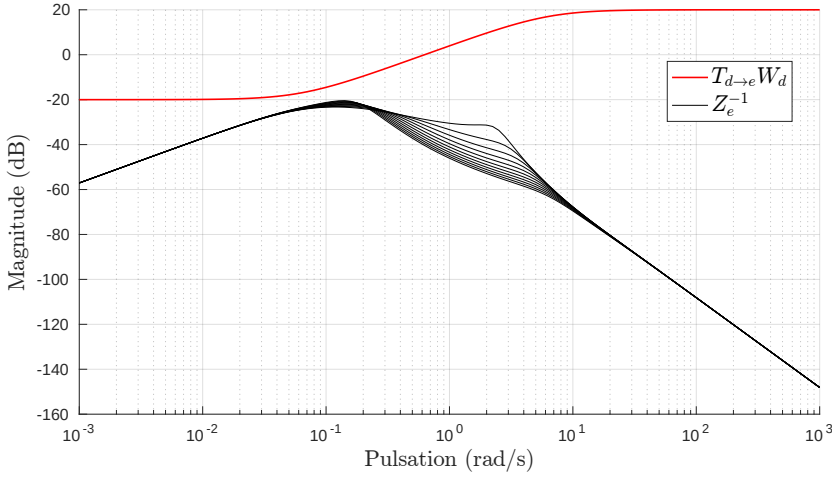


Figure 5.9: Diagram of $T_{d\to e}$ and Z_e^{-1}.

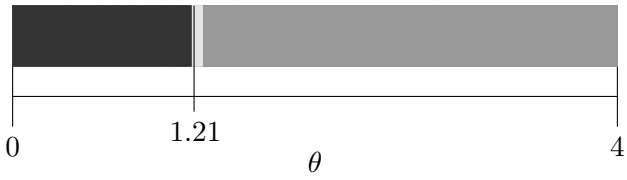


Figure 5.10: Feasible subset of the uncertainty domain, in dark gray, with respect to the H_∞ criteria.

$K(\tilde{k})$ is a suited control law at low angular velocities, and its performances declines as the velocity increased. In addition, it must be noted that this analysis is limited since the Ciscrea is a non linear system, and the model G use for the synthesis is an approximation of the real system. Ultimately, we propose to keep this controller and to validate its performances through simulations with the non-linear model and experiments in the next section.
Before doing so, two other PID controllers are synthesized in order to provide comparisons between the H_∞ solution and other synthesis processes.

5.2.3 Two empirical solutions

When tuning a PID controller, empirical methods can be found in the literature to compute the coefficients. In particular, we consider the *Ziegler-Nichols* tuning method which is well-known in automatic control since 1942 [135], and another tuning method allowing to take delay into account taken from [102]. Both methods are based on the time response of the system, that is considering the yaw angle time response to a step reference signal. These two methods are chosen on purpose as they consider only one tuning objective, the tracking error, and cannot take into account a second objective, the external perturbations. In this way, it is possible to discuss the advantages and disadvantages of H_∞ synthesis compare to two other easy to use methods.
Following the Ziegler Nichols design process, we obtain the controller

$$k_{ZN} = (1.32, 0.22, 1.89, 0.5).$$

The tuning rules taken from [102, p 395], based on Chien's work [31] require a second order linear model, and provide the coefficients of the PID in function of the time delay. To do so, $L(\theta = 2)$, the Ciscera's dynamical model linearized at 2 rad/s, is considered, and the controller $K(k_{Chien})$ is obtained,

$$k_{Chien} = (1.82, 0.12, 6.4, 0.35).$$

5.3 Simulations and experiments

The three controllers are compared over simulations and real experiments. The main objective is to show the robustness of the proposed controller against perturbations. The performances of the controllers are evaluated with respect to two criteria.

- The root mean square error,

$$\sqrt{\sum_{i=1}^{N} \frac{|r_i - y_i|}{N}}$$

- and the bias,

$$\sum_{i=1}^{N} \frac{r_i - y_i}{N}.$$

Here N is the number of measures provided by the Ciscrea's compass during the experiments or the simulations outputs. y_i is the yaw angle and r_i is the reference. Concerning the experiments, since the compass suffers from delay, the measures are shifted over the time by the delay of the compass.

5.3.1 Simulations

The simulations were done using the non-linear model described in Subsection 5.1. Three simulations are performed:

- step response,
- response to step disturbance, and
- response to a random disturbance.

Figure 5.11a shows the step response obtained with the three controllers. In this figure, the *Hinf* controller has a higher over pass than *Chien* controller, but at the same time the settling time is shorter. This overshoot is a consequence of the design objectives of the Chien and H_∞ tuning: the Chien controller has been designed specifically from the step response in contrast to the Hinf controller.

Figure 5.11b shows the response of the system to a step perturbation filtered by W_d, applied to the control input. It appears that the *Hinf* controller rejects this perturbation well, contrary to *ZN* and *Chien* controllers which do not take into account perturbation rejection as a constraint in their design process.

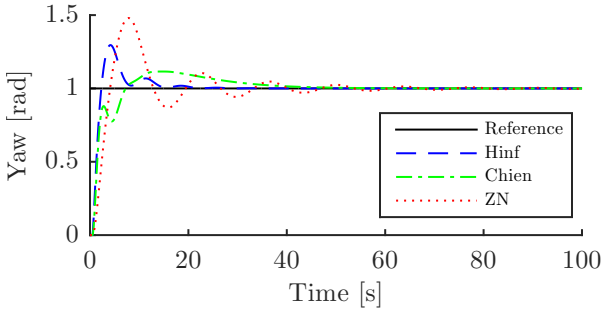
In the last simulation a white uniform noise signal filtered by the weight function W_d (in this way the system is excited in the bandwidth where the disturbances are expected) is applied as a disturbance to the control input. Figure 5.11c shows the yaw output. The *Hinf* controller has the best performance in these conditions. The values of the Root-Mean-Square Error (RMSE) and the bias given in Table 5.3 confirms this observation.

Table 5.3: Random perturbation errors

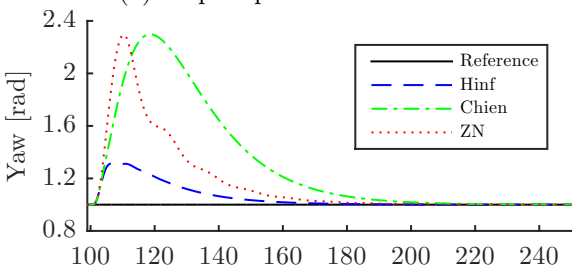
Simulation	RMSE	Bias
ZN	1.7132	-0.1263
Hinf	0.6017	-0.0480
Chien	1.6816	-0.1334

From these simulations, it can be concluded that Hinf controller is good at rejecting perturbation contrary to the ZN and Chien controllers. These results emphasized two aspects:

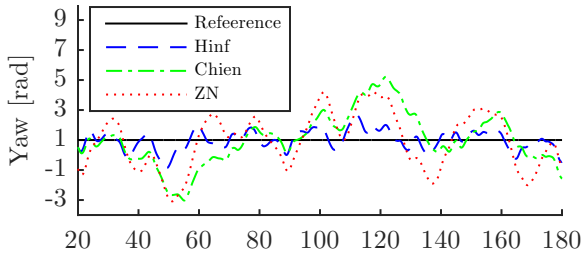
- The sensitivity of the Ciscrea to perturbations cannot be ignored in the design process if one wants a robust controller. ZN and Chien controllers does not take perturbations into account and are sensitive to them.
- The Hinf controller shows the performances expected from the design specifications and the robustness analysis, despite it has been synthesized from a linear approximation of the Ciscrea.



(a) Step response simulation



(b) Step perturbation simulation



(c) Random perturbation simulation.

Figure 5.11: Simulation results.

Since the simulations validate the performances of the Hinf controller, the next step is to perform real experiments.

5.3.2 Experimental Results

Experiments setup

The three controllers are compared over three experiments conducted at the ENSTA-Bretagne facilities. Each experiment consists in testing the performance of the three controllers on the real robot subject to perturbations. In all the cases, the perturbation was generated by an external 12 Volts propeller with a constant rotational speed. The setup is depicted on Figure 5.12.

The Cisrea is hanged to a winch chain in order to keep it in the center of the pool. Indeed, only the yaw angle is aimed to be controlled, and the perturbation caused by the propeller tends to push the Ciscrea in the opposite side of the pool. In addition, the Ciscrea

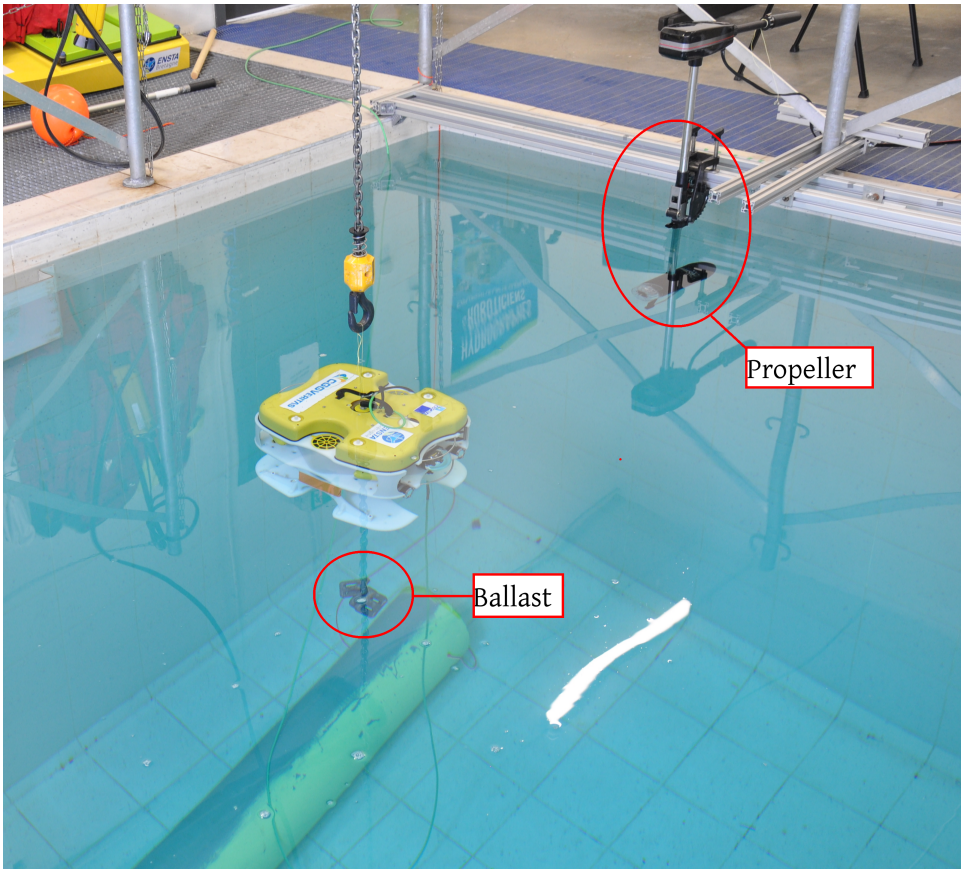


Figure 5.12: Experimental setup at ENSTA Bretagne pool.

is ballasted so that its pitch and roll angles remain low when the Ciscrea swing because of the generated flow disturbance. Particular care has been given to keep the torque resulting from the mechanical link between the Ciscrea and the winch chain as low as possible.
The experiments consist in creating a current with the propeller to disturb the Ciscrea while this last keep a steady yaw angle. Doing so, the angle velocity remains low, where the Hinf controller should perform best as suggested by the robustness analysis. The reference signal is thus constant. The disturbance generated by the propeller is oriented in three different directions with respect to the reference yaw angle, corresponding to a front current (surge direction), a 45 degree orientated current and a side current (sway direction). Figure 5.13 summarized the three experiment setups. In all cases, the perturbation is generated by the same constant rotational speed of the propeller. The first experiment consists in undergoing the AUV to an external perturbation aligned to its sway direction (see A in Fig. 5.13). In Figure 5.14a, the yaw measurement is displayed for each controller. In this case between 0 to 40s, the experimental step response is appreciated, and then at 40s, the external perturbation is applied. From this figure, we can observe the same behaviour as the one predicted by the simulation with respect to the step response, and a good rejection of the perturbation for all the techniques employed.

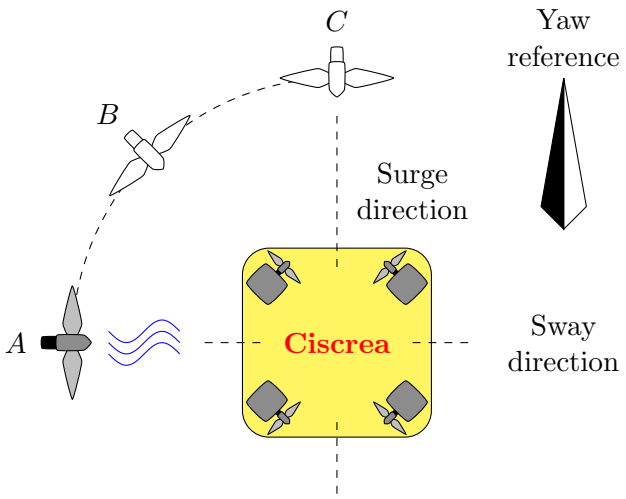


Figure 5.13: Top view of experiment setup

Table 5.4: Errors: Perturbation in sway direction

Experiment	RMSE	BIAS
ZN	0.2166	-0.0204
Hinf	0.1355	-0.0230
Chien	0.1738	-0.0137

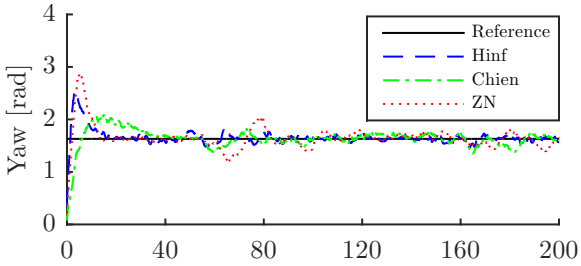
The second experiment consists in exposing the AUV to a perturbation at 45 degree of sway direction (see B in Fig. 5.13). Figure 5.14b shows the results and Table 5.5 provides the error values. In this case, Hinf and Chien controllers have good performances but ZN is clearly sensitive to the perturbation.

Table 5.5: Errors: Perturbation at 45 degree of surge direction

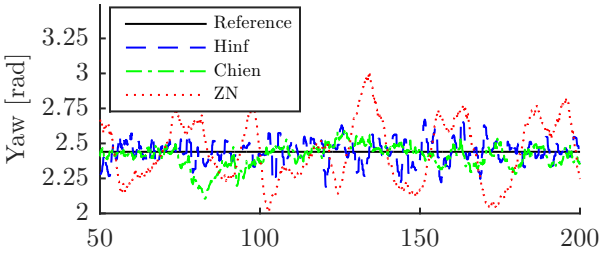
Experiment	RMSE	Bias
ZN	0.1742	0.0137
Hinf	0.0650	0.0037
Chien	0.0755	0.0172

The last experience consists in applying a perturbation in the surge direction (see C in Fig. 5.13). In Figure 5.14c, the worst perturbation condition is observed. The errors for this experiment are provided in Table 5.6. This experiment clearly shows from the plot of the yaw angle and the RMSE value that Hinf is better at rejecting perturbation, which was not so obvious from the two other experiments.

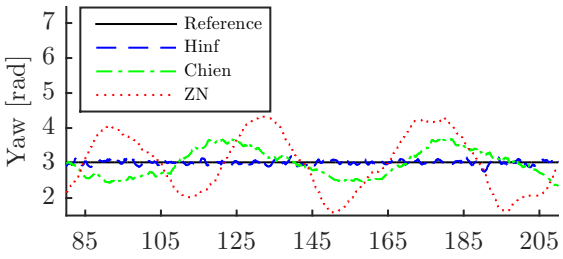
Those three experiments highlight that the Hinf controller is the best at rejecting perturbations, and that the Ciscrea sensitivity to the perturbation depends on its orientation. The step response for the three controllers obtained by simulations and experiments are similar, proving the validity of the non linear model. On the contrary, the behavior of the Ciscrea



(a) Perturbation in sway direction



(b) Perturbation at 45 degree of surge direction



(c) Perturbation in surge direction

Figure 5.14: Experiment results.

Table 5.6: Errors: Perturbation in surge direction

Experiment	RMSE	Bias
ZN	0.3957	-0.0037
Hinf	0.0371	-7.161e-04
Chien	0.2548	0.0256

subject to perturbation differs a bit between simulations and experiments. In that respect, the perturbation generated by the propeller cannot be simply modeled by an additional torque if precise simulations are wanted. Nevertheless, this assumption is sufficient to tune a robust controller for H_∞ objectives as demonstrated by the experiments.

5.4 Conclusion

This chapter shows that the H_∞ synthesis is a relevant method to control the Ciscrea. As emphasized by the comparison with two controllers designed from single objective, the main advantage of the H_∞ approach is that several specifications can be taken into account in the synthesis process. However, the main limitation of H_∞ synthesis is that a linear model is needed and the robots' dynamics are generally non linear. The solution we proposed here is to approximate the non-linear system by a set of linear systems. Unfortunately, MMIBBA cannot provide a good feasible solution in a reasonable amount of time, and the Systune toolbox have to be used to compute a controller. However, using robust analysis tools, it is possible to study the performances of the H_∞ objectives with respect to the uncertain parameter in detail.To this extent both the optimization tools of Chapter 2 and the formulation of the robustness analysis problem of Chapter 4 was necessary. The experiments show that with a constant yaw angle reference, ensuring low angular velocity, Hinf rejects the perturbations well as expected from the robustness analysis.

Future studies

Additional experiments would be needed to validate the robustness analysis over the full range of angular velocity. To do so, the behavior of the Ciscrea to non constant yaw reference must be studied. According to the robustness analysis result, the tracking error should grow accordingly with the angular velocity.
Further studies are also needed to model the disturbance properly. By now, they are only considered as additional torque on the Ciscrea. However, it can be expected that the perturbations also modified the flow around the Ciscrea, influencing the added inertia coefficient and the damping coefficients. In addition, the disturbance simulated by the propeller may not describes the one the Ciscrea might face during missions in the ocean. Hence experiments in real environment are also necessary to correctly model the perturbation.

Chapter 6

Conclusion

This thesis proposes to deal with the three classical H_∞ problems: the synthesis, the robust analysis and robust synthesis problems. It is shown how these problems can be reformulated as minmax, maximization and semi-infinitely constrained minmax problems respectively. The necessary algorithms to solve such problems are proposed, and applied on several H_∞ control examples. Finally, H_∞ synthesis is employed to compute a structured control law for an underwater robot. This control law is compared with two other control laws through simulations and experiments.

6.1 Contributions

The contributions of this thesis can be divided between the contributions in optimization and the ones in H_∞ control.

Contribution in control

Concerning the structured synthesis problem, a method is proposed to solve it in a global way. Strictly speaking, this method does not solve the structured synthesis problem as stated in Section 1, but the problem is reformulated by considering each performance inputs independently. From an optimization point of view, our solution is conservative since the initial problem is modified. However, from a control point of view, such a formulation can be more interesting since it is closer to the H_∞ design objectives. In addition, our approach makes it possible to compute a lower bound on the global minimum which guarantees, if it is greater than one, that no controller in a bounded set can achieve the required H_∞ performances.
Concerning the robustness analysis problem, it is shown how it can be cast as a regular maximization problem. Although this problem is not convex, it can be solved in a global way with IBBA or efficient existing global optimization solvers [116, 74]. The main advantage of this formulation is that the worst case H_∞ norm is directly computed over the uncertain parameters. This avoid to use the Δ-structure reformulation of the uncertainty, which is unable to represent the set of parametric uncertain systems, and lead to conservative analysis results.
The robust synthesis problem is approached under its worst-case minimization formulation. To this extent, our approach differs from the majority of the other approaches based on global optimization that characterize the feasible set of the sub-optimal problem. Indeed, MMIBBA

permits to find the controller that minimizes the worst-case H_∞ norm, which might be more interesting than having a set of sub-optimal controllers. In addition, MMIBBA provides a lower bound on the global minimum, which, if it is greater than one, enable to prove in a reliable way that no controller exists such that the H_∞ criteria are respected for all the values of uncertainties.

Contribution in optimization

The H_∞ synthesis problem, in order to be solved in a global way under its non-convex form, is formulated as a continuous min-max problem. Unfortunately, few attention has been given to the global resolution of this class of problems. The main difficulty lies in computing an enclosure of all the global maxima of a set of functions. To do so, the SIBBA is proposed, and the common acceleration techniques (monotonicity test and constraints propagation) are extended to the maximization of a set of function to improve the convergence. SIBBA provides bounds on the maximum objective function, which enables to integrate it directly in a classical branch and bound algorithm being MMIBBA.
As a semi infinite constraint can be reformulated as a maximization problem, SIBBA is also used to prove the feasibility or the infeasibility of a box. As a consequence, minmax problems subject to semi infinite constraints can be solved in a global way by MMIBBA, and by extension the robust structured synthesis problem which is expressed under this form.
The convergence of MMIBBA is improved by using the inheritance (or propagation) strategy. The efficiency of this strategy is illustrated on several minmax and SIP examples.

6.2 Prospects

This thesis proposes the necessary tools to solve the H_∞ problems globally under their initial non-convex formulation. However, these problems are difficult to solve, which has motivated the numerous different reformulations and resolution methods. To this extent, a global optimization approach is suited for small scale problems (few tunable parameters). In order to enable the resolution of larger problems, the convergence of MMIBBA can be improved by implementing advanced heuristics or using other methods to compute bounds than natural inclusion functions. In addition, the choice of the parameters of MMIBBA plays a major role for the convergence of the algorithm as illustrated by the benchmark results of Section 3. Further tests could be led to find good heuristics to tune these parameters.
Although the development of algorithms to solve minmax problems subject to semi infinite constraints is motivated by the resolution of the H_∞ problems, other control problems can be framed under this form. In this way a preliminary work [94] in cooperation with the National University of La Plata, given in Appendix B, shows how sliding mode control synthesis and analysis problems can be expressed as SIP and solved with MMIBBA.

Bibliography

[1] Systune webpage. https://fr.mathworks.com/help/slcontrol/ug/systune.html. Accessed: 2018-08-3.

[2] IEEE Standard for Interval Arithmetic. *IEEE Std. 1788*, 2015.

[3] Hirokazu Anai and Shinji Hara. Fixed-structure robust controller synthesis based on sign definite condition by a special quantifier elimination. In *Proceedings of the 2000 IEEE American Control Conference*, volume 2, pages 1312–1316, 2000.

[4] Hirokazu Anai and Shinji Hara. Linear programming approach to robust controller design by a quantifier elimination. In *Proceedings of the 41st IEEE Society of Instrument and Control Engineers of Japan Annual Conference*, volume 1, pages 656–661, 2002.

[5] Hirokazu Anai and Shinji Hara. A parameter space approach to fixed-order robust controller synthesis by quantifier elimination. *International Journal of Control*, 79(11):1321–1330, 2006.

[6] Daniel Ankelhed. *On low order controller synthesis using rational constraints*. PhD thesis, Linköping University, 2009.

[7] Pierre Apkarian. Nonsmooth μ-synthesis. *International Journal of Robust and Nonlinear Control*, 21(13):1493–1508, 2011.

[8] Pierre Apkarian, Minh Ngoc Dao, and Dominikus Noll. Parametric robust structured control design. *IEEE Transactions on Automatic Control*, 60(7):1857–1869, 2015.

[9] Pierre Apkarian and Dominikus Noll. Nonsmooth H_∞ synthesis. *IEEE Transactions on Automatic Control*, 51(1):71–86, 2006.

[10] Pierre Apkarian and Dominikus Noll. Worst-case stability and performance with mixed parametric and dynamic uncertainties. *International Journal of Robust and Nonlinear Control*, 27(8):1284–1301, 2017.

[11] Ignacio Araya, Bertrand Neveu, and Gilles Trombettoni. Exploiting common subexpressions in numerical csps. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, pages 342–357. Springer, 2008.

[12] Ignacio Araya, Victor Reyes, and Cristian Oreallana. More smear-based variable selection heuristics for ncsps. In *Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1004–1011, 2013.

[13] Charles Audet and Warren Hare. *Derivative-free and blackbox optimization*. Springer, 2017.

[14] Robert J Aumann and Sergiu Hart. *Handbook of game theory with economic applications*, volume 2. Elsevier, 1992.

[15] Maryam Babazadeh and Amin Nobakhti. Direct synthesis of fixed-order controllers. *IEEE Transactions on Automatic Control*, 60(10):2704–2709, 2015.

[16] Helio JC Barbosa. A genetic algorithm for min-max problems. In *Proceedings of the First IEEE International Conference on Evolutionary Computation and its Applications (EvCA96)*, pages 99–109, 1996.

[17] Helio JC Barbosa. A coevolutionary genetic algorithm for constrained optimization. In *Proceedings of the 1999 IEEE Congress on Evolutionary Computation (CEC)*, volume 3, pages 1605–1611, 1999.

[18] B Ross Barmish. Necessary and sufficient condition for quadratic stabilizability of an uncertain system. *J. Optimization Theory and Applications*, 46(4), August 1985.

[19] Eckart Baumann. Optimal centered forms. *BIT Numerical Mathematics*, 28(1):80–87, 1988.

[20] Binita Bhattacharjee, Panayiotis Lemonidis, William H Green Jr, and Paul I Barton. Global solution of semi-infinite programs. *Mathematical Programming*, 103(2):283–307, 2005.

[21] Shankar P Bhattacharyya and Lee H Keel. *Robust control: the parametric approach*. Prentice Hall, 1997.

[22] Jerry W Blankenship and James E Falk. Infinitely constrained optimization problems. *Journal of Optimization Theory and Applications*, 19(2):261–281, 1976.

[23] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*, volume 15. SIAM, 1994.

[24] Stephen Boyd, Martin Hast, and Karl Johan Åström. Mimo PID tuning via iterated LMI restriction. *International Journal of Robust and Nonlinear Control*, 2015.

[25] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[26] Richard P Braatz, Peter M Young, John C Doyle, and Manfred Morari. Computational complexity of μ calculation. *IEEE Transactions on Automatic Control*, 39(5):1000–1002, 1994.

[27] James V Burke, Didier Henrion, Adrian S Lewis, and Micheal L Overton. Hifoo-a matlab package for fixed-order controller design and H_∞ optimization. In *Fifth IFAC Symposium on Robust Control Design, IFAC Proceedings Volumes*, volume 39, pages 339–344. Elsevier, 2006.

[28] James V Burke, Adrian S Lewis, and Michael L Overton. A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

[29] Gilles Chabert and Luc Jaulin. Computing the pessimism of inclusion functions. *Reliable computing*, 13(6):489–504, 2007.

[30] Gilles Chabert and Luc Jaulin. Contractor programming. *Artificial Intelligence*, 173(11):1079–1100, 2009.

[31] I Lung Chien. IMC-PID controller design an extension. *IFAC Proceedings Volumes*, 21(7):147–152, 1988.

[32] Frank H Clarke. *Optimization and nonsmooth analysis*, volume 5. SIAM, 1990.

[33] George E Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In *Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern*, pages 134–183. Springer, 1975.

[34] Aaron M Cramer, Scott D Sudhoff, and Edwin L Zivi. Evolutionary algorithms for minimax problems in robust design. *IEEE Transactions on Evolutionary Computation*, 13(2):444–453, 2009.

[35] Tibor Csendes and Dietmar Ratz. Subdivision direction selection in interval methods for global optimization. *SIAM Journal on Numerical Analysis*, 34(3):922–938, 1997.

[36] Raquel Stella da Silva de Aguiar, Pierre Apkarian, and Dominikus Noll. Structured robust control against mixed uncertainty. *IEEE Transactions on Control Systems Technology*, 2017.

[37] Luiz Henrique De Figueiredo and Jorge Stolfi. Affine arithmetic: concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004.

[38] Hatim Djelassi and Alexander Mitsos. A hybrid discretization algorithm with guaranteed feasibility for the global solution of semi-infinite programs. *Journal of Global Optimization*, 68(2):227–253, 2017.

[39] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[40] Peter Dorato. Quantified multivariate polynomial inequalities. the mathematics of practical control design problems. *IEEE Control Systems*, 20(5):48–58, 2000.

[41] John Doyle. Analysis of feedback systems with structured uncertainties. In *IEE Proceedings D-Control Theory and Applications*, volume 129, pages 242–250, 1982.

[42] John C Doyle, Keith Glover, Pramod P Khargonekar, and Bruce A Francis. State-space solutions to standard H_2 and H_∞ control problems. *IEEE Transactions on Automatic Control*, 34(8):831–847, 1989.

[43] Gonzalez T Ferreres and Jean M Biannic. A μanalysis technique without frequency gridding. In *Proceedings of the IEEE 1998 American Control Conference*, volume 4, pages 2294–2298, 1998.

[44] Christodoulos A Floudas and Oliver Stein. The adaptive convexification algorithm: a feasible point method for semi-infinite programming. *SIAM Journal on Optimization*, 18(4):1187–1208, 2007.

[45] Thor I Fossen. *Marine control systems: Guidance, navigation and control of ships, rigs and underwater vehicles*. Marine Cybernetics Trondheim, 2002.

[46] Vincent Fromion and Gérard Scorletti. A theoretical framework for gain scheduling. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 13(10):951–982, 2003.

[47] Pascal Gahinet and Pierre Apkarian. A linear matrix inequality approach to H_∞ control. *International journal of Robust and Nonlinear control*, 4(4):421–448, 1994.

[48] Felix R Gantmacher. *The theory of matrices, vol. 2*, volume 220. Chelsea, New York, 1959.

[49] Philip J Goddard and Keith Glover. Controller approximation: approaches for preserving H_∞ performance. *IEEE transactions on Automatic Control*, 43(7):858–871, 1998.

[50] Alexandre Goldsztejn and Luc Jaulin. Inner and outer approximations of existentially quantified equality constraints. In *International Conference on Principles and Practice of Constraint Programming*, pages 198–212. Springer, 2006.

[51] Alexandre Goldsztejn, Claude Michel, and Michel Rueher. Efficient handling of universally quantified inequalities. *Constraints*, 14(1):117–135, 2009.

[52] Laureano González-Vega, Tomás Recio, Henri Lombardi, and Marie F Roy. Sturmhabicht sequences, determinants and real roots of univariate polynomials. *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 300–316, 1998.

[53] Graham C Goodwin, Stefan F Graebe, and Mario E Salgado. *Control system design*. Prentice Hall, 2001.

[54] Frédéric Goualard. How do you compute the midpoint of an interval? *ACM Transactions on Mathematical Software (TOMS)*, 40(2):11, 2014.

[55] Laurent Granvilliers. Adaptive bisection of numerical CSPs. *Principles and Practice of Constraint Programming*, pages 290–298, 2012.

[56] Laurent Granvilliers, Eric Monfroy, and Frédéric Benhamou. Symbolic-interval cooperation in constraint programming. In *Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, pages 150–166. ACM, 2001.

[57] Karolos M Grigoriadis and Robert E Skelton. Low-order control design for LMI problems using alternating projection methods. *Automatica*, 32(8):1117–1125, 1996.

[58] Serkan Gugercin and Athanasios C Antoulas. A survey of model reduction by balanced truncation and some new results. *International Journal of Control*, 77(8):748–766, 2004.

[59] Zeynep H Gümüş and Christodoulos A Floudas. Global optimization of nonlinear bilevel programming problems. *Journal of Global Optimization*, 20(1):1–31, 2001.

[60] Sangjin Han, Lee H Keel, and Shankar P Bhattacharyya. PID controller design with an H_∞ criterion. 51(4):400–405, 2018.

[61] Eldon Hansen. Bounding the solution of interval linear equations. *SIAM journal on Numerical Analysis*, 29(5):1493–1503, 1992.

[62] Martin Hast, Karl J Åström, Bo Bernhardsson, and Stephen Boyd. PID design by convex-concave optimization. In *Proceedings of the 2013 European Control Conference*, pages 4460–4465, 2013.

[63] Didier Henrion. LMI optimization for fixed-order H_∞ controller design. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, volume 5, pages 4646–4651, 2003.

[64] Rainer Hettich and Kenneth O Kortanek. Semi-infinite programming: theory, methods, and applications. *SIAM review*, 35(3):380–429, 1993.

[65] Milan Hladík. Bounds on eigenvalues of real and complex interval matrices. *Applied Mathematics and Computation*, 219(10):5584–5591, 2013.

[66] Milan Hladík, David Daney, and Elias Tsigaridas. Bounds on real eigenvalues and singular values of interval matrices. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2116–2129, 2010.

[67] Kozo Ichida and Yasuo Fujii. An interval arithmetic method for global optimization. *Computing*, 23(1):85–97, 1979.

[68] Luc Jaulin. *Solution globale et guarantie de problèmes ensemblistes: Application à l'estimation non léaire et à la commande robuste*. PhD thesis, Université de Paris Sud, 1994.

[69] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Éric Walter. *Applied Interval Analysis*. Springer London, 2001.

[70] Luc Jaulin and Éric Walter. Guaranteed tuning, with application to robust control and motion planning. *Automatica*, 32(8):1217 – 1221, 1996.

[71] Alireza Karimi and Gorka Galdos. Fixed-order H_∞ controller design for nonparametric models by convex optimization. *Automatica*, 46(8):1388–1394, 2010.

[72] Alireza Karimi, Gorka Galdos, and Roland Longchamp. Robust fixed-order H_∞ controller design for spectral models by convex optimization. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 921–926, 2008.

[73] R Baker Kearfott. An interval branch and bound algorithm for bound constrained optimization problems. *Journal of Global Optimization*, 2(3):259–280, 1992.

[74] R Baker Kearfott. Globsol user guide. *Optimization Methods & Software*, 24(4-5):687–708, 2009.

[75] R Baker Kearfott. *Rigorous global search: continuous problems*, volume 13. Springer Science & Business Media, 2013.

[76] Hassan K Khalil. *Nonlinear systems*. Prentice Hall, 2002.

[77] Polyxeni-Margarita Kleniati and Claire S Adjiman. Branch-and-sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. part i: Theoretical development. *Journal of Global Optimization*, 60(3):425–458, 2014.

[78] Olaf Knüppel. Profil/bias-a fast interval library. *Computing*, 53(3-4):277–287, 1994.

[79] Renato A Krohling and Leandro dos Santos Coelho. Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 36(6):1407–1416, 2006.

[80] Ailsa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.

[81] F Leibfritz. Compleib: Constrained matrix-optimization problem library, a collection of test examples for nonlinear semidefinite programs, control system design and related problems. Technical report, 2003.

[82] Michael Lerch, German Tischler, Jürgen Wolff Von Gudenberg, Werner Hofschuster, and Walter Krämer. FILIB++, a fast interval library supporting containment computations. *ACM Transactions on Mathematical Software (TOMS)*, 32(2):299–324, 2006.

[83] Baiquan Lu, Yuan Cao, Min jie Yuan, and Jianzhen Zhou. Reference variable methods of solving min–max optimization problems. *Journal of Global Optimization*, 42(1):1–21, 2008.

[84] Stefano Malan, Mario Milanese, and Michele Taragna. Robust tuning for PID controllers with multiple performance specifications. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 3, pages 2684–2689, 1994.

[85] Stefano Malan, Mario Milanese, and Michele Taragna. Robust analysis and design of control systems using interval arithmetic. *Automatica*, 33(7):1363–1372, 1997.

[86] Stefano Malan, Mario Milanese, Michele Taragna, and Jürgen Garloff. B^3 algorithm for robust performances analysis in presence of mixed parametric and dynamic perturbations. In *Proceedings of the 31st IEEE Conference on Decision and Control*, pages 128–133, 1992.

[87] Frederic Messine. Deterministic global optimization using interval constraint propagation techniques. *RAIRO-Operations Research*, 38(4):277–293, 2004.

[88] Alexander Mistos. A test set for semi-infinite programs. http://web.mit.edu/mitsos/www/pubs/siptestset.pdf. Accessed: 2018-06-20.

[89] Alexander Mitsos. Global optimization of semi-infinite programs via restriction of the right-hand side. *Optimization*, 60(10-11):1291–1308, 2011.

[90] Dominique Monnet, Jordan Ninin, and Benoît Clement. Global optimization of H_∞ problems: Application to robust control synthesis under structural constraints. In *International Conference on Mathematical Aspects of Computer and Information Sciences*, pages 550–554. Springer, 2015.

[91] Dominique Monnet, Jordan Ninin, and Benoît Clement. A global optimization approach to structured regulation design under H_∞ constraints. In *Proceedings of the IEEE 55th Conference on Decision and Control*, pages 658–663, 2016.

[92] Dominique Monnet, Jordan Ninin, and Benoît Clement. A global optimization approach to H_∞ synthesis with parametric uncertainties applied to auv control. *Proceedings of 20th IFAC World Congress, IFAC-PapersOnLine*, 50(1):3953–3958, 2017.

[93] Dominique Monnet, Jordan Ninin, and Luc Jaulin. Computing an inner and an outer approximation of the viability kernel. *Reliable Computing*, 22:138–148, 2016.

[94] Dominique Monnet, Juan Luis Rosendo, Hernán De Battista, Benoît Clement, Fabricio Garelli, and Jordan. Ninin. A global optimization approach for non-linear sliding mode control analysis and design. In *Accepted to 9th IFAC Symposium on Robust Control Design ROCOND*. Elsevier, September 2018.

[95] Ramon E Moore. *Interval analysis*. Prince-Hall, Englewood Cliffs, NJ, 1966.

[96] D Nesic. Two algorithms arising in analysis of polynomial models. In *Proceedings of the 1998 IEEE American Control Conference*, volume 3, pages 1889–1893, 1998.

[97] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

[98] Arnold Neumaier. *Interval methods for systems of equations*, volume 37. Cambridge university press, 1990.

[99] Jordan Ninin, Frédéric Messine, and Pierre Hansen. A reliable affine relaxation method for global optimization. *4OR*, 13(3):247–277, 2015.

[100] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 2006.

[101] Jean-Marie Normand, Alexandre Goldsztejn, Marc Christie, and Frédéric Benhamou. A branch and bound algorithm for numerical max-csp. *Constraints*, 15(2):213–237, 2010.

[102] Aidan O'Dwyer. *Handbook of PI and PID controller tuning rules*. Imperial College Press, 2009.

[103] Andy Packard, John Doyle, and Gary Balas. Linear, multivariable robust control with a μ perspective. *Journal of Dynamic Systems, Measurement, and Control*, 115(2B):426–438, 1993.

[104] Dimitri Peaucelle. *Formulation Générique de Problèmes en Analyse et Commande Robuste par les Fonctions de Lyapunov Dépendant des Paramètres*. PhD thesis, Université Toulouse III – Paul Sabatier, France, July 2000.

[105] Dimitri Peaucelle, Denis Arzelier, Olivier Bachelier, and Jacques Bernussou. A new robust D-stability condition for real convex polytopic uncertainty. *Systems and Control Letters*, 40(1):21–30, May 2000.

[106] Dimitri Peaucelle, Denis Arzelier, Didier Henrion, and Frederic Gouaisbaut. Quadratic separation for feedback connection of an uncertain matrix and an implicit linear transformation. *Automatica*, 43:795–804, 2007.

[107] Stefan Ratschan. Approximate quantified constraint solving by cylindrical box decomposition. *Reliable Computing*, 8(1):21–42, 2002.

[108] Helmut Ratschek and Jon Rokne. *New computer methods for global optimization*. Horwood, Chichester, 1988.

[109] Juan L Rosendo, Benoît Clement, and Fabricio Garelli. Sliding mode reference conditioning for path following applied to an auv. In *10th IFAC Conference on Control Applications in Marine Systems*, volume 49, pages 8–13, 2016.

[110] Juan Luis Rosendo, Dominique Monnet, Benoît Clement, Fabricio Garelli, and Jordan Ninin. Control of an autonomous underwater vehicle subject to robustness constraints. In *Accepted to 9th IFAC Symposium on Robust Control Design ROCOND*. Elsevier, September 2018.

[111] Siegfried M Rump. INTLAB–interval laboratory. In *Developments in reliable computing*, pages 77–104. Springer, 1999.

[112] Berc Rustem and Melendres Howe. *Algorithms for worst-case design and applications to risk management*. Princeton University Press, 2002.

[113] Mahdieh S Sadabadi and Dimitri Peaucelle. From static output feedback to structured robust static output feedback: A survey. *Annual reviews in control*, 42:11–26, 2016.

[114] Arash Sadeghzadeh, Hamidreza Momeni, and Alireza Karimi. Fixed-order H_∞ controller design for systems with ellipsoidal parametric uncertainty. *International Journal of Control*, 84(1):57–65, 2011.

[115] Masami Saeki. Fixed structure PID controller design for standard H_∞ control problem. *Automatica*, 42(1):93–100, 2006.

[116] Nikolaos V Sahinidis. Baron: A general purpose global optimization software package. *Journal of Global Optimization*, 8(2):201–205, 1996.

[117] Miguel Á Sainz, Pau Herrero, Joaquim Armengol, and Josep Vehí. Continuous minimax optimization using modal intervals. *Journal of Mathematical Analysis and Applications*, 339(1):18–30, 2008.

[118] Gerard Scorletti and Laurent El Ghaoui. Improved LMI conditions for gain scheduling and related control problems. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 8(10):845–877, 1998.

[119] Gilberto AS Segundo, Renato A Krohling, and Rodrigo C Cosme. A differential evolution approach for solving constrained min-max optimization problems. *Expert Systems with Applications*, 39(18):13440–13450, 2012.

[120] Yuhui Shi and Renato A Krohling. Co-evolutionary particle swarm optimization to solve min-max problems. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, volume 2, pages 1682–1687, 2002.

[121] Emile Simon. *A perspective for optimization in systems and control: from LMIs to derivative-free methods*. PhD thesis, Université catholique de Louvain, 2012.

[122] Stig Skelboe. Computation of rational interval functions. *BIT Numerical Mathematics*, 14(1):87–95, 1974.

[123] Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: analysis and design*, volume 2. Wiley New York, 2007.

[124] Sophie Tarbouriech, Germain Garcia, João Manoel Gomes da Silva Jr, and Isabelle Queinnec. *Stability and stabilization of linear systems with saturating actuators*. Springer, 2011.

[125] Xuan-Ha Vu, Hermann Schichl, and Djamila Sam-Haroud. Using directed acyclic graphs to coordinate propagation and search for numerical constraint satisfaction problems. In *Proceedings of the 16th International Conference on Tools with Artificial Intelligence*, pages 72–81. IEEE, 2004.

[126] Rui Yang. *Modeling and robust control approach for autonomous underwater vehicles*. PhD thesis, Université de Bretagne Occidentale-Brest, 2015.

[127] Rui Yang, Benoît Clement, Ali Mansour, Ming Li, and Nailong Wu. Modeling of a complex-shaped underwater vehicle for robust control scheme. *Journal of Intelligent & Robotic Systems*, 2015.

[128] Rui Yang, Benoît Clement, Ali Mansour, Ming Li, and Nailong Wu. Robust heading control and its application to ciscrea underwater vehicle. In *IEEE OCEANS'15*, 2015.

[129] Peter M Young and John C Doyle. Computation of mu with real and complex uncertainties. In *Proceedings of the 29th IEEE Conference on Decision and Control*, pages 1230–1235, 1990.

[130] Peter M Young, Matthew P Newlin, and John C Doyle. μ-analysis with real parametric uncertainty. In *Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1251–1256, 1991.

[131] George Zames. Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. *IEEE Transactions on Automatic Control*, 26(2):301–320, 1981.

[132] Kemin Zhou. A comparative study of H_∞ controller reduction methods. In *Proceedings of the IEEE 1995 American Control Conference*, volume 6, pages 4015–4019, 1995.

[133] Kemin Zhou and John C Doyle. *Essentials of robust control*, volume 104. Prentice Hall, 1998.

[134] Kemin Zhou, John Comstock Doyle, and Keith Glover. *Robust and optimal control*, volume 40. Prentice Hall, 1996.

[135] John G Ziegler and Nathaniel B Nichols. Optimum settings for automatic controllers. *Trans. ASME*, 64(11), 1942.

[136] Shen Zuhe, A Neumaier, and MC Eiermann. Solving minimax problems by interval methods. *BIT Numerical Mathematics*, 30(4):742–751, 1990.

List of Figures

List of Algorithms

Appendices

Appendix A

Optimization benchmark problems

This appendix gathers the benchmark problems used to obtained the numerical results presented in Chapter 3.

A.1 Semi infinite problems

The SIP problems are taken from [88]. Only the first twelve examples are considered, and reproduced here. The problems are formulated as

$$\begin{cases} \min\limits_{x \in \mathbb{X}} & f(x) \\ \text{s. t.} & g(x,y) \leq 0,\ \forall y \in \mathbb{Y} \end{cases}$$

- Problem 1.

$$\begin{aligned} &f(x) = x_1^2/43 + x_2^2 + x_1/2 \\ &g(x,y) = (1 - x_2^2 y^2)^2 - x_1 y^2 - x_2^2 + x2 \\ &\mathbb{X} = [-1000, 1000]^2 \\ &\mathbb{Y} = [0,1] \end{aligned}$$

- Problem 2.

$$\begin{aligned} &f(x) = \exp(x_1) + \exp(x_2) + exp(x_3) \\ &g(x,y) = 1/(1+y^2) - x1 - x2y - xy^2 \\ &\mathbb{X} = [-1000, 1000]^2 \\ &\mathbb{Y} = [0,1] \end{aligned}$$

- Problem 3.

$$\begin{aligned} &f(x) = (x_1 - 2x_2 + 5x_2^2 - x_2^2 x_2 - 13)^2 + (x_1 - 14x_2 + x_2^2 + x_2^3 - 29)^2 \\ &g(x,y) = x_1^2 + 2x2y^2 + \exp(x1 + x2) - \exp(y) \\ &\mathbb{X} = [-1000, 1000]^2 \\ &\mathbb{Y} = [0,1] \end{aligned}$$

- Problem 4.

$$f(x) = x_1^2 + x_2^2 + x_3^2$$
$$g(x,y) = x_1(y_1 + y_2^2 + 1) + x_2(y_1 y_2 - y_2^2) + x_3(y_1 y_2 + y_2^2 + y_2) + 1$$
$$\mathbb{X} = [-1000, 1000]^3$$
$$\mathbb{Y} = [0,1]^2$$

- Problem 5.

$$f(x) = x_1 + x_2/2 + x_3/2 + x_4/3 + x_5/4 + x_6/3$$
$$g(x,y) = \exp(y_1^2 + y_2^2) - x_1 - x_2 y_1 - x_3 y_2 - x_4 y_1^2 - x_5 y_1 y_2 - x_6 y_2^2$$
$$\mathbb{X} = [-1000, 1000]^6$$
$$\mathbb{Y} = [0,1]^2$$

- Problem 6.

$$f(x) = -4x_1 - 2/3(x_4 + x_6)$$
$$g(x,y) = x_1 + x_2 y_1 + x_3 y_2 + x_4 y_1^2 + x_5 y_1 y_2 + x_6 y_2^2 - 3 - (y_1^2 - y_2^2))^2$$
$$\mathbb{X} = [-1000, 1000]^6$$
$$\mathbb{Y} = [-1,1]^2$$

- Problem 7.

$$f(x) = x_2$$
$$g(x,y) = -(x_1 - y)^2 - x_2$$
$$\mathbb{X} = [0,1] \times [-1000, 1000]$$
$$\mathbb{Y} = [-1,1]$$

- Problem 8.

$$f(x) = x_2$$
$$g(x,y) = 2x_1^2 y^2 - y^4 + x_1^2 - x_2$$
$$\mathbb{X} = [0,1] \times [-1000, 1000]$$
$$\mathbb{Y} = [-1,1]$$

- Problem 9.

$$f(x) = x_1 + x_2/2 + x_3/3$$
$$g(x,y) = \exp(y-1) - x_1 - x_2 y - x_3 y^2$$
$$\mathbb{X} = [-1000, 1000]^3$$
$$\mathbb{Y} = [0,1]$$

- Problem 10.

$$f(x) = x_1 + x_2/2 + x_3/3 + x_4/4 + x_5/5 + x_6/6$$
$$g(x,y) = \exp(y-1) - x_1 - x_2 y - x_3 y^2 - x_4 y^3 - x_5 y^4 - x_6 y^5$$
$$\mathbb{X} = [-1000, 1000]^6$$
$$\mathbb{Y} = [0,1]$$

- Problem 11.

$$f(x) = -1/((x_1-4)^2+(x_2-4)^2+0.1) - 1/((x_1-1)^2+(x_2-1)^2+0.2)$$
$$-1/((x_1-8^2)+(x_2-8^2+0.2))$$
$$g(x,y) = 0.1-(x_1-y)^2-(x_2-y)^2$$
$$\mathbb{X} = [-1000,1000]^2$$
$$\mathbb{Y} = [-10,10]$$

- Problem 12.

$$f(x) = 10-x$$
$$g(x,y) = y^2/(1+\exp(-40(x-y))) + x - y - 2$$
$$\mathbb{X} = [0,6]$$
$$\mathbb{Y} = [2,6]$$

A.2 Minmax problems

The unconstrained minmax problems 1-16 are taken from [112] and the constrained minmax problems 17-20 from [119].
The minmax problems are expressed as:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{X}} \max\limits_{y\in\mathbb{Y}} & f(x,y) \\ \text{s. t.} & g_i(x,y) \leq 0. \end{array} \tag{A.1}$$

- Problem 1.

$$f(x,y) = 5\sum_{i=1}^{2} x_i^2 - \sum_{i=1}^{2} y_i^2 + x_1(-y_1+y_2+5) + x_2(y_1-y_2+3)$$
$$\mathbb{X} = [-100,100]^2$$
$$\mathbb{Y} = [-5,5]^2$$

- Problem 2.

$$f(x,y) = 4(x_1-2)^2 - 2y_1^2 + x_1^2y_1 - y_2^2 + 2x_2^2y_2$$
$$\mathbb{X} = [-100,100]^2$$
$$\mathbb{Y} = [-5,5]^2$$

- Problem 3.

$$f(x,y) = x_1^4y_2 + 2x_1^3y_1 - x_2^2y_2(y_2-3) - 2x_2(y_1-3)^2$$
$$\mathbb{X} = [-100,100]^2$$
$$\mathbb{Y} = [0,3]^2$$

- Problem 4.

$$f(x,y) = -\sum_{i=1}^{3}(y_i-1)^2 + \sum_{i=1}^{2}(x_i-1)^2 + y_3(x_2-1) + y_1(x_1-1) + y_2x_1x_2$$
$$\mathbb{X} = [-100,100]^2$$
$$\mathbb{Y} = [-3,3]^3$$

- Problem 5.

$$f(x,y) = -(x_1-1)y_1 - (x_2-2)y_2 - (x_3-1)y_3 + 2x_1^2 + 3x_2^2 + x_3^2 - \sum_{i=1}^{3} y_i^2$$
$$\mathbb{X} = [-100,100]^3$$
$$\mathbb{Y} = [-1,1]^3$$

- Problem 6.

$f(x,y) = y_1(x_1^2 - x_2 + x_3 - x_4 + 2) + y_2(-x_1 + 2x_2^2 - x_3^2 + 2x_4 + 1) + y_3(2x_1 - x_2 + 2x_3 - x_4^2 + 5) +$
$5x_1^2 + 4x_2^2 + 3x_3^2 + 2x_4^2 - \sum_{i=1}^{3} y_i^2$
$\mathbb{X} = [-100,100]^4$
$\mathbb{Y} = [-2,2]^3$

- Problem 7.

$$f(x,y) = 2x_5x_1 + 3x_4x_2 + x_5x_3 + 5x_4^2 + 5x_5^2 - x_4(y_4 - y_5 - 5) + x_5(y_4 - y_5 + 3)$$
$$+ \sum_{i=1}^{3}(y_i(x_i^2 - 1)) - \sum_{i=1}^{5} y_i^2$$
$$\mathbb{X} = [-100,100]^5$$
$$\mathbb{Y} = [-3,3]^5$$

- Problem 8.

$$f(x,y) = \frac{2}{3}(\frac{1}{2}x_1 + (4+x_1)y_1)^2$$
$$\mathbb{X} = [-100,100]$$
$$\mathbb{Y} = [-2,2]$$

- Problem 9.

$$f(x,y) = \frac{1}{2}((2x_2 + 4y_1 + x_1y_1)^2 + (x_1 + 2x_1y_1 + x_2y_2)^2)$$
$$\mathbb{X} = [-100,100]^2$$
$$\mathbb{Y} = [-5,5]^2$$

- Problem 10.

$$f(x,y) = \frac{1}{2}((5x_2 + 5x_1 + 3x_1y_1)^2 + (2x_1 + 5x_1y_1 + 3x_2y_2)^2)$$
$$\mathbb{X} = [-100,100]^2$$
$$\mathbb{Y} = [-5,5]^2$$

- Problem 11.

$$f(x,y) = \frac{1}{2}((x_1x_2 - y_1(1 - x_3x_4))^2 + (x_2x_3 - y_2(2 + x_4x_1))^2 + \sum_{i=1}^{2} y_i^2)$$
$$\mathbb{X} = [-100,100]^4$$
$$\mathbb{Y} = [-5,5]^2$$

- Problem 12.

$$f(x,y) = \sum_{i=1}^{3} y_i^2 + y_1(x_1^2 - x_2 + x_3 - x_4 + 2) + y_2(-x_1 + 2x_2^2 - x_3^2 + 2x_4 - 10)$$
$$+y_3(2x_1 - x_2 + 2x_3 - x_4^2 - 5) + 5\sum_{i=1}^{4} x_i^2$$
$$\mathbb{X} = [-100,100]^4$$
$$\mathbb{Y} = [-2,2]^3$$

- Problem 13.

$f(x,y) = \sum_{i=1}^{4} y_i^2 + y_1(x_1^2 - x_2 + x_3 - x_4 + 2) + y_2(-x_1 + x_2^2 - x_3^2 + 2x_4 - 10)$
$+y_3(2x_1 - x_2 + 2x_3 - x_4^2 - 5) + 5y_4(x_1^2) + 5\sum_{i=3}^{4} x_i^2$
$\mathbb{X} = [-100, 100]^4$
$\mathbb{Y} = [-2, 2]^4$

- Problem 14.

$f(x,y) = \sum_{i=1}^{4} y_i^2 + y_1(x_1^2 - 2.2x_2 + x_3 - 10x_4 + 10) + y_2(-2x_1 + 2x_2^2 - x_3^2 + 3x_4 - 10)$
$+y_3(2x_1 - x_2 + 6x_3 - x_4^2 - 5) + 5y_4(x_1^2 + x_2^2) + 5\sum_{i=3}^{4} x_i^2$
$\mathbb{X} = [-100, 100]^4$
$\mathbb{Y} = [-2, 2]^4$

- Problem 15.

$f(x,y) = \sum_{i=1}^{4} y_i^2 + y_1(x_1^2 - 2.2x_2 + x_3 - 10x_4 + 10) + y_2(-2x_1 + 2x_2^2 - x_3^2 + 3x_4 - 10)$
$+y_3(2x_1 - x_2 + 5.91x_3 - x_4^2 - 15) + 5y_4(x_1^2 + x_2^2) + 5\sum_{i=3}^{4} x_i^2$
$\mathbb{X} = [-100, 100]^4$
$\mathbb{Y} = [-2, 2]^4$

- Problem 16.

$f(x,y) = \frac{1}{2}\sum_{i=1}^{4} y_i^2 + y_1(x_1^2 - 2x_2 + x_3 - 10x_4 + 2) + y_2(-2x_1 + 2x_2^2 - x_3^2 + 3x_4 - 5)$
$+y_3(2x_1 - x_2 + 5x_3 - x_4^2 + 2) + y_4(x_1^2 + x_2^2 + \sum_{i=3}^{4} x_i^2)$
$\mathbb{X} = [-100, 100]^4$
$\mathbb{Y} = [-2, 2]^4$

- Problem 17.

$f(x,y) = (\cos(y) + \cos(2y + x))^2$
$g_1(x,y) = y - x(x + 6.28)$
$g_2(x,y) = y - x(x - 6.28)$
$\mathbb{X} = [-3.14, 3.14]$
$\mathbb{Y} = [-3.14, 3.14]$

- Problem 18.

$f(x,y) = x^2 + y^2 + 2xy - 20x - 20y + 100$
$g_1(x,y) = -(x-5)^2 - (y-3)^2 + 4$
$g_2(x,y) = (x-5)^2 + (y-3)^2 - 16$
$\mathbb{X} = [0, 6]$
$\mathbb{Y} = [2, 8]$

- Problem 19.

$$
\begin{aligned}
&f(x,y) = \sin(x)^2 - x\cos(y) + 2\sin(x) - \cos(y)^2 + y - 1\\
&g_1(x,y) = -x^2 - y^2 + 25\\
&\mathbb{X} = [-5,5]\\
&\mathbb{Y} = [-5,5]
\end{aligned}
$$

- Problem 20.

$$
\begin{aligned}
&f(x,y) = \frac{(x_1+y_1)(x_2+y_2)}{1+x_1y_1+x_2y_2}\\
&g_1(x,y) = x_1^2 + x_2^2 - 100\\
&g_2(x,y) = y_1 - x_1\\
&g_3(x,y) = y_2 - x_2\\
&\mathbb{X} = [0,10]^2\\
&\mathbb{Y} = [0,10]^2
\end{aligned}
$$

Appendix B

Preliminary work on sliding mode control

A global optimization approach for non-linear sliding mode control analysis and design ⋆

Dominique Monnet * **Juan Luis Rosendo** **
Hernán De Battista ** **Benoit Clement** * **Jordan Ninin** *
Fabricio Garelli **

* *ENSTA Bretagne / Lab-STICC UMR CNRS 6285, 2 rue Francois Verny, F-29200 Brest, France, (e-mail: dominique.monnet@ensta-bretagne.org)*
** *GCA, LEICI, University of La Plata (UNLP), 1900 La Plata, Argentina, (e-mail: juanluisrosendo@gmail.com).*

Abstract: The design of sliding mode (SM) comprises the selection of a sliding manifold on the state space and a switching logic. The sliding manifold design is associated with the desired dynamics and closed loop specifications, whereas the switching logic is designed to drive and keep the state on the prescribe manifold. The classical design can lead to over or underestimation of the sliding domain, the closed loop robustness and the necessary control power. Here the design of SM is addressed from the global optimization approach using interval arithmetic. A solution to the analysis and synthesis problems of SM design is provided, where the necessary and sufficient conditions are fulfilled in a guaranteed way. For the analysis problem the proposed methodology allows checking sliding mode behaviour over given state domain and parameter sets. For the synthesis problem, the methodology allows designing the sliding manifold and switching logic with a given optimization criterion. The methodology is illustrated with a concluding example.

Keywords: Sliding Modes, Robust control design, Global optimization, Interval analysis.

1. INTRODUCTION

The application of SM to the control of non linear system is well known. Several examples of application could be found in the literature (Khalil (2002), Sira-Ramrez (1993), Rosendo et al. (2016)). The behaviour of this kind of control is composed of two phases. The first phase consists in reaching the sliding surface, and the second one in sliding over it (Utkin et al. (2009)). The design of this control requires choosing an adequate switching function and an appropriate sliding surface, in accordance with the desired dynamics and the SM establishing conditions.

Knowledge of state and parameter excursions are essential in the SM control design. Usually the states and parameters imperfectly known are estimated by their maximal values, and then the resulting control design is tested through simulation. However, even a great number simulations from different initial conditions cannot prove in a guaranteed way that the control law satisfied the sliding condition over the state space. In addition, the design parameters chosen by the operator may not be optimal with respect to criteria such as energy consumption.

In this paper, a control design method based on global optimization and interval analysis techniques is proposed. As a result we derive a method to check in a guaranteed way if the SM necessary and sufficient conditions are fulfilled over bounded state domain and parameter ranges. Concerning the synthesis SM problem, our methodology provides an optimized design based on a given criterion (such as minimal energy consumption, or maximal possible dynamic of the system). Contrary to stochastic methods Wu et al. (2012); Niu et al. (2005); Li et al. (2014) which model uncertain systems as a finite set of deterministic ones, our approach enables to consider continuous uncertainties i.e. to consider an infinite set of systems. However, our method does not applies to time delay systems.

Interval analysis has been applied in the context of SM by Rauh and Aschemann (2012) and Senkel et al. (2014). Their objectives were to obtain online controllers based on interval arithmetic, where the control amplitude is continuously adapted. On the other hand, our approach uses the traditional SM design but we add robustness to the design and give guarantees of the proposed solutions for any initial condition over the given domain.

This paper is organized as follows: Section 2 recalls the SM theory and its associated analysis and synthesis problems. Section 3 introduces global optimization tools and formulates analysis and synthesis problems as optimization ones. Section 4 illustrates our approach with an example. Finally, Section 5 provides some comments and future works.

⋆ This research is supported by DGA (French Defense Procurement Agency), the city of Brest (Brest Metropole), CONICET (PIP0837) and UNLP(I216) (Argentina).

2. SM CONTROL THEORY

The sliding modes were originally developed for dynamic systems whose essential open-loop behavior can be modeled with ordinary differential equations (Utkin et al. (2009)). In these systems, it is possible to determinate a robust closed-loop dynamics by applying a discontinuous control action. According to the sign of a switching function, the control signal can take one of two different values, leading to a discontinuous control law with an associated manifold on the state-space (sliding surface). The idea is to enforce the state to reach the prescribed sliding surface and then to slide on it through a very fast switching action. Once this particular mode of operation is established, known as sliding mode, the prescribed manifold imposes the new and desired system dynamics. Among other attractive features sliding regimes are easy to implement, reduce the order of the system dynamics, and provide robustness to matched uncertainties and external disturbances.

The design procedure consists of two stages. First, the equation of the manifold where the system slides is selected in accordance with some performance criterion for the desired dynamics. Then, the discontinuous control should be found such that the system states reach the manifold and sliding mode exists on this manifold.

In order to present the theory, let us consider the dynamic system:

$$\begin{cases} \dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})u \\ y = h(\mathbf{x}) \end{cases} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the system state, u is the control signal, y is the output system, and $f(\mathbf{x})$, $g(\mathbf{x})$, $h(\mathbf{x})$ are vector fields in $\mathbb{R}^n$. The variable structure control law is defined as

$$u = \begin{cases} u^- & if \quad \sigma(\mathbf{x}) < \mathbf{0} \\ u^+ & if \quad \sigma(\mathbf{x}) > \mathbf{0} \end{cases} \tag{2}$$

according to the sign of the auxiliary output $\sigma(\mathbf{x})$. The sliding surface S is defined as the manifold where the auxiliary output, also called switching function, vanishes:

$$S = \{\mathbf{x} \in \mathbb{R}^n \mid \sigma(\mathbf{x}) = \mathbf{0}\}. \tag{3}$$

As a result of the switching policy in (2), the reaching condition

$$\begin{cases} \dot{\sigma}(\mathbf{x}) < \mathbf{0} & if \quad \sigma(\mathbf{x}) > \mathbf{0} \\ \dot{\sigma}(\mathbf{x}) > \mathbf{0} & if \quad \sigma(\mathbf{x}) < \mathbf{0} \end{cases} \tag{4}$$

locally holds on both sides of the surface, a switching sequence of very high frequency (ideally infinite) occurs, constraining the system state trajectory to slide on S.

For sliding motion to exist on S (i.e. for satisfying condition (4)), the auxiliary output $\sigma(\mathbf{x})$ must have unitary relative degree with respect to the discontinuous signal, i.e. its first derivative must explicitly depend on u (Utkin et al. (2009)).

Also, it is possible to define the ideal sliding mode using the equivalent control concept. Taking the invariant conditions over the SM surface, we get:

$$\begin{cases} \sigma(\mathbf{x}) = 0 \\ \dot{\sigma}(\mathbf{x}) = \frac{d\sigma}{d\mathbf{x}}\dot{\mathbf{x}} = L_{f+gu_{eq}}\sigma = L_f\sigma + L_g\sigma u_{eq} = 0 \end{cases} \tag{5}$$

where the generic operator $L_f h(x) : \mathbb{R}^n \to \mathbb{R}$ (directional or Lie derivative) denotes the derivative of a scalar field $h(x) : \mathbb{R}^n \to \mathbb{R}$ in the direction of a vector field $f(x) : \mathbb{R}^n \to \mathbb{R}^n$

$$L_f h(x) = \frac{\partial h}{\partial x} f(x). \tag{6}$$

From (5) is possible to obtain $u_{eq}(\mathbf{x})$ a soft control law which makes S an invariant subset.

$$u_{eq}(\mathbf{x}) = -\frac{L_f\sigma}{L_g\sigma} \tag{7}$$

Following this approach is possible to arrive to the necessary and sufficient condition for the SM. It is observed in (7) that $L_g\sigma \neq 0$ is necessary for the existence of u_{eq} and, therefore of SM. Furthermore, a necessary and sufficient condition for the local existence of the sliding mode over S can be derived from (4) and (5). If we consider (without loss of generality) $u^+ > u^-$ it must hold:

$$u^-(\mathbf{x}) < u_{eq}(\mathbf{x}) < u^+(\mathbf{x}) \tag{8}$$

From (8), $u_{eq}(\mathbf{x})$ can be interpreted as an average control action between the maximal and minimal of the system.

From the control designer point of view, it is possible to divide the SM control design into two separate problems:

Problem 1. *SM analysis problem: Given a desired sliding surface $\sigma(\mathbf{x}, \mathbf{k})$ with $\mathbf{x}$ states of the system and $\mathbf{k}$ a vector of fixed tuning parameters (σ with relative degree one with respect to the discontinuous signal u). Verify if u_{eq} fulfills condition given by (8).*

Problem 2. *SM synthesis problem: Given a system with constrained control actions (u^+ and u^-), and an expression of $\sigma(\mathbf{x}, \mathbf{k})$ with $\mathbf{x}$ states of the system and $\mathbf{k}$ a vector of free tuning parameters (σ with relative degree one with respect to the discontinuous signal u). Find the best possible sliding surface σ ($\mathbf{k}$ values) according to a design criterion, which fulfill condition given by (8).*

3. GLOBAL OPTIMIZATION APPROACH

Let us consider a continuous constrained optimization problem formulated as:

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbb{R}^n} & f(\mathbf{x}) \\ \text{subject to} & C(\mathbf{x}) \leq 0, \end{cases} \tag{9}$$

where f is the objective function which maps $\mathbb{R}^n$ into $\mathbb{R}$, $\mathbf{x} \in \mathbb{R}^n$ is the optimization variable, and C is a function that maps $\mathbb{R}^n$ into $\mathbb{R}$ used to define a subset of $\mathbb{R}^n$ in which the solution is searched. The solution, also called the minimizer, is denoted as $\mathbf{x}^*$ and is the point where f is minimum over the set defined by $\{\mathbf{x} \in \mathbb{R}^n, C(\mathbf{x}) \leq 0\}$. The minimum is denoted as $f^* = f(\mathbf{x}^*)$. From the definition of the minimum, Property (10) holds.

$$\forall \mathbf{x} \in \mathbb{R}^n \text{ such as } C(\mathbf{x}) \leq 0,\ f(\mathbf{x}) \geq f^*. \tag{10}$$

If f and C are not convex functions, local optimization techniques have no warranty to converge to the global solution $\mathbf{x}^*$. On the other hand, global optimization methods converge to the global minimum and provide an enclosure $[\underline{f^*}, \overline{f^*}]$ of f^*. One well-known technique from global optimization is the Branch and Bound algorithm based on interval arithmetic Kearfott (1992).

3.1 Branch and Bound based on interval arithmetic

In order to present the Interval Branch and Bound Algorithm (IBBA), several definitions must be given.

Definition 1. *An interval $\boldsymbol{x}$ is a closed connected subset of $\mathbb{R}$ (Moore et al. (2009)), described by its endpoints $\underline{x}$ and $\overline{x}$:*

$$\boldsymbol{x} = [\underline{x}, \overline{x}] = \{x \mid \underline{x} \leq x \leq \overline{x}\},$$
$$\text{with } \underline{x} \in \mathbb{R} \cup \{-\infty\} \text{ and } \overline{x} \in \mathbb{R} \cup \{+\infty\}$$

The set of real intervals is denoted by $\mathbb{IR}$ and the set of n-dimensional interval vectors, also called boxes, is denoted by $\mathbb{IR}^n$.

Definition 2. *Let $\boldsymbol{x} \in \mathbb{IR}^n$ be a box. An inclusion function $[f]$ of f maps $\mathbb{IR}^n$ into $\mathbb{IR}$ and respects the following property:*

$$f(\boldsymbol{x}_i) = \{f(x),\ x \in \boldsymbol{x}_i\} \subseteq [f](\boldsymbol{x}_i).$$

Interval arithmetic extends common operators ($+$, $-$, $\times$, *sin*, *cos*, exp, log,...) to $\mathbb{IR}$ and provide inclusion function of most of analytic functions. Let us suppose that inclusion functions of f and C can be defined, and the $\mathbf{x}^*$ is searched in $\mathbb{X} \in \mathbb{IR}^n$. The IBBA computes a guaranteed lower bound $\underline{f}$ and an upper bound $\overline{f}$ of f^*. To do so, IBBA repeatedly bisects $\mathbb{X}$ in smaller boxes $\boldsymbol{x_i}$ and discards them if it is proven that $\mathbf{x}^* \notin \boldsymbol{x_i}$. This happens if the constraint is not satisfied over $\boldsymbol{x}_i$:

$$\begin{aligned} \underline{[C](\boldsymbol{x}_i)} > 0 &\iff \forall \mathbf{x} \in \boldsymbol{x}_i, C(\mathbf{x}) > 0, \\ &\iff \mathbf{x}^* \notin \boldsymbol{x}_i, \end{aligned} \tag{11}$$

or if a feasible point $\tilde{x}$ has been found such that any points in $\boldsymbol{x}_i$ can provide a better feasible solution:

$$\underline{[f](\boldsymbol{x}_i)} > f(\tilde{x}) \geq f^* \iff \mathbf{x}^* \notin \boldsymbol{x}_i. \tag{12}$$

The IBBA stops when the distance between $\underline{f}$ and $\overline{f}$ reaches the desired precision d, with

$$\underline{f} = \min_i \underline{[f](\boldsymbol{x}_i)},\ \overline{f} = f(\tilde{x}) \tag{13}$$

Fig. 1 illustrates IBBA. The box $\boldsymbol{x}_{11}$ is proved not to contain $\mathbf{x}^*$ due to Property (11), as well as boxes $\boldsymbol{x}_{12}$ and $\boldsymbol{x}_{21}$ due to Property (12).
The Set Inversion Via Interval Analysis (SIVIA) algorithm is a branch and bound and allows to approximate the feasible region of $\mathbb{X}$, described by the constraints, by a sub-paving, which is a union of non-overlapping boxes. SIVIA algorithm bisects $\mathbb{X}$ in smaller boxes $\boldsymbol{x}_i$ until the constraint

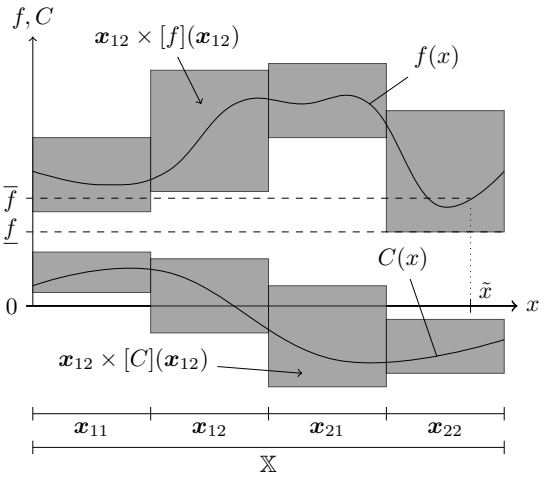


Fig. 1. Illustration of IBBA and SIVIA algorithms.

is proved to be fulfilled over $\boldsymbol{x}_i$ thanks to (14) or not to be fulfilled thanks to (11).

$$\overline{[C](\boldsymbol{x}_i)} \leq 0 \iff \forall \mathbf{x} \in \boldsymbol{x}_i, C(\mathbf{x}) \leq 0 \tag{14}$$

SIVIA algorithm stops when boxes $\boldsymbol{x}_i$ reach a minimum size ϵ. In Figure 1, SIVIA returns the sub-paving made of $\boldsymbol{x}_{11}$, $\boldsymbol{x}_{12}$, $\boldsymbol{x}_{21}$ and $\boldsymbol{x}_{22}$ indicating that $\boldsymbol{x}_{11}$ is not a subset of the feasible set, $\boldsymbol{x}_{22}$ is a subset of the feasible set, and that nothing could be proved for $\boldsymbol{x}_{12}$ and $\boldsymbol{x}_{21}$. That is, $\boldsymbol{x}_{11}$ is an inner approximation the feasible set and $\boldsymbol{x}_{11} \cup \boldsymbol{x}_{12} \cup \boldsymbol{x}_{21}$ is an outer approximation. These approximations can be improved by bisecting $\boldsymbol{x}_{12}$ and $\boldsymbol{x}_{21}$ in smaller boxes.
Finally, IBBA has $[f]$, $[C]$, $\mathbb{X}$ and d as inputs and provides a feasible point $\tilde{\mathbf{x}}$ and a guaranteed enclosure $[\underline{f}, \overline{f}]$ of the global minimum f^*. SIVIA algorithm has $[C]$, $\overline{\mathbb{X}}$ and ϵ as inputs and provides a sub-paving which characterizes the feasible region.

3.2 Analysis problems

Let us consider the analysis problem defined in Section 2. This problem can be rewritten as (15), where IBBA can provide an enclosure $[\underline{u_{eq}(\theta)}, \overline{u_{eq}(\theta)}]$ of the minimum $u_{eq}(\theta)^*$. Being θ the vector of tuning parameters given by the operator, δ a set of variable parameters and Δ a subset of $\mathbb{IR}^{n_\delta}$ with n_δ the dimension of δ.

$$\left\{ \min_{\delta \in \Delta} \min(u_{eq}(\theta, \delta) - u^-, -u_{eq}(\theta, \delta) + u^+) \right. \tag{15}$$

We will show how IBBA can be used to ensure that θ is a feasible solution of this constraint satisfaction problem (CSP).

From Property (10), we can derive Property (16) and Property (17)

$$\begin{aligned} &\underline{u_{eq}(\theta)} > 0 \\ &\implies u_{eq}(\theta)^* > 0 \\ &\iff \forall \delta \in \Delta, \min(u_{eq}(\theta, \delta) - u^-, -u_{eq}(\theta, \delta) + u^+) > 0 \\ &\iff \forall \delta \in \Delta, u_{eq}(\theta, \delta) - u^- > 0 \text{ and }, -u_{eq}(\theta, \delta) + u^+ > 0 \\ &\iff \forall \delta \in \Delta, u^- < u_{eq}(\theta, \delta) \text{ and }, u_{eq}(\theta, \delta) < u^+ \end{aligned} \tag{16}$$

$$\begin{array}{l}\overline{u_{eq}(\theta)} < 0 \\ \implies u_{eq}(\theta)^* < 0 \Rightarrow u_{eq}(\theta, \delta^*) < 0 \\ \iff \min(u_{eq}(\theta, \delta^*) - u^-, -u_{eq}(\theta, \delta^*) + u^+) < 0 \\ \iff u_{eq}(\theta, \delta^*) - u^- < 0 \text{ or }, -u_{eq}(\theta, \delta^*) + u^+ < 0 \\ \iff \exists \delta \in \Delta, u^- > u_{eq}(\theta, \delta) \text{ or }, u_{eq}(\theta, \delta) > u^+ \end{array} \quad (17)$$

According to Proposition (16), if $\underline{u_{eq}(\theta)} > 0$, θ is a feasible solution to Problem (1), which ensures that the system will slide on the sliding surface S over the subset Δ. According to Proposition (17), if $\overline{u_{eq}(\theta)} < 0$, θ is a not feasible solution to Problem 1, which means that the system will not slide over S in all Δ. Actually, the system will leave S at least at δ^* the solution to Problem (15).

If $0 \in [\underline{u_{eq}(\theta)}, \overline{u_{eq}(\theta)}]$, it is not possible to prove whether or not θ is a feasible solution. In the case where θ is not a feasible solution, SIVIA algorithm can be used to characterize the largest subset of Δ where the sliding condition is established. That is, the set:

$$\left\{\delta \in \Delta | u_{eq}(\theta, \delta) < u^+ \text{ and } u^- < u_{eq}(\theta, \delta)\right\}$$

can be approximated by a sub-paving.

3.3 Synthesis problems

Synthesis problems consist either in characterizing the set of feasible tuning parameters with respect to SM conditions and let the operator choose θ in this set, or in minimizing a given cost function over this feasible set. SIVIA algorithm and IBBA are suited to perform such computation.

Let Θ be a subset of $\mathbb{IR}^{n_\theta}$, $f : \mathbb{R}^{n_\theta} \mapsto \mathbb{R}$ be a cost function given by the system designer, and C^*_θ be the minimum of Problem (15) with θ fixed. We suppose that an inclusion function of f is available. Let $C^* : \mathbb{R}^{n_\theta} \mapsto \mathbb{R}$ be the function that maps θ into C^*_θ. The synthesis problem can be expressed in a general way as the optimization problem (18).

$$\begin{cases} \min\limits_{\theta \in \Theta} f(\theta) \\ \text{s.t. } C^*(\theta) \leq 0 \end{cases} \quad (18)$$

The constraint of Problem (18) implies the resolution of an analysis problem over Δ. Using interval analysis, it is possible to provide an enclosure of C^* over a box $\boldsymbol{\theta}$ Monnet et al. (2016). As a consequence IBBA can be used to solve Problem (18) and SIVIA to characterized the set defined by the constraint:

$$\{\theta \in \Theta, C^*(\theta) < 0\} \quad (19)$$

More generally, such a constraint is called a Semi Infinite Constraint (SIC), since it is equivalent to the infinite set of constraint $C(\theta, \delta) \leq 0$, $\forall \delta \in \Delta$, but involves only a finite number of variables. Optimization problems involving SIC are called Semi Infinite Programs (SIP) and can be solved in a global way with different methods Mitsos (2011); Bhattacharjee et al. (2005), and the characterization of the set defined by SICs has been studied in several works Goldsztejn et al. (2009); Ratschan (2002).

4. CASE STUDY

In this section we illustrate the application of the proposed approach to a non-linear system. This is a simplified version of the angular control of a satellite based on the Cayley-Rodriguez parameter.

The system behaviour is modeled in a simplified way by the following equations:

$$\begin{cases} \dot{x}_1 = \frac{1}{2}(1 + x_1^2)x_2 \\ \dot{x}_2 = \frac{1}{J}u \end{cases} \quad (20)$$

where the parameters involved are:

- x_1 Cayley-Rodrigues parameter to define orientation.
- x_2 angular velocity.
- u control action.
- J system inertia.

Assuming it is desired to impose a closed loop dynamics given by:

$$\dot{x}_1 = -\lambda(x_1 - r) \quad (21)$$

with r the position reference and λ an approaching rate tuning parameter. Then, we can propose a sliding mode control with: $u = sign(\sigma)$ and a sliding surface of the form:

$$\Sigma = \left\{\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad : \quad \sigma(\mathbf{x}) = -x_2 - \tfrac{2\lambda(x_1 - r)}{1 + x_1^2} = 0\right\} \quad (22)$$

It is possible to observe that the necessary condition for SM is fulfilled:

$$L_g\sigma = \frac{\partial \sigma}{\partial x_1}\frac{\partial \sigma}{\partial x_2}\begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} = \frac{1}{J} \neq 0 \quad (23)$$

And given σ it is possible to find u_{eq} as:

$$u_{eq} = \frac{2J\lambda^2(x_1 - r)(-x_1^2 + 2rx_1 + 1)}{(1 + x_1^2)^2} \quad (24)$$

Resulting the necessary and sufficient condition for the SM:

$$u^- < u_{eq} < u^+ \quad (25)$$

In the following, we pose this system according to a given operating condition in the form of the problems explained in Section 2, and an analysis of the results is made.

Example 1. *Analysis SM:*

Given $u^+ = 1$, $u^- = -1$, $J = 1$, $\lambda = 0.5$, $r = 1$ *we desire to know if the selected configuration results in a satisfactory sliding behaviour. This means to solve the problem establish by (15). In this case, we could establish the following relations to the problem as:*

$$\begin{cases} \theta \leftrightarrow \textit{no variable: fixed by the operator} \\ \delta \leftrightarrow x_1 \\ \Delta \leftrightarrow [-5, 5] \end{cases} \quad (26)$$

As a result we get the enclosure of the global minimum

$$\min(u_{eq}(\theta,\delta^*)-u^-,-u_{eq}(\theta,\delta^*)+u^+) \in [0.527,0.517]$$

proving satisfactory that it is a good choice for the domain Δ *tested.*

Remark : *From this result, it is possible to conclude that* $u^+ = -u^- = 1-0.527 = 0.473$ *is the smallest value of the control input such that the sliding condition holds over* $x_1 \in [-5,5]$*. Choosing this value of control input over the initial value* $u^+ = -u^- = 1$ *will result in energy consumption savings. In addition, with IBBA one can compute the global minimum of* u_{eq} *over* Δ *and also the global maximum. These two values correspond to the greatest value of* u^- *and the lowest value of* u^+ *which ensure the sliding condition, respectively.*

Example 2. *Synthesis SM:*

Given $u^+ = 1$, $u^- = -1$, $J = 1$, *we desire to know what is the highest possible value of* λ *that result in a satisfactory sliding behaviour for a position reference in* $[-1,1]$*. This can be done by solving Problem (18). In this case, we can establish the variable relation to the problem as:*

$$\begin{cases} \theta & \leftrightarrow \lambda \\ \Theta & \leftrightarrow [0,5] \\ \delta & \leftrightarrow (x_1,r)^T \\ \Delta & \leftrightarrow ([-10,10],[-1,1])^T \end{cases} \tag{27}$$

The objective function is given by $f : \lambda \to -\lambda$ *in order to have a minimization problem. The IBBA algorithm provides* $[-0.70,-0.69]$ *as an enclosure of the minimum. The best feasible point found, with respect to the sliding condition, is* $\lambda = 0.69$*. In addition it is guaranteed that no value of* λ *greater than* 0.7 *exists such as the sliding condition holds over* Δ*.*

Example 3. *Synthesis SM:*

Given $u^+ = 1$, $u^- = -1$, $J = 1$, *we now want to know which are the possible values of* λ *and* r *that result in a satisfactory sliding behaviour. This means to solve the problem established by (19). In this case, we can establish the variable relation to the problem as:*

$$\begin{cases} \theta & \leftrightarrow (\lambda,r)^T \\ \Theta & \leftrightarrow ([0,5],[-5,5])^T \\ \delta & \leftrightarrow x_1 \\ \Delta & \leftrightarrow [-5,5] \end{cases} \tag{28}$$

As a result we get the sub-paving of Fig. 2, where red boxes imply no satisfaction of the conditions imposed, green boxes satisfaction of them, and finally blue boxes indicate that the algorithm cannot determine the conditions. One can remark that the solution of the synthesis problem proposed for Example 2 is consistent with the sub-paving of Fig. 2, since the subset defined by $\{(\lambda,r), \lambda = 0.69$ *and* $r \in [-1,1]\}$ *belongs to the union of the green and blue sets.*

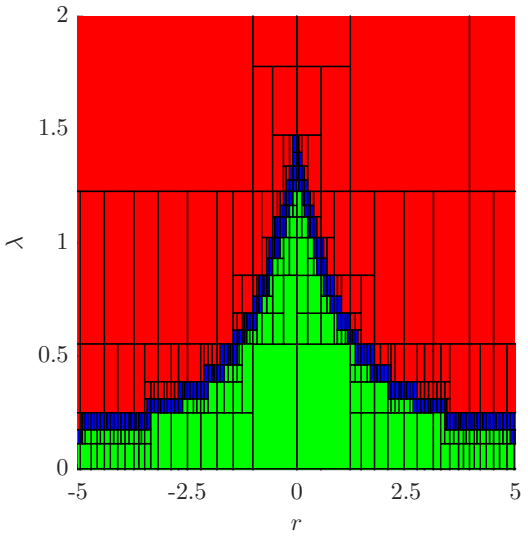


Fig. 2. SM guaranteed existence domain for Example 3

In Fig. 3, it is possible to see the system behaviour with different sliding surfaces (different λ *values) for a reference* $r = 2$*. Notes from Fig. 2 that for* $r = 2$ *over* $\lambda = 0.4$ *the SM condition is no longer satisfied with the given* Δ*, which is also verified by the blue trajectory of Fig.3 which leaves the sliding surface once it is reached.*

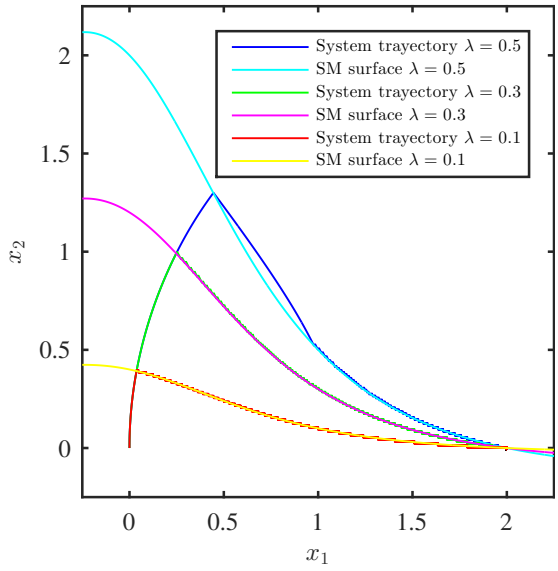


Fig. 3. SM surfaces for different λ values

Remark : *Although the reference signal cannot be really considered as a tuning variable, it is the case in this example. Doing so, we get Figure 2 which indicates for which values of* r *the SM condition holds given a value of* λ*. Variables that would normally belong to* δ *can be changed as tuning variables to provide additional information about the system.*

Example 4. *Synthesis SM:*

Given the same conditions of Example 3, now, we are concerned about the parameter uncertainty over the system, and its effect over the SM conditions. Here it is considered a J parameter variation bounded in the interval $[0.5, 1.5]$. *In this case we can also establish the problem as in (19). In this case, we can establish the variable relation to this problem as:*

$$\begin{cases} \theta & \leftrightarrow (\lambda, r)^T \\ \Theta & \leftrightarrow ([0,5],[-5,5])^T \\ \delta & \leftrightarrow (x_1, J)^T \\ \Delta & \leftrightarrow ([-5,5],[0.5,1.5])^T \end{cases} \tag{29}$$

As a result, we get the sub-paving shown in Fig. 4 where it is possible to see how the area satisfying the SM conditions is smaller than in Fig. 2 due to the uncertainty of J.

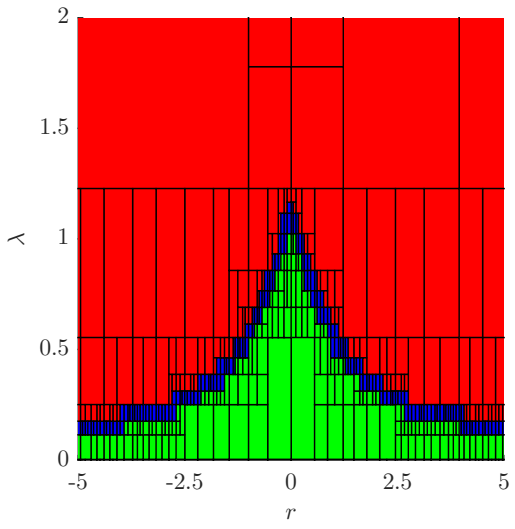


Fig. 4. SM guaranteed existence domain for Example 4

5. CONCLUSIONS

The chosen approach presented in this work shows to be an efficient method to complement the traditional SM control design. Using the interval analysis tools to solve a non-convex global optimization problem, our approach optimizes the SM design for a given criterion. Furthermore, it adds robustness and guarantees the SM set up in front of the process variations and the constrained analyzed state space. To do so, global optimization methods must be used since the synthesis and analysis problems are not convex contrary to the problems emerging in the stochastic approaches which are generally formulated as linear matrix inequalities (therefore convex). The complexity of IBBA grows exponentially with the number of variables, and may fails to solve very large problems.

A particular point to mention is the construction of the sub-paving graphics as design tools. They allow not only to know a particular solution but also to know which is the domain, with respect to the analyzed variables, where the solution is valid.

REFERENCES

Bhattacharjee, B., Lemonidis, P., Green Jr, W.H., and Barton, P.I. (2005). Global solution of semi-infinite programs. *Mathematical Programming*, 103(2), 283–307.

Goldsztejn, A., Michel, C., and Rueher, M. (2009). Efficient handling of universally quantified inequalities. *Constraints*, 14(1), 117–135.

Kearfott, R.B. (1992). An interval branch and bound algorithm for bound constrained optimization problems. *Journal of Global Optimization*, 2(3), 259–280.

Khalil, H.K. (2002). *Nonlinear Systems, 3rd ed.* Prentice Hall.

Li, H., Gao, H., Shi, P., and Zhao, X. (2014). Fault-tolerant control of markovian jump stochastic systems via the augmented sliding mode observer approach. *Automatica*, 50(7), 1825–1834.

Mitsos, A. (2011). Global optimization of semi-infinite programs via restriction of the right-hand side. *Optimization*, 60(10-11), 1291–1308.

Monnet, D., Ninin, J., and Clément, B. (2016). A global optimization approach to structured regulation design under h_∞ constraints. *55th IEEE Conference on Decision and Control (CDC), Las Vegas.*

Moore, R.E., Kearfott, R.B., and Cloud, M.J. (2009). *Introduction to Interval Analysis.* Society for Industrial and Applied Mathematics.

Niu, Y., Ho, D.W., and Lam, J. (2005). Robust integral sliding mode control for uncertain stochastic systems with time-varying delay. *Automatica*, 41(5), 873–880.

Ratschan, S. (2002). Approximate quantified constraint solving by cylindrical box decomposition. *Reliable Computing*, 8(1), 21–42.

Rauh, A. and Aschemann, H. (2012). Interval-based sliding mode control and state estimation for uncertain systems. *17th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdrojie, Poland.*

Rosendo, J.L., Clément, B., and Garelli, F. (2016). Sliding mode reference conditioning for path following applied to an auv. *10th IFAC Conference on Control Applications in Marine Systems (CAMS)*, 49(23), 8–13.

Senkel, L., Rauh, A., and Aschemann, H. (2014). Robust sliding mode techniques for control and state estimation of dynamic systems with bound and stochastic uncertainty. *Second International Conference on Vulnerability and Risk Analysis and Management (ICVRAM) and the Sixth International Symposium on Uncertainty, Modeling, and Analysis (ISUMA), Liverpool, UK.*

Sira-Ramrez, H. (1993). On the dynamical sliding mode control of nonlinear systems. *International Journal of Control*, 57(5), 1039–1061.

Utkin, V., Guldner, J., and Shi, J. (2009). *Sliding mode control in electro-mechanical systems*, volume 34. CRC press.

Wu, L., Su, X., and Shi, P. (2012). Sliding mode control with bounded $\mathcal{L}_2$gain performance of markovian jump singular time-delay systems. *Automatica*, 48(8), 1929–1933.

Titre : Optimisation Globale Minmax pour la commande robuste H_∞

Mots clés : Optimisation globale, Optimisation minmax, Commande H_∞, Systèmes incertains

Résumé : La commande H_∞ est de nos jours utilisée pour la régulation de nombreux systèmes. Cette technique de contrôle permet de synthétiser des lois de commande robustes, dans le sens où le comportement du système régulé est peu sensible aux perturbations externes. De plus, la commande H_∞ permet de prendre en compte des incertitudes liés au modèle décrivant le système à réguler. Par conséquence, cette technique de contrôle est robuste vis-à-vis des perturbations et des incertitudes de modèle.

Afin de synthétiser une loi de commande robuste, les spécifications des performances du système en boucle fermée sont traduites en critères H_∞ à partir desquels est formulé un problème d'optimisation. La loi de commande est une solution de ce problème, qui est non convexe dans le cas général.

Les deux principales approches pour la résolution de ce problème sont basées sur la reformulation convexe et les méthodes d'optimisations locales, mais ne garantissent pas l'optimalité de la loi de commande vis-à-vis des critères H_∞ .

Cette thèse propose une approche de la commande H_∞ par des méthodes d'optimisation globales, rarement considérées jusqu'à présent. Contrairement aux approches classiques, bien qu'au prix d'une complexité algorithmique supérieure, la convergence vers la loi de commande optimale est garantie par les méthodes globales. De plus, les incertitude de modèle sont prises en compte de manière garantie, ce qui n'est pas nécessairement le cas avec les approches convexes et locales.

Title : Global Minmax Optimization for robust H_∞ control

Keywords : Global optimization, Minmax optimization, H_∞ control, Uncertain systems

Abstract : H_∞ control is nowadays used in many applications. This control technique enables to synthesize control laws which are robust with respect to external disturbances. Moreover, it allows to take model uncertainty into account in the synthesis process. As a consequence, H_∞ control laws are robust with respect to both external disturbances and model uncertainty.

A robust control law is a solution to an optimization problem, formulated from H_∞ criteria. These criteria are the mathematical translations of the desired closed loop performance specifications. The two classical approaches to the optimization problem rely on the convex reformulation and local optimization methods. However, such approaches are unable to guarantee the optimality, with respect to the H_∞ criteria, of the control law.

This thesis proposes to investigate a global optimization approach to H_∞ control. Contrary to convex and local approaches, global optimization methods enable to guarantee the optimality of the control, and also to take into account model uncertainty in a reliable way.