UNIVERSITÉ D'ANGERS

ISTIA

N° Ordre : 764

ÉCOLE DOCTORALE D'ANGERS

2006

Thèse de DOCTORAT

Spécialité : AUTOMATIQUE ET INFORMATIQUE APPLIQUÉE

Présentée et soutenue publiquement par

Massa DAO

le 7 Septembre 2006

à Cholet - Université d'Angers

Caractérisation d'ensembles par des méthodes intervalles. Applications en automatique

Jury

Président	:	Jean-Jacques Loiseau, Directeur de Recherche	CNRS, IRCCyN, Nantes
Rapporteurs	:	Nacim Ramdani, Maître de Conférences (HDR)	CERTES, Université de Paris XII, Paris
		Philippe Bonnifait, Maître de Conférences (HDR)	UTC de Compiègne
Examinateurs	:	Luc Jaulin, Professeur	ENSIETA, Brest
		Laurent Hardouin, Professeur	Université d'Angers
		Mehdi Lhommeau, Maître de Conférences	Université d'Angers

Directeur de thèse : Luc Jaulin

LABORATOIRE D'INGÉNIERIE DES SYSTÈMES AUTOMATISÉS.
62, avenue Notre Dame du Lac, F-49000 ANGERS

ED 363

CARACTÉRISATION D'ENSEMBLES PAR DES MÉTHODES INTERVALLES. APPLICATIONS EN AUTOMATIQUE

Massa DAO

Université d'Angers

Massa DAO

Caractérisation d'ensembles par des méthodes intervalles. Applications en automatique

xviii+

Ce document a été préparé avec LaTeX2$_\varepsilon$ et la classe `these-IRIN` version 0.92 de l'association de jeunes chercheurs en informatique LOGIN, Université de Nantes. La classe `these-IRIN` est disponible à l'adresse :

`http://www.sciences.univ-nantes.fr/info/Login/`

Résumé

L'étude et la conception des systèmes non linéaires (étude de la stabilité, synthèse de lois de commandes stabilisantes, analyse en robustesse, ...) posent des problèmes numériques difficiles, que les méthodes classiques ont du mal à résoudre. Les formulations ensemblistes de ces problèmes où l'idée de base est de remplacer une valeur ponctuelle par un intervalle qui la contient se sont déjà montrées très efficaces pour leur résolution.

Dans cette thèse, nous proposons de nouveaux algorithmes ensemblistes dédiés à des tâches plus spécifiques telles que la projection d'un ensemble sur un sous-espace. Afin d'améliorer l'efficacité de ces algorithmes, dans un cadre général, nous avons présenté deux idées. La première de ces idées consiste à former à partir des contraintes, des graphes connexes de variables quantifiées. L'objectif est de décomposer la projection d'un ensemble à n dimensions, en une intersection de projections d'ensembles dont la somme des dimensions vaut n. Notre deuxième idée consiste à rendre les algorithmes de projection plus efficace en améliorant l'approximation intérieure.

Dans un deuxième temps, les algorithmes développés ont été mis en oeuvre afin de traiter des problèmes d'automatique liés aux systèmes à retards, à la commande d'un bateau à voile et à la conception de filtres numériques et analogiques.

Mots-clés : analyse par intervalles, problème de satisfaction de contraintes (CSP), projection d'ensembles, inversion ensembliste, estimation à erreurs bornées, systèmes à retards, commande d'un bateau à voile, conception de filtres.

Abstract

The study and the design of nonlinear systems (stability study, stabilizing control laws synthesis, robustness analysis, ...) arise difficult numerical problems that classical methods have difficulty to solve. Then, set-membership formulations of these problems, where the basic idea is to replace punctual value by interval that include it, where already show very efficient for their resolution.

In this thesis, we propose new set-membership algorithms dedicated for most special tasks like set projection over a subspace. In order to improve, the efficiency of these algorithms in general context, we have presented two ideas. The first one consist to decompose the projection of a n-dimensional set, in an intersection of sets projections where the sum of dimensions is equal to n. The second idea consists in yielding the projection algorithms more efficient by improving inner approximation.

In a second time, the developed algorithms have been implemented in order to treat stability and control problem of time-delay systems, sailing boat control and filters design.

Keywords : interval analysis, constraints satisfaction problem (CSP), set projection, bounded-error estimation, set inversion, time-delays system, sailing boat control, filters design.

Remerciements

Ce travail de thèse a été réalisé grâce à un financement de la **Communauté d'Agglomération du Choletais**. A ce titre, j'adresse mes remerciements aux membres de la Communauté d'Agglomération du Choletais. La préparation ainsi que la présentation de cette thèse constituent un témoignage de leurs intérêts et contributions à la recherche publique.

Je remercie **Jean-Louis Ferrier** et **Jean-Louis Boimond**, Responsables du Laboratoire des Systèmes Automatisés à Angers, pour m'avoir accueilli et facilité mes recherches au sein de ce Laboratoire.

Je tiens à exprimer ma profonde gratitude à mon directeur de thèse, **Luc Jaulin**, Professeur à l'ENSIETA à Brest. Durant les trois années passées en votre compagnie, je ne dirais jamais assez, combien j'ai appris. Merci, pour votre disponibilité et vos précieux conseils. Ce travail n'aurait pas été possible sans votre aide.

J'exprime toute ma reconnaissance à **Nacim Ramdani**, Maître de Conférence à l'Université de Paris XII et à **Philippe Bonnifait**, Maître de Conférence à l'Université Technologique de Compiègne, pour l'honneur qu'ils m'ont fait en acceptant de rapporter cette thèse. Vos conseils et suggestions m'ont permis d'apporter les corrections nécessaires à cette thèse.

Je voudrais également remercier **Jean-Jacques Loiseau**, Directeur de recherche CNRS à l'IRCCyN et **Laurent Hardouin**, Professeur à l'Université d'Angers pour l'intérêt qu'ils ont porté à ces travaux en acceptant de les examiner.

Je tiens à faire part, de ma reconnaissance à **Medhi Lhommeau**, Maître de Conférences à l'Université d'Angers, pour ses conseils et sa grande sollicitude. Il a été mon encadrant durant la dernière année de ma thèse.

Je remercie **Michaël Di-Loreto**, les rapports de travail que nous avons entretenus ont été enrichissants et constructifs. Notre collaboration m'a initié aux systèmes à retards.

Je remercie mes collègues, les doctorants du LISA, pour les précieux échanges et l'esprit de camaraderie qu'ils ont su faire régner au sein du laboratoire. J'en profite pour adresser ma gratitude à **Xavier Baguenard** pour sa participation active aux développements informatiques de mes recherches.

Je ne pourrais terminer ces remerciements, sans avoir une pensée pour mes frères et sœurs **Zangou**, **Katangnia**, **Bogoba**, et mon amie **Bernadette Guillot**. Vous m'avez soutenu moralement dans les moments difficiles. Ce travail est aussi le vôtre.

Enfin, je remercie mon père, pour lequel j'ai un profond respect. Je n'oublie pas que c'est la rigueur de ton éducation qui m'a permis de faire des études scientifiques.

À la mémoire de ma mère,

Sommaire

Table des matières

CHAPITRE 1

Introduction générale

1.1 Contexte de l'étude

Depuis déjà quelques décennies (voir [Moore, 1966, Moore, 1979]), les méthodes par intervalles sont utilisées pour résoudre des problèmes non linéaires. De manière générale, ces approches sont inefficaces, en terme de temps de calcul, face à des problèmes de grandes dimensions (des problèmes d'estimation où le nombre de paramètres à déterminer est supérieur à 4). Dans cette thèse, nous avons donc adopté le positionnement suivant. Premièrement, au lieu de caractériser complètement l'ensemble de solution, nous avons choisi de réaliser sa projection sur deux dimensions pour ne caractériser qu'une partie de cet ensemble. Ainsi, les problèmes d'estimations de paramètres ont été formulés en termes de projection d'ensembles. Deuxièmement, nous avons étendu les champs d'applications de cette méthode à différents problèmes d'automatique.

Avant de détailler l'ensemble les travaux de cette thèse, il nous a semblé opportun de rappeler quelques généralités sur l'automatique et les systèmes. L'automatique est une discipline scientifique qui vise à l'étude des *systèmes* qui décrivent le comportement de phénomènes ou processus physique, chimique, biologique, *etc.*

Un système est un ensemble de relations causales établies entre les grandeurs d'entrées, d'états et de sorties. De ce fait, les sorties d'un système (effets) dépendent, en général, de son état et des entrées (causes). Les systèmes se répartissent en trois catégories : les systèmes à temps continu, les systèmes à événements discrets et les systèmes hybrides, cette dernière catégorie est considérée comme une combinaison des deux premières. La dynamique des systèmes à événements discrets est régie par des équations récurrentes, tandis que celle des systèmes continus obéit à des équations différentielles ordinaires.

Les études menées sur les systèmes concernent essentiellement trois thèmes principaux :

- **Modélisation** : Cette notion consiste à doter le système d'une structure qui possède des propriétés algébriques intéressantes, dans des espaces abstraits. Cette structure a pour but de décrire les comportements du système. Par exemple les systèmes à temps continu utilisent des représentations de type temporelles ou fréquentielles. Ces systèmes modélisent souvent des phénomènes naturels,

de ce fait, la représentation temporelle requiert parfois l'utilisation des principes de la physique classique (mécanique newtonienne, électromagnétisme, thermodynamique, *etc.*).

- **Commande** : La problématique liée à la commande peut être résumée ainsi, comment déterminer les entrées d'un système afin d'obtenir des sorties désirées ? En d'autres termes, cela consiste à imposer, à partir des entrées d'un système, un comportement particulier. Plusieurs ouvrages d'automatique traitent de la commande des systèmes linéaires à travers des problèmes de régulation, d'asservissement, de rejet de perturbation, de commande robuste, *etc.* (voir entre autre [de Larminat, 1993], [Landau, 1993] et [Barre et al., 1995]).
- **Identification** : Cette étude consiste à trouver une structure pour un système, à partir de connaissances sur des données entrée-sortie. Ceci nécessite d'abord d'établir un modèle pour le système puis d'ajuster "au mieux" les paramètres dynamiques de ce modèle afin que le comportement du modèle soit le plus fidèle possible aux mesures entrée-sortie dont nous disposons ([Ljung, 1987], [Landau, 1993], [Rivoire and Ferrier, 1997]).

Dans le cas d'une famille de systèmes ou de systèmes paramétriques, les trois thèmes d'études soulevés ci-dessus posent des problèmes d'estimation difficiles à résoudre. Un de ces problèmes concerne, la satisfaction d'une multitude de contraintes due à des impératifs de conception. Ces contraintes se traduisent souvent sous forme d'inégalités non linéaires. Ainsi, le problème qui nous est posé revient à rechercher les solutions à des inégalités non linéaires, dans des compacts de $\mathbb{R}^n$. Ce problème constitue un thème important des mathématiques et en particulier du calcul numérique. Une méthode désormais classique est la méthode du *point fixe* (méthode de Newton-Raphson) pour la recherche de racines d'équations non linéaires. Une autre méthode met à profit les méthodes d'optimisation afin de minimiser la norme euclidienne d'une fonction vectorielle.

L'inconvénient des deux méthodes que nous venons de citer relève de conditions de convergence relativement restrictives (choix difficile d'une condition initiale pour la méthode du point fixe, convergence des méthodes d'optimisation vers des minimums locaux). Dans le meilleur des cas, ces méthodes permettent de déterminer un nombre fini de racines dans un compact, lorsque celles-ci existent. Par exemple, ces méthodes ne peuvent prouver l'absence de racines dans un compact.

Face à ces inconvénients, nous avons recours à des méthodes dites de *caractérisation d'ensembles*. Ces méthodes s'intéressent à la représentation, au sens de la frontière et du volume, d'un sous-ensemble de $\mathbb{R}^n$. La technique consiste à utiliser le calcul par intervalles afin de réaliser une approximation des ensembles par des structures de compact dans $\mathbb{R}^n$ de type *pavé*. Un pavé est un compact de $\mathbb{R}^n$ défini par le produit cartésien d'intervalles. Le développement d'outils algébriques pour ces structures ensemblistes permet d'effectuer des calculs, en temps fini, avec des compacts constitués d'une infinité de vecteurs

(voir [Moore, 1966], [Moore, 1968], [Hansen, 1975], [Hansen, 1965], [Krawczyk and Neumaier, 1985], [Kurzhanski and Valyi, 1997], [Durieu and Walter, 2001], *etc.*). Comparé aux approches intervalles, les méthodes stochastiques ou aléatoires utilisées par les heuristiques sondent l'espace dans un voisinage point par point. Par exemple, pour trouver les points situés à l'intérieur d'un disque de rayon un, il faudrait à un calculateur, en utilisant une approche type *Monté Carlo*, un temps infini.

Au cours de nos travaux, nous nous sommes intéressés à la caractérisation de quatre types d'ensembles dont les expressions sont données par,

1. $\mathbb{S}_1 = \{\mathbf{x} \in \mathbb{X} | \, \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\}$;
2. $\mathbb{S}_2 = \{\mathbf{f}(\mathbf{x}) | \, \mathbf{x} \in \mathbb{X}\}$;
3. $\mathbb{S}_3 = \{\mathbf{p} \in \mathbb{P} | \, \exists \mathbf{q} \in \mathbb{Q}, \mathbf{f}(\mathbf{p}, \mathbf{q}) \in \mathbb{Y}\}$;
4. $\mathbb{S}_4 = \{\mathbf{p} \in \mathbb{P} | \, \forall \mathbf{q} \in \mathbb{Q}, \mathbf{f}(\mathbf{p}, \mathbf{q}) \in \mathbb{Y}\}$.

avec $\mathbb{X}$, $\mathbb{Y}$, $\mathbb{P}$, $\mathbb{Q}$ des compacts $\mathbb{R}^n$, $\mathbf{f}$ correspond à une fonction vectorielle quelconque. En ce sens nos travaux s'inscrivent dans la liste des thèses déjà menées sur le sujet : [Jaulin, 1994], [Kieffer, 1999], [Didrit, 1997], [Braems, 2002], *etc.*
Les applications que nous avons traitées concernent surtout des systèmes à temps continu, une adaptation des méthodes par intervalles est concevable pour des systèmes à événements discrets ou même hybrides, à condition de trouver une formulation adéquate pour les problèmes associés à ces systèmes.

1.2 Organisation de la thèse

Le contenu de ce mémoire est divisé en deux parties. La première partie (chapitres 2-5) est destinée à l'étude des notions et principes qui rentrent dans l'élaboration des méthodes par intervalles. La deuxième partie (chapitres 6-9) concerne les applications de ces méthodes à des problèmes liés aux systèmes à retards, à la commande de systèmes non linéaires (recherche d'un régulateur optimal pour un bateau à voile) et à la conception de filtres.

1.2.1 Partie 1 (Principes et méthodes)

– **Le chapitre 2** porte sur des rappels sur l'analyse par intervalles. Nous présenterons, dans ce chapitre, les concepts qui sont à la base du calcul par intervalles. Les structures fondamentales de compact sur $\mathbb{R}^n$ seront représentées par des intervalles, des pavés et des *sous-pavages*. Nous verrons, à travers l'arithmétique par intervalles et les *fonctions d'inclusion*, les outils algébriques permettant d'étendre, les opérations mathématiques sur les réels, aux calculs sur des intervalles de valeurs réelles. Ainsi, dans le cas général, la fonction d'inclusion permet d'approximer l'image d'un pavé par une fonction vectorielle.

– **Le chapitre 3** est consacré à la recherche du plus petit pavé renfermant les vecteurs satisfaisants une conjonction de contraintes de type inégalités non linéaires. La résolution de ces inégalités sera abordée dans le cadre plus global du *problème de satisfaction de contraintes* connu sous le nom de CSP (***C**onstraints **S**atisfaction **P**roblem*). Dans ce contexte, nous allons présenter les notions de *consistance locale ou globale* mais aussi de *contracteurs*. Il sera proposé deux catégories de méthodes de consistance. La première catégorie, appelée méthode de consistance "faible", utilise la décomposition en contraintes dites primitives puis l'algorithme de Waltz étendu aux intervalles afin de trouver un pavé qui possède des propriétés adéquates pour la consistance locale (voir [Waltz, 1975], [Cleary, 1987], [Davis, 1987]). Pour la deuxième catégorie de méthodes (les méthodes de consistance "forte"), nous proposerons différentes stratégies qui visent à améliorer les méthodes de consistance faible, ceci pour sortir des situations de blocage que constitue la consistance locale et d'atteindre, à terme, la consistance globale. Par la suite, ces différentes méthodes de consistance seront appliquées dans l'objectif de réaliser des contracteurs pour des ensembles définis par des inégalités non linéaires.
– **Le chapitre 4** s'intéresse à la représentation d'un sous-ensemble $\mathbb{S} \subset \mathbb{R}^n$ par deux sous-pavages (unions de pavés) $\underline{\mathbb{S}}$ et $\overline{\mathbb{S}}$ tels que $\underline{\mathbb{S}} \subset \mathbb{S} \subset \overline{\mathbb{S}}$. Dans ce chapitre, nous ferons un bref tour d'horizon des principales méthodes de caractérisation d'ensembles à travers des problèmes d'inversion ensembliste, l'image d'un compact par une fonction vectorielle ou encore la connexité d'un ensemble. Les algorithmes employés seront construits à partir des contracteurs du chapitre précédent. Nous indiquerons sous quelles conditions ces méthodes sont efficaces et quels en sont les inconvénients.
– Une classe de CSPs constitués d'inégalités non linéaires, et dont certaines variables sont associées à des quantificateurs existentiels ou universels ($\exists$,$\forall$), est formulée en terme de projection d'ensembles. **Le chapitre 5** est dédié à ce problème. Dans ce chapitre, une partie technique est consacrée à la conception et au développement d'un logiciel nommé PROJ2D. Ce logiciel réalise la projection en deux dimensions d'ensembles définis par des inégalités non linéaires. Au cours de ces travaux, nous avons proposé la représentation des CSPs sous forme d'arbre. Les techniques d'analyses syntaxiques ont été ensuite utilisées afin d'effectuer le passage d'un fichier texte, traduisant un CSP, à des structures de type arbres aisément manipulable dans un algorithme. Nous avons ensuite combiné ce dispositif aux algorithmes de projection d'ensembles. Afin de généraliser ces algorithmes et d'en améliorer l'efficacité, nous avons présenté deux idées. La première de ces idées a consisté à former à partir des contraintes, des sous-graphes connexes de variables quantifiées. L'objectif de ceci est de décomposer dans le cadre général, la projection d'un ensemble à n dimensions en une intersection de projections d'ensembles dont la somme des dimensions vaut n. Notre deuxième idée consiste à rendre les algorithmes de projection plus efficace en améliorant

l'approximation intérieure. Pour cela, nous avons présenté, sous le nom de `IAVIC` (*Inner Approximation Via Interval Computation*), une approche qui utilise une méthode de recherche locale pour déterminer dans un pavé une partie ou un domaine acceptable.

1.2.2 Partie 2 (Applications)

- **Le chapitre 6** aborde des exemples, de problèmes d'estimation de paramètres dans un contexte à erreurs bornées, d'identifiabilité, de conception de robot et de commande robuste, en terme de projection d'ensembles. Au cours de ce chapitre, des mesures ont été prélevées afin d'analyser, les améliorations apportées par l'utilisation de la méthode `IAVIC` proposée dans le chapitre précédent.
- **Le chapitre 7** a été mené en collaboration avec des membres de l'IRCCyN, les travaux concernent l'utilisation des méthodes intervalles comme outil numérique pour l'étude des systèmes linéaires à retards. Pour des systèmes à retards commensurables de type retardé et neutre, nous avons présenté des problèmes de stabilité, d'analyse en stabilité robuste, ainsi que de stabilisation par l'estimation de paramètres stabilisants pour une loi de commande.
- **Le chapitre 8** traite de la réalisation d'une loi de commande optimale pour un bateau à voile. Pour cela, nous allons devoir résoudre deux problèmes. Le premier problème consiste à déterminer, dans un contexte non linéaire, un régulateur pour le cap et l'ouverture de voile du bateau. Le deuxième problème concerne l'ajustement de l'ouverture de voile, dans le but d'atteindre, pour un cap fixé et un vent constant avec une direction donnée, une vitesse maximale pour le bateau. Notre système (bateau à voile + système de commande) sera régulé pour aller le plus vite possible sur l'eau, ceci en fonction du cap du bateau et de la direction du vent. Des tests de simulation ont permis de valider les performances de notre système de commande optimale. Ces tests ont consisté à la stabilisation approximative du bateau autour d'une balise de référence statique ou mobile.
- **Le chapitre 9** concerne la conception de filtres dans un contexte à erreurs bornées. Ce chapitre montre, à travers l'estimation des coefficients de filtres (analogique ou numérique), que le calcul par intervalles combiné aux techniques de recherche d'une solution ponctuelle (sélection par critères) est un outil permettant la résolution de problèmes d'estimation, de grandes dimensions et "mal conditionnés". Durant le traitement des exemples, nous tenterons de répondre aux questions suivantes :
  - Est-il possible de trouver un filtre d'ordre n satisfaisant des contraintes sur le gain ou la phase ?
  - Comment garantir l'ordre minimal n^* pour un filtre compatible avec des contraintes non linéaires ? En d'autres termes, déterminer un ordre n^* au dessous duquel il n'existe aucun filtre compatible avec nos contraintes.

– En conclusion nous ferons un bilan sur le travail réalisé, nous discuterons ensuite les pistes envisageables pour de nouvelles perspectives de recherche.

PARTIE 1 : MÉTHODES ET PRINCIPES

CHAPITRE 2

Rappels sur l'analyse par intervalles

Les calculateurs électroniques représentent les réels suivant un type de codage à précision limitée appelé *flottants*. Pour des raisons que nous pouvons aisément comprendre, les machines codent un réel par un arrondi avec un nombre fini de chiffres après la virgule. Les opérations mathématiques effectuées sur les flottants à la place de réels provoquent un cumul d'erreurs dûs aux arrondis. A l'issu de certains calculs sur les flottants, le résultat obtenu peut être éloigné du résultat attendu. La solution adoptée grâce aux travaux de Ramon Moore dans [Moore, 1966] se décrit comme suit. D'une part, les réels sont représentés par des intervalles bornés par des flottants avec un nombre de décimales fixes, par exemple $\pi = 3.14159...$ sera représenté par $[3.1415, 3.1416]$ car $\pi \in [3.1415, 3.1416]$. D'autre part, des outils de calcul avec les intervalles sont définis afin de représenter le résultat d'une opération ou d'une fonction mathématique par un intervalle qui le contient de façon garantie. De ce fait, nous arrivons à quantifier les erreurs introduites par l'utilisation des flottants. Bien entendu, la représentation par des intervalles n'a pas que des avantages. Un des inconvénients sérieux est la surévaluation des bornes d'incertitude du résultat obtenu à l'issu d'un calcul avec des intervalles.

Ainsi, le calcul par intervalles rassemble toutes les notions qui permettent d'effectuer des opérations mathématiques non pas avec des valeurs réelles, mais avec des intervalles de valeurs réelles. Le développement de cet outil mathématique ainsi que les lois engendrées permettent la manipulation de structures ensemblistes fondamentales. Une conséquence intéressante de ceci est l'analyse, en un temps fini, d'un compact[1] constitué d'une infinité de points. Par exemple, cette analyse peut consister à montrer qu'un système d'inégalités n'admet aucune solution dans un compact de $\mathbb{R}^n$. Le calcul par intervalles est à la base d'une multitude d'algorithmes développés pour la résolution d'inégalités non linéaires, pour l'optimisation globale de fonctions coûts dans un domaine continu, *etc.* Les chapitres qui suivent traiteront de ces problèmes ainsi que des méthodes associées.

Nous rappelons dans ce chapitre quelques notions sur le calcul par intervalles, vu que le sujet n'est pas nouveau dans le domaine du calcul ensembliste, une abondante littérature est disponible dans [Moore, 1966], [Moore, 1979], [Krawczyk and Neumaier, 1985], [Neumaier, 1990], [Hansen, 1992b] et

[1] sous-ensemble connexe, fermé et borné

[Jaulin et al., 2001]. La section 2.1 présente, d'une manière générale, la notion de sous-ensembles de $\mathbb{R}^n$ et le sens des opérations ensemblistes ($\cap, \cup, \times, ...$). La section 2.2 fait état de définitions et notations pour les intervalles et les pavés ainsi que les opérations arithmétiques ($+, -, /, *, ...$) sur les intervalles. La section 2.3 rappelle ce qu'est une fonction d'inclusion. A cet effet, plusieurs méthodes d'évaluation de fonctions d'inclusion seront présentées et comparées. Comme nous l'avons souligné précedemment, les différentes méthodes d'évaluations induisent un pessimisme (ou une surévaluation) sur les bornes de l'intervalle obtenu à l'issu du calcul. Ce pessimisme est dû dans certains cas à une dépendance multiple d'une même variable dans l'expression d'une fonction, nous appellerons cela, le *problème de multi-occurrences*. Ainsi plus la largeur de l'intervalle résultat sera petite, plus la méthode d'évaluation sera jugée efficace. Enfin, la section 2.4 donne une analyse globale des principes que nous aurons présenté ainsi que les propriétés intéressantes à en tirer pour la suite.

2.1 Opérations ensemblistes pures

Définition 2.1 (Inclusion ensembliste). Soit $\mathbb{A}$ et $\mathbb{B}$ deux sous-ensembles de $\mathbb{R}^n$. Rappelons que $\mathbb{A}$ est inclus dans $\mathbb{B}$ si et seulement si tout élément de $\mathbb{A}$ appartient à $\mathbb{B}$. Il vient de ce fait

$$\mathbb{A} \subset \mathbb{B} \Leftrightarrow (\forall\, \mathbf{x} \in \mathbb{A},\ \mathbf{x} \in \mathbb{B})\,. \tag{2.1}$$

Les opérations suivantes s'appliquent sur des sous-ensembles de $\mathbb{R}^n$ en général. Elles regroupent l'union, l'intersection, le produit scalaire et la projection. Étant donnés $\mathbb{X}$ et $\mathbb{Y}$ deux sous-ensembles de $\mathbb{R}^n$ et de $\mathbb{R}^m$. Nous avons

$$\text{Proj}_i\,(\mathbb{X}) \triangleq \left\{ x_i \in \mathbb{R} |\ \exists \mathbf{x} = (x_1, ..., x_i, ..., x_n)^{\mathrm{T}} \in \mathbb{X} \right\}, \tag{2.2}$$

$$\mathbb{X} \cap \mathbb{Y} \triangleq \{ \mathbf{x} \in \mathbb{R}^n |\ \mathbf{x} \in \mathbb{X} \text{ et } \mathbf{x} \in \mathbb{Y} \}\,, \text{ avec } n = m, \tag{2.3}$$

$$\mathbb{X} \cup \mathbb{Y} \triangleq \{ \mathbf{x} \in \mathbb{R}^n |\ \mathbf{x} \in \mathbb{X} \text{ ou } \mathbf{x} \in \mathbb{Y} \}\,, \text{ avec } n = m, \tag{2.4}$$

$$\mathbb{X} \times \mathbb{Y} \triangleq \left\{ (\mathbf{x}, \mathbf{y})^{\mathrm{T}} \in \mathbb{R}^{n+m} \middle|\ \mathbf{x} \in \mathbb{X} \text{ ou } \mathbf{y} \in \mathbb{Y} \right\}. \tag{2.5}$$

D'une manière générale, il va de soit que des principes valables pour les sous-ensembles, le seront en particulier pour les pavés ainsi que les intervalles.

2.2 Intervalles, pavés et sous-pavages

Comme cela a été souligné dans l'introduction, cette section a pour objet un rappel des notions sur les intervalles et les pavés. Ces notions constituent des éléments fondamentaux de l'analyse par intervalles. La plupart des définitions et opérations ensemblistes présentées sont tirées de [Moore, 1966] et [Hansen, 1992b].

2.2.1 Définitions et notations

Définition 2.2 (Intervalle). Un intervalle $[x]$ est un sous-ensemble connexe, borné et fermé de réels. Nous avons

$$[x] = [\underline{x}, \overline{x}] \triangleq \{x \in \mathbb{R},\ \underline{x} \leq x \leq \overline{x}\}\,, \tag{2.6}$$

où $\underline{x}$ et $\overline{x}$ sont respectivement les bornes inférieures et supérieures de $[x]$. L'ensemble des intervalles est noté $\mathbb{IR}$. La largeur d'un intervalle est donnée par $w\left([x]\right) \triangleq \overline{x} - \underline{x}$.

Définition 2.3 (Pavé). De la même façon que nous représentons un réel par un intervalle, il est possible de concevoir une structure de compact simple afin de représenter un vecteur à composantes réelles. Un pavé (ou vecteur intervalle) $[\mathbf{x}]$ est un compact de $\mathbb{R}^n$ défini par le produit cartésien de n intervalles. On note

$$\begin{array}{rcl} [\mathbf{x}] & = & [\underline{x}_1, \overline{x}_1] \times [\underline{x}_2, \overline{x}_2] \times ... \times [\underline{x}_n, \overline{x}_n] \\ & = & [x_1] \times [x_2] \times ... \times [x_n]\,, \end{array} \tag{2.7}$$

ou encore

$$[\mathbf{x}] = \begin{pmatrix} [\underline{x}_1, \overline{x}_1] \\ \vdots \\ [\underline{x}_i, \overline{x}_i\ ] \\ \vdots \\ [\underline{x}_n, \overline{x}_n] \end{pmatrix}. \tag{2.8}$$

Pour tout $i \in \{1, 2, ..., n\}$, l'intervalle $[x_i] = [\underline{x}_i, \overline{x}_i]$ correspond à la *ième* composante de $[\mathbf{x}]$. Les caractéristiques du pavé sont décrites comme suit,

- La longueur du pavé $[\mathbf{x}]$ est donnée par

$$w\left([\mathbf{x}]\right) \triangleq \max_{i=1}^{n} \left(w\left([x_i]\right)\right). \tag{2.9}$$

- Le centre de $[\mathbf{x}]$ est défini par

$$m\left([\mathbf{x}]\right) \triangleq \left(\frac{\overline{x}_1 + \underline{x}_1}{2}, ..., \frac{\overline{x}_i + \underline{x}_i}{2}, ..., \frac{\overline{x}_n + \underline{x}_n}{2}\right)^{\mathrm{T}}. \tag{2.10}$$

– Le volume de $[\mathbf{x}]$ est donné par

$$\begin{aligned} \text{Vol}\left([\mathbf{x}]\right) &\triangleq \prod_{i=1}^{n} w\left([x_i]\right) \\ &= (\overline{x}_1 - \underline{x}_1) \cdot (\overline{x}_2 - \underline{x}_2) \cdot ... \cdot (\overline{x}_n - \underline{x}_n)\,. \end{aligned} \tag{2.11}$$

Enfin, l'ensemble des pavés de $\mathbb{R}^n$ est noté : $\mathbb{IR}^n$.

Remarque 2.4. *Par généralisation, un réel x est un intervalle dégénéré c'est-à-dire un intervalle dont les deux bornes sont égales, $\overline{x} = \underline{x} = x$. De la même manière, un vecteur à n composantes réelles est aussi un pavé,*

$$\mathbf{x} = \left([x_1, x_1]\,, ..., [x_i, x_i]\,, ..., [x_n, x_n]\right)^T. \tag{2.12}$$

Remarque 2.5. *Considérons en particulier, trois pavés $[\mathbf{x}]$, $[\mathbf{y}]$, $[\mathbf{z}]$ de $\mathbb{R}^n$ et une opération ensembliste $\Diamond \in \{\cap, \cup, \times\}$ tels que :*

$$[\mathbf{z}] = [\mathbf{x}] \Diamond [\mathbf{y}]\,, \tag{2.13}$$

alors nous avons

$$[z_i] = [x_i] \Diamond [y_i]\,, \quad i = \{1, 2, ..., n\}\,. \tag{2.14}$$

Définition 2.6 (Sous-pavage). Un sous pavage est défini par une union de pavés. En particulier, un sous-pavage régulier est constitué d'un ensemble de pavés disjoints ou des pavés qui ne partagent que leurs frontières.

Définition 2.7 (Bissection). La bissection est une opération qui partitionne un pavé $[\mathbf{x}]$ en deux autres pavés $L\,[\mathbf{x}]$ et $R\,[\mathbf{x}]$ qui s'écrivent :

$$L\,[\mathbf{x}] \triangleq [\underline{x}_1, \overline{x}_1] \times ... \times \left[\underline{x}_j, \frac{\underline{x}_j + \overline{x}_j}{2}\right] \times ... \times [\underline{x}_n, \overline{x}_n]\,, \tag{2.15}$$

$$R\,[\mathbf{x}] \triangleq [\underline{x}_1, \overline{x}_1] \times ... \times \left[\frac{\underline{x}_j + \overline{x}_j}{2}, \overline{x}_j\right] \times ... \times [\underline{x}_n, \overline{x}_n]\,, \tag{2.16}$$

avec $i \triangleq \min\left\{\, i \middle|\, \overline{x}_i - \underline{x}_i = w\left([\mathbf{x}]\right)\right\}$ qui correspond à l'indice de la composante de plus grande longueur de $[\mathbf{x}]$. Par la suite, nous avons : $[\mathbf{x}] = L\,[\mathbf{x}] \cup R\,[\mathbf{x}]$.

2.2.2 Arithmétique des intervalles

Cette partie constitue les fondements du calcul par intervalles. L'analyse par intervalles est le premier outil de calcul ensembliste mais aussi le plus utilisé. Par rapport à d'autres représentations d'ensembles comme les intervalles modaux, les ellipsoïdes, les zonotopes, *etc*. (voir par exemple [Durieu and Walter, 2001], [Vehi et al., 1997]), il est nettement plus facile d'effectuer des calculs mathématiques sur des structures de sous-ensembles de type intervalles ou pavés. En effet, il faut relativement peu d'arguments pour définir un intervalle ou un pavé. Comme pour les réels, le calcul par intervalles manipule des opérations

arithmétiques ($+$, $-$, $\cdot$, $/$). Une opération mathématique quelconque $\circ$ entre deux sous-ensembles $\mathbb{X}$ et $\mathbb{Y}$ de $\mathbb{R}^n$ est définie comme suit :

$$\mathbb{X} \circ \mathbb{Y} \triangleq \{\mathbf{x} \circ \mathbf{y} |\ \mathbf{x} \in \mathbb{X} \text{ et } \mathbf{y} \in \mathbb{Y}\}. \tag{2.17}$$

Les opérations arithmétiques considérées ici sont stables. Autrement dit, les résultats de ces opérations sont de même type que leurs arguments. Par exemple, le résultat d'une addition ou du produit de deux intervalles est aussi un intervalle. Pour $\circ \in \{+, -, \cdot, /\}$, le récapitulatif des opérations sur les intervalles $[x]$ et $[y]$ est donné par :

$$[x] + [y] = \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right], \tag{2.18}$$

$$[x] - [y] = \left[\underline{x} - \overline{y}, \overline{x} - \underline{y}\right], \tag{2.19}$$

$$[x] \cdot [y] = \left[\min\left(\underline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\underline{y}, \overline{x}.\overline{y}\right), \max\left(\underline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\underline{y}, \overline{x}.\overline{y}\right)\right]. \tag{2.20}$$

Dans le cas de la division, nous avons

$$1/[y] = [\{1/y |\ y \in [y]\}] \tag{2.21}$$

qui représente le plus petit intervalle qui contient l'ensemble des réels $\frac{1}{y}$ pour tout y appartenant à $[y]$. Plus concrètement, il vient :

$$1/[y] = \begin{cases} \mathbb{R} \text{ si } [y] = [0,0] \\ \left[\frac{1}{\overline{y}}, \frac{1}{\underline{y}}\right] \text{ si } 0 \notin [y] \\ \left[\frac{1}{\overline{y}}, +\infty\right[\text{ si } \underline{y} = 0 \text{ et } \overline{y} > 0 \\ \left]-\infty, \frac{1}{\underline{y}}\right] \text{ si } \underline{y} < 0 \text{ et } \overline{y} = 0 \\ \mathbb{R} \text{ si } \underline{y} < 0 \text{ et } \overline{y} > 0. \end{cases} \tag{2.22}$$

Ainsi $[x] / [y] = [x] * (1/[y])$.

2.3 Fonctions d'inclusion

Nous avons vu dans les sections précédentes quelques bases du calcul par intervalles. Ici, nous allons nous intéresser à l'évaluation de fonctions vectorielles dont les variables sont des intervalles. Il sera présenté à cet effet plusieurs techniques d'évaluation plus ou moins efficaces.

2.3.1 Définition

Soit $\mathbf{f}$ une fonction vectorielle définie de $\mathbb{R}^n$ dans $\mathbb{R}^m$. Une *fonction d'inclusion* est une fonction de $\mathbb{IR}^n \rightarrow \mathbb{IR}^m$ qui satisfait,

$$[\mathbf{f}]([\mathbf{x}]) \supset \{\mathbf{f}(\mathbf{x}) | \mathbf{x} \in [\mathbf{x}]\}, \tag{2.23}$$

ce qui revient à

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}]) \tag{2.24}$$

où $[\mathbf{x}]$ est un pavé de $\mathbb{R}^n$.

Étant données les fonctions d'inclusion pour chaque composante f_i de $\mathbf{f}$ telles que : $[f_i]([\mathbf{x}]) \supset \{ f_i(\mathbf{x}) | \mathbf{x} \in [\mathbf{x}] \}$ ($i = 1, 2, ..., m$), il vient

$$[\mathbf{f}]([\mathbf{x}]) = [f_1]([\mathbf{x}]) \times [f_2]([\mathbf{x}]) \times ... \times [f_m]([\mathbf{x}]). \tag{2.25}$$

Dans le cadre d'application des méthodes intervalles, il est clair que nous avons intérêt à trouver le plus petit pavé contenant $\mathbf{f}([\mathbf{x}])$ pour tout $[\mathbf{x}] \in \mathbb{IR}^n$, c'est la *fonction d'inclusion minimale* pour $\mathbf{f}$. Nous avons : $[\mathbf{f}]^*([\mathbf{x}]) = [\mathbf{f}([\mathbf{x}])]$ où $[\mathbb{A}]$ est le plus petit pavé qui contient l'ensemble $\mathbb{A}$ (voir illustration sur la figure 2.1).

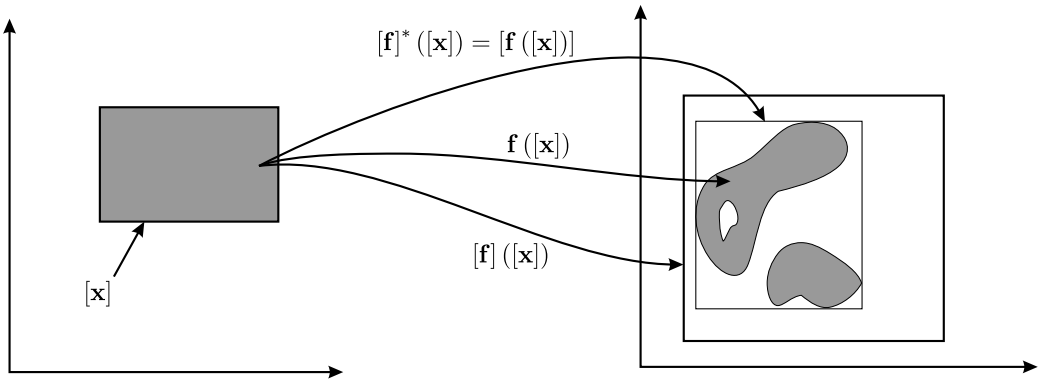

Figure 2.1 – Image d'un pavé par les fonctions d'inclusion $[\mathbf{f}]([\mathbf{x}])$, $[\mathbf{f}]^*([\mathbf{x}])$ et $\mathbf{f}([\mathbf{x}])$.

2.3.1.1 Propriétés

– Une fonction d'inclusion $[\mathbf{f}](.)$ est *convergente* si et seulement si pour toutes suites de pavés $[\mathbf{x}]_k$, nous avons

$$\lim_{k \to \infty} w([\mathbf{x}]_k) = 0 \Rightarrow \lim_{k \to \infty} w([\mathbf{f}]([\mathbf{x}]_k)) = 0 \tag{2.26}$$

ce qui entraîne,

$$\forall \mathbf{x} \in \mathbb{R}^n, [\mathbf{f}](\mathbf{x}) = \mathbf{f}(\mathbf{x}). \tag{2.27}$$

– Une fonction d'inclusion $[\mathbf{f}](.)$ est monotone si et seulement si

$$[\mathbf{x}] \subset [\mathbf{y}] \Rightarrow [\mathbf{f}]([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{y}]) \tag{2.28}$$

avec $[\mathbf{x}]$ et $[\mathbf{y}]$ sont deux pavés de $\mathbb{R}^n$.

– La fonction d'inclusion minimale est unique pour chaque fonction vectorielle.

2.3.1.2 Fonctions élémentaires

Considérons $\mathbb{E}$ et $\mathbb{F}$ deux sous-ensembles de $\mathbb{R}$. On appelle *fonctions élémentaires* toute fonction f définie de $\mathbb{E}$ dans $\mathbb{F}$ pour laquelle, nous disposons d'une fonction d'inclusion minimale $[f]^* : \mathbb{IR} \rightarrow \mathbb{IR}$, telle que $[f]^*([x]) \triangleq [\{f(x) | x \in [x]\}]$.

Toute fonction élémentaire f, continue et monotone, admet comme fonction d'inclusion minimale,

$$[f]^*([x]) = [\min(f(\underline{x}), f(\overline{x})), \max(f(\underline{x}), f(\overline{x}))]. \tag{2.29}$$

Si f n'est pas monotone, la réalisation de la fonction d'inclusion minimale dépend alors d'une étude globale des variations de la fonction. Dans ce cas, la méthode employée consiste à exploiter la "monotonie par morceaux" de f. Notons $\mathbb{Z}$ l'ensemble des entiers relatifs. Le domaine $\mathbb{E}$ est décomposé en sous-domaines $\mathbb{E}_k$ ($\mathbb{E} = \bigcup_{k\in\mathbb{Z}} \mathbb{E}_k$) dans lesquels la fonction est continue et garde un comportement monotone (strictement croissante ou décroissante). Dans la suite, nous appellerons les domaines $\mathbb{E}_k$ ($k \in \mathbb{Z}$), *les domaines de monotonie* de f. Notons que le nombre de ces domaines n'est pas nécessairement fini, par exemple dans le cas de fonctions périodiques comme $f(x) = \sin(x)$.

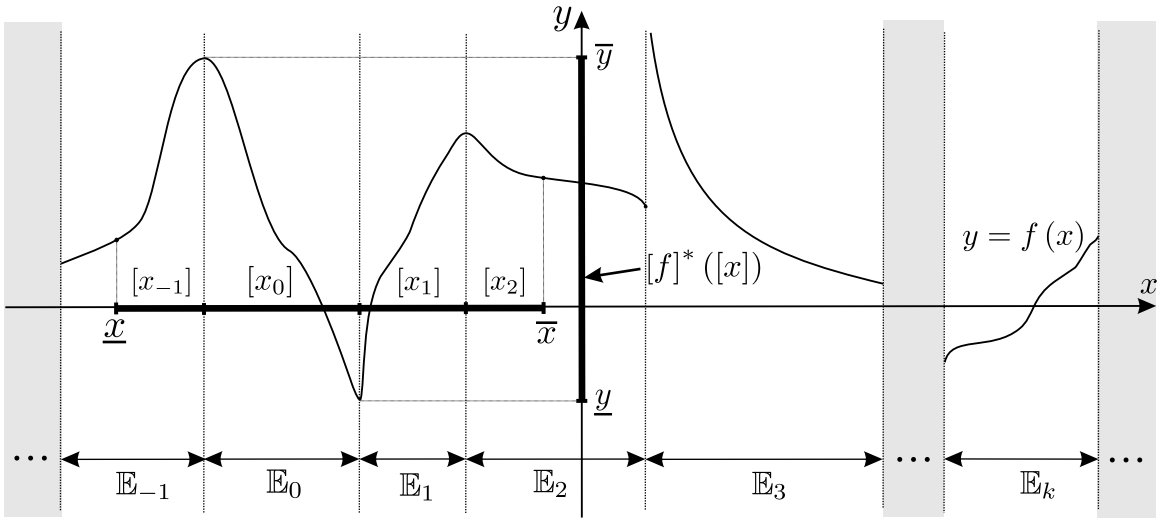

Figure 2.2 – Fonction d'inclusion minimale dans le cas d'une fonction f non monotone. Décomposition en domaines de monotonie $\mathbb{E}_k$ ($k \in \mathbb{Z}$).

Comme illustré sur la figure 2.2, nous considérons un intervalle $[x] \in \mathbb{IR}$ qui s'étend sur un nombre fini de domaines $\mathbb{E}_k$. Notons $[x_k] = [x] \cap \mathbb{E}_k$, la fonction d'inclusion minimale pour f devient :

$$[f]^*([x]) = [f]^*\left(\bigcup_{k=r}^{s} [x_k]\right) = \left[\bigcup_{k=r}^{s} [f]^*([x_k])\right] \tag{2.30}$$

avec s et r deux entiers relatifs tels que $s \geq r$. La fonction f étant continue et monotone sur $[x_k]$, il vient

$$[f]^*([x]) = \left[\bigcup_{k=r}^{s} [\min(f(\underline{x}_k), f(\overline{x}_k)), \max(f(\underline{x}_k), f(\overline{x}_k))]\right]. \tag{2.31}$$

Après développement, nous obtenons,

$$[f]^*([x]) = \left[\underline{y}, \overline{y}\right] \tag{2.32}$$

avec

$$\underline{y} = \min\left(f\left(\underline{x}_r\right), f\left(\underline{x}_{r+1}\right), ..., f\left(\underline{x}_s\right), f\left(\overline{x}_r\right), f\left(\overline{x}_{r+1}\right), ..., f\left(\overline{x}_s\right)\right) \tag{2.33}$$

et

$$\overline{y} = \max\left(f\left(\underline{x}_r\right), f\left(\underline{x}_{r+1}\right), ..., f\left(\underline{x}_s\right), f\left(\overline{x}_r\right), f\left(\overline{x}_{r+1}\right), ..., f\left(\overline{x}_s\right)\right). \tag{2.34}$$

Il est ainsi possible de définir des fonctions d'inclusion minimales pour de nombreuses fonctions élémentaires ($\cos$, $\sin$, $\tan$, $\exp$, $\log$, $(.)^n$, *etc.*).

Exemple 2.8. *La fonction exponentielle étant continue et croissante sur $\mathbb{R}$, nous appliquons la relation (2.29), afin d'obtenir :*

$$[\exp]([x]) = [\exp(\underline{x}), \exp(\overline{x})]. \tag{2.35}$$

Exemple 2.9. *Considérons la fonction $f(x) = x^2$, la fonction d'inclusion minimale pour f conformément à la figure 2.3 est donnée par,*

$$[f]^*([x]) = \begin{cases} \left[0, \max\left(\underline{x}^2, \overline{x}^2\right)\right] \textit{ si } \underline{x} \cdot \overline{x} \leq 0 \\ \left[\underline{x}^2, \overline{x}^2\right] \textit{ sinon.} \end{cases} \tag{2.36}$$

Exemple 2.10. *Soit la fonction cosinus non monotone, nous procédons de la façon suivante*

$$[\cos]([x]) = [a, b], \tag{2.37}$$

avec

$$a = \begin{cases} -1, \textit{ si } \exists\, k \in \mathbb{Z} |\, (2k+1)\,\pi \in [x] \\ \min(\cos(\underline{x}), \cos(\overline{x})), \textit{ sinon} \end{cases} \tag{2.38}$$

et

$$b = \begin{cases} 1, \textit{ si } \exists\, k \in \mathbb{Z} |\, 2k\pi \in [x] \\ \max(\cos(\underline{x}), \cos(\overline{x})), \textit{ sinon.} \end{cases} \tag{2.39}$$

Dans les sections suivantes, nous verrons les différentes méthodes d'évaluation des fonctions d'inclusion dans le cas des fonctions à plusieurs variables.

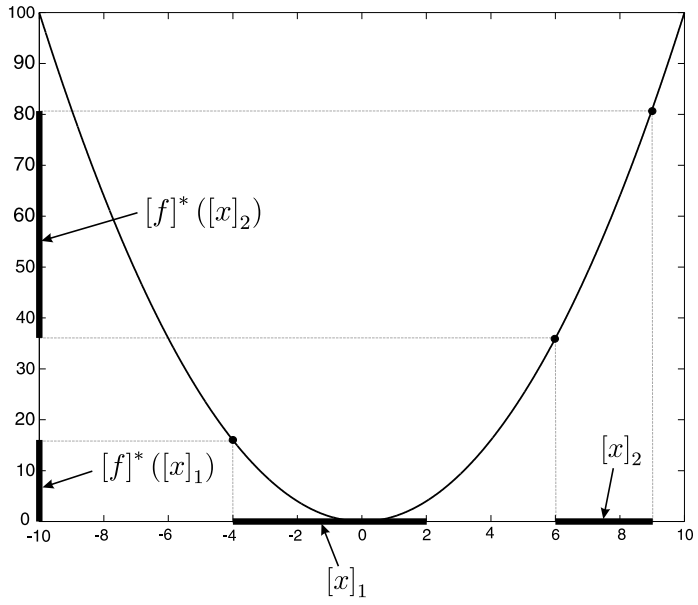

Figure 2.3 – Fonction d'inclusion minimale pour $f(x) = x^2$.

2.3.2 Fonction d'inclusion naturelle

Le théorème suivant permet de déterminer une fonction d'inclusion naturelle pour des fonctions de plusieurs variables. Ces fonctions sont construites à partir de compositions d'opérateurs arithmétiques ($+$, $-$, $\cdot$, $/$) ainsi que de fonctions élémentaires usuelles ($\tan$, $\cos$, $\sin$, $\exp$, $\log$, $\sqrt{(.)}$, *etc.*).

Théorème 2.11. *Considérons une fonction définie par*

$$\begin{array}{rccc} f: & \mathbb{R}^n & \longrightarrow & \mathbb{R} \\ & (x_1, x_2, ..., x_n) & \longmapsto & f(x_1, x_2, ..., x_n) \end{array} \tag{2.40}$$

Une fonction d'inclusion convergente $[f] : \mathbb{IR}^n \to \mathbb{IR}$ pour f est obtenue par la substitution, de chaque variable réelle x_i par son intervalle d'appartenance $[x_i]$ ($i = 1, 2, ..., n$), ainsi que chaque opérateur ou fonction élémentaire par la fonction d'inclusion minimale équivalente. La fonction d'inclusion ainsi définie est appelée fonction d'inclusion naturelle pour f.

Toutefois, la fonction d'inclusion naturelle n'est minimale qu'à certaines conditions. Nous reviendrons sur ces conditions à travers l'étude de l'exemple suivant.

Exemple 2.12. *Soit la fonction $f : \mathbb{R}^2 \to \mathbb{R}$ dont l'expression est donnée par*

$$f(x_1, x_2) = \frac{x_2}{x_1 + x_2} + \sin(x_1)\cos(x_1) \tag{2.41}$$

avec $x_1 \in [-1,2]$ *et* $x_2 \in [3,5]$. *En appliquant directement le théorème 2.11 sur* f, *nous obtenons la fonction d'inclusion naturelle suivante,*

$$[f]_1([x_1],[x_2]) = \frac{[x_2]}{[x_1]+[x_2]} + [\sin]([x_1]) \cdot [\cos]([x_1]). \tag{2.42}$$

Il vient après évaluation : $[f]_1([-1,2],[3,5]) = [-0.42, 3.5]$. *Il est possible de trouver une autre fonction d'inclusion naturelle* $[f]_2$ *pour* f, *en manipulant l'expression (2.41) puis en utilisant le théorème 2.11. Nous avons :*

$$[f]_2([x_1],[x_2]) = \frac{1}{1+[x_1]/[x_2]} + [\sin]([x_1]) \cdot [\cos]([x_1]). \tag{2.43}$$

Nous obtenons par la suite $[f]_2([-1,2],[3,5]) = [-0.2415, 2.5]$. *Enfin, par le même procédé que précédemment, il vient une troisième fonction d'inclusion* $[f]_3$ *pour* f *donnée par*

$$[f]_3([x_1],[x_2]) = \frac{1}{1+[x_1]/[x_2]} + \frac{[\sin](2 \cdot [x_1])}{2}. \tag{2.44}$$

Nous calculons $[f]_3([-1,2],[3,5]) = [0.1, 2]$.

Dans l'exemple 2.12, $[f]_1$, $[f]_2$, et $[f]_3$ représentent les extensions intervalles de la même fonction f. Pourtant nous constatons lors de l'évaluation avec les mêmes arguments intervalles, des résultats nettements différents. En effet, $[f]_3$ est ici, la fonction d'inclusion la plus précise, même si celle-ci n'est pas minimale. Le pessimisme (ou la surévaluation des bornes d'incertitudes) observé sur les trois fonctions d'inclusion dépend du nombre d'occurrences des variables intervalles. Pour $[f]_1$, il y a 3 occurrences de x_1 et 2 occurrences de x_2. Dans le cas de $[f]_2$, nous avons 3 occurrences de x_1 et une occurrence de x_2. Dans le cas de $[f]_3$, nous ne comptons plus que 2 occurrences de x_1 et une occurrence de x_2. Ainsi, plus le nombre d'occurrences des variables est important dans une fonction, plus le pessimisme sur les évaluations des fonctions d'inclusion naturelles est important. Nous expliquons cette situation par le point de vu suivant. Dans l'expression d'une fonction d'inclusion, les intervalles spécifient des variables qui n'entretiennent aucune relation de dépendance entre-elles, cette situation est connue sous le nom de *problème de dépendance* (voir [Moore, 1979]). Considérons la fonction

$$g(x) = (x \cdot x) + x + 1 = \left(x + \frac{1}{2}\right)^2 + \frac{3}{4}, \tag{2.45}$$

nous avons par définition la fonction d'inclusion naturelle

$$\begin{aligned} [g]_1([x]) &= [x] \cdot [x] + [x] + 1 \\ &= \{(x \cdot y) + z + 1 |\ x \in [x], y \in [x] \text{ et } z \in [x]\}, \end{aligned} \tag{2.46}$$

qui est différente de

$$\begin{aligned} [g]_2([x]) &= \left([x] + \tfrac{1}{2}\right)^2 + \tfrac{3}{4} \\ &= \left\{ \left(x + \tfrac{1}{2}\right)^2 + \tfrac{3}{4} \middle|\ x \in [x] \right\} \\ &= \left\{ x^2 + x + 1 \middle|\ x \in [x] \right\} \\ &= g([x]). \end{aligned} \tag{2.47}$$

D'où

$$\forall [x] \in \mathbb{IR},\ [g]_2([x]) \subset [g]_1([x]). \tag{2.48}$$

Ainsi, la fonction d'inclusion naturelle est minimale si chacune des variables n'intervient qu'une seule fois dans l'expression de la fonction.

Parmi les types d'extensions intervalles de fonctions présentés, la fonction d'inclusion naturelle reste la plus facile à mettre en oeuvre. Cependant, afin de réduire le problème de pessimisme des fonctions d'inclusion naturelles, il est utile de procéder à des manipulations algébriques sur les expressions des fonctions. Nous pouvons ainsi tenter de réduire les occurrences des différentes variables.

2.3.3 Fonction d'inclusion centrée

Ce type de fonction d'inclusion permet dans certaines situations, d'atténuer le problème de multi-occurrences des variables (voir [Hansen and Greenberg, 1983] et [Krawczyk and Neumaier, 1985]). Considérons une fonction $f : \mathbb{R}^n \to \mathbb{R}$ différentiable sur un pavé $[\mathbf{x}]$, notons la variable $\mathbf{x} = (x_1, x_2, ..., x_n)^{\mathrm{T}}$ et $\mathbf{x}_C = m([\mathbf{x}])$ le centre du pavé $[\mathbf{x}]$. Le théorème de la valeur moyenne s'écrit,

$$\forall \mathbf{x} \in [\mathbf{x}], \exists\, \mathbf{x}_0 \in [\mathbf{x}]\ \ f(\mathbf{x}) = f(\mathbf{x}_C) + \nabla f(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_C) \tag{2.49}$$

où $\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, ..., \frac{\partial f(\mathbf{x})}{\partial x_n}\right)$ représente le gradient de f. Par la suite, nous avons pour tout $\mathbf{x} \in [\mathbf{x}]$,

$$f(\mathbf{x}) \in f(\mathbf{x}_C) + \nabla f([\mathbf{x}]) \cdot (\mathbf{x} - \mathbf{x}_C). \tag{2.50}$$

Le théorème 2.11 sur la fonction d'inclusion naturelle permet d'établir la relation

$$f([\mathbf{x}]) \subset f(\mathbf{x}_C) + \nabla f([\mathbf{x}]) \cdot ([\mathbf{x}] - \mathbf{x}_C). \tag{2.51}$$

A partir de ceci, nous définissons la *fonction d'inclusion centrée* ou *forme centrée* pour f par

$$[f]_c([\mathbf{x}]) \triangleq f(\mathbf{x}_C) + [\mathbf{g}]([\mathbf{x}]) \cdot ([\mathbf{x}] - \mathbf{x}_C) \tag{2.52}$$

avec $[\mathbf{g}]([\mathbf{x}])$ qui correspond à une fonction d'inclusion pour le gradient de f. Cette fonction est construite à partir de fonctions d'inclusion naturelle ou centrée. Dans le cas où la forme centrée est utilisée, $[f]_c([\mathbf{x}])$ devient alors une fonction d'inclusion dite de Taylor notée $[f]_{\mathrm{T}}([\mathbf{x}])$. La fonction d'inclusion centrée est convergente car elle est essentiellement composée de fonctions d'inclusion naturelles qui sont convergentes.

Afin de comprendre le principe de la forme centrée, considérons une fonction scalaire à une variable $f(x)$ dérivable sur l'intervalle $[x]$. La fonction d'inclusion centrée pour f est donnée par

$$[f]_c([x]) = f(x_C) + [g]([x]) \cdot ([x] - x_C) \tag{2.53}$$

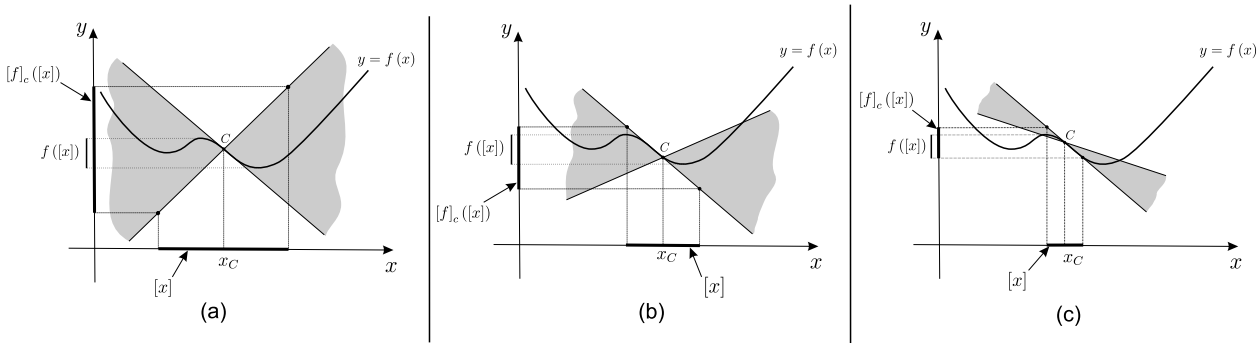

Figure 2.4 – Principe de la forme centrée dans le cas à une dimension.

avec $[g]([x])$ une fonction d'inclusion pour $f'(x)$. Les figures 2.4a, 2.4b et 2.4c représentent le principe de la forme centrée pour trois intervalles de largeurs différentes $[x]$. Sur ces figures, les secteurs ou cônes en gris sont formés par les intersections au point $C(x_C = m([x]), y_C = f(m([x])))$ des plus grande et plus petite pentes atteintes par $f(x)$ sur $[x]$.

Les différents cônes sont représentés dans un cas où nous supposons disposer de la fonction d'inclusion minimale pour la dérivée de f. Nous remarquons sur les figures 2.4a, 2.4b et 2.4c, selon la taille des intervalles $[x]$, un pessimisme nettement différent sur l'approximation de $f([x])$ par $[f]_c([x])$. En effet, le pessimisme relevé suite à l'évaluation de $[f]_c([x])$ est relativement important sur la figure 2.4a , tandis qu'il diminue sur la figure 2.4b et encore plus sur la figure 2.4c. Ces évolutions laissent présumer d'une efficacité de la méthode sur des pavés de faibles longueurs. La forme centrée sera aussi utilisée dans le cadre de fonctions à variables multi-occurrentes et dont il est difficile de réduire les occurrences par des simples manipulations algébriques.

2.3.4 Comparaison

Dans cette section, nous allons apporter quelques éléments de comparaison pour différentes fonctions d'inclusion. La plupart de ces éléments sont issus de [Moore, 1966].

Le taux de convergence d'une fonction d'inclusion $[\mathbf{f}] : \mathbb{IR}^n \rightarrow \mathbb{IR}^m$ est défini par le plus grand coefficient α tel que

$$\exists\, \beta,\ \forall\, [\mathbf{x}] \in \mathbb{IR}^n |\, w([\mathbf{f}]([\mathbf{x}])) - w([\mathbf{f}([\mathbf{x}])]) \leq \beta . w([\mathbf{x}])^{\alpha} \tag{2.54}$$

quand $w([\mathbf{x}])$ tend vers 0. Pour une fonction d'inclusion minimale le taux de convergence est infini ($\alpha \rightarrow +\infty$) car $w([\mathbf{x}]) \rightarrow 0$. Pour une fonction d'inclusion naturelle, nous avons au moins une convergence linéaire ($\alpha \geq 1$), tandis que la convergence d'une fonction d'inclusion centrée ou de Taylor est au moins d'ordre quadratique ($\alpha \geq 2$). Par conséquent, pour des pavés infinitésimaux, l'évaluation d'une

forme centrée sera moins pessimiste que celle d'une fonction d'inclusion naturelle. Par contre pour des pavés relativement volumineux et des fonctions avec peu de dépendances multi-occurrentes, nous privilégierons l'utilisation de fonctions d'inclusion naturelles.

Reprenons l'exemple 2.12 où $f(x_1, x_2) = \frac{1}{1+x_1/x_2} + \frac{\sin(2 \cdot x_1)}{2}$. Nous cherchons à comparer pour f, l'efficacité de la fonction d'inclusion naturelle et de la forme centrée. Pour cela, nous considérons 3 pavés de longueurs et de volumes différents : $[\mathbf{x}]_1 = [-1, 2] \times [3, 5]$, $[\mathbf{x}]_2 = [-0.1, 0.2] \times [3.9, 4.1]$ et $[\mathbf{x}]_3 = [-0.01, 0.02] \times [3.98, 4.02]$. Nous constatons effectivement que le pavé $[\mathbf{x}]_1$ est de longueur et de volume supérieurs à ceux des deux autres pavés $[\mathbf{x}]_2$ et $[\mathbf{x}]_3$. La forme centrée pour f s'écrit :

$$\begin{aligned} [f]_c([\mathbf{x}]) &= f(m_1, m_2) + [g_1]([\mathbf{x}]) \cdot ([x_1] - m_1) \\ &\quad + [g_2]([\mathbf{x}]) \cdot ([x_2] - m_2) \end{aligned} \tag{2.55}$$

avec $[g_1]([\mathbf{x}])$ et $[g_2]([\mathbf{x}])$ qui sont respectivement les fonctions d'inclusion de $\frac{\partial f(x_1,x_2)}{\partial x_1}$ et $\frac{\partial f(x_1,x_2)}{\partial x_2}$, m_1 et m_2 sont les centres respectifs des intervalles de $[x_1]$ et $[x_2]$. Nous avons plus précisément

$$\begin{aligned} [f]_c([\mathbf{x}]) &= f(m_1, m_2) \\ &\quad + \left(\cos(2 \cdot [x_1]) - \frac{[x_1]}{[x_2] \cdot (1+[x_1]/[x_2])^2}\right) \cdot ([x_1] - m_1) \\ &\quad + \frac{[x_2]-m_2}{[x_2]^2 \cdot (1+[x_1]/[x_2])^2}. \end{aligned} \tag{2.56}$$

Les résultats obtenus à l'issu de l'évaluation des fonctions d'inclusion naturelle $[f]_n$ et centrée $[f]_c$ sont consignés dans le tableau ci-dessous.

	$[\mathbf{x}] = [-1, 2] \times [3, 5]$,	$[\mathbf{x}] = [-0.1, 0.2] \times [3.9, 4.1]$,	$[\mathbf{x}] = [-0.01, 0.02] \times [3.98, 4.02]$,
$w([\mathbf{x}])$	3	0.3	0.03
$[f]_n([\mathbf{x}])$	$[0.1, 2]$	$[0.8518, 1.221025]$	$[0.985, 1.022514]$
$[f]_c([\mathbf{x}])$	$[-1.8154, 4.435]$	$[0.91928, 1.155852]$	$[0.9924, 1.0151]$
$\Delta([f]([\mathbf{x}]))$	4.3504	-0.132653	-0.014814

(2.57)

Dans ce tableau, nous avons calculé les erreurs entre les largeurs des intervalles évalués : $\Delta([f]([\mathbf{x}])) = w([f]_c([\mathbf{x}])) - w([f]_n([\mathbf{x}]))$. Ces résultats viennent confirmer les tendances et les comportements attendus selon l'expression (2.54). Dans le premier cas ($[\mathbf{x}] = [-1, 2] \times [3, 5]$), la fonction d'inclusion naturelle se révèle nettement moins pessimiste que la forme centrée ($\Delta([f]([\mathbf{x}])) < 0$). Dans les deux autres cas ($[\mathbf{x}] = [-0.1, 0.2] \times [3.9, 4.1]$ et $[\mathbf{x}] = [-0.01, 0.02] \times [3.98, 4.02]$), les pavés sont beaucoup plus petits au sens du volume et de la longueur, il se trouve que la fonction d'inclusion naturelle devient moins efficace donc plus pessimiste que la forme centrée, en effet nous remarquons pour ces cas que l'erreur $\Delta([f]([\mathbf{x}]))$ est négative.

Finalement, nous pouvons envisager dans la majorité des cas, l'utilisation d'une fonction d'inclusion hybride entre fonction d'inclusion naturelle et forme centrée, nous aurons alors :

$$[f]_{nc}([\mathbf{x}]) = [f]_n([\mathbf{x}]) \cap [f]_c([\mathbf{x}]). \tag{2.58}$$

2.4 Conclusion

Dans ce chapitre, nous avons présenté les différents éléments du calcul et de l'analyse par intervalles. En d'autres termes, nous avons décrit comment et dans quel cadre, il était possible d'effectuer des calculs non pas avec des réels mais avec des données de type intervalle. En cela, la notion de fonction d'inclusion constitue la base des algorithmes ensemblistes développés dans les chapitres suivants. Ici, nous avons choisi de présenter deux types de réalisation de fonctions d'inclusion : la fonction d'inclusion naturelle et la forme centrée. Nous avons ensuite discuté de l'efficacité de ces différentes extensions intervalles sur un exemple de fonction. Nous avons ainsi montré que la fonction d'inclusion naturelle était efficace pour des pavés larges et volumineux, alors que la forme centrée retrouvait sa précision sur des petits pavés.

Pour tous les vecteurs issus d'un pavé, une fonction d'inclusion permet de déterminer un pavé qui contient toutes les images par une fonction vectorielle. Ceci est essentiel à la résolution d'une multitude de problèmes. Par exemple pour démontrer que des inégalités non linéaires n'admettent aucune solution dans un compact, ou encore afin de prouver en un temps fini, que tous les vecteurs d'un pavé satisfont un système d'inégalités. Dans les prochains chapitres, ce type d'information sera d'une grande utilité pour résoudre des problèmes d'automatique réputés difficiles tel que des problèmes d'estimation non linéaire, d'optimisation ou de commande robuste.

CHAPITRE 3

Contraction de domaines sous contraintes (CSP et contracteurs)

La résolution d'inégalités non linéaires est un problème NP-difficile. Dans l'état actuel des connaissances scientifiques, il n'est pas envisageable de déterminer dans un temps fini, l'ensemble des solutions à des inégalités non linéaires. Les approches ensemblistes et les heuristiques (recuit simulé, recherche tabou, etc.) élaborent une exploration et une énumération des vecteurs solutions *a priori* possibles dans un compact de $\mathbb{R}^n$. Bien entendu, ces méthodes et algorithmes obéissent à des lois de complexité exponentielle, elles deviennent moins efficaces à mesure que le nombre de variables à déterminer augmente.

Le terme de *réduction de pavés* est propre au fait de "filtrer" un pavé ou un domaine initial, les vecteurs qui ne satisfont pas une ou plusieurs contraintes. Au départ, les méthodes utilisées pour la réduction de pavés ont été développées pour résoudre des inégalités dont les variables appartiennent à des domaines discrets (voir [Waltz, 1975]). Ces approches ont ensuite été étendues à des problèmes dont les variables ou paramètres appartiennent à des domaines continus, qui sont ici des pavés (voir [Cleary, 1987] et [Davis, 1987]). Ce chapitre n'a pas la prétention de trouver les solutions d'inégalités non linéaires. Le problème auquel nous nous consacrerons est l'approximation du plus petit pavé enveloppe de l'ensemble des solutions possibles. Comme nous allons le voir, concevoir des algorithmes de complexité polynomiale permettant de trouver un tel pavé n'est déjà pas une tâche aisée.

La section 3.1 fait état de rappels sur les notions fondamentales de satisfaction de contraintes (CSP) et de consistances. Dans cette même section, une différenciation entre consistance locale et consistance globale sera donnée. La section 3.2 est consacrée aux méthodes de consistance plus ou moins fortes par rapport à des contraintes. Il existe de multiples méthodes de réduction de pavé, méthode de Newton par intervalles, linéarisation parallèle, ajout de contraintes redondantes et

dynamiques (voir [Hansen and Greenberg, 1983], [Lhomme, 1993], [Floudas and Visweswaran, 1993], [Benhamou et al., 1994], [Benhamou et al., 1999], [van Hentenryck et al., 1998], *etc.*). Parmi celles-ci, avons privilégié comme méthodes de consistance local, le développement de contracteurs primitifs pour chaque opération ou fonction mathématique, associé à la propagation de contraintes. Dans [Braems, 2002], des techniques similaires ont été utilisées pour réaliser certains contracteurs, nous présenterons dans ce chapitre un principe global pour déterminer le contracteur optimal pour les contraintes binaires et ternaires (voir section 3.2.1.1). Nous proposerons ensuite quelques améliorations de ces méthodes afin d'obtenir une consistance plus forte. Dans la section 3.3, les différentes méthodes de consistance seront testées sur un exemple de localisation d'une plate-forme dans un plan. Enfin, les commentaires sur l'efficacité ainsi que les difficultés de mise en oeuvre de chaque algorithme feront l'objet de la section 3.4.

3.1 Contraintes et Consistance

3.1.1 Problème de satisfaction de contraintes (CSP)

Définition 3.1 (Contrainte). Étant donné un vecteur $\mathbf{x}$ à n composantes de variables réelles x_1, x_2,..., x_n, une contrainte $\mathcal{C}$ est une relation définie sur les composantes de $\mathbf{x}$. Cette relation spécifie un ensemble de vecteurs *compatibles* dans $\mathbb{R}^n$. Il existe plusieurs sortes de contraintes parmi lesquelles, nous avons les contraintes floues (voir [Zadeh, 1971] et [Dubois and Prade, 1980]), probabilistes , *etc.* Dans notre cas, les relations seront définies par des équations ou inéquations non linéaires.

Exemple 3.2. *Soit le vecteur réel* $\mathbf{v} = (x, y, z)^T$*, la contrainte sphérique* $\mathcal{C}_{sphère}$ *pour le vecteur* $\mathbf{v} \in \mathbb{R}^3$ *s'écrit :*

$$\mathcal{C}_{sphère} : \ ||\mathbf{v}||_2 = 1 \Rightarrow x^2 + y^2 + z^2 = 1, \tag{3.1}$$

où $||\mathbf{v}||_2$ *est la norme euclidienne du vecteur* $\mathbf{v}$.

Définition 3.3 (CSP : *Constraint Satisfaction Problem*). Un problème de satisfaction de contraintes est caractérisé par un triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$. Si n est le nombre de variables et m le nombre de contraintes, alors :

- $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ représente l'ensemble des variables,
- $\mathcal{D} = \{D_1, D_2, ..., D_n\}$ est l'ensemble de domaines associés à chaque variable,
- $\mathcal{C} = \{C_1, C_2, ..., C_m\}$ correspond à l'ensemble des contraintes.

Remarque 3.4. *Nous nous intéresserons particulièrement aux CSPs qui sont sous la forme du triplet* $(\mathbf{x}, [\mathbf{x}], \mathbf{f})$*, où* $\mathbf{x}$ *est un vecteur de* $\mathbb{R}^n$*,* $[\mathbf{x}]$ *un pavé de* $\mathbb{R}^n$ *et* $\mathbf{f}$ *une fonction définie de* $\mathbb{R}^n$ *dans* $\mathbb{R}^m$*. Une*

contrainte entre les variables de $\mathbf{f}$ *peut se mettre sous la forme :*

$$\mathcal{C}: \begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ \vdots \\ f_m(x_1, x_2, ..., x_n) = 0 \end{cases} \quad \textit{tel que :}\ x_i \in [x_i]\,, \forall i \in \{1, 2, ..., n\} \tag{3.2}$$

$$\Rightarrow \begin{cases} \mathbf{f}(\mathbf{x}) = \mathbf{0} \\ \mathbf{x} \in [\mathbf{x}]\,. \end{cases} \tag{3.3}$$

Dans ce document, nous utiliserons la même notation pour désigner tout CSP composé d'inégalités. En effet la plupart de ces contraintes peuvent s'écrire de la même façon que le système (3.3). Il est possible de montrer ceci de la façon suivante.

Considérons la contrainte $\mathcal{C}_1$ sur la variable vectorielle $\mathbf{x}$ et une fonction $\mathbf{g}$ définie de $\mathbb{R}^n$ dans $\mathbb{R}^m$,

$$\mathcal{C}_1 : \mathbf{g}(\mathbf{x}) \geq \mathbf{a}\textbf{,} \text{ tel que } \mathbf{x} \in \mathbb{R}^n, \tag{3.4}$$

où $\mathbf{a}$ est une constante vectorielle de $\mathbb{R}^m$. L'inégalité de (3.4) s'écrit aussi :

$$\mathbf{g}(\mathbf{x}) \in [\mathbf{a}]\,. \tag{3.5}$$

avec $[\mathbf{a}]$ un pavé de $\mathbb{R}^m$ tel que $[\mathbf{a}] = [a_1, +\infty[\times ... \times [a_m, +\infty[$. Nous avons :

$$\mathcal{C}_1 : \begin{cases} \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{g}(\mathbf{x}) - \mathbf{y} = \mathbf{0} \\ (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times [\mathbf{a}]. \end{cases} \tag{3.6}$$

Ainsi, nous retrouvons bien la forme du système (3.3). Nous montrons de la même façon que les contraintes $\mathcal{C}_2$ et $\mathcal{C}_3$ dont les expressions sont données par :

$$\begin{aligned} \mathcal{C}_2 \quad &: \quad \mathbf{g}(\mathbf{x}) \leq \mathbf{a}\textbf{,} \text{ tel que } \mathbf{x} \in \mathbb{R}^n \text{ et } \mathbf{a} \in \mathbb{R}^m, \\ \mathcal{C}_3 \quad &: \quad \mathbf{g}(\mathbf{x}) = \mathbf{a}\textbf{,} \text{ tel que } \mathbf{x} \in \mathbb{R}^n \text{ et } \mathbf{a} \in \mathbb{R}^m, \end{aligned}$$

s'écrivent de la même façon que (3.3).

3.1.2 Notions de consistance

La résolution d'un CSP consiste à réduire "au mieux" les domaines d'appartenance des variables afin d'en éliminer les vecteurs *inconsistants*, c'est-à-dire des vecteurs qui sont incompatibles avec l'une des contraintes du CSP. Considérons un CSP défini par l'expression (3.3). L'ensemble des points *consistants* par rapport à ce CSP est donné comme suit

$$\mathbb{S} = \{\, \mathbf{x} \in [\mathbf{x}] |\ \mathbf{f}(\mathbf{x}) = \mathbf{0} \}\,. \tag{3.7}$$

Reprenons le cas de l'exemple 3.2, un CSP pour la contrainte sphérique est défini par

$$\begin{cases} x^2+y^2+z^2=1 \\ (x,y,z)\in[x]\times[y]\times[z]\,. \end{cases} \tag{3.8}$$

Dans un premier cas, supposons que $[x]=[-1,1]$, $[y]=[-1,1]$ et $[z]=[-1,1]$, alors les points consistants avec le CSP (3.8) sont donnés par

$$\mathbb{S}_1=\mathcal{D}_{r=1}=\left\{(x,y,z)\in[-1,1]^{\times 3}\,\middle|\; x^2+y^2+z^2=1\right\}, \tag{3.9}$$

où $\mathcal{D}_{r=1}$ est l'ensemble des points situés sur la surface d'une sphère en trois dimensions de rayon $r=1$. Dans un deuxième cas : $(x,y,z)\in\left[-\frac{1}{2},\frac{1}{2}\right]^{\times 3}$, l'ensemble des points consistants pour la contrainte devient

$$\mathbb{S}_2{=}\mathcal{D}_{r=1}\cap\left[-\frac{1}{2},\frac{1}{2}\right]^{\times 3}=\emptyset. \tag{3.10}$$

Autrement dit tous les points du pavé $\left[-\frac{1}{2},\frac{1}{2}\right]^{\times 3}$ sont inconsistants avec la contrainte $x^2+y^2+z^2=1$. Comme nous avons pu le constater, la consistance d'un point est une notion assez simple. Elle se traduit par la satisfaction de chacune des inégalités qui composent la contrainte. Par contre, qu'en est-il pour un domaine continu : un intervalle ou un pavé de $\mathbb{R}^n$ en général ? Quand pouvons nous affirmer et, sous quelles conditions, qu'un pavé est consistant par rapport à des contraintes ?

Étant donnés des ensembles $\mathbb{S}_k$ définis par les d'inégalités $f_k\left(x_1,x_2,...,x_n\right)\in[y_k]$ avec $k=1,2,...,m$. Les variables appartiennent *a priori* à un pavé noté $[\mathbf{x}]_0$. Afin de dissocier les variables, posons ${}^i\mathbf{u}=(x_1,x_2,...,x_{i-1})$, ${}^i\mathbf{v}=(x_{i+1},x_{i+2},...,x_n)$ et

$${}^i[\mathbf{x}]_0=[x_1]_0\times...\times[x_{i-1}]_0\times[x_{i+1}]_0\times...\times[x_n]_0\,. \tag{3.11}$$

La notion de consistance pour un pavé peut être définie de deux façons.

Définition 3.5 (Consistance locale). Un pavé $[\mathbf{x}]$ est dit *localement consistant* si pour tout $i=1,2,...,n$, nous avons

$$\begin{aligned}[x_i] \;\subset\; \big\{ x_i\in[x_i]_0 |\; &\exists\left({}^i\mathbf{u}^1,{}^i\mathbf{v}^1\right)\in{}^i[\mathbf{x}]_0\,,\;\left({}^i\mathbf{u}^1,x_i,{}^i\mathbf{v}^1\right)\in\mathbb{S}_1,\\ &\exists\left({}^i\mathbf{u}^2,{}^i\mathbf{v}^2\right)\in{}^i[\mathbf{x}]_0\,,\;\left({}^i\mathbf{u}^2,x_i,{}^i\mathbf{v}^2\right)\in\mathbb{S}_2,\\ &...,\exists\left({}^i\mathbf{u}^m,{}^i\mathbf{v}^m\right)\in{}^i[\mathbf{x}]_0\,,\;\left({}^i\mathbf{u}^m,x_i,{}^i\mathbf{v}^m\right)\in\mathbb{S}_m\big\}\,,\end{aligned} \tag{3.12}$$

qui devient

$$\begin{aligned}[x_i] \;\subset\; \big\{ x_i\in[x_i]_0 |\exists\;&\left({}^i\mathbf{u}^1,{}^i\mathbf{v}^1\right)\in{}^i[\mathbf{x}]_0\,,\;f_1\left({}^i\mathbf{u}^1,x_i,{}^i\mathbf{v}^1\right)\in[y_1]\,,\\ \exists\;&\left({}^i\mathbf{u}^2,{}^i\mathbf{v}^2\right)\in{}^i[\mathbf{x}]_0\,,\;f_2\left({}^i\mathbf{u}^2,x_i,{}^i\mathbf{v}^2\right)\in[y_2]\,,\\ ...,\exists\;&\left({}^i\mathbf{u}^m,{}^i\mathbf{v}^m\right)\in{}^i[\mathbf{x}]_0\,,\;f_2\left({}^i\mathbf{u}^m,x_i,{}^i\mathbf{v}^m\right)\in[y_m]\big\}\,.\end{aligned} \tag{3.13}$$

Définition 3.6 (Consistance globale). Un pavé $[\mathbf{x}]$ est *globalement consistant* si chacune de ces composantes intervalles vérifient

$$\begin{aligned}[x_i] \quad \subset \quad & \left\{ x_i \in [x_i]_0 | \, \exists \ \left({}^i\mathbf{u}, {}^i\mathbf{v}\right) \in {}^i[\mathbf{x}]_0 \,, \ \left({}^i\mathbf{u}, x_i, {}^i\mathbf{v}\right) \in \mathbb{S}_1, \ \left({}^i\mathbf{u}, x_i, {}^i\mathbf{v}\right) \in \mathbb{S}_2, \right. \\ & \left. ..., \ \left({}^i\mathbf{u}, x_i, {}^i\mathbf{v}\right) \in \mathbb{S}_m \right\}, \end{aligned} \tag{3.14}$$

il en résulte,

$$[\mathbf{x}] \subset (\mathbb{S}_1 \cap \mathbb{S}_2 \cap ... \cap \mathbb{S}_m) \,. \tag{3.15}$$

En remplaçant les ensembles par les inégalités qui les définissent, nous obtenons

$$\begin{aligned}[x_i] \quad \subset \quad & \left\{ x_i \in [x_i]_0 | \, \exists \ \left({}^i\mathbf{u}, {}^i\mathbf{v}\right) \in {}^i[\mathbf{x}]_0 \,, \ f_1\left({}^i\mathbf{u}, x_i, {}^i\mathbf{v}\right) \in [y_1] \,, \right. \\ & \left. f_2\left({}^i\mathbf{u}, x_i, {}^i\mathbf{v}\right) \in [y_2] \,, ..., \ f_m\left({}^i\mathbf{u}, x_i, {}^i\mathbf{v}\right) \in [y_m] \right\}. \end{aligned} \tag{3.16}$$

Globalement, il vient

$$[\mathbf{x}] \subset \{ \mathbf{x} \in [\mathbf{x}]_0 | \ \mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \} \,. \tag{3.17}$$

Remarque 3.7. *Bien entendu un pavé globalement consistant est nécessairement localement consistant, pour des raisons évidentes la réciproque est fausse.*

La figure 3.1 illustre cette notion de consistance locale et globale par rapport à la conjonction de deux contraintes représentées par $\mathbb{S}_1$ et $\mathbb{S}_2$. Sur cette figure, $[\mathbf{x}]_1$ et $[\mathbf{x}]_2$ sont respectivement les plus grands pavés localement et globalement consistant vis-à-vis de $\mathbb{S}_1 \cap \mathbb{S}_2$. Si nous prenons le cas du pavé $[\mathbf{x}]_1 = [x_1]_1 \times [x_2]_1$ et conformément à la définition de "consistance locale", nous avons : $\forall x \in [x_1]_1$, $\exists y_1 \in [x_2]_1$, $\exists y_2 \in [x_2]_1$ tel que

$$(x, y_1) \in \mathbb{S}_1 \text{ et } (x, y_2) \in \mathbb{S}_2. \tag{3.18}$$

De même $\forall y \in [x_2]_1$, $\exists x_1 \in [x_1]_1$, $\exists x_2 \in [x_1]_1$ tel que

$$(x_1, y) \in \mathbb{S}_1 \text{ et } (x_2, y) \in \mathbb{S}_2. \tag{3.19}$$

Dans le cas du pavé globalement consistant $[\mathbf{x}]_2 = [x_1]_2 \times [x_2]_2$, il vient :

$$\forall x \in [x_1]_2 \,, \exists y \in [x_2]_2 \text{ tel que } (x, y) \in \mathbb{S}_1 \cap \mathbb{S}_2. \tag{3.20}$$

Réciproquement

$$\forall y \in [x_2]_2 \,, \exists x \in [x_1]_2 \text{ tel que } (x, y) \in \mathbb{S}_1 \cap \mathbb{S}_2. \tag{3.21}$$

Nous passons dans la suite à la présentation de quelques méthodes de réduction de domaines pour les variables. En effet, nous allons aborder les approches qui permettent d'obtenir dans l'idéal les pavés $[\mathbf{x}]_1$ et $[\mathbf{x}]_2$ à partir d'un domaine initial $[\mathbf{x}]_0$.

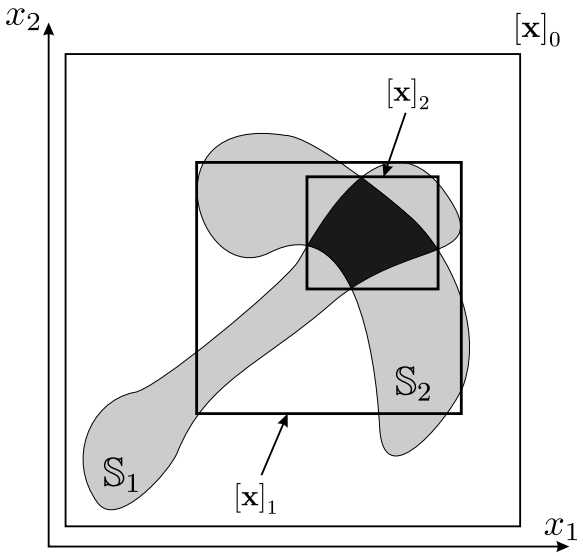

Figure 3.1 – Consistance locale et globale. Les pavés $[\mathbf{x}]_1$ et $[\mathbf{x}]_2$ sont respectivement les plus grands pavés localement et globalement consistant vis-à-vis de $\mathbb{S}_1 \cap \mathbb{S}_2$.

3.2 Méthodes de consistance

Comme cela a déjà été souligné, cette section a pour objectif la description des concepts et principes pour la recherche de pavés localement et globalement consistants. L'une des premières notions que nous définirons est la notion de contracteur pour ensembles.

Définition 3.8 (Contracteur). Soit $[\mathbf{x}]$ un pavé de $\mathbb{R}^n$ et $\mathbb{S}$ un ensemble défini par des inégalités non linéaires, un *contracteur* pour $\mathbb{S}$ est une application

$$\begin{array}{rl} \mathcal{C}_{\mathbb{S}}: & \mathbb{IR}^n \longrightarrow \mathbb{IR}^n \\ & [\mathbf{x}] \longmapsto \mathcal{C}_{\mathbb{S}}\left([\mathbf{x}]\right) \subset \left[[\mathbf{x}] \cap \mathbb{S}\right], \end{array} \tag{3.22}$$

où $\mathbb{IR}^n$ est l'ensemble des pavés de $\mathbb{R}^n$ et $[\mathbb{A}]$ le plus petit pavé qui contient l'ensemble $\mathbb{A}$. Une illustration de la contraction de $[\mathbf{x}]$ par rapport $\mathbb{S}$ est donnée par la figure 3.2.

Les différentes propriétés associées aux contracteurs sont tirées de [Jaulin et al., 2001].

$\mathcal{C}^*_{\mathbb{S}}$ est optimal si $\quad \forall\, [\mathbf{x}] \in \mathbb{IR}^{n_x}, \mathcal{C}^*_{\mathbb{S}}\left([\mathbf{x}]\right) = \left[[\mathbf{x}] \cap \mathbb{S}\right],$

$\mathcal{C}_{\mathbb{S}}$ est monotone si $\quad [\mathbf{x}] \subset [\mathbf{y}] \Rightarrow \mathcal{C}_{\mathbb{S}}\left([\mathbf{x}]\right) \subset \mathcal{C}_{\mathbb{S}}\left([\mathbf{y}]\right),$

$\mathcal{C}_{\mathbb{S}}$ est idempotent si $\quad \forall\, [\mathbf{x}] \in \mathbb{IR}^{n_x}, \mathcal{C}_{\mathbb{S}}\left(\mathcal{C}_{\mathbb{S}}\left([\mathbf{x}]\right)\right) = \mathcal{C}_{\mathbb{S}}\left([\mathbf{x}]\right),$

$\mathcal{C}_{\mathbb{S}}$ est plus efficace que $\mathcal{C}'_{\mathbb{S}}$ $\quad [\mathbf{x}] \in \mathbb{IR}^{n_x} \Rightarrow \mathcal{C}_{\mathbb{S}}\left([\mathbf{x}]\right) \subset \mathcal{C}'_{\mathbb{S}}\left([\mathbf{x}]\right).$

Soient $\mathcal{C}_{\mathbb{S}_1}$ et $\mathcal{C}_{\mathbb{S}_2}$ deux contracteurs monotones associés respectivement aux ensembles $\mathbb{S}_1$ et $\mathbb{S}_2$, nous avons par définition :

$$\mathcal{C}_{\mathbb{S}_1} \cap \mathcal{C}_{\mathbb{S}_2}\left([\mathbf{x}]\right) \triangleq \mathcal{C}_{\mathbb{S}_1}\left([\mathbf{x}]\right) \cap \mathcal{C}_{\mathbb{S}_2}\left([\mathbf{x}]\right), \tag{3.23}$$

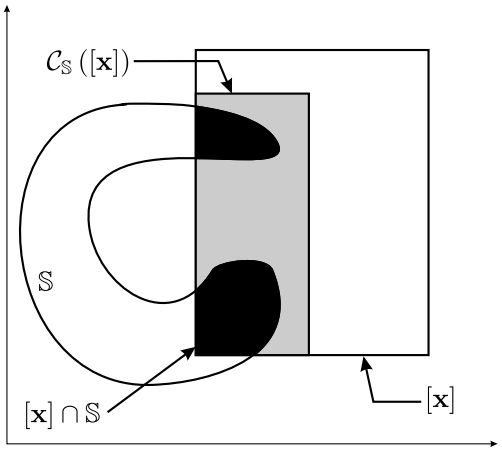

Figure 3.2 – Contraction d'un pavé $[\mathbf{x}]$ par rapport à $\mathbb{S}$.

$$\mathcal{C}_{\mathbb{S}_1} \cup \mathcal{C}_{\mathbb{S}_2}([\mathbf{x}]) \stackrel{\Delta}{=} [\mathcal{C}_{\mathbb{S}_1}([\mathbf{x}]) \cup \mathcal{C}_{\mathbb{S}_2}([\mathbf{x}])]\,. \tag{3.24}$$

A partir des relations (3.23) et (3.24), il est trivial de montrer que les propriétés suivantes sont vraies :

- $\mathbb{S}_1 \subset \mathbb{S}_2 \Rightarrow \mathcal{C}_{\mathbb{S}_2}$ est aussi un contracteur pour $\mathbb{S}_1$,
- $\mathcal{C}_{\mathbb{S}_1} \cap \mathcal{C}_{\mathbb{S}_2}$ est un contracteur pour $\mathbb{S}_1 \cap \mathbb{S}_2$,
- $\mathcal{C}_{\mathbb{S}_1} \cup \mathcal{C}_{\mathbb{S}_2}$ est un contracteur pour $\mathbb{S}_1 \cup \mathbb{S}_2$.

Il n'existe pas de méthode générale qui permette de trouver le contracteur optimal pour tout ensemble défini par des inégalités. Cependant plusieurs techniques de réduction de pavé sous des contraintes ont été étudiées et analysées dans [Lhomme, 1993], [Floudas and Visweswaran, 1993], [Benhamou et al., 1994], [Benhamou et al., 1999] et [van Hentenryck et al., 1998]. Le terme de consistance faible désigne pour un pavé, la satisfaction de la propriété de consistance locale. En effet, dans la plupart des cas, les contracteurs dits de "consistance faible" convergent vers le plus grand pavé localement consistant vis-à-vis de contraintes. En ce qui concerne la recherche d'un pavé globalement consistant, nous utiliserons la combinaison de techniques de consistances locales et de partitionnement du domaine d'appartenance des variables. Le contracteur ainsi obtenu, ne garanti pas la convergence vers le plus grand pavé globalement consistant, néanmoins il permet de s'en approcher, nous parlerons alors de contracteur de forte consistance.

3.2.1 Méthode de consistance faible

Nous présentons ici, le principe de *la propagation de contraintes sur les intervalles* (voir [Davis, 1987], [Cleary, 1987] et [Waltz, 1975]). Cette méthode servira à la réalisation de la plupart de nos contracteurs. La propagation des contraintes sur les intervalles permet, sur un domaine de variables prédéfini, une

réduction considérable. Les parties du domaine qui sont localement inconsistantes par rapport à une contrainte peuvent être systématiquement réduites à un domaine vide. Cette section rappelle les techniques de réduction de domaines basées sur la décomposition des contraintes en contraintes primitives et, la propagation de contraintes sur les intervalles. Dans cette section, notre contribution s'articule autour de l'élaboration d'une approche générale qui détermine un contracteur optimal pour certains types de contraintes.

3.2.1.1 Contracteurs primitifs

Les relations du type :

$$x_1 = x_2 \diamond x_3 \tag{3.25}$$

ou

$$x_1 = g\left(x_2\right) \tag{3.26}$$

avec $\diamond \in \{+, -, \cdot, /, \min, \max ...\}$ et $g\left(.\right) \in \{\exp\left(.\right), \log\left(.\right), \sin\left(.\right), \left(.\right)^n, ...\}$ sont appelées *contraintes primitives*. L'intérêt de la décomposition des inégalités en contraintes primitives réside dans le fait que nous disposons *a priori*, de contracteurs optimaux pour ces types de contraintes.

3.2.1.1.a Principe Considérons dans un premier temps, le cas des contraintes binaires (3.26). La réalisation des contracteurs primitifs est fondée sur les fonctions d'inclusion élémentaires (voir section 2.3.1.2). Notre objectif est de déterminer le contracteur optimal pour l'ensemble $\mathbb{S}$ des couples (x_1, x_2) qui satisfont $x_2 - g(x_1) = 0$, g est une fonction définie de $\mathbb{E}$ dans $\mathbb{F}$ deux sous-ensembles de $\mathbb{R}$. Notons $\mathbb{E}_k$ (k un entier relatif) les domaines de monotonie [1] pour g. Sur chacun de ces domaines, c'est à dire pour $k \in \mathbb{Z}$, g est substituée par des fonctions g_k telles que :

$$\forall x_1 \in \mathbb{E}_k,\ g_k\left(x_1\right) = g\left(x_1\right) \tag{3.27}$$

et pour lesquelles il existe des fonctions réciproques g_k^{-1}.

Soit un pavé $[\mathbf{x}] \in \mathbb{IR}^2$, posons

$$[\mathbf{u}]_k = [\mathbf{x}] \cap \left\{\mathbb{E}_k \times \mathbb{F}\right\}, \tag{3.28}$$

avec $r \leq k \leq s$ car $[\mathbf{x}]$ s'étend sur un nombre fini de domaines $\mathbb{E}_k \times \mathbb{F}$. Le contracteur optimal pour la contrainte $x_2 = g\left(x_1\right)$ est donné par,

$$\mathcal{C}_{\mathbb{S}}^{*}\left([\mathbf{x}]\right) = [v] \times [w]\,. \tag{3.29}$$

[1] Domaines dans lesquels nous considérons qu'une fonction est continue et monotone

Commençons par le calcul de l'intervalle $[w]$. Par définition, il vient

$$\begin{aligned}[w] &= [x_2] \cap [g]^* ([x_1]) \\ &= [x_2] \cap [g]^* \left(\bigcup_{k=r}^{s} \{[u_1]_k\} \right),\end{aligned} \tag{3.30}$$

en développant, nous obtenons

$$[w] = [x_2] \cap \left[\bigcup_{k=r}^{s} \{[g]^* ([u_1]_k)\} \right], \tag{3.31}$$

nous rappelons au passage que $[\mathbb{A}]$ correspond au plus petit intervalle qui contient l'ensemble $\mathbb{A} \subset \mathbb{R}$. La fonction g est monotone sur $[u_1]_k = \left[\underline{u}_{1,k}, \overline{u}_{1,k}\right]$ pour $k \in \mathbb{Z}$, d'où

$$[w] = [x_2] \cap \left[\bigcup_{k=r}^{s} \left[\min \left(g\left(\underline{u}_{1,k}\right), g\left(\overline{u}_{1,k}\right) \right), \max \left(g\left(\underline{u}_{1,k}\right), g\left(\overline{u}_{1,k}\right) \right) \right] \right], \tag{3.32}$$

qui devient après calcul

$$[w] = [x_2] \cap [a, b], \tag{3.33}$$

avec

$$a = \min \left(g\left(\underline{u}_{1,r}\right), g\left(\underline{u}_{1,r+1}\right), ..., g\left(\underline{u}_{1,s}\right), g\left(\overline{u}_{1,r}\right), g\left(\overline{u}_{1,r+1}\right), ..., g\left(\overline{u}_{1,s}\right) \right) \tag{3.34}$$

et

$$b = \max \left(g\left(\underline{u}_{1,r}\right), g\left(\underline{u}_{1,r+1}\right), ..., g\left(\underline{u}_{1,s}\right), g\left(\overline{u}_{1,r}\right), g\left(\overline{u}_{1,r+1}\right), ..., g\left(\overline{u}_{1,s}\right) \right). \tag{3.35}$$

Intéressons nous à présent au calcul de $[v]$,

$$[v] = [x_1] \cap \left[\bigcup_{k=r}^{s} \{[h_k]^* ([u_2]_k)\} \right] \tag{3.36}$$

où $[h_k]^* (.)$ correspond à la fonction d'inclusion minimale pour g_k^{-1}. Comme d'une part, pour $k \in \mathbb{Z}$,

$$[u_2]_k = [x_2] \cap \mathbb{F} = [x_2], \tag{3.37}$$

et d'autre part les fonctions réciproques g_k^{-1} sont monotones sur $[x_2]$, nous en déduisons

$$[v] = [x_1] \cap [c, d] \tag{3.38}$$

où

$$c = \min \left(g_r^{-1}\left(\underline{x}_2\right), g_{r+1}^{-1}\left(\underline{x}_2\right), ..., g_s^{-1}\left(\underline{x}_2\right), g_r^{-1}\left(\overline{x}_2\right), g_{r+1}^{-1}\left(\overline{x}_2\right), ..., g_s^{-1}\left(\overline{x}_2\right) \right) \tag{3.39}$$

et

$$d = \max \left(g_r^{-1}\left(\underline{x}_2\right), g_{r+1}^{-1}\left(\underline{x}_2\right), ..., g_s^{-1}\left(\underline{x}_2\right), g_r^{-1}\left(\overline{x}_2\right), g_{r+1}^{-1}\left(\overline{x}_2\right), ..., g_s^{-1}\left(\overline{x}_2\right) \right). \tag{3.40}$$

Dans un cadre général, la figure 3.3 illustre la contraction optimale de cinq pavés $[\mathbf{x}]_1$, $[\mathbf{x}]_2$, $[\mathbf{x}]_3$, $[\mathbf{x}]_4$ et $[\mathbf{x}]_5$ par rapport à la contrainte $x_2 - g(x_1) = 0$. Pour chacun de ces pavés, les zones grises correspondent à des domaines inconsistants[2] par rapport à notre contrainte, les zones claires sont les pavés obtenus après contraction. Les domaines de monotonies $\mathbb{E}_k$ ($k \in \mathbb{Z}$) de g ont été délimités en pointillés.

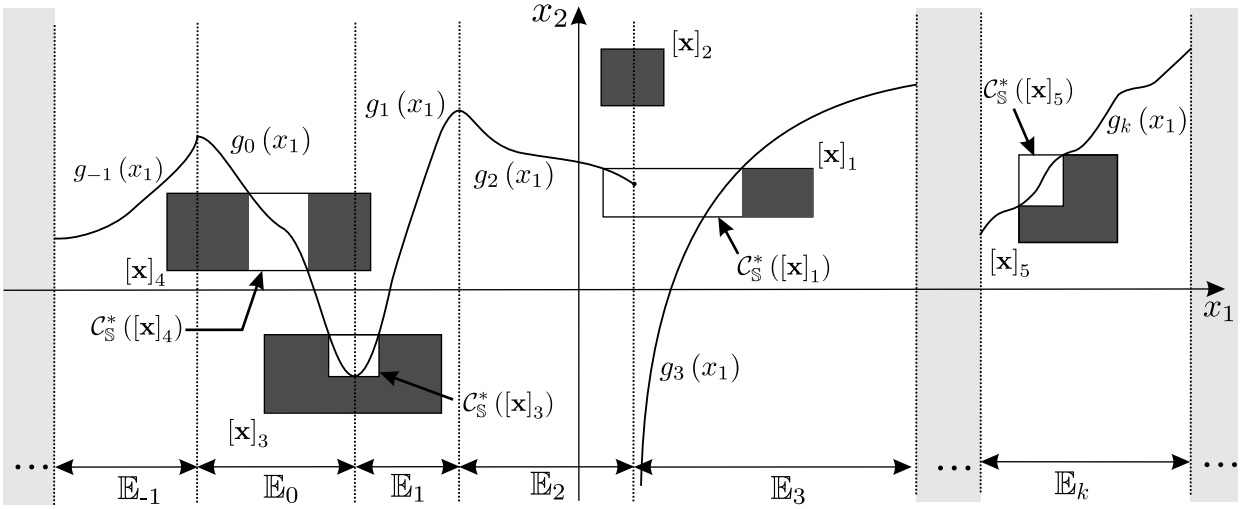

Figure 3.3 – Contraction optimale de pavés par rapport à la contrainte $x_2 - g(x_1) = 0$.

Dans le cas des contraintes ternaires (3.25), la construction du contracteur optimal, par une approche globale, n'est pas simple. Ceci s'explique car les contraintes concernées établissent une dépendance entre, non plus deux, mais trois variables. Pour cela, nous préconisons une approche au "cas par cas". Dans les exemples de contraintes ternaires que nous verrons, la méthode utilisée nécessite une étude de la projection des contraintes sur les variables x_1 et x_2. Nous définissons un niveau $C \in \mathbb{R}$, par les couples (x_1, x_2) tels que : $x_1 \diamond x_2 = C$, ici x_3 est fixée à une constante C. Ainsi, nous réalisons dans le plan (x_1, x_2), une surface constituée d'un ensemble de niveaux C.

Afin de déterminer le contracteur optimal, nous avons recours à la décomposition du plan (x_1, x_2), en plusieurs zones dans lesquelles les différents niveaux de la contrainte conservent des propriétés intéressantes (continuité, monotonie, *etc.*). Ici, nous parlons de monotonie ou de continuité suivant chaque variable prise séparément dans un compact de $\mathbb{R}^n$, ce qui signifie que pour chaque variable considérée les autres variables sont fixées à des constantes. De façon similaire à la méthode employée dans le cas des contraintes binaires, un pavé de $\mathbb{R}^3$ est partitionné sur les différentes zones de monotonie. Il devient alors possible de déterminer, pour les pavés issus de ce partitionnement, les plus grands pavés localement consistants. Le contracteur optimal résulte alors du plus petit pavé qui contient les pavés ainsi obtenus

[2] Un compact $\mathbb{D}$ de $\mathbb{R}^n$ est inconsistant par rapport à $\mathbb{S}$ si : $\mathbb{D} \subset \mathbb{S}^c$ ($\forall \mathbf{x} \in \mathbb{D}$ alors $\mathbf{x} \notin \mathbb{S}$).

sur chaque zone de monotonie. Nous verrons les détails de cette approche à travers les exemples 3.12 et 3.13.

3.2.1.1.b Exemples d'applications Dans les exemples suivants, nous avons choisi d'établir les contracteurs optimaux pour quelques cas de contraintes binaires et ternaires comme : puissance 2, cosinus, produit et max. Pour trouver les contracteurs optimaux associés à ces contraintes primitives, nous suivrons le principe présenté dans la section précédente, nous en utiliserons aussi les notations. Bien entendu, les exemples traités ici, nous inspirerons pour réaliser des contracteurs pour d'autres contraintes primitives.

Exemple 3.9. *Considérons l'ensemble,*

$$\mathbb{S}_1 = \left\{ (x_1, x_2) \in \mathbb{R}^{\times 2} \middle| \, x_2 = x_1^2 \right\} \tag{3.41}$$

Dans cet exemple, nous mettons en évidence deux domaines de monotonie lorsque $x_1 \in \mathbb{E}_0 =]-\infty, 0]$ et $x_1 \in \mathbb{E}_1 = [0, +\infty[$. Sur chacun de ces domaines ($k \in \{0, 1\}$), la contrainte devient

$$x_1 - g_k\left(x_2\right) = 0, \tag{3.42}$$

avec

$$\begin{array}{rccc} g_k: & \mathbb{E}_k & \longrightarrow & \mathbb{R}^+ \\ & x & \longmapsto & x^2. \end{array} \tag{3.43}$$

Pour déterminer les fonctions réciproques de g_k, nous partons du fait que nous savons calculer $g_1^{-1}(x)$ qui n'est autre que $\sqrt{x}$ pour $x \in \mathbb{R}^+$. En posant $g_0(x) = g_1(-x)$, il vient de façon générale,

$$\begin{array}{rccc} g_k^{-1}: & \mathbb{R}^+ & \longrightarrow & \mathbb{E}_k \\ & x & \longmapsto & (-1)^{k+1}\sqrt{x}. \end{array} \tag{3.44}$$

Grâce aux fonctions ainsi établies et aux expressions (3.30) et (3.36), nous concevons un algorithme nommé `CSQR` *qui réalise, pour les pavés $[\mathbf{x}]$ de $\mathbb{R}^2$, le contracteur $\mathcal{C}^*_{\mathbb{S}_1}([\mathbf{x}])$. Une version de cet algorithme est donnée ci-après. Les entrées $[x_1]$ et $[x_1]$ sont considérées non vides. Les notations* `SQR`$(.)$ *et* `SQRT`$(.)$ *qui sont utilisées représentent respectivement les fonctions d'inclusion pour $f_1(x) = x^2$ et $f_2(x) = \sqrt{x}$.*

Algorithme : `CSQR(Entrées-Sorties :` $[x_1]$, $[x_2]$`)`	
1	$[x_d] \leftarrow [x_1] \cap [0, +\infty[$;
2	$[x_g] \leftarrow [x_1] \cap]-\infty, 0]$;
3	$[x_2] \leftarrow [x_2] \cap$ `SQR` $([x_1])$;
4	$[x_g] \leftarrow [x_g] \cap (-$`SQRT` $([x_2]))$;
5	$[x_d] \leftarrow [x_d] \cap$ `SQRT` $([x_2])$;
6	$[x_1] \leftarrow [[x_g] \cup [x_d]]$;
7	`Fin.`

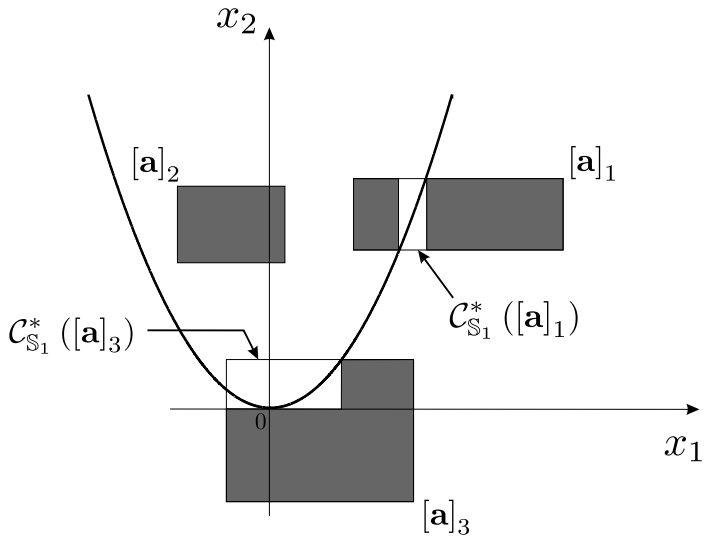

Figure 3.4 – Réduction optimale de pavés par rapport à la contrainte $x_2 = x_1^2$.

La figure 3.4 illustre le processus de réduction de différents pavés par rapport à la contrainte $x_2 = x_1^2$. Les domaines initiaux sont réduit après contraction aux pavés en blanc.

Exemple 3.10. *Soit un ensemble défini par,*

$$\mathbb{S}_2 = \left\{ (x_1, x_2) \in \mathbb{R}^{\times 2} \middle| \, x_2 = \cos(x_1) \right\}. \tag{3.45}$$

Dans ce cas, il existe une infinité de domaines de monotonie pour $x_1 \in \mathbb{E}_k = [k\pi, (k+1)\pi]$, avec $k \in \{..., -2, -1, 0, 1, 2, ...\}$. Notons g_0 la fonction définie de $\mathbb{E}_0 = [0, \pi]$ dans $[-1, 1]$ telle que $g_0(x) = \cos(x)$. Cette fonction représente cosinus restreint à $[0, \pi]$. La propriété de périodicité permet de reconstituer la fonction cosinus sur $\mathbb{R}$, grâce uniquement au tronçon g_0. Étant donné l'existence d'une fonction réciproque pour g_0, l'objectif de notre démarche est de déterminer des fonctions réciproques pour cosinus dans chaque domaine $\mathbb{E}_k$. Notons p un entier relatif, nous distinguons les cas de figures suivants,

– *Si k est pair ($k = 2p$), nous avons*

$$\begin{array}{rccl} g_{2p}: & \mathbb{E}_{2p} & \longrightarrow & [-1, 1] \\ & x & \longmapsto & g_0(x - 2p\pi) \end{array} \tag{3.46}$$

dont la fonction réciproque est :

$$\begin{array}{rccl} g_{2p}^{-1}: & [-1, 1] & \longrightarrow & \mathbb{E}_{2p} \\ & x & \longmapsto & 2p\pi + \arccos(x) \end{array} \tag{3.47}$$

car $g_0^{-1}(x) = \arccos(x)$.

– *Si k est impair ($k = 2p+1$), il vient*

$$\begin{array}{rccl} g_{2p+1}: & \mathbb{E}_{2p+1} & \longrightarrow & [-1,1] \\ & x & \longmapsto & -g_0\left(x-(2p+1)\,\pi\right) \end{array} \tag{3.48}$$

qui admet comme fonction réciproque,

$$\begin{array}{rccl} g_{2p+1}^{-1}: & [-1,1] & \longrightarrow & \mathbb{E}_{2p+1} \\ & x & \longmapsto & 2\,(p+1)\,\pi - \arccos(x) \end{array} \tag{3.49}$$

Après avoir déterminer les fonctions g_k et g_k^{-1}, nous obtenons en utilisant les expressions (3.30) et (3.36), le contracteur optimal pour $\mathbb{S}_2$. L'algorithme appelé `CCOS` *réalise ce contracteur pour tout pavé $[\mathbf{x}] \subset \mathbb{R}^2$. Les détails de cet algorithme sont présentés ci-dessous. En entrée, les intervalles $[x_1]$ et $[x_2]$ sont supposés non vides. La notation* `E`(x) *désigne la partie entière de x.*

Algorithme : `CCOS(Entrées-Sorties :` $[x_1]$, $[x_2]$`)`

```
1   [x2] ← [x2] ∩ [−1, 1];
2   r ← E(x1/π); s ← E(x̄1/π);
3   [v] ← ∅;  [w] ← ∅;
4   k ← r;
5   Tant que (k ≤ s) faire :
6         [u] ← [x1] ∩ [k.π, (k+1).π];
7         Si (k pair) alors:
8               [w] ← [[w] ∪ [cos(ū), cos(u)]];
9               [v] ← [[v] ∪ [k.π + arccos(x̄2), k.π + arccos(x2)]];
10        Sinon :
11              [w] ← [[w] ∪ [cos(u), cos(ū)]];
12              [v] ← [[v] ∪ [(k+1).π − arccos(x̄2), (k+1).π − arccos(x2)]];
13        Fin sinon (10); fin si (7);
14        k ← k+1;
15  Fin tant que (5);
16  [x1] ← [x1] ∩ [v];
17  [x2] ← [x2] ∩ [w];
18  Fin.
```

La figure 3.5 montre les effets de la procédure `CCOS` sur les pavés $[\mathbf{z}]_i$ ($i = 1,2,3,4,5$). Comme précédemment, les zones grises correspondent à des domaines inconsistants pour $\mathbb{S}_2$. Les pavés blancs résultent de la contraction par rapport à $x_2 = \cos(x_1)$, nous relevons par exemple,

$$\mathcal{C}^*_{\mathbb{S}_2}\left([\mathbf{z}]_1\right) = [\mathbf{z}]_1 \text{ et } \mathcal{C}^*_{\mathbb{S}_2}\left([\mathbf{z}]_2\right) = \emptyset.$$

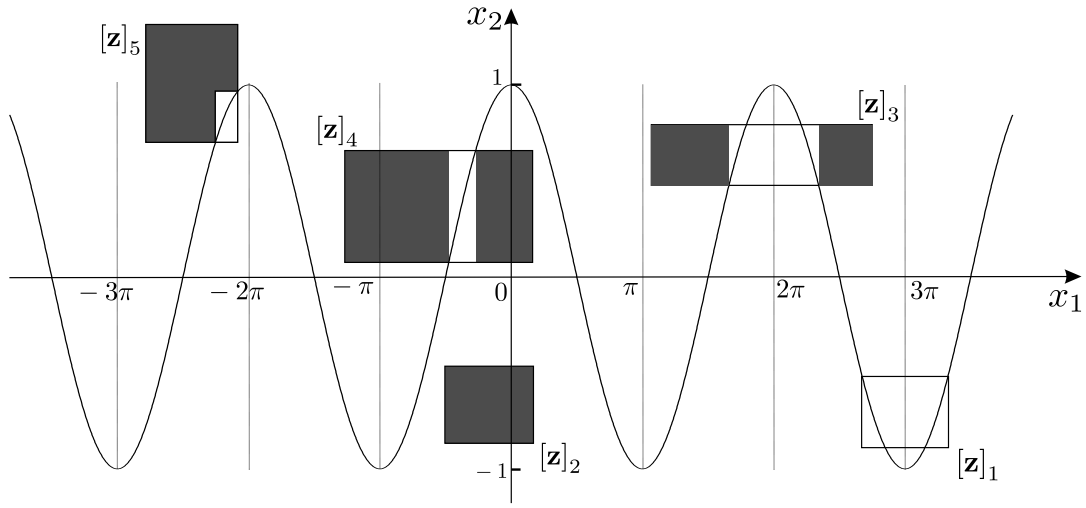

Figure 3.5 – Contracteur optimal pour la contrainte cosinus.

Remarque 3.11. *Pour les contraintes décrites par des fonctions périodiques comme* $\sin(x)$ *et* $\tan(x)$*, nous utiliserons le même principe que celui présenté dans le cas de cosinus. Ainsi, pour réaliser un contracteur optimal pour sinus, il suffit d'appliquer l'algorithme* `CCOS` *en tenant compte de la formule trigonométrique* $\sin(x) = \cos(x - \frac{\pi}{2})$*. Concernant la contrainte* $y = \tan(x)$*, une étude globale de la fonction permet de déduire,* k *étant un entier relatif,* $\mathbb{E}_k = \left[(2k-1)\,\frac{\pi}{2}, (2k+1)\,\frac{\pi}{2}\right]$ *et*

$$g_k(x) = g_0(x - k\pi) \tag{3.50}$$

avec $g_0(x) = \tan(x)$*. Les fonctions réciproques associées à* g_k *sont données par,*

$$g_k^{-1}(x) = k\pi + \arctan(x). \tag{3.51}$$

Avec tous ces éléments, il devient relativement facile d'établir le contracteur optimal pour $\tan(x)$*.*

Exemple 3.12. *A présent, considérons une contrainte ternaire définie par l'ensemble*

$$\mathbb{S}_3 = \left\{ (x_1, x_2, x_3) \in \mathbb{R}^3 \middle| \; x_3 = x_1 \cdot x_2 \right\}. \tag{3.52}$$

Comme nous l'avions annoncé dans le principe (voir section 3.2.1.1a), la réalisation du contracteur optimal pour $\mathbb{S}_3$ *nécessite une projection de la contrainte dans le plan* $(x_1, x_2) \in \mathbb{R}^2$*. En fixant* x_3 *à des constantes réels, nous obtenons dans le plan* (x_1, x_2) *un ensemble de niveaux* $C \in \mathbb{R}$ *tel que* $x_3 = C = x_1 \cdot x_2$*. Une brève analyse de la contrainte met en évidence l'existence de symétries, quelque soit* $(x_1, x_2, x_3) \in \mathbb{S}_3$ *alors* $(-x_1, -x_2, x_3) \in \mathbb{S}_3$*. De ce fait, nous distinguons dans le plan* (x_1, x_2)*, quatre zones ou cadrans pour lesquels :*

- *Cas* $x_3 \geq 0$
  cadran 1 : $x_1 \geq 0$ *et* $x_2 \geq 0$*;*
  cadran 2 : $x_1 < 0$ *et* $x_2 < 0$*;*
- *Cas* $x_3 < 0$
  cadran 3 : $x_1 < 0$ *et* $x_2 > 0$*;*
  cadran 4 : $x_1 > 0$ *et* $x_2 < 0$*.*

Une première procédure appelée CPROD *réalise une contraction optimale pour un pavé* $[\mathbf{x}]$ *inclus dans la zone du cadran 1. Désignons par* $\mathcal{C}^*_{\mathbb{S}_3}([\mathbf{x}]) = [u] \times [v] \times [w]$*, le pavé issu de cette contraction. Posons* $x_3 = f(x_1, x_2) = x_1 \cdot x_2$ *et définissons deux fonctions* h_1 *et* h_2 *telles que :* $x_1 = h_1(x_2, x_3)$ *et* $x_2 = h_2(x_1, x_3)$*. Ici,* f *est croissante suivant* x_1 *et* x_2*. Ainsi quelque soit* $x_2 = C_2 \geq 0$*, la fonction* $f(x_1, C_2)$ *est croissante. De même, quelque soit* $x_1 = C_1 \geq 0$*,* $f(C_1, x_2)$ *est croissante. Il vient donc,*

$$\begin{aligned} [w] &= [x_3] \cap [f]^*([x_1], [x_2]) \\ &= [x_3] \cap [f(\underline{x}_1, \underline{x}_2), f(\overline{x}_1, \overline{x}_2)] \\ &= [x_3] \cap [\underline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2]. \end{aligned} \tag{3.53}$$

La fonction h_2 *est croissante suivant* x_3 *et décroissante suivant* x_1*. Nous traduisons cela par, quelque soit* $x_1 = C_1 > 0$*,* $h_2(C_1, x_3)$ *est croissante et, quelque soit* $x_3 = C_3 \geq 0$*,* $h_2(x_1, C_3)$ *est décroissante. De ce fait, nous avons :*

$$\begin{aligned} [v] &= [x_2] \cap [h_2]^*([x_1], [x_3]) \\ &= [x_2] \cap [f(\overline{x}_1, \underline{x}_3), f(\underline{x}_1, \overline{x}_3)], \end{aligned} \tag{3.54}$$

qui devient après calcul,

$$[v] = [x_2] \cap \left[\frac{\underline{x}_3}{\overline{x}_1}, \frac{\overline{x}_3}{\underline{x}_1}\right]. \tag{3.55}$$

Pour le calcul de $[u]$*, nous remarquons que* h_1 *est croissante suivant* x_3 *et décroissante suivant* x_2*, ce qui permet d'établir*

$$[u] = [x_1] \cap [h_1]^*([x_2], [x_3]), \tag{3.56}$$

il en résulte ensuite,

$$[u] = [x_1] \cap \left[\frac{\underline{x}_3}{\overline{x}_2}, \frac{\overline{x}_3}{\underline{x}_2}\right]. \tag{3.57}$$

Pour définir les contracteurs dans les autres cadrans, nous exploitons les symétries engendrées par la contrainte $x_3 = x_1 \cdot x_2$*. Les pavés provenant des cadrans 2, 3 et 4 sont amenés au cadran 1 par des translations. Ainsi les pavés contractés par* CPROD *sont replacés dans leurs cadrans d'origines par translations inverses. Pour finir, nous déterminons le plus petit pavé qui contient les domaines contractés sur les quatre cadrans. L'algorithme* CMULT *permet ainsi d'obtenir le contracteur optimal pour* $\mathbb{S}_3$*. Les détails des algorithmes* CPROD *et* CMULT *sont décrits comme suit.*

Algorithme : CPROD(Entrées-Sorties : $[x_1]$, $[x_2]$, $[x_3]$)	
1	Si $([x_1] \times [x_2] \times [x_3] \neq \emptyset)$:
2	$[u] \leftarrow [x_1]$; $[v] \leftarrow [x_2]$; $[w] \leftarrow [x_3]$;
3	$[x_3] \leftarrow [x_3] \cap [\underline{u} \cdot \underline{v}, \overline{u} \cdot \overline{v}]$;
4	Si $(\overline{u} \cdot \overline{v} > 0)$:
5	Si $(\underline{v} \cdot \overline{v} > 0)$: $[x_1] \leftarrow [x_1] \cap \left[\frac{w}{\overline{v}}, \frac{\overline{w}}{\underline{v}}\right]$;
6	Sinon : $[x_1] \leftarrow [x_1] \cap \left[\frac{w}{\overline{v}}, +\infty\right[$;
7	Si $(\underline{u} \cdot \overline{u} > 0)$: $[x_2] \leftarrow [x_2] \cap \left[\frac{w}{\overline{u}}, \frac{\overline{w}}{\underline{u}}\right]$;
8	Sinon : $[x_2] \leftarrow [x_2] \cap \left[\frac{w}{\overline{u}}, +\infty\right[$;
9	Fin si(4)
10	Fin si(1)
11	Si $([x_1] \times [x_2] \times [x_3] = \emptyset)$:
12	$[x_1] \leftarrow \emptyset$; $[x_2] \leftarrow \emptyset$; $[x_3] \leftarrow \emptyset$;
13	Fin.

Algorithme : CMULT(Entrées-Sorties : $[x_1]$, $[x_2]$, $[x_3]$)	
1	$[x_g] \leftarrow [x_1] \cap \mathbb{R}^-$; $[x_d] \leftarrow [x_1] \cap \mathbb{R}^+$;
2	$[y_g] \leftarrow [x_2] \cap \mathbb{R}^-$; $[y_d] \leftarrow [x_2] \cap \mathbb{R}^+$;
3	$[z_g] \leftarrow [x_3] \cap \mathbb{R}^-$; $[z_d] \leftarrow [x_3] \cap \mathbb{R}^+$;
4	$[u_1] \leftarrow [x_d]$; $[u_2] \leftarrow [y_d]$; $[u_3] \leftarrow [z_d]$;
5	CPROD($[u_1], [u_2], [u_3]$);
6	$[v_1] \leftarrow -[x_g]$; $[v_2] \leftarrow -[x_d]$; $[v_3] \leftarrow [z_d]$;
7	CPROD($[v_1], [v_2], [v_3]$);
8	$[v_1] \leftarrow -[v_1]$; $[v_2] \leftarrow -[v_2]$;
9	$[w_1] \leftarrow -[x_g]$; $[w_2] \leftarrow [y_d]$; $[w_3] \leftarrow -[z_g]$;
10	CPROD($[w_1], [w_2], [w_3]$);
11	$[w_1] \leftarrow -[w_1]$; $[w_3] \leftarrow -[w_3]$;
12	$[t_1] \leftarrow [x_d]$; $[t_2] \leftarrow -[y_g]$; $[t_3] \leftarrow -[z_g]$;
13	CPROD($[t_1], [t_2], [t_3]$);
14	$[t_2] \leftarrow -[y_g]$; $[t_3] \leftarrow -[z_g]$;
15	$[x_1] \leftarrow [x_1] \cap [[u_1] \cup [v_1] \cup [w_1] \cup [t_1]]$;
16	$[x_2] \leftarrow [x_2] \cap [[u_2] \cup [v_2] \cup [w_2] \cup [t_2]]$;
17	$[x_3] \leftarrow [x_3] \cap [[u_3] \cup [v_3] \cup [w_3] \cup [t_3]]$;
18	Fin.

La figure 3.6 illustre la contraction optimale de pavés dans le cadran 1. Sur cette figure, les pavés représentent un domaine où $(x_1, x_2, x_3) \in ([\mathbf{b}]_1 \cup [\mathbf{b}]_2 \cup [\mathbf{b}]_3) \times [x_3]$. Pour ces pavés, nous pouvons relever les indications suivantes, (i) : $\mathcal{C}^*_{\mathbb{S}_3}([\mathbf{b}]_1 \times [x_3]) = \emptyset$, (ii) : $\mathcal{C}^*_{\mathbb{S}_3}([\mathbf{b}]_2 \times [x_3]) = [\mathbf{c}]_2 \times [x_3]$, (iii) : $\mathcal{C}^*_{\mathbb{S}_3}([\mathbf{b}]_3 \times [x_3]) = [\mathbf{c}]_3 \times [x_3^*, \overline{x}_3]$.

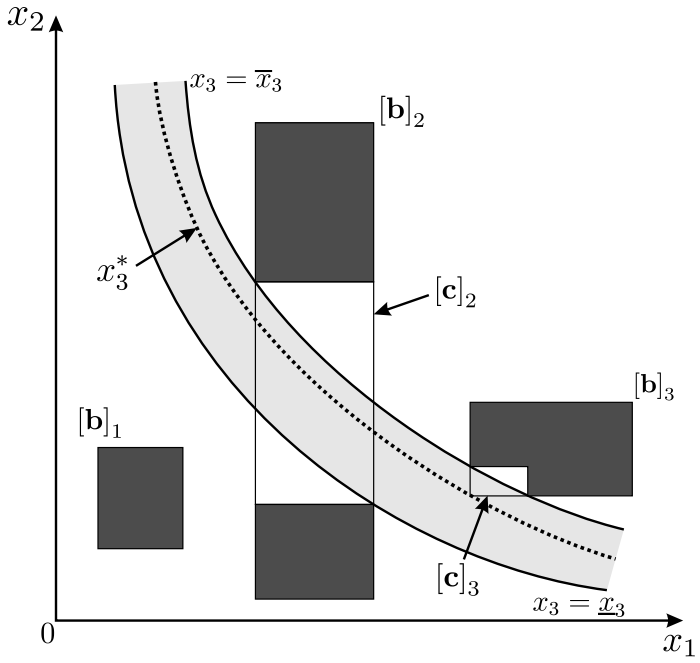

Figure 3.6 – Réduction de pavés par rapport à la contrainte primitive $x_3 = x_1 \cdot x_2$.

Exemple 3.13. *On note $\mathbb{S}_4$, l'ensemble des triplets $(x_1, x_2, x_3) \in \mathbb{R}^3$ qui satisfont la contrainte :*

$$x_3 = \max(x_1, x_2). \tag{3.58}$$

Un contracteur pour $\mathbb{S}_4$ sera utile pour résoudre certains problèmes dans les chapitres suivants. Prenons un pavé $[\mathbf{x}]$ de $\mathbb{R}^3$, posons ensuite $\mathcal{C}^*_{\mathbb{S}_4}([\mathbf{x}]) = [\mathbf{v}]$. Une des composantes du pavé contracté est calculée de la façon suivante,

$$[v_3] = [x_3] \cap \max([x_1], [x_2]), \tag{3.59}$$

où $\max([x], [y]) \triangleq \{\max(x, y) | \, x \in [x]$ et $y \in [y]\}$. La fonction $\max$ est croissante suivant chacune des deux variables x_1 et x_2, d'où

$$[v_3] = [x_3] \cap [\max(\underline{x}_1, \underline{x}_2), \max(\overline{x}_1, \overline{x}_2)]. \tag{3.60}$$

Pour déterminer les deux autres composantes $[v_1]$ et $[v_2]$, les considérations suivantes sont prises en compte. Quelque soit $x_3 \in \mathbb{R}$, nous avons deux possibilités pour x_1 et x_2, soit $x_1 \leq x_3$ et $x_2 = x_3$, soit

$x_1 = x_3$ *et* $x_2 \leq x_3$ *car* $\max$ *est commutatif. Ainsi, si* $x_3 \in [x_3]$*, nous avons*

$$x_1 \leq \overline{x}_3 \text{ et } x_2 \in [x_3]$$

ou

$$x_1 \in [x_3] \text{ et } x_2 \leq \overline{x}_3.$$

Ce qui permet d'établir d'une part,

$$[v_1] = [\{[x_1] \cap]-\infty, \overline{x}_3]\} \cup \{[x_1] \cap [x_3]\}] \tag{3.61}$$

et d'autre part,

$$[v_2] = [\{[x_2] \cap [x_3]\} \cup \{[x_2] \cap]-\infty, \overline{x}_3]\}]\,. \tag{3.62}$$

La figure 3.7 illustre, par rapport à la contrainte (3.58), la contraction optimale de différents pavés de $\mathbb{R}^3$. Rappelons que les zones grises correspondent à des domaines inconsistants. L'ensemble des niveaux $x_3 \in [x_3]$ représentent la projection de la contrainte (3.58) dans le plan (x_1, x_2). En outre, nous présentons les configurations pour lesquelles :

- (i) : $\mathcal{C}^*_{\mathbb{S}_4}([\mathbf{d}]_1 \times [x_3]) = \emptyset$,
- (ii) : $\mathcal{C}^*_{\mathbb{S}_4}([\mathbf{d}]_2 \times [x_3]) = [\mathbf{e}]_2 \times [x_3]$,
- (iii) : $\mathcal{C}^*_{\mathbb{S}_4}([\mathbf{d}]_3 \times [x_3]) = [\mathbf{e}]_3 \times [a, \overline{x}_3]$,
- (iv) : $\mathcal{C}^*_{\mathbb{S}_4}([\mathbf{d}]_4 \times [x_3]) = [\mathbf{d}]_4 \times [\underline{x}_3, b]$.

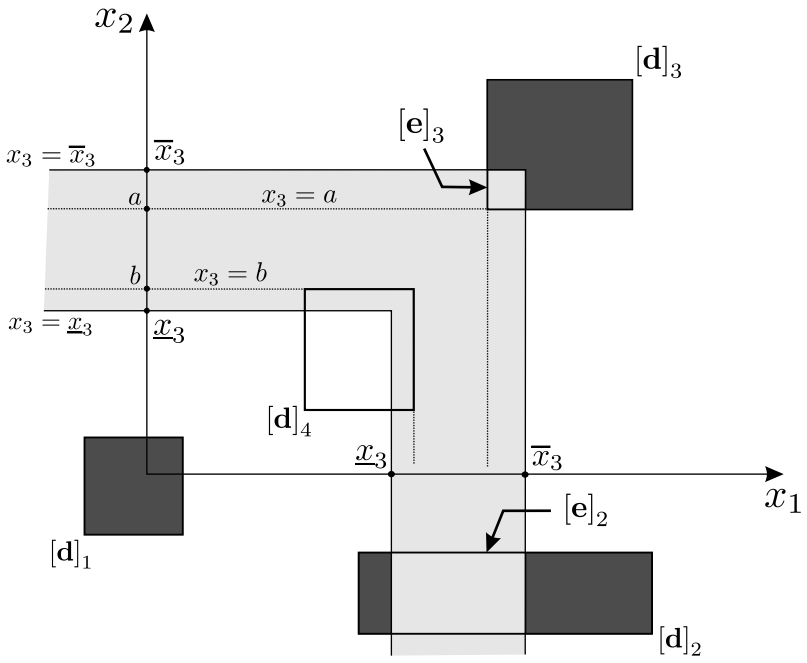

Figure 3.7 – Opérateur de contraction pour la contrainte (3.58).

Remarque 3.14. *Grâce au contracteur pour* $\max$*, nous déduisons naturellement un contracteur optimal pour l'opérateur* $\min$*. Pour cela, nous utilisons l'expression :* $\min(x,y) = -\max(-x,-y)$*. Ainsi, trouver le contracteur optimal pour* $z = \min(x,y)$ *revient à déterminer le contracteur optimal pour la contrainte* $t = \max(s,r)$*, avec* $t = -z$*,* $s = -x$ *et* $r = -y$*. De même, nous réalisons un contracteur optimal pour* $y = |x|$ *en remarquant que :* $|x| = \max(-x,x)$*.*

3.2.1.2 Propagation de contraintes

Soit un sous-ensemble de $\mathbb{R}^n$ défini par

$$\mathbb{S} = \{\mathbf{x} \in \mathbb{R}^n | \, \mathbf{f}(\mathbf{x}) = \mathbf{0}\}, \tag{3.63}$$

où $\mathbf{f}$ est une fonction vectorielle de $\mathbb{R}^n$ dans $\mathbb{R}^m$. La contrainte $f_i(\mathbf{x}) = 0$ ($i \in \{1,2,...m\}$) peut être décomposée en un système de contraintes primitives grâce à l'utilisation de variables intermédiaires. Un contracteur pour $\mathbb{S}$ devient un regroupement de *contracteurs élémentaires*[3].

Exemple 3.15. *Nous désirons réaliser un contracteur pour l'ensemble* $\mathbb{F}$ *des variables* (x_1, x_2, x_3) *qui satisfont les contraintes :* $x_1^3 + x_2 \in [0,1]$ *et* $x_1 - x_2 \in [0.3, 0.5]$*. La décomposition en contraintes primitives est donnée par*

$$\begin{cases} a_1 = x_1^3 \\ a_2 = a_1 + x_2 \\ a_3 = -x_2 \\ a_4 = x_1 + a_3 \end{cases} \tag{3.64}$$

où a_1*,* a_2*,* a_3*,* a_4 *sont des variables intermédiaires. Nous désignons respectivement par* `CPOWCUBE`*,* `CADD`*,* `CMINUS` *les contracteurs optimaux pour les 4 contraintes du système (3.64). L'algorithme* `CONTRACT` *détaillé ci-dessous réalise sur un pavé* $[\mathbf{x}]$ *un contracteur par rapport à* $\mathbb{F}$*. Dans* `CONTRACT`*, le terme de "significative" dépend de la contrainte traitée et du résultat voulu ou attendu par l'utilisateur.*

Algorithme : `CONTRACT(Entrée-Sortie :` $[\mathbf{x}]$`)`	
1	$[\mathbf{a}] = \mathbb{R}^{\times 4}$; $[a_2] = [0,1]$; $[a_4] = [0.3, 0.5]$;
2	`Tant que contraction de` $[\mathbf{x}]$ `est significative faire`
3	`CPOWCUBE`$([a_1],[x_1])$;
4	`CADD`$([a_2],[x_2],[a_1])$;
5	`CMINUS`$([a_3],[x_2])$;
6	`CADD`$([a_4],[a_3],[x_1])$;
7	`Fin tant que`
8	`Fin.`

[3] Contracteurs optimaux associés aux contraintes primitives.

Soit $(x_1, x_2) \in [\mathbf{x}] = [-2, 2] \times [-2, 2]$, l'algorithme `CONTRACT` converge vers le pavé :

$$\mathcal{C}_{\mathbb{F}}([\mathbf{x}]) = [-1.02, 1.361] \times [-1.52, 1.061]. \tag{3.65}$$

La figure 3.8 montre les étapes intermédiaires de convergence vers le plus grand pavé localement consistant. L'algorithme de propagation atteint alors une situation de blocage. Nous pouvons remarquer que nous sommes encore loin du plus grand pavé globalement consistant.

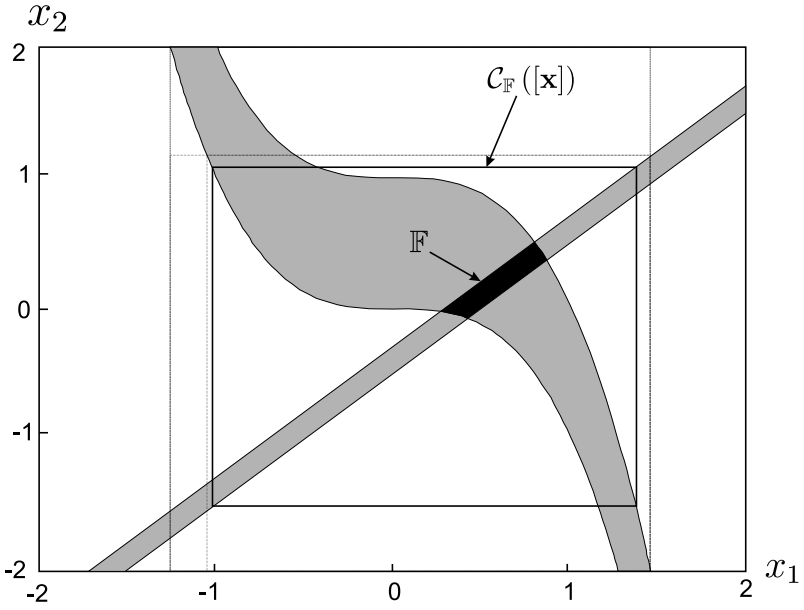

Figure 3.8 – Convergence de la propagation de contraintes vers le grand pavé localement consistant $\mathcal{C}_{\mathbb{F}}([\mathbf{x}])$.

Remarque 3.16. *Dans certains cas, le plus grand pavé localement consistant se confond avec le plus grand pavé globalement consistant, la technique de propagation de contraintes devient alors une méthode de consistance forte. Par exemple, lorsque nous considérons les contraintes données par :* $x_1^2 + x_2 = 0$ *et* $x_1 - x_2 = 0$ *avec* $x_1 \in [-10, 10]$, $x_2 \in [-10, 10]$. *Les solutions évidentes aux deux contraintes sont les couples* $(-1, -1)$ *et* $(0, 0)$, *grâce à la propagation sur chacune des contraintes nous obtenons le pavé :* $[-1, 0] \times [-1, 0]$, *ce qui correspond au plus petit pavé enveloppant l'ensemble solution.*

Plusieurs solveurs permettent de tester différentes méthodes de réduction de domaines, parmi lesquels REALPAVER ([Granvilliers, 2002]) et INTERVALPEELER[4] (X. Baguenard et *al.*). Le deuxième solveur utilise la représentation des contraintes sous forme de DAG (Direct Acyclic Graph) associée à la propagation des contraintes et au calcul par intervalles.

[4]Librement disponible sur `http://www.istia.univ-angers.fr/~baguenar/IntervalPeeler.zip`

3.2.2 Méthodes de consistance forte

Ici, nous présenterons quelques moyens qui permettent d'améliorer la méthode de propagation de contraintes, ceci dans le but d'atteindre la consistance globale. Bien entendu, les méthodes exposées ne permettront pas à coup sûr de trouver le plus grand pavé globalement consistant. Ces méthodes ont pour vertu fondamentale la sortie des situations de blocage qui constituent, dans de multiples cas, le plus grand pavé localement consistant.

3.2.2.1 Contraintes redondantes

Il s'agit d'ajouter au CSP original des contraintes issues de manipulations algébriques avec des contraintes existantes, ceci a pour objectif d'améliorer l'efficacité de la propagation des contraintes en ajoutant de l'information de réduction supplémentaire pour les domaines. La génération de ce type de contraintes peut être traitée automatiquement et de façon aléatoire (voir [Benhamou and Granvilliers, 1997]), néanmoins il s'avère que le traitement automatique des expressions afin de faire apparaître le moins de variables est une tâche très complexe même pour les meilleurs logiciels de calcul formel comme MAPLE® ou MATHEMATICA©. Ainsi pour obtenir une contraction plus efficace, le traitement manuel des expressions en fonction de chaque problème reste le moyen le plus sûr.

Les figures 3.9 indiquent les étapes de la réduction d'un pavé au fur et à mesure des ajouts de contraintes redondantes. Sur la figure 3.9a, nous considérons l'intersection de 2 contraintes représentées par les ensembles $\mathbb{S}_1$ et $\mathbb{S}_2$, nous obtenons après la propagation des contraintes sur $[\mathbf{x}]_0$, le pavé $[\mathbf{x}]_1$ qui correspond au plus grand pavé localement consistant avec $\mathbb{S}_1 \cap \mathbb{S}_2$. La figure 3.9b montre l'ajout d'une contrainte redondante représenté par $\mathbb{S}_3$, ceci a pour but de débloquer la situation de consistance locale établie par les deux premières contraintes, nous obtenons après propagation des contraintes $\mathbb{S}_3$ et $\mathbb{S}_1$ une nouvelle situation de blocage sur le pavé $[\mathbf{x}]_3$. Une autre contrainte redondante $\mathbb{S}_4$ est définie, puis nous recommençons une procédure de contraction par rapport à $\mathbb{S}_4$ et $\mathbb{S}_3$. Il vient alors le pavé $[\mathbf{x}]_5$ qui est déjà très proche du plus grand pavé globalement consistant (voir figure 3.9c).

Reprenons l'exemple 3.15, en sommant les deux contraintes, nous en obtenons une nouvelle qui s'écrit de deux façons,

- (1) : $x_1^3 + x_1 \in [0.3, 1.5]$;
- (2) : $x_1 \cdot \left(x_1^2 + 1\right) \in [0.3, 1.5]$.

L'ajout de la contrainte redondante (1) n'apporte aucune amélioration à la contraction du pavé initial, tandis qu'en utilisant la contrainte (2), nous obtenons lors de la propagation des contraintes, le pavé :

$$\mathcal{C}_{\mathbb{F}}\left([\mathbf{x}]\right) = [0.1344, 1.11] \times [-0.3656, 0.80943]. \tag{3.66}$$

Nous pouvons alors noter une véritable amélioration par rapport au résultat de la contraction donnée par (3.65).

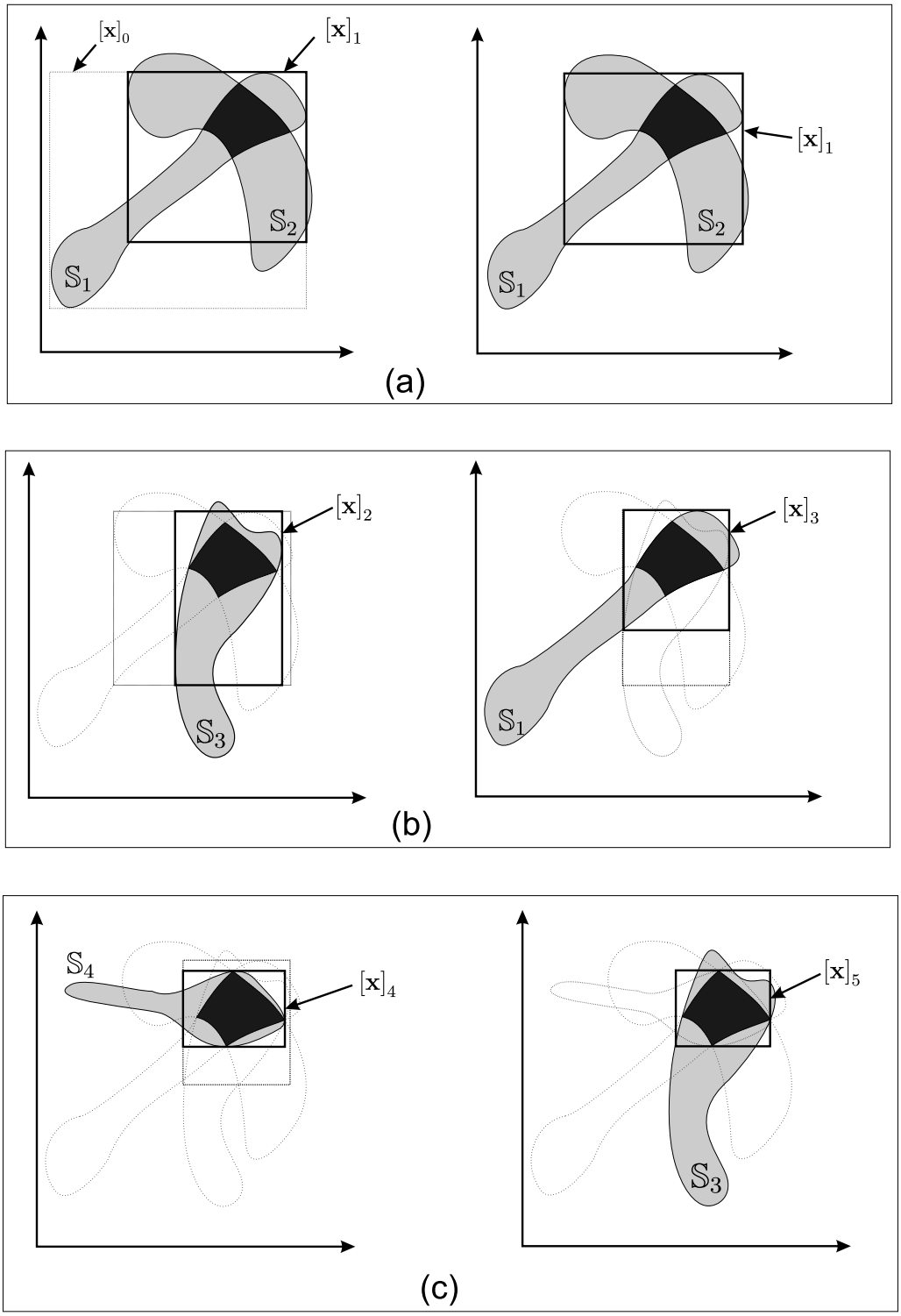

Figure 3.9 – Illustration du principe de la contraction grâce aux contraintes redondantes.

3.2.2.2 Epluchage

C'est un procédé qui consiste à prélever des fines tranches sur les bords d'un pavé, afin de démontrer par propagation de contraintes qu'elles sont localement et donc globalement inconsistantes par rapport aux contraintes. Comme précédemment nous avons représenté sur les figures 3.10a et 3.10b la stratégie employée. Nous pouvons donc observer sur la figure de gauche 3.10a, la première itération qui consiste à prélever le pavé $[\mathbf{x}]_2$ sur $[\mathbf{x}]_0$. La tranche $[\mathbf{x}]_2$ est ensuite éliminée (grâce à la propagation de contraintes) car elle est localement inconsistante. Le pavé restant $[\mathbf{x}]_1$ est réduit par propagation des contraintes (voir figure de droite 3.10a). L'ensemble de la procédure est reprise pour le pavé $[\mathbf{x}]_1$. Lorsque nous atteignons une situation où la tranche prélevée ne peut être réduite, il est préférable d'effectuer les prélèvement sur un autre coté du pavé (voir figures gauche, centre et droite 3.10b).

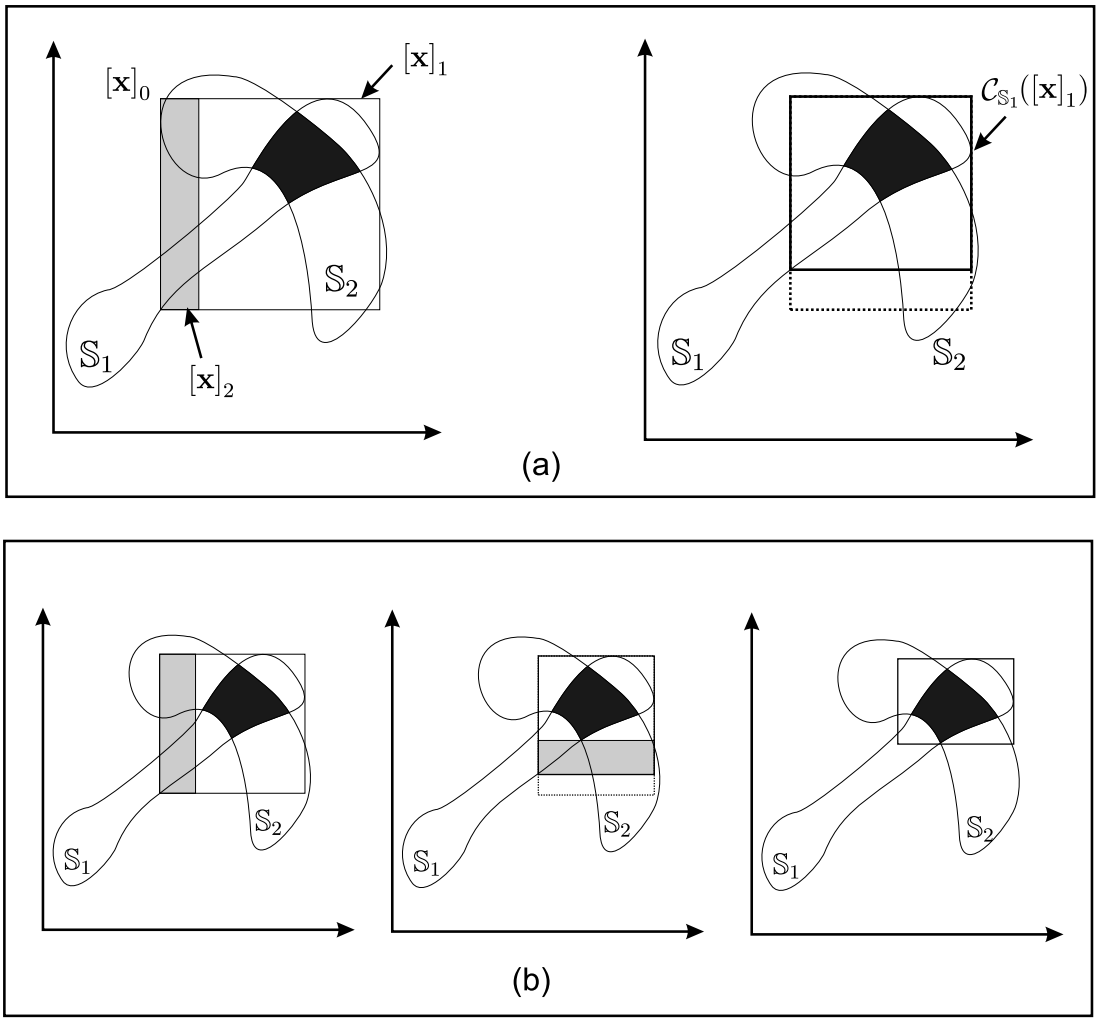

Figure 3.10 – Illustration de la réduction d'un pavé par épluchage.

L'algorithme SHAVING[5] que nous décrivons ci-dessous permet de traiter les opérations effectuées sur les figures 3.10a et 3.10b. Nous disposons au préalable d'un contracteur $\mathcal{C}_{\mathbb{S}}$ pour les contraintes considérées, d'un paramètre λ qui représente la plus petite largeur de la tranche prélevée et d'un entier n qui correspond à la dimension (le nombre de cotés) de $[\mathbf{x}]$.

Algorithme : SHAVING(Entrée - Sortie : $[\mathbf{x}]$)	
1	i $\leftarrow$ 1;
2	Tant que (i $\leq$ n) faire :
3	k $\leftarrow$ 1;
4	Tant que (k $>$ 0) faire :
5	$([\mathbf{x}]_1, [\mathbf{x}]_2) \leftarrow$ Part(i, $[\mathbf{x}]$, k.λ);
6	$[\mathbf{x}] \leftarrow [\mathcal{C}_{\mathbb{S}}([\mathbf{x}]_1) \cup \mathcal{C}_{\mathbb{S}}([\mathbf{x}]_2)]$;
7	Si la contraction de $[\mathbf{x}]$ est significative alors
8	k $\leftarrow$ k+1;
9	Sinon k $\leftarrow$ k-1;
10	Fin Tant que
11	i $\leftarrow$ i+1;
12	Fin Tant que
13	Fin.

La fonction Part$(i, [\mathbf{x}], \alpha)$ génère deux pavés. Si $\alpha < \overline{x}_i - \underline{x}_i$ nous avons

$$[\mathbf{x}]_1 = [x_1] \times ... \times [\underline{x}_i - \alpha,\ \overline{x}_i] \times ... \times [x_n], \tag{3.67}$$

$$[\mathbf{x}]_2 = [x_1] \times ... \times [\underline{x}_i,\ \underline{x}_i - \alpha] \times ... \times [x_n], \tag{3.68}$$

sinon $[\mathbf{x}]_1 = [\mathbf{x}]$ et $[\mathbf{x}]_2 = \emptyset$. L'algorithme SHAVING constitue une méthode de consistance globale. Comme nous pouvons le constater, la complexité de cet algorithme est polynomiale, néanmoins il peut être gourmand en temps de calcul surtout si la dimension du pavé $[\mathbf{x}]$ est élevée. Dans les faits, la principale difficulté de mise en oeuvre de l'algorithme d'épluchage SHAVING est le choix du paramètre λ. La largeur des tranches prélevées joue un grand rôle dans la rapidité d'exécution de l'algorithme. En effet, plus la tranche de pavé prélevée est fine (λ est choisi petit) meilleure en sera la réduction du pavé, par contre le temps de calcul deviendra plus long. D'un autre côté, choisir des tranches plus épaisses (grandes valeurs de λ) permet de gagner en temps de calcul, la technique perd en terme d'efficacité de contraction du pavé. Il faudrait donc trouver un compromis entre l'efficacité de la réduction et le temps de calcul. Le dilemme ainsi soulevé n'est pas simple à résoudre car cela dépend de chaque CSP.

[5] Qui signifie en anglais copeau ou epluchure

3.2.2.3 Linéarisation des contraintes

Le principe de cette méthode part du constat suivant, nous savons obtenir à coup sûr le plus grand pavé globalement consistant lorsque les contraintes sont des inégalités linéaires. En effet, l'utilisation de la méthode du simplexe permet sur les contraintes linéaires de définir les sommets du polytope constituant l'ensemble des solutions. Pour n variables, il faudrait résoudre $2n$ problèmes d'optimisation linéaires sous contraintes. Dans le cas de contraintes non linéaires, la linéarisation est effectuée en remplaçant les parties non linéaires par des variables intermédiaires. Nous déterminons ainsi pour ces variables le plus petit pavé renfermant les solutions aux contraintes linéaires. Les informations obtenues pour les variables intermédiaires sont communiquées aux autres variables grâce à la propagation de contraintes sur les parties non linéaires.

Afin de formaliser ce qui vient d'être expliqué, considérons les contraintes du type $\mathbf{f}(\mathbf{x}) \in [\mathbf{y}]$. Rappelons que $\mathbf{f}$ est une fonction définie de $\mathbb{R}^n$ dans $\mathbb{R}^m$et $[\mathbf{y}]$ un pavé de $\mathbb{R}^m$. Après linéarisation en utilisant des variables intermédiaires $\mathbf{z}$, il vient

$$\underline{\mathbf{y}} \leq \mathbf{A}\mathbf{z} \leq \bar{\mathbf{y}} \tag{3.69}$$

où $\mathbf{z} = \mathbf{g}(\mathbf{x})$ avec $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^p$ une fonction non linéaire et

$$\mathbf{A} = \begin{pmatrix} a_{11} & \dots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{np} \end{pmatrix}. \tag{3.70}$$

Grâce à la méthode d'optimisation du simplexe, nous calculons pour le vecteur $\mathbf{z}$ le plus petit pavé $[\mathbf{z}^*]$ contenant les solutions des inégalités données par (3.69). Pour $[\mathbf{z}^*] = \overset{p}{\underset{i=1}{\times}} [\underline{z}_i^*, \overline{z}_i^*]$, nous avons

$$\underline{z}_i^* = \min_{z_i \in \mathbb{R}} (z_i) \text{ et } \overline{z}_i^* = \max_{z_i \in \mathbb{R}} (z_i), \forall i \in \{1, 2, ..., p\} \tag{3.71}$$

avec les contraintes $\mathbf{A}\mathbf{z} - \underline{\mathbf{y}} \geq \mathbf{0}$ et $\bar{\mathbf{y}} - \mathbf{A}\mathbf{z} \geq \mathbf{0}$. La propagation des contraintes est ensuite appliquée sur les expressions $\mathbf{g}(\mathbf{x}) \in [\mathbf{z}^*]$ pour réduire le pavé $[\mathbf{x}]$, le domaine d'appartenance de $\mathbf{x}$.

3.3 Application à la localisation d'une plate-forme

Cette section pose le problème de la localisation d'une plate-forme en terme de résolution de CSP. La démarche choisie est l'étude d'une plate-forme dont les mouvements sont astreints dans un plan. Ce problème a l'avantage de faciliter la visualisation et la vérification des résultats. De plus, cette simple application permet de tester et de valider l'efficacité des méthodes de consistance décrites dans la section précédente.

Notre dispositif est composé de trois segments sur lesquels repose une plate-forme mobile (voir figure 3.11). Chaque segment est constitué d'une liaison prismatique et à ses extrémités de deux articulations rotoïdes. Les points de jonctions entre les articulations rotoïdes et la plate-forme sont notés $\mathbf{b}_1$, $\mathbf{b}_2$ et $\mathbf{b}_3$, ceux situés à la base du dispositif sont $\mathbf{a}_1$, $\mathbf{a}_2$ et $\mathbf{a}_3$. Les longueurs des segments ℓ_1, ℓ_2 et ℓ_3 varient en fonction des mouvements de translations induits par les articulations prismatiques. La plate-forme possède ainsi trois degrés de liberté défini par la position du centre (x_p, y_p) et son orientation θ.

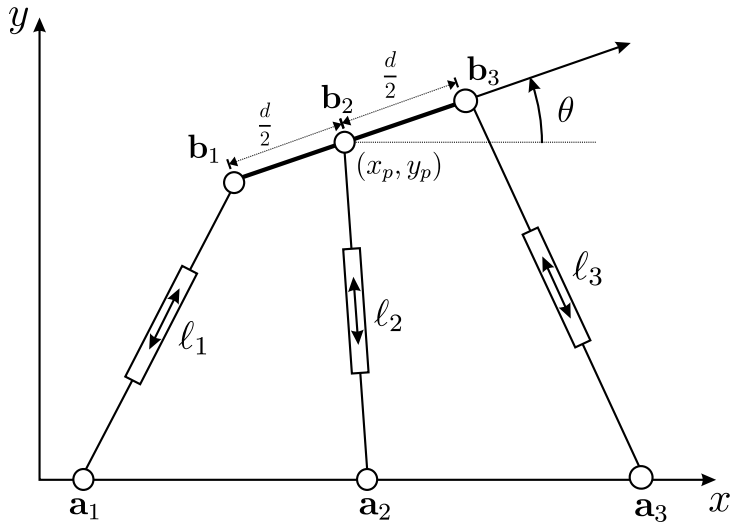

Figure 3.11 – Structure de la plate-forme dans un plan.

Le problème de localisation de la plate-forme est énoncé comme suit : pour des longueurs de segments fixées ℓ_1^*, ℓ_2^* et ℓ_3^*, comment déterminer toutes les configurations possibles pour la plate-forme ? Soit de caractériser l'ensemble

$$\mathbb{Z} = \{ (x_p, y_p, \theta) | \, \mathbf{h}\,(x_p, y_p, \theta) = \mathbf{0} \} \tag{3.72}$$

avec $h_i\,(x_p, y_p, \theta) = ||\mathbf{b}_i - \mathbf{a}_i||_2$ $(i = 1, 2, 3)$, il vient

$$\mathbf{h}\,(x_p, y_p, \theta) = \begin{pmatrix} \sqrt{\left(x_p + \frac{d}{2}\cos\theta - a_{1_x}\right)^2 + \left(y_p + \frac{d}{2}\sin\theta\right)^2} - \ell_1^* \\ \sqrt{\left(x_p - a_{2_x}\right)^2 + y_p^2} - \ell_2^* \\ \sqrt{\left(x_p - \frac{d}{2}\cos\theta - a_{3_x}\right)^2 + \left(y_p - \frac{d}{2}\sin\theta\right)^2} - \ell_3^* \end{pmatrix}. \tag{3.73}$$

Les paramètres de la plate-forme sont donnés par : $a_{1_x} = 2.5$, $a_{2_x} = 10$, $a_{3_x} = 15$, $d = 5$, $\ell_1^* = 8$, $\ell_2^* = 7$ et $\ell_3^* = 9$.

Les solutions analytiques possibles pour $\mathbf{h}\,(x_p, y_p, \theta) = \mathbf{0}$ ont été proposées dans [Gosselin and Merlet., 1994]. En considérant les paramètres ci-dessus, nous admettons que les configurations possibles pour la plate-forme sont :

	1	**2**	**3**	**4**
$x_p \approx$	10.32892	10.32892	5.57545	5.57545
$y_p \approx$	6.992267	-6.992267	-5.42433	5.42433
$\theta \approx$	0.685166	-0.685166	1.16577	-1.16577

Notre objectif est de déterminer le plus petit pavé enveloppe de $\mathbb{Z}$. Soit les domaines d'appartenance de chaque variables tel que $x_p \in [-100, 100]$, $y_p \in [-100, 100]$ et $\theta \in [-\pi, \pi]$. Le pavé résultant du produit cartésien de ces domaines est noté $[\mathbf{x}] = [-100, 100] \times [-100, 100] \times [-\pi, \pi]$. La propagation des contraintes sur $[\mathbf{x}]$ donne le pavé

$$\mathcal{C}_{\mathbb{Z}}\left([\mathbf{x}]\right) = [3.5, 13] \times [-7, 7] \times [-\pi, \pi]\,. \tag{3.74}$$

Concernant le pavé $\mathcal{C}_{\mathbb{Z}}\left([\mathbf{x}]\right)$, nous ne relevons aucune réduction suivant le domaine d'orientation de la plate-forme. L'ajout de contraintes redondantes ne permet pas de remédier efficacement à ce problème. Par contre, nous avons remarqué qu'une procédure d'épluchage du type de l'algorithme `SHAVING` améliore la contraction et permet d'échapper à cette situation de blocage. Il vient alors le pavé

$$[5.4, 11.6] \times [-7, 7] \times [-1.2, 1.2]\,. \tag{3.75}$$

Nous n'avons pas jugé utile d'indiquer les temps de calcul car la procédure de contraction est quasi instantanée (les temps de calcul sont inférieurs à la seconde). La largeur des tranches choisi pour l'épluchage du pavé est de l'ordre de $\lambda = 0.1$. Pour obtenir encore une réduction sur le pavé donné par (3.75), il suffit de diminuer la largeur des tranches de domaine λ, cela aura pour conséquence d'augmenter les temps de calcul.

3.4 Conclusion

Au cours de ce chapitre, nous nous sommes intéressés aux notions concernant le pavé enveloppe de l'ensemble des solutions d'inégalités non linéaires. Dans l'idéal, notre objectif a été de déterminer le plus petit pavé qui contient les solutions à des inégalités. Pour cela, nous avons présenté deux classes de méthodes de filtrage de domaines appelées aussi techniques de consistance. La propagation des contraintes, considérée comme une méthode de consistance faible, converge à coup sûr vers le plus grand pavé localement consistant. En ce qui concerne les méthodes de consistance forte, nous avons spécifié plusieurs stratégies qui améliorent la réduction des pavés en débloquant les situations de consistance locale. Il est cependant difficile de comparer ces différentes stratégies car leur efficacité est intimement liée au type de CSP traité. En fait, afin de réaliser un contracteur globalement consistant, une collaboration entre toutes ces stratégies est souhaitable. Bien que le contracteur résultant ne converge pas nécessairement vers le plus grand pavé globalement consistant, ces approches ont suscité un réel intérêt pour la résolution de

CSP constitué d'un grand nombre de paramètres. Sur ce sujet, l'application des méthodes de consistance fortes à l'étalonnage d'un robot de type série ont fait l'objet de publication dans [Baguenard et al., 2003].

Nous avons vu dans ce chapitre comment réduire des domaines qui ne sont pas consistants avec des contraintes, nous pouvons employer pour cela le terme d'approximation extérieure grossière. Le chapitre suivant utilisera les notions de contracteurs définies ici afin d'affiner la caractérisation de l'ensemble solutions à des inégalités non linéaires. Il sera question notamment d'approximation intérieure, soit de prouver que tous les points d'un pavé satisfont aux contraintes issues d'un CSP.

CHAPITRE 4

Caractérisation d'ensembles grâce au calcul par intervalles et aux contracteurs

La caractérisation d'ensembles est une méthode de représentation d'un sous-ensemble de $\mathbb{R}^n$. Il existe principalement deux méthodes pour la caractérisation des ensembles. La première est la représentation des sous-ensembles de $\mathbb{R}^n$ par des inégalités linéaires ou non. Cette représentation s'avère insuffisante car bien qu'elle soit exacte et définisse un critère d'appartenance pour les vecteurs de $\mathbb{R}^n$, elle ne fournie aucune information sur les éléments topologiques du sous-ensemble tels que : la forme, la frontière, la connexité, le volume, *etc.* La deuxième méthode de caractérisation d'ensembles réalise, dans un compact de $\mathbb{R}^n$, une énumération complète et systématique de tous les vecteurs issus du sous-ensemble considéré. Plus précisément, cette méthode consiste à approximer par recouvrement d'un sous-ensemble par un sous-pavage (union ou groupe de pavés).

Dans le chapitre précédent nous avons vu dans quelle mesure, il était possible d'obtenir le plus petit pavé qui contient, dans un compact, toutes les solutions à des inégalités. Ici, nous allons affiner ces travaux en proposant la représentation de l'ensemble solution à ces inégalités par des pavés (voir figure 4.1). Lors de la caractérisation d'un ensemble, nous distinguons deux catégories de sous-pavages. Le premier sous-pavage est constitué de pavés dont il a été prouvé que tous les points sont à l'intérieur de l'ensemble, ce sous-pavage est une approximation intérieure de l'ensemble. Le deuxième sous-pavage où tous les pavés se situent à cheval sur la frontière de l'ensemble. Sur l'illustration donnée par la figures 4.1a, nous avons représenté un ensemble $\mathbb{S}$ délimité par un trait plein. Sur les figures 4.1b et 4.1c, le groupe de pavés noirs $\underline{\mathbb{S}}$ correspond à l'approximation intérieure de $\mathbb{S}$. Quant au groupe des pavés blancs $\partial\mathbb{S}$, ils constituent une approximation de la frontière de $\mathbb{S}$. Nous avons un encadrement de l'ensemble $\mathbb{S}$ donnée par

$$\underline{\mathbb{S}} \subset \mathbb{S} \subset \overline{\mathbb{S}} \tag{4.1}$$

où $\overline{\mathbb{S}} = \underline{\mathbb{S}} \cup \partial\mathbb{S}$. La figure 4.1c illustre une meilleure caractérisation de $\mathbb{S}$ par rapport à la figure 4.1b. En effet, $\underline{\mathbb{S}}$ et $\partial\mathbb{S}$ sont respectivement plus volumineux et moins volumineux sur la première figure comparé à la deuxième figure.

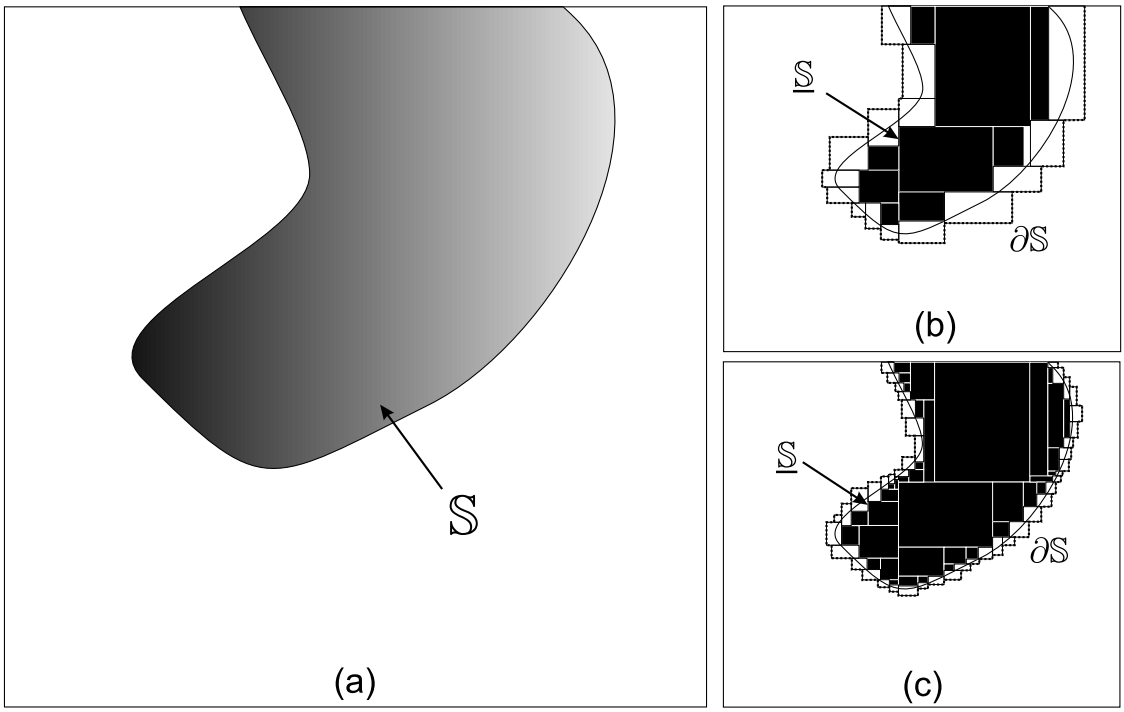

Figure 4.1 – Représentation de l'ensemble $\mathbb{S}$ par les sous-pavages $\underline{\mathbb{S}}$ et $\partial\mathbb{S}$.

La possibilité de raisonner en terme d'ensembles permet de résoudre des problèmes d'estimation non linéaires et d'optimisation de critères non convexes. Une multitude d'applications en automatique découlent de ces types de problèmes, l'analyse en stabilité de systèmes complexes, l'estimation de paramètres dans le cadre de la conception de systèmes ou encore dans la synthèse de lois de commande robuste.

Comme nous allons le voir dans ce chapitre, les techniques de caractérisation d'ensembles sont basées sur le calcul par intervalles, les contracteurs mais aussi des algorithmes de type *branch and bound*[1]. Dans cette section, nous allons décrire les principales méthodes de caractérisation d'ensembles ainsi que les formalismes qui leurs sont associés. Ces méthodes ont en commun la représentation d'un sous-ensemble de $\mathbb{R}^n$ par des pavés, c'est-à-dire son encadrement par deux sous-pavages. Les algorithmes que nous verrons mettent en œuvre des stratégies de partitionnement et d'exploration de l'espace, en cela ces méthodes ont le grand avantage de proposer des solutions numériques à des problèmes NP-difficiles.

[1] Algorithme de structure arborescent à complexité exponentielle.

4.1 Inversion ensembliste

L'une des méthodes de caractérisation d'ensembles ayant eu le plus d'applications dans le domaine particulier de l'automatique est la notion d'inversion ensembliste. Cette approche a été formalisée, développée et utilisée pour résoudre des problèmes d'estimations dans [Jaulin and Walter, 1993], [Moore, 1992].

Définition 4.1. Considérons une fonction continue de $\mathbb{R}^n$ dans $\mathbb{R}^m$ et un ensemble $\mathbb{Y} \subset \mathbb{R}^m$, l'inversion ensembliste de $\mathbb{Y}$ par $\mathbf{f}$ revient à caractériser l'ensemble

$$\mathbb{X} = \mathbf{f}^{-1}(\mathbb{Y}) = \{\mathbf{x} \in \mathbb{R}^n | \, \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\}. \tag{4.2}$$

La figure 4.2 illustre dans un espace de départ à gauche et dans un espace d'arrivée à droite l'inversion ensembliste de $\mathbb{Y}$ par la fonction $\mathbf{f}$. La caractérisation obtenue grâce à l'algorithme `SIVIA` *(Set Inversion Via Interval Analysis)* permet de déterminer les sous-pavages $\underline{\mathbb{X}}$ et $\overline{\mathbb{X}} = \underline{\mathbb{X}} \cup \partial\mathbb{X}$.

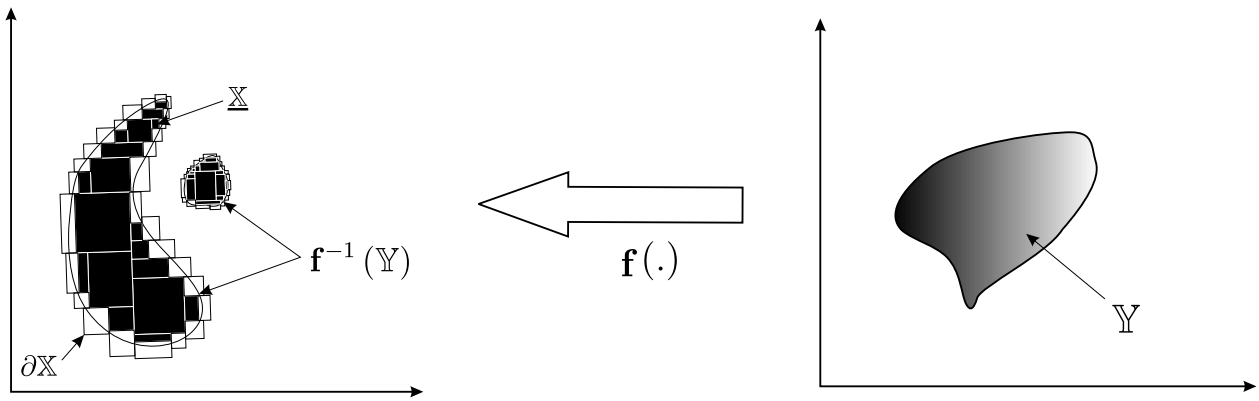

Figure 4.2 – Inversion ensembliste de $\mathbb{Y}$ par une fonction $\mathbf{f}$.

4.1.1 Principe

Les algorithmes d'inversion ensemblistes réalisent des partitions sur un domaine initial dans l'espace de départ. L'image de chaque partition ou pavé est évaluée grâce aux fonctions d'inclusion sur $\mathbf{f}$ issues du calcul par intervalles ou des contracteurs, ceci afin de démontrer entre autre que tous les points d'une partition ou d'un pavé appartiennent à $\mathbb{X}$ ainsi qu'il est défini en (4.2). Par conséquent, ces algorithmes, dont `SIVIA` en est une version, se compose de trois étapes qui sont : le choix d'un pavé initial censé contenir $\mathbb{X}$ ou du moins une partie de $\mathbb{X}$, les tests d'inclusion et la bissection.

4.1.1.1 Tests d'inclusion

Les différents tests sont réalisés à partir d'une fonction d'inclusion ou d'un contracteur pour $\mathbf{f}$. Notons $C_{\mathbb{X}}(.)$ et $C_{\mathbb{X}^c}(.)$ des contracteurs respectifs pour $\mathbb{X}$ et son complémentaire $\mathbb{X}^c$. L'ensemble $\mathbb{X}$ repré-

sente les vecteurs $\mathbf{x}$ qui satisfont la relation $\mathbf{f}(\mathbf{x}) \in \mathbb{Y}$. En considérant un cas où $\mathbb{Y}$ est un pavé ($\mathbb{Y} = [\mathbf{y}]$), $\mathbb{X}^c$ est défini par tout $\mathbf{x}$ qui vérifie

$$\mathbf{f}(\mathbf{x}) \notin [\mathbf{y}], \tag{4.3}$$

ce qui est traduit par,

$$\min_{i=1,\ldots,n} \left(\min \left(f_i(\mathbf{x}) - \underline{y}_i, \overline{y}_i - f_i(\mathbf{x}) \right) \right) < 0. \tag{4.4}$$

Ainsi formulé, il devient facile de concevoir un contracteur $C_{\mathbb{X}^c}(.)$ pour l'ensemble complémentaire de $\mathbb{X}$. Pour un pavé $[\mathbf{x}]_0$, nous sommes en mesure de démontrer trois cas de figure possibles :

1. Le pavé $[\mathbf{x}]_0$ est dit *inacceptable* si $\forall \mathbf{x} \in [\mathbf{x}]_0$, $\mathbf{f}(\mathbf{x}) \notin \mathbb{Y}$. Il suffit pour cela de montrer que soit $[\mathbf{f}]([\mathbf{x}]_0) \cap \mathbb{Y} = \emptyset$ ou que $C_{\mathbb{S}}([\mathbf{x}]_0) = \emptyset$.
2. Le pavé $[\mathbf{x}]_0$ est *acceptable* si $\forall \mathbf{x} \in [\mathbf{x}]_0$, $\mathbf{f}(\mathbf{x}) \in \mathbb{Y}$. Pour cela, nous montrons que $[\mathbf{f}]([\mathbf{x}]_0) \subset \mathbb{Y}$ ou $C_{\mathbb{S}^c}([\mathbf{x}]_0) = \emptyset$.
3. Le pavé $[\mathbf{x}]_0$ est dit *incertain* ou *ambigu* si $[\mathbf{x}]_0$ ne satisfait aucun des deux premiers cas de figure.

4.1.1.2 Bissection

Dans le cas où un pavé est incertain, nous réalisons une partition sur celui-ci afin d'effectuer à nouveau des tests d'inclusion sur les pavés générés. Rappelons que la bissection d'un pavé $[\mathbf{x}]$ de $\mathbb{R}^n$ permet de générer les pavés,

$$L[\mathbf{x}] = [\underline{x}_1, \overline{x}_1] \times ... \times \left[\underline{x}_j, \frac{\underline{x}_j + \overline{x}_j}{2} \right] \times ... \times [\underline{x}_n, \overline{x}_n], \tag{4.5}$$

$$R[\mathbf{x}] = [\underline{x}_1, \overline{x}_1] \times ... \times \left[\frac{\underline{x}_j + \overline{x}_j}{2}, \overline{x}_j \right] \times ... \times [\underline{x}_n, \overline{x}_n], \tag{4.6}$$

avec j qui correspond à la l'indice du plus grand axe ou côté de $[\mathbf{x}]$. L'avantage de cette opération résulte des propriétés issus du calcul par intervalles. En effet, plus la taille et le volume du pavé sont faibles, meilleurs sont les résultats obtenus par des fonctions d'inclusion, par conséquent par des contracteurs. Par exemple, il arrive qu'un pavé $[\mathbf{z}]$ soit incertain alors que chacune des parties $L[\mathbf{z}]$ ou $R[\mathbf{z}]$ est acceptable.

4.1.2 Algorithme SIVIA

`SIVIA` ainsi que ses propriétés de convergence sont décrites dans [Jaulin and Walter, 1993]. On se contente ici d'en donner une version récursive. L'implémentation de cet algorithme suppose que nous ayons à notre disposition une pile $\mathcal{P}$ qui nous permet de stocker les pavés incertains de grande taille et deux listes de pavés $\mathcal{L}_1$ et $\mathcal{L}_2$. Cette pile et ces deux listes sont initialement vides. Nous avons aussi des contracteurs $C_{\mathbb{X}}(.)$ et $C_{\mathbb{X}^c}(.)$ pour les ensembles $\mathbb{X}$ et $\mathbb{X}^c$. Pour un pavé initial $[\mathbf{x}]_0$, le premier appel de l'algorithme se fait par `SIVIA`$([\mathbf{x}]_0)$.

Algorithme : SIVIA(Entrée : $[\mathbf{x}]$)	
1	$\mathcal{P} \leftarrow \emptyset$; $\mathcal{P} \leftarrow [\mathbf{x}] \cup \mathcal{P}$;
2	Tant que $(\mathcal{P} \neq \emptyset)$ faire :
3	$[\mathbf{x}] \leftarrow$pop$(\mathcal{P})$; $[\mathbf{x}] \leftarrow C_{\mathbb{X}}([\mathbf{x}])$;
4	Si $([\mathbf{x}] \neq \emptyset)$:
5	Si $(C_{\mathbb{X}^c}([\mathbf{x}]) = \emptyset)$: $\mathcal{L}_2 \leftarrow \mathcal{L}_2 \cup \{[\mathbf{x}]\}$;
6	Sinon :
7	Si $(w([\mathbf{x}]) \leq \varepsilon)$: $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup \{[\mathbf{x}]\}$;
8	Sinon : $\mathcal{P} \leftarrow \{L[\mathbf{x}]\} \cup \{R[\mathbf{x}]\} \cup \mathcal{P}$;
9	Fin si
10	Fin si
11	Fin tant que
12	Fin.

La procédure pop$(\mathcal{P})$ extrait le pavé du sommet de la pile $\mathcal{P}$. L'algorithme SIVIA fonctionne de la façon suivante. La pile $\mathcal{P}$ contient les pavés qui n'ont pas encore été définis acceptables ou inacceptables par les tests d'inclusion. Les pavés incertains sont partitionnés par bissection puis insérer au sommet de $\mathcal{P}$ jusqu'à ce que leur taille soit inférieure à ε. Ce paramètre est inversement proportionnel à la précision requise pour la caractérisation de $\mathbb{X}$. Quant aux pavés incertains $[\mathbf{x}]$ tel que $w([\mathbf{x}]) < \varepsilon$, ils sont retirer de la pile $\mathcal{P}$ puis rangés dans la liste $\mathcal{L}_1$ qui représente $\partial\mathbb{X}$. Les pavés acceptables sont stockés dans la liste $\mathcal{L}_2$, cette liste caractérise l'ensemble $\underline{\mathbb{X}}$.

4.1.3 Limite et complexité de SIVIA

Lorsque l'ensemble $\mathbb{X} = \mathbf{f}^{-1}(\mathbb{Y})$ est de grande dimension (supérieure par exemple à 5), il devient quasiment impossible de réaliser une bonne approximation de $\mathbb{X}$. En effet, les temps de calcul tendent vers l'infini si nous exigeons une bonne précision sur la caractérisation (un volume de $\partial\mathbb{X}$ très faible).

Pour se donner un ordre de grandeur concernant les temps de calcul, nous allons étudier la complexité de l'algorithme SIVIA. Rappelons que la complexité d'un algorithme A est une fonction C_A qui donne dans le pire des cas, une approximation du nombre d'instructions effectuées lors d'une exécution de A. Nous avons donc,

$$C_{\texttt{SIVIA}}(n) \approx \left(\frac{w([\mathbf{x}]_0)}{\varepsilon}\right)^n \tag{4.7}$$

avec $[\mathbf{x}]_0$ le pavé de recherche initial, n la dimension de $[\mathbf{x}]_0$ et ε la précision requise. Selon la formule (4.7), SIVIA est de complexité exponentielle.

Exemple 4.2. *Considérons un pavé* $[\mathbf{x}]_0$ *de dimension 5, avec les longueurs de chaque côté égales à 1. Si nous découpons systématiquement les pavés issus de* $[\mathbf{x}]$ *tant que leurs tailles sont supérieures à* $\varepsilon = 10^{-2}$*, le nombre de bissections atteint alors un milliard. Le temps de calcul est alors directement proportionnel au nombre de bissections réalisées.*

Nous pouvons nous contenter d'une grossière approximation de $\mathbb{X}$ (choix d'un paramètre ε assez grand) dans certains problèmes, nous limiterons dans ce cas l'explosion des calculs numériques. D'autres pistes ont été explorées afin d'atténuer l'explosion des calculs. Parmi celles-ci nous avons : des critères plus élaborés et plus diversifiés pour le choix de l'axe de bissection (voir à ce sujet [Hansen, 1992a] et [Cendes and Ratz, 1997]), ou encore la réalisation de fonctions d'inclusion moins pessimistes (voir [Raïssi, 2004]). Les améliorations sont difficilement mesurable car ces techniques sont utilisées dans des cadres d'applications particuliers. Par exemple, certaines stratégies de bissection ont permis d'améliorer les temps de calcul pour les ensembles de faibles dimension. Ces mêmes stratégies se sont montrées nettement moins efficaces que les stratégies classiques, indiquées par les relations (4.5) et (4.6), pour des ensembles de grande dimension (voir [Raïssi et al., 2004]).

4.2 Image ensembliste

La méthode présentée ici consiste à approximer l'image d'un sous-ensemble $\mathbb{X}$ par une fonction $\mathbf{f}:\mathbb{R}^n \rightarrow \mathbb{R}^m$. Ce qui revient à caractériser l'ensemble

$$\mathbf{f}(\mathbb{X}) \triangleq \{\mathbf{f}(\mathbf{x}) | \, \mathbf{x} \in \mathbb{X}\}. \tag{4.8}$$

Bien que la caractérisation de $\mathbf{f}(\mathbb{X})$ trouve moins d'applications dans le domaine de l'estimation de paramètres en automatique, les principes de la méthode utilisée méritent d'être rappelés dans l'inventaire des méthodes ensemblistes.

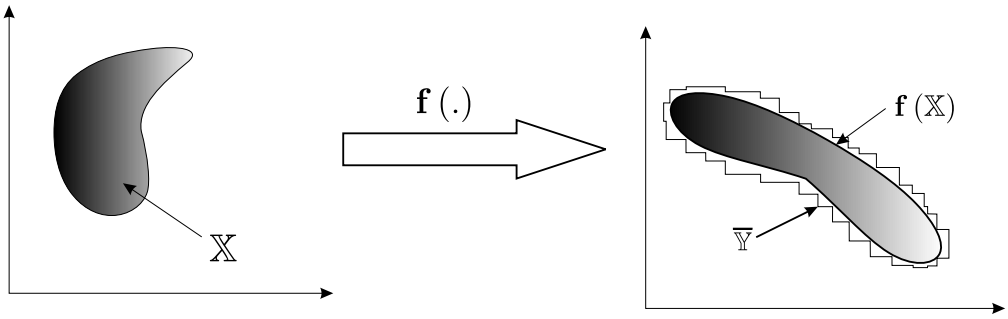

Figure 4.3 – Image d'un ensemble $\mathbb{X}$ par une fonction.

L'algorithme développé dans [Kieffer, 1999] permet d'obtenir une approximation extérieure $\overline{\mathbb{Y}} \subset$

$\mathbf{f}(\mathbb{X})$ (voir figure 4.3). Par contre, il est moins simple de calculer une approximation intérieure $\underline{\mathbb{Y}}$, en effet il faudrait réussir à montrer pour un pavé que $[\mathbf{y}] \subset \mathbf{f}(\mathbb{X})$, autrement dit

$$\forall \mathbf{y} \in [\mathbf{y}], \exists \mathbf{x} \in \mathbb{X} \text{ tel que } \mathbf{y} = \mathbf{f}(\mathbf{x}), \tag{4.9}$$

ce qui n'est pas facile à prouver.

La caractérisation de $\overline{\mathbb{Y}}$ suit trois étapes qui sont :

- **Le hachage :** consiste en un partitionnement de $\mathbb{X}$ afin de générer une liste $\mathcal{L}_x$ de pavés $[\mathbf{x}]$ tels que $w([\mathbf{x}]) < \varepsilon$, où $\varepsilon > 0$ est un réel qui indique la précision de la l'approximation extérieure.
- **L'évaluation :** consiste à réaliser une première approximation de $\overline{\mathbb{Y}}$ en calculant les images par $\mathbf{f}$ des pavés issus de $\mathcal{L}_x$. Ces évaluations sont stockées dans une liste $\mathcal{L}_y$.
- **La régularisation :** permet de réaliser une approximation plus fine de $\overline{\mathbb{Y}}$ en réalisant un sous-pavage régulier à partir de $\mathcal{L}_y$ (voir chapitre 2, section 2.2.1). Pour cela, nous réalisons un algorithme qui reprend les mêmes principes que `SIVIA`. En entrée, nous avons le plus petit pavé qui contient le sous-pavage représenté par $\mathcal{L}_y$. La fonction d'inclusion est donnée par $\mathrm{Id}([\mathbf{x}]) = [\mathbf{x}]$, où $\mathrm{Id}(.)$ correspond à la fonction identité.

Ces étapes aboutissent à la construction d'un algorithme d'approximation extérieure pour $\mathbb{Y}$. Comme il est décrit ci-dessous, cet algorithme est constitué par deux routines. La routine `ImageSP` qui traduit le hachage et l'évaluation et la routine `Regularize` qui représente l'étape de régularisation.

Algorithme : `ImageSP(Entrée` : $\mathbb{X}$`)`	
1	$\mathcal{L}_x \leftarrow \emptyset$; $\mathcal{L}_y \leftarrow \emptyset$;
2	$\mathcal{L}_x \leftarrow$ `Hachage`$(\mathbb{X}, \varepsilon)$;
3	$\forall [\mathbf{x}] \subset \mathcal{L}_x$: $\mathcal{L}_y \leftarrow \mathcal{L}_y \cup \{[\mathbf{f}]([\mathbf{x}])\}$;
4	$\mathcal{L}_y \leftarrow$ `Regularize`$(\mathcal{L}_y)$;
5	`Retouner` $\mathcal{L}_y$;

Algorithme : `Regularize(Entrée` : $\mathcal{L}_y$`)`	
1	$\forall [\mathbf{y}] \subset \mathcal{L}_y$: $[\mathbf{y}]_0 \leftarrow [[\mathbf{y}]_0 \cup [\mathbf{y}]]$;
2	$\mathcal{P} \leftarrow [\mathbf{y}]_0$; $\mathcal{L} \leftarrow \emptyset$;
3	`Tant que` $(\mathcal{P} \neq \emptyset)$ `faire :`
4	$[\mathbf{y}]_0 \leftarrow \mathtt{pop}(\mathcal{P})$;
5	`Si` $([\mathbf{y}]_0 \subset \mathcal{L}_y)$: $\mathcal{L} \leftarrow \mathcal{L} \cup [\mathbf{y}]_0$;
6	`Sinon :`
7	`Si` $(w([\mathbf{y}]_0) < \varepsilon)$: $\mathcal{L} \leftarrow \mathcal{L} \cup [\mathbf{y}]_0$;
8	`Sinon` : $\mathcal{P} \leftarrow \{L[\mathbf{y}]_0, R[\mathbf{y}]_0\} \cup \mathcal{P}$;
9	`Fin tant que`
10	`Retourner` $\mathcal{L}$;

4.3 Connexité d'un ensemble

Pour clore ce tour d'horizon sur les méthodes ensemblistes, nous présentons ici des travaux traitants de certaines propriétés topologiques d'un ensemble comme par exemple *la connexité*.

Définition 4.3 (Connexité). Un ensemble topologique $\mathbb{S}$ est connexe, si pour tous les couples de points $(\mathbf{x}, \mathbf{y}) \in \mathbb{S}$, il existe une fonction continue $\mathbf{f} : [0, 1] \rightarrow \mathbb{S}$ telle que $\mathbf{f}(0) = \mathbf{x}$ et $\mathbf{f}(1) = \mathbf{y}$. Cette définition est illustrée par la figure 4.4. A droite, nous avons un ensemble connexe, il existe un chemin dans $\mathbb{S}$ qui relie chaque couple de points $(\mathbf{x}, \mathbf{y}) \in \mathbb{S}$. A gauche est représenté un ensemble non connexe constitué de trois composantes connexes.

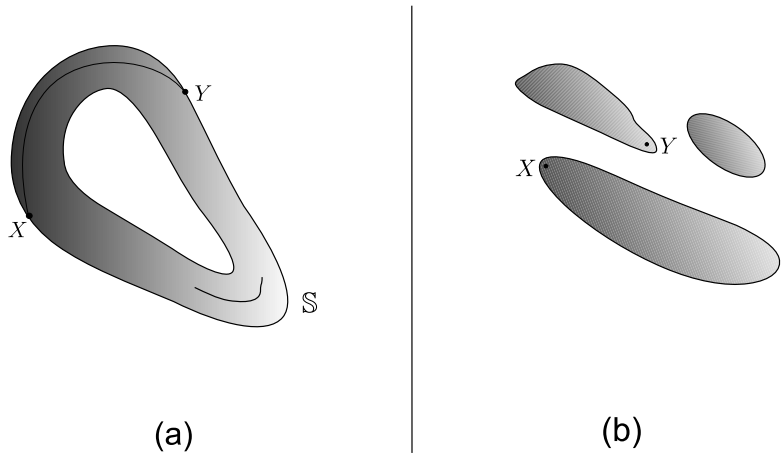

Figure 4.4 – (a) : Ensemble connexe; (b) : Ensemble non connexe avec trois composantes connexes.

Dans [Delanoue et al., 2004], N. Delanoue et *al.* présentent sous le nom de `CIA` (*path-**C**onnected using **I**nterval **A**nalysis*), un algorithme qui prouve la connexité et détermine le nombre de composantes connexes d'un ensemble défini par des inégalités non linéaires. Cet algorithme est basé sur la notion de *sous-ensembles étoilés*.

Définition 4.4 (Point étoile et ensemble étoilé). Un point v est une étoile pour un sous-ensemble Euclidien $\mathbb{X}$, si $\mathbb{X}$ contient tous les segments de droite qui relient v à chaque point de $\mathbb{X}$. L'ensemble $\mathbb{X}$ est dit v-étoilé (voir figure 4.5).

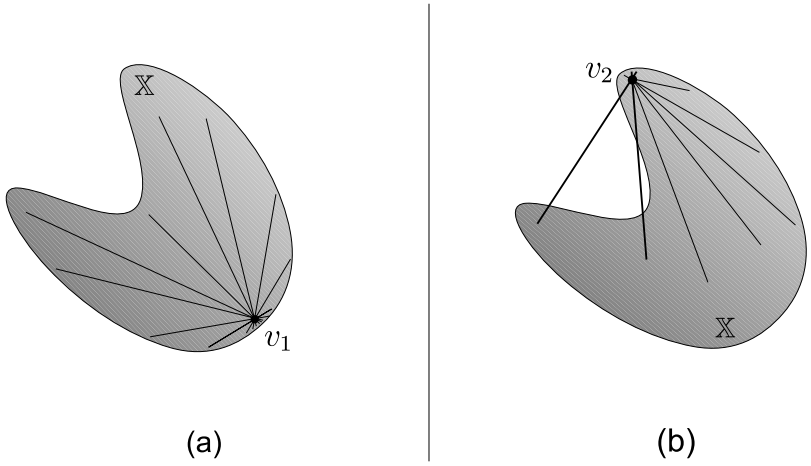

Figure 4.5 – (a) : Le point v_1 est une étoile pour $\mathbb{X}$; (b) : Par contre, v_2 ne l'est pas.

Considérons l'ensemble $\mathbb{S} = \{\mathbf{x} |\, f(\mathbf{x}) \leq 0\}$ où f est une fonction de classe C^1 définie de $\mathbb{R}^n$ vers $\mathbb{R}$. Afin de déterminer le nombre de composantes connexes de $\mathbb{S}$, la méthode `CIA` met en oeuvre un partitionnement du pavé censé contenir $\mathbb{S}$. Chaque pavé $[\mathbf{x}]$ issu de ce partitionnement est ensuite soit éliminé par bissection et par contraction, soit qualifié *d'étoilé* s'il est possible de trouver une étoile $\mathbf{v}^*$ pour l'ensemble $\mathbb{X} = \{\mathbf{x} \in [\mathbf{x}] |\ f(\mathbf{x}) \leq 0\}$. Ceci revient à montrer, grâce au calcul par intervalles, qu'il existe un point $\mathbf{v}^* \in [\mathbf{x}]$ tel que

$$\forall \mathbf{x} \in [\mathbf{x}]\,,\ f(\mathbf{x}) \neq 0 \text{ ou } \nabla f(\mathbf{x}) \cdot (\mathbf{x} - \mathbf{v}^*) > 0. \tag{4.10}$$

Dans la pratique, pour trouver un vecteur $\mathbf{v}^*$, nous considérons les coins du pavé $[\mathbf{x}]$. Des graphes sont construits en reliant, par des segments de droite, les centres des couples de pavés étoilés et voisins[2] (voir les détails de l'algorithme `CIA` dans [Delanoue et al., 2004]).

4.4 Conclusion

Dans ce chapitre, nous avons présenté un tour d'horizon de différentes méthodes de caractérisation d'ensembles par des techniques intervalles. La principale de ces méthodes permet de caractériser l'image réciproque d'un ensemble par une fonction vectorielle. Cette méthode appelée inversion ensembliste a prouvé son efficacité dans la résolution de problèmes d'estimation non linéaire (caractérisation d'ensembles stabilisants pour les coefficients de pôlynomes caractéristiques ainsi que l'estimation d'état). Quelques ouvrages et articles de références sur le sujet sont donnés par [Walter and Jaulin, 1994], [Jaulin et al., 2001], [Moore, 1979] et [Moore, 1992]. Quant à la connexité d'ensembles, elle concerne

[2]Les pavés $[\mathbf{x}]$ et $[\mathbf{y}]$ sont dits voisins si $[\mathbf{x}] \cap [\mathbf{y}] \neq \emptyset$.

surtout les problèmes de planification de trajectoire. Comme nous l'avons expliqué précédemment, les méthodes ensemblistes ont toutes l'inconvénient majeur d'être gourmandes en temps de calcul. De ce fait, elles s'appliquent dans le cadre des problèmes d'estimation avec un petit nombre de paramètres, c'est-à-dire pour la caractérisation d'ensembles de petite dimension (typiquement < 5). Nous avons tenté de réaliser, grâce aux contracteurs, l'étalonnage d'un robot 6 axes (voir [Baguenard et al., 2003]). Ce problème a été présenté comme un problème de grande dimension (avec 14 paramètres). Pour des temps de calcul raisonnable, nous avons obtenu en utilisant les techniques de propagation de contraintes un petit pavé qui contient les paramètres solutions. Ce résultat ne donne aucune solution exacte, notre problème d'étalonnage n'a donc été résolu que partiellement. Une méthode de recherche locale serait complémentaire afin de déterminer pour notre robot, une configuration possible du robot dans le pavé obtenu. Nous posons comme perspective, la résolution de problèmes de grandes dimensions par des approches combinant le calcul par intervalles à des heuristiques. Ainsi, nous abandonnons l'idée de caractériser un ensemble de grande dimension au profit de la recherche d'une solution ponctuelle ou d'un pavé qui satisfait un certain nombre de contraintes.

Dans les chapitres suivants, nous nous intéresserons au cas de la projection d'ensembles et aux applications envisageables. Après des études menées, nous avons conçu un solveur capable de traiter, par l'intermédiaire d'un fichier d'entrée de type texte, les problèmes dont les solutions s'expriment comme la projection d'un ensemble défini par des inégalités non linéaires. Nous présenterons à cet effet, la structure ainsi que le principe de fonctionnement de ce solveur.

CHAPITRE 5

Projection d'ensembles

Nous abordons dans ce chapitre, la notion principale sur laquelle nos travaux de thèse ont porté. A cet effet, *la projection d'ensembles* apparaît comme une notion importante. D'une part, parce qu'elle englobe comme nous allons le voir le principe d'inversion ensembliste. D'autre part parce qu'elle permet la caractérisation de certains ensembles dont les expressions contiennent des quantificateurs universels et existenciels ($\forall$, $\exists$).

Afin d'automatiser la résolution des problèmes grâce au calcul par intervalles et aux algorithmes de caractérisation d'ensembles, nous avons développé tout au long de cette thèse un solveur nommé PROJ2D. Ce solveur permettra dans la suite de résoudre la plupart des problèmes abordés dans la deuxième partie de ce document.

Pour d'améliorer l'efficacité des algorithmes de projection présentés dans [Braems, 2002], nous apporterons principalement deux contributions. La première de ces contributions consiste à former, à partir des contraintes, des graphes connexes de variables quantifiées. L'objectif est de décomposer, dans le cadre général, la projection d'un ensemble à n dimensions en une intersection de projections d'ensembles dont la somme des dimensions vaut n (voir section 5.2.3.2). Notre deuxième contribution consiste à rendre les algorithmes de projection plus efficace en améliorant l'approximation intérieure, c'est-à-dire la caractérisation de l'intérieur de la projection. Pour cela, nous présenterons sous le nom de `IAVIC` (*Inner Approximation Via Interval Computation*) une méthode qui détermine dans un pavé incertain, grâce à un algorithme de recherche ponctuelle, une partie ou un domaine acceptable (voir section 5.2.6).

5.1 Définition et principe

5.1.1 Rappels

Définition 5.1 (Projection d'un ensemble). Considérons un sous-ensemble de $\mathbb{R}^n$ défini par,

$$\mathbb{S} = \{ (\mathbf{p}, \mathbf{q}) \in \mathbb{P} \times \mathbb{Q} | \, \mathbf{f}(\mathbf{p}, \mathbf{q}) \in \mathbb{Y} \}, \tag{5.1}$$

avec $\mathbf{f}$ une fonction vectorielle définie de $\mathbb{R}^n$ vers $\mathbb{R}^m$, $\mathbb{Y}$ un sous-ensemble de $\mathbb{R}^m$, $\mathbb{P}$ et $\mathbb{Q}$ deux sous-ensembles respectivement de $\mathbb{R}^{n_p}$ et de $\mathbb{R}^{n_q}$ tels que $n_p + n_q = n$. La *projection* de $\mathbb{S}$ suivant $\mathbb{P}$ est

donnée par

$$\begin{aligned} \text{Proj}_{\mathbf{p}}(\mathbb{S}) &= \{\mathbf{p} \in \mathbb{P} | \, \exists \mathbf{q} \in \mathbb{Q}, \ (\mathbf{p}, \mathbf{q}) \in \mathbb{Z}\} \\ &= \{\mathbf{p} \in \mathbb{P} | \, \exists \mathbf{q} \in \mathbb{Q}, \ \mathbf{f}(\mathbf{p}, \mathbf{q}) \in \mathbb{Y}\}, \end{aligned} \tag{5.2}$$

que l'on note aussi $\Pi_{\mathbf{p}}(\mathbb{S})$.

Remarque 5.2. *Les problèmes d'inversion ensemblistes (voir chapitre 4) peuvent être formulés en terme de projection d'ensembles. Précisons dans le même temps que la réciproque est fausse. En effet, soit l'ensemble* $\mathbb{X}$ *défini par*

$$\mathbb{X} = \mathbf{f}^{-1}(\mathbb{Y}) = \{\mathbf{x} \in \mathbb{R}^n | \ \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\}, \tag{5.3}$$

ce qui est équivalent à

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n | \ \exists \mathbf{y} \in \mathbb{Y}, \ \mathbf{f}(\mathbf{x}) = \mathbf{y}\}. \tag{5.4}$$

Par conséquent, $\mathbb{X} =$*Proj*$_{\mathbf{x}}(\mathbb{Z})$ *avec*

$$\mathbb{Z} = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{Y} | \ \mathbf{f}(\mathbf{x}) - \mathbf{y} = \mathbf{0}\}. \tag{5.5}$$

Il peut être intéressant d'illustrer cette remarque par l'exemple ci-dessous.

Exemple 5.3. *Considérons,*

$$\mathbb{X} = \left\{(x, y) \in \mathbb{R}^2 \middle| \ x^2 + y^2 \leq 1\right\}, \tag{5.6}$$

$\mathbb{X}$ *représente un disque de centre* $O(0, 0)$ *et de rayon* $r = 1$ *(voir figure 5.1a). La formulation en terme de projection d'ensembles nous donne* $\mathbb{X} =$*Proj*$_{(x,y)}(\mathbb{Z})$ *avec*

$$\mathbb{Z} = \left\{(x, y, z) \in \mathbb{R}^2 \times [0, 1] \middle| \ x^2 + y^2 - z = 0\right\}, \tag{5.7}$$

qui représente une paraboloïde en trois dimensions (voir figure 5.1b).

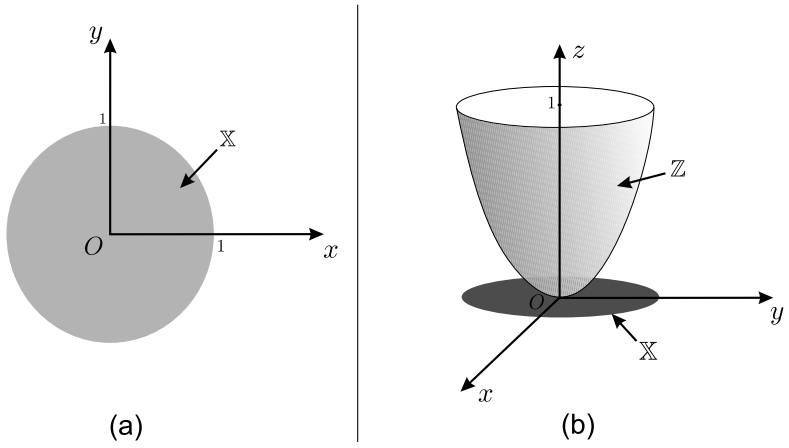

Figure 5.1 – (a) : Représentation de $\mathbb{S}$; (b) : Projection de la paraboloïde $\mathbb{Z}$.

Remarque 5.4. *La projection d'ensembles propose des solutions à deux types de problèmes. Ces problèmes se présentent sous forme de caractérisation des ensembles*

$$\mathbb{S}_1 = \{\,\mathbf{p} | \,\exists\,\mathbf{q}\,,\,\mathbf{f}\,(\mathbf{p},\mathbf{q}) \in [\mathbf{y}]\} \tag{5.8}$$

et

$$\mathbb{S}_2 = \{\,\mathbf{p} | \,\forall\mathbf{q}\,,\,\mathbf{f}\,(\mathbf{p},\mathbf{q}) \in [\mathbf{y}]\}\,. \tag{5.9}$$

La caractérisation de $\mathbb{S}_1$ découle de la définition (5.1). Quant au cas de $\mathbb{S}_2$, nous présenterons sa caractérisation dans les termes qui suivent. Considérons le complémentaire de $\mathbb{S}_2$,

$$\begin{array}{rcl} \mathbb{S}_2^c = \neg\mathbb{S}_2 & = & \neg\,\{\,\mathbf{p} | \,\forall\mathbf{q}\,,\,\mathbf{f}\,(\mathbf{p},\mathbf{q}) \in [\mathbf{y}]\} \\ & = & \{\,\mathbf{p} | \,\exists\,\mathbf{q}\,,\,\neg\,(\mathbf{f}\,(\mathbf{p},\mathbf{q}) \in [\mathbf{y}])\}\,. \end{array} \tag{5.10}$$

L'expression $\mathbf{f}\,(\mathbf{p},\mathbf{q}) \in [\mathbf{y}]$ est équivalent à $r\,(\mathbf{p},\mathbf{q}) \geq 0$, avec

$$r\,(\mathbf{p},\mathbf{q}) = \min_{i=1,\ldots,m} \Big(\min\Big(f_i\,(\mathbf{p},\mathbf{q}) - \underline{y}_i,\, \overline{y}_i - f_i\,(\mathbf{p},\mathbf{q})\Big)\Big) \tag{5.11}$$

D'où

$$\mathbb{S}_2^c = \{\,\mathbf{p} | \,\exists\,\mathbf{q}\,,\,r\,(\mathbf{p},\mathbf{q}) < 0\}\,. \tag{5.12}$$

Nous comprenons qu'en caractérisant $\mathbb{S}_2^c$ dans un pavé initial $[\mathbf{p}]_0$, nous obtenons par la même occasion une approximation de $\mathbb{S}_2$. Il vient effectivement,

$$\mathbb{S}_2 = [\mathbf{p}]_0 \setminus \mathbb{S}_2^c = \{\,\mathbf{p} \in [\mathbf{p}]_0 |\ \mathbf{p} \notin \mathbb{S}_2^c\}\,. \tag{5.13}$$

5.1.2 Principe de la projection d'ensembles

La projection d'ensembles s'inscrit dans la classe des méthodes de caractérisation d'ensembles de type *branch and bound* basées sur le calcul par intervalles. A l'instar de ces méthodes, la projection d'un ensemble nécessite une stratégie de partitionnement par bissection, mais à ceci près que l'on dissocie les variables dites quantifiées ($\mathbf{q} = \left(q_1, q_2, ..., q_{n_q}\right)^{\mathrm{T}}$) de celles qui ne le sont pas ($\mathbf{p} = \left(p_1, p_2, ..., p_{n_p}\right)^{\mathrm{T}}$).

Soit l'ensemble $\mathbb{S}$ donné par (5.1), le principe de la méthode se résume à ceci, puisque nous avons besoin de peu d'informations sur les variables quantifiées, nous partitionnons moins les domaines d'appartenance de ces variables. Ceci se traduit par un algorithme de partitionnement de type `SIVIA` pour le domaine des variables non quantifiées (ou projetées), puis un contracteur de haut niveau qui permet de prouver trois situations possibles :

1. $[\mathbf{p}]$ est inacceptable ($[\mathbf{p}] \cap \mathrm{Proj}_{\mathbf{p}}\,(\mathbb{S}) = \emptyset$) si $C_{\mathbb{S}}\,([\mathbf{p}] \times [\mathbf{q}]) = \emptyset$;
2. $[\mathbf{p}]$ est acceptable ($[\mathbf{p}] \subset \mathrm{Proj}_{\mathbf{p}}\,(\mathbb{S})$) si $C_{\mathbb{S}^c}\,([\mathbf{p}] \times [\mathbf{q}]) = \emptyset$ ou s'il existe $\mathbf{q}_0 \in [\mathbf{q}]$ tel que : $C_{\mathbb{S}^c}\,([\mathbf{p}] \times \mathbf{q}_0) = \emptyset$ (voir figure 5.2);

3. $[\mathbf{p}]$ est incertain si aucune des deux premières conditions n'est satisfaite.

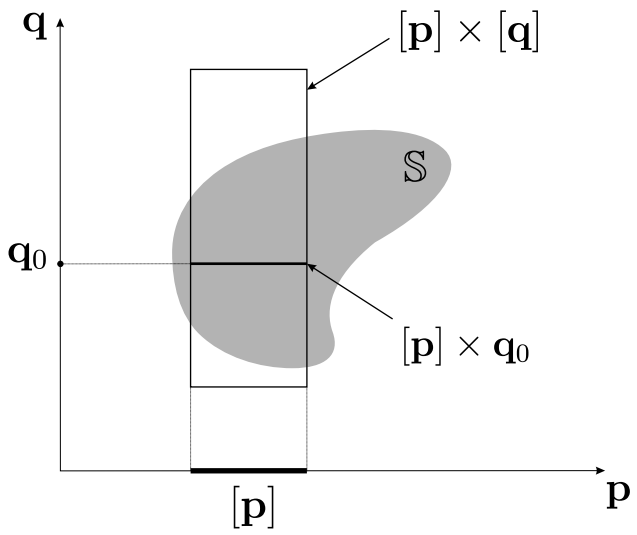

Figure 5.2 – Approximation intérieure pour la caractérisation de $\text{Proj}_{\mathbf{p}}(\mathbb{S})$.

La façon de montrer que $[\mathbf{p}]$ est acceptable constitue la différence notable avec les algorithmes d'inversion ensembliste. Comme nous l'avons illustré par la figure 5.2, nous avons $C_{\mathbb{S}^c}([\mathbf{p}] \times [\mathbf{q}]) \neq \emptyset$ car le pavé $[\mathbf{p}] \times [\mathbf{q}]$ n'est pas inclus dans $\mathbb{S}$. Grâce à un contracteur garantissant la consistance locale, il est possible de montrer que $C_{\mathbb{S}^c}([\mathbf{p}] \times \mathbf{q}_0) = \emptyset$ car $[\mathbf{p}] \times \mathbf{q}_0 \subset \mathbb{S}$. Il devient alors évident que trouver un pavé $[\mathbf{p}] \times [\mathbf{q}] \subset \mathbb{S}$ engendre plus de bissections que de trouver un vecteur $\mathbf{q}_0 \in [\mathbf{q}]$ tel que $[\mathbf{p}] \times \mathbf{q}_0 \subset \mathbb{S}$. D'où notre problème fondamental, à savoir comment trouver ce vecteur $\mathbf{q}_0$? Une première stratégie (voir [Braems, 2002]) consiste à partitionner le domaine $[\mathbf{q}]$ afin de tester les centres des pavés générés. Nous verrons dans les sections 5.2.4 et 5.2.6, les détails de l'algorithme de projection d'ensembles. Nous proposerons par la même occasion une stratégie plus élaborée afin de trouver un vecteur $\mathbf{q}_0$.

Nous présenterons dans la suite un solveur qui combine la projection d'ensembles à la représentation des contraintes sous forme d'arbre. Ce type de représentation a pour principal avantage, l'automatisation du traitement de problèmes qui utilisent le même formalisme.

5.2 Solveur - Proj2D

5.2.1 Présentation

Nous avons conçu et développé pour les besoins de cette thèse un solveur appelé PROJ2D[1] (**Proj**ection en **2 D**imensions). Ce solveur constitue à notre connaissance l'un des premiers logiciels qui utilise le cal-

[1] Logiciel librement disponible sur `http://www.istia.univ-angers.fr/~dao/Proj2DV5.zip`

cul par intervalles, les contracteurs et des algorithmes de partitionnement du type SIVIA. Parmi ces logiciel, nous avons : [Spa, 2000], [Lottaz, 2000], [Ratschan, 2000], [Granvilliers, 2002], [van Hentenryck et al., 1997], [Benhamou and Older, 1997], *etc.*

Étant donné un ensemble $\mathbb{S}$ défini par les variables $x_i \in [x_i]$ avec $i \in \{1, 2, ...n\}$, qui satisfont les inégalités

$$f_j\left(x_1, x_2, ..., x_n\right) \in [y_j], \forall\, j = 1, 2, ..., m, \tag{5.14}$$

PROJ2D permet de caractériser les ensembles $\underline{\mathbb{S}}_{(x_k,x_l)}$ et $\overline{\mathbb{S}}_{(x_k,x_l)}$ tels que

$$\underline{\mathbb{S}}_{(x_k,x_l)} \subset \text{Proj}_{(x_k,x_l)}\left(\mathbb{S}\right) \subset \overline{\mathbb{S}}_{(x_k,x_l)} \tag{5.15}$$

avec $k \neq l$ et $\text{Proj}_{(x_k,x_l)}\left(\mathbb{S}\right)$ défini selon la relation (5.2).

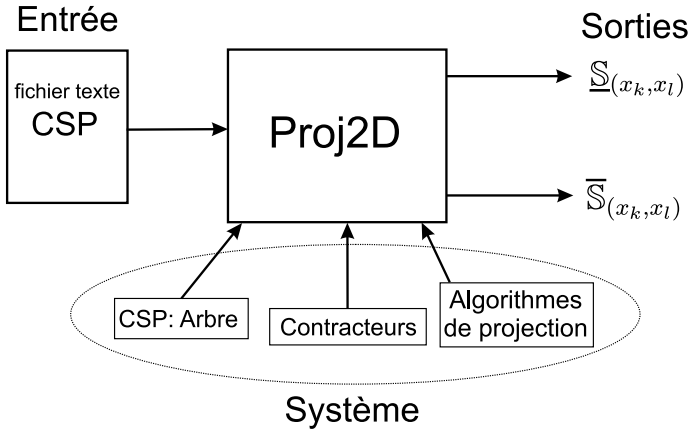

Figure 5.3 – Solveur Proj2D.

Comme le montre la figure 5.3, le solveur est constitué en entrée d'un fichier texte et en sortie de deux listes de pavés qui représentent $\underline{\mathbb{S}}_{(x_k,x_l)}$ et $\overline{\mathbb{S}}_{(x_k,x_l)}$. Le fichier texte (*.pb) mentionne les différentes informations concernant un CSP[2], notamment les domaines d'appartenance des variables, les contraintes, les deux variables de projection, *etc.* Les données du fichier d'entrée sont saisies suivant le modèle ci-dessous.

Format du fichier d'entrée pour PROJ2D (extension *.*pb*)

Variables
x_1 in $[\underline{x}_1, \overline{x}_1]$
x_2 in $[\underline{x}_2, \overline{x}_2]$
$\vdots$
x_n in $[\underline{x}_n, \overline{x}_n]$

[2] *Constraints Satisfaction Problem* (Problème de satisfaction de contraintes)

...
Constraints
$f_1(x_1, x_2, ..., x_n)$ in $\left[\underline{y}_1, \overline{y}_1\right]$
$f_2(x_1, x_2, ..., x_n)$ in $\left[\underline{y}_2, \overline{y}_2\right]$
$\vdots$
$f_m(x_1, x_2, ..., x_n)$ in $\left[\underline{y}_m, \overline{y}_m\right]$
Projected Variables
x_k;x_l;
Epsilon
ε
End.

Dans cette section, nous avons pu donner un aperçu global de PROJ2D. A cet effet, les entrées et les sorties du solveur ont fait l'objet d'une description succincte. Les différents éléments qui composent le système (voir figure 5.3) seront détaillés dans les sections suivantes.

5.2.2 Passage des CSP aux arbres

Ici, nous présentons la structure choisie pour une représentation formelle des CSP. Nous utilisons un compilateur afin de générer à partir d'un fichier texte, des éléments ou structures manipulables par un langage de programmation (C++, C, Pascal, ou autres). Ces éléments peuvent être par exemple des tableaux, des pointeurs, *etc.*

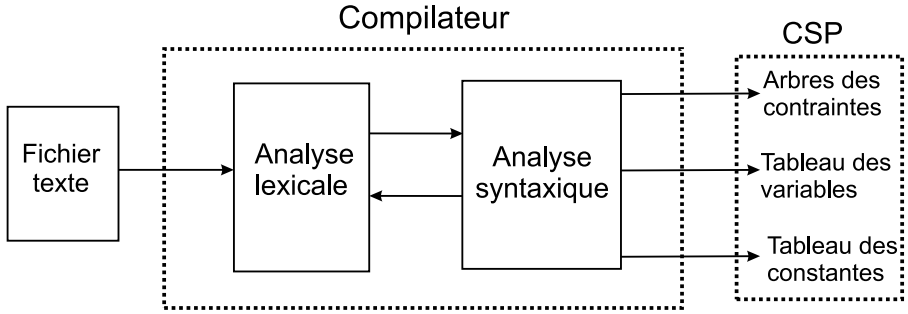

Figure 5.4 – Processus d'analyse du fichier d'entrée de Proj2D.

Comme le montre la figure 5.4, nous avons implémenté des automates afin d'effectuer une analyse lexicale et syntaxique des termes utilisés dans le fichier texte. Ces automates rendent possible la reconnaissance et l'interprétation des caractères d'un texte. Sur la programmation des automates, le lecteur

pourra consulter des ouvrages [Levine et al., 1992], [Aho et al., 1991] et [Appel and Ginsburg, 1998]. Le compilateur flex&bison[3] permet de traduire un langage d'automate programmable de type Lex et Yacc en code C. Grâce au code C généré, une lecture et une analyse du fichier texte nous donnent trois éléments qui caractérise le CSP (voir figure 5.4).

– **Arbres de contraintes :** cette structure est un tableau dont les éléments sont des arbres. Ces arbres représentent les expressions des différentes contraintes. Un arbre est un tableau constitué de *nœuds*. Chaque nœud correspond à un objet ou enregistrement composé de six champs comme indiqué dans la table ci-dessous.

Objet : Nœud		
	Champs	*Types*
1.	Code	Caractère
2.	Indice	Entier
3.	FD	Nœud
4.	FG	Nœud

Le champ **Code** correspond au symbole identificateur du nœud. Par exemple, pour une opération arithmétique, une fonction, une variable ou une constante, nous avons la table des équivalences ci-dessous.

	Types		*Code*
Opérations :	*Addition*	→	'+'
	Soustraction	→	'-'
	Multiplication	→	'*'
	Division	→	'/'
	...	...	...
Fonctions :	*Cosinus*	→	'C'
	Sinus	→	'S'
	Exponentiel	→	'E'
	Logarithme	→	'L'
	Tangente	→	'T'
	...	...	...
Autres :	*Variable*	→	'V'
	Constante	→	'N'
	...	...	...

Le champ **Indice** donne l'indice d'une variable dans le tableau des variables mais aussi d'une

[3] Librement et gratuitement disponible sur `http://gnuwin32.sourceforge.net`

constante dans le tableau des constantes. Les deux pointeurs de nœuds **FD** (Fils Droit) et **FG** (Fils Gauche) indiquent les références sur les nœuds descendants.

– **Tableau des variables :** Les variables ainsi que les renseignements les concernant sont stockées dans le tableau des variables. Ce tableau est constitué par des objets de type *Variables* dont les champs informent sur les caractéristiques des variables : le nom, le domaine d'appartenance, *etc.* Les champs ainsi que les types de ces objets sont détaillés à travers la table ci-dessous.

Objet : Variable		
	Champs	*Types*
1.	Nom	Chaîne de caractère
2.	Domaine	Intervalle
3.	Type	Entier

Le champ **Type** est un entier qui indique accessoirement certaines particularités, par exemple si la variable est définie comme variable intermédiaire, quantifiée (projetée) ou non quantifiée.

– **Tableau des constantes :** Ce tableau sert à stocker les constantes de type réel. L'emplacement de ces constantes est donné par le champ Indice, ceci pour les nœuds dont le Code est 'N' dans les arbres des contraintes.

Exemple 5.5. *Pour illustrer les différents éléments que nous venons de décrire, considérons le CSP suivant,*

$$\begin{cases} 1 - \exp(x.y) + x \in [0, 2] \\ (x, y) \in [-1, 3]^{\times 2}. \end{cases} \tag{5.16}$$

Un modèle sous forme d'arbre de ce CSP est représenté par la figure 5.5.

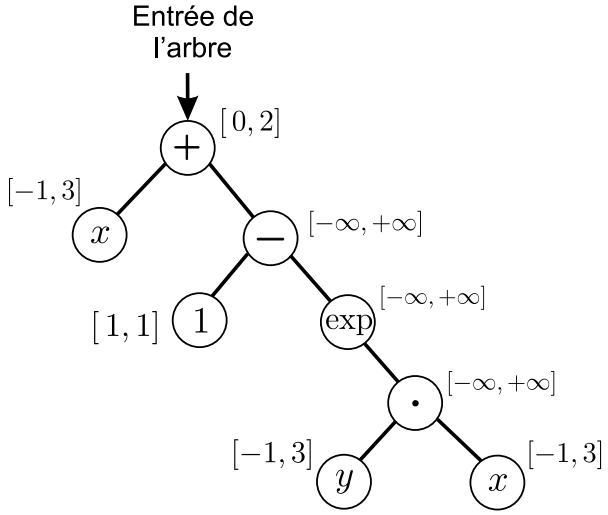

Figure 5.5 – Représentation sous forme d'arbre du CSP (5.16).

La figure 5.6 présente pour la contrainte (5.16), des structures exploitables par des langages de programmation. En effet, les figures 5.6a, 5.6b et 5.6c indiquent respectivement, l'arbre de contrainte, le tableau des variables et le tableau des contraintes. Sur la figure 5.6a, les champs FD et FG correspondent à l'emplacement dans le tableau des deux nœuds descendants. Ces champs sont à -1 si les nœuds descendants sont inexistants. Par exemple, les variables (Code='V') et les constantes (Code='N') n'ont aucune descendance, d'où FD=FG=-1. Le champ Indice indique pour les variables l'emplacement de celles-ci dans le tableau des variables, il en est de même pour les constantes. Le champ Indice est à -1 (la valeur par défaut) pour les nœuds associés à des opérations arithmétiques ou des fonctions. Cependant, une exception est faite pour le nœud racine (ou l'entrée) de l'arbre de contrainte. En effet, nous avons fait le choix d'associer la racine de l'arbre à une variable intermédiaire. La lecture de la structure se fait de la façon suivante. Le deuxième élément de l'arbre des contraintes est une variable car Code='V', le nom ainsi que le domaine d'appartenance de cette variable sont donnés par l'élément occupant l'emplacement Indice=0 du tableau des variables (Nom $= "x"$, Domaine $= [-1, 3]$, Type $= 1$).

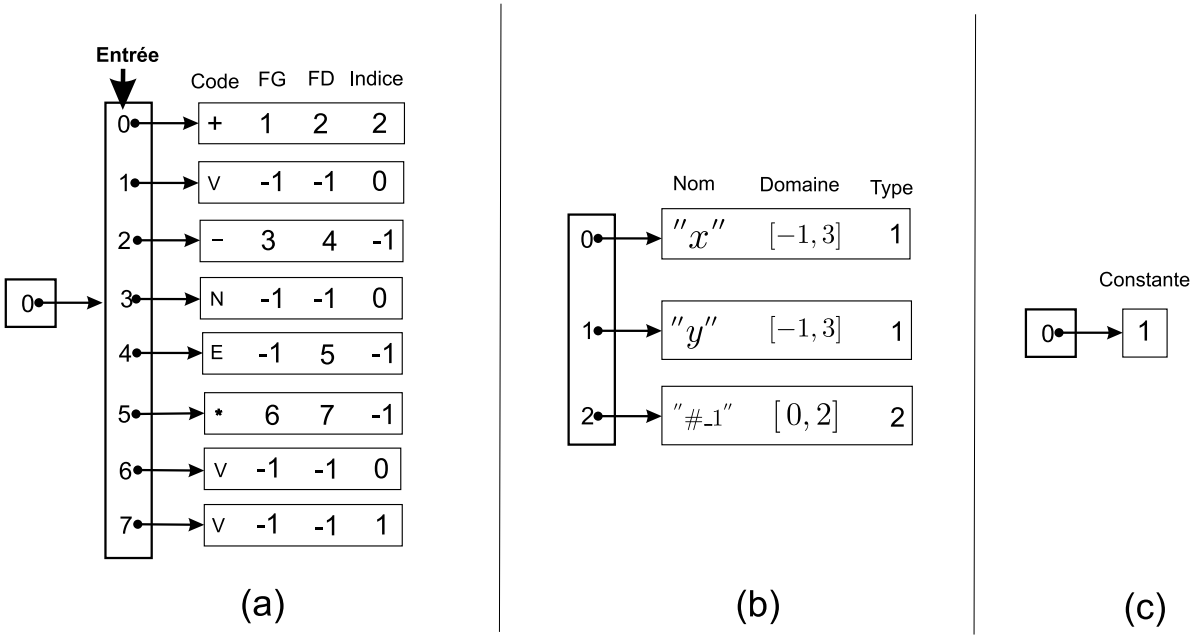

Figure 5.6 – Structure numérique du CSP (5.16); (a) : Arbre des contraintes; (b) : Tableau des variables; (c) : Tableau des constantes.

5.2.3 Gestion de multiples contraintes

Dans un cadre général, il est nécessaire de décomposer un problème de projection d'ensembles en intersections de plusieurs projections. Nous partons du principe qu'il est plus facile de résoudre n pro-

blèmes d'estimation à un paramètre que de résoudre un problème d'estimation à n paramètres. Soit l'ensemble

$$\mathbb{X} = \{\, \mathbf{p} | \, \exists \, \mathbf{q}, \, \mathbf{f}\,(\mathbf{p}, \mathbf{q}) \in [\mathbf{y}]\} \tag{5.17}$$

avec $\mathbf{p} \in \mathbb{R}^{n_p}$, $\mathbf{q} \in \mathbb{R}^{n_q}$, $[\mathbf{y}] \in \mathbb{R}^m$ et $\mathbf{f}$ une fonction définie de $\mathbb{R}^n$ dans $\mathbb{R}^m$ ($n = n_p + n_q$).

Remarque 5.6. *Toute fonction* $\mathbf{f}$ *est décomposable en plusieurs fonctions. Ainsi, pour* $i = 1, 2, ..., r$*, il existe des fonctions* $\mathbf{g}_i : \mathbb{R}^{n_p} \times \mathbb{R}^{n_{q_i}} \rightarrow \mathbb{R}^{m_i}$ *(*$n = \sum_{i=1}^{r} n_{q_i}$, $m = \sum_{i=1}^{r} m_i$*) telles que*

$$\mathbf{f}\,(\mathbf{p}, \mathbf{q}) \in [\mathbf{y}] \tag{5.18}$$

est équivalent à

$$\mathbf{g}_1\left(\mathbf{p}, \mathbf{q}^1\right) \in [\mathbf{z}]_1 \,, \; \mathbf{g}_2\left(\mathbf{p}, \mathbf{q}^2\right) \in [\mathbf{z}]_2 \,, ..., \; \mathbf{g}_r\left(\mathbf{p}, \mathbf{q}^r\right) \in [\mathbf{z}]_r \tag{5.19}$$

avec $\mathbf{q}^i \in \mathbb{R}^{n_{q_i}}$ *et* $[\mathbf{z}]_i \in \mathbb{IR}^{m_i}$.

De la remarque 5.6, nous déduisons,

$$\begin{aligned} \mathbb{X} \;\; = \;\; & \left\{ \mathbf{p} | \;\; \exists \, \mathbf{q}^1, \mathbf{g}_1\left(\mathbf{p}, \mathbf{q}^1\right) \in [\mathbf{z}]_1 \,, \; \exists \, \mathbf{q}^2, \; \mathbf{g}_2\left(\mathbf{p}, \mathbf{q}^2\right) \in [\mathbf{z}]_2 \,, \right. \\ & \left. ..., \; \exists \, \mathbf{q}^r, \; \mathbf{g}_r\left(\mathbf{p}, \mathbf{q}^r\right) \in [\mathbf{z}]_r \right\}. \end{aligned} \tag{5.20}$$

En décomposant l'expression (5.20) suivant les quantificateurs existentiels, il vient

$$\mathbb{X} = \bigcap_{i=1}^{r} \mathbb{X}_i = \mathbb{X}_1 \cap \mathbb{X}_2 \cap ... \cap \mathbb{X}_r \tag{5.21}$$

où

$$\mathbb{X}_i = \left\{ \mathbf{p} | \, \exists \, \mathbf{q}^i, \mathbf{g}_i\left(\mathbf{p}, \mathbf{q}^i\right) \in [\mathbf{z}]_i \right\}. \tag{5.22}$$

Dans la suite de cette section, nous utiliserons les notions de graphe de variables quantifiées et de connexité d'un graphe afin de déterminer les fonctions $\mathbf{g}_1$, $\mathbf{g}_2$, ..., $\mathbf{g}_r$, ceci pour toutes fonctions vectorielles $\mathbf{f}$. Nous commencerons par modéliser les contraintes sous forme de graphes de variables quantifiées, puis nous procéderons à la décomposition en *composantes connexes* de ces contraintes.

5.2.3.1 Graphe des variables quantifiées

Le graphe des variables quantifiées est défini par $\mathcal{G} = (\mathcal{Q}, \mathcal{C})$. Les éléments de $\mathcal{Q}$ sont les *sommets* du graphe $\mathcal{G}$, ceux de $\mathcal{C}$ sont les *arcs* ou encore les *arrêtes* de $\mathcal{G}$, selon que le graphe est orienté ou non. Nous avons,

$$\begin{aligned} & \mathcal{Q} : \left\{ q_1, q_2, ..., q_{n_q} \right\} \\ & \mathcal{C} : \bigcup_{i<j} \{(q_i, q_j)\} \end{aligned} \tag{5.23}$$

où $\mathcal{Q}$ représente l'ensemble des variables quantifiées et $\mathcal{C}$ les couples de variables qui interviennent dans une même contrainte. En effet, chaque expression $f_k(\mathbf{p},\mathbf{q}) \in [y_k]$, $k = 1, 2, ..., m$, est représentée par une contrainte c_k qui regroupe toutes les variables q_i ($i \in \{1, 2, ..., n_q\}$) de f_k. Le couple $(q_i, q_j) \in \mathcal{C}$ ($i < j$) indique qu'il existe une contrainte c_k telle que q_i et $q_j \in c_k$. Ici nous considérons que $\mathcal{G}$ est un graphe non orienté.

Exemple 5.7. *Considérons 4 contraintes définies par,*

$$\begin{cases} \textit{(1) :}\ 2p_1 + q_2 \sin(q_1.p_2) \in [-1, 2]\ ; \\ \textit{(2) :}\ p_2 + q_1.q_3 \in [0, 1]\ ; \\ \textit{(3) :}\ q_2 - q_3 \in [1, 2]\ ; \\ \textit{(4) :}\ p_1 - q_4 \in \left[-1, -\frac{1}{2}\right]. \end{cases} \tag{5.24}$$

le vecteur $(p_1, p_2, q_1, q_2, q_3, q_4) \in \mathbb{R}^6$*. Le graphe se note* $\mathcal{G} = (\mathcal{Q}, \mathcal{C})$ *avec*

$$\mathcal{Q} : \{q_1, q_2, q_3, q_4\} \tag{5.25}$$

et

$$\mathcal{C} : \{(q_1, q_2), (q_1, q_3), (q_2, q_3)\}, \tag{5.26}$$

- q_1 et q_2 apparaîsent dans la contrainte (1), ceci correspond à la liaison (q_1, q_2);
- q_1 et q_3 apparaîssent dans la contrainte (2), ceci correspond à la liaison (q_1, q_3);
- q_2 et q_3 apparaîssent dans la contrainte (3), ceci correspond à la liaison (q_2, q_3).

La figure 5.7 représente les nœuds et les arrêtes de $\mathcal{G}$.

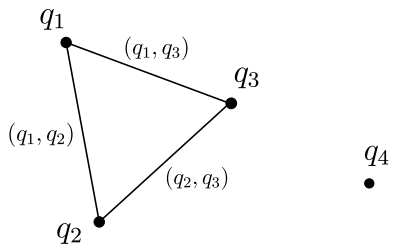

Figure 5.7 – Graphe $\mathcal{G}$.

5.2.3.2 Décomposition en composantes connexes

Définition 5.8 (graphe connexe). Un graphe $\mathcal{G}$ est dit connexe si pour tout couple de variables quantifiées (q_i, q_j) tel que $i, j \in \{1, 2, ..., n_q\}$, il existe une chaîne qui relie q_i à q_j.

Définition 5.9 (Groupe). Un *groupe* ou composante connexe est défini par un sous-graphe connexe de variables quantifiées. Ces composantes $\mathcal{G}_i$, avec $i \in \{1, ..., r\}$ où r est le nombre de composantes connexes, définissent pour une fonction vectorielle $\mathbf{f}$, les fonctions scindées $\mathbf{g}_1$, $\mathbf{g}_2$, ..., $\mathbf{g}_r$ (voir figure 5.8).

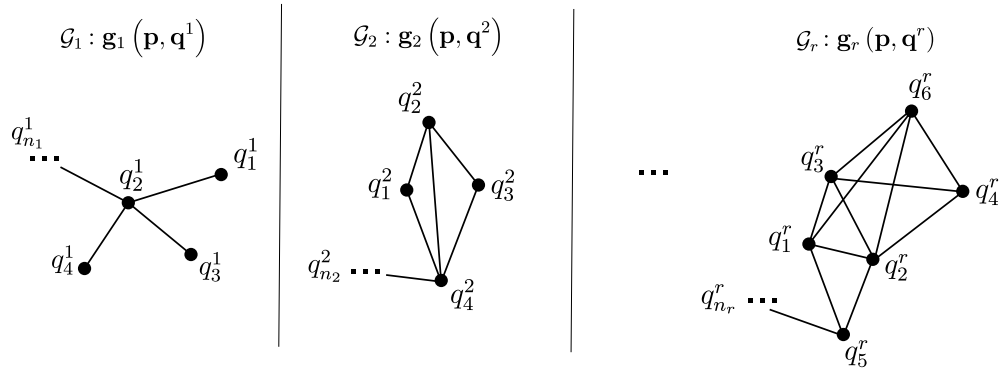

Figure 5.8 – Composantes connexes $\mathcal{G}_1$, $\mathcal{G}_2$, ..., $\mathcal{G}_r$ du graphe $\mathcal{G}$.

Dans l'exemple 5.7, le graphe est composé de deux groupes ou composantes connexes donnés par $G_1 = \{q_1, q_2, q_3\}$ et $G_2 = \{q_4\}$.

Définition 5.10 (Matrice d'adjacence). Représentation matricielle des différents nœuds ainsi que les différentes arrêtes d'un graphe (orienté ou non). Cette matrice constitue un moyen de stockage efficace d'un graphe au point de vue place mémoire. La matrice d'adjacence d'un graphe $\mathcal{G}$ est donnée par,

$$\mathbf{M}(\mathcal{G}) = \begin{array}{c} \\ q_1 \\ \vdots \\ q_{n_q} \end{array} \begin{array}{c} \begin{array}{ccc} q_1 & \cdots & q_{n_q} \end{array} \\ \begin{pmatrix} m_{1,1} & \cdots & m_{1,n_q} \\ \vdots & \ddots & \vdots \\ m_{n_q,1} & \cdots & m_{n_q,n_q} \end{pmatrix} \end{array}. \tag{5.27}$$

Pour ne pas alourdir les notations, nous remplacerons $\mathbf{M}(\mathcal{G})$ par $\mathbf{M}$. Les coefficients $m_{i,j}$, avec i et $j \leq n_q$, indiquent si la variable q_i est liée ou non par une ou plusieurs contraintes à q_j. Nous en déduisons que :

$$m_{i,j} = \begin{cases} 1 \text{ si } (q_i, q_j) \in \mathcal{C} \\ 0 \text{ sinon} \end{cases} \tag{5.28}$$

Proposition 5.11. *Le graphe $\mathcal{G}$ est non orienté d'où,*

$$m_{i,j} = m_{j,i},\ \forall i, j \in \{1, ..., n_q\}. \tag{5.29}$$

Selon l'égalité (5.29), la matrice $\mathbf{M}$ est symétrique. Dans nos manipulations algorithmiques, cette propriété nous amène à tenir compte uniquement de la partie inférieure ou supérieure à la diagonale de $\mathbf{M}$. Ceci facilite la gestion en mémoire et permet un gain de temps non négligeable lorsque n_q est grand.

Pour déterminer les composantes connexes du graphe des variables, nous considérons la *fermeture transitive* (voir [Gondran and Minoux, 1985]). Nous avons alors,

$$\mathbf{M}^* = \sum_{i=0}^{n_q} \mathbf{M}^i = \mathbf{I}_{n_q} + \mathbf{M} + \mathbf{M}^2 + ... + \mathbf{M}^{n_q}. \tag{5.30}$$

Les différentes composantes connexes du graphe sont données par les variables q_i ($i \leq n_q$) qui constituent les lignes linéairement indépendantes de $\mathbf{M}^*$. Lorsque n_q devient grand (≥ 10) le calcul de la matrice $\mathbf{M}^*$ devient très vite inenvisageable. Nous utilisons alors une méthode qui explore en profondeur les nœuds du graphe. Nous partons d'un nœud initial appelé *sommet.* Chaque voisin[4] de ce sommet ainsi que les voisins des voisins sont explorés puis ajoutés à la liste de la composante connexe. Chaque nœud exploré est marqué afin d'éviter toutes explorations redondantes. Ce procédé est réitéré pour tous les nœuds du graphe qui n'ont pas été pris en compte dans la liste. Ainsi, nous obtenons avec une complexité en $O\left(n_q^2\right)$, les groupes d'un graphe à partir de sa matrice d'adjacence.

Notre algorithme de recherche de composantes connexes dans un graphe est composé de deux sous algorithmes. Le premier nommé `Groups`, détermine à partir de la matrice d'adjacence $\mathbf{M}$ l'ensemble des groupes ou composantes connexes du graphe. Le second algorithme appelé `Conex_Component`, recherche à partir d'un nœud ou sommet q_i l'ensemble des nœuds qui appartiennent au même groupe. En guise de variables globales, nous disposons de listes d'entiers $\mathcal{L}_1$, $\mathcal{L}_2$, ..., $\mathcal{L}_{n_q}$. Ces listes servent à stocker les nœuds ou variables $\{q_i\}_{i=1,2,...,n_q}$ qui appartiennent aux différents groupes du graphe. Un tableau de booléens (`Mark`) indique le marquage pour chaque variable.

Algorithme : `Groups(Entrée : M)`	
1	`Pour i = 1, 2, ..., n`$_q$ `faire :`
2	`Mark[i]←faux;` $\mathcal{L}_i \leftarrow \emptyset$`;`
3	`Fin pour`
4	`k ← 1;`
5	`Tant que (k ≤ n`$_q$`) faire :`
6	`Si (Mark[k]=faux) :`
7	`Conex_Component(M,k,k);`
8	`Fin si`
9	`k ←k +1;`
10	`Fin tant que`
11	`Fin.`

[4]Un voisin d'un sommet est un nœud pour lequel il existe une arrête (ou liaison) entre ce nœud et le sommet.

Algorithme : Conex_Component(Entrées : M,k,s)	
1	Si (Mark[k]=faux) :
2	Mark[k]←vrai; $\mathcal{L}_s \leftarrow \mathcal{L}_s \cup$ {k};
3	i ← k+1;
4	Tant que (i ≤ n_q) faire :
5	Si (M[k,i]=1) :
6	Conex_Component(M,i,s);
7	Fin si(**5**)
8	i← i+1;
9	Fin tant que
10	Fin si(**1**)
11	Fin.

5.2.4 Algorithme de projection pour les groupes (SPVIA)

Considérons un ensemble $\mathbb{X}$ défini par (5.17). Une catégorie d'algorithmes permettant de caractériser $\mathbb{X} = \text{Proj}_{\mathbf{p}}(\mathbb{S})$ est présentée dans [Braems, 2002] et [Ratschan, 2002]. Rappelons que la caractérisation de $\mathbb{X}$ revient à déterminer les deux sous-pavages $\underline{\mathbb{X}}$ et $\overline{\mathbb{X}}$ tels que

$$\underline{\mathbb{X}} \subset \mathbb{X} \subset \overline{\mathbb{X}}, \tag{5.31}$$

où $\underline{\mathbb{X}}$ représente un ensemble de pavés acceptables (approximation intérieure) et $\overline{\mathbb{X}} = \underline{\mathbb{X}} \cup \partial\mathbb{X}$ représente une approximation extérieure de $\mathbb{X}$.

Bien sûr, il convient d'utiliser pour chaque problème, la décomposition en composantes connexes des différentes contraintes. Cette procédure permet d'améliorer l'efficacité de ces algorithmes (voir section 5.2.3). Ici, nous avons choisi de nommer de tels algorithmes : SPVIA (***S**et **P**rojection **V**ia **I**nterval **A**nalysis*). Ces algorithmes s'inscrivent dans la classe des méthodes de caractérisation d'ensembles de type branch and bound basées sur le calcul par intervalles.

Au vu de la relation (5.21), notons les ensembles $\mathbb{S}_i$ ($i = 1, ..., r$) tels que :

$$\mathbb{X}_i = \text{Proj}_{\mathbf{p}}(\mathbb{S}_i). \tag{5.32}$$

Afin de réaliser notre algorithme de projection d'ensembles, nous disposons globalement de $2r$ contracteurs : $\mathcal{C}_{\mathbb{S}_1}$, $\mathcal{C}_{\mathbb{S}_2}$, ..., $\mathcal{C}_{\mathbb{S}_r}$ et $\mathcal{C}_{\mathbb{S}_1^c}$, $\mathcal{C}_{\mathbb{S}_2^c}$, ..., $\mathcal{C}_{\mathbb{S}_r^c}$ associés respectivement aux ensembles $\mathbb{S}_1$, $\mathbb{S}_2$, ..., $\mathbb{S}_r$ et à leurs complémentaires. Nous avons aussi deux listes de pavés $\mathcal{L}_1$ et $\mathcal{L}_2$, deux opérateurs de bissection $L(.)$ et $R(.)$ et un pavé initial $[\mathbf{p}]_0 \times [\mathbf{q}]_0$. La liste $\mathcal{L}_1$ sera chargée de stocker les pavés acceptables, tandis que $\mathcal{L}_2$ stockera les pavés incertains pour $\mathbb{X}$.

L'algorithme SPVIA se compose des routines SPVIA1 et SPVIA2 dont les détails sont donnés ci-dessous. Ces routines sont établies selon les principes abordées dans la section 5.1.2. Le premier algorithme SPVIA1 met en œuvre un processus de partitionnement sur le pavé $[\mathbf{p}]_0$ et fait appel à SPVIA2. Pour chaque partition générée sur $[\mathbf{p}]_0$ et grâce à un autre partitionnement cette fois sur le pavé $[\mathbf{q}]_0$, SPVIA2 sert d'une part de contracteur de haut niveau[5]. D'autre part, ce même algorithme permet de trouver les points $\mathbf{q}_0^i \in [\mathbf{q}^i]$ tels que : $\mathcal{C}_{\mathbb{X}_i^c}([\mathbf{p}], \mathbf{q}_0^i) \neq \emptyset$ avec $i = 1, 2, ..., r$. Pour cela, les centres des partitions $[\mathbf{q}^i]$ sont systématiquement testés, c'est-à-dire : $\mathbf{q}_0^i = m([\mathbf{q}^i])$.

Algorithme : SPVIA1(Entrées : $[\mathbf{p}]_0$, $[\mathbf{q}]_0$)	
1	$\mathcal{L}_1 \leftarrow \mathcal{L}_2 \leftarrow \mathcal{L} \leftarrow \emptyset$;
2	$\mathcal{L} \leftarrow \mathcal{L} \cup \{([\mathbf{p}]_0, [\mathbf{q}^1]_0, [\mathbf{q}^2]_0, ..., [\mathbf{q}^r]_0)\}$;
3	Tant que ($\mathcal{L} \neq \emptyset$) faire :
4	$([\mathbf{p}], [\mathbf{q}^1], [\mathbf{q}^2], ..., [\mathbf{q}^r]) \leftarrow \text{pop}(\mathcal{L})$;
5	i $\leftarrow$ 1; k $\leftarrow$ 1;
6	Tant que (i $\leq$ r) faire :
7	SPVIA2(i,k,$[\mathbf{p}]$,$[\mathbf{q}^{\text{i}}]$);
8	i $\leftarrow$ i+1;
9	Fin tant que (**6**)
10	Si (k = r) : $\mathcal{L}_2 \leftarrow \mathcal{L}_2 \cup \{[\mathbf{p}]\}$;
11	Sinon :
12	Si $(w([\mathbf{p}]) < \varepsilon)$ alors : $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup \{[\mathbf{p}]\}$;
13	Sinon : $\mathcal{L} \leftarrow \mathcal{L} \cup \{(L[\mathbf{p}], [\mathbf{q}^1], [\mathbf{q}^2], ..., [\mathbf{q}^r])\}$;
14	$\mathcal{L} \leftarrow \mathcal{L} \cup \{(R[\mathbf{p}], [\mathbf{q}^1], [\mathbf{q}^2], ..., [\mathbf{q}^r])\}$;
15	Fin sinon (**13**), fin si (**12**)
16	Fin sinon (**11**), fin si (**10**)
17	Fin tant que (**3**)
18	Fin.

[5] Contracteur associée à la consistance globale.

	Algorithme : SPVIA2(Entrées - Sorties : i, k, $[\mathbf{p}]$, $[\mathbf{q}]$)
1	$\mathcal{P} \leftarrow \emptyset$; $\mathcal{P} \leftarrow \{([\mathbf{p}], [\mathbf{q}])\} \cup \mathcal{P}$;
2	$[\mathbf{p}]_{\mathtt{old}} \leftarrow [\mathbf{p}]$; $[\mathbf{p}] \leftarrow [\mathbf{q}] \leftarrow [\mathbf{p}]_1 \leftarrow [\mathbf{q}]_1 \leftarrow \emptyset$;
3	Tant que ($\mathcal{P} \neq \emptyset$) faire :
4	$([\mathbf{p}]_1, [\mathbf{q}]_1) \leftarrow \mathtt{pop}(\mathcal{P})$;
5	$([\mathbf{p}]_1, [\mathbf{q}]_1) \leftarrow \mathcal{C}_{\mathbb{S}_\mathtt{i}}([\mathbf{p}]_1, [\mathbf{q}]_1)$;
6	Si ($[\mathbf{p}]_1 \times [\mathbf{q}]_1 \neq \emptyset$) alors :
7	Si ($\mathcal{C}_{\mathbb{S}_\mathtt{i}^c}([\mathbf{p}_{\mathtt{old}}], m([\mathbf{q}]_1)) = \emptyset$) alors :
8	$[\mathbf{p}] \leftarrow [[\mathbf{p}] \cup [\mathbf{p}_{\mathtt{old}}]]$; $[\mathbf{q}] \leftarrow [[\mathbf{q}] \cup [\mathbf{q}]_1]$;
9	k $\leftarrow$ k+1;
10	Tant que ($\mathcal{P} \neq \emptyset$) faire :
11	$([\mathbf{p}]_1, [\mathbf{q}]_1) \leftarrow \mathtt{pop}(\mathcal{P})$;
12	$[\mathbf{q}] \leftarrow [[\mathbf{q}] \cup [\mathbf{q}]_1]$;
13	Fin tant que (**10**)
14	Sinon :
15	Si ($w([\mathbf{q}]_1) < \alpha w([\mathbf{p}]_1)$) alors :
16	Tant que ($\mathcal{P} \neq \emptyset$) faire :
17	$([\mathbf{p}]_1, [\mathbf{q}]_1) \leftarrow \mathtt{pop}(\mathcal{P})$;
18	$[\mathbf{p}] \leftarrow [[\mathbf{p}] \cup [\mathbf{p}]_1]$; $[\mathbf{q}] \leftarrow [[\mathbf{q}] \cup [\mathbf{q}]_1]$;
19	Fin tant que (**16**)
20	Sinon : $\mathcal{P} \leftarrow \{([\mathbf{p}]_1, L[\mathbf{q}]_1)\} \cup \{([\mathbf{p}]_1, R[\mathbf{q}]_1)\} \cup \mathcal{P}$;
21	Fin sinon (**20**), fin si (**15**)
22	Fin sinon (**14**), fin si (**7**)
23	Fin si (**6**)
24	Fin tant que (**3**)
25	Fin.

Les paramètres ε et α servent respectivement à limiter la profondeur de la bissection pour $[\mathbf{p}]$ et à adapter celle de $[\mathbf{q}]$. Dans une certaine mesure, ε représente la taille maximale des pavés qui constituent le sous-pavage incertain. Ces paramètres sont fixés arbitrairement en fonction de la précision désirée sur la caractérisation de la projection.

La pile $\mathcal{P}$ dans SPVIA2, permet de stocker et de traiter les pavés issus des partitionnements sur $[\mathbf{q}]$. L'utilisation d'une pile au lieu d'une file permet de réduire considérablement le nombre de bissections sur $[\mathbf{q}]$. En effet, pour démontrer *l'acceptabilité* d'un pavé, la recherche des points solutions $\mathbf{q}_0^i$ ($i = 1, 2, ..., n_q$) est effectué uniquement dans une zone locale du pavé $[\mathbf{q}]$. Ceci traduit le fait que les plus petits pavés sont placés en tête de pile puis récupérés afin de tester leurs centres. Dans le pire des cas

(lorsqu'on ne trouve aucun point solution), l'algorithme SPVIA2 atteint son point d'arrêt dès que la taille du pavé situé en tête de pile est inférieure à $\alpha \cdot w([\mathbf{p}])$.

En utilisant une structure de file d'attente $\mathcal{F}$ à la place de la pile $\mathcal{P}$ (voir l'algorithme SPVIA2Bis ci-dessous), le point d'arrêt de SPVIA2 est atteint lorsque tous les pavés de la file sont de tailles plus petites que $\alpha \cdot w([\mathbf{p}])$. Ceci a bien entendu pour conséquence d'augmenter le nombre de bissections sur $[\mathbf{q}]$ et par la même occasion le temps de calcul.

Algorithme : SPVIA2Bis(Entrées - Sorties : i, k, $[\mathbf{p}]$, $[\mathbf{q}]$)

```
1  F ← ∅;  F ← F ∪ {([p], [q])};
2  [p_old] ← [p];  [p] ← [q] ← [p]_1 ← [q]_1 ← ∅;
3  Tant que (F ≠ ∅) faire :
4        ([p]_1, [q]_1) ← pop(F);
5        ([p]_1, [q]_1) ← C_{S_i}([p]_1, [q]_1);
6        Si ([p]_1 × [q]_1 ≠ ∅) alors :
7            Si (C_{S_i^c}([p_old], m([q]_1)) = ∅) alors :
8                [p] ← [[p] ∪ [p_old]]; [q] ← [[q] ∪ [q]_1];
9                k ← k+1;
10               Tant que (F ≠ ∅) faire :
11                     ([p]_1, [q]_1) ← pop(F);
12                     [q] ← [[q] ∪ [q]_1];
13               Fin tant que (10)
14           Sinon :
15                 Si (w([q]_1) > αw([p]_1)) alors :
16                     F ← F ∪ {([p]_1, L[q]_1)} ∪ {([p]_1, R[q]_1)};
17                 Sinon :
18                     [p] ← [[p] ∪ [p]_1]; [q] ← [[q] ∪ [q]_1];
19                 Fin sinon (17), fin si (15)
20           Fin sinon (14), fin si (7)
21       Fin si (6)
22 Fin tant que (3)
23 Fin.
```

Il n'est souvent pas très simple de comparer les deux versions de SPVIA2. En effet, l'efficacité de l'algorithme SPVIA2 dépend non seulement des objectifs à atteindre mais aussi des problèmes traités. Par exemple, dans les problèmes pour lesquels une approximation intérieure (la recherche des pavés acceptables) est d'une importance capitale, nous aurons plutôt intérêt à choisir une file comme structure de stockage et d'analyse des pavés intermédiaires. D'un autre côté, pour les problèmes où l'on s'intéresse

à une approximation extérieure ou globale de la projection, nous gagnerons un temps de calcul non négligeable en utilisant une pile dans SPVIA2.

5.2.5 Avantage de la décomposition en groupes

Notons que dans le pire des cas, la complexité de SPVIA1 reste exponentielle quelle que soit la structure de stockage de pavés utilisée dans l'algorithme SPVIA2. Cependant la décomposition de contraintes en composantes connexes (voir la remarque 5.6) apporte dans le cas général, une notable amélioration sur la complexité de SPVIA1. Afin de le montrer, nous étudions la complexité des algorithmes de projection suivant deux cas.

5.2.5.1 Sans la décomposition en groupes de contraintes

Rappelons que SPVIA1 permet de caractériser l'ensemble

$$\mathbb{X} = \text{Proj}_{\mathbf{p}}(\mathbb{S}) = \{\mathbf{p} \in [\mathbf{p}] | \, \exists \mathbf{q} \in [\mathbf{q}] \, , \, \mathbf{f}(\mathbf{p}, \mathbf{q}) = \mathbf{0}\} \, . \tag{5.33}$$

Ici, la complexité de SPVIA1, dans le pire des cas, est donnée par

$$C^{\text{SDG}}_{\text{SPVIA1}} \simeq K \left(\frac{w([\mathbf{p}])}{\varepsilon} \right)^{n_p} \cdot \left(\frac{w([\mathbf{q}])}{\varepsilon} \right)^{n_q} \tag{5.34}$$

où n_p et n_q représentent respectivement les dimensions des pavés $[\mathbf{p}]$ et $[\mathbf{q}]$, K correspond à une constante de proportionnalité et ε définie la taille maximale d'un pavé incertain (cœfficient de précision du sous-pavage).

5.2.5.2 Avec la décomposition en groupes

Nous avons selon l'expression (5.17),

$$\mathbb{X} = \bigcap_{i=1}^{r} \mathbb{X}_i \tag{5.35}$$

avec

$$\mathbb{X}_i = \left\{ \mathbf{p} \in [\mathbf{p}] | \, \exists \mathbf{q}^i \in \left[\mathbf{q}^i\right] , \mathbf{g}_i\left(\mathbf{p}, \mathbf{q}^i\right) = \mathbf{0} \right\} . \tag{5.36}$$

et r le nombre de groupes ou composantes connexes constituées. Dans ce cas, la complexité de SPVIA1 s'écrit

$$C^{\text{ADG}}_{\text{SPVIA1}} \approx K \left(\frac{w([\mathbf{p}])}{\varepsilon} \right)^{n_p} \cdot \sum_{i=1}^{r} \left(\frac{w\left(\left[\mathbf{q}^i\right]\right)}{\varepsilon} \right)^{n_{q_i}} , \tag{5.37}$$

avec n_{q_i} ($n_q = \sum_{i=1}^{r} n_{q_i}$) qui correspond au nombre de paramètres non quantifiées dans le $ième$ groupe de contraintes. Les paramètres n_p, n_q, K et ε sont définis de la même façon que dans le premier cas.

Ainsi, nous établissons la relation qui justifie l'efficacité de la décomposition en composantes connexes. Par conséquent, nous avons

$$C^{\texttt{ADG}}_{\texttt{SPVIA1}} \leq C^{\texttt{SDG}}_{\texttt{SPVIA1}}. \tag{5.38}$$

5.2.6 Vers une meilleure approximation de $\underline{\mathbb{X}}$

Dans cette section, nous proposerons une méthode qui améliore l'efficacité de l'approximation intérieure, c'est-à-dire réussir à obtenir dans les mêmes temps un ensemble $\underline{\mathbb{X}}$ plus volumineux.

Rappelons que pour prouver qu'un pavé $[\mathbf{p}]$ est inclus dans $\mathbb{X}$, défini selon (5.17), il suffit de trouver un vecteur $\mathbf{q}_0 \in [\mathbf{q}]$ tel que $\mathcal{C}_{\mathbb{S}^c}([\mathbf{p}], \mathbf{q}_0) = \emptyset$, avec $\mathbb{S}$ donné par (5.1), nous disons alors que $[\mathbf{p}]$ est acceptable. Par conséquent, nous avons élaboré dans un premier temps, une méthode de recherche locale pour déterminer un vecteur $\mathbf{q}_0$. Dans un deuxième temps, nous avons mis en place une procédure afin de trouver dans un pavé incertain, une partie acceptable.

5.2.6.1 Algorithme de recherche locale (`LSBIC`)

La méthode de recherche locale est basée sur une recherche localisée dans des domaines sélectionnés par critère de *proximité*. La technique fonctionne de la façon suivante. L'espace de recherche $[\mathbf{q}]$ est partitionné par bissection. Les centres des pavés obtenus sont testés pour montrer que

$$\mathcal{C}_{\mathbb{S}^c}([\mathbf{p}], m(L[\mathbf{q}])) = \emptyset \text{ ou } \mathcal{C}_{\mathbb{S}^c}([\mathbf{p}], m(R[\mathbf{q}])) = \emptyset$$

avec $L[\mathbf{q}]$ et $R[\mathbf{q}]$ qui sont des parties de $[\mathbf{q}]$ telles que, le pavé $[\mathbf{q}] = L[\mathbf{q}] \cup R[\mathbf{q}]$ et les volumes $\text{Vol}(L[\mathbf{q}]) = \text{Vol}(R[\mathbf{q}]) = \frac{\text{Vol}([\mathbf{q}])}{2}$. Si aucun des centres ne convient, nous réitérons le procédé sur un seul des deux pavés. Le critère pour le choix du pavé constitue une des clés de la technique. Ainsi, nous définissons un critère de proximité entre deux pavés $[\mathbf{u}]$ et $[\mathbf{v}] \in \mathbb{IR}^{n_v}$ par,

$$j_{\text{p}}([\mathbf{u}], [\mathbf{v}]) \triangleq \max_{i=1,2,\ldots,n_v} \left(j_{\text{p}}([u_i], [v_i])\right). \tag{5.39}$$

avec $j_{\text{p}}([u_i], [v_i]) = \max(\underline{u}_i - \underline{v}_i, \overline{v}_i - \overline{u}_i)$. Grâce à ce critère, nous avons principalement trois situations possibles pour les pavés $[\mathbf{u}]$ et $[\mathbf{v}]$ (voir figure 5.9) :

- **Cas (a)** : $[\mathbf{v}] \subset [\mathbf{u}] \Rightarrow j_{\text{p}}([\mathbf{u}], [\mathbf{v}]) < 0$;
- **Cas (b)** : $[\mathbf{v}] \subset [\mathbf{u}] \Rightarrow j_{\text{p}}([\mathbf{u}], [\mathbf{v}]) = 0$;
- **Cas (c)** et **(d)** : $[\mathbf{v}]$ n'est pas inclus dans $[\mathbf{u}]$, d'où $j_{\text{p}}([\mathbf{u}], [\mathbf{v}]) > 0$.

Dans le cas d'ensembles définis par des inégalités du type $\mathbf{f}(\mathbf{p}, \mathbf{q}) \in [\mathbf{y}]$, avec $\mathbf{p} \in [\mathbf{p}]$ et $\mathbf{q} \in [\mathbf{q}]$, nous sélectionnons après chaque bissection le pavé dont l'image par $\mathbf{f}$ est la plus "proche" (au sens d'un critère de proximité plus petit) du pavé $[\mathbf{y}]$. Les détails de la méthode de recherche sont donnés par l'algorithme `LSBIC` (*Local Search By Interval Computation*). Il existe deux principaux critères d'arrêts

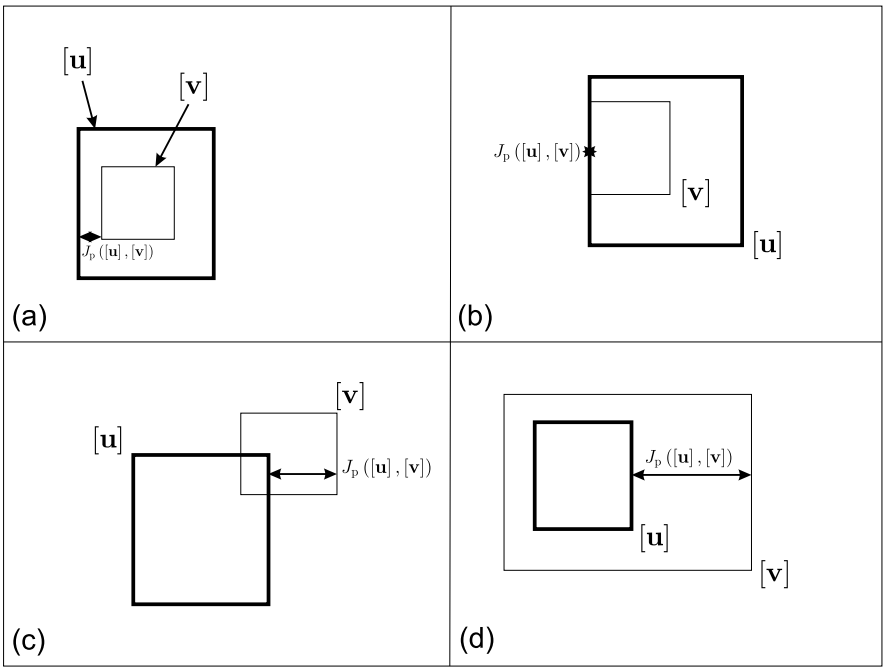

Figure 5.9 – Critère de proximité pour les pavés $[\mathbf{u}]$ et $[\mathbf{v}]$ dans différentes situations.

pour `LSBIC`, si d'une part le centre d'une partition correspond à l'une des solutions recherchées et si d'autre part, le dernier pavé de recherche est jugé trop petit ($w([\mathbf{q}]) > \varepsilon$).

Algorithme : `LSBIC`(`Entrées:` $[\mathbf{p}]$, $[\mathbf{q}]$; `Sortie:` $\mathbf{q}_0$)	
1	`Tant que` $(w([\mathbf{q}]) > \varepsilon)$ `faire`
2	$[\mathbf{y}]_1 \leftarrow [\mathbf{f}]([\mathbf{p}], m(L[\mathbf{q}]))$;
3	$[\mathbf{y}]_2 \leftarrow [\mathbf{f}]([\mathbf{p}], m(R[\mathbf{q}]))$;
4	`Si` $\left(j_p([\mathbf{y}],[\mathbf{y}]_1) < j_p([\mathbf{y}],[\mathbf{y}]_2)\right)$:
5	$[\mathbf{q}] \leftarrow L[\mathbf{q}]$;
6	`Sinon :` $[\mathbf{q}] \leftarrow R[\mathbf{q}]$;
7	`Fin tant que`
8	`Si` $(\mathcal{C}_{\mathbb{X}^c}([\mathbf{p}],[\mathbf{q}]) = \emptyset)$ `: renvoyer` $m([\mathbf{q}])$;
9	`Sinon : renvoyer` $\emptyset$;
10	`Fin.`

Remarque 5.12. *La méthode `LSBIC` ne converge pas nécessairement vers une solution même lorsque celle-ci existe, cela est dû à un critère de proximité qui ne garantit pas la convergence de l'algorithme. Il existe des situations particulières où le critère de sélection conduit à une "impasse". Nous reviendrons sur cette remarque dans le chapitre 9.*

Retenons globalement que la méthode a l'avantage, d'une part d'être de complexité polynomiale,

$$C_{\texttt{LSBIC}}(n) \approx n \cdot \ln\left(\frac{w([\mathbf{p}] \times [\mathbf{q}])}{\varepsilon}\right). \tag{5.40}$$

D'autre part, il est possible d'utiliser la méthode pour l'estimation de paramètres dans des domaines de recherche continus. A ce sujet, nous traiterons dans le chapitre 9, une application de ces résultats à la conception de filtres numériques et analogiques.

5.2.6.2 Approximation intérieure (`IAVIC`)

En ce qui concerne la méthode d'approximation intérieure `IAVIC` (*Inner Approximation Via Interval Computation*). Comme nous l'avons déjà expliqué, l'approche vise à déterminer un pavé acceptable $[\mathbf{p}]_{in}$dans un pavé incertain $[\mathbf{p}]$. Le principe utilisé est décrit comme suit. Nous commençons par vérifier *l'acceptabilité* des faces de $[\mathbf{p}]$, pour cela nous pourrons mettre à profit l'algorithme `LSBIC`. Dès que nous trouvons une face acceptable pour $[\mathbf{p}]$, l'épaisseur de la face est augmentée de façon progressive. L'objectif est de rechercher un grand pavé acceptable à partir de la face choisie (voir les illustrations sur les figures 5.10a et 5.10b).

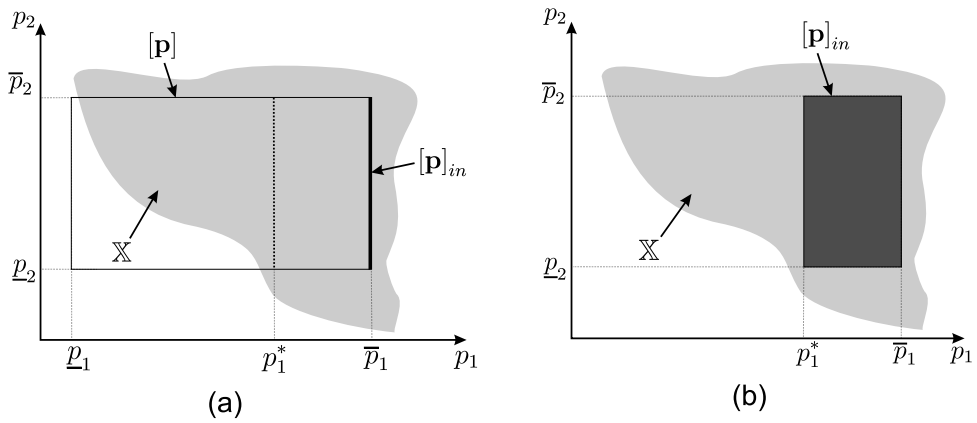

Figure 5.10 – Approximation intérieure.

L'algorithme `IAVIC` admet en entrées : les domaines d'appartenance $[\mathbf{p}]$ et $[\mathbf{q}]$ des paramètres $\mathbf{p}$ et $\mathbf{q}$, un indice i qui désigne l'axe de $[\mathbf{p}]$ orthogonal à la face considérée pour l'approximation intérieure. En sortie, nous avons une liste $\mathcal{L}_{in}$ qui contient des parties acceptables de $[\mathbf{p}]$. Nous avons conçu un algorithme `IAVIC` qui utilise deux routines, `IAVIC_L` et `IAVIC_R` qui déterminent les pavés acceptables

à partir de faces situées respectivement aux extrémités gauche et droite de l'axe i de $[\mathbf{p}]$.

Algorithme : IAVIC(Entrées : i, $[\mathbf{p}]$, $[\mathbf{q}]$; Sortie : $\mathcal{L}_{in}$)	
1	$\mathcal{L}_{in} \leftarrow$ IAVIC_L $([\mathbf{p}], [\mathbf{q}]) \cup$ IAVIC_R $([\mathbf{p}], [\mathbf{q}])$;
2	Renvoyer $\mathcal{L}_{in}$;
3	Fin.

Nous revenons ici sur quelques détails de IAVIC_L. Quant à l'algorithme IAVIC_R, son fonctionnement est similaire à celui de IAVIC_L. Notons qu'il ne suffit pas de trouver un vecteur quelconque $\mathbf{q}_0$ pour une face acceptable, il s'agit de déterminer un vecteur $\mathbf{q}_0^*$ qui reste valable lorsque nous augmentons l'épaisseur de la face. Nous utilisons pour cela, la stratégie suivante. Nous prenons un point $\underline{\mathbf{p}}$ situé sur la face à l'extrémité gauche de l'axe i du pavé $[\mathbf{p}]$. L'objectif premier étant de trouver un vecteur $\mathbf{q}_0^*$ afin que notre critère de proximité soit le plus faible possible. Nous cherchons donc $\mathbf{q}_0^* \in [\mathbf{q}]$ tel que d'une part $\mathbf{f}\left(\underline{\mathbf{p}}, \mathbf{q}_0^*\right) \in [\mathbf{y}]$ qui se traduit par $\mathcal{C}_{\mathbb{X}^c}\left(\underline{\mathbf{p}}, \mathbf{q}_0^*\right) = \emptyset$ et d'autre part

$$\mathbf{q}_0^* = \min_{\mathbf{q}_0} \left|\left|\mathbf{f}\left(\underline{\mathbf{p}}, \mathbf{q}_0\right) - \mathbf{y}_C\right|\right|_{2,\infty} \tag{5.41}$$

avec $\mathbf{y}_C = m([\mathbf{y}])$. Nous considérons ci-dessus un point $\underline{\mathbf{p}}$ plutôt qu'une face car l'évaluation des fonctions d'inclusion est plus précise et les contracteurs sont plus efficaces.

Algorithme : IAVIC_L(Entrées : i, $[\mathbf{p}]$, $[\mathbf{q}]$; Sortie : $[\mathbf{p}]_{in}$)	
1	$\underline{\mathbf{p}} \leftarrow \left(\underline{p}_1, ..., \underline{p}_i, ..., \underline{p}_{n_p}\right)^{\mathrm{T}}$;
2	$\mathbf{q}_0 \leftarrow$ LSBIC $\left(\underline{\mathbf{p}}, [\mathbf{q}]\right)$;
3	Si $(\mathbf{q}_0 = \emptyset)$: renvoyer $\emptyset$;
4	$\left[\underline{\mathbf{p}}\right] \leftarrow [p_1] \times ... \times \left[\underline{p}_i, \underline{p}_i\right] \times ... \times \left[p_{n_p}\right]$;
5	Si $\left(\mathcal{C}_{\mathbb{X}}\left(\left[\underline{\mathbf{p}}\right] \times [\mathbf{q}]\right) = \emptyset\right)$: renvoyer $\emptyset$;
6	$p \leftarrow \overline{p}_i$;
7	Répéter
8	$\left[\underline{\mathbf{p}}\right] \leftarrow [p_1] \times ... \times \left[\underline{p}_i, p\right] \times ... \times \left[p_{n_p}\right]$;
9	Si $\left(\mathcal{C}_{\mathbb{X}^c}\left(\left[\underline{\mathbf{p}}\right] \times \mathbf{q}_0\right) = \emptyset\right)$: renvoyer $\left[\underline{\mathbf{p}}\right]$;
10	Sinon $p = m\left(\left[\underline{p}_i, p\right]\right)$;
11	Jusqu'à $\left(w\left(\left[\underline{p}_i, p\right]\right) < \varepsilon\right)$;
12	Fin.

	IAVIC_R(Entrées : i, $[\mathbf{p}]$, $[\mathbf{q}]$; Sortie : $[\mathbf{p}]_{in}$)
1	$\overline{\mathbf{p}} \leftarrow \left(\overline{p}_1, ..., \overline{p}_i, ..., \overline{p}_{n_p}\right)^{\mathrm{T}}$;
2	$\mathbf{q}_0 \leftarrow$ LSBIC $(\overline{\mathbf{p}}, [\mathbf{q}])$;
3	Si $(\mathbf{q}_0 = \emptyset)$: renvoyer $\emptyset$;
4	$[\overline{\mathbf{p}}] \leftarrow [p_1] \times ... \times [\overline{p}_i, \overline{p}_i] \times ... \times \left[p_{n_p}\right]$;
5	Si $(\mathcal{C}_{\mathbb{X}}([\overline{\mathbf{p}}] \times [\mathbf{q}]) = \emptyset)$: renvoyer $\emptyset$;
6	$p \leftarrow \underline{p}_i$;
7	Répéter
8	$[\overline{\mathbf{p}}] \leftarrow [p_1] \times ... \times [p, \overline{p}_i] \times ... \times \left[p_{n_p}\right]$;
9	Si $(\mathcal{C}_{\mathbb{X}^c}([\overline{\mathbf{p}}] \times \mathbf{q}_0) = \emptyset)$: renvoyer $[\overline{\mathbf{p}}]$;
10	Sinon $p \leftarrow m([p, \overline{p}_i])$;
11	Jusqu'à $(w([p, \overline{p}_i]) < \varepsilon)$;
12	Fin.

5.2.7 Cas particulier

Dans le cas où

$$\mathbb{X} = \{\mathbf{p} | \exists \mathbf{q},\ f(\mathbf{p}, \mathbf{q}) = 0\}, \tag{5.42}$$

avec f définie de $\mathbb{R}^n$ dans $\mathbb{R}$, la caractérisation de $\underline{\mathbb{X}} \subset \mathbb{X}$ (approximation intérieure) nous pose un véritable problème. En effet, pour une fonction d'inclusion $[f]$ et un pavé non ponctuel $[\mathbf{p}]$ ($w([\mathbf{p}]) > 0$), il est difficile, voire impossible de trouver, grâce au calcul par intervalles, un vecteur $\mathbf{q}^* \in \mathbb{R}^{n_q}$ tel que : $[f]([\mathbf{p}], \mathbf{q}^*) = 0$. De même, il n'est pas simple de réaliser un contracteur $\mathcal{C}_{\mathbb{X}^c}$. Afin de résoudre ces problèmes, nous avons recours au théorème suivant,

Théorème 5.13. *Soit une fonction continue $h : \mathcal{D} \rightarrow \mathbb{R}$, avec $\mathcal{D}$ un compact de $\mathbb{R}^n$, il existe $\mathbf{w} \in \mathcal{D}$ tel que $h(\mathbf{w}) = 0$ si*

$$\exists\ \mathbf{u}, \mathbf{v} \in \mathcal{D} \textit{ tel que } h(\mathbf{u}) \cdot h(\mathbf{v}) \leq 0. \tag{5.43}$$

Ainsi, pour prouver qu'un pavé $[\mathbf{p}]_0$ est acceptable ($[\mathbf{p}]_0 \subset \mathbb{X}$), il suffit de trouver deux vecteurs $\mathbf{q}_1$ et $\mathbf{q}_2 \in [\mathbf{q}]_0$ tel que :

$$\forall \mathbf{p} \in [\mathbf{p}]_0,\ f(\mathbf{p}, \mathbf{q}_1) \geq 0 \text{ et } f(\mathbf{p}, \mathbf{q}_2) < 0 \tag{5.44}$$

ce qui revient pour nous à montrer que,

$$[f]([\mathbf{p}]_0, \mathbf{q}_1) \subset [0, +\infty[\text{ et } [f]([\mathbf{p}]_0, \mathbf{q}_2) \subset]-\infty, 0[. \tag{5.45}$$

Nous en déduisons que

$$\begin{array}{rcl} \mathbb{X} & = & \{\mathbf{p} | \exists \mathbf{q}_1, \exists \mathbf{q}_2,\ f(\mathbf{p}, \mathbf{q}_1) \geq 0 \text{ et } f(\mathbf{p}, \mathbf{q}_2) < 0\} \\ & = & \mathbb{X}_1 \cap \mathbb{X}_2 \end{array} \tag{5.46}$$

avec

$$\mathbb{X}_1 = \{\, \mathbf{p} |\, \exists \mathbf{q},\ f\,(\mathbf{p}, \mathbf{q}) \geq 0\} \tag{5.47}$$

et

$$\mathbb{X}_2 = \{\, \mathbf{p} |\, \exists \mathbf{q},\ f\,(\mathbf{p}, \mathbf{q}) < 0\}\,. \tag{5.48}$$

Remarque 5.14. *En ce qui concerne les contraintes de type* $\mathbf{f}\,(\mathbf{p}, \mathbf{q}) = \mathbf{0}$ *avec* $\mathbf{f}$ *une fonction vectorielle définie de* $\mathbb{R}^n$ *dans* $\mathbb{R}^m$*, une méthode simple consiste à déterminer une seule équation à partir de toutes les autres contraintes. Des substitutions de variables ainsi que des manipulations algébriques sont effectuées afin d'obtenir une équation qui contienne les informations des différentes contraintes. La solution proposée dépend en grande partie de la simplicité des expressions des contraintes, de ce fait notre méthode n'est pas applicable dans un cadre général. Une méthode générale pour ce problème fait l'objet d'études récentes.*

5.3 Perspective

Dans cette section, nous présentons une autre piste pour calculer une approximation intérieure et extérieure de la projection d'ensembles définis par des inégalités.

Revenons à l'ensemble $\mathbb{S}$ défini selon (5.1), nous désirons trouver des conditions suffisantes pour montrer qu'un pavé $[\mathbf{p}]$ est acceptable ou inacceptable pour $\mathbb{X} = \text{Proj}_{\mathbf{p}}\,(\mathbb{S})$.

Jusqu'à présent, nous avons mis au point des méthodes pour rechercher un vecteur $\mathbf{q}_0 \in \mathbb{R}^{n_q}$ tel que : $\mathcal{C}_{\mathbb{S}^c}\,([\mathbf{p}]\,, \mathbf{q}_0) = \emptyset$ ($[\mathbf{p}] \times \mathbf{q}_0 \subset \mathbb{S}$). Une autre approche consiste à considérer un vecteur $\mathbf{q}$ dont les composantes sont linéairement dépendants de ceux de $\mathbf{p}$. Nous avons $\mathbf{q} = \mathbf{A}.\mathbf{p}$, avec $\mathbf{A} \in \mathbb{R}^{n_q \times n_p}$ une matrice à coefficients réels. Le pavé $[\mathbf{p}]$ est acceptable ($[\mathbf{p}] \subset \mathbb{X}$) si,

$$\forall\ \mathbf{p} \in [\mathbf{p}]\,,\ \mathbf{f}\,(\mathbf{p}, \mathbf{A}.\mathbf{p}) \in [\mathbf{y}] \tag{5.49}$$

qui revient à montrer que :

$$\mathcal{C}_{\mathbb{S}^c}\,([\mathbf{p}]\,, \mathbf{A}.\,[\mathbf{p}]) = \emptyset. \tag{5.50}$$

Dans le même temps, $[\mathbf{p}]$ est inacceptable ($[\mathbf{p}] \subset \mathbb{X}^c$) si,

$$\exists\ \mathbf{p} \in [\mathbf{p}]\,,\ r\,(\mathbf{p}, \mathbf{A}.\mathbf{p}) < 0 \tag{5.51}$$

où $r\,(\mathbf{p}, \mathbf{q})$ est donnée par (5.11). Dans la pratique, on démontre l'expression (5.51), en testant les différents coins du pavé $[\mathbf{p}]$. Nous obtenons ainsi une condition supplémentaire pour prouver que $[\mathbf{p}]$ est inacceptable.

Nous relevons que si la méthode nous paraît intéressante, elle n'est pas simple à mettre en oeuvre. En effet, comment choisir les coefficients de la matrice $\mathbf{A}$? Pour s'assurer d'une efficacité réelle de

la méthode, une connaissance sur la topologie de l'ensemble projeté peut être utile pour "choisir" les coefficients de **A**. A ce stade, nous n'avons malheureusement pas réussi à trouver une méthode ou une heuristique convenable pour déterminer, dans un cadre d'applications général, la matrice **A**.

5.4 Conclusion

Nous avons contribué dans ce chapitre à une étude approfondie de la projection d'ensembles. Ces études nous ont permis d'automatiser, de généraliser et d'améliorer le concept de projection d'ensembles. A cet effet, nous avons associé pour la première fois, d'une part les méthodes d'analyse syntaxique, la mise sous forme d'arbre des contraintes et, d'autre part, le calcul par intervalles et la propagation de contraintes qui permettent d'obtenir des résultats garantis. Ceci nous a permis de dévélopper le solveur PROJ2D dont nous avons présenté les différentes étapes de conception (voir sections 5.2.1, 5.2.2, 5.2.3, et 5.2.4). D'un autre côté, nous avons proposé une méthode de complexité polynomiale afin de réaliser une meilleure approximation intérieure. Lors de la réalisation de cette méthode appelée `IAVIC`, il est tout à fait concevable d'utiliser une heuristique mieux élaborée à la place de l'algortihme `LSBIC` (voir section 5.2.6.1). L'idée étant de combiner, les techniques intervalles qui garantissent la satisfaction des contraintes et une heuristique qui privilégie à travers un critère, les "meilleures" zones de recherche pour les paramètres quantifiées. Le chapitre suivant fera l'objet d'applications concernant la projection d'ensembles. Sur les différents exemples, nous donnerons les éléments de comparaison et d'analyse entre les différents algorithmes de projection (avec et sans l'utilisation de la méthode `IAVIC`). Ainsi, nous verrons dans quelles conditions l'algorithme `IAVIC` peut être efficace.

PARTIE 2 : APPLICATIONS

CHAPITRE 6

Quelques exemples d'application

De nombreux problèmes d'automatique nécessitent l'identification de paramètres incertains qui obéissent à des contraintes de type équations ou inéquations non linéaires (voir par exemple [Jaulin et al., 2001], [Barmish, 1994], [Milanese and Vicino, 1991], ou [Walter and Pronzato, 1997]). Pour résoudre ce genre de problème, il est possible de caractériser par un *sous-pavage* (c'est-à-dire une union de pavés) l'ensemble des paramètres qui satisfont les contraintes imposées. Pour des raisons de rapidité de calcul, au lieu de chercher l'ensemble de tous les paramètres *acceptables*, nous allons chercher à encadrer leur projection.

Nous avons fait le choix de présenter des problèmes désormais académiques. Ces problèmes en apparence différents seront tous ici abordés dans le cadre de la projection d'ensembles. Ceci nous permet de montrer, à terme, en quoi la projection d'ensembles constitue une approche intéressante pour traiter différents problèmes d'automatique. Nous en profiterons pour comparer les algorithmes `SPVIA` associés à la technique d'approximation intérieure `IAVIC`, cette méthode est développée dans la partie précédente (voir chapitre 5). Pour cela, une étude de performance des méthodes employées seront réalisées pour chaque exemple traité.

Les résultats que nous présenterons ici ont fait l'objet d'une publication [Dao et al., 2003].

6.1 Estimation à erreurs bornées

L'estimation ensembliste consiste à déterminer un ensemble de modèles de systèmes compatibles avec un certain nombre de contraintes. Rappelons que ces contraintes peuvent être décrites soit par des inégalités non linéaires, soit par des expressions plus complexes (lois de probabilité ou contraintes de type floue). Nous parlerons de problèmes d'estimation à erreurs bornées lorsque les sorties des modèles sont soumises à des bornes d'incertitude. Ici, les méthodes proposées sont basées sur la caractérisation d'ensembles par des approches intervalles.

6.1.1 Problème de Milanese et Vicino

6.1.1.1 Présentation

Ce premier exemple est tiré de la publication [Milanese and Vicino, 1991]. Il est défini par deux caractéristiques principales :

– **Modèle du processus :** Il s'agit ici d'un processus non linéaire qui dépend de la variable t et de 4 paramètres p_1, p_2, p_3, p_4. Le modèle théorique du processus s'écrit

$$y_{\mathrm{m}}(\mathbf{p}, t) = p_1 \exp(p_2 t) + p_3 \exp(p_4 t), \tag{6.1}$$

avec $\mathbf{p} = (p_1, p_2, p_3, p_4)^{\mathrm{T}}$.

– **Contraintes à satisfaire :** Les contraintes imposées au système s'écrivent

$$y_{\mathrm{m}}(\mathbf{p}, t_i) \in [y_i]\,, \quad \forall\, i \in \{1, .., 10\}\,, \tag{6.2}$$

où $[y_i]$ correspondent aux intervalles d'incertitudes associés aux temps de mesure t_i. Les données du problème sont rassemblées dans le tableau ci dessous.

i	t_i	$[y_i]$
1	0.75	$[6.9205, 7.8595]$
2	1.5	$[3.7855, 4.3945]$
3	2.25	$[1.553, 1.927]$
4	3	$[-0.00785, 0.20185]$
5	6	$[-2.7985, -2.3415]$
6	9	$[-2.9455, -2.4745]$
7	13	$[-2.2735, -1.8665]$
8	17	$[-1.612, -1.268]$
9	21	$[-1.129, -0.831]$
10	25	$[-0.793, -0.527]$

(6.3)

Les paramètres sont supposés appartenir au pavé

$$[\mathbf{p}] = [2, 60] \times [0, 1] \times [-30, -1] \times [0, 0.5]\,. \tag{6.4}$$

L'ensemble des paramètres acceptables est donc donné par

$$\mathbb{S} = \{\, \mathbf{p} \in [\mathbf{p}] |\ \ y_{\mathrm{m}}\,(\mathbf{p}, t_i) \in [y_i]\,,\ i = 1, ..., 10\}\,. \tag{6.5}$$

Milanese et Vicino ont proposé une méthode dans [Milanese and Vicino, 1991] pour déterminer un pavé qui contient $\mathbb{S}$. Ils ont obtenu en 8mn sur un PC 386Mhz l'inclusion suivante

$$\begin{aligned} \mathbb{S} \ \subset\ & [17.2, 26.9] \times [0.3, 0.49] \times [-16.1, -5.4] \\ & \times [0.077, 0.136]\,. \end{aligned} \tag{6.6}$$

Ici, notre approche consiste à caractériser les projections $\mathbb{P}_{j,k}$ de $\mathbb{S}$ suivant les paramètres p_j, p_k avec j et $k \in \{1, 2, 3, 4\}$. Ces projections s'écrivent :

$$\mathbb{P}_{j,k} = \text{Proj}_{(p_j, p_k)}(\mathbb{S}). \tag{6.7}$$

PROJ2D génère les sous-pavages gris et noirs qui encadrent les ensembles $\mathbb{P}_{1,3}$ (figure 6.1a) et $\mathbb{P}_{2,4}$ (figure 6.1b). L'intérêt de ces caractérisations consiste à choisir un couple de paramètres dans les ensembles $\mathbb{P}_{j,k}$ (j, $k \in \{1, 2, 3, 4\}$), puis de réaliser, par des projections successives, une estimation des paramètres manquants.

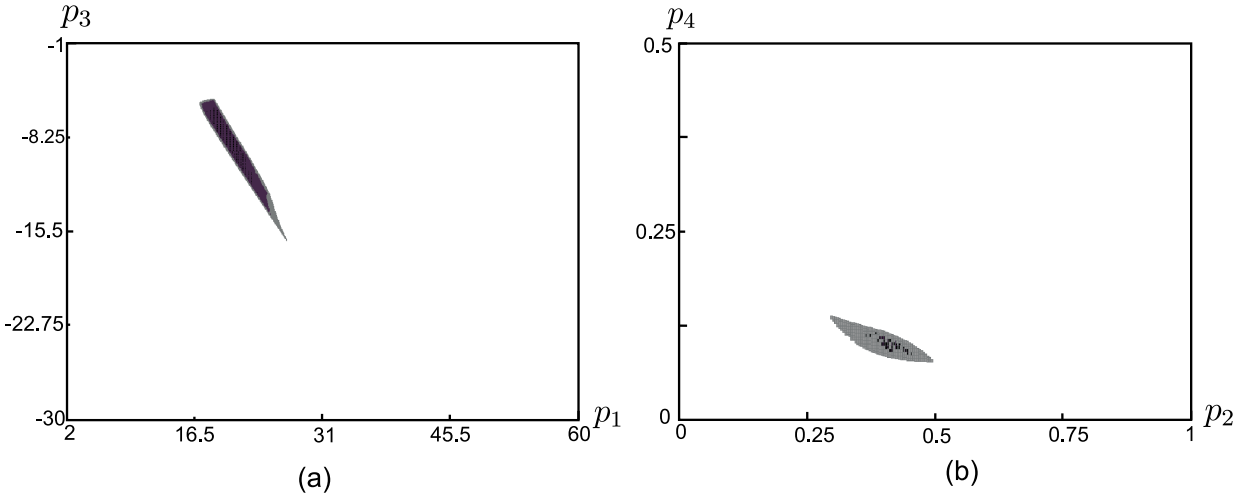

Figure 6.1 – Caractérisation de $\mathbb{P}_{1,3}$ et de $\mathbb{P}_{2,4}$.

6.1.1.2 Performances des méthodes employées

Nous allons analyser sur l'exemple ci-dessus, l'efficacité des différentes versions des algorithmes de projection présentés dans le chapitre précédent. Les étiquettes associées aux versions de `SPVIA` sont décrites comme suit,

- **Cas (1) :** `SPVIA2(Pile)`, sans méthode `IAVIC`;
- **Cas (2) :** `SPVIA2(Pile)`, avec méthode `IAVIC`;
- **Cas (3) :** `SPVIA2(File)`, sans méthode `IAVIC`;
- **Cas (4) :** `SPVIA2(File)`, avec méthode `IAVIC`.

Pour les cas ci-dessus, les dénominations `Pile` ou `File` correspondent aux structures de stockage des pavés intermédiaires associés aux variables quantifiées $\mathbf{q}$. Nous nous intéressons particulièrement à la caractérisation de $\mathbb{P}_{2,4}$. Notons Vol $\left(\underline{\mathbb{P}}_{2,4}\right)$ et Vol $(\partial\mathbb{P}_{2,4})$, les volumes respectifs de l'approximation intérieure et de la frontière pour $\mathbb{P}_{2,4}$. Les résultats obtenus à l'issu des tests sur PROJ2D sont consignés dans les tableaux 6.8 et 6.9. Ces tests ont été effectués sur un Pentium 1.8GHz.

	Vol $\left(\underline{\mathbb{P}}_{2,4}\right)$ $\left(\times 10^4\right)$			
t	**Cas (1)**	**Cas (2)**	**Cas (3)**	**Cas (4)**
2s	0	0	0	0
10s	0	0	0	0
50s	0	0.1315	0	0
1mn	1.39324	0.28536	0	0
1mn 30s	2.7754	2.1112	0	0
2mn 10s	4.18336	3.8425	0	0
3mn	6.49384	6.4874	0	0
3mn 30s	6.7513	6.5661	0	0
4mn	6.75127	7.6265	0	0
4mn 30s	6.7512	11.84569	0	0
5mn	7.308	15.01069	0	0
6mn	8.662	15.0107	0	0
8mn	8.9091	16.469	0	0
9mn	9.0255	19.6569	0	0
10mn	9.43	20.2	0	0

(6.8)

	Vol $\left(\partial\mathbb{P}_{2,4}\right)$ $\left(\times 10^3\right)$			
t	**Cas (1)**	**Cas (2)**	**Cas (3)**	**Cas (4)**
2s	10.02	11.3	17.5575	18.901
10s	5.55	6.183	12.01	15.525
50s	4.5433	4.9747	8.8635	9.7127
1mn	4.30544	4.7074	7.576	7.9813
1mn 30s	4.1012	4.3524	6.9626	6.98362
2mn 10s	3.87	4.133	6.8683	6.963
3mn	3.6153	3.77476	6.6461	6.6757
3mn 30s	3.581	3.74466	6.04337	6.01935
4mn	3.552	3.6231	5.89166	5.891664
4mn 30s	3.53973	3.183	5.70214	5.702136
5mn	3.47133	2.82	5.5813	5.589166
6mn	3.32834	2.79462	5.58121	5.58121
8mn	3.27438	2.6241	5.57957	5.579574
9mn	3.25847	2.2904	5.56646	5.566618
10mn	3.213	2.2031	5.222	5.221877

(6.9)

Nous analysons puis comparons à travers les variations dans le temps de Vol $\left(\underline{\mathbb{P}}_{2,4}\right)$ et Vol $(\partial\mathbb{P}_{2,4})$, les performances des 4 versions de SPVIA. La méthode est jugée efficace ou performante lorsque son comportement présente d'un côté, une rapide augmentation de Vol $\left(\underline{\mathbb{P}}_{2,4}\right)$, puis de l'autre une rapide diminution de Vol $(\partial\mathbb{P}_{2,4})$.

Dans notre exemple (voir figures 6.2), nous illustrons au mieux la compétition entre les méthodes (1) (`SPVIA2(Pile)`) et (2) (`SPVIA2(Pile) + IAVIC`). Sur l'intervalle de temps $[0, t_1]$, la méthode (1) est nettement plus efficace que les autres méthodes. La figure 6.2a indique pour la méthode (1) au delà de t_1, une augmentation du volume de $\underline{\mathbb{P}}_{2,4}$ supérieure à celle de la méthode (2). Au delà de t_2 (voir la figure 6.2b), la méthode (2) devient la plus performante des 4 méthodes.

Au bout d'un certain temps, nous constatons que la méthode `IAVIC` améliore la caractérisation de $\underline{\mathbb{P}}_{2,4}$ ce qui entraîne globalement une meilleure caractérisation de $\mathbb{P}_{2,4}$. Ainsi, une procédure qui favorise une meilleure approximation intérieure provoque au cours du temps, une baisse du nombre de partitions réalisées sur des domaines des variables quantifiées. Cette baisse peut être d'autant plus forte que la dimension des domaines de recherche est élevée.

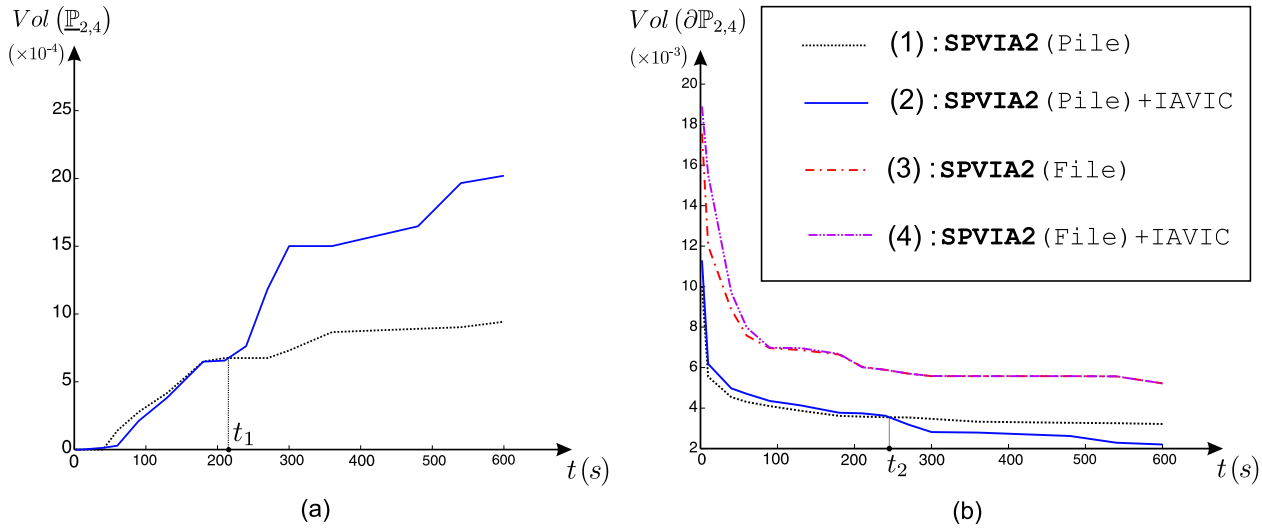

Figure 6.2 – Performances des méthodes SPVIA. A gauche, les variations en fonction du temps de Vol $\left(\underline{\mathbb{P}}_{2,4}\right)$. A droite, l'évolution du volume de $\partial\mathbb{P}_{2,4}$.

6.1.2 Problème avec temps de mesure incertains

6.1.2.1 Contexte

Nous allons maintenant traiter un cas similaire à l'exemple précédent. Le modèle s'écrit

$$y_{\mathrm{m}}\left(\mathbf{p},t\right) = 20\exp\left(-p_1 t\right) - 8\exp\left(-p_2 t\right). \tag{6.10}$$

Cette fois-ci, les instants de mesure sont incertains. Le tableau des incertitudes sur t et y est donné ci-dessous.

i	$[t_i]$	$[y_i]$
1	$[-0.25, 1.75]$	$[2.7, 12.1]$
2	$[0.5, 2.5]$	$[1.04, 7.14]$
3	$[1.25, 3.25]$	$[-0.13, 3.61]$
4	$[2, 4]$	$[-0.95, 1.15]$
5	$[5, 7]$	$[-4.85, -0.29]$
6	$[8, 10]$	$[-5.06, -0.36]$
7	$[12, 14]$	$[-4.1, -0.04]$
8	$[16, 18]$	$[-3.16, 0.3]$
9	$[20, 22]$	$[-2.5, 0.51]$
10	$[24, 26]$	$[-2, 0.67]$

(6.11)

La figure 6.3 constitue une représentation graphique des rectangles d'incertitude associés.

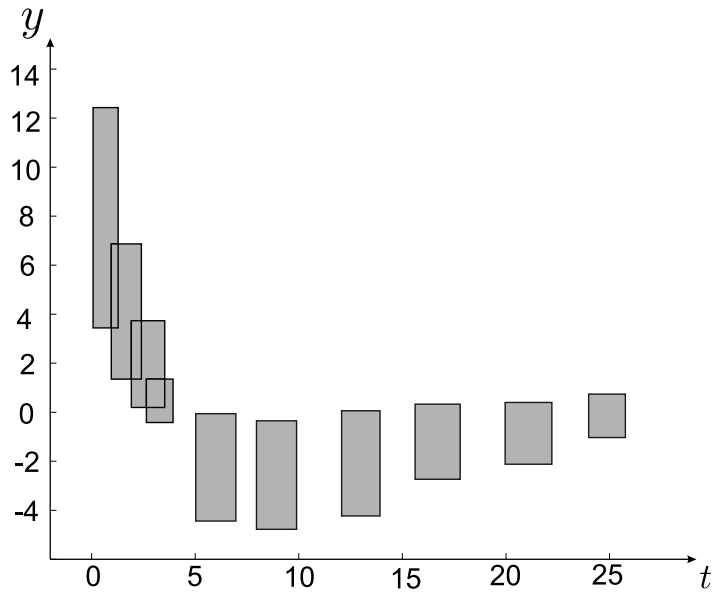

Figure 6.3 – Représentation des barres d'incertitude associées aux vecteurs **t** et **y** dans le cadre $[1, 26] \times [-7, 13]$.

L'ensemble des paramètres acceptables s'écrit

$$\begin{aligned} \mathbb{S} \;=\; & \{\mathbf{p} \in [\mathbf{p}] \,|\, \exists\, t_1 \in [t_1]\,,\; y_{\mathrm{m}}\,(\mathbf{p}, t_1) \in [y_1], \\ & \ldots,\; \exists\, t_{10} \in [t_{10}]\,,\; y_{\mathrm{m}}\,(\mathbf{p}, t_{10}) \in [y_{10}]\}. \end{aligned} \tag{6.12}$$

En regroupant les quantificateurs existentiels, il vient

$$\begin{aligned}\mathbb{S} \;=\; & \{\mathbf{p}\in[\mathbf{p}]|\,\exists\, t_1\in[t_1],\ldots,\exists\, t_{10}\in[t_{10}]\,,\\ & y_{\mathrm{m}}(\mathbf{p},t_1)\in[y_1],\ldots,y_{\mathrm{m}}(\mathbf{p},t_{10})\in[y_{10}]\},\end{aligned} \tag{6.13}$$

qui peut aussi s'écrire sous forme vectorielle

$$\mathbb{S}=\{\mathbf{p}\in[\mathbf{p}]|\,\exists\,\mathbf{t}\in[\mathbf{t}]\,,\ \mathbf{y}_{\mathrm{m}}(\mathbf{p})\in[\mathbf{y}]\}\,. \tag{6.14}$$

On reconnaît l'expression de la projection d'un ensemble (*cf.* chapitre 5) :

$$\mathbb{S}=\mathrm{Proj}_{\mathbf{p}}\,\{(\mathbf{p},\mathbf{t})\in[\mathbf{p}]\times[\mathbf{t}]\,,\ \mathbf{y}_{\mathrm{m}}(\mathbf{p})\in[\mathbf{y}]\}\,. \tag{6.15}$$

Pour $[\mathbf{p}]=[0,1.2]\times[0,0.5]$ et une précision de $\varepsilon=10^{-3}$, PROJ2D génère en 40 secondes, le sous-pavage de la figure 6.4. Les pavés noirs et gris encadrent l'ensemble solution. Notons, que ce problème a été résolu, par des techniques similaires dans [Jaulin and Walter, 1999], nous venons ici de montrer pour la première fois que l'estimation avec temps de mesure incertains, dans un contexte à erreurs bornées, est un problème de projection d'ensembles.

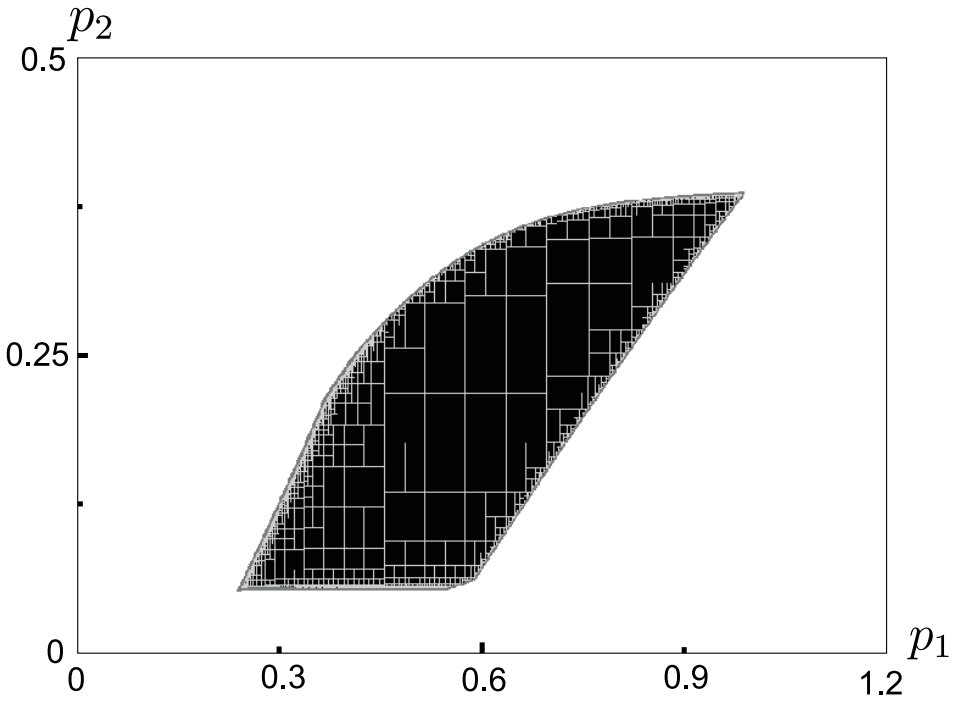

Figure 6.4 – Sous-pavage obtenu pour le problème d'estimation avec des temps de mesures incertains. Le pavé de recherche est $[0,1.2]\times[0,0.5]$.

6.1.2.2 Performances de SPVIA

Tout comme dans l'exemple précédent (section 6.1.1), nous avons relevé dans les tableaux qui suivent les évolutions des volumes de $\underline{\mathbb{S}}$ et $\partial\mathbb{S}$ (Vol $(\underline{\mathbb{S}})$ et Vol $(\partial\mathbb{S})$), ceci pour les 4 versions de `SPVIA`.

	Vol $(\underline{\mathbb{S}})$			
t	**Cas (1)**	**Cas (2)**	**Cas (3)**	**Cas (4)**
2s	0.06163	$2.58{\cdot}10^{-4}$	0.0128	0
5s	0.09251	0.0205	0.064	0.03486
10s	0.10296	0.0673	0.0832	0.06337
20s	0.11	0.0879	0.0978	0.08496
40s	0.114	0.1017	0.1046	0.101
1mn	0.1153	0.10456	0.10898	0.1033
3mn	0.118	0.1125	0.11405	0.111

(6.16)

	Vol $(\partial\mathbb{S})$			
t	**Cas (1)**	**Cas (2)**	**Cas (3)**	**Cas (4)**
2s	0.0736	0.17	0.12959	0.15115
5s	0.0331	0.1413	0.06603	0.1063
10s	0.02	0.07523	0.0419	0.06755
20s	0.0109	0.043	0.02503	0.04197
40s	0.00636	0.0248	0.01666	0.0217
1mn	0.00452	0.01976	0.0116	0.01876
3mn	0.00136	0.00883	0.0056	0.00922

(6.17)

Les résultats dans les tableaux indiquent une nette efficacité des méthodes (1) et (2). Nous sommes ici dans un cadre où la dimension de l'espace de recherche des variables quantifiées est relativement faible, il s'agit de la résolution de 10 problèmes d'estimation à 3 paramètres. Comme nous en avons fait la remarque ci-dessus, l'amélioration due à la méthode `IAVIC` est effectivement atténuée. Comme nous l'avons expliqué dans le chapitre 4, l'algorithme `SPVIA`, avec une structure de stockage de type File, ne sont efficaces qu'en dimension élevée et pour des ensembles volumineux.

6.1.3 Application de distributions étendues aux intervalles

Revenons sur le problème d'estimation de paramètres de la section (6.1.1). Une variante à ce problème consiste à déterminer les modèles qui sont compatibles non pas avec toutes les contraintes (6.2), mais avec au moins k_1 et au plus k_2 contraintes ($k_1 \leq k_2 \leq n$). Pour cela, nous reformulons notre problème d'estimation de paramètres en utilisant la distribution $\pi(x)$ qui se nomme *fonction porte*. Cette

fonction est définie par

$$\pi(x) = \begin{cases} 1 \text{ si } x \in \left[-\frac{1}{2}, \frac{1}{2}\right] \\ 0 \text{ sinon.} \end{cases} \tag{6.18}$$

Grâce à des manipulations algébriques, les contraintes (6.2) s'écrivent pour $i = 1, 2, ..., n$,

$$-\frac{1}{2} \leq \frac{y_{\mathrm{m}}(\mathbf{p}, t_i) + \frac{\overline{y}_i + \underline{y}_i}{2}}{\overline{y}_i - \underline{y}_i} \leq \frac{1}{2}. \tag{6.19}$$

En utilisant la fonction $\pi(x)$, (6.19) devient

$$\pi\left(\frac{y_{\mathrm{m}}(\mathbf{p}, t_i) + \frac{\overline{y}_i + \underline{y}_i}{2}}{\overline{y}_i - \underline{y}_i}\right) = 1. \tag{6.20}$$

Ainsi, quelque soit $k \in [k_1, k_2]$, les modèles $y_{\mathrm{m}}(.)$ compatibles avec k contraintes sont définis par l'ensemble des paramètres $\mathbf{p}$ tels que :

$$\sum_{i=1}^{n} \pi\left(\frac{y_{\mathrm{m}}(\mathbf{p}, t_i) + \frac{\overline{y}_i + \underline{y}_i}{2}}{\overline{y}_i - \underline{y}_i}\right) \in [k_1, k_2]. \tag{6.21}$$

Remarque 6.1. *Afin d'intégrer la fonction $z = \pi(x)$ dans nos algorithmes de caractérisation d'ensembles, il est relativement facile de réaliser un contracteur $\mathcal{C}_{\mathbb{Z}}$ pour l'ensemble :*

$$\mathbb{Z} = \{(x, z) |\, y = \pi(x)\}. \tag{6.22}$$

6.2 Analyse de sensibilité

Pour les notions sur l'identification et l'analyse de sensibilité, le lecteur pourra se référer au livre de Walter et Pronzato [Walter and Pronzato, 1997]. Nous nous intéressons ici à l'identification de paramètres incertains d'un système par minimisation d'un critère non convexe. Les résultats obtenus par des méthodes d'optimisation locales dépendent le plus souvent des conditions initiales. Ces méthodes convergent parfois vers des optimums locaux qui peuvent être éloignés de ou des optima globaux. La démarche que nous adopterons nous permet de visualiser les variations du critère en fonction des paramètres à l'intérieur de différentes régions de confiance. Cela nous aidera à nous prononcer sur l'identifiabilité du système à analyser.

Soit un système non linéaire dépendant de 2 paramètres p_1 et p_2 défini par

$$y_{\mathrm{m}}(\mathbf{p}, t) = 100 \exp(-p_1 t) + 101 \exp(-p_2 t). \tag{6.23}$$

Supposons qu'aux temps

$$\mathbf{t} = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)^{\mathrm{T}}, \tag{6.24}$$

nous ayons déterminer le vecteur des valeurs en sortie $\mathbf{y}$ de notre système en prenant comme paramètres $p_1^* = 0.5$ et $p_2^* = 1$. Ces valeurs de sortie sont ensuite volontairement "bruitées". Pour cet exemple nous avons choisi un bruit déterministe dans le but d'analyser le comportement du critère d'erreur. Les valeurs de sortie bruitées $\hat{y}_i$ s'expriment en fonction des valeurs y_i du modèle (6.23) par la relation

$$\hat{y}_i = \mathrm{E}\left(y_i + 0.5\right), \tag{6.25}$$

où $\mathrm{E}(x)$ représente la partie entière du réel x et y_i correspond à $y_{\mathrm{m}}\left(\mathbf{p}^*,t_i\right)$ pour $i = 1, ..., 10$.

Nous déduisons de ce qui précède que l'ensemble des mesures bruitées est donnée par le vecteur

$$\hat{\mathbf{y}} = (201, 98, 50, 27, 15, 9, 5, 3, 2, 1)^{\mathrm{T}}. \tag{6.26}$$

NB : il aurait été possible de choisir un bruit aléatoire de type uniforme, gaussien, *etc*.

Le critère d'erreur s'écrit

$$j\left(\mathbf{p}\right) = \left\|\mathbf{y}_{\mathrm{m}}\left(\mathbf{p}\right) - \hat{\mathbf{y}}\right\|_{\infty} = \max_{i=1,...,10} \left|y_{\mathrm{m}}\left(\mathbf{p}, t_i\right) - \hat{y}_i\right|. \tag{6.27}$$

A priori, il pourrait exister plusieurs optimums globaux $\hat{\mathbf{p}}$, dans ce cas le système ne serait pas globalement identifiable. Afin de visualiser le défaut d'identifiablité ainsi que la sensibilité de cet optimum global pour une variation infinitésimale du critère, définissons l'*épigraphe* de $j\left(\mathbf{p}\right)$

$$\mathbb{E} = \left\{ \left(\mathbf{p}, \tilde{j}\right) \in [\mathbf{p}] \times \left[\tilde{j}\right] \middle| \, j\left(\mathbf{p}\right) \leq \tilde{j} \right\}, \tag{6.28}$$

avec $[\mathbf{p}] = [0, 3]^{\times 2}$ et $\left[\tilde{j}\right] = [0, 5]$. Retenons que notre objectif n'est pas de réaliser un estimateur robuste, nous voulons simplement observer les variations de l'épigraphe en fonction d'un bruit de mesure associé. Les sous-pavages blancs et noirs sur figure 6.5 caractérisent la projection de l'épigraphe $\mathbb{E}$ suivant p_1 et $\tilde{j}$ ($\mathrm{Proj}_{p_1,\tilde{j}}\left(\mathbb{E}\right)$). Cette figure suggère l'existence de deux minima globaux. Une analyse plus précise de ces sous-pavages montre l'existence d'un seul minimum global représenté par le pic de gauche, le pic de droite étant un minimum local. Cette figure illustre le fait qu'une infime variation du critère, par exemple due à une légère modification du bruit de mesure, peut engendrer un optimum global complètement différent.

Remarque 6.2. *Dans le cadre de la caractérisation de projection d'épigraphe, il est possible d'améliorer considérablement l'approximation intérieure de* $\mathrm{Proj}_{p_i,\tilde{j}}\left(\mathbb{E}\right)$. *Nous procédons de la façon suivante. Pour chaque partition de domaine* $\left[\underline{p}_i, \overline{p}_i\right]_1 \times \left[\underline{\tilde{j}}, \overline{\tilde{j}}\right]_1$, *il suffit de montrer que la face inférieure est acceptable pour prouver que le partition entière est acceptable. Nous avons*

$$\mathcal{C}_{\mathbb{E}^c}\left(\left[\underline{p}_i, \overline{p}_i\right]_1 \times \underline{\tilde{j}}_1\right) = \emptyset \Rightarrow \mathcal{C}_{\mathbb{E}^c}\left(\left[\underline{p}_i, \overline{p}_i\right]_1 \times \left[\underline{\tilde{j}}, \overline{\tilde{j}}\right]_1\right) = \emptyset. \tag{6.29}$$

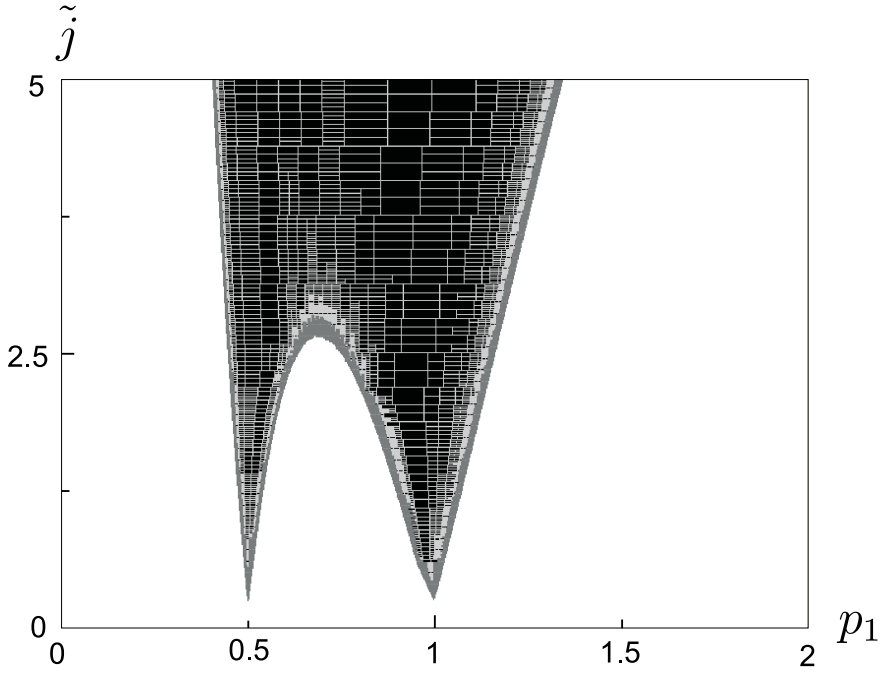

Figure 6.5 – Pavage caractérisant la projection de l'épigraphe du critère $j(\mathbf{p})$ dans l'espace $\left(p_1, \tilde{j}\right) \in [0,3] \times [0,5]$.

Sur un Pentium 1.8 GHz, les résultats sur les performances des méthodes de `SPVIA` sont présentées dans les tableaux 6.30 et 6.31.

	Vol $(\underline{\mathbb{S}})$			
t	**Cas (1)**	**Cas (2)**	**Cas (3)**	**Cas (4)**
2s	0	0.00376	0	0.00376
10s	0.693226	0.685462	0.27736	0.67143
40s	1.78605	1.5910323	1.476125	1.534351
2mn	2.3791446	2.15	2.1468	2.12135
5mn	2.534267	2.45733	2.403855	2.45276

(6.30)

	Vol $(\partial\mathbb{S})$			
t	**Cas (1)**	**Cas (2)**	**Cas (3)**	**Cas (4)**
2s	2.777	3.512413	2.80965	3.51241
10s	2.01738	2.13344	2.45402	2.147224
40s	0.9069	1.133823	1.22667	1.19232
2mn	0.3047	0.55	0.543257	0.58176
5mn	0.14543	0.231963	0.2798	0.236422

(6.31)

Rappelons les différentes étiquettes,

– **Cas (1)** : `SPVIA2(Pile)`
– **Cas (2)** : `SPVIA2(Pile)` + `IAVIC`
– **Cas (3)** : `SPVIA2(File)`
– **Cas (4)** : `SPVIA2(Pile)` + `IAVIC`.

6.3 Conception de robot

Dans cette partie, nous allons montrer et ceci constitue une des contributions de notre thèse, que certains problèmes de conception de robots sont des problèmes de projection d'ensembles. A titre d'illustration, nous allons considérer un exemple inspiré de l'article [Merlet, 1996]. Il s'agit de concevoir un robot (type grue de chantiers) constitué de 2 axes et de 3 articulations rotoïdes (voir figure 6.6). Plus précisément, il nous faut déterminer les longueurs ℓ_1 et ℓ_2 des 2 axes qui permettent à l'organe terminal du robot d'atteindre simultanément un nombre m de points $\mathbf{x}^i$ dans l'espace. Les couples (ℓ_1, ℓ_2) admissibles *a priori* sont contenus dans un pavé noté : $[\mathbf{l}] = [\ell_1] \times [\ell_2]$. Les angles de rotation des articulations θ_1, θ_2, et θ_3 doivent appartenir respectivement aux trois intervalles $[\theta_1]$, $[\theta_2]$ et $[\theta_3]$.

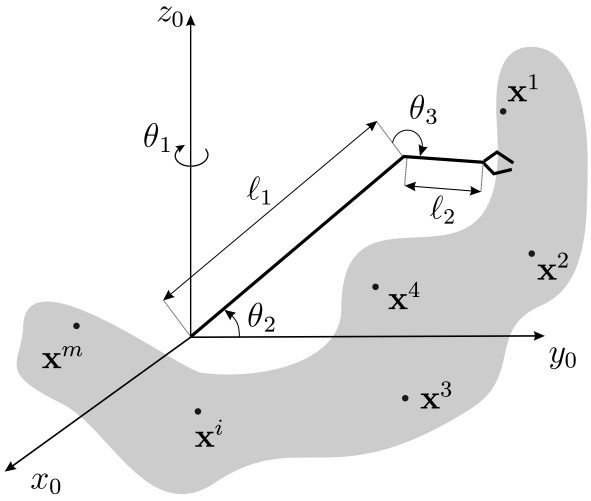

Figure 6.6 – Situation des axes du robot dans l'espace tridimensionnel.

Le modèle géométrique direct du robot (voir [Dombre and Khalil, 1999]) permet d'exprimer la position de l'organe terminal $\mathbf{x}$ en fonction du vecteur $\theta = (\theta_1, \theta_2, \theta_3)^{\mathrm{T}}$ des variables articulaires et du vecteur des longueurs $\mathbf{l} = (\ell_1, \ell_2)^{\mathrm{T}}$. L'expression associée s'écrit sous la forme :

$$\mathbf{x} = \mathbf{f}(\mathbf{l}, \theta), \tag{6.32}$$

avec

$$\mathbf{f}(\mathbf{l},\theta) = \begin{pmatrix} (\ell_1 \cos\theta_2 - \ell_2 \sin(\theta_2+\theta_3))\cos\theta_1 \\ (\ell_1 \cos\theta_2 - \ell_2 \sin(\theta_2+\theta_3))\sin\theta_1 \\ \ell_1 \sin\theta_2 + \ell_2 \cos(\theta_2+\theta_3) \end{pmatrix}. \tag{6.33}$$

L'ensemble des vecteurs $\mathbf{l}$ acceptables est

$$\begin{array}{rcl} \mathbb{S} & = & \{\mathbf{l} \in [\mathbf{l}] \mid \forall\, i \leq m,\ \exists \theta^i \in [\theta]\,,\ \mathbf{x}^i = \mathbf{f}\left(\mathbf{l},\theta^i\right)\} \\ & = & \{\mathbf{l} \in [\mathbf{l}] \mid \exists\ \theta^1 \in [\theta]\,,\ \mathbf{x}^1 = \mathbf{f}\left(\mathbf{l},\theta^1\right), \\ & & \ldots, \exists\ \theta^m \in [\theta]\,,\ \mathbf{x}^m = \mathbf{f}(\mathbf{l},\theta^m)\}. \end{array} \tag{6.34}$$

En rassemblant les quantificateurs existentiels, il vient

$$\begin{array}{rcl} \mathbb{S} & = & \{\mathbf{l} \in [\mathbf{l}] \mid \exists\ \left(\theta^1,\ldots,\theta^m\right) \in [\theta] \times \cdots \times [\theta]\,, \\ & & \mathbf{x}^1 = \mathbf{f}\left(\mathbf{l},\theta^1\right), \ldots,\ \mathbf{x}^m = \mathbf{f}(\mathbf{l},\theta^m)\}, \end{array} \tag{6.35}$$

qui peut s'exprimer en terme de projection par

$$\begin{array}{rcl} \mathbb{S} & = & \mathrm{Proj}_{\mathbf{l}}\{(\mathbf{l},\theta^1,\ldots,\theta^m) \in [\mathbf{l}] \times [\theta]^{\times m} \big| \\ & & \mathbf{x}^1 = \mathbf{f}\left(\mathbf{l},\theta^1\right), \ldots,\ \mathbf{x}^m = \mathbf{f}(\mathbf{l},\theta^m)\}. \end{array} \tag{6.36}$$

Nous avons choisi 6 points $\mathbf{x}^i$ à atteindre par le robot dont les coordonnées sont données dans le tableau ci-dessous.

i	$\mathbf{x}^i$
1	$(1,0,0)^{\mathrm{T}}$
2	$(0,1,0)^{\mathrm{T}}$
3	$(0,0,1)^{\mathrm{T}}$
4	$(1,1,0)^{\mathrm{T}}$
5	$(1,1,1)^{\mathrm{T}}$
6	$(0,1,2)^{\mathrm{T}}$

(6.37)

Considérons 4 situations pour lesquelles les angles de rotations θ^i sont soumis à des contraintes de butées.

- **Cas (a)** : $[\theta] = [-\pi,\pi] \times [-\pi,\pi] \times [-\pi,\pi]$;
- **Cas (b)** : $[\theta] = [-\pi,\pi] \times [-\pi,\pi] \times \left[-\frac{\pi}{4},\frac{\pi}{4}\right]$;
- **Cas (c)** : $[\theta] = [-\pi,\pi] \times \left[-\frac{\pi}{4},\frac{\pi}{4}\right] \times [-\pi,\pi]$;
- **Cas (d)** : $[\theta] = [-\pi,\pi] \times \left[-\frac{\pi}{4},\frac{\pi}{4}\right] \times \left[-\frac{\pi}{4},\frac{\pi}{4}\right]$.

L'espace de recherche des longueurs ℓ_1 et ℓ_2 choisi pour les 4 cas est à *priori* le pavé $[\mathbf{l}] = [0,2]^{\times 2}$. A la fin du calcul, les ensembles de couples (ℓ_1,ℓ_2) solutions sont délimités par les sous-pavages en gris (*cf.* figure 6.7 a, b, c, d). Sur un Pentium 1.8GHz, les temps de calcul sont de l'ordre de : (a) 10mn, (b) 25s, (c) 5s, (d) 0.5s et pour une précision fixée à $\varepsilon_r = 10^{-2}$. Notons que la figure 6.7d représente un amas de pavés, quasi-ponctuel et centré en $(1.4252, 0.9908)$.

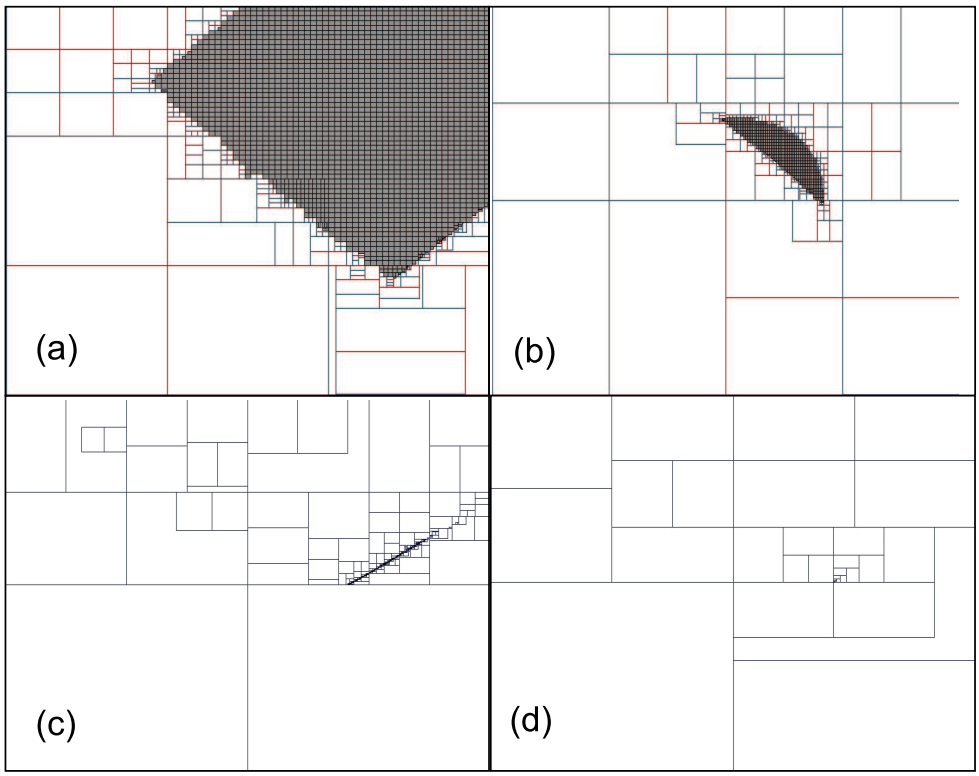

Figure 6.7 – a, b, c et d Pavages caractérisant l'ensemble des longueurs (ℓ_1, ℓ_2) qui permettent au robot d'atteindre les points $\mathbf{x}^i$. Les cadres correspondent au pavé $[\mathbf{l}] = [0, 2]^{\times 2}$.

6.4 Commande robuste

Ce problème de commande robuste provient de [Braems, 2002]. Nous désirons stabiliser un système (Σ_1) dont la fonction de transfert est donnée par

$$(\Sigma_1) : H(\mathbf{q},s) = \frac{q_1 q_2^2}{(q_2 s + 1)\left(s^2 + q_3 s + q_3^2\right)} \tag{6.38}$$

où $\mathbf{q} = (q_1, q_2, q_3)^{\mathrm{T}}$ est le vecteur des paramètres incertains supposé appartenir au pavé $[\mathbf{q}] = [0.9, 1.1]^{\times 3}$. Afin de stabiliser ce système, nous utilisons une commande de type proportionnelle et intégrale

$$(\Sigma_2) : C(\mathbf{p},s) = p_1 + \frac{p_2}{s} \tag{6.39}$$

dont les coefficients p_1 et p_2 peuvent être choisis arbitrairement dans le pavé $[\mathbf{p}] = [0, 1]^{\times 2}$.

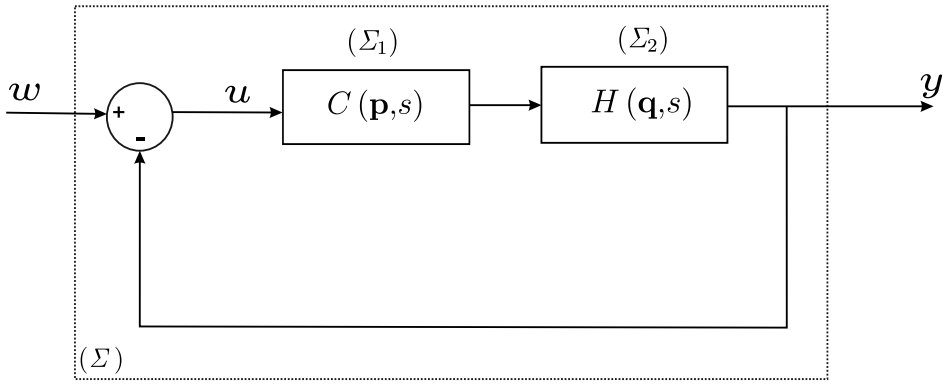

Figure 6.8 – Système (Σ_1) régulé par une commande proportionnelle - intégrale.

Le système bouclé, représenté sur la figure 6.8, a pour fonction de transfert

$$(\Sigma) : F(\mathbf{p},\mathbf{q},s) = \frac{H(\mathbf{q},s).C(\mathbf{p},s)}{1 + H(\mathbf{q},s).C(\mathbf{p},s)} \tag{6.40}$$

Trouver une commande robuste consiste à déterminer les coefficients q_1 et q_2 de $C(\mathbf{p},s)$, afin que le système en boucle fermée soit stable pour tous les paramètres $(q_1, q_2, q_3)^{\mathrm{T}}$ appartenant à $[\mathbf{q}]$. Définissons l'ensemble

$$\mathbb{S}_{\mathbf{p}} = \{\, \mathbf{p} \in [\mathbf{q}] \,|\, \forall \mathbf{q} \in [\mathbf{q}]\,,\ (\Sigma) \text{ est stable} \}\,. \tag{6.41}$$

La contrainte de stabilité du système (Σ) peut être transformée en inégalités de la forme $\mathbf{g}(\mathbf{p},\mathbf{q}) > \mathbf{0}$ grâce au critère de Routh. Ainsi,

$$\mathbb{S}_{\mathbf{p}} = \{\, \mathbf{p} \in [\mathbf{p}] \,|\, \forall \mathbf{q} \in [\mathbf{q}]\,,\ \mathbf{g}(\mathbf{p},\mathbf{q}) > \mathbf{0} \}\,. \tag{6.42}$$

Pour notre exemple, $\mathbf{g}(\mathbf{p},\mathbf{q})$ est donnée par

$$\begin{pmatrix} q_2 \\ 1 + q_2 q_3 \\ \left(q_3^2 q_2 + q_3 - \frac{q_2 q_3^2 (1+p_2 q_1)}{1+q_2 q_3}\right) \\ 1 + p_2 q_1 - \frac{(1+q_2 q_3)^2 p_1 q_1}{(1+q_2 q_3)\left(q_2 q_3^2 + q_3\right) - q_3^2 q_2 (1+p_2 q_1)} \\ p_1 q_1 q_3^2 \end{pmatrix}, \tag{6.43}$$

Notons

$$r(\mathbf{p},\mathbf{q}) = \min\left(g_1(\mathbf{p},\mathbf{q}), g_2(\mathbf{p},\mathbf{q}), \ldots, g_5(\mathbf{p},\mathbf{q})\right). \tag{6.44}$$

L'expression (6.42) devient :

$$\mathbb{S}_{\mathbf{p}} = \{\, \mathbf{p} \in [\mathbf{p}] \,|\, \forall \mathbf{q} \in [\mathbf{q}]\,,\ r(\mathbf{p},\mathbf{q}) > \mathbf{0} \}\,. \tag{6.45}$$

Or, le complémentaire de $\mathbb{S}_{\mathbf{p}}$ dans $[\mathbf{p}]$ s'exprime par

$$\begin{aligned}\mathbb{S}_{\mathbf{p}}^{c} &= \{\mathbf{p}\in[\mathbf{p}]|\neg(\forall\mathbf{q}\in[\mathbf{q}]\,,\ r(\mathbf{p},\mathbf{q})>\mathbf{0})\}\\ &= \{\mathbf{p}\in[\mathbf{p}]|\exists\mathbf{q}\in[\mathbf{q}]\,,\ r(\mathbf{p},\mathbf{q})\leq\mathbf{0}\}\\ &= \text{Proj}_{\mathbf{q}}\{(\mathbf{p},\mathbf{q})\in[\mathbf{p}]\times[\mathbf{q}]|\ r(\mathbf{p},\mathbf{q})\leq\mathbf{0}\}\,.\end{aligned} \tag{6.46}$$

Nous retrouvons l'expression d'une projection d'un ensemble. La caractérisation de $\mathbb{S}_{\mathbf{p}}^{c}$ dans le pavé initial $[\mathbf{p}]$ nous donne un encadrement de $\mathbb{S}_{\mathbf{p}}$ par la même occasion.

La figure 6.9 représente 3 sous-pavages. Les pavés noirs sont à l'intérieur de $\mathbb{S}_{\mathbf{p}}^{c}$ donc à l'extérieur de $\mathbb{S}_{\mathbf{p}}$. De même la zone blanche est à l'intérieur de $\mathbb{S}_{\mathbf{p}}^{c}$ et donc à l'extérieur de $\mathbb{S}_{\mathbf{p}}$. Les pavés gris indiquent la zone frontière ou encore la zone d'incertitude. Ces ensembles ont été obtenus par PROJ2D.

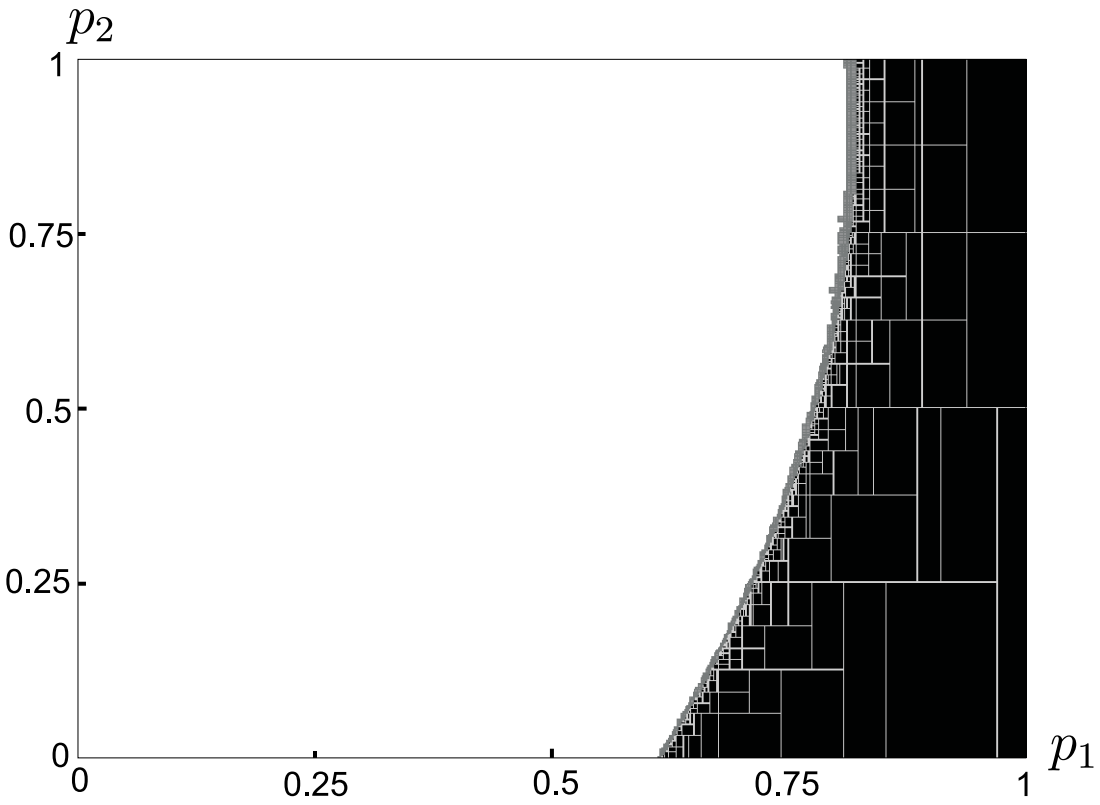

Figure 6.9 – Caractérisation de l'ensemble des commandes robustes $\mathbb{S}_{\mathbf{q}}$ par le sous-pavage gris. Le cadre correspond au pavé $[0,1]^{\times 2}$ dans l'espace (q_1,q_2).

A l'instar des exemples précédents, nous avons recueilli dans les tableaux 6.47 et 6.48, les données concernant les performances des algorithmes SPVIA.

	Vol $(\underline{\mathbb{S}})$ $(\times 10^2)$			
t	**Cas (1)**	**Cas (2)**	**Cas (3)**	**Cas (4)**
2s	6.25	18.455	14.45313	17.79585
5s	6.34766	19.5352	15.2344	19.2987
10s	7.3486	22.6815	19.043	21.81
20s	7.49512	23.1	20.02	22.552
30s	7.72705	23.42	20.63	23.03741
40s	7.9834	23.747	21.753	23.2869
2mn	8.40454	24.01	22.85351	23.7921
5mn	8.667	24.124	23.4359	24.01

(6.47)

	Vol $(\partial\mathbb{S})$ $(\times 10^2)$			
t	**Cas (1)**	**Cas (2)**	**Cas (3)**	**Cas (4)**
2s	22.07031	20.34906	14.9314	20.653
5s	20.84961	14.6147	13.5111	16.381
10s	19.043	6.52	8.11924	7.4627
20s	18.4326	4.81	6.81614	6.30327
30s	17.981	3.543	5.93765	4.3736
40s	17.4927	2.393	4.2562	3.51535
2mn	16.68	1.443	2.5034	1.84231
5mn	16.18042	0.9851	1.4766	1.26783

(6.48)

Il est intéressant dans ce cas, d'analyser les performances de SPVIA à travers les figures 6.10a et 6.10b. Parmi les 4 méthodes SPVIA, les méthodes (2) et (4) permettent d'obtenir la meilleure caractérisation pour $\mathbb{S}_{\mathbf{q}}$. Tandis que la méthode (1) qui demeurait jusqu'à présent la plus efficace enregistre le plus mauvais comportement. Ceci illustre, dans le cadre d'un problème où la dimension de l'espace des variables quantifiées (q_1, q_2, q_3 et q_4) est relativement élevée, l'efficacité des méthodes qui utilisent la procédure IAVIC par rapport aux autres méthodes. Nous expliquons ce phénomène en considérant de façon séparée, les effets des algorithmes SPVIA et IAVIC sur un pavé. Le premier algorithme de complexité exponentielle réalise, à partir de stratégie de partitionnement, un contracteur globalement consistant (cf. chapitre 3), le second de complexité polynomiale permet de déterminer un domaine acceptable dans un pavé. Rappelons qu'un domaine acceptable est constitué de vecteurs compatibles avec nos contraintes. Il s'établit comme nous l'avons indiqué une compétition entre les deux algorithmes. La réduction du pavé engendré par IAVIC entraîne une diminution du nombre de bissections. Ainsi la combinaison de SPVIA et de IAVIC permet, en grande dimension et pour des ensembles volumineux, une amélioration notable des algorithmes employés.

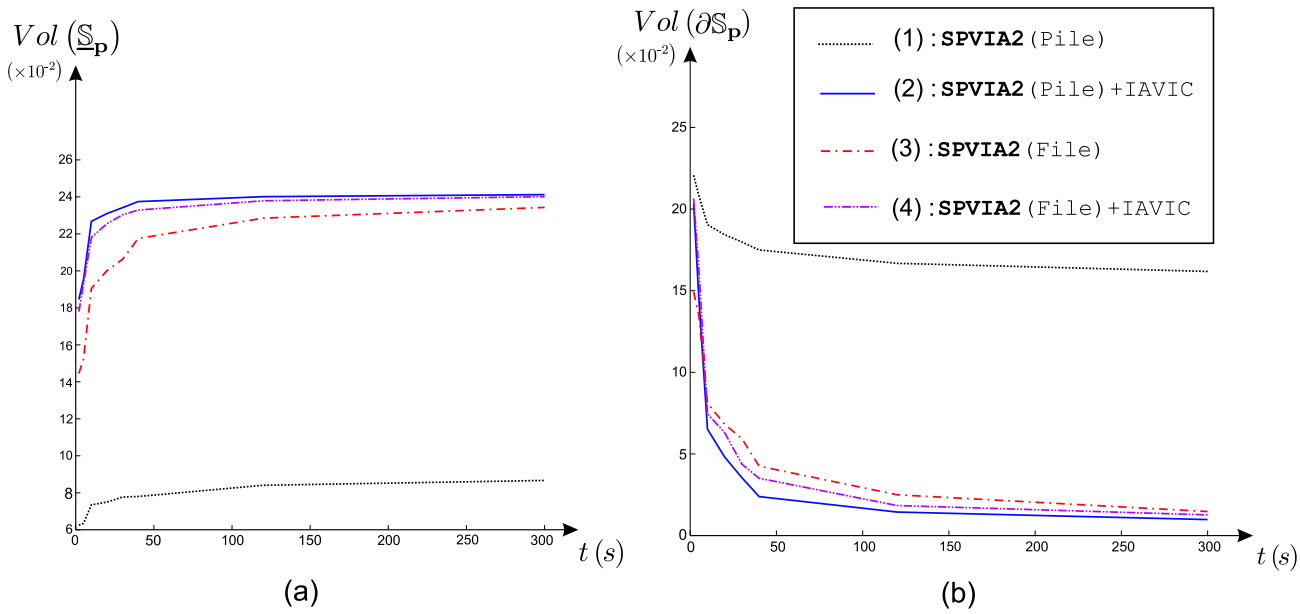

Figure 6.10 – Performances des méthodes SPVIA. A gauche, les variations en fonction du temps de Vol $\left(\underline{\mathbb{P}}_{2,4}\right)$. A droite, l'évolution de Vol $\left(\partial\mathbb{P}_{2,4}\right)$.

6.5 Conclusion

Ce chapitre nous a permis de montrer que plusieurs problèmes d'automatique, *a priori*, complètement différents, se ramènent au calcul de la projection d'un ensemble défini par des inégalités non linéaires.

Quelques exemples illustratifs ont été traités à l'aide de PROJ2D. Les temps de calcul obtenus sont malheureusement très sensibles au nombre de paramètres, phénomène qui a pu être observé à travers les exemples traités. En effet, il convenait d'améliorer l'efficacité de notre solveur afin d'atténuer cette sensibilité. Comme nous l'avons expliqué dans le chapitre précédent, lorsque le nombre de paramètres à estimer est important (≥ 4), la stratégie adoptée est basée sur la combinaison de méthodes intervalles et de certaines heuristiques. Ce procédé permet de trouver en un temps raisonnable le ou les meilleurs jeux de paramètres au sens du critère de proximité. Par conséquent, nous avons observé grâce à des études de performances, une amélioration des méthodes employées due à l'utilisation de l'algorithme d'approximation intérieur IAVIC. Ces améliorations sont perceptibles principalement pour les problèmes de grandes dimensions (cf. sections 6.1.1 et 6.4).

La plupart des exemples que nous avons vus, ici sont issus de la littérature. Les chapitres suivants traiterons d'autres champs d'application du concept de projection d'ensembles. Nous présenterons des problèmes sur la stabilité et la stabilisation des systèmes à retards, sur la commande de systèmes non linéaires (bateau à voile) ou encore sur la conception de filtres. Nous ferons sur ces problèmes un réel

travail de modélisation d'analyse et de conception. En effet, la façon de poser chaque type de problèmes permet d'envisager les méthodes de résolution les mieux appropriées.

CHAPITRE 7

Contribution à l'étude des Systèmes à retards

Les systèmes à retards ont une dynamique dont l'évolution à un instant donné dépend de son comportement passé sur une période finie. L'étude de ces systèmes trouvent de nombreuses applications dans la modélisation et la commande de processus chimique, biologique, médical, économique, de transmission de signaux, *etc.* (voir [Kolmanovskii and Myshkis, 1992]).

Dans le cas linéaire et lorsque les paramètres qui induisent les retards sont invariants dans le temps, l'étude de leur dynamique est basée sur celle de leur équation caractéristique, qui est une équation algébrique transcendante. Fondamentalement, ceci a amené les chercheurs à s'intéresser aux zéros d'un type de fonctions complexes de la forme,

$$P(s) = \sum_{i=0}^{n}\sum_{k=0}^{m} a_{ik} s^i e^{\tau_k s},\ s \in \mathbb{C}, \tag{7.1}$$

$P(s)$ est appelé *quasipolynôme*. Il existe à ce propos, de nombreux résultats concernant le test de stabilité des quasipolynômes ou de certaines familles de quasipolynômes (voir par exemple [Niculescu, 1996], [Niculescu, 2001] et [Gu et al., 2002] ainsi que leurs références). Beaucoup de ces travaux concernent la stabilité robuste en fonction d'incertitudes paramétriques sur les coefficients de quasipolynômes. Un problème classique consiste par exemple à déterminer les intervalles $[\underline{a}_{ik}, \overline{a}_{ik}] \subset \mathbb{R}$ de variation des coefficients a_{ik} (pour $i = 0, ..., n$ et $k = 0, ..., m$), tels que la stabilité de la fonction caractéristique $P(s)$ soit conservée à l'intérieur du pavé correspondant, avec un retard τ_k constant (voir [Kharitonov and Zhabko, 1994] et [Santos et al., 2003]). Un autre problème fréquemment abordé est la stabilité robuste de système à retard en fonction de retards incertains.

Un quasipolynôme $P(s)$ est stable s'il existe un réel $\sigma > 0$ tel que toutes les racines r_j de $P(s)$ soient à partie réelle strictement inférieure à $-\sigma$. Ainsi, tester la stabilité pour un système à retards revient à vérifier l'absence de racines à partie réelle positive de l'équation caractéristique. Une variante consiste à vérifier que le transfert du système est borné à la droite de cette verticale d'abscisse $-\sigma$. Ces tests sont en général semi-analytiques et leur mise en oeuvre numérique peut être complexe ([Niculescu, 2001], [Gu et al., 2002]).

Dans ce chapitre, nous proposons d'appliquer les approches fondées sur l'analyse et le calcul par intervalles aux systèmes à retards, afin de caractériser leurs propriétés fondamentales, et d'analyser la stabilité robuste, la norme H_∞ de leurs fonctions de transfert ou encore de lieu des racines d'un quasi-polynôme.

Comme nous l'avons indiqué ci-dessus, la caractérisation de ces propriétés fondamentales ont été largement étudiées dans cette dernière décennie, avec l'obtention de nombreuses méthodes numériques et algorithmiques afin de résoudre des problèmes classiques en automatique, tels que la stabilisation robuste ou encore le rejet de perturbation. Cependant, les méthodes proposées ne s'appliquent qu'à des classes très particulières de systèmes à retards et ces méthodes présentent en général des difficultés de mise en oeuvre. Dans ce sens, "les approches intervalles" permettent d'apporter une solution globale et garantie à certains problèmes numériques. Les problématiques liées à ces systèmes sont des sujets de recherche actuels (voir par exemple [Hohenbichler and Ackermann, 2003], [Li et al., 1998], [Santos et al., 2003] et [Vyhlìdal and Zìtek, 2003]).

Notre contribution à l'étude des systèmes à retards à fait l'objet des publications [Dao et al., 2004] et [Di-Loreto et al., 2005]. Ces études concernent les systèmes à retards invariants dans le temps. Les travaux effectués ont permis de montrer que les techniques de caractérisation ainsi que de projection d'ensembles sont adaptées aux problèmes rencontrés, comme par exemple tester l'absence garantie de zéros de quasipolynôme ou montrer qu'une fonction de transfert transcendantale est bornée, principalement quand les paramètres du système sont soumis à des incertitudes.

7.1 Généralités

Avant d'aborder les problèmes auxquels nous allons être confrontés, il semble approprié de rappeler quelques notions fondamentales sur les systèmes à retards. Ainsi, dans cette section, nous présenterons quelques définitions ainsi que des éléments concernant la stabilité des systèmes à retards. Parmi, les systèmes linéaires à retards invariants dans le temps, nous distinguons principalement deux catégories : les systèmes à retard de type *retardé* et *neutre.* Une variante à ces systèmes concerne les paramètres de retard qui peuvent être *commensurables* ou non.

7.1.1 Définitions

Définition 7.1 (Systèmes à retard de type retardé et neutre). Un système linéaire à retards indépendants du temps est un processus dont la dynamique est décrite par,

$$\dot{\mathbf{x}}(t) = \sum_{i=0}^{p} \mathbf{A}_i \mathbf{x}(t-\tau_i) + \sum_{i=0}^{p} \mathbf{E}_i \dot{\mathbf{x}}(t-\tau_i) + \sum_{i=0}^{p} \mathbf{B}_i \mathbf{u}(t-\tau_i), \tag{7.2}$$

où :

$\mathbf{x}(t) \in \mathbb{R}^n$ est le vecteur d'état;

$\mathbf{u}(t) \in \mathbb{R}^m$ est le vecteur des entrées du système;

$\mathbf{A}_i$, $\mathbf{E}_i \in \mathbb{R}^{n\times n}$ et $\mathbf{B}_i \in \mathbb{R}^{n\times m}$ représentent des matrices à coefficients réels;

$\tau_i \in \mathbb{R}^+$ ($i = 0, ..., p$) sont les paramètres de retard.

Le système à retards (7.2) est dit de *type retardé* si, pour tout $i = 0, ..., p$, $\mathbf{E}_i$ est une matrice à coefficients nuls. Dans ce cas, l'équation d'état devient,

$$\dot{\mathbf{x}}(t) = \sum_{i=0}^{p} \mathbf{A}_i \, \mathbf{x}(t - \tau_i) + \sum_{i=0}^{p} \mathbf{B}_i \, \mathbf{u}(t - \tau_i). \tag{7.3}$$

S'il existe, pour $\tau_i \neq 0$, au moins un coefficent non nul pour les matrices $\mathbf{E}_i$ ($i = 0, ..., p$), alors le système (7.2) est dit de *type neutre*.

Définition 7.2 (Système à retards commensurables). Un système dynamique est dit à *retards commensurables*, si le rapport entre deux paramètres de retard est rationnel, nous avons $\frac{\tau_i}{\tau_j} \in \mathbb{Q}$, quelque soit i et j. Ce type de système à retards possède des propriétés intéressantes pour l'étude du lieu des pôles.

7.1.2 Rappels sur la stabilité des systèmes linéaires

Les travaux de R. Bellman et J. R. Cooke dans [Bellman and Cooke, 1963] constituent un ouvrage de référence sur les systèmes à retards et les questions de stabilité. Dans un soucis de clarté, nous allons reprendre les points principaux concernant les différents types de stabilités de systèmes linéaires à retards.

7.1.2.1 Conditions de stabilité

En général, les systèmes dynamiques admettent une stabilité qui peut être *asymptotique*, *énergétique* ou encore dite de *Lyapunov*. Soit $P(s)$ la fonction caractéristique d'un système à retards, des conditions suffisantes pour la stabilité asymptotique sont données comme suit,

- S'il existe $\sigma < 0$ tel que $P(s) = 0$ si et seulement si $\mathrm{Re}(s) \leq \sigma$, le système est stable;
- S'il existe $s^* \in \mathbb{C}$ tel que : $\mathrm{Re}(s^*) > 0$ et $P(s^*) = 0$, le système est instable.

La stabilité asymptotique d'un système est souvent difficile à prouver, nous avons alors recours à la stabilité dite énergétique.

Définition 7.3 (Stabilité energétique - L_2 stable). Une fonction de transfert $h(s)$ est énergétiquement ou L_2 stable si elle est analytique et bornée dans le plan complexe droit $\mathbb{C}^+$.

Afin de démontrer qu'une fonction de transfert est L_2, nous calculons la norme H_∞ qui s'écrit,

$$H_\infty = \sup_{\mathrm{Re}(s)>0} |h(s)| < +\infty. \tag{7.4}$$

Pour des raisons de simplicité, (7.4) est valable pour des systèmes décrits par des fonctions de transfert. Dans le cas de matrices de transfert, une expression de la norme H_∞ est disponible dans [de Larminat, 1993].

Les approches utilisées afin de démontrer la stabilité (asymptotique, énergétique ou même Lyapunov) des systèmes à retard se classe en deux catégories. Nous avons,

- **Approche fréquentielle** (voir [Pontryagin, 1942], [Bellman and Cooke, 1963]) : Cette approche utilise les propriétés de l'équation caractéristique du système pour l'étude du lieu des pôles dans le plan complexes $\mathbb{C}^+$.
- **Approche temporelle** (voir [Niculescu, 1996], [Kolmanovskii and Myshkis, 1992]) : Elle est basée sur l'utilisation des fonctionnelles de Lyapunov (théorie de Lyapunov). La technique définit, à partir des équations d'état du système, un critère de stabilité au sens Lyapunov.

Les méthodes intervalles que nous proposerons par la suite peuvent être classées parmi les approches dites fréquentielles.

7.1.2.2 Cas de systèmes à retards "commensurables"

La fonction caractéristique du système (7.2) est donnée par,

$$P(s) = \det\left(\left(\mathbf{I}_n - \sum_{i=0}^{p}\mathbf{E}_i e^{-\tau_i s}\right)s - \sum_{i=0}^{p}\mathbf{A}_i e^{-\tau_i s}\right) \tag{7.5}$$

où $s \in \mathbb{C}$ est une variable complexe et $\mathbf{I}_n$ la matrice identité ($n \times n$). Après développement, cette fonction prend la forme du quasipolynôme

$$P(s) = \sum_{i=0}^{n}\sum_{k=0}^{p} a_{ik} s^i e^{-\tau_k s}. \tag{7.6}$$

Pour des retards commensurables, nous avons $\tau_k = k\tau$ tel que $\tau \in \mathbb{R}^+$. Ainsi, (7.6) devient

$$P(s,z) = \sum_{i=0}^{n}\sum_{k=0}^{p} a_{ik} s^i z^k \tag{7.7}$$

avec $z = e^{-\tau s}$. Des études désormais classiques sur les zéros de quasipolynômes du type (7.7) ont été effectuées dans [Pontryagin, 1942]. Les travaux menés dans cette article ont prouvé l'existence d'une infinité de racines pour $P(s,z)$. Dans certains cas, ces racines sont réparties suivant des directions asymptotiques. Nous n'exposerons pas ici le détails des développements de ces études, néanmoins nous rappelons quelques conséquences intéressantes pour la stabilité des systèmes à retards commensurables.

7.1.2.2.a Système à retard de type retardé Dans le cas présent, nous considérons pour l'expression (7.2) : $\mathbf{E}_i = \mathbf{0}^{n\times m}$ quelque soit $i = 1, ..., p$. Ce qui signifie pour (7.7), $a_{nk} = 0$, $\forall\, k \geq 1$. En posant $a_{n0} = 1$, nous obtenons,

$$P(s,z) = s^n + \sum_{i=0}^{n-1}\sum_{k=0}^{p} a_{ik} s^i z^k. \tag{7.8}$$

Pour la fonction (7.8), les théorèmes de Pontryagin stipulent que les racines instables de $P(s, e^{-\tau s})$ sont bornées. De ce fait, il est possible par des manipulations algébriques de borner les zéros du quasipolynôme (7.8) dans le plan complexe droit $\mathbb{C}^+$. Ainsi, il existe $M \in \mathbb{R}^+$ tel que si $P(s, e^{-\tau s}) = 0$ et $Re(s) \geq 0$ alors $|s| \leq M$. L'intérêt d'une telle démarche est de montrer l'absence de racines sur l'axe imaginaire.

Notre système est asymptotiquement stable si la fonction caractéristique $P(s, e^{-\tau s})$ n'admet aucun zéros dans le domaine $[0, M] \times [-M, M]$. Ceci se traduit de la façon suivante. Posons $q(s) = P(s, e^{-\tau s})$, en décomposant la variable complexe s, il vient

$$q(x + iy) = f_1(x, y) + i f_2(x, y), \tag{7.9}$$

où f_1 et f_2 sont des fonctions définies de $\mathbb{R}^2$ vers $\mathbb{R}$. Ainsi, l'algorithme `SIVIA` permet de prouver que

$$([0, M] \times [-M, M]) \cap \mathbf{f}^{-1}(\mathbf{0}) = \emptyset. \tag{7.10}$$

Selon le formalisme associé à l'inversion ensembliste,

$$\mathbf{f}^{-1}(\mathbf{0}) = \{(x, y) |\, f_1(x, y) = 0 \text{ et } f_2(x, y) = 0\}. \tag{7.11}$$

7.1.2.2.b Système à retard de type neutre Revenons à la fonction (7.7), dans le cas où il existe $a_{nk} \neq 0$ si $k \geq 1$, les zéros de $\sum_{k=0}^{p} a_{nk} z^k$ indiquent les directions asymptotiques du système de type neutre (voir [Pontryagin, 1942]). En effet, les zéros de (7.7) tendent à se répartir, lorsque $\text{Im}(s) \to \pm\infty$, sur des directions asymptotiques verticales. Dans le plan complexe, les positions de ces directions sont données par les parties réelles des solutions à l'équation : $\sum_{k=0}^{p} a_{nk} s^{-k\tau s} = 0$. Concernant la stabilité d'un système à retards neutre, un cas litigieux se pose lorsque l'axe imaginaire constitue la seule direction asymptotique située dans le plan complexe droit $\mathbb{C}^+$. Les conditions suffisantes de stabilité asymptotique (voir section 7.1.2.1) ne sont pas satisfaites. Nous essayerons alors de montrer la stabilité énergétique du système grâce au calcul de la norme H_∞. Cette situation sera illustrée par l'exemple de la section 7.2.

Dans les sections qui suivent, nous avons repris plusieurs exemples issus de la littérature afin d'étudier les questions de stabilité sur des systèmes à retards. Notre objectif est de montrer que les méthodes par intervalles sont des outils numériques intéressants pour l'étude des systèmes linéaires et à retards invariants dans le temps. Les résultats graphiques fournis par PROJ2D [1] permettent une visualisation des

[1] Disponible sur `http://www.istia.univ-angers.fr/~dao/Proj2DV5.zip`

zéros de quasipolynômes dans le plan complexe. Ceci est d'une grande utilité pour tester par exemple l'absence de pôles à notre système dans un large domaine du plan complexe droit. Bien sûr, ce type d'information ne garantit pas la stabilité asymptotique des systèmes à retard. Par contre, dès lors que nous avons le moyen, par des procédées analytiques ou algébriques, de déterminer une frontière qui borne les zéros de la fonction caractéristique dans le plan complexe droit, les approches intervalles peuvent apporter une réponse crédible au problème de stabilité. Les approches d'inversion ensembliste ainsi que de projection d'ensembles constituent donc des alternatives intéressantes aux méthodes semi-analytiques présentées par exemple dans [Niculescu, 2001] et [Gu et al., 2002].

7.2 Vers la Stabilité d'un système à retards neutre

Cette section est consacrée à la stabilité d'un système à retards de type neutre. Comme nous le verrons, la particularité de l'exemple choisi réside dans l'existence d'une seule direction asymptotique située sur l'axe imaginaire. De ce fait, nous allons fonder nos études de stabilité sur le tracé du gain ainsi qu'à la possibilité du calcul de la norme H_∞. Nous montrerons dans ce cadre, l'intérêt d'un algorithme d'inversion ensembliste type `SIVIA` pour le tracé garanti d'un graphe d'une fonction. On utilise MATLAB® et PROJ2D afin de comparer les résultats. Une étude des pôles dans le plan complexe complète cette section.

Considérons le système à retard de type neutre dont la fonction de transfert est

$$H(s) = \frac{y(s)}{u(s)} = \frac{1}{(s+1)(s(1-e^{-s})+1)}. \tag{7.12}$$

Sur l'axe imaginaire, le gain de cette fonction de transfert s'écrit $G(\omega) = |H(j\omega)|$, avec

$$G(\omega) = \frac{1}{\sqrt{1+\omega^2}} \frac{1}{\sqrt{(1-\omega\sin(\omega))^2 + \omega^2(1-\cos(\omega))^2}}. \tag{7.13}$$

Le graphe du gain en fréquence $h = G(\omega)$ est donné par l'ensemble

$$\mathbb{S} = \{(\omega, h) \in [\omega] \times [h] |\ G(\omega) - h = 0\} \tag{7.14}$$

où $[h]$ et $[\omega]$ sont 2 intervalles supposés contenir respectivement les valeurs *a priori* possibles du gain (7.13) et celles des fréquences pour le tracé du gain.

7.2.1 Tracé du graphe sous MATLAB®

A l'aide de MATLAB®, on effectue le tracé de graphes par interpolation et lissage. La technique consiste à relier les éléments d'un nuage de points par des segments de droite ou des tronçons de courbe. L'inconvénient de cette méthode réside dans le risque de ne pas détecter sur le tracé d'une courbe, la

présence de "pics" de faibles largeur. Ce problème reste présent quelque soit le pas d'échantillonnage choisi. En effet, une brève analyse de (7.12) montre que les pôles du système se situent d'une part en -1 pour le pôle simple, et pour la partie neutre, admettent l'axe imaginaire comme direction asymptotique, avec comme approximation $s_k = j\left(2k\pi + \frac{1}{2k\pi}\right) + o\left(\frac{1}{k^2}\right)$, $k \in \mathbb{Z}^*$ et $H(s_k) = -2 - \frac{2j}{3k\pi} + o\left(\frac{1}{k^2}\right)$, (voir [Pontryagin, 1942] et [Bellman and Cooke, 1963]). Par conséquent, dès que k augmente en module ($|k| \geq 3$), le gain (7.13) présente des pics régulièrement espacés de 2π, dont la largeur est décroissante en fonction de k, et dont l'amplitude est constante.

Nous risquons donc d'obtenir, pour des fréquences non nulles des amplitudes erronées de ces pics, voir même l'inexistence de certain pics par suite de l'échantillonnage réalisé sous MATLAB®. Ce phénomène s'accentue pour les hautes fréquences, car la largeur de ces pics est de plus en plus faibles. Nous illustrons ce problème avec les tracés de $h = G(\omega)$ pour 2 pas d'échantillonnage différents, respectivement $\Delta\omega_1 = 0.1$ Hz et $\Delta\omega_2 = 0.001$ Hz, avec $\omega \in [-1000, 1000]$.

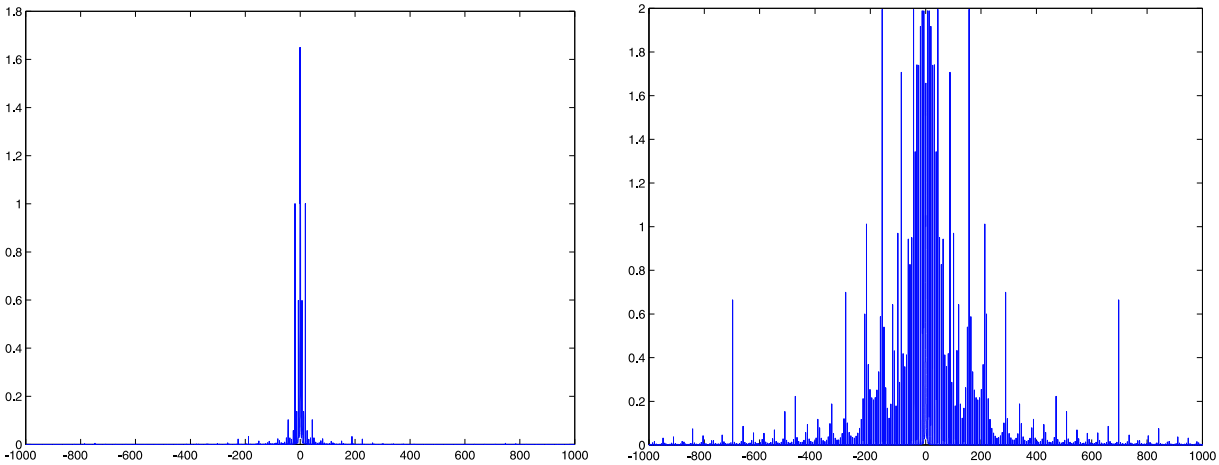

Figure 7.1 – Gain $h = G(\omega)$ sous MATLAB avec les pas d'échantillonnages de (a) : $\Delta\omega_1 = 0.1$ Hz et (b) : $\Delta\omega_1 = 0.001$ Hz, pour $\omega \in [-1000, 1000]$.

Sur les figures 7.1 a et b, très peu de pics sont présents dans le graphe du gain, et cela malgré un pas d'échantillonnage très petit. Les amplitudes des pics obtenus sont par ailleurs différentes, et n'ont aucune signification. La question que l'on peut alors se poser est la suivante : lequel de ces 2 graphes est conforme à la courbe $h = G(\omega)$?

7.2.2 Tracé du graphe avec PROJ2D

Le solveur PROJ2D permet grâce à `SIVIA`, d'approximer par des pavés tous les points d'un graphe, et donc de palier à l'inconvénient mis en évidence dans la section 7.2.1. Avec ces informations, les

valeurs prises par le gain seront bornées, et ceci dans un intervalle de fréquence initiale choisi. A titre de comparaison avec ce qui a été fait précédemment, nous réalisons sur PROJ2D le tracé de plusieurs graphes dans des domaines de fréquences différents. Les zones grises renferment de façon garantie tous les points possibles du graphe $h = G(\omega)$. Les figures 7.2 et 7.3 permettent de vérifier que les pics, de largeur décroissante en fonction de la fréquence ont tous une même amplitude de manière asymptotique, et qu'ils sont régulièrement espacés de 2π. Évidemment, en utilisant MATLAB®, le résultat malgré un pas d'échantillonnage choisi aussi petit que possible, ne sera jamais compatible avec celui du calcul par intervalles, par suite de sa structure.

De plus, l'approche intervalle permet au vue des résultats obtenus, de déterminer la norme H_∞ du transfert (7.12). En effet, le gain maximal vaut environ $\gamma \simeq 2$, et ce dernier correspond à la norme H_∞ du transfert si celui-ci est stable (voir [Desoer and Vidyasagar, 1975]). Afin d'affiner notre analyse, nous avons tenter de vérifier que $|H(s)| \leq \gamma$ pour $s \in \mathbb{C}^+$, ce qui se traduit par,

$$\forall s \in \mathbb{C}^+,\ |H(s)| \cap [\gamma, +\infty[= \emptyset. \tag{7.15}$$

Posons $g(x, y) = |H(x + iy)|$, en terme d'inversion ensembliste (7.15) devient

$$\left(\mathbb{R}^+ \times \mathbb{R}\right) \cap g^{-1}([\gamma, +\infty[) = \emptyset \tag{7.16}$$

avec

$$g^{-1}([\gamma, +\infty[) = \{(x, y) |\, g(x, y) \in [\gamma, +\infty[\}. \tag{7.17}$$

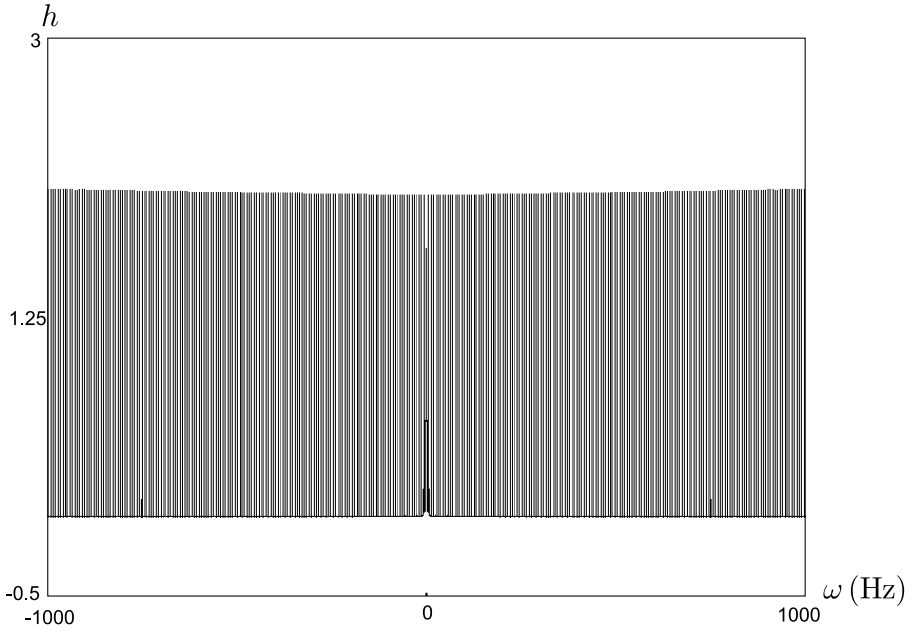

Figure 7.2 – Gain $h = G(\omega)$ sur l'intervalle fréquentielle $[-1000, 1000]$ sous PROJ2D, avec $h \in [-0.5, 3]$.

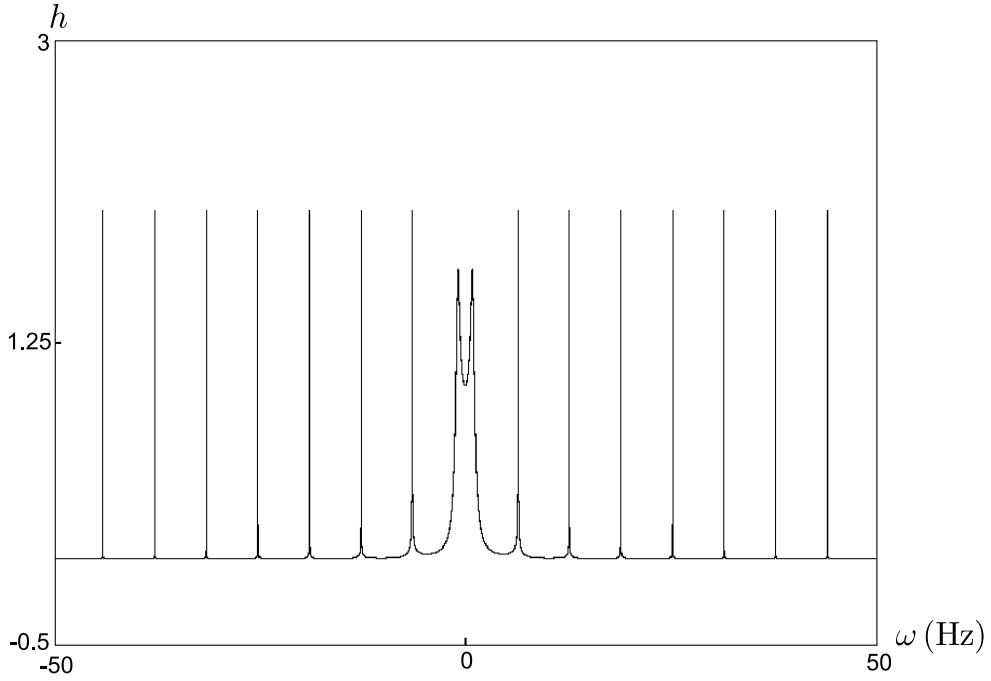

Figure 7.3 – Gain $h = G(\omega)$ sur l'intervalle fréquentiel $[-50, 50]$ sous PROJ2D, avec $h \in [-0.5, 3]$.

7.2.3 Recherche de pôles dans le plan complexe

Dans cette partie, on s'intéresse à la localisation des pôles du système (7.12). Pour cela, on utilise PROJ2D afin de résoudre l'équation caractéristique de notre système dans un domaine borné du plan complexe. Cette équation s'écrit,

$$(s+1) \cdot \left(s\left(1-e^{-s}\right)+1\right) = 0,\, s \in \mathbb{C}. \tag{7.18}$$

En notant une racine de (7.18) sous la forme $r_k = (\mathrm{Re}r_k, \mathrm{Im}r_k)$, l'ensemble $\mathbb{S}$ des pôles du systèmes sont de la forme $\mathbb{S} = (-1, 0) \cup \mathbb{S}_n$, avec $\mathbb{S}_n$ l'ensemble des racines du terme neutre $s\left(1-e^{-s}\right)+1 = 0$. En particulier, on a

$$\mathbb{S}_n = \left\{ (x,y) \in \mathbb{R}^2 \middle| \, \mathbf{f}\,(x,y) = \mathbf{0} \right\}, \tag{7.19}$$

avec

$$\mathbf{f}\,(x,y) = \begin{pmatrix} x - (x\cos(y) + y\sin(y))\, e^{-x} + 1 \\ y + (x\sin(y) + y\cos(y))\, e^{-x} \end{pmatrix}^{\mathrm{T}}. \tag{7.20}$$

En termes d'inversion ensembliste, on a $\mathbb{S}_n = \mathbf{f}^{-1}(\mathbf{0})$. Sur la figure 7.4, les pavés noirs indiquent l'emplacement des pôles dans le plan complexe. On vérifie de cette manière que l'axe imaginaire constitue la direction asymptotique des pôles du terme neutre.

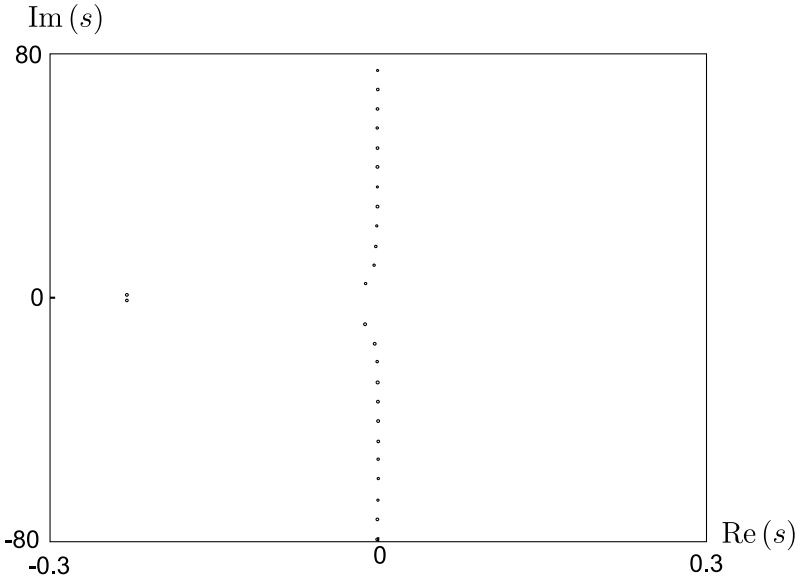

Figure 7.4 – Lieu de l'ensemble $\mathbb{S}_n$ des pôles du système (7.12) dans le plan complexe (x, y).

7.3 Stabilité robuste de systèmes à retards

Dans cette section, nous présentons quelques tests de stabilité robuste appliqués à des systèmes à retards. La caractérisation de la stabilité nécessite le calcul des racines des équations caractéristiques de systèmes paramétriques, qui sont non plus des quasipolynômes mais des familles de quasipolynômes en la variable s, en se limitant notamment au demi plan complexe droit fermé. Nous parlerons ici, de stabilisé robuste ou paramétrique car les systèmes à retards dépendent de paramètres incertains. Notons que ce type de problème est difficile à résoudre par des méthodes analytiques ou semi-analytiques. En effet, ces méthodes permettent souvent de définir, soit des conditions de stabilité robuste suffisantes mais pas nécessaires, soit l'inverse, à savoir des conditions nécessaires et pas suffisantes. De plus, une grande variétés de méthodes analytiques ou semi-analytiques sont nécessaires pour traiter différents types de systèmes à retards, tandis que les méthodes intervalles ont l'avantage de proposer des approches globales.

7.3.1 Système à retard de type retardé

Considérons un système à retard de type retardé (voir [Niculescu, 1996]) dont l'équation d'état s'écrit,

$$\begin{cases} \dot{x}(t) = -ax(t) - bx(t-\tau) \\ (a, b, \tau) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+ \end{cases} \tag{7.21}$$

avec une condition initiale appropriée. L'équation caractéristique d'un tel système est donnée par :

$$s + a + be^{-s\tau} = 0, \tag{7.22}$$

cette équation transcendantale admet en général une infinité de solutions. Son analyse peut se faire par exemple par la méthode de $\mathcal{D}$-subdivision ([Niculescu, 2001], [Schoen and Gerring, 1993]), dans l'espace paramétrique (a, b).

Dans [Niculescu, 1996] et [Niculescu, 2001], sont introduis les espaces suivants :

- $\mathcal{S}_{\omega,\infty} = \{(a, b)$: (7.21) est asymptotiquement stable $\forall \tau \in \mathbb{R}^+\}$,
- $\mathcal{S}_{s,\infty} = \{(a, b) \in \mathcal{S}_{\omega,\infty}$: (7.22) n'a pas de racine $s = j\omega$ quand $\tau \to \infty\}$,
- $\mathcal{S}_\tau = \{\, (a, b) : \exists\, \tau^* \in [0, +\infty[|$ (7.21) est asymptotiquement stable $\forall \tau \in [0, \tau^*[$ et (7.21) instable si $\tau \geq \tau^*\}$.

Ces trois espaces caractérisent la stabilité asymptotique du système (7.21) en fonction des paramètres a et b.

Théorème 7.4. *[Niculescu, 2001] Pour le système décrit par (7.21), nous avons :*

1. $\mathcal{S}_{\omega,\infty} = \{(a, b)$ *:* $a > |b|\}$
2. $\mathcal{S}_{s,\infty} = \{(a, b)$ *:* $a \geq |b|$ *et* $a + b > 0\}$
3. $\mathcal{S}_\tau = \{(a, b)$ *:* $b > |a|\}$

De plus, si $(a, b) \in \mathcal{S}_\tau$, alors le système est asymptotiquement stable pour n'importe quel retard $\tau \in [0, \tau^*[$, avec

$$\tau^*(a, b) = \frac{\arccos\left(-\frac{a}{b}\right)}{\sqrt{b^2 - a^2}}. \tag{7.23}$$

Le théorème 7.4 permet de représenter dans l'espace paramétrique (a, b) les zones de stabilité asymptotique. Les espaces $\mathcal{S}_{s,\infty}$ et $\mathcal{S}_{\omega,\infty}$ représentent les zones de stabilité, $\forall \tau \in \mathbb{R}^+$ (hormis la distribution lorsque le retard tend vers $+\infty$). L'espace $\mathcal{S}_\tau$ représente la zone de stabilité du système pour un retard τ tel que $0 \leq \tau < \tau^*$, qui est délimité par la droite $b = a$ et par la fonction implicite qui dépend du retard τ donnée par (7.23). Ainsi, nous avons dans le plan (a, b), la figure 7.5.

Il est donc intéressant d'appliquer les techniques de projection d'ensembles pour analyser les zones de stabilité indiquées sur la figure 7.5. Pour ce faire il suffit de prouver l'absence de solution à l'équation caractéristique (7.22) dans le demi plan complexe droit. Nous pouvons ainsi valider les résultats théoriques issus de la $\mathcal{D}$-subdivision.

Notons $P(s) = s + a + be^{-s\tau}$ la fonction caractéristique du système paramétrique (7.21). D'un point de vue purement ensembliste, l'ensemble des systèmes instables s'écrit

$$\mathbb{S} = \{\, (a, b, x, y, \tau) \in \mathcal{D} |\, P(x + iy) = 0\} \tag{7.24}$$

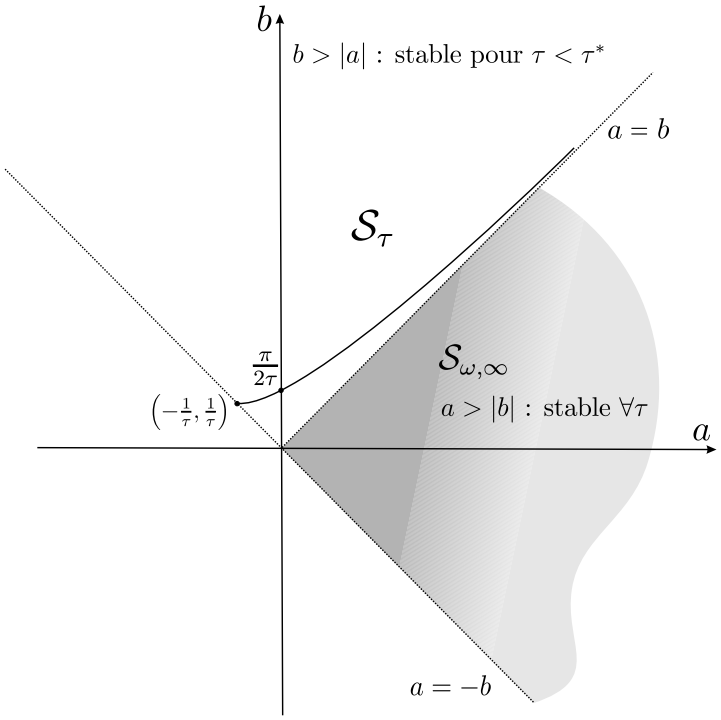

Figure 7.5 – Zones de stabilité -instabilité du système à retard (7.21) dans le plan (a, b).

avec $\mathcal{D} = \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+ \times \mathbb{R} \times \mathbb{R}^+$. Grâce à PROJ2D, nous avons alors la possibilité de visualiser des parties de la projection de $\mathbb{S}$ sur le plan (a, b) :

$$\mathbb{S}_1 = \text{Proj}_{(a,b)}(\mathbb{S}) = \{(a,b) | \exists (x, y, \tau) \in \mathcal{D}_1, P(x + iy) = 0\}, \tag{7.25}$$

avec $\mathcal{D}_1 = \mathbb{R}^+ \times \mathbb{R} \times \mathbb{R}^+$, ou encore le lieu des pôles dans $\mathbb{C}$:

$$\mathbb{S}_2 = \text{Proj}_{(x,y)}(\mathbb{S}) = \{(x,y) | \exists (a, b, \tau) \in \mathcal{D}_2, P(x + iy) = 0\}, \tag{7.26}$$

avec $\mathcal{D}_2 = \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+$.

Les figures 7.6 a et b constituent une illustration des projections $\mathbb{S}_1$ et $\mathbb{S}_2$. Une analyse de ces deux ensembles nous permet d'apporter une analyse sur la stabilité robuste, "l'instabilité potentielle" ou partielle (lorsqu'il existe au moins un retard $\tau_1 \in \mathbb{R}^+$ pour lequel le système est instable) ou encore l'instabilité totale qui signifie que pour certains paramètres a et b le système est instable quel que soit $\tau \geq 0$.

Considérons de ce fait, pour les paramètres incertains de notre système, les 3 situations suivantes,

- Cas 1 : $(a, b, \tau) \in [-1, 1] \times [2, 3] \times [0, 0.4]$;
- Cas 2 : $(a, b, \tau) \in [-1, 1] \times [2, 3] \times [0, 0.5]$;
- Cas 3 : $(a, b, \tau) \in [0.5, 300] \times [-0.4, 0.4] \times [0, 0.5]$.

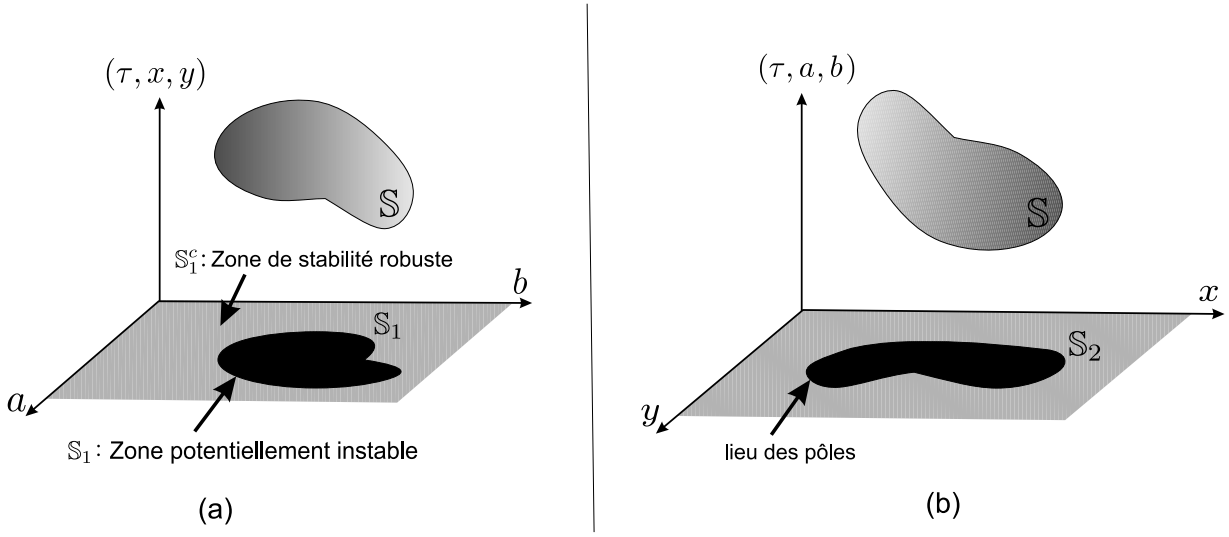

Figure 7.6 – Projection des systèmes instables dans les plan (a, b) et $\mathbb{C}$.

Dans les cas 1 et 2, on se place dans la zone $\mathcal{S}_\tau$, avec $\tau^* \simeq 0.4352$. Dans le dernier cas, on se place dans $\mathcal{S}_{\omega,\infty}$. Nous avons représenté sur la figure 7.7, la projection dans le plan (a, b) de systèmes partiellement instables pour le cas 2. La zone blanche garantit la stabilité du système malgré l'incertitude portant sur les paramètres et le retard. La zone gris-foncé correspond à la zone "potentiellement instable", c'est-à-dire pour laquelle il existe au moins une valeur de $\tau \in [2, 3]$ telle que le système (7.21) est instable. On se rend compte que c'est la zone correspondant à $\tau \in [\tau^*, 3]$ qui crée cette instabilité (voir le théorème 7.4). Au contraire, dans les cas 1 et 3, le système est stable, quel que soient a, b et τ pris dans leurs intervalles d'incertitude respectifs, aucune présence de zone instable ou "potentiellement instable" n'est détectée.

7.3.2 Système à retard de type neutre

On s'intéresse ici à la stabilité robuste dans le cas d'un système à retard de type neutre ([Gu et al., 2002], [Niculescu, 2001]).

Soit le système

$$\begin{cases} \dot{x}(t) - d\dot{x}(t-\tau) = -ax(t) - bx(t-\tau) \\ (a, b, d, \tau) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+ \end{cases} \tag{7.27}$$

avec une condition initiale appropriée. L'équation caractéristique de (7.27) est:

$$s\left(1 - de^{-s\tau}\right) + a + be^{-s\tau} = 0. \tag{7.28}$$

Si $|d| > 1$, alors (7.27) admet une infinité de racines à parties réelles positives.

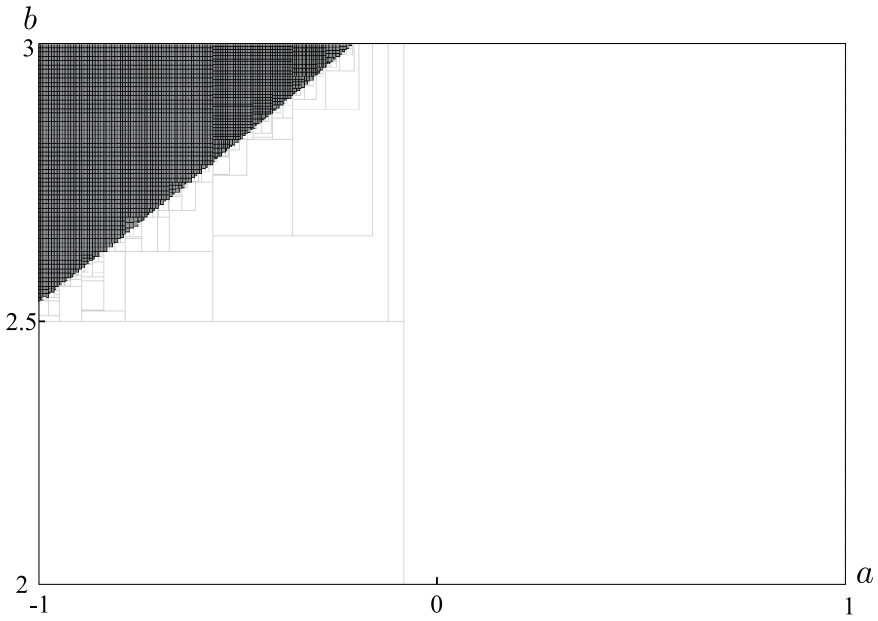

Figure 7.7 – Cas 2, zones de stabilité par projection de $\mathbb{S}$ dans le plan (a, b). La zone en gris-foncé est une zone d'instabilité dépendant du retard, la zone blanche garantit la stabilité robuste du système.

Théorème 7.5. *[Niculescu, 2001] Soit le système décrit par (7.27), et on suppose* $|d| < 1$. *Alors :*

1. $\mathcal{S}_{\omega,\infty} = \{(a, b, d) \, : a \geq |b| \, , a + b > 0\}$

2. $\mathcal{S}_{\tau} = \{(a, b, d) \, : b > |a|\}$

De plus si $(a, b, d) \in \mathcal{S}_{\tau}$, alors le système est asymptotiquement stable pour n'importe quel retard $\tau \in [0, \tau^*[$, avec

$$\tau^* = \sqrt{\frac{1-d^2}{b^2-a^2}} \arcsin\left(\frac{\sqrt{(b^2-a^2)\,(1-d^2)}}{b-ad}\right) \tag{7.29}$$

On retrouve donc pour ce système neutre une zone de stabilité tout à fait similaire à celle obtenue dans le cas retardé (voir section 7.3.1). Par rapport à la figure 7.5, une des principales modifications est le changement du point critique $\left(-\frac{1}{\tau}, \frac{1}{\tau}\right)$ en $\left(-\frac{1-d}{\tau}, \frac{1-d}{\tau}\right)$. Là encore, les techniques de projections des systèmes instables dans différents plans se montre utile pour caractériser la stabilité robuste de (7.27), principalement lorsque l'on se place dans la zone paramétrique $\mathcal{S}_{\tau}\,(a, b, d)$, où $(a, b, d) \in [a_1, a_2] \times [b_1, b_2] \times [d_1, d_2]$. Les résultats obtenus restent similaires à ceux de la section 7.3.1.

7.3.3 Problème de stabilisation

Cette section est consacrée à la stabilisation d'un système à retard. Pour cela, nous proposons de caractériser un ensemble de paramètres *stabilisants* pour une loi de commande.

Soit le système instable

$$\dot{x}(t) = x(t) + u(t-1). \tag{7.30}$$

Nous cherchons à stabiliser ce système par une loi de commande du type

$$u(t) = \alpha x(t) + \beta x(t-1) \tag{7.31}$$

avec $(\alpha, \beta) \in \mathbb{R} \times \mathbb{R}$. Une telle loi de commande appliquée à (7.30) est stabilisante si par exemple nous prenons comme paramètres $\alpha^* = -1.5$ et $\beta^* = 0.4$. Cependant, elle est stabilisante seulement si les paramètres (α^*, β^*) sont soumis à des variations inférieures en valeur absolue à 0.2, notons que c'est une condition nécessaire et pas suffisante. Le système (7.30) est régulé par retour d'état avec la commande (7.31), l'équation d'état du système en boucle fermée devient

$$\dot{x}(t) = x(t) + \alpha x(t-1) - \beta x(t-2). \tag{7.32}$$

Notre problème revient à déterminer les paramètres (α, β) pour lesquelles le système (7.32) n'admet aucun pôle dans le plan complexe $\mathbb{C}^+$. Nous caractérisons dans le plan paramétrique (α, β), la zone des valeurs de ces paramètres garantissant la stabilité du système régulé (7.32).

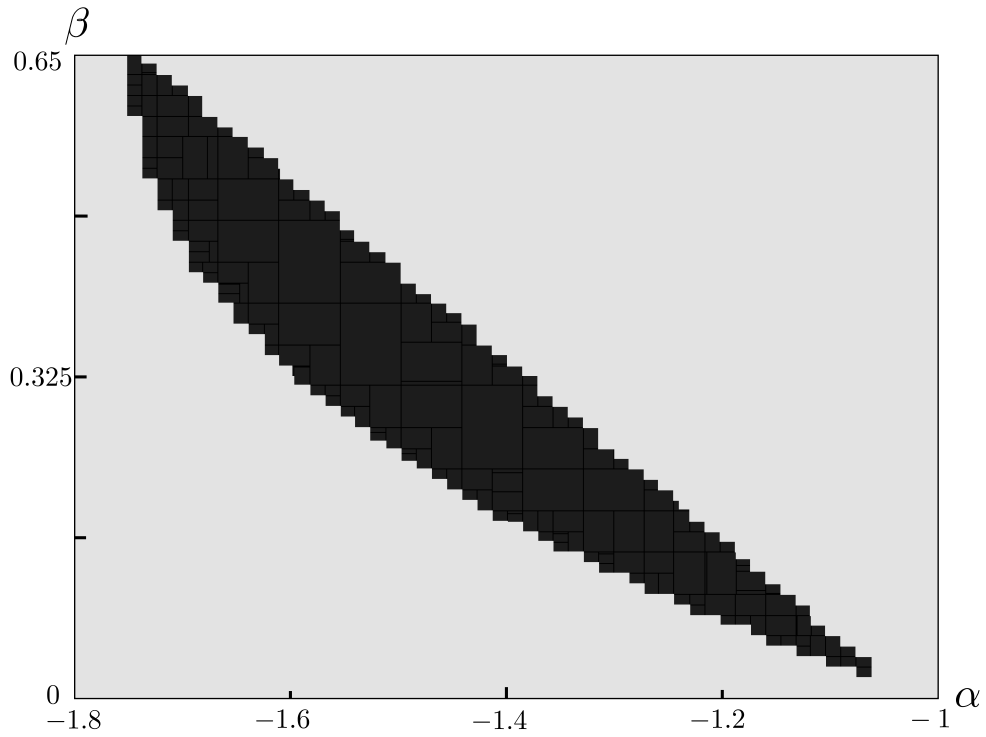

Figure 7.8 – Zone de stabilité du système (7.32) dans le plan (α, β); la loi de commande stabilisante est déterminée par (7.31). La zone sombre représente la zone de stabilité garantie du système en boucle fermée.

Sur la figure 7.8, les pavés gris foncés indiquent une zone de stabilité garantie du système en boucle fermée. Ces résultats sont obtenus pour des intervalles d'incertitude paramétrique fixés ($\alpha \in [-1.8, -1]$, $\beta \in [0, 0.65]$). Cette figure illustre la difficulté d'une modélisation robuste pour les systèmes à retards. En effet, nous constatons que la surface qui définie les paramètres (α, β) stabilisants est relativement "petite" (inférieure à 0.5). Comme nous l'avons déjà souligné, une analyse détaillée montre que de petites variations de ces paramètres (supérieures en valeur absolue à 0.2) suffisent pour le système régulé bascule dans l'instabilité.

Dans le soucis constant de montrer la fiabilité des approches intervalles, nous avons entrepris d'effectuer avec succès, des tests de stabilité robuste pour plusieurs autres exemples issus de la littérature sur les systèmes à retards (voir [Naimark et al., 1998], [Li et al., 1998], [Michiels et al., 2003], [Birdwell et al., 2003], [Hohenbichler and Ackermann, 2003], *etc.*). Dans la plupart de ces exemples, des domaines de stabilité robuste ont été calculés par les auteurs grâce à des méthodes analytiques ou semi-analytiques. Ces domaines établissent des conditions de stabilité pour des familles de quasipolynômes de systèmes à retards de type retardé. Nous avons donc repris et amélioré, dans certains cas, ces domaines afin de vérifier de façon garantie l'absence de zéros instables dans $\mathbb{C}^+$.

7.4 Conclusion

Dans ce chapitre, nous avons traité quelques problèmes liés aux systèmes à retards linéaires, dont les retards sont commensurables et invariants par rapport au temps. Ces problèmes ont concerné les thèmes suivants :

- la vérification des critères de stabilité (localisation des racines d'un quasipolynôme, l'approximation de la norme H_∞ pour un système donné, etc.);
- l'analyse de la stabilité robuste pour des familles de quasipolynômes, dont les coefficients et le retard sont incertains dans des intervalles donnés;
- stabilisation d'un système à retard par retour d'état (estimation de paramètres *stabilisants* pour une loi de commande).

Nous avons proposé pour résoudre ces problèmes deux approches basées sur le calcul par intervalles : les caractérisations de l'inversion ensembliste et de la projection d'ensembles. Les exemples incluent des systèmes à retards de type retardé et d'autres de type neutre. Comme cela a déjà été remarqué, la plupart des cas traités sont des systèmes à retards commensurables. Les techniques ensemblistes employées ici ont été étendues à des systèmes à retards éventuellement non commensurables dans [Di-Loreto et al., 2005].

Nous avons volontairement choisi des exemples pour lesquels il existe des méthodes analytiques d'étude. Notre démarche étant originale dans le domaine des systèmes à retards, les résultats obtenus

avec PROJ2D ont permis de valider nos travaux dans ce domaine. L'exemple de la section 7.3.3 est un peu plus complexe et montre que les méthodes intervalles s'appliquent indépendamment de l'existence d'une solution analytique. Il illustre aussi le fait que la représentation graphique des résultats proposée par PROJ2D se prête particulièrement bien à la conception des systèmes de commande et à l'évaluation de leur robustesse.

CHAPITRE 8

Synthèse d'une loi de commande optimale pour un bateau à voile

Le bateau à voile est un système non linéaire qui se déplace grâce à la force exercée par le vent. Fondamentalement, ce phénomène s'explique par les collisions de molécules d'air en mouvement sur les parois de la voile. De ce fait, un bateau à voile consomme une énergie renouvelable. D'où l'intérêt scientifique pour l'étude du comportement d'un tel système. La commande d'un bateau à voile reste un sujet fort peu abordé dans la littérature sur la commande des systèmes non linéaires. Le travail effectué ici permet de compléter quelques études qui ont été menées sur le bateau à voile (voir [Bryson and Ho, 1975], [Iachkine, 1999], [Jaulin, 2005]).

Dans ce chapitre, nous proposons une application de la projection d'ensembles pour la conception d'une commande d'ouverture de voile, ceci dans l'objectif de stabiliser un bateau à voile le plus rapidement possible autour d'une zone de l'océan. Par exemple, cette zone peut être repérer par une balise mobile. Dans un premier temps, il s'agit de trouver un régulateur pour le cap et l'orientation du bateau. Puis dans un deuxième temps, nous traiterons le problème qui concerne le réglage de l'ouverture de voile en fonction du cap. Nous utiliserons le logiciel PROJ2D[1] afin de caractériser sur 2 dimensions la projection d'ensembles complexes. Les résultats graphiques obtenus grâce à PROJ2D nous permettrons d'analyser *a fortiori* la fiabilité du modèle établi.

En conséquence, ce chapitre est organisé de la façon suivante. La section 8.1 est consacrée aux présentations du modèle et du système de commande général pour notre bateau à voile. Par ailleurs nous indiquons succinctement dans cette section, les notations utilisées par la suite. Dans la section 8.2, nous avons calculé un régulateur pour le cap et l'ouverture de voile grâce à une technique de linéarisation par retour d'état. Cette méthode est utilisée dans le cadre de la commande de systèmes non linéaires (voir [Isidory, 1995] et [Marino and Tomei, 1995]). Le réglage de l'ouverture de voile en fonction du cap est traité dans la section 8.3. Nous observerons lors du suivi d'une balise par notre bateau à voile, les

[1] Disponible sur `http://www.istia.univ-angers.fr/~ dao/Proj2DV5.zip`

performances de la commande conçue précédemment. Les éléments de cette simulation seront analysés puis critiqués dans la section 8.4. Enfin, la section 8.5 fait état d'un bref récapitulatif et de perspectives envisageables sur les études qui ont été réalisées.

8.1 Présentation

8.1.1 Modèle du bateau à voile

Nous considérons un modèle simpliste qui est déterminer grâce aux principes de la mécanique newtonienne (voir [Jaulin, 2005]). Les forces qui s'exercent sur le bateau sont uniquement dû au vent et à l'eau. Afin de simplifier le modèle de notre bateau, nous admettons être en présence d'un vent à vitesse constant ($V = Cte$). La dynamique de notre bateau à voile (voir figures 8.1a et 8.1b) est régie par les équations suivantes,

$$\left\{\begin{array}{rcll} \dot{x} & = & v\cos\theta, & \text{(i)} \\ \dot{y} & = & v\sin\theta - \beta V, & \text{(ii)} \\ \dot{\theta} & = & \omega, & \text{(iii)} \\ \dot{\delta}_v & = & u_1, & \text{(iv)} \\ \dot{\delta}_g & = & u_2, & \text{(v)} \\ \dot{v} & = & \frac{f_v\sin\delta_v - f_g\sin\delta_g - \alpha_f v}{m}, & \text{(vi)} \\ \dot{\omega} & = & \frac{(\ell - r_v\cos\delta_v)f_v - r_g\cos\delta_g f_g - \alpha_\theta\omega}{J}, & \text{(vii)} \\ f_v & = & \alpha_v\left(V\cos\left(\theta+\delta_v\right) - v\sin\delta_v\right), & \text{(viii)} \\ f_g & = & \alpha_g v\sin\delta_g. & \text{(ix)} \end{array}\right. \tag{8.1}$$

Les entrées u_1 et u_2 du système sont les dérivées des angles δ_v and δ_g. Le vecteur d'état $\mathbf{x} = (x, y, \theta, \delta_v, \delta_g, v, \omega)^{\mathrm{T}} \in \mathbb{R}^7$ est composé par,

- les coordonnées x, y du centre d'inertie G du bateau,
- θ l'orientation (cap),
- δ_v l'ouverture de voile,
- δ_g l'ouverture du gouvernail,
- V la vitesse tangentiel de G,
- ω la vitesse angulaire du bateau autour de G.

Les variables intermédiaires sont :

- f_v la force exercée par le vent sur la voile,
- f_g la force de frottement de l'eau sur le gouvernail.

Les paramètres constants (supposés connus) sont :

- V la vitesse du vent,

- r_g la distance entre le gouvernail et G,
- r_v la distance entre le mât et G,
- α_g la portance du gouvernail (si le gouvernail se trouve perpendiculaire à la marche du bateau, l'eau exerce une force de $\alpha_g v$ Newton sur gouvernail),
- α_v la portance de la voile (si la voile se trouve immobile, perpendiculaire au vent, ce dernier exerce une force de $\alpha_v V$ Newton),
- α_f le coefficient de frottement du bateau sur l'eau (l'eau exerce sur le bateau une force de frottement opposée au sens de la marche égale à $-\alpha_f v$),
- α_θ le coefficient angulaire de frottement du bateau sur l'eau (l'eau exerce sur le bateau un couple de frottement égal à $-\alpha_\theta \omega$; le bateau étant profilé pour garder un cap, α_θ sera grand devant α_f),
- J le moment d'inertie du bateau,
- ℓ la distance entre le mât et le centre de poussée de la voile,
- et le coefficient de dérive β (sans voile, le bateau tend à dériver à une vitesse de βV dans le sens du vent).

Ces paramètres, dont les unités sont conformes au système international (S.I), sont choisis comme suit,

$$\begin{aligned} \beta &= 0.05,\ r_s = 1 \text{ m},\ r_r = 2 \text{ m},\ V = 10 \text{ m/s}, \\ m &= 1000 \text{ kg},\ J = 2000 \text{ kg.m}^{-2},\ \alpha_f \in 60 \text{ kg.s}^{-1}, \\ \alpha_\theta &\in 500 \text{ kg.m}^2\text{.s}^{-1},\ \alpha_s = 500 \text{ kg.s}^{-1},\ \alpha_r = 300 \text{ kg.s}^{-1}. \end{aligned}$$

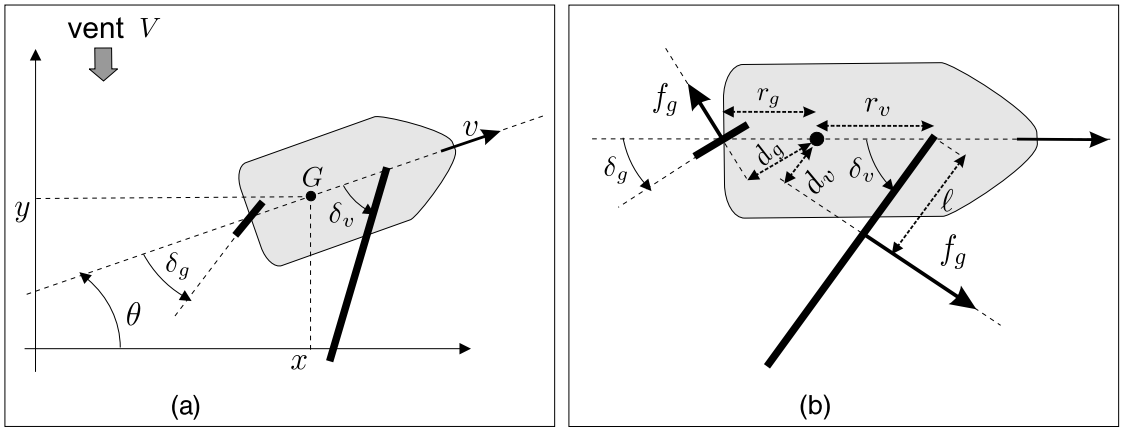

Figure 8.1 – Structure d'un bateau à voile.

8.1.2 Commande du bateau à voile

Ici, nous nous intéressons à la commande d'un bateau à voile. L'objectif est décrit comme suit. Nous désirons, à partir d'une consigne sur le cap $\hat{\theta}$, concevoir un régulateur qui permet d'une part d'imposer au

bateau une orientation (ou un cap) et d'autre part de régler les variables d'entrées du bateau pour atteindre une vitesse maximum. La figure 8.2 illustre un tel régulateur, avec en entrée une consigne d'orientation $\hat{\theta}$ et en sortie la variable $\mathbf{u} = \left(\dot{\delta}_g, \dot{\delta}_v\right)^{\mathrm{T}}$.

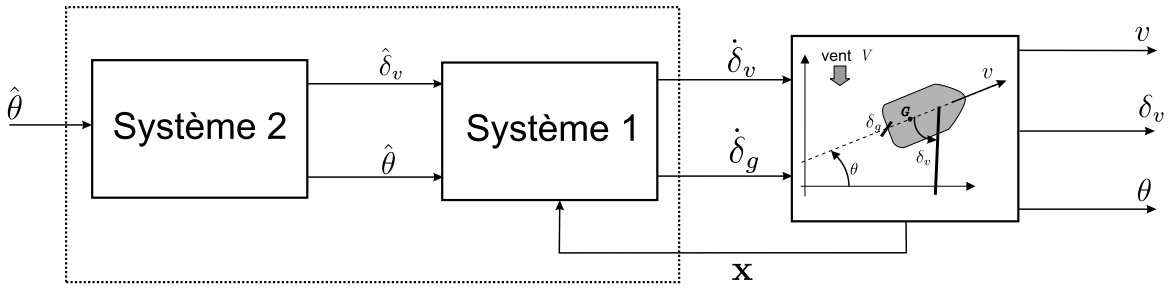

Figure 8.2 – Système de commande du bateau à voile.

Le système de commande se compose de 2 parties. Tout d'abord, le système 1 est chargé de réguler l'ouverture de voile et le cap autour de la consigne $\left(\hat{\delta}_v, \hat{\theta}\right)$. Ensuite, le système 2 règle l'ouverture de voile en fonction du cap, dans le but d'avoir un maximum de vitesse. Ce système est appelé *commande optimale de voile*. Nous allons voir dans les sections qui suivent les méthodes pour la conception des systèmes 1 et 2.

8.2 Régulateur pour le cap et l'ouverture de voile (système 1)

Dans cette section, nous traiterons de la réalisation du système 1 (*cf.* figure 8.2), un régulateur qui impose à notre bateau un cap $\hat{\theta}$ et une ouverture de voile $\hat{\delta}_v$. La technique que nous utilisons dans le cadre de la commande de systèmes non linéaires est la *linéarisation par retour d'état* (voir [Isidory, 1995] et [Marino and Tomei, 1995]). Cette méthode est répartie en deux étapes que nous rappelons brièvement.

– **1ère Étape :** cela consiste à dériver par rapport au temps successivement les variables d'états jusqu'à ce que celles-ci dépendent linéairement des entrées u_1 et u_2 du système. Nous obtenons le système suivant

$$\left\{\begin{array}{rcl} \dddot{x} & = & \dddot{v}\cos\theta - 2\dot{v}\dot{\theta}\sin\theta \\ & & -v\ddot{\theta}\sin\theta - v\dot{\theta}^2\cos\theta \\ \dddot{y} & = & \dddot{v}\sin\theta + 2\dot{v}\dot{\theta}\cos\theta + \\ & & v\ddot{\theta}\cos\theta - v\dot{\theta}^2\sin\theta \\ \ddot{\theta} & = & \ddot{\omega}. \end{array}\right. \tag{8.2}$$

Notons que les équations (8.2) peuvent s'exprimer, grâce aux équations (8.1), en fonction de l'état $\mathbf{x} = (x, y, \theta, \delta_v, \delta_g, v, \omega)^{\mathrm{T}}$ et des entrées du bateau $u_1 = \dot{\delta}_v$ et $u_2 = \dot{\delta}_g$.

– **2ème Étape :** Rappelons que nous souhaitons asservir le cap et l'ouverture de voile de notre bateau. Pour cela nous choisissons comme sorties de notre système $y_1 = \delta_v$ et $y_2 = \theta$. D'après les équations (8.1) et (8.2), Il vient

$$\begin{pmatrix} \dot{y}_1 \\ \dddot{y}_2 \end{pmatrix} = \begin{pmatrix} \dot{\delta}_v \\ \dddot{\theta} \end{pmatrix} = (\mathbf{A}_1 + \mathbf{A}_2\mathbf{A}_3)\,\mathbf{u} + (\mathbf{A}_2\mathbf{b}_2 + \mathbf{b}_1) \tag{8.3}$$

avec

$$\mathbf{A}_1(\mathbf{x}) = \begin{pmatrix} 1 & 0 \\ \frac{r_v f_v \sin\delta_v}{J} & \frac{r_g f_g \sin\delta_g}{J} \end{pmatrix}, \tag{8.4}$$

$$\mathbf{A}_2(\mathbf{x}) = \begin{pmatrix} 0 & 0 \\ \frac{\ell - r_v\cos\delta_v}{J} & -\frac{r_g\cos\delta_g}{J} \end{pmatrix}, \tag{8.5}$$

$$\mathbf{A}_3(\mathbf{x}) = \begin{pmatrix} -\alpha_v\left(V\sin(\theta+\delta_v) + v\cos\delta_v\right) & 0 \\ 0 & \alpha_g v\cos\delta_g \end{pmatrix}, \tag{8.6}$$

$$\mathbf{b}_1(\mathbf{x}) = \begin{pmatrix} 0 \\ -\frac{\alpha_\theta\dot{\omega}}{J} \end{pmatrix}. \tag{8.7}$$

et

$$\mathbf{b}_2(\mathbf{x}) = \begin{pmatrix} -\alpha_v\left(V\omega\sin(\theta+\delta_v) + \dot{v}\sin\delta_v\right) \\ \alpha_g\dot{v}\sin(\delta_g) \end{pmatrix}. \tag{8.8}$$

Posons $\mathbf{A}(\mathbf{x}) = \mathbf{A}_1 + (\mathbf{A}_2.\mathbf{A}_3)$ et $\mathbf{b}(\mathbf{x}) = \mathbf{b}_1 + (\mathbf{A}_2.\mathbf{b}_2)$, il vient

$$\begin{pmatrix} \dot{y}_1 \\ \dddot{y}_2 \end{pmatrix} = \mathbf{A}(\mathbf{x})\,\mathbf{u} + \mathbf{b}(\mathbf{x}). \tag{8.9}$$

Soit une consigne $\mathbf{v} = (v_1, v_2)^{\mathrm{T}}$ tel que $\dot{y}_1 = v_1$ et $\dddot{y}_2 = v_2$. D'après l'équation (8.9), nous avons

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x})\left(\mathbf{v} - \mathbf{b}(\mathbf{x})\right). \tag{8.10}$$

Nous disposons des matrices $\mathbf{A}(\mathbf{x})$ et $\mathbf{b}(\mathbf{x})$, reste maintenant à déterminer $\mathbf{v}$. Pour cela, revenons aux équations différentielles $\dot{y}_1 = v_1$ et $\dddot{y}_2 = v_2$, ces équations décrivent respectivement des systèmes linéaires d'ordre 1 et 2. En introduisant des états intermédiaires, le second système se décompose comme suit,

$$\dot{a}_1 = v_2,\ \dot{a}_2 = a_1 \text{ et } \dot{y}_2 = a_2. \tag{8.11}$$

Désignons par $w_1 = \hat{\delta}_s$ et $w_2 = \hat{\theta}$ les consignes pour les sorties $y_1 = \delta_s$ et $y_2 = \theta$. Les deux systèmes sont régulés par retour d'état et placement de pôles de façon à avoir des pôles égaux à -1. Si w_1 et w_2 constituent les entrées des systèmes régulés, alors nous avons $v_1 = w_1 - \alpha y_1$ et $v_2 = w_2 - l_1a_1 - l_2a_2 - l_3y_2$, leurs polynômes caractéristiques sont respectivement

$$\begin{aligned} P_1(s) &= s + \alpha, \\ P_2(s) &= s^3 + l_1s^2 + l_2s + l_3. \end{aligned} \tag{8.12}$$

Les conditions pour le placement de tous les pôles à -1 entraînent $P_1(s) = s + 1$ et $P_2(s) = (s+1)^3 = s^3 + 3s^2 + 3s + 1$, soit $\alpha = l_3 = 1$, $l_1 = l_2 = 3$. Les équations d'état du régulateur par retour d'état pour notre système non linéaire deviennent

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x}) \left(\begin{pmatrix} w_1 - \delta_v \\ w_2 - \theta - 3\dot{\theta} - 3\ddot{\theta} \end{pmatrix} - \mathbf{b}(\mathbf{x}) \right). \tag{8.13}$$

Or d'après les équations (8.1), $\dot{\theta}$ et $\ddot{\theta}$ sont des fonctions analytiques de l'état $\mathbf{x}$. D'où

$$\mathbf{u} = \mathbf{r}\left(\mathbf{x}, \mathbf{w}\right) = \mathbf{r}(\mathbf{x}, \hat{\delta}_v, \hat{\theta}). \tag{8.14}$$

Le schéma du régulateur obtenu par retour d'état est donné par la figure 8.3.

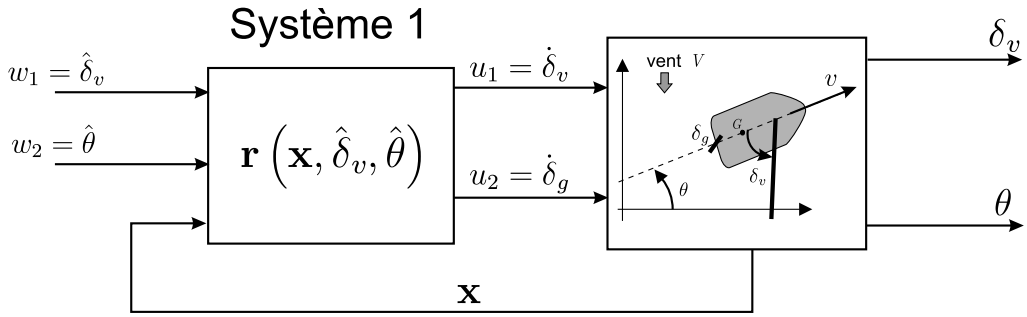

Figure 8.3 – Régulateur du cap et de l'ouverture de la voile.

Remarque 8.1. *Le régulateur obtenu en (8.14) génère des situations de singularités lorsque* $\det(\mathbf{A}(\mathbf{x}))$ *est nul. Dans un cadre expérimental, il faudra éviter les configurations du bateau (cap et ouverture de voile) qui mènent à ces singularités.*

8.3 Réglage optimal de l'ouverture de voile (système 2)

Afin de compléter la réalisation du système de commande, il nous vient une question à savoir : comment régler l'ouverture de voile en fonction du cap ? Soit de concevoir une fonction $\delta_v = \varphi(\theta)$ qui détermine une ouverture de voile δ_v pour chaque cap fixé θ. Un des objectifs principaux du *skipper*[2] est de régler l'ouverture de voile pour aller le plus vite possible sur l'eau, pour un cap donné. Cette section est consacrée à ce problème. En régime de fonctionnement permanent, la vitesse du bateau, son cap, sa vitesse angulaire, . . . , sont constants, c'est à dire,

$$\dot{\theta} = 0, \dot{\delta}_v = 0, \dot{\delta}_g = 0, \dot{v} = 0, \dot{\omega} = 0. \tag{8.15}$$

[2] terme anglais désignant celui qui tiend la barre d'un navire.

Des équations (8.1), il vient

$$\left\{\begin{array}{rcl} \alpha_v \left(V\cos\left(\theta+\delta_v\right)-v\sin\delta_v\right)\sin\delta_v-\alpha_g v\sin\delta_g\sin\delta_g-\alpha_f v & = & 0 \\ \left(\ell-r_v\cos\delta_v\right)\alpha_v\left(V\cos\left(\theta+\delta_v\right)-v\sin\delta_v\right)-r_g\cos\delta_g\alpha_g v\sin\delta_g & = & 0. \end{array}\right. \tag{8.16}$$

De plus, v est maximal lorsque la force de poussée,

$$P(\delta_v)=\left(V\cos\left(\theta+\delta_v\right)-v\sin\delta_v\right)\sin\delta_v \tag{8.17}$$

est maximale. Ainsi $\frac{dP(\delta_v)}{d\delta_v}=0$, ce qui équivaut à

$$\left(V\sin\left(\theta+\delta_v\right)+2v\cos\delta_v\right)\tan\delta_v-V\cos\left(\theta+\delta_v\right)=0. \tag{8.18}$$

Toutes ces conditions (régime permanent et force de poussée maximale) peuvent s'écrire sous la forme

$$\mathbf{g}\left(\theta,\delta_v,\delta_g,v\right)=\mathbf{0}, \tag{8.19}$$

où $\mathbf{g}\left(\theta,\delta_v,\delta_g,v\right)$ est donnée par

$$\left(\begin{array}{c} \alpha_v\left(V\cos\left(\theta+\delta_v\right)-v\sin\delta_v\right)\sin\delta_v-\alpha_g v\sin^2\delta_g-\alpha_f v \\ \alpha_v\left(\ell-r_v\cos\delta_v\right)\left(V\cos\left(\theta+\delta_v\right)-v\sin\delta_v\right)-r_g\alpha_g v\cos\delta_g\sin\delta_g \\ \left(V\sin\left(\theta+\delta_v\right)+2v\cos\delta_v\right)\tan\delta_v-V\cos\left(\theta+\delta_v\right) \end{array}\right). \tag{8.20}$$

Considérons l'ensemble

$$\mathbb{W}=\left\{(v,\theta)\mid\exists\delta_v,\ \exists\delta_g,\ \mathbf{g}\left(\theta,\delta_v,\delta_g,v\right)=\mathbf{0}\right\}, \tag{8.21}$$

qui correspond à la projection sur le plan (v,θ) des configurations possibles du bateau en régime permanent et pour une force de poussée maximale. Rappelons que PROJ2D nous permet de caractériser la projection en 2 dimensions d'un ensemble défini par des inégalités non linéaires. Ainsi, la représentation de cet ensemble est obtenue grâce à PROJ2D sur la figure 8.4. Pour chaque cap θ, il existe au moins deux triplets $\left(v^1,\delta_v^1,\delta_g^1\right)$ et $\left(v^2,\delta_v^2,\delta_g^2\right)$ tels que $\mathbf{g}\left(\theta,v,\delta_v,\delta_g\right)=\mathbf{0}$. Ces situations traduisent pour la force de poussée $P\left(\delta_v\right)$ des maximums ou minimums qui peuvent être locaux ou globaux.

Sur la figure 8.4, nous observons globalement 2 courbes. Pour le même cap θ, la vitesse sur la courbe de droite est plus élevée que celle de la courbe de gauche. En effet, la courbe des faibles vitesses à gauche correspond aux situations où l'angle du gouvernail δ_g est proche de $-\frac{\pi}{2}$ ou de $\frac{\pi}{2}$, bien que $\frac{dP(\delta_v)}{d\delta_v}=0$, la force de traînée du gouvernail est alors suffisante pour ralentir le bateau. Pour cela, la situation à laquelle on s'intéresse est celle de la courbe de droite sur la figure 8.4.

Pour chaque cap, la courbe correspondant aux vitesses maximales atteintes par le bateau est définie par,

$$\begin{array}{rcl} \partial\mathbb{W} & = & \left\{(v,\theta)\mid\exists\delta_v,\exists\delta_g,\ \mathbf{g}\left(\theta,\delta_v,\delta_g,v\right)=\mathbf{0},\right. \\ & & \left.\forall v'>v,\ \forall\delta_v',\forall\delta_g',\ \mathbf{g}\left(\theta,\delta_v',\delta_g',v'\right)\neq\mathbf{0}\right\}. \end{array} \tag{8.22}$$

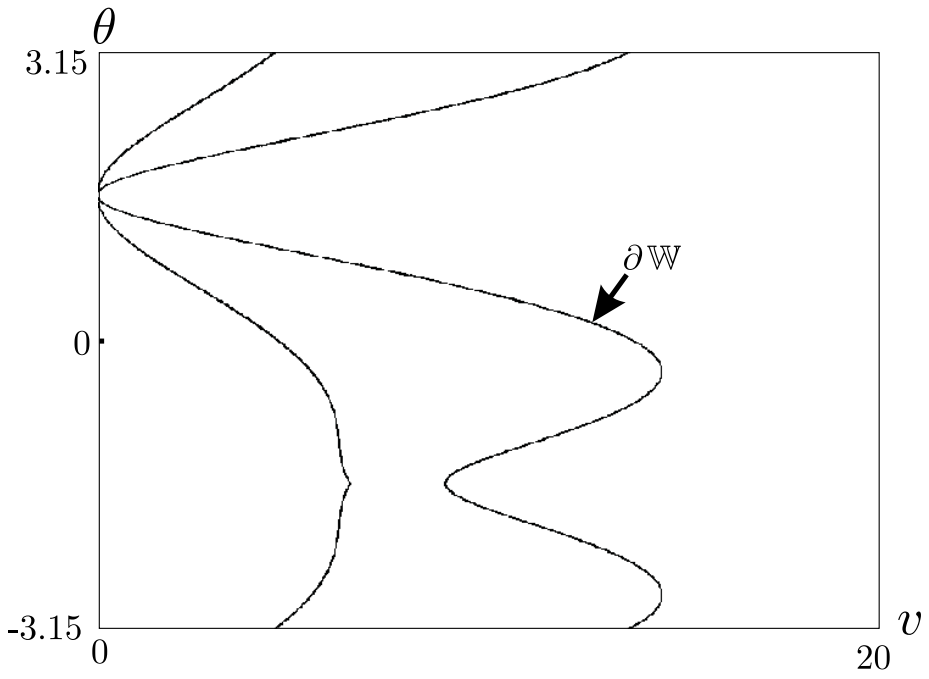

Figure 8.4 – Caractérisation de $\mathbb{W}$.

Une approximation de $\partial\mathbb{W}$ peut être obtenue en utilisant une technique d'interpolation. Ici, nous obtiendrons cette approximation en restreignant l'angle du gouvernail δ_g entre -0.3 et 0.3 radians. Dans cette intervalle, chaque valeur d'ouverture du gouvernail permet de stabiliser le cap du bateau pour une ouverture de voile donnée et une vitesse maximale. Par conséquent, nous avons

$$\begin{aligned} \partial\mathbb{W} \quad = \quad & \{(v,\theta) \,|\, \exists\, (\delta_v, \delta_g) \in [\delta_v] \times [\delta_g]\,, \\ & \mathbf{g}\,(\theta, \delta_v, \delta_g, v) = \mathbf{0}\}\,, \end{aligned} \tag{8.23}$$

avec $[\delta_v] = \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ et $[\delta_g] = [-0.3, 0.3]$.

Grâce aux changements de variables

$$v = \sqrt{x^2 + y^2} \text{ et } \tan(\theta) = \frac{y}{x} \tag{8.24}$$

nous obtenons (voir figure 8.5), la représentation de $\partial\mathbb{W}$ en coordonnées polaires appelée aussi *la polaire des vitesses*. Sur ce diagramme, les lignes radiales indiquent les directions pour le cap, tandis que les cercles concentriques représentent les niveaux de vitesse. Conformément aux directions indiquées sur cette courbe, nous pouvons voir que les plus grandes vitesses de notre bateau sont atteintes, en vent arrière, pour les deux caps θ_1 et θ_2. Les constructeurs de voilier font figurer avec les caractéristiques du bateau, des diagrammes polaires qui indiquent les vitesses maximales en fonction du cap et de l'intensité

du vent. Ces types de graphiques permettent de sélectionner le meilleur compromis cap-vitesse pour une force du vent donnée. Si le voilier avance trop contre le vent, il perd de la vitesse. S'il progresse dans une direction plus rabattue, il avancera plus vite mais perdra en cap.

De plus, une comparaison entre le diagramme de la figure 8.5 et les polaires de vitesses obtenues par des simulations numériques (programmes VPP : Velocity Prediction Program) permet *a fortiori* de valider le modèle dynamique (8.1).

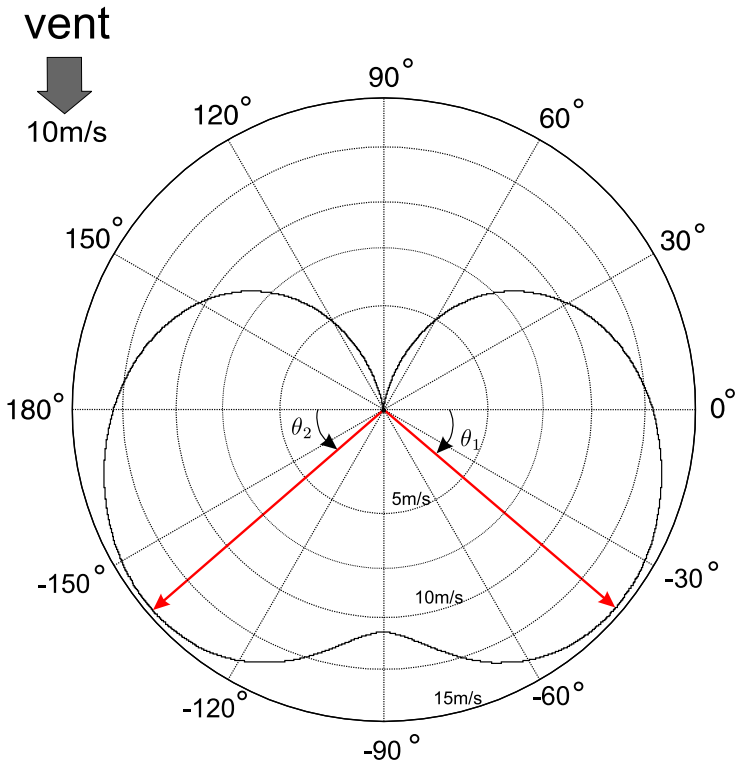

Figure 8.5 – Polaire des vitesses : courbe polaire des vitesses maximales en fonction du cap.

Nous rappelons que l'une des consignes de sortie du système 2 correspond à l'orientation désirée θ pour le bateau. Dans la suite, nous verrons que les configurations (cap, vitesse) représentées sur la polaire des vitesses peuvent être atteintes, si l'ouverture de voile δ_v est convenablement réglée. Pour cela définissons, *l'ouverture optimale de voile* comme,

$$\begin{aligned} \mathbb{S} \;=\; & \{(\theta,\delta_v)\,|\,\exists v,\exists\delta_g,\ \mathbf{g}\,(\theta,\delta_v,\delta_g,v)=\mathbf{0}, \\ & \forall v'>v,\ \forall\delta'_v,\forall\delta'_g,\ \mathbf{g}\left(\theta,\delta'_v,\delta'_g,v'\right)\neq\mathbf{0}\}\,. \end{aligned} \tag{8.25}$$

Noter que si le couple (θ,δ_v) appartient à $\mathbb{S}$, lorsque le bateau évolue avec un cap θ on se situe dans une configuration où l'ouverture de voile doit être réglée sur l'angle δ_v pour aller aussi rapidement que possible. N'importe quel autre angle de voile ralentirait le bateau. Nous avons,

$$\begin{aligned}
\mathbb{S} &= \{(\theta,\delta_v)\,|\exists v,\exists\delta_g,\mathbf{g}\,(\theta,\delta_v,\delta_g,v)=\mathbf{0}, \\
&\quad \forall v'>v,\forall\delta'_v,\forall\delta'_g,\mathbf{g}\left(\theta,\delta'_v,\delta'_g,v'\right)\neq\mathbf{0}\} \\
&= \{(\theta,\delta_v)\,|\exists v,(\exists\delta_g,\mathbf{g}\,(\theta,\delta_v,\delta_g,v)=\mathbf{0}) \\
&\quad (\exists\delta_v,\exists\delta_g,\mathbf{g}\,(\theta,\delta_v,\delta_g,v)=\mathbf{0}\,, \\
&\quad \forall v'>v,\forall\delta'_v,\forall\delta'_g,\mathbf{g}\left(\theta,\delta'_v,\delta'_g,v'\right)\neq\mathbf{0})\} \\
&= \{(\theta,\delta_v)\,|\exists v,\exists\delta_g,\mathbf{g}\,(\theta,\delta_v,\delta_g,v)=\mathbf{0},\ (v,\theta)\in\partial\mathbb{W}\}\,. \qquad (8.26)
\end{aligned}$$

Sur la figure 8.6, les deux tronçons de courbe représentent une approximation de $\mathbb{S}$.

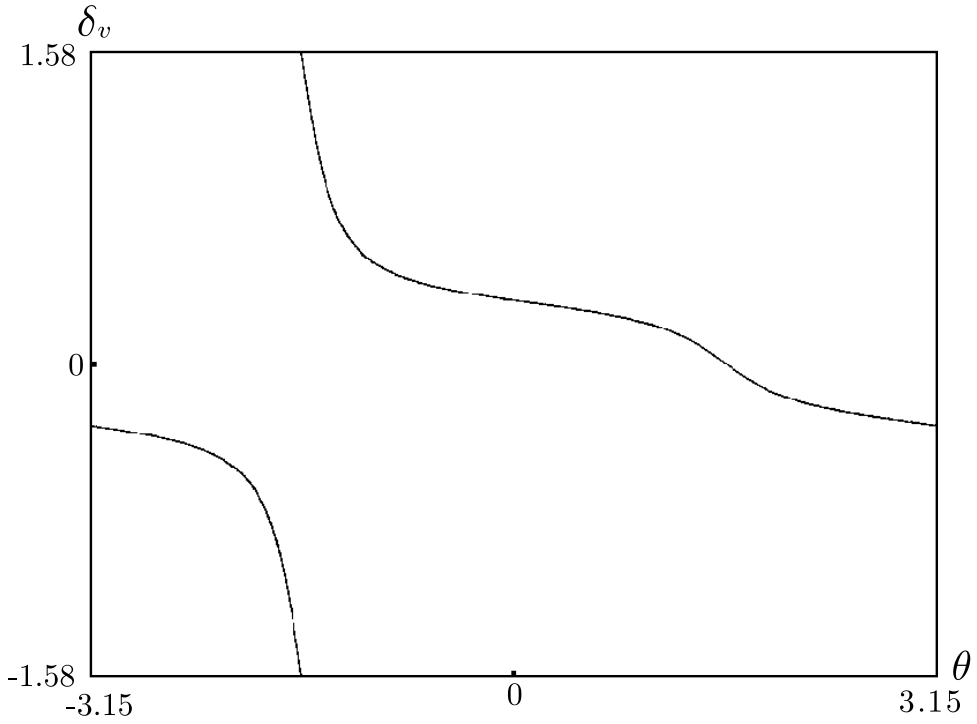

Figure 8.6 – Caractérisation de $\mathbb{S}$, l'ouverture de voile δ_v choisi en fonction du cap θ, dans le but d'atteindre une vitesse maximale.

La commande d'ouverture de la voile $\delta_v=\varphi^*(\theta)$ est définie par la fonction 2π - périodique

$$\varphi^*(\theta)=f\left(\Phi\left(\theta-\frac{\pi}{2}\right)+\frac{\pi}{2}\right), \qquad (8.27)$$

où $\Phi(x)\in[0,2\pi]$ est une fonction qui donne pour chaque variable $x\in\mathbb{R}$, une équivalence angulaire entre 0 et 2π radians. Il vient

$$\Phi(x)=x-2\pi.\mathrm{E}\left(\frac{x}{2\pi}\right), \qquad (8.28)$$

où $\mathrm{E}(a)$ correspond à la partie entière de a.

La fonction $f(x)$ est déterminée par une interpolation polynomiale sur un échantillon de points issus de $\mathbb{S}$. Nous obtenons

$$f(x) = \sum_{i=0}^{9} a_i x^i, \tag{8.29}$$

avec

i	0	1	2	3	4
a_i	0.3024	-0.1354	0.1397	0.05952	-0.2229

...

5	6	7	8	9
0.0346	0.07175	-0.03861	$7.434\mathrm{E}^{-3}$	$-5.077\mathrm{E}^{-4}$

Au final, nous avons

$$\delta_v = \varphi^*(\theta) = \sum_{i=0}^{9} a_i \left(\theta - 2\pi.\mathrm{E}\left(\frac{\theta}{2\pi} - \frac{1}{4}\right)\right)^i. \tag{8.30}$$

8.4 Simulations et résultats

Un bateau à voile n'admet aucun point d'équilibre si la vitesse du vent est différente de zéro ($V \neq 0$m/s), cela signifie qu'il est impossible d'immobiliser sur l'eau un bateau à voile lorsqu'il y a du vent. Par contre, il est tout à fait concevable de confiner la position du bateau à l'intérieure d'une zone de l'océan. En effet, lorsque le bateau dépasse les limites d'une zone fixée autour de la balise, des brusques changements de cap (opérations d'empannages) permettent alors de ramener la position du bateau près de la balise. Lors de nos simulations, les changements de cap du bateau sont gérés par un automate. Au lieu d'obtenir une stabilisation exacte, le bateau se met à tourner autour de notre balise sur l'océan.

Pour montrer l'efficacité de la fonction d'ouverture de voile optimale $\delta_v = \varphi^*(\theta)$, nous présentons des résultats de simulations[3] numériques sur SCILAB©. Les expériences sont choisies, afin de tester le système régulé (voir figure 8.2) avec différentes commandes d'ouverture de voile, sur différentes trajectoires.

Pour les besoins de nos comparaisons, nous analysons les comportements de deux bateaux munis de fonctions d'ouverture de voile différentes. Le premier bateau (bateau 1) possède une ouverture de voile, en fonction du cap, déterminée précédemment par (8.30), il vient

$$\varphi_1(\theta) = \sum_{i=1}^{9} a_i.g(\theta)^i. \tag{8.31}$$

[3]Les codes sources sont disponibles sur `http://www.istia.univ-angers.fr/~dao/boat_simul.sce`.

Le second bateau (bateau 2) est muni d'une commande d'ouverture de voile proposée dans [Jaulin, 2005], cette commande est supposée linéaire, formellement

$$\varphi_2(\theta) = a.g(\theta) + b \tag{8.32}$$

avec $g(\theta) = \Phi\left(\theta - \frac{\pi}{2}\right) + \frac{\pi}{2}$, $\forall\, \theta \in \mathbb{R}$, Φ est la fonction donnée par (8.28).

Notons (x_d, y_d), la position de notre balise sur l'océan. Afin de quantifier les performances des 2 bateaux, nous allons mesurer au cours du temps des distances $d_1(t)$ et $d_2(t)$ entre les bateaux et la balise de référence. Ainsi, pour le bateau i, nous avons

$$d_i(t) = \sqrt{(x_d - x_i(t))^2 + (y_d - y_i(t))^2} \tag{8.33}$$

où $(x_i(t), y_i(t))$ est la position du centre d'inertie G (voir figure 8.1) du bateau i (avec $i \in \{1, 2\}$). A l'instant $t = 0$, les deux bateaux ont les mêmes conditions initiales :

$$x_0 = 0\text{m},\ y_0 = -50\text{m},\ \theta_0 = -0.5\text{rd},\ \delta_{v0} = 0\text{rd}, \tag{8.34}$$

et

$$\delta_{g0} = 0\text{rd},\ v_0 = 2\text{m.s}^{-1},\ \omega_0 = 0\text{rd.s}^{-1}. \tag{8.35}$$

8.4.1 Stabilisation autour d'une balise statique

Ici, la balise est supposée statique sur l'océan. Les variations des distances $d_1(t)$ et $d_2(t)$ sont analysées dans les deux cas suivants.

Le premier cas concerne la stabilisation autour d'une balise dont la position est repérée par $x_d = 50\text{m}$ et $y_d = 150\text{m}$. Pour atteindre la balise, les bateaux 1 et 2 évoluent en situation de vent de face, c'est à dire que l'ouverture de voile ainsi que le cap sont ajustés afin de remonter le vent. Les résultats de la simulation sont indiqués sur la figure 8.7. Comme attendu, nous remarquons globalement que la distance $d_1(t)$ décroît plus vite que $d_2(t)$. Ainsi, pour le même état initial, le bateau 1 rejoint la balise plus vite que le bateau 2. Au bout d'environ 90s le bateau 1 atteint la balise, tandis que le bateau 2 met environ 165s pour atteindre la même balise.

Dans le deuxième cas, la balise se situe en $x_d = -100\text{m}$ et $y_d = -400\text{m}$. Les deux bateaux se trouvent en situation de vent arrière, ils atteignent plus rapidement la balise que dans le premier cas (voir figure 8.8). A l'instar de ce cas, le bateau 1 se rapproche plus vite de la balise que le bateau 2 ($d_1(t)$ diminue plus vite que $d_2(t)$).

8.4.2 Suivi d'une balise mobile

Afin de corroborer les résultats obtenus ci-dessus, nous avons effectué des tests de simulations avec une balise mobile. Le déplacement de la balise est décrit par la courbe paramétrée $(x_d(t), y_d(t))$. Nous

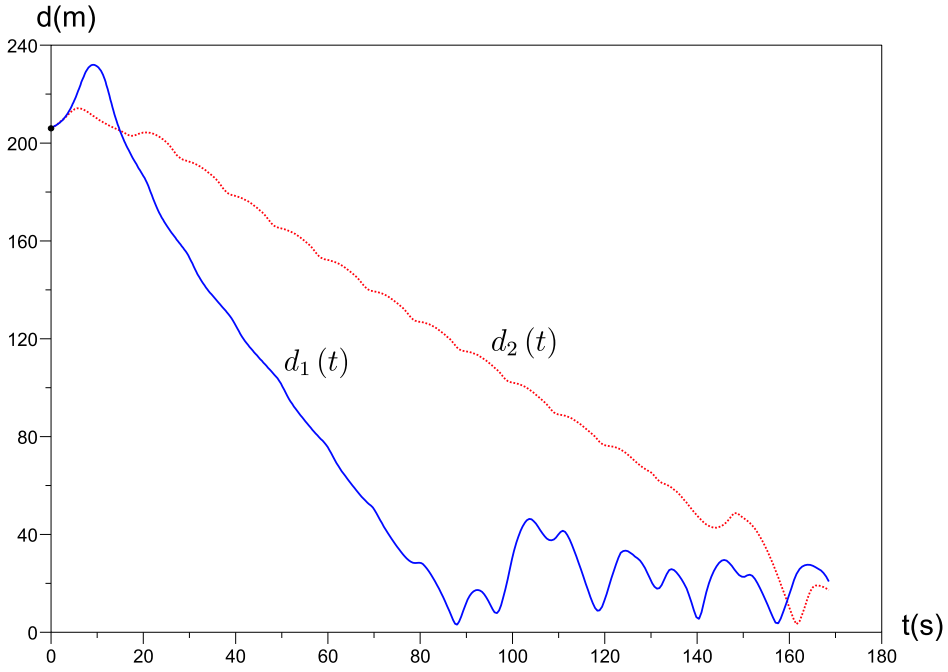

Figure 8.7 – Stabilisation autour d'une balise statique : variations des distances $d_1\,(t)$ et $d_2\,(t)$ pour les bateaux 1 et 2 en vent de face.

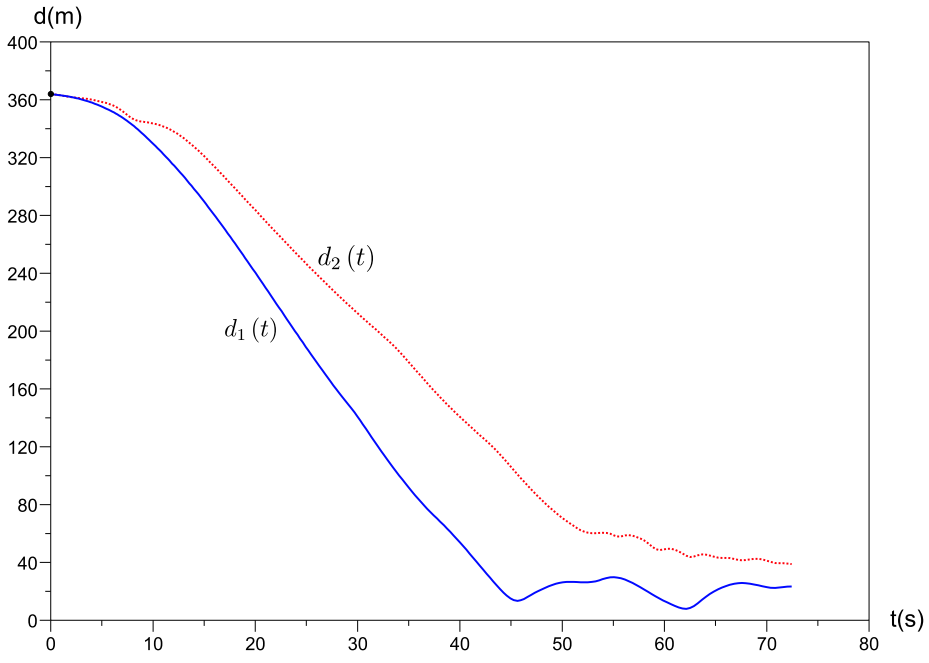

Figure 8.8 – Stabilisation autour d'une balise statique : variations de $d_1\,(t)$ et $d_2\,(t)$ pour les bateaux en vent arrière.

allons étudier les comportements (variations des distances $d_1(t)$ et $d_2(t)$) des deux bateaux sur deux types de trajectoires. Ces études feront l'objet des tests A et B.

8.4.2.1 Test A

Considérons la trajectoire de la balise donnée par,

$$x_d(t) = a.\cos(\alpha t) \text{ et } y_d(t) = b.\sin(\alpha t) \tag{8.36}$$

où a, b et $\alpha \in \mathbb{R}$. Ces différents paramètres sont choisis tels que,

$$a = 200\text{m}, b = 100\text{m et } \alpha = 0.03\text{rd.s}^{-1}. \tag{8.37}$$

L'allure d'une telle trajectoire est celle d'une ellipse. Nous avons d'une part la figure 8.9a qui illustre le suivi de la balise par les deux bateaux sur une ellipse. D'autre part la figure 8.9b permet d'analyser sur un horizon de temps, les évolutions des distances déterminées par (8.33) entre la balise $(x_d(t), y_d(t))$ et les bateaux respectifs 1 et 2.

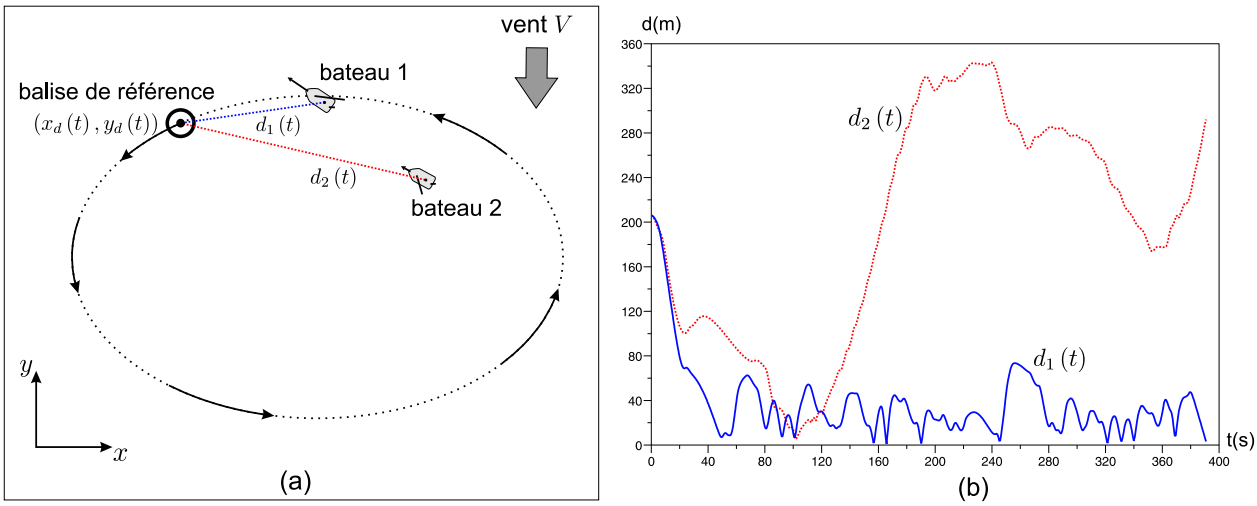

Figure 8.9 – (a) Suivi d'une trajectoire elliptique par les bateaux 1 et 2. (b) Variations par rapport au temps de $d_1(t)$ et $d_2(t)$.

A $t = 0$, les deux bateaux avec les mêmes conditions initiales ($\mathbf{x}_0 = (x_0, y_0, \theta_0, \delta_{v0}, \delta_{g0}, v_0, \omega_0)^{\mathrm{T}}$) sont distants de la balise d'environ 206m. Les variations de $d_1(t)$ et $d_2(t)$ observées sur la figure 8.9b indiquent que le bateau 1 reste le plus proche de la balise en mouvement. En effet, la distance $d_1(t)$ est globalement inférieure à $d_2(t)$. Lorsque la vitesse de la balise est élevée (α élevée), le bateau 2 n'arrive plus du tout à suivre la trajectoire et génère de multiples singularités pour notre régulateur. Ceci explique

des différences entre les amplitudes maximales des distances : $d_2(t)$ atteint 346 mètres tandis que $d_1(t)$ ne dépasse guère les 100 mètres.

8.4.2.2 Test B

Pour ce second test, la trajectoire de la balise est définie comme suit,

$$x_d(t) = a.\cos(\alpha t + \varphi_1) \text{ et } y_d(t) = b.\cos\left(\frac{\alpha}{2}t + \varphi_2\right) \tag{8.38}$$

avec a, b, α, φ_1 et $\varphi_2 \in \mathbb{R}$. Ces paramètres sont fixés comme suit,

$$a = 200\text{m}, b = 150\text{m et } \alpha = 0.05\text{rd.s}^{-1}, \tag{8.39}$$

$$\varphi_1 = 0\text{rd et } \varphi_2 = -1\text{rd}. \tag{8.40}$$

Un type particulier de cette trajectoire (φ_1 et φ_2 convenablement fixés) est représenté par la figure 8.10a. En ce qui concerne les distances $d_1(t)$ et $d_2(t)$ (voir figure 8.10b), elles semblent conforter la tendance selon laquelle : le bateau 1 se situe dans le voisinage le plus proche de la balise.

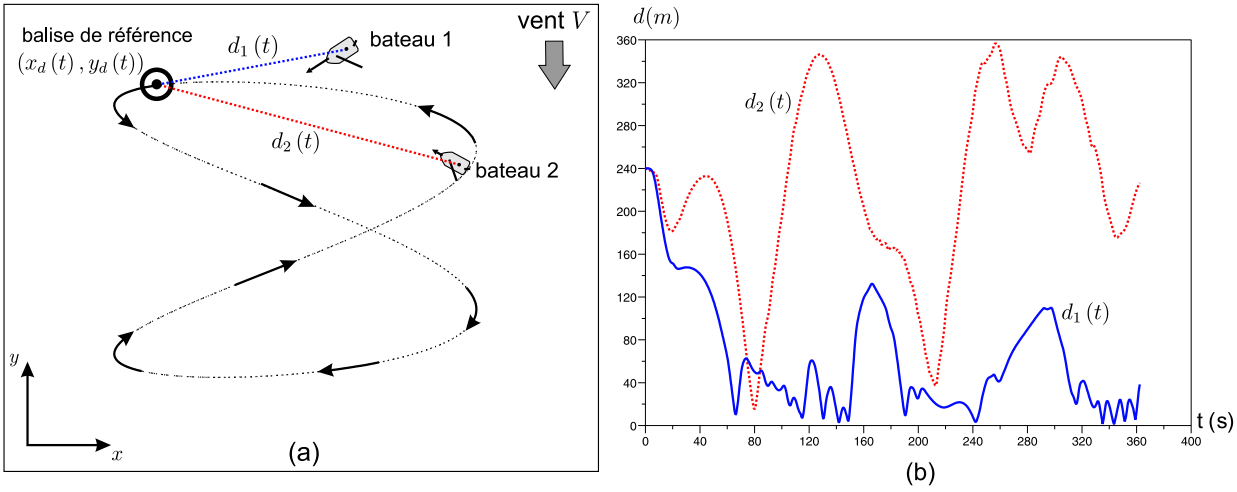

Figure 8.10 – (a) Suivi d'une trajectoire de type lissajou par les 2 bateaux. (b) Evolutions de $d_1(t)$ et $d_2(t)$.

8.5 Conclusion

La contribution apportée dans ce chapitre est l'utilisation de méthodes ensemblistes (projection d'ensembles) pour la conception d'un système de commande pour bateau à voile. Au cours de notre travail,

nous avons déterminé dans un premier temps, un régulateur pour le cap et l'ouverture de voile. Puis dans un deuxième temps, les performances de cette régulation ont été améliorées en terme de vitesse. En effet, nous avons proposé comme commande de voile optimale un polynôme de degré 9. Cette loi de commande a été présentée dans le cadre du suivi d'une balise, les simulations par ordinateur ont été effectuées à partir du modèle dynamique du bateau. Nous avons considéré un premier bateau équipé d'une commande d'ouverture de voile optimale, puis un second bateau muni d'une fonction d'ouverture de voile linéaire. Sur un horizon de temps, l'efficacité de notre dispositif de commande a été analysée grâce aux variations des distances entre les bateaux et la balise. Rappelons que tout ceci a été effectué dans le contexte d'un système fortement non linéaire et d'un vent constant. De plus, l'utilisation des approches intervalles est rendue possible à cause du faible nombre de paramètres (v, θ, δ_v et δ_g) qui interviennent dans les équations d'état du bateau.

D'un point de vue strictement technique, le dispositif composé du bateau à voile et du système de commande constitue un robot autonome qui se déplace sur l'eau. Il devient alors possible de concevoir un bateau à voile équipé de plaques solaires, de balises GPS pour la localisation, et de deux moteurs électriques. Les plaques solaires alimentent les moteurs afin de commander les ouvertures de voile et du gouvernail. Ainsi, le bateau pourrait être téléguidé à partir d'une station qui se situe sur terre. Beaucoup d'applications peuvent être imaginées pour un tel système : la surveillance côtière, le prélèvement de mesures dans un environnement marin pour la recherche scientifique, *etc*.

CHAPITRE 9

Conception de filtres par des méthodes intervalles

Les filtres sont des éléments importants dans le domaine du traitement du signal. A ce titre, il est intéressant de réaliser les filtres les plus simples possibles répondants à des caractéristiques données sur le gain et la phase. Nous entendons par filtre simple, un filtre dont la fonction de transfert dépend d'un faible nombre de coefficients. En effet, plus l'ordre du filtre est élevé, plus sa réalisation technique utilise de composants. Ainsi, le problème posé est double. D'une part, nous cherchons un filtre compatible avec des caractéristiques fréquentielles spécifiques (contraintes sur le gain et la phase). D'autre part, l'ordre de ce filtre doit être le plus faible possible.

L'originalité du travail effectué réside essentiellement dans la façon de poser le problème d'estimation d'un filtre d'ordre minimal. En des termes plus détaillés, ce problème consiste à trouver le plus petit ordre n^* au dessous duquel il n'existe aucun filtre satisfaisant nos contraintes (gabarit sur le gain et la phase). Nous abordons ce problème suivant deux étapes. La première étape consiste à montrer que les ensembles de filtres respectivement d'ordre $1, 2, ..., n^* - 1$ sont vides. La deuxième étape est consacrée à l'estimation d'un filtre d'ordre n^* qui respecte notre gabarit.

Un algorithme de projection d'ensembles basé sur le calcul par intervalles permet de répondre au problème de la première étape. En ce qui concerne la deuxième étape, les algorithmes ensemblistes s'avèrent très peu efficaces pour caractériser des filtres impliquant un grand nombre de coefficients (> 5). Nous proposerons une méthode combinant le calcul par intervalles et une stratégie de recherche d'une solution ponctuelle. Ainsi la conception de filtres pose le problème de la résolution d'un système d'inégalités non linéaires.

Ce chapitre est organisé de la façon suivante. Dans la section 9.1, nous exposerons en détails le problème de conception de filtre d'ordre minimal. Cette section présente les notations et les principes utilisés pour résoudre notre problème. La section 9.2 est consacrée à la mise en oeuvre d'une méthode de recherche locale basée sur le calcul par intervalles. La section 9.3 propose les résultats obtenus à l'issue de la conception de plusieurs filtres aussi bien analogique que numérique. Enfin la section 9.4 fait état des avantages, des inconvénients et des perspectives d'applications concernant les méthodes mises au point.

9.1 Contexte

Tout comme il a été souligné dans l'introduction, cette section a pour objet la présentation des caractéristiques fréquentielles d'un filtre et des notions associées au problème d'estimation de filtre d'ordre minimal.

9.1.1 Gain et phase d'un filtre

Considérons la fonction de transfert complexe d'un filtre analogique ou numérique d'ordre n,

$$H(x) = \frac{N(x)}{D(x)} = \frac{\sum_{p=0}^{m} a_p x^p}{1 + \sum_{k=1}^{n} b_k x^k}, \tag{9.1}$$

avec $x \in \mathbb{C}$. Notons $\mathbf{a} = (a_0, a_1, ..., a_m)^{\mathrm{T}}$ et $\mathbf{b} = (b_1, b_2, ..., b_n)^{\mathrm{T}}$ avec $(m \leq n)$ respectivement les vecteurs coefficients du numérateur et du dénominateur de la fonction de transfert. Nous avons, dans le cas d'un filtre numérique : $x \equiv z = e^{j2\pi\nu}$, pour un filtre analogique $x \equiv s = j2\pi\nu$. Ici ν (Hz) correspond à la fréquence des signaux.

Le gain en fréquence de $H(x)$ est donné par $g(\nu) = |H(x)|$, la phase s'écrit

$$\varphi(\nu) = \arctan\left(\frac{\mathrm{Im}(N(x))}{\mathrm{Re}(N(x))}\right) - \arctan\left(\frac{\mathrm{Im}(D(x))}{\mathrm{Re}(D(x))}\right), \tag{9.2}$$

où $\mathrm{Re}(z)$ et $\mathrm{Im}(z)$ sont respectivement les parties réelle et imaginaire de z.

9.1.2 Estimation d'un filtre d'ordre minimal

Définissons l'ensemble des filtres (m, n) - acceptables (ou compatibles) noté $\mathbb{S}_{(n,m)}$, tous les filtres pour lesquels $\deg[N(x)] = m$ et $\deg[D(x)] = n$ et dont le gain puis la phase sont compatibles avec des contraintes de type inégalités (voir figure 9.1). Il vient

$$\mathbb{S}_{(m,n)} = \{(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^m \times \mathbb{R}^n |\ \mathbf{u}(\mathbf{a}, \mathbf{b}) \in [\mathbf{u}] \text{ et } \mathbf{w}(\mathbf{a}, \mathbf{b}) \in [\mathbf{v}]\}, \tag{9.3}$$

avec $[\mathbf{u}]$ et $[\mathbf{v}]$ des pavés de $\mathbb{R}^r$ et $\mathbb{R}^l$, $\mathbf{u}(\mathbf{a}, \mathbf{b})$ et $\mathbf{v}(\mathbf{a}, \mathbf{b})$ des fonctions vectorielles dont les composantes sont données par $u_i(\mathbf{a}, \mathbf{b}) = g(\nu_i)$ et $w_q(\mathbf{a}, \mathbf{b}) = \varphi(\nu'_q)$. Les paramètres ν_i et ν'_q ($i = 1, 2, ..., r$ et $q = 1, 2, ..., l$) correspondent à des échantillons de fréquences choisies.

Posons $\mathbf{f}(\mathbf{a}, \mathbf{b}) = (\mathbf{u}(\mathbf{a}, \mathbf{b}), \mathbf{v}(\mathbf{a}, \mathbf{b}))^{\mathrm{T}}$ et $[\mathbf{f}] = [\mathbf{u}] \times [\mathbf{v}]$, l'expression 9.3 devient

$$\mathbb{S}_{(m,n)} = \{(\mathbf{a}, \mathbf{b}) \in \mathbb{R}^m \times \mathbb{R}^n |\ \mathbf{f}(\mathbf{a}, \mathbf{b}) \in [\mathbf{f}]\}. \tag{9.4}$$

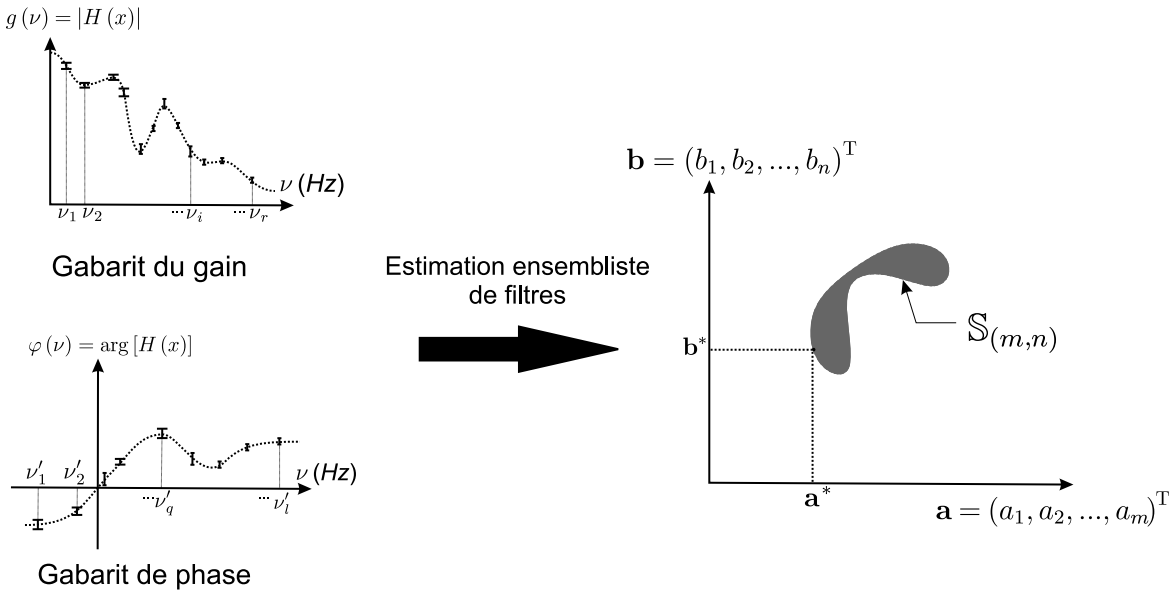

Figure 9.1 – Estimation ensembliste de filtres d'ordre n donné.

L'estimation d'un filtre d'ordre minimal revient à trouver un filtre de $\mathbb{S}_{(m^*,n^*)}$ tel que :

$$\mathbb{S}_{(i,k)} = \emptyset,\ \forall\, k < n^*,\ i < k \tag{9.5}$$

et

$$\mathbb{S}_{(i,n^*)} = \emptyset,\ \forall\, i < m^*. \tag{9.6}$$

Exemple 9.1. *On considère un gabarit de filtre pour lequel* $n^* = 3$ *et* $m^* = 1$. *Nous en déduisons :* $\mathbb{S}_{(0,1)} = \emptyset$, $\mathbb{S}_{(0,2)} = \emptyset$, $\mathbb{S}_{(1,2)} = \emptyset$ *et* $\mathbb{S}_{(0,3)} = \emptyset$.

9.2 Méthodes intervalles pour l'estimation de paramètres

Une première difficulté, non négligeable, posée par notre problème est de pouvoir démontrer qu'un ensemble défini par des inégalités est vide. La deuxième difficulté réside dans la recherche d'un filtre vérifiant un gabarit avec un grand nombre de coefficients. Face à cette situation, la plupart des méthodes classiques d'estimation de paramètres n'apportent, dans une certaine mesure, pas de réponse satisfaisante.

9.2.1 Approche classique, optimisation globale

La majorité des méthodes d'estimation de paramètres sont basées sur des techniques d'optimisation globale. Dans l'idéal, ces techniques sont censées minimiser une distance entre les points de deux ensembles caractérisés par des inégalités non linéaires (voir illustration en figure 9.2).

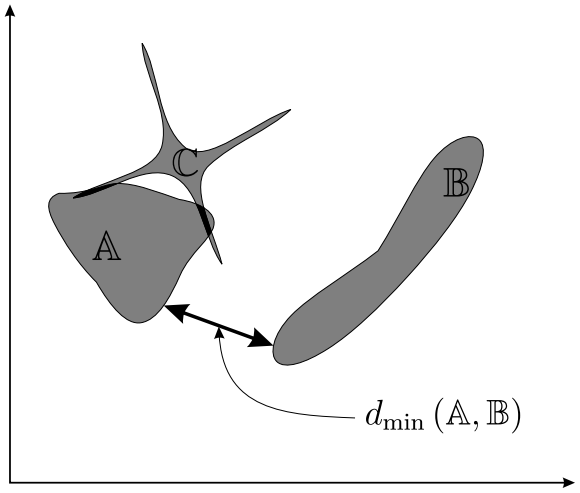

Figure 9.2 – Distances entre des ensembles.

Définissons une distance entre deux sous-ensembles $\mathbb{E}$ et $\mathbb{F}$ de $\mathbb{R}^n$ par :

$$d(\mathbb{E}, \mathbb{F}) = \min_{\substack{\mathbf{x}\in\mathbb{E} \\ \mathbf{y}\in\mathbb{F}}} \|\mathbf{x} - \mathbf{y}\|_{2,\infty} . \tag{9.7}$$

Dans le cadre d'une estimation classique de paramètres, les ensembles $\mathbb{E}$ et $\mathbb{F}$ sont décrits par des inégalités. Ainsi $\mathbf{x} \in \mathbb{E}$ correspond à $\mathbf{x} = \mathbf{f}(\mathbf{p}_1)$, de même $\mathbf{y} \in \mathbb{F}$ revient à $\mathbf{y} = \mathbf{g}(\mathbf{p}_2)$, avec les fonctions $\mathbf{f} : \mathbb{R}^{n_1} \to \mathbb{R}^m$ et $\mathbf{g} : \mathbb{R}^{n_2} \to \mathbb{R}^m$. L'expression 9.7 devient :

$$\begin{aligned} d(\mathbb{E}, \mathbb{F}) &= \min_{\mathbf{p}_1, \mathbf{p}_2} \|\mathbf{f}(\mathbf{p}_1) - \mathbf{g}(\mathbf{p}_2)\|_{2,\infty} \\ &= \min_{\mathbf{p}\in\mathbb{R}^n} j(\mathbf{p}) , \end{aligned} \tag{9.8}$$

avec $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2)^{\mathrm{T}}$, $j(\mathbf{p}) = \|\mathbf{f}(\mathbf{p}_1) - \mathbf{g}(\mathbf{p}_2)\|_{2,\infty}$ et $n = n_1 + n_2$. Nous rappelons ici que j est une norme définie de $\mathbb{R}^n$ vers $\mathbb{R}^+$. Réaliser une estimation du paramètre $\mathbf{p}$ revient à trouver un minimiseur global $\mathbf{p}^*$ tel que pour tout $\mathbf{p} \in \mathbb{R}^n$, $j(\mathbf{p}^*) \leq j(\mathbf{p})$. Les méthodes classiques d'optimisation globale dans des domaines continus ne donnent pas de solution dans le cas général. Ces méthodes ne s'appliquent qu'à des cas très restreints : existence d'un unique optimum (systèmes identifiables), nombre de paramètres estimés restreint ($n < 5$), *etc.* (voir [Walter, 1982], [Fliess et al., 2004] et [Walter and Pronzato, 1997]).

Sur la figure 9.2 la distance entre les ensembles $\mathbb{A}$ et $\mathbb{B}$ est $d(\mathbb{A}, \mathbb{B}) > 0$ car $\mathbb{A} \cap \mathbb{B} = \emptyset$, par contre $d(\mathbb{A}, \mathbb{C}) = 0$ car $\mathbb{A} \cap \mathbb{C} \neq \emptyset$ (les zones noires représentent l'intersection entre $\mathbb{A}$ et $\mathbb{C}$).

9.2.2 Estimation de paramètres, approches intervalles

Une méthode d'estimation à erreurs bornées désormais classique est l'algorithme `SIVIA`. Nous rappelons qu'un algorithme d'inversion ensembliste permet de caractériser l'ensemble

$$\mathbb{X} = \{ \mathbf{x} \in [\mathbf{x}]_0 |\ \mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \} = \mathbf{f}^{-1}([\mathbf{y}]), \tag{9.9}$$

où $\mathbf{f}$ est une fonction continue définie de $\mathbb{R}^n$ vers $\mathbb{R}^m$, $[\mathbf{y}]$ un pavé de $\mathbb{R}^m$ et $[\mathbf{x}]_0$ est un quelconque pavé. Cette approximation est faite par un sous-pavage (union de pavé) qui contient $\mathbb{X}$. La méthode `SIVIA` engendre des temps de calcul prohibitifs lorsque $\mathbb{X}$ est de grande dimension. Malgré cela, si $\mathbb{X} = \emptyset$, les techniques de consistance globale se révèlent efficaces. Dans notre cas, cela revient à montrer qu'un système d'inégalités, avec de nombreuses inconnues, n'admet aucune solution dans un compact.

Par la suite, nous proposons une méthode de recherche capable de trouver une solution à des inégalités, lorsque celle-ci existe. En termes ensemblistes, cela revient à la recherche d'un point appartenant à $\mathbb{X}$, lorsque celui-ci est un ensemble de grande dimension. Bien entendu, cette approche s'inspire du calcul par intervalles et des notions de la section 9.2.1. Rappelons que la proximité entre $[\mathbf{r}]$ et $[\mathbf{s}]$, deux pavés de $\mathbb{R}^n$, est définie par :

$$j_{\mathrm{p}}([\mathbf{r}], [\mathbf{s}]) = \max_{i=1,\ldots,n} (\max(\underline{r}_i - \underline{s}_i, \overline{s}_i - \overline{r}_i)). \tag{9.10}$$

Prenons l'ensemble $\mathbb{X}$ donné par (9.9). La technique que nous présentons consiste à utiliser comme critère de sélection, la proximité entre les pavés (9.10). Une illustration du procédé est donnée par la figure 9.3. A droite, dans l'espace image de la fonction $\mathbf{f}$, nous avons $[\mathbf{f}]([\mathbf{a}])$ et $[\mathbf{f}]([\mathbf{b}])$ les pavés images respectifs de $[\mathbf{a}]$ et $[\mathbf{b}]$. Les critères de proximités entre les pavés images et $[\mathbf{y}]$ sont représentés par : $d_1 = j_{\mathrm{p}}([\mathbf{f}]([\mathbf{a}]), [\mathbf{y}])$ et $d_2 = j_{\mathrm{p}}([\mathbf{f}]([\mathbf{b}]), [\mathbf{y}])$. A gauche, dans l'espace des antécédents (ou espace réciproque), le pavé $[\mathbf{b}]$ correspond à la zone de recherche sélectionnée car $d_2 < d_1$. Nous avons développé ci-après, un algorithme qui s'articule autour de ce critère de selection de pavé.

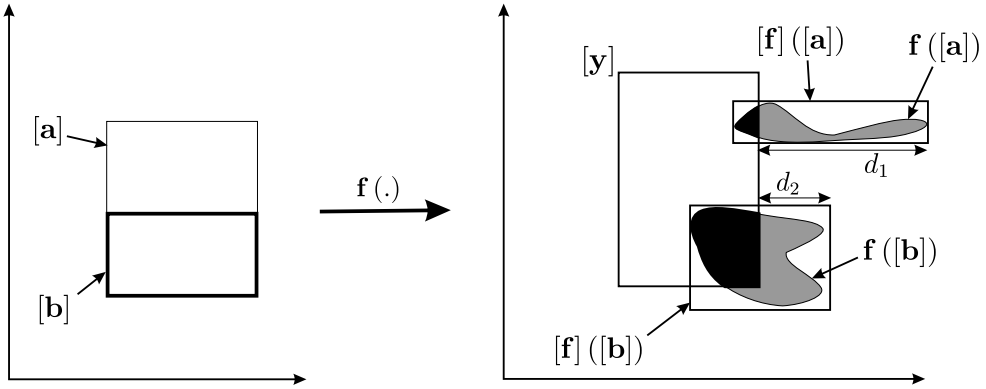

Figure 9.3 – Illustration de la sélection de pavé par le critère de proximité.

A présent, analysons le fonctionnement de notre algorithme de recherche locale. Tout d'abord, nous partitionnons le pavé initial $[\mathbf{x}]_0$ en deux pavés. Ensuite, le pavé dont l'image par $\mathbf{f}$ est *le plus proche* (au sens du critère de proximité) du pavé $[\mathbf{y}]$ est sélectionné. Le pavé ainsi obtenu sera de nouveau partitionné puis le procédé sera répété jusqu'à l'obtention d'un pavé pour lequel la proximité entre son image par $\mathbf{f}$ et le pavé $[\mathbf{y}]$ est inférieure ou égale à 0. Supposons que nous ayons à notre disposition un contracteur $\mathcal{C}_{\mathbb{X}}$ pour $\mathbb{X}$, le détail de la routine qui permet de trouver des points de $\mathbb{X}$ est donné ci-dessous.

Algorithme : LSBIC(Entrées : $[\mathbf{x}]_0$, $[\mathbf{y}]$; Sortie : $[\mathbf{x}^*]$)	
1	Si $(w([\mathbf{x}]_0) > \varepsilon_r)$:
2	$[\mathbf{x}]_0 \leftarrow \mathcal{C}_{\mathbb{X}}([\mathbf{x}])$;
3	Si $([\mathbf{x}]_0 \neq \emptyset)$:
4	$d_1 \leftarrow j_{\mathrm{p}}([\mathbf{f}](L[\mathbf{x}]_0), [\mathbf{y}])$; $d_2 \leftarrow j_{\mathrm{p}}([\mathbf{f}](R[\mathbf{x}]_0), [\mathbf{y}])$;
5	Si $(d_1 \leq 0)$: $[\mathbf{x}^*] \leftarrow L[\mathbf{x}]_0$;
6	Sinon :
7	Si $(d_2 \leq 0)$: $[\mathbf{x}^*] \leftarrow R[\mathbf{x}]_0$;
8	Sinon :
9	Si $(d_1 < d_2)$: LSBIC$(L[\mathbf{x}]_0, [\mathbf{y}], [\mathbf{x}^*])$;
10	Si $(d_1 \geq d_2)$: LSBIC$(R[\mathbf{x}]_0, [\mathbf{y}], [\mathbf{x}^*])$;
11	Fin sinon **(8)**, fin si **(7)**
12	Fin sinon **(6)**, fin si **(5)**
13	Fin si **(3)**
14	Fin si **(1)**
15	Fin.

L'algorithme LSBIC (*Local Search By Interval Computation*) propose comme résultat un pavé $[\mathbf{x}^*]$ inclus dans $\mathbb{X}$. Dans cet l'algorithme, $L(.)$ et $R(.)$ sont des opérateurs de bissection de pavé tels que $[\mathbf{x}] = L[\mathbf{x}] \cup R[\mathbf{x}]$ et $\mathrm{Vol}(L[\mathbf{x}]) = \mathrm{Vol}(L[\mathbf{x}]) = \frac{\mathrm{Vol}([\mathbf{x}])}{2}$.

L'avantage de l'algorithme LSBIC provient de sa complexité polynomiale. Ceci se comprend aisément car LSBIC privilégie successivement (grâce au critère de proximité) des zones de l'espace où s'effectuera une recherche approfondie. L'inconvénient de notre approche est dû au manque de fiabilité du critère de proximité. En effet, notre critère de sélection est fortement lié aux pavés utilisés pour approximer $\mathbf{f}([\mathbf{a}])$ et $\mathbf{f}([\mathbf{b}])$. Ainsi la routine LSBIC provoque dans certaines situations (voir figure 9.4) une convergence vers $[\mathbf{x}^*] = \emptyset$ alors que $\mathbb{X} \neq \emptyset$.

Sur la figure 9.4, le pavé qui possède la meilleure proximité avec $[\mathbf{y}]$ dans l'espace image est $[\mathbf{b}]$. Hors comme le montre cette figure, $\mathbf{f}([\mathbf{b}]) \cap [\mathbf{y}] = \emptyset$, malgré que $[\mathbf{f}]([\mathbf{b}]) \cap [\mathbf{y}] \neq \emptyset$. Nous allons privilégier la recherche de solution dans le domaine $[\mathbf{b}]$ où il n'existe aucune solution, tandis que des solutions existent dans $[\mathbf{a}]$. Face à ce type de problèmes, nous présentons un second algorithme issu de la

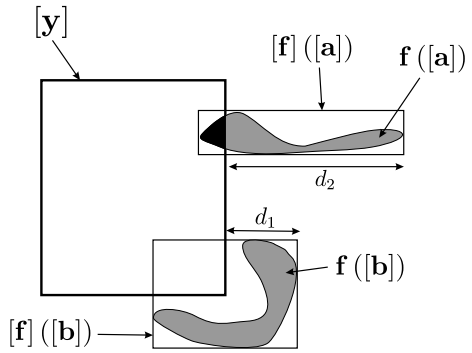

Figure 9.4 – Illustration d'une situation problématique pour le critère de proximité.

combinaison de SIVIA et LSBIC. Cette deuxième approche privilégie certes des domaines sur la base du critère de proximité, mais contrairement à LSBIC ne rejette aucun domaine susceptible de contenir des solutions. Le nouvel algorithme conçu est composé de deux sous algorithmes décrits ci-dessous.

Algorithme : LSBIC1(Entrée - Sortie : $[\mathbf{x}]$)	
1	$[\mathbf{x}] \leftarrow \mathcal{C}_{\mathbb{X}}([\mathbf{x}])$;
2	Si $([\mathbf{x}] \neq \emptyset)$ et $(w([\mathbf{x}]) > \varepsilon_r)$:
3	$\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{C}_{\mathbb{X}}(L[\mathbf{x}])$; $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{C}_{\mathbb{X}}(R[\mathbf{x}])$;
4	Si $(d > 0)$:
5	$d \leftarrow +\infty$;
6	LSBIC1$([\mathbf{x}])$; SELECT_BEST$([\mathbf{x}], d)$;
7	Fin si **(4)**
8	Fin si **(2)**
9	Fin.

Algorithme : SELECT_BEST(Entrées - Sorties : $[\mathbf{x}]$, d)	
1	$[\mathbf{x}] \leftarrow \emptyset$; $i \leftarrow 0$; $k \leftarrow 0$;
2	Tant que $(\mathcal{L} \neq \emptyset)$:
3	$d^* \leftarrow j_{\mathrm{p}}([\mathbf{f}]([\mathbf{x}]), [\mathbf{y}])$;
4	Si $(d^* < d)$: $d \leftarrow d^*$; $k \leftarrow i$; Fin si **(4)**
5	$i \leftarrow i + 1$;
6	Fin tant que
7	Extraire le $k^{\text{ème}}$ élément de $\mathcal{L}$ pour le mettre dans $[\mathbf{x}]$;
8	Fin.

Nous disposons, comme variable globale, d'une liste $\mathcal{L}$ qui constitue notre population de pavés. Le but de la routine `SELECT_BEST` consiste à sélectionner puis à extraire le meilleur pavé $[\mathbf{x}]$ (au sens du critère de proximité) de la liste $\mathcal{L}$. Grâce aux deux algorithmes `LSBIC1` et `SELECT_BEST`, nous assurons d'une part, la recherche de solution à chaque fois sur le meilleur pavé de la liste $\mathcal{L}$, et d'autre part, le stockage dans $\mathcal{L}$ des pavés les moins intéressants. Lorsque le meilleur pavé n'est pas inclus dans $\mathbb{X}$, les pavés $L[\mathbf{x}]$ et $R[\mathbf{x}]$ issus de la bissection de $[\mathbf{x}]$ sont ajoutés à la population des pavés $\mathcal{L}$.

9.3 Application à la conception de filtres d'ordre minimal

Notre but est de présenter dans les exemples qui suivent, d'une part les modèles de filtres recherchés, ainsi que les intervalles de tolérance associés aux gains pour des fréquences données (contraintes sur le gain). D'autre part, nous utiliserons les algorithmes d'estimation `LSBIC` et `LSBIC1` pour concevoir des filtres d'ordre minimal.

Rappelons que la conception d'un filtre d'ordre minimal consiste à trouver le plus petit nombre de coefficients a_i et b_k adéquats pour un filtre, de façon à satisfaire des contraintes sur le module ou la phase. Notons n^* l'ordre minimal des filtres compatibles avec nos contraintes. La stratégie employée vise à démontrer que $\mathbb{S}_{(i,j)} = \emptyset$ avec $i < j$ et $j \leq n^*$ (voir section 9.1.2). Un filtre compatible d'ordre n^* est ensuite obtenu grâce à `LSBIC1`.

Nous montrerons dans les sections qui suivent quelques résultats concernant la conception de filtres numériques mais aussi analogiques.

9.3.1 Filtres numériques

Le gain en fréquence d'un filtre numérique est donné par

$$g_{(m,n)}(\mathbf{a}, \mathbf{b}, \nu) = \left|H\left(e^{j2\pi\nu}\right)\right| = \sqrt{\frac{A_1 + A_2}{B_1 + B_2}}, \tag{9.11}$$

la fonction $H(x)$ correspond à la fonction de transfert (9.1), dans le cas général, nous avons

$$A_1 = \left(\sum_{p=0}^{m} a_p \cos(2p\pi\nu)\right)^2, \; A_2 = \left(\sum_{p=0}^{m} a_p \sin(2p\pi\nu)\right)^2 \tag{9.12}$$

et

$$B_1 = \left(1 + \sum_{k=1}^{n} b_k \cos(2k\pi\nu)\right)^2, \; B_2 = \left(1 + \sum_{k=1}^{n} b_k \sin(2k\pi\nu)\right)^2. \tag{9.13}$$

avec m et n qui sont respectivement les degrés du numérateur et du dénominateur de la fonction de transfert $H(x)$.

Les contraintes sur le gain en fréquence sont de la forme

$$\forall i = 1, 2, ...i_{\max}, \underline{h}_i \leq g_{(m,n)}(\mathbf{a}, \mathbf{b}, \nu_i) \leq \overline{h}_i, \tag{9.14}$$

où les ν_i représentent des échantillons de fréquence appartenant à $[-\pi, \pi]$, les intervalles $[h_i] = \left[\underline{h}_i, \overline{h}_i\right]$ correspondent aux domaines de tolérance pour le gain. En utilisant des notations vectorielles, l'expression 9.14 devient :

$$\mathbf{g}_{(n,m)}(\mathbf{a}, \mathbf{b}) \in [\mathbf{h}], \tag{9.15}$$

avec

$$\mathbf{g}_{(m,n)}(\mathbf{a}, \mathbf{b}) = \begin{pmatrix} g_{(m,n)}(\mathbf{a}, \mathbf{b}, \nu_1) \\ g_{(m,n)}(\mathbf{a}, \mathbf{b}, \nu_2) \\ \vdots \\ g_{(m,n)}(\mathbf{a}, \mathbf{b}, \nu_{i_{\max}}) \end{pmatrix} \tag{9.16}$$

et $[\mathbf{h}] = [h_1] \times [h_2] \times ... \times [h_{i_{\max}}]$.

Exemple 9.2. *Soit trois gabarits de filtre numérique donnés par le tableau 9.17. Les intervalles de tolérances ont été choisis autour des valeurs du gain d'un filtre de référence d'ordre 5.*

i	$2\pi\nu_i$	***cas 1 :*** $\left[h_i^1\right]$	***cas 2 :*** $\left[h_i^2\right]$	***cas 3 :*** $\left[h_i^3\right]$
1	0	$[0.2, 0.4]$	$[0.29, 0.31]$	$[0.38, 0.40]$
2	0.5	$[0.3, 0.5]$	$[0.39, 0.41]$	$[0.60, 0.62]$
3	1	$[0.5, 0.7]$	$[0.59, 0.61]$	$[0.96, 0.98]$
4	1.2	$[0.7, 0.9]$	$[0.79, 0.81]$	$[0.92, 0.94]$
5	1.5	$[0.9, 1.1]$	$[0.99, 1.01]$	$[0.91, 0.93]$
6	1.8	$[1.2, 1.4]$	$[1.29, 1.31]$	$[1.05, 1.07]$
7	2	$[1.4, 1.6]$	$[1.49, 1.51]$	$[1.27, 1.29]$
8	2.5	$[2.1, 2.4]$	$[2.19, 2.31]$	$[1.78, 1.80]$
9	2.8	$[2.4, 2.6]$	$[2.49, 2.51]$	$[1.47, 1.49]$
10	3.1	$[2.4, 2.6]$	$[2.49, 2.51]$	$[1.30, 1.32]$

(9.17)

Sur les cas 1, 2 et 3, nous avons considéré des domaines d'incertitudes pour 10 échantillons de fréquences ($i_{\max} = 10$), la représentation graphique des intervalles d'incertitudes est donnée par les figures 9.5a, 9.5b et 9.5c. La différence entre les deux premiers cas se situe au niveau de la largeur des intervalles de tolérances. Quant au cas 3, le gabarit impose une forme "moins lisse" au gain du filtre.

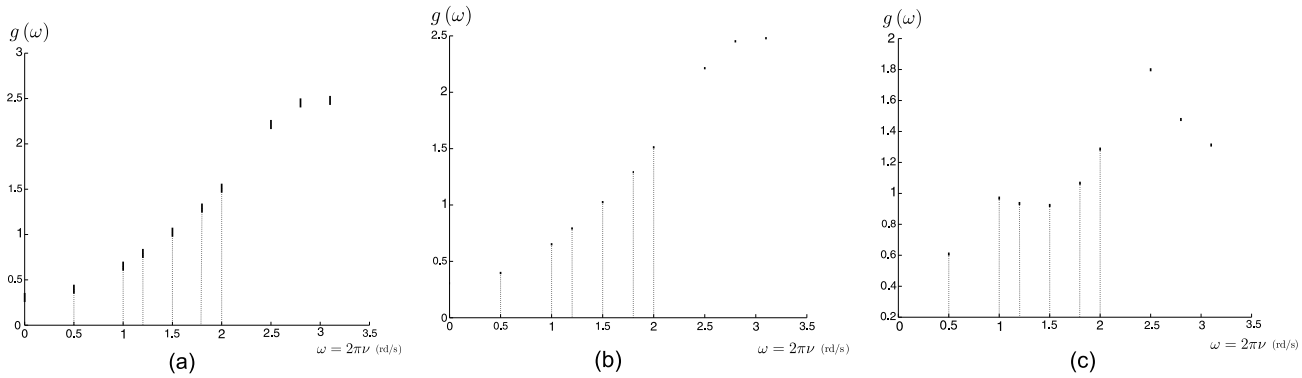

Figure 9.5 – Représentation des intervalles d'incertitude pour les cas 1, 2 et 3.

Notre objectif dans les trois cas est de déterminer des filtres d'ordre minimal. Dans la pratique, pour montrer que $\mathbb{S}_{(n,m)} = \emptyset$ (m et n deux entiers fixés) les domaines d'appartenance des coefficients sont pris finis et assez grands, par exemple $(\mathbf{a}, \mathbf{b}) \in \left[-10^8, 10^8\right]^{\times n} \times \left[-10^8, 10^8\right]^{\times m}$. L'estimation d'un vecteur de coefficients acceptables est effectuée pour $\mathbf{a} \in [-1, 1]^{\times n}$ et $\mathbf{b} \in [-1, 1]^{\times m}$. Grâce aux algorithmes `SIVIA` et `LSBIC1`, le détail des résultats obtenus est décrit comme suit.

– **Cas 1** : $n^* = 2$ et $m^* = 1$; nous avons préalablement vérifié pour de grands intervalles que : $\mathbb{S}_{(0,1)} = \emptyset$ et $\mathbb{S}_{(0,2)} = \emptyset$. L'algorithme `LSBIC1` détermine en environs $20s$ les coefficients d'un filtre supposé d'ordre minimal : $a_0 = -0.99968144080874$, $a_1 = 0.46034161208025$, $b_0 = 1$, $b_1 = 0.52579176165286$ et $b_2 = 0.10288725736099$.
La projection de l'ensemble des filtres d'ordre minimal $\mathbb{S}_{(1,2)}$ suivant les coefficients a_0 et b_1 est donnée par

$$\begin{aligned} \mathbb{Z}_1 &= \text{Proj}_{(a_0,b_1)}\left(\mathbb{S}_{(1,2)}\right) \\ &= \left\{ (a_0, b_1) \middle| \exists\, (a_1, b_2)\,, \mathbf{g}_{(1,2)}\,(\mathbf{a}, \mathbf{b}) \in \left[\mathbf{h}^1\right] \right\}. \end{aligned} \tag{9.18}$$

avec $a = (a_0, a_1)^{\mathrm{T}}$, $b = (b_1, b_2)^{\mathrm{T}}$. Ici, la faible dimension de $\mathbb{S}_{(1,2)}$ ($\dim\left(\mathbb{S}_{(1,2)}\right) = 4$) rend possible une caractérisation de $\mathbb{Z}_1$, avec des temps calcul de l'ordre de 53 minutes et une précision de $\varepsilon = 10^{-4}$ (voir figure 9.6). L'ensemble $\mathbb{Z}_1$ permet alors d'avoir quelques éléments de visualisation et d'interprétation sur les résultats précédemment obtenus par `LSBIC1`. Comme il est indiqué sur la figure 9.6, chaque point de $\mathbb{Z}_1$ correspond à un ensemble de filtres compatibles. En effet, chaque couple de coefficients (a_0, b_1) de $\mathbb{Z}_1$ est associé à un ensemble de points dont les coordonnées sont (a_1, b_2).

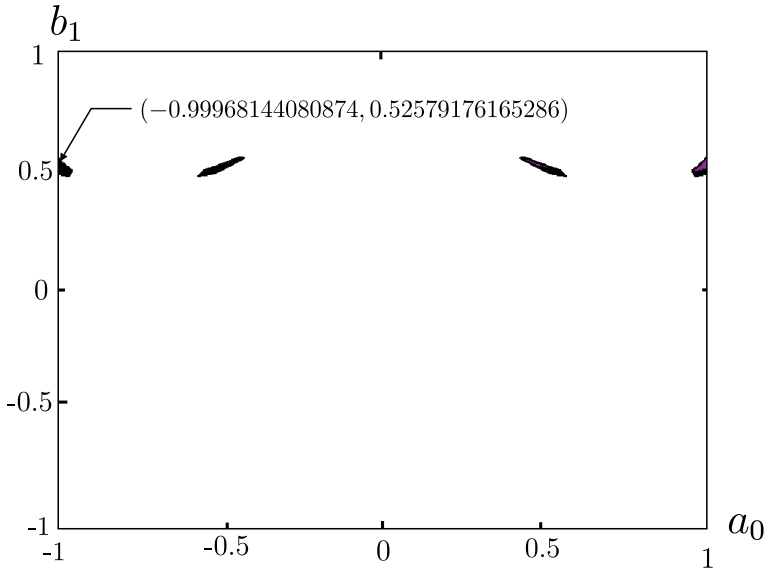

Figure 9.6 – Caractérisation de l'ensemble $\mathbb{Z}_1$.

Considérons l'ensemble défini par

$$\mathbb{Z}_2 = \left\{ (a_1, b_2) |\ \mathbf{g}_{(1,2)} \left(a_0^*, a_1, b_1^*, b_2\right) \in \left[\mathbf{h}^1\right] \right\}, \tag{9.19}$$

avec les paramètres $a_0^* = -0.99968144080874$ et $b_1^* = 0.52579176165286$. La figure 9.7 représente un amas de pavés quasi-ponctuel qui caractérise $\mathbb{Z}_2$. La vérification de $(a_1^*, b_2^*) \in \mathbb{Z}_2$ avec $a_1^* = 0.46034161208025$ et $b_2^* = 0.10288725736099$ permet la validation des résultats obtenus grâce à l'algorithme de recherche de solution `LSBIC1`.

Remarque 9.3. *Les calculs de $\mathbb{Z}_1$ et $\mathbb{Z}_2$ sont possibles dans un temps raisonnable si le nombre de coefficients des filtres d'ordre minimal est relativement petit (inférieur à 5). Dans les cas où les coefficients sont nombreux, nous limiterons l'étude à l'estimation d'un filtre d'ordre minimal quelconque.*

– **Cas 2** : nous avons $n^* = 2$ et $m^* = 4$ avec

$$\begin{aligned} a_0 &= -0.51873743444441 \\ a_1 &= 0.99949087266166 \\ b_0 &= 1 \\ b_1 &= 0.48396192117769 \\ b_2 &= 0.087678306738687 \\ b_3 &= 0.023629693079392 \\ b_4 &= 0.0296984403172177 \end{aligned}$$

Dans ce second cas le temps de calcul est beaucoup plus élevé (environs 2h), ceci est dû principale-

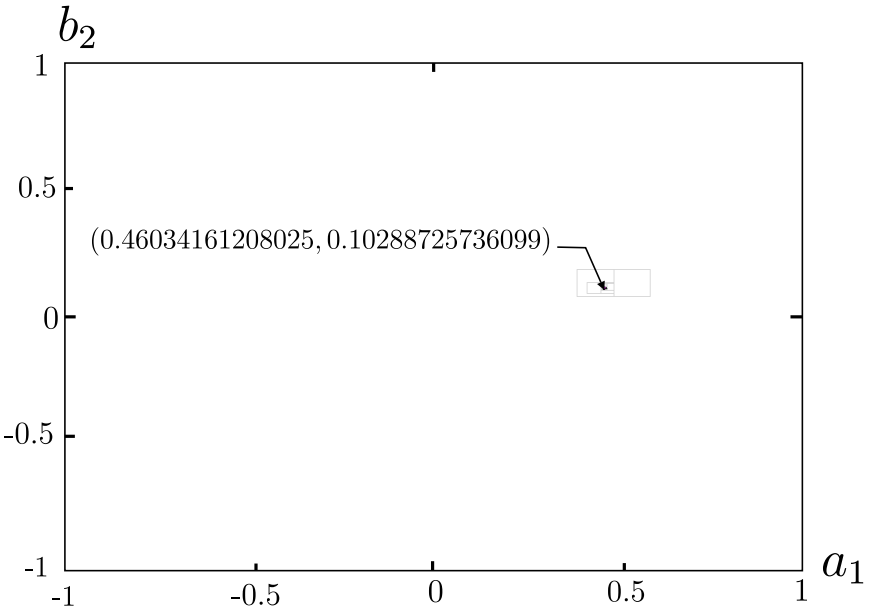

Figure 9.7 – Caractérisation de $\mathbb{Z}_2$.

ment au fait que les intervalles de tolérances sont de petite largeur $l = 2.10^{-2}$. L'ensemble solution $\mathbb{S}_{(m,n)}$ devenant moins volumineux, il devient beaucoup plus difficile de trouver des points issus de cet ensemble. De plus, l'ordre du filtre obtenu est plus élevé que celui estimé dans le cas 1.
La figure 9.8a et 9.8b représentent respectivement les tracés de gains de filtres conçus dans les cas 1 et 2 et les intervalles de tolérances associés à leurs contraintes.

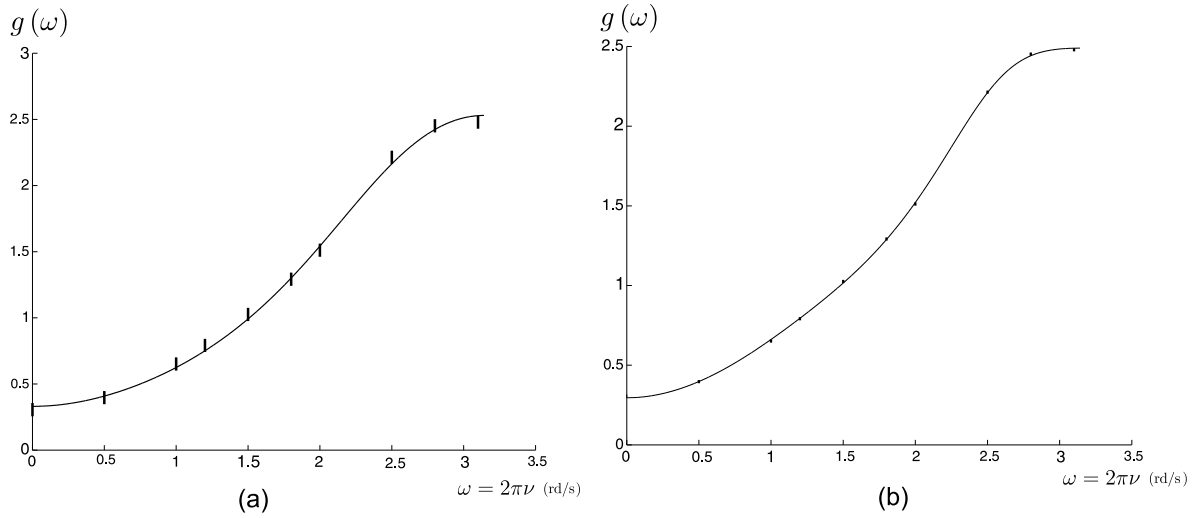

Figure 9.8 – a) Tracé du module et des intervalles de tolérances pour le cas 1. b) Tracé du module et du gabarit dans le cas 2.

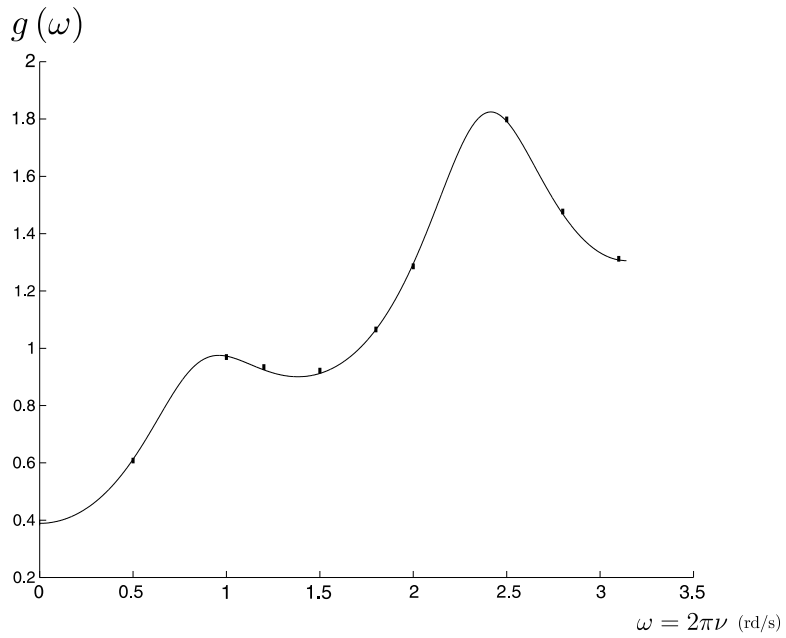

Figure 9.9 – Gain issu de l'estimation d'un filtre numérique d'ordre minimal pour le cas 3.

– **Cas 3 :** il vient $n^* = 2$, $m^* = 4$, avec

$$\begin{aligned} a_0 &= -0.99978613139664 \\ a_1 &= 0.49175104892268 \\ b_0 &= 1 \\ b_1 &= 0.055845079070777 \\ b_2 &= 0.014010938967069 \\ b_3 &= 0.02607233477639 \\ b_4 &= 0.21017176615096 \end{aligned}$$

Nous avons ici la même situation que le cas 2, les domaines de tolérances sont choisis assez petits $\left(l = 2.10^{-2}\right)$. Ceci explique l'ordre relativement élevé du filtre d'ordre minimal. La figure 9.9 montre le tracé du gain du filtre estimé et les intervalles d'incertitudes.

9.3.2 Filtres analogiques

Dans le cas de filtres analogiques, nous avons

$$g_{(n,m)}\left(\mathbf{a}, \mathbf{b}, \nu\right) = \left|H\left(j2\pi\nu\right)\right| = \sqrt{\frac{A_1 + A_2}{B_1 + B_2}}, \tag{9.20}$$

avec

$$A_1 = \left(\sum_{p=0}^{m'} (-1)^p (2\pi\nu)^{2p} a_{2p} \right)^2 , A_2 = \left(\sum_{p=0}^{m'} (-1)^p (2\pi\nu)^{2p+1} a_{2p+1} \right)^2 \tag{9.21}$$

et

$$B_1 = \left(1 + \sum_{k=1}^{n'} (-1)^k (2\pi\nu)^{2k} b_{2k} \right)^2 , B_2 = \left(\sum_{k=0}^{n'} (-1)^k (2\pi\nu)^{2k+1} b_{2k+1} \right)^2 . \tag{9.22}$$

Les entiers m' et n' sont les résultats de divisions entières par 2 de m et n. Ici, les contraintes sont de même forme que celles de la section 9.3.2 ($g_{(n,m)}(\mathbf{a}, \mathbf{b}, \nu_i) \in [h_i]$, $i = 1, 2, ..., i_{\max}$).

Nous considérons le gabarit du cas 1 (voir l'exemple 9.2). A l'issu des calculs effectués grâce à `LSBIC1`, nous obtenons un filtre analogique d'ordre quasi minimal, $n^* = 2$, $m^* = 1$ et les coefficients,

$$\begin{aligned} a_0 &= 0.2869517349555 \\ a_1 &= -0.54952394656682 \\ b_0 &= 1 \\ b_1 &= -0.22039125226251 \\ b_2 &= 0.10524466358619. \end{aligned}$$

Ces résultats ont été obtenus en environ 15 secondes.

Exemple 9.4. *Nous choisissons un filtre de référence d'ordre 6. Les gabarits pour le gain sont indiqués dans le tableau ci-dessous.*

i	$2\pi\nu_i$	***cas 4 :*** $\left[h_i^4\right]$	***cas 5 :*** $\left[h_i^5\right]$
1	0	$[0.95, 1.05]$	$[0.99, 1.01]$
2	0.5	$[0.97, 1.07]$	$[1.01, 1.03]$
3	1	$[0.81, 0.91]$	$[0.85, 0.87]$
4	1.2	$[0.57, 0.67]$	$[0.61, 0.63]$
5	1.5	$[0.24, 0.34]$	$[0.28, 0.3]$
6	1.8	$[0.07, 0.17]$	$[0.11, 0.13]$
7	2	$[0.02, 0.12]$	$[0.06, 0.08]$
8	2.5	$[-0.03, 0.07]$	$[0.01, 0.03]$
9	2.8	$[-0.04, 0.06]$	$[0.002, 0.02]$
10	3.1	$[-0.04, 0.05]$	$[-0.003, 0.02]$

(9.23)

On considère le gabarit donné par le tableau 9.23. Après avoir éliminé les filtres d'ordre incompatibles avec ce gabarit, les ensembles de filtres d'ordre minimal s'écrivent pour le cas 4 : $\mathbb{S}_{(m^*,n^*)} = \mathbb{S}_{(0,3)}$ et pour le cas 5 : $\mathbb{S}_{(m^*,n^*)} = \mathbb{S}_{(2,3)}$. Les estimations de filtres pour les deux cas sont données par,

– **Cas 4 :** $m^* = 0$ et $n^* = 3$

$$\begin{aligned} a_0 &= -1 \\ b_0 &= 1 \\ b_1 &= 0.66585681819813 \\ b_2 &= 0.046761736430157 \\ b_3 &= 1.2683654162864 \end{aligned}$$

A l'instar de la section 9.3.1 (**cas 1**), le faible nombre de coefficients du filtre d'ordre minimal permet de caractériser $\mathbb{F}_1$, la projection de $\mathbb{S}_{(0,3)}$ suivant les coefficients a_0 et b_1. Il vient

$$\mathbb{F}_1 = \left\{ (a_0, b_1) | \, \exists \; (b_2, b_3) \, , \mathbf{g}_{(0,3)} \left(a_0, \mathbf{b}\right) \in \left[\mathbf{h}^4\right] \right\} \tag{9.24}$$

avec $\mathbf{b} = (b_1, b_2, b_3)^{\mathrm{T}}$. La figure 9.10 donne une approximation intérieure et extérieure de $\mathbb{F}_2$. Pour $a_0 = a_0^* = -1$ et $b_1 = b_1^* = 0.66585681819813$, les coefficients b_2, b_3 compatibles avec le gabarit du cas 4 sont définis par l'ensemble

$$\mathbb{F}_2 = \left\{ (b_2, b_3) | \, \mathbf{g}_{(0,3)} \left(a_0^*, b_1^*, b_2, b_3\right) \in \left[\mathbf{h}^4\right] \right\}. \tag{9.25}$$

Une caractérisation de $\mathbb{F}_2$ est représentée par la figure 9.11. Nous vérifions conformément aux résultats obtenus par la recherche de solution ponctuelle que : $(b_2^*, b_3^*) \in \mathbb{F}_2$ avec $b_2^* = 0.046761736430157$ et $b_3^* = 1.2683654162864$. Tous les points de $\mathbb{F}_2$ correspondent à des filtres d'ordre minimal pour le cas 4.

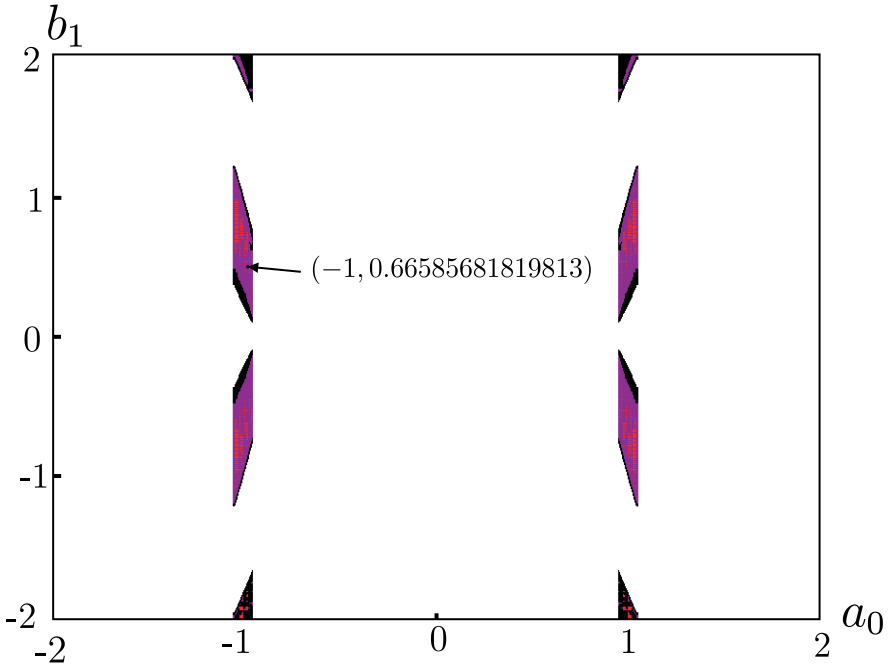

Figure 9.10 – Approximations intérieure et extérieure de $\mathbb{F}_1$.

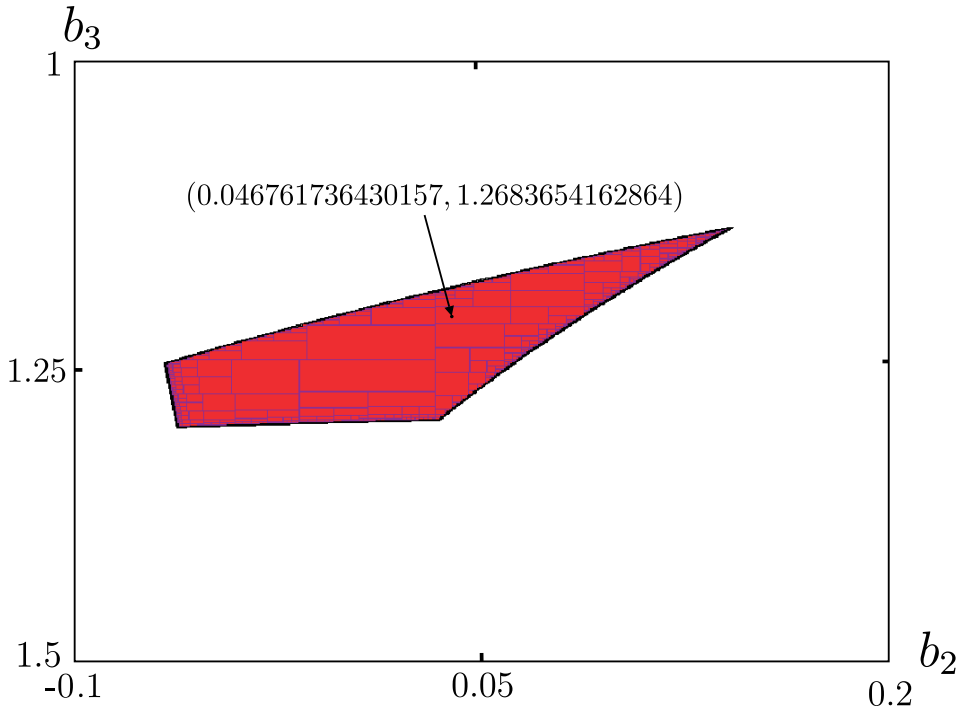

Figure 9.11 – Caractérisation de $\mathbb{F}_2$.

– **Cas 5 :** $m^* = 2$ et $n^* = 3$

$$\begin{aligned} a_0 &= 1.0096989743281 \\ a_1 &= -0.025416753146296 \\ a_2 &= 0.14229711065094 \\ b_0 &= 1 \\ b_1 &= 1.6708877101979 \\ b_2 &= 1.4976891059891 \\ b_3 &= 0.80151007877198 \end{aligned}$$

Les figures 9.12a et 9.12b représentent respectivement pour les cas 4 et 5, les gains en fréquence des filtres d'ordre minimal estimés.

9.3.3 Remarques

Dans les différents cas d'estimation de filtres d'ordre minimal, nous n'avons pas pris en compte les contraintes sur la phase. En effet, les tests d'estimation de filtres avec l'ajout d'un gabarit de phase n'ont pas été concluants. Le comportements de l'algorithme `LSBIC1` indique l'existence de singularités, cette situation est due aux fonctions rationnelles qui composent les contraintes sur la phase. Une interprétation des problèmes de singularité et de leurs conséquences est donnée dans la suite.

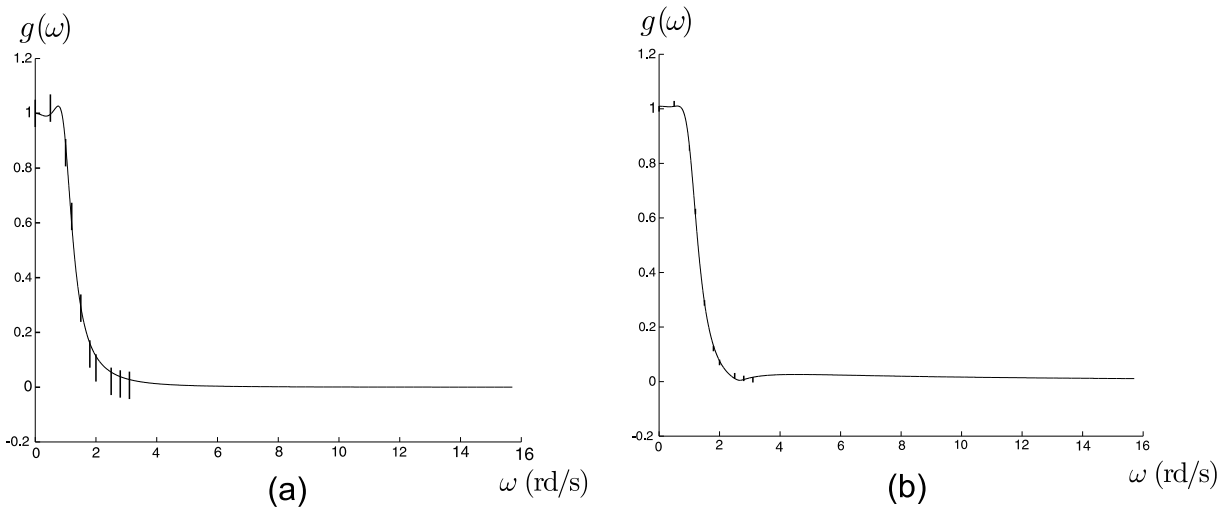

Figure 9.12 – Gain des filtres analogiques d'ordre minimal. (a) : cas 4, (b) : cas 5.

Soit $h : \mathbb{R}^n \rightarrow \mathbb{R}$ une fonction rationnelle,

$$h\left(x_1, x_2, ..., x_n\right) = \frac{f\left(x_1, x_2, ..., x_n\right)}{g\left(x_1, x_2, ..., x_n\right)}. \tag{9.26}$$

Pour $f\left(x_1, x_2, ..., x_n\right) = 0$ et $g\left(x_1, x_2, ..., x_n\right) = 0$, le calcul de $h\left(x_1, x_2, ..., x_n\right)$ aboutit à la forme indéterminée "$\frac{0}{0}$". Le calcul par intervalles admet par convention que $\frac{0}{0} =]-\infty, +\infty[$. Considérons par exemple, l'expression $z = \frac{x}{y}$ avec $(x, y, z) \in [x] \times [y] \times [z]$. Si les trois intervalles $[x]$, $[y]$ et $[z]$ contiennent la valeur 0, aucune réduction de domaines n'est possible à l'aide des procédures de contraction (voir [Benhamou et al., 1999], [van Hentenryck et al., 1998] et [Lhomme, 1993]). Cette situation contribue à une augmentation démesurée du nombre de bissections. Nous n'avons pas rencontré ce type de problème dans les cas d'estimations de filtres avec gabarit sur le gain, par contre cela est nettement différent lorsque nous ajoutons un gabarit sur la phase.

Pour contourner la présence de singularités dans l'expression des contraintes sur la phase, nous nous interdisons une annulation simultanée du numérateur et du dénominateur de la fonction rationnelle. Cette mesure a pour conséquence de restreindre l'ensemble des filtres acceptables ou compatibles. Pour l'expression (9.26), nous évitons les singularités en ajoutant les contraintes : $f\left(x_1, x_2, ..., x_n\right) \neq 0$ ou $g\left(x_1, x_2, ..., x_n\right) \neq 0$. Bien entendu, il s'agit de choisir judicieusement les domaines d'appartenance des variables $x_1, x_2, ..., x_n$ de façon à ce que les fonctions f et g ne s'annulent pas pour un même vecteur $\mathbf{x}$. Ce type de procédé pourra être utilisé dans le cadre de l'estimation de filtre d'ordre minimal avec prise en compte des contraintes sur la phase (gabarit de phase).

9.4 Conclusion

Au cours de ce chapitre, nous avons formulé le problème de conception de filtre en terme de recherche de filtre d'ordre minimal. Le problème se résume ainsi, comment trouver un filtre, avec le plus petit nombre de coefficients de la fonction de transfert (numérateur et dénominateur compris), compatible avec des contraintes sur le gain et éventuellement la phase ?

Nous avons commencé par introduire la notion d'ensemble de filtres (m, n) compatibles $\mathbb{S}_{(m,n)}$. Ensuite, le problème d'estimation de filtre d'ordre minimal a été décomposé en deux sous problèmes, à savoir prouver qu'un ensemble de filtres compatibles d'ordre j est vide, lorsque ceci est réalisé, nous passons à l'ordre supérieur $j + 1$. Ces tests sont réitérés ainsi de suite jusqu'à l'obtention d'un ensemble de filtres - (m^*, n^*) compatibles non vide, n^* devient alors l'ordre minimal des filtres compatibles.

Afin de résoudre notre problème de conception de filtre, une collaboration entre des méthodes intervalles a été proposée. Nous sommes partis du constat suivant, les approches ensemblistes sont souvent utilisées pour traiter des problèmes d'estimation non linéaire. Ces méthodes présentent l'avantage de caractériser un ensemble de vecteurs paramètres solutions dans un compact de $\mathbb{R}^n$, par contre elles ne sont réellement efficaces qu'en petite dimension (nombre de paramètres inférieur à 5). Il convient dans un premier temps, d'utiliser des méthodes ensemblistes pour démontrer qu'un ensemble défini par des inégalités non linéaires est vide, puis, dans un deuxième temps d'élaborer une méthode de recherche de solutions ponctuelles, lorsque la caractérisation de l'ensemble des paramètres solutions n'est plus possible.

Nous avons présenté dans plusieurs exemples les résultats qui concernent la conception de filtres d'ordre minimal. Les cas de filtres numériques et analogiques ont été abordés. Nous avons proposé pour différents gabarits de gain, des filtres compatibles d'ordre minimal ou quasi-minimal. Comme il a été évoqué dans nos dernières remarques, les méthodes d'estimation de filtres rencontrent des difficultés lorsqu'un gabarit de phase est pris en compte dans les contraintes. Ainsi, une mesure à été proposée pour contourner les problèmes dus aux singularités issues des contraintes sur la phase. Cette mesure consiste à fixer certaines conditions sur les fonctions rationnelles qui apparaissent dans les contraintes sur la phase. Des domaines d'appartenance judicieusement choisis pour les coefficients du filtre permettent d'éviter les situations de singularités que nous avons évoqué dans la section 9.3.3.

Le travail effectué jusqu'ici a montré l'importance des approches ensemblistes dans le traitement de problèmes d'estimation non linéaire. En effet, la combinaison entre méthodes ensemblistes et méthodes de recherche ponctuelle permet d'apporter des solutions intéressantes même lorsque le nombre de paramètres est relativement élevé. Bien entendu, les méthodes utilisées restent imparfaites, elles sont gourmandes en temps de calcul en particulier lorsque les intervalles de tolérances pour le gabarit sont de faibles largeurs. Dans ce cas une collaboration entre différents contracteurs (voir [Lhomme, 1993], [van Hentenryck et al., 1998] ou [Benhamou and Granvilliers, 1997]) pourrait s'avérer fortement utile.

Même si nous réussissons à garantir que le filtre conçu est d'ordre minimal, il reste en suspend quelques questions fondamentales à savoir : parmi une infinité de filtres d'ordre minimal, lequel choisir ? Plus précisément, par rapport à quelles critères chaque coefficient devra t-il être choisi ? Les problèmes ainsi soulevés peuvent être complexes, notons néanmoins que des solutions sont envisageables dans le cadre de l'optimisation dite multi-objectifs, ce thème fait l'objet de recherches actuelles (voir la thèse [Barichard, 2003] et la bibliographie associée).

Enfin, les résultats que nous avons obtenus semblent ouvrir des pistes intéressantes pour la conception de systèmes dans les domaines du traitement du signal.

CHAPITRE 10

Conclusion et Perspectives

Tout au long de ce mémoire, nous avons eu le soucis constant de mettre en évidence les contributions apportées par une formulation ensembliste de problèmes d'automatique. Les méthodes étudiées proposent, d'une part, un ensemble de solutions pour les CSPs de type inégalités non linéaires. D'autre part, grâce aux outils algébriques développés dans les chapitres 1 et 2 (analyse par intervalles et contracteurs), nous avons la garantie de ne perdre aucune solution à des inégalités définies dans un compact de $\mathbb{R}^n$.

Un inconvénient des approches utilisées réside dans le fait qu'au delà de quatre dimensions, pour les raisons que nous avons indiquées dans le chapitre 3, une caractérisation précise par méthode d'inversion ensembliste est laborieuse, sauf, si d'une part nous disposons de contracteurs efficaces et d'autre part si nous sommes dans l'un des cas suivant :

- inexistence de vecteurs admissibles dans le domaine considéré;
- les vecteurs du domaine initial sont tous admissibles;
- ensembles de vecteurs admissibles ponctuels et dénombrables.

Nous nous sommes particulièrement intéressés à la caractérisation de la projection d'ensembles définis par des inégalités non linéaires. Cette technique permet de projeter un ensemble de dimension n sur deux dimensions. Ainsi, au lieu de caractériser un ensemble de vecteurs admissibles dans $\mathbb{R}^n$, nous avons choisi de focaliser notre caractérisation sur les vecteurs dans $\mathbb{R}^2$, pour lesquels, nous avons réussi à démontrer qu'il existe des solutions complètes dans $\mathbb{R}^n$. L'intérêt d'une telle démarche serait à terme, de choisir selon un critère, un vecteur admissible dans $\mathbb{R}^2$, puis de rechercher les composantes ou paramètres manquants, par réalisation de projections successives dans les plans correspondants. Par conséquent, comparée à des méthodes ensemblistes plus classiques type inversion ensembliste, l'approche par projections successives permet de réduire partiellement les temps de calcul pour des problèmes de grandes dimensions.

Notons, que dans le "pire" des cas, tous les algorithmes de caractérisation d'ensembles restent de complexité exponentielle. Afin d'améliorer ces algorithmes, en plus de la projection d'ensembles, deux

axes de recherches complémentaires ont été développés dans cette thèse. Ces axes concernent l'amélioration des contracteurs et de l'approximation intérieure, à travers :

- la recherche du plus grand pavé globalement consistant (cf. chapitre 2);
- la présentation d'un algorithme d'approximation intérieure pour la projection d'ensembles `IAVIC` (cf. chapitre 4).

Ces outils ont ensuite été implémentés dans le logiciel PROJ2D, afin d'automatiser le traitement des problèmes liés à la résolution de CSP de type inégalités non linéaires. Nous avons indiqué dans le chapitre 4, une partie technique sur les différentes étapes de conception de ce logiciel. Nous rappelons les deux points principaux de cette conception, la représentation des CSPs sous forme d'arbre, ainsi que la décomposition des inégalités en sous-graphes connexes dont les nœuds sont les variables quantifiées ou non projetées. Nous avons montré l'efficacité de cette décomposition, grâce au calcul des complexités des différents algorithmes conçus (voir section 5.2.5). Nous mettons à la disposition des lecteurs, les codes sources en C++ ainsi que l'executable de ce logiciel.

Tout ce que nous venons de présenter jusqu'ici concerne les études menées dans la première partie de ce mémoire, qui nous le rappelons s'intitule "Méthodes et principes".

Dans la partie "Applications" de la thèse , nous avons traité principalement de problèmes liés au domaine de l'automatique. Nous avons utilisé les résultats graphiques fournis par PROJ2D, afin de traiter des problèmes issus de l'étude des systèmes à retards, de la commande optimale d'un bateau à voile et de la conception de filtres dans le domaine du traitement de signal.

Dans le chapitre 5, plusieurs exemples d'application ont été traités. Nous avons prélevé, au cours du temps, des mesures sur les variations des volumes des sous-pavages caractérisés. Ces mesures nous ont permis de noter, les performances des algorithmes de projection (`SPVIA`), lorsque nous combinons ces algorithmes à la méthode d'approximation intérieure `IAVIC` proposée dans le chapitre 4. Des améliorations ont été remarquées pour les problèmes impliquant le plus de paramètres.

Le principal résultat du chapitre 6 concerne la stabilité les systèmes à retards. Grâce aux méthodes ensemblistes, nous avons décrit, comment il était possible de vérifier l'absence de zéros d'une une fonction transcendantale, dans un compact de $\mathbb{R}^n$. Nous avons vu que dans certains cas, il était possible de borner les pôles instables du système à retards, nous obtenons ainsi une preuve numérique pour la stabilité de ce système. Dans les cas où il est difficile d'avoir une telle preuve, les méthodes par intervalles permettent d'apporter, des éléments d'analyse à des problèmes de stabilité ou de stabilisation robuste de systèmes à retards.

A travers la commande de systèmes non linéaires, nous avons illustré la diversité des problèmes traités grâce à la projection d'ensembles (voir chapitre 8). Nous avons ainsi détaillé la conception d'une "commande optimale" pour un bateau à voile. Les résultats graphiques fournis par PROJ2D ont été es-

sentiels à la conception de cette commande. Nous avons montré à travers diverses simulations l'efficacité d'une telle commande. Les codes sources des simulations sont mis à la disposition des lecteurs.

Enfin dans le chapitre 9, nous avons abordé quelques perspectives de recherches pour cette thèse. Dans le cadre d'un problème d'estimation à erreurs bornées, nous nous intéressons à la conception de filtres numériques et analogiques. Ce problème nécessite dans certains cas l'estimation d'un grand nombre de paramètres (jusqu'à 7 paramètres). De ce fait, les méthodes qui consistent à caractériser systématiquement tous les vecteurs admissibles dans un compact ne sont plus envisageables. Il en est de même pour les méthodes de projection d'ensembles. Nous avons donc abandonné une approche strictement ensembliste afin de proposer une stratégie de recherche hybride. Cette stratégie intègre des critères discriminants pour privilégier la recherche de solutions dans certaines parties du domaine initial. Ainsi, le nouveau type de méthodes sera basé sur la combinaison d'heuristiques et de méthodes ensemblistes, avec pour objectif de converger, grâce à un critère de selection, vers un seul vecteur admissible. Les problèmes de robotique (étalonnage de robot et conception de robot) étant des problèmes d'estimation avec de nombreux paramètres (en général supérieur à 4), le développement de ce thème pourrait constituer un apport intéressant pour ces problèmes.

Bibliographie

[Spa, 2000] (2000). Spacesolver, disponible sur `http://liawww.epfl.ch/~lottaz/spacesolver/`.

[Aho et al., 1991] Aho, A. V., Sethi, R., and Ulman, J. D. (1991). *Compilateurs : principes, techniques et outils*. InterEditions.

[Appel and Ginsburg, 1998] Appel, A. W. and Ginsburg, M. (1998). *Modern Compiler Implementation in C*. Cambrige University Press.

[Baguenard et al., 2003] Baguenard, X., Dao, M., Jaulin, L., and Khalil, W. (2003). Méthodes ensemblistes pour l'étalonnage géométrique. *JESA*, 37(9):1060–1074.

[Barichard, 2003] Barichard, V. (2003). *Approches hybrides pour les problèmes multiobjectifs*. Thèse de doctorat, spécialité informatique, Université d'Angers, France.

[Barmish, 1994] Barmish, B. R. (1994). *New Tools for Robustness of Linear Systems*. MacMillan, New York, NY.

[Barre et al., 1995] Barre, P.-J., Carron, J.-P., Hautier, J.-P., and Legrand, M. (1995). *Systèmes automatiques, Tome 2 : Commande des processus*. Editions ellipses, Paris.

[Bellman and Cooke, 1963] Bellman, R. and Cooke, J. R. (1963). *Differential-difference equations*. Academic Press, New York.

[Benhamou et al., 1999] Benhamou, F., Goualard, F., Granvilliers, L., and Puget, J. F. (1999). Revising hull and box consistency. In *Proceedings of the International Conference on Logic Programming*, pages 230–244, Las Cruces, NM.

[Benhamou and Granvilliers, 1997] Benhamou, F. and Granvilliers, L. (1997). Automatic generation of numerical redundancies for nonlinear constraint solving. *Reliable Computing*, 3(3):335–344.

[Benhamou et al., 1994] Benhamou, F., McAllester, D., and van Hentenryck, P. (1994). CLP (intervals) revisited. In Bruynooghe, M., editor, *Proceedings of the International Logic Programming Symposium*, pages 124–138, Ithaca, NY.

[Benhamou and Older, 1997] Benhamou, F. and Older, W. (1997). Applying interval arithmetic to real, integer and Boolean constraints. *Journal of Logic Programming*, pages 1–24.

[Birdwell et al., 2003] Birdwell, N., Chiasson, J., Tang, Z., Abdallah, C. T., Hayat, M., and Wang, T. (2003). Dynamic time-delay models for load balancing, part 1 : Deterministic models. In *Proc. 1st Workshop CNRS-NSF, Advances in Time-Delay Systems*, Paris, France.

[Braems, 2002] Braems, I. (2002). *Méthodes ensemblistes guaranties pour l'estimation de grandeurs physiques*. Thèse de doctorat, spécialité automatique, Université Paris Sud, Orsay, France.

[Bryson and Ho, 1975] Bryson, A. E. and Ho, Y.-C. (1975). *Applied Optimal Control*. Hemisphere Publishing Corp., New York.

[Cendes and Ratz, 1997] Cendes, T. and Ratz, D. (1997). Subdivision direction selection in interval methods for global optimization.

[Cleary, 1987] Cleary, J. G. (1987). Logical arithmetic. *Future Computing Systems*, 2(2):125–149.

[Dao et al., 2003] Dao, M., Baguenard, X., and Jaulin, L. (2003). Projection d'ensembles pour l'estimation de paramètres, la conception de robot et la commande robuste. In *Proc. Journées Doctorales d'Automatique (JDA)*, Valencienne, France.

[Dao et al., 2004] Dao, M., Di-Loreto, M., Jaulin, L., Lafay, J.-J., and Loiseau, J.-J. (2004). Application du calcul par intervalles aux systèmes à retards. In *Proc. Conférence Internationale Francophone d'Automatique*, Douz-Tunisie.

[Davis, 1987] Davis, E. (1987). Constraint propagation with interval labels. *Artificial Intelligence*, 32(3):281–331.

[de Larminat, 1993] de Larminat, P. (1993). *Automatique : Commande des systèmes linéaires*. Editions Hermes, Paris.

[Delanoue et al., 2004] Delanoue, N., Jaulin, L., and Cottenceau, B. (2004). Using interval arithmetic to prove that a set is path-connected. *Theorical Computer Science*.

[Desoer and Vidyasagar, 1975] Desoer, C. A. and Vidyasagar, M. (1975). *Feedback Systems : Output-Input Properties*. Academic Press, New York.

[Di-Loreto et al., 2005] Di-Loreto, M., Dao, M., Jaulin, L., Lafay, J.-J., and Loiseau, J.-J. (2005). *Applied interval computation : A new approach for time-delays systems analysis*. Springer Verlag.

[Didrit, 1997] Didrit, O. (1997). *Analyse par intervalles pour l'automatique; résolution globale et garantie de problèmes non linéaires en robotique et commande robuste*. PhD dissertation, Université Paris-Sud, Orsay, France.

[Dombre and Khalil, 1999] Dombre, E. and Khalil, W. (1999). *Modélisation, identification et commande des robots, Collection Robotique*. Hermès, Paris.

[Dubois and Prade, 1980] Dubois, D. and Prade, H. (1980). *Fussy Sets and Systems-Theory and Applications*. Academic Press, New York, NY.

[Durieu and Walter, 2001] Durieu, C. and Walter, E. (2001). *Estimation ellipsoïdale à erreur bornée*. Hermès, Paris.

[Fliess et al., 2004] Fliess, M., Mboup, M., Neves, A., and Sira-Ramirez (2004). Une autre vision de l'identification des signaux et systèmes. In *Proc. Conférence Internationale Francophone d'Automatique*, Douz-Tunisie.

[Floudas and Visweswaran, 1993] Floudas, C. and Visweswaran, V. (1993). Primal relaxed dual global optimization approach. *Journal of Optimization, Theory and its Applications*, 78:187–225.

[Gondran and Minoux, 1985] Gondran, M. and Minoux, M. (1985). *Graphes et algorithmes*. Eyrolles, Paris, France.

[Gosselin and Merlet., 1994] Gosselin, C. and Merlet., J.-P. (1994). On the direct kinematics of planar parallel manipulators: special architectures and number of solutions. *Mechanism and Machine Theory, Novembre 1994*, 29(8):1083–1097.

[Granvilliers, 2002] Granvilliers, L. (2002). *RealPaver, available at* `http://www.sciences.univ-nantes.fr/info/perso/permanents/granvil/realpaver/`. IRIN, University of Nantes.

[Gu et al., 2002] Gu, K., Kharitonov, V. L., and Chen, J. (2002). *Stability of time-delay systems*. Birkhauser, Boston.

[Hansen, 1965] Hansen, E. R. (1965). Interval arithmetic in matrix computations - part I. *SIAM Journal on Numerical Analysis: Series B*, 2(2):308–320.

[Hansen, 1975] Hansen, E. R. (1975). A generalized interval arithmetic. In Nickel, K., editor, *Interval Mathematics 1975*, Lecture Notes in Computer Science, pages 7–18. Springer-Verlag.

[Hansen, 1992a] Hansen, E. R. (1992a). Bounding the solution of interval linear equations. *SIAM Journal on Numerical Analysis*, 29(5):1493–1503.

[Hansen, 1992b] Hansen, E. R. (1992b). *Global Optimization using Interval Analysis*. Marcel Dekker, New York, NY.

[Hansen and Greenberg, 1983] Hansen, E. R. and Greenberg, R. I. (1983). An interval Newton method. *Applied Mathematical Computing*, 12:89–98.

[Hohenbichler and Ackermann, 2003] Hohenbichler, N. and Ackermann, J. (2003). Computing stable regions in parameter spaces for a class of quasipolynomials. In *Proc. IFAC Workshop on Time-Delay System (TDS'03)*, Rocquencourt, France.

[Iachkine, 1999] Iachkine, P. (1999). *Contribution à la caractérisation et au choix d'éléments de structures composites pour les bateaux de series olympiques*. PhD dissertation, Ecole Centrale de Nantes, Nantes, France.

[Isidory, 1995] Isidory, A. (1995). *Nonlinear Control Systems : An Introduction, 3rd Edition*. Springer-Verlag, New-York.

[Jaulin, 1994] Jaulin, L. (1994). *Solution globale et garantie de problèmes ensemblistes ; application à l'estimation non linéaire et à la commande robuste*. PhD dissertation, Université Paris-Sud, Orsay, France. Available at: http://www.istia.univ-angers.fr/~jaulin/thesejaulin.zip.

[Jaulin, 2005] Jaulin, L. (2005). *Représentation d'état pour la modélisation et la commande des systèmes*. Hermes-Sciences, Paris.

[Jaulin et al., 2001] Jaulin, L., Kieffer, M., Didrit, O., and Walter, E. (2001). *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer-Verlag, London.

[Jaulin and Walter, 1993] Jaulin, L. and Walter, E. (1993). Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4):1053–1064.

[Jaulin and Walter, 1999] Jaulin, L. and Walter, E. (1999). Guaranteed bounded-error parameter estimation for nonlinear models with uncertain experimental factors. *Automatica*, 35(5):849–856.

[Kharitonov and Zhabko, 1994] Kharitonov, L. V. and Zhabko, A. P. (1994). Robust stability of time-delay systems.

[Kieffer, 1999] Kieffer, M. (1999). *Estimation ensembliste par analyse par intervalles, application à la localisation d'un véhicule*. PhD dissertation, Université Paris-Sud, Orsay, France.

[Kolmanovskii and Myshkis, 1992] Kolmanovskii, V. and Myshkis, A. (1992). *Applied Theory of Functional Differential Equations*. Kluwer Academic Publishers, Netherlands.

[Krawczyk and Neumaier, 1985] Krawczyk, R. and Neumaier, A. (1985). Interval slopes for rational functions and associated centered forms. *SIAM Journal of Numerical Analysis*, 22:604–616.

[Kurzhanski and Valyi, 1997] Kurzhanski, A. and Valyi, I. (1997). *Ellipsoidal Calculus for Estimation and Control*. Birkhäuser, Boston, MA.

[Landau, 1993] Landau, I. D. (1993). *Identification et commande des systèmes, 2ème édition revue et augmentée*. Editions Hermes, Paris.

[Levine et al., 1992] Levine, J., Masson, T., and Brown, D. (1992). *Lex & Yacc*. O'Reilly and Associates, 2nd edition.

[Lhomme, 1993] Lhomme, O. (1993). Consistency techniques for numeric CSPs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 232–238, Chambéry, France.

[Li et al., 1998] Li, X., de Souza, C. E., and Trofino, A. (1998). Delay-dependent robust stabilization of uncertain linear state-delayed systems via static output feedback. In *Proc. IFAC Workshop LTDS*, Grenoble, France.

[Ljung, 1987] Ljung, L. (1987). *System Identification Theory for the user*. Prentice Hall.

[Lottaz, 2000] Lottaz, C. (2000). *Collaborative design using solution spaces*. PhD dissertation 2119, Swiss Federal Institute of Technology in Lausanne, Switzerland.

[Marino and Tomei, 1995] Marino, M. and Tomei, P. (1995). *Nonlinear Control Design : Geometric, Adaptative and Robust*. Prentice Hall.

[Merlet, 1996] Merlet, J.-P. (1996). Designing a parallel manipulator for a specific workspace. In *Internat. Symp. on Robotics and Manufacturing*, Montpellier.

[Michiels et al., 2003] Michiels, W., Niculescu, S. I., and Moreau, L. (2003). Improving the static output feedback stabilizability of linear systems by introducing delays and time-varying gains : a comparison. In *Proc. IFAC Workshop on Time-Delay System (TDS'03)*, Rocquencourt, France.

[Milanese and Vicino, 1991] Milanese, M. and Vicino, A. (1991). Estimation theory for nonlinear models and set membership uncertainty. *Automatica*, 27(2):403–408.

[Moore, 1966] Moore, R. E. (1966). *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ.

[Moore, 1968] Moore, R. E. (1968). Practical aspects of interval computation. *Appl. Math.*, (13):52–92.

[Moore, 1979] Moore, R. E. (1979). *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, PA.

[Moore, 1992] Moore, R. E. (1992). Parameter sets for bounded-error data. *Mathematics and Computers in Simulation*, 34(2):113–119.

[Naimark et al., 1998] Naimark, L., Zeheb, E., and Kogan, J. (1998). Stabilizing rational controllers for a class of a time-delay system. In *Proc. IFAC Workshop LTDS*, Grenoble, France.

[Neumaier, 1990] Neumaier, A. (1990). *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, UK.

[Niculescu, 1996] Niculescu, S. I. (1996). *Sur la stabilité et la stabilisation des systèmes linéaires à états retardés*. Thèse de doctorat, INPG, Grenoble, France.

[Niculescu, 2001] Niculescu, S. I. (2001). *Delay Effects on stability : A Robust Control Approach*. Springer, New York.

[Pontryagin, 1942] Pontryagin, L. S. (1942). On the zeros of some elementary transcendental functions. *Izvestiya Akademii Nank, SSSR (Russian)*, 6:115–131.

[Raïssi, 2004] Raïssi, T. (2004). *Méthodes ensemblistes pour l'estimation d'état et de paramètres*. Thèse de doctorat, Université Paris XII Val de Marne, Paris, France.

[Raïssi et al., 2004] Raïssi, T., Ramdani, N., Ibos, L., and Candau, Y. (2004). Analyse de propriétés diélectriques dans un contexte à erreurs bornées. In *Conférence Internationale Francophone d'Automatique 2004*, Douz, Tunisie.

[Ratschan, 2000] Ratschan, S. (2000). Approximate quantified constraint solving (AQCS). Available at: http://www.risc.uni-linz.ac.at/research/software/AQCS.

[Ratschan, 2002] Ratschan, S. (2002). Approximate quantified constraint solving by cylindrical box decomposition. *Reliable Computing*, 8(1):21–42.

[Rivoire and Ferrier, 1997] Rivoire, M. and Ferrier, J.-L. (1997). *Commande par calculateur, Identification*. Eyrolles, Paris.

[Santos et al., 2003] Santos, J., Mondié, S., and Kharitonov, V. L. (2003). Robust stability of time-delay systems and the finite inclusions theorem. In *4th IFAC Workshop TDS*, Rocquencourt, France.

[Schoen and Gerring, 1993] Schoen, G. M. and Gerring, H. P. (1993). Stability condition for a delay differential system.

[van Hentenryck et al., 1997] van Hentenryck, P., Deville, Y., and Michel, L. (1997). *Numerica: A Modeling Language for Global Optimization*. MIT Press, Boston, MA.

[van Hentenryck et al., 1998] van Hentenryck, P., Michel, L., and Benhamou, F. (1998). `Newton`: Constraint programming over nonlinear constraints. *Science of Computer Programming*, 30(1–2):83–118.

[Vehi et al., 1997] Vehi, J., Armengol, J., Rodellar, J., and Sainz, M. A. (1997). Using interval methods for control systems design in the parameter space. In *7th Symposium on Computer Aided Control Systems Design*, pages 371–375, Gent.

[Vyhlìdal and Zìtek, 2003] Vyhlìdal, T. and Zìtek, P. (2003). Quasipolynomial mapping based root-finder for analysis of time delay system. In *4th IFAC Workshop TDS*, Rocquencourt, France.

[Walter, 1982] Walter, E. (1982). *Identifiability of state space models*. Springer-Verlag, Berlin, Germany.

[Walter and Jaulin, 1994] Walter, E. and Jaulin, L. (1994). Guaranteed characterization of stability domains via set inversion. *IEEE Transactions on Automatic Control*, 39(4):886–889.

[Walter and Pronzato, 1997] Walter, E. and Pronzato, L. (1997). *Identification of Parametric Models from Experimental Data*. Springer-Verlag, London, UK.

[Waltz, 1975] Waltz, D. (1975). Generating semantic descriptions from drawings of scenes with shadows. In Winston, P. H., editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, New York, NY.

[Zadeh, 1971] Zadeh, L. (1971). Quantitative fuzzy semantics. *Information Sciences*, (3):159–176.

Index

Caractérisation d'ensembles par des méthodes intervalles. Applications en automatique

Massa DAO

Résumé

L'étude et la conception des systèmes non linéaires (étude de la stabilité, synthèse de lois de commandes stabilisantes, analyse en robustesse, ...) posent des problèmes numériques difficiles, que les méthodes classiques ont du mal à résoudre. Les formulations ensemblistes de ces problèmes où l'idée de base est de remplacer une valeur ponctuelle par un intervalle qui la contient se sont déjà montrées très efficaces pour leur résolution.

Dans cette thèse, nous proposons de nouveaux algorithmes ensemblistes dédiés à des tâches plus spécifiques telles que la projection d'un ensemble sur un sous-espace. Afin d'améliorer l'efficacité de ces algorithmes, dans un cadre général, nous avons présenté deux idées. La première de ces idées consiste à former à partir des contraintes, des graphes connexes de variables quantifiées. L'objectif est de décomposer la projection d'un ensemble à n dimensions, en une intersection de projections d'ensembles dont la somme des dimensions vaut n. Notre deuxième idée consiste à rendre les algorithmes de projection plus efficace en améliorant l'approximation intérieure.

Dans un deuxième temps, les algorithmes développés ont été mis en oeuvre afin de traiter des problèmes d'automatique liés aux systèmes à retards, à la commande d'un bateau à voile et à la conception de filtres numériques et analogiques.

Mots-clés : analyse par intervalles, problème de satisfaction de contraintes (CSP), projection d'ensembles, inversion ensembliste, estimation à erreurs bornées, systèmes à retards, commande d'un bateau à voile, conception de filtres.

Abstract

The study and the design of nonlinear systems (stability study, stabilizing control laws synthesis, robustness analysis, ...) arise difficult numerical problems that classical methods have difficulty to solve. Then, set-membership formulations of these problems, where the basic idea is to replace punctual value by interval that include it, where already show very efficient for their resolution.

In this thesis, we propose new set-membership algorithms dedicated for most special tasks like set projection over a subspace. In order to improve, the efficiency of these algorithms in general context, we have presented two ideas. The first one consist to decompose the projection of a n-dimensional set, in an intersection of sets projections where the sum of dimensions is equal to n. The second idea consists in yielding the projection algorithms more efficient by improving inner approximation.

In a second time, the developed algorithms have been implemented in order to treat stability and control problem of time-delay systems, sailing boat control and filters design.

Keywords: interval analysis, constraints satisfaction problem (CSP), set projection, bounded-error estimation, set inversion, time-delays system, sailing boat control, filters design.