Le calcul ensembliste par analyse par intervalles et ses applications

Luc Jaulin

▶ **To cite this version:**

Luc Jaulin. Le calcul ensembliste par analyse par intervalles et ses applications. Modélisation et simulation. Université d'Angers, 2000. tel-00457239

HAL Id: tel-00457239
https://theses.hal.science/tel-00457239

Submitted on 16 Feb 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Table des matières

Chapitre 1

Présentation générale

Dans ce chapitre, sont rassemblés quelques renseignements administratifs concernant ma formation, les postes que j'ai occupés, mes activités de recherche et mes publications. La description scientifique de mes recherches, dont une présentation est donnée au paragraphe 1.5, ne débute qu'au chapitre 2. En annexe, sont jointes seize publications significatives publiées, acceptées ou soumises.

1.1 Curriculum vitae

1.1.1 Etat civil

Nom :	Jaulin
Prénoms :	Luc Jean François
Date et lieu de naissance :	15 mars 1967 à Nevers
Nationalité :	Française
Adresse :	L2S, Supélec, Plateau de Moulon, 91192, Gif-sur Yvette
Téléphone :	01 69 85 17 59 (travail) ; 01 43 36 74 18 (domicile)
Télécopieur :	01 69 41 30 60
Adresse électronique :	`jaulin@lss.supelec.fr`

1.1.2 Formation et diplômes

1990-94 J'ai préparé ma thèse de Doctorat en Sciences (Automatique et Traitement du Signal) au *Laboratoire des Signaux et Systèmes* (CNRS-Supélec, Université de Paris-Sud), sous la responsabilité d'Eric Walter. Je l'ai soutenue le 14 février 1994 devant un jury composé de P. Bertrand (président), D. Meizel et R. Moore (rapporteurs), J.

Richalet, J. Vignes et E. Walter (examinateurs). J'ai obtenu la mention très honorable avec félicitations. Le sujet en était : *Solution globale et garantie de problèmes ensemblistes ; applications à l'estimation non-linéaire et à la commande robuste.*

1990 DEA *Automatique et Traitement du Signal* de l'Université de Paris-Sud, stage au centre de Recherche en Physique de l'Environnement, CNET-CNRS. Sujet : "*Restitution de l'atténuation d'une cellule de pluie par une technique variationnelle*".

1984-90 Ingénieur ESIEA (Informatique, Electronique et Automatique).

1984 Baccalauréat série E.

1.1.3 Postes occupés

Depuis Septembre 1998 : J'ai obtenu pour deux ans une délégation de l'Université d'Angers vers le CNRS. Ma demande a été motivée par l'écriture d'un livre sur mon thème de recherche et la volonté de passer mon habilitation à diriger des recherches. J'ai été accueilli au LSS-CNRS dans l'équipe de E. Walter.

Septembre 1994 - Août 1998 : Maître de Conférences en 61$^{\text{ième}}$ section. Enseignement en Physique (plus particulièrement : Automatique, Electronique, Electrotechnique et Thermodynamique) à la Faculté des Sciences de l'Université d'Angers. Laboratoire de recherche : LISA.

1992-1993 : Service national : VSNA enseignement en Mathématiques, Physique, Chimie, Informatique et Sport au lycée Européen de Manille (Philippines).

1.2 Présentation de ma thèse

Ma thèse de doctorat [Jau94], soutenue le 10 février 1994 en Automatique et Traitement du Signal, a été encadrée par E. Walter au LSS. Le sujet en était "*Solution globale et garantie de problèmes ensemblistes ; application à l'estimation non-linéaire et à la commande robuste*". Elle a donné lieu aux publications [JW92], [JW93b], [JW93e], [JW93a], [JW93c], [JW93d], [WJ94], [JW94] et [JW96b].

Le sujet initial de ma thèse était l'utilisation d'outils nouveaux pour traiter des problèmes d'estimation à erreurs bornées de façon globale et garantie. Très rapidement, j'ai proposé d'utiliser une approche basée sur *l'analyse par intervalles*. En bref, l'analyse par intervalles peut être présentée comme un outil numérique, encore assez confidentiel dans la communauté de l'automatique, qui permet la résolution rigoureuse d'une grande classe

de problèmes non-linéaires. Comme ceux-ci sont très présents en automatique et que les méthodes formelles sont souvent incapables de les résoudre, E. Walter et moi-même avons pensé qu'elle avait sa place dans notre communauté. Ma thèse s'est alors réorientée vers l'application de cet outil pour la résolution de problèmes ensemblistes en automatique. Je l'ai ainsi appliqué pour résoudre des problèmes de stabilité, de commande robuste, d'estimation de régions de confiance et d'optimisation globale.

Il semblerait qu'avec E. Walter, nous soyons les premiers de la communauté des automaticiens à avoir utilisé le calcul par intervalles pour résoudre des problèmes d'estimation et de commande. Depuis, d'autres chercheurs nous ont suivi dans cette voie comme l'a montré le congrès ayant pour sujet "*Applications of Interval Analysis to Systems and Control*", qui a eu lieu à Girone, en février 1999.

Pendant cette thèse, j'ai introduit, les notions fondamentales d'*inversion ensembliste*, de *sous-pavages* et de *test d'inclusion.* Mes travaux se trouvent maintenant référencées et/ou utilisées par beaucoup d'autres chercheurs.

- **En estimation et observation** : G. Chen, J. Wang et L.S. Shieh [CWS97] ; A.L. Dontchev, M. P. Polis ; V. M. Veliov ([DPV99]) ; J. P. Norton [Nor96] ; F. J. Rivas, S. T. Kolaczkowski, F. J. Beltran et D. B. Mclurgh [RKBM98].
- **En automatique et robotique** : A. Piazzi, G. Marro et Visioli [PM96], [PV98], [PV99] ; J. Yang et W. Wang [YW99] ; Garloff et Zettler [ZG98], [Gar99] ; Y. Ohta [Oht99] ; Candev [Can96].

1.3 Activités de recherche

Juste après ma thèse, j'ai été nommé comme maître de conférences à la Faculté des Sciences de l'Université d'Angers. Le laboratoire de recherche, auquel je me suis rattaché, est le LISA (Laboratoire d'Ingénierie et des Systèmes Automatisés). J'y ai poursuivi ma recherche, dans la continuité de mon travail de thèse, avec un intérêt plus particulier pour les systèmes à événements discrets. Je suis resté quatre années à Angers puis j'ai obtenu, voici un an, une délégation vers le CNRS au LSS (Laboratoire des Signaux et Systèmes).

1.3.1 Coencadrement de thèses

Pendant ces cinq dernières années post-doctorales, j'ai coencadré trois doctorants[1] qui ont soutenu leur thèse :

[1] Pour chacun de ces encadrements, j'ai participé au jury de thèse en tant qu'examinateur.

- **Olivier Didrit** a préparé sa thèse [Did97] au LSS intitulée "*Analyse par intervalles pour l'automatique ; résolution globale et garantie de problèmes non linéaires en robotique et en commande robuste*". Son directeur de thèse était E. Walter et son travail a donné lieu aux publications [DJW95],[DJW97] et [JWD96]. Olivier a soutenu le 30 juin 1997 et occupe actuellement un poste d'ingénieur à Canal +. Nous continuons à collaborer puisque O. Didrit, M. Kieffer, E. Walter et moi-même sommes actuellement en phase de rédaction d'un livre sur les applications de l'analyse par intervalles à des problèmes d'Automatique. Ce livre doit être prêt en septembre 2000 et paraîtra chez Springer-Verlag.

- **Olivier Lévêque** a préparé sa thèse [LÓ8] à l'HEUDIASYC de l'Université de Technologie de Compiègne et son travail a donné lieu aux publications [LJMW97] et [JWLM99]. Son sujet de thèse était "*Méthodes ensemblistes pour la localisation de véhicules*" et son directeur de thèse était D. Meizel. Il a soutenu le 14 janvier 1998 et est actuellement ingénieur chez ALSTOM Belgium à Charleroi où il travaille sur la localisation des voitures sur un réseau ferré.

- **Michel Kieffer** a préparé au LSS une thèse [Kie99] intitulée "*Estimation ensembliste par analyse par intervalles ; application à la localisation d'un véhicule*", sous la direction d'E. Walter. Michel a soutenu sa thèse le 18 janvier 1999 et son travail a donné lieu aux publications [JKWM98], [KJW98], [KJW99], [KJWM99d] et [KJWM99b], [KJWM99a] et [KJWM99c]. Depuis Septembre 1999, il est Maître de Conférences à l'Université d'Orsay.

Je coencadre actuellement **Isabelle Braems** qui prépare sa thèse, sous la direction d'E. Walter, depuis septembre 1999 au LSS, toujours sur le domaine du calcul par intervalles. Elle vient de soutenir brillamment son stage de DEA [Bra99] sur le sujet : "*Estimation non-linéaire robuste et garantie grâce à l'analyse par intervalles ; Applications en Electrochimie*".

1.3.2 Collaborations scientifiques

J'ai développé de nombreuses autres collaborations scientifiques avec, principalement, des collègues et doctorants de l'Université d'Angers.

- Avec J. Burger (LISA), j'ai utilisé le calcul par intervalles comme outil de démonstration automatique de l'inclusion de deux ensembles définis par des inégalités. Nous avons ensuite appliqué notre méthode à l'analyse de stabilité de modèles incertains [JB99].

- Avec L. Hardouin et J.L. Boimond (LISA), j'ai proposé une nouvelle approche pour l'estimation de paramètres des modèles à événements discrets [JBH99].

- J.L. Godet, A. Elliasmine, Y. Leduff (POMA), E. Walter (L2S) et moi-même, avons travaillé sur l'estimation de constantes de la physique à partir de données expérimentales [JGW+97].

- Avec A. Godon (LISA), j'ai mis au point une méthode capable de trouver (lorsqu'il existe) un chemin reliant deux points d'un même ensemble. L'approche, qui combine analyse par intervalles et théorie des graphes a été appliquée à la planification de trajectoire. Ce travail a été présenté [JG99] lors d'un congrès intitulé "*Applications of Interval Analysis to Systems and Control",* à Girone en février 1999, où j'étais membre du comité de programme.

- Avec S. Brahim-Belhouari, G. Fleury (service des mesures de Supélec), M. Kieffer et E. Walter (L2S), j'ai travaillé sur la sélection de modèle, dans un contexte à erreurs bornées [BBKF+99].

- Avec D. Meizel, O. Lévêque (HEUDIASYC, UTC), M. Kieffer et E. Walter (L2S), j'ai développé un logiciel pour la localisation statique et dynamique d'un robot mobile par des techniques ensemblistes. Cette collaboration est associée aux thèses de O. Lévêque et M. Kieffer [LJMW97] [JWLM99] [JKWM98] [KJW98] [KJWM99c] [KJWM99b].

1.3.3 Comité de lecture de revues internationales

J'ai évalué des articles pour *Automatica*, *IEEE Transactions on Automatic Control*, *International Journal of Control* et *Reliable Computing.*

1.3.4 Activités administratives

J'ai participé au comité de programme du Workshop on *Applications of Interval Analysis to Systems and Control,* Girone, 24-26 février 1999.

Depuis deux ans, j'appartiens aux commissions de spécialistes suivantes :

- 60-63$^{\text{ième}}$ section de l'Université d'Angers, en tant que suppléant (titulaire : F. Chapeau, MCF, LISA).

- 61$^{\text{ième}}$ section de l'Université de Compiègne, en tant que membre extérieur titulaire (suppléant : B. Brogliato CR, CNRS, LAG).

1.4 Activités d'enseignement

De septembre 1994 à juin 1998, j'ai enseigné au département de Physique de la Faculté des Sciences d'Angers. A titre d'exemple, pour l'année scolaire 97-98 (c'est-à-dire la dernière année où j'ai enseigné), mes enseignements (en heures) sont récapitulés dans le tableau ci-dessous.

Niveau	**Matière**	**CM**	**TD**	**TP**
Maîtrise	Automatique[2]	15	15	30
Maîtrise	Electronique[3]	11	11	30
Licence	Electronique[4]		18	
DEUG SM2	Thermodynamique[5]		44	
DEUG SV-ST	Electrostatique[6]		28	

Soit un total d'environ 192 heures équivalent TD. Outre les encadrements de thèse et de DEA, j'ai comptabilisé une trentaine de projets en Licence et Maîtrise de Physique et Application.

En 1994-95, j'ai participé très activement à mise en place des travaux pratiques de la formation CAPES de Physique.

En 1996, j'ai participé à la création d'une salle informatique mise à la disposition des étudiants de Licence et Maîtrise de Physique et Applications.

J'ai mis en place un nouvel enseignement (cours, TD et TP) d'Automatique en Maîtrise de Physique et Applications.

1.5 Présentation du document de recherche

Ce mémoire a pour vocation de présenter en détails une grande partie de mes travaux de recherche effectués depuis ma thèse. Le fil conducteur de l'approche que j'ai choisie est la représentation de chaque variable incertaine d'un problème par un ensemble censé la contenir. Cette approche, dite ensembliste, est encore relativement confidentielle, et

[2] Approche interne : représentation d'état, commandabilité, observabilité, stabilisation par placement de pôles, observateur, simulation. Avec de nombreux TD sous Matlab.

[3] Composants électroniques élémentaires, portes logiques, bascules, compteurs, microprocesseur puis programmation assembleur sur le microcontrôleur MC68HC11.

[4] Diodes, transistors, ampli-opérationnels, montages élémentaires, filtres.

[5] Premier et second principes, machines thermiques.

[6] Rappels mathématiques, champ et force électriques, théorème de Gauss.

s'oppose à l'approche probabiliste beaucoup plus classique. Jusqu'à présent, une formulation ensembliste d'un problème ne permettait d'envisager une résolution fiable que dans le cadre d'exemples très particuliers. Or, si nous nous plaçons sous l'hypothèse idéale que l'on sait manipuler les ensembles comme on sait manipuler les réels, il est très relativement aisé de concevoir un algorithme ensembliste simple apportant la solution au problème. Le principal obstacle qui se présente alors est que la mise en oeuvre du calcul ensembliste semble hors de portée. Pourtant, comme nous allons le montrer dans ce document, en combinant les *sous-pavages* pour représenter les ensembles et l'*analyse par intervalles* comme outil de calcul sur ces sous-pavages, il est possible de mettre en oeuvre le calcul ensembliste. Pour illustrer cette approche, nous traiterons un problème de localisation statique et dynamique d'un robot. Sur ce problème, qui a pourtant fait couler beaucoup d'encre, seule l'approche ensembliste que nous suivrons est capable de donner une solution garantie.

Le document est structuré de la façon suivante. Le chapitre 2 présente l'approche ensembliste, à la fois comme un outil de modélisation d'une variable incertaine et comme un outil de résolution fiable de problèmes non-linéaires. Dans ce même chapitre, les sous-pavages seront introduits afin d'approximer convenablement les sous-ensembles qui nous intéressent et de les rendre facilement manipulables par un ordinateur. Au chapitre 3, les outils de base fournis par l'analyse par intervalles (fonction d'inclusion, test d'inclusion et opérateur de réduction) seront rappelés. Ils permettront une mise en oeuvre du calcul ensembliste, comme nous le montrerons au chapitre 4. Le chapitre 5 s'intéressera à l'application des techniques ensemblistes à l'estimation et à l'observation. Plus particulièrement, nous traiterons l'exemple de la localisation d'un robot mobile sur lequel j'ai travaillé avec les trois doctorants que j'ai coencadrés. Les notations et sigles utilisés sont donnés après la conclusion et avant l'annexe.

1.6 Liste des publications

Les publications dont les numéros apparaissent en gras sont données dans le document annexe.

Revues internationales avec comité de lecture

[1] **L. Jaulin** et E. Walter (1993). Guaranteed nonlinear parameter estimation from bounded-error data via interval analysis, *Math. Comput. Simulation*, **35**, 123-127.

[**2**] **L. Jaulin** et E. Walter (1993). Set inversion via interval analysis, *Automatica*, **29(**4), 1053-1064.

[3] **L. Jaulin** et E. Walter (1993). Guaranteed nonlinear parameter estimation via interval computations, *Interval computation,* **3**, 61-75.

[**4**] E. Walter et **L. Jaulin** (1994). Guaranteed characterization of stability domains via set inversion, *IEEE Trans. on Autom. Control*, **39(**4), 886-889.

[**5**] **L. Jaulin** et E. Walter (1996). Guaranteed tuning, with application to robust control and motion planning. *Automatica,* **32**(8), 1217-1221.

[**6**] O. Didrit, **L. Jaulin** et E. Walter (1995). Guaranteed analysis and optimization of parametric systems, *European Journal of Control*, **3**, 66-80.

[**7**] **L. Jaulin** et E. Walter (1997). Global numerical approach to nonlinear discrete-time control, *IEEE Trans. on Autom. Control,* **42,** 872-875.

[**8**] **L. Jaulin**, J.L Godet, E. Walter, A. Elliasmine et Y. Leduff (1997). Light scattering data analysis via set inversion, *Journal of Physics A: Mathematical and General,* **30**, 7733-7738.

[**9**] **L. Jaulin** et J. Burger (1999), Proving stability of uncertain parametric models, *Automatica*, **35**(4).

[**10**] **L. Jaulin**, J.L. Boimond et L. Hardouin (1999), Estimation via set-inversion: application to discrete-event systems, *Reliable Computing*, **5**(2), 165-173.

[**11**] **L. Jaulin** et E. Walter (1999), Guaranteed bounded-error parameter estimation for nonlinear models with uncertain experimental factors, *Automatica,* **35**(5).

[**12**] M. Kieffer, **L. Jaulin**, E. Walter et D. Meizel, Robust autonomous robot localization using interval analysis, accepté par *Reliable Computing.*

[**13**] **L. Jaulin**, E. Walter, O. Lévêque et D. Meizel, Set inversion for χ algorithms, with application to guaranteed robot localization, soumis à *Automatica.*

[**14**] **L. Jaulin**, M. Kieffer, E Walter et D. Meizel, Guaranteed robust nonlinear estimation, with application to robot localization, soumis à *Automatica*.

[15] M. Kieffer, **L. Jaulin** et E. Walter, Guaranteed recursive nonlinear state estimation using interval analysis, soumis à *IEEE TAC*.

[**16**] **L. Jaulin**, Interval constraint propagation with application to bounded-error estimation, soumis à *Automatica*.

[**17**] **L. Jaulin**, Path planning using intervals and graphs, soumis à *Reliable Computing*.

[**18**] **L. Jaulin**, Reliable Minimax Parameter Estimation, soumis à *Reliable Computing*.

Congrès avec comité de lecture

[19] **L. Jaulin**. et E. Walter (1992). Set inversion, with application to guaranteed nonlinear estimation and robust control, *Proc. of the Workshop on Modeling Techniques for Uncertain Systems*, Sopron.

[20] **L. Jaulin** et E. Walter (1993). Guaranteed nonlinear estimation and robust stability analysis via set inversion. *Proc. 2nd European Control Conference,* Groningen, 818- 821.

[21] **L. Jaulin** et E. Walter (1993). Guaranteed nonlinear parameter estimation via interval computations, *Conference on Numerical Analysis with Automatic Result Verification,* Lafayette. (pas d'actes).

[22] O. Didrit, **L. Jaulin** et E. Walter (1995). Guaranteed analysis and optimization of parametric systems, with application to their stability degree, *Proc. 2nd European Control Conference,* 1412-1417.

[**23**] **L. Jaulin**, E. Walter et O. Didrit (1996). Guaranteed robust nonlinear parameter bounding, *CESA'96 IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)*, Lille, **2**, 1156-1161.

[24] O. Lévêque, **L. Jaulin**, D. Meizel et E. Walter (1997). Vehicle localization from inaccurate telemetric data: a set inversion approach, *Proc. 5th IFAC Symp. on Robot Control SY.RO.CO.'97*, Nantes, France, **1**, 179-186.

[25] M. Kieffer, **L. Jaulin**, E. Walter et D. Meizel, Guaranteed mobile robot tracking using interval analysis, *MISC 99,* Girona, Espagne, 347-360.

[26] **L. Jaulin** et A. Godon. Motion planning using interval analysis, *MISC 99,* Girona, Espagne, 335-346.

[27] M. Kieffer, **L. Jaulin** et E. Walter, Guaranteed recursive nonlinear state estimation

using interval analysis, *CDC 98,* Tampa, Floride, 3966-3971.

[28] S. Brahim-Belhouari, M. Kieffer, G. Fleury, **L. Jaulin** et E. Walter. Model selection via worst-case criterion for nonlinear bounded-error estimation. Proceedings 16th {IEEE} Instrumentation and Measurement Tech. Conf., Venise, 24-26 mai 1999.

[29] M. Kieffer, **L. Jaulin**, E. Walter et D. Meizel (1999). Localisation et suivi garantis d'un robot par analyse par intervalles, JDA'99.

Chapitre d'ouvrages collectifs

[30] **L. Jaulin** et E. Walter (1994). Set inversion, with application to guaranteed nonlinear estimation and robust control, *Modeling Techniques for Uncertain Systems*. A.B. Kurzhanski, V.M. Veliov (Eds), Birkhäuser, Boston. 3-20.

[31] **L. Jaulin** et E. Walter (1996). Guaranteed nonlinear set estimation via interval analysis, *Bounding Approaches to System Identification*, M. Milanese, J. Norton, H. Piet-Lahanier et E. Walter (Eds.), Plenum, New York. 363-382.

[32] M. Kieffer, **L. Jaulin**, E. Walter et D. Meizel (1999). Nonlinear Identification Based on Unreliable Priors and Data, with Application to Robot Localization, In A. Garulli, A. Tesi and A. Vicino (eds.), *Robustness in Identification and Control*, Springer, Londres, 190-203

Chapitre 2

L'approche ensembliste

2.1 Introduction

Dans ce chapitre, nous allons présenter les principes de l'approche ensembliste et montrer son intêret dans les sciences pour l'ingénieur. Une comparaison avec l'approche probabiliste, plus classique, sera alors entreprise. Les bases théoriques du calcul ensembliste seront données, mais nous ferons momentanément l'impasse sur sa mise en oeuvre car celle-ci nécessite l'introduction du calcul par intervalles, présenté au prochain chapitre. Pour approximer les ensembles et les représenter en machine, nous définirons la notion de sous-pavages en section 2.6.

2.2 Motivations

Les ensembles, qui vont nous intéresser, sont des sous-ensembles de $\mathbb{R}^n$. Considérons un sous-ensemble $\mathbb{X}$ de $\mathbb{R}^n$.

- D'un côté, $\mathbb{X}$ peut représenter une variable vectorielle incertaine, que l'on sait contenue dans $\mathbb{X}$. L'ensemble rentre ainsi en compétition avec la densité de probabilité, classiquement utilisée pour modéliser l'incertitude.
- De l'autre côté, $\mathbb{X}$ peut représenter tous les points qu'il contient. Ainsi, une propriété établie sur l'ensemble $\mathbb{X}$ s'applique à tous ses éléments. L'ensemble offre ainsi la possibilité d'obtenir la garantie des résultats tout comme le permet le calcul formel.

L'approche ensembliste offre donc une alternative à la fois à l'approche probabiliste et à l'approche formelle. Cette double nature de l'ensemble, développée dans cette section,

sera régulièrement exploitée dans ce document.

2.2.1 Représentation d'une variable incertaine

Dans ce paragraphe, nous allons présenter et confronter deux approches statistiques (c'est-à-dire capables de manipuler des variables incertaines reliées par des contraintes) : l'approche probabiliste et l'approche ensembliste.

L'approche probabiliste est la plus répandue en mathématiques comme dans les sciences pour l'ingénieur. Dans cette approche, une variable réelle ou vectorielle $\mathbf{x}$ de $\mathbb{R}^n$, est représentée (ou codée) par une *densité de probabilité* (appelée aussi *loi*), c'est-à-dire, par une fonction $\pi : \mathbb{R}^n \to \mathbb{R}^+$ telle que $\int_{\mathbb{R}^n} \pi(x)\, dx = 1$. Ce codage nous donne à la fois un ensemble, appelé *support*, $\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n | \pi(\mathbf{x}) \neq 0\}$ qui contient à coup sûr la variable incertaine $\mathbf{x}$, mais il nous donne aussi une information sur la manière dont est distribuée cette variable aléatoire sur ce support.

Dans une approche ensembliste cette variable est représentée uniquement par un ensemble $\mathbb{X}$, appelé *ensemble de vraisemblance* ou *domaine* , supposé contenir $\mathbf{x}$. Notons qu'il est possible que ce domaine contienne des zones où l'on sait que $\mathbf{x}$ ne se trouve pas, comme l'illustre l'exemple suivant.

Exemple 2.2.1 *Supposons que pour une variable réelle x, le domaine soit $\mathbb{X}_1 = [0,1] \cup [2,3]$. On peut, par commodité dans nos calculs, vouloir avoir un domaine qui soit un intervalle. On prendra alors $\mathbb{X}_2 = [0,3]$ qui est aussi un domaine pour x puisqu'il le contient et pourtant, $\mathbb{X}_2$ contient une zone $]1,2[$ où x ne peut se trouver.* ♢

La représentation ensembliste est donc plus pauvre que la représentation probabiliste, mais elle permet une manipulation beaucoup plus aisée des variables aléatoires. De plus, elle demande moins de connaissances statistiques sur ces variables et permet ainsi de traiter une classe beaucoup plus grande de problèmes.

Tentons maintenant d'expliquer, en nous aidant d'un exemple académique, pourquoi une représentation ensembliste permet une manipulation plus aisée des variables aléatoires, que l'approche probabiliste.

Exemple 2.2.2 *Considérons trois variables aléatoires réelles x, y et z reliées par la contrainte $z = x + y$. Cherchons à déduire les propriétés statistiques de z à partir de celles de x et y par les approches probabiliste puis ensembliste.*

***a) Approche probabiliste**. Supposons que x et y ont des densités de probabilité uniformes $\pi(x) = 1/5$ sur $[0,5]$ et $\pi(y) = 1/2$ sur $[1,3]$. Pour calculer $\pi(z)$, il nous faut une*

information sur le comportement conjoint de x et y. Nous allons donc supposer que x et y sont indépendantes. Ainsi, la densité du couple est uniforme avec $\pi(x,y) = 1/10$ sur le pavé $[0,5] \times [1,3]$. Un calcul d'intégrale élémentaire donne

$$\begin{aligned}
\pi(z) &= 0 \text{ si } z \leq 1 && (2.1)\\
\pi(z) &= \frac{1}{10}(z-1) \text{ si } z \in [1,3] && (2.2)\\
\pi(z) &= \frac{1}{5} \text{ si } z \in [3,6] && (2.3)\\
\pi(z) &= \frac{1}{10}(-z+8) \text{ si } z \in [6,8] && (2.4)\\
\pi(z) &= 0 \text{ si } z \geq 8 && (2.5)
\end{aligned}$$

Mais ce calcul d'intégrale est généralement très difficile à effectuer sauf dans des cas particuliers comme lorsque le couple (x,y) est gaussien ou uniforme.

b) Approche ensembliste. *L'incertitude associée à x et y est donnée par un ensemble. Pour être cohérent avec a), nous allons prendre l'intervalle $[x] = [0,5]$ pour x et l'intervalle $[y] = [1,3]$, pour y. Il est alors clair que z est dans l'intervalle $[z] = [0+1, 5+3] = [1,8]$.*

Sur cet exemple, l'approche ensembliste engendre des calculs plus simples et demande moins d'hypothèses sur le comportement conjoint que l'approche probabiliste. ♢

Représenter un vecteur aléatoire $\mathbf{x}$ par un ensemble $\mathbb{X}$, ne revient pas à supposer que la loi $\pi(\mathbf{x})$ est uniforme sur $\mathbf{x}$. Cela revient juste à supposer que $\mathbf{x}$ est dans $\mathbb{X}$ et rien de plus. En terme probabiliste, cela signifie que le support de la loi $\pi(\mathbf{x})$ est inclus dans $\mathbb{X}$. Dans l'exemple 2.2.2, la variable aléatoire z est représentée par l'intervalle $[1,8]$, et pourtant, $\pi(z)$ n'est pas uniforme sur $[1,8]$.

Faisons maintenant une comparaison entre les approches probabiliste et ensembliste :

- Les manipulations non-linéaires sont plus simples avec l'approche ensembliste qu'avec l'approche probabiliste. Le calcul de $\pi(z)$, de l'exemple 2.2.2, est élémentaire parce que nous avons considéré le cas uniforme et une dépendance linéaire entre x, y et z. Mais en général, la moindre non-linéarité rend l'approche probabiliste difficilement exploitable. Si dans l'exemple 2.2.2, z est donné par $z = x^2.y$, le calcul de $\pi(z)$ devient très difficile. Avec l'approche ensembliste, on obtient immédiatement, pour la variable z, l'intervalle $[z] = [0,5]^2.[1,3] = [0,25].[1,3] = [1,75]$ (ce calcul utilise l'analyse par intervalles qui sera donnée dans le chapitre suivant).

- L'approche probabiliste réclame des connaissances sur les lois conjointes, qui sont souvent difficiles à obtenir en pratique. Dans notre exemple 2.2.2 où $z = x + y$, nous avons supposé que x et y étaient indépendantes. Si cette information n'avait pas

été disponible, nous n'aurions pas pu obtenir une représentation probabiliste pour z. En revanche, l'approche ensembliste ne réclame aucune information statistique sur le couple (x, y). Notons que par simplicité, il est souvent supposé que certaines variables sont indépendantes dans le but d'avoir les densités conjointes[1]. Or très souvent, cette indépendance ne correspond pas à la situation réelle. Par exemple, lorsqu'un même capteur effectue une série de mesures, le biais systématique du capteur engendre une dépendance entre toutes les erreurs de mesure. Le rajout de telles hypothèses produit alors des résultats difficiles à interpréter.

- Il arrive fréquemment que l'erreur de mesure sur une donnée expérimentale soit donnée en terme de tolérance par l'appareil de mesure et non en terme de densité de probabilité. Dans ce cas, l'approche ensembliste est une approche naturelle pour traiter les mesures fournies par le capteur. Par contre, lorsque les propriétés probabilistes des variables aléatoires sont connues, il est clair qu'utiliser une approche ensembliste peut s'avérer inadapté. Dans les problèmes linéaires gaussiens, par exemple, les calculs probabilistes sont élémentaires et très efficaces, si bien qu'une approche ensembliste ne présente aucun intérêt.

- L'incertitude due aux erreurs numériques peut être incorporée lors du calcul des domaines, ce qui semble être très difficile pour l'approche probabiliste. Si $x \in \mathbb{X}$, $y \in \mathbb{Y}$, et $z = x + y$, le domaine $\mathbb{Z}$ pour z, sera donné par

$$\mathbb{Z} = \mathbb{X} + \mathbb{Y} + \mathbb{E} \triangleq \{x + y + e | x \in \mathbb{X}, y \in \mathbb{Y}, e \in \mathbb{E}\} \tag{2.6}$$

  où $\mathbb{E}$ est l'ensemble représente le pessimisme introduit lors du calcul de $\mathbb{Z}$.

- Enfin une représentation ensembliste permet l'introduction de formalismes et d'outils nouveaux autorisant la résolution de problèmes non-linéaires dont le traitement semble impensable avec une approche probabiliste.

Malgré tous ses avantages, l'approche ensembliste reste encore confidentielle dans le milieu des sciences pour l'ingénieur. On peut la trouver par exemple en estimation où elle est connue sous le nom d'approche à erreurs bornées (voir par exemple, [Wal90] et [MNPLW96]) et en commande robuste [Bar94].

[1] Considérons 3 variables aléatoires x, y et z ayant pour lois $\pi(x), \pi(y)$ et $\pi(z)$. La loi conjointe $\pi(x, y, z)$, qui représente le comportement statistique collectif de ces trois variables, ne peut se déduire des lois $\pi(x), \pi(y)$ et $\pi(z)$ que dans certains cas particuliers. Par exemple lorsque x, y et z sont indépendantes, on a $\pi(x, y, z) = \pi(x).\pi(y).\pi(z)$.

2.2.2 Recouvrement d'un espace de recherche

Beaucoup d'algorithmes, utilisés dans les sciences pour l'ingénieur, consistent à balayer un espace de recherche (contenant une infinité non dénombrable de points) dans le but de trouver des solutions à un problème ou bien de montrer qu'une propriété est valide pour tout cet espace. Une approche ponctuelle ne permet d'analyser qu'une infime portion de cet espace de recherche. Rien ne permet alors de conclure avec certitude, sauf pour des cas particuliers (problèmes convexes ou linéaires), que la ou les solutions du problème ont été trouvées.

Or, l'espace de recherche peut fréquemment être recouvert par un nombre fini de sous-ensembles simples, sur lesquels on sait calculer, et qui eux, contiennent une infinité de points. On peut ainsi résoudre avec garantie des problèmes de démonstration automatique ou de recherche de solutions de problèmes non-linéaires, en s'aidant des principes suivants :

- Si une même propriété est établie pour chacun des sous-ensembles d'une partition de l'espace de recherche, alors cette propriété est valide pour l'espace de recherche tout entier.
- Si l'on arrive à prouver qu'un sous-ensemble ne contient aucune solution du problème, on peut l'éliminer de l'espace de recherche.
- Si l'on n'arrive pas à prouver qu'un sous-ensemble ne contient aucune solution, on peut toujours le décomposer en plusieurs sous-ensembles sur lesquels on recommence une analyse.

Parmi la classe des problèmes qui vérifient ces principes, on trouve les algorithmes de type *branch and bound* ainsi que tous les algorithmes qui seront présentés dans ce manuscrit. Notons que les méthodes formelles permettent, elles aussi, d'obtenir des résultats garantis. Mais la classe des problèmes qu'elles peuvent traiter en un temps convenable est très réduite. Nous verrons que pour les applications qui vont nous intéresser au chapitre 5, il est impensable d'espérer obtenir des résultats avec des méthodes formelles.

2.3 Opérations sur les ensembles

Les opérations que l'on peut effectuer sur les ensembles sont de deux types :

- *Les opérations étendues* : Ce sont des opérations qui sont déjà définies pour les élements des ensembles considérés et dont la définition est étendue aux ensembles (addition, produit scalaire, ...).

- *Les opérations ensemblistes pures* : ces opérations ont peu de sens si l'on considère les éléments des ensembles (union, intersection, produit cartésien ...).

2.3.1 Les opérations étendues

Soient $\mathbb{X}$ un ensemble de $\mathbb{R}^n$, $\mathbb{Y}$ un ensemble de $\mathbb{R}^m$ et $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, une fonction vectorielle. On définit

$$\begin{array}{lll} \text{l'}\textit{image directe}\text{ de } \mathbb{X} \text{ par } \mathbf{f} & \mathbf{f}(\mathbb{X}) & \triangleq \{\mathbf{y} \in \mathbb{R}^m | \exists \mathbf{x} \in \mathbb{X}, \mathbf{f}(\mathbf{x}) = \mathbf{y}\}, \\ \text{l'}\textit{image inverse}\text{ de } \mathbb{Y} \text{ par } \mathbf{f} & \mathbf{f}^{-1}(\mathbb{Y}) & \triangleq \{\mathbf{x} \in \mathbb{R}^n | \exists \mathbf{y} \in \mathbb{Y}, \mathbf{f}(\mathbf{x}) = \mathbf{y}\}. \end{array} \tag{2.7}$$

Parmi les propriétés importantes de ces ensembles rappelons

$$\mathbf{f}(\mathbb{X}_1 \cap \mathbb{X}_2) \subset \mathbf{f}(\mathbb{X}_1) \cap \mathbf{f}(\mathbb{X}_2), \tag{2.8}$$

$$\mathbf{f}(\mathbb{X}_1 \cup \mathbb{X}_2) = \mathbf{f}(\mathbb{X}_1) \cup \mathbf{f}(\mathbb{X}_2), \tag{2.9}$$

$$\mathbf{f}^{-1}(\mathbb{Y}_1 \cap \mathbb{Y}_2) = \mathbf{f}^{-1}(\mathbb{Y}_1) \cap \mathbf{f}^{-1}(\mathbb{Y}_2), \tag{2.10}$$

$$\mathbf{f}^{-1}(\mathbb{Y}_1 \cup \mathbb{Y}_2) = \mathbf{f}^{-1}(\mathbb{Y}_1) \cup \mathbf{f}^{-1}(\mathbb{Y}_2), \tag{2.11}$$

$$\mathbf{f}\left(\mathbf{f}^{-1}(\mathbb{Y})\right) \subset \mathbb{Y}, \tag{2.12}$$

$$\mathbf{f}^{-1}(\mathbf{f}(\mathbb{X})) \supset \mathbb{X}. \tag{2.13}$$

De même, les opérateurs classiques peuvent être aussi définis pour les ensembles. Ainsi, si $\mathbb{X}$ et $\mathbb{Y}$ sont deux sous-ensembles de $\mathbb{R}^n$,

$$\mathbb{X} + \mathbb{Y} \triangleq \{\mathbf{x} + \mathbf{y}, \mathbf{x} \in \mathbb{X}, \mathbf{y} \in \mathbb{Y}\}. \tag{2.14}$$

Notons que ce type d'extension peut entraîner certaines confusions. Par exemple $\mathbb{X} - \mathbb{X}$ doit être compris comme l'ensemble $\{\mathbf{x} - \mathbf{y}, \mathbf{x} \in \mathbb{X}, \mathbf{y} \in \mathbb{X}\}$ et non pas comme l'ensemble $\{\mathbf{x} - \mathbf{x}, \mathbf{x} \in \mathbb{X}\} = \{\mathbf{0}\}$. Pourtant, si $\mathbf{f}(\mathbf{x}) = \mathbf{x} - \mathbf{x}$, $\mathbf{f}(\mathbb{X}) = \{\mathbf{0}\}$.

2.3.2 Les opérations ensemblistes pures

Elles regroupent, par exemple, la projection canonique, l'union, l'intersection et le produit cartésien. Définissons ces opérations dans le cas où $\mathbb{X}$ et $\mathbb{Y}$ sont deux sous-ensembles de $\mathbb{R}^n$ et $\mathbb{R}^m$.

$$\text{proj}_i(\mathbb{X}) \triangleq \{x_i \in \mathbb{R} | \exists \mathbf{x} = (x_1, \ldots, x_i, \ldots, x_n)^{\mathrm{T}} \in \mathbb{R}^n, \mathbf{x} \in \mathbb{X}\}, \tag{2.15}$$

$$\mathbb{X} \cap \mathbb{Y} \triangleq \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} \in \mathbb{X} \text{ et } \mathbf{x} \in \mathbb{Y}\}, \text{ avec } n = m, \tag{2.16}$$

$$\mathbb{X} \cup \mathbb{Y} \triangleq \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} \in \mathbb{X} \text{ ou } \mathbf{x} \in \mathbb{Y}\}, \text{ avec } n = m, \tag{2.17}$$

$$\mathbb{X} \times \mathbb{Y} \triangleq \left\{ \left(\mathbf{x}^{\mathrm{T}} \ \mathbf{y}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{n+m} | \mathbf{x} \in \mathbb{X} \text{ et } \mathbf{y} \in \mathbb{Y} \right\}. \tag{2.18}$$

De plus, nous utiliserons les relations binaires ensemblistes classiques

$$\mathbb{X} \subset \mathbb{Y} \Leftrightarrow \forall \mathbf{x} \in \mathbb{X}, \mathbf{x} \in \mathbb{Y}, \tag{2.19}$$

$$\mathbb{X} = \mathbb{Y} \Leftrightarrow \mathbb{X} \subset \mathbb{Y} \text{ et } \mathbb{Y} \subset \mathbb{X}. \tag{2.20}$$

2.3.3 Pessimisme dû aux occurrences multiples

Une expression ensembliste du type

$$\mathbb{Z} = \mathbb{X} - \mathbb{X} \tag{2.21}$$

peut *a priori* s'interpréter de la façon suivante

$$\mathbb{Z} = \{\mathbf{x} - \mathbf{x},\ \mathbf{x} \in \mathbb{X}\}. \tag{2.22}$$

Ainsi $\mathbb{Z}$ se réduit à un singleton qui contient pour seul élément le vecteur nul. Nous l'avons vu, cette interprétation est fausse et la confusion vient du fait que, dans l'expression ensembliste, des variables sont associées à ces ensembles. L'ambiguïté disparaît si les ensembles sont manipulés sans les nommer par l'intermédiaire d'une variable. Par exemple, il nous semble naturel d'écrire $\mathbb{Z} = [1,2] - [1,2] \Leftrightarrow \mathbb{Z} = [-1,1]$ car l'intervalle $[1,2]$ n'a pas été nommé par une variable $\mathbb{X}$. L'expression (2.21) doit donc se comprendre ainsi

$$\mathbb{Z} = \{\mathbf{f}(\mathbf{x},\mathbf{y}),\ \mathbf{x} \in \mathbb{X}, \mathbf{y} \in \mathbb{X}\} \text{ avec } \mathbf{f}(\mathbf{x},\mathbf{y}) = \mathbf{x} - \mathbf{y}. \tag{2.23}$$

L'écriture d'une expression ensembliste oublie les relations de dépendance qui peuvent exister entre les variables représentées par les ensembles. Ce problème est connu sous le nom du *problème de dépendance* et a été abondamment étudié dans la littérature (voir par exemple [Moo79]). Il faudra donc résister à la tentation d'écrire des équivalences[2] du type

$$\mathbb{Z} = (\mathbb{X} + \mathbb{Y}) - \mathbb{X} \Leftrightarrow \mathbb{Z} + \mathbb{X} = (\mathbb{X} + \mathbb{Y}).$$

Par contre, il est correct d'écrire

$$\mathbb{X} + \mathbb{X} - \mathbb{X} - \mathbb{X} = 2(\mathbb{X} - \mathbb{X}). \tag{2.24}$$

L'origine mathématique de ces ambiguïtés vient du fait que beaucoup de propriétés de calcul dans $\mathbb{R}^n$ ne se généralisent pas aux parties $\mathcal{P}(\mathbb{R}^n)$ de $\mathbb{R}^n$. Par exemple $(\mathbb{R}^n, +)$ est un groupe, alors que $(\mathcal{P}(\mathbb{R}^n), +)$ est seulement un monoïde car, pour tout $\mathbb{X}$ différent du singleton $\{\mathbf{0}\}$ (élément neutre pour l'addition), il n'existe pas d'ensemble $\mathbb{Y}$ tel que

[2] Cette équivalence est bien évidemment fausse. Pour s'en convaincre, on peut construire un contre-exemple en prenant $\mathbb{Y} = \{0\}$, $\mathbb{X} = [0,1]$ et $\mathbb{Z} = [-1,1]$. La proposition de gauche de trouve satisfaite, contrairement à la proposition de droite.

$\mathbb{X} + \mathbb{Y} = \{\mathbf{0}\}$. Puisque la plupart des propriétés algébriques du calcul dans $\mathbb{R}^n$, ne s'étendent pas aux ensembles de $\mathbb{R}^n$, nous nous efforcerons, dans ce manuscrit, d'éviter toute manipulation algébrique sur les ensembles.

Soit f une fonction $\mathbb{R}^n \to \mathbb{R}$, $(x_1, \ldots, x_n) \to y$ dont on connaît une expression permettant de l'évaluer. Si $\mathbb{F}(\mathbb{X}_1, \ldots, \mathbb{X}_n)$ est le résultat d'un calcul ensembliste sur $\mathbb{X}$, utilisant cette expression pour f. Alors, on a l'inclusion

$$f(\mathbb{X}_1, \ldots, \mathbb{X}_n) \subset \mathbb{F}(\mathbb{X}_1, \ldots, \mathbb{X}_n), \tag{2.25}$$

où

$$f(\mathbb{X}_1, \ldots, \mathbb{X}_n) = \{f(x_1, \ldots, x_n), x_1 \in \mathbb{X}_1, \ldots, x_n \in \mathbb{X}_n\}.$$

On dit que l'évaluation ensembliste est *pessimiste*. Cette inclusion peut être une égalité dans certains cas particuliers, mais en général, elle ne l'est pas. Cela provient du problème de dépendance que nous rappelons : Si, dans l'expression de f, certains x_i apparaîssent plusieurs fois, alors $f(\mathbb{X}) \subset \mathbb{F}(\mathbb{X})$, sinon, $f(\mathbb{X}) = \mathbb{F}(\mathbb{X})$.

Exemple 2.3.1 *Si* $f(x_1, x_2) = x_1 + x_2 - x_1$*;*

$$f(\mathbb{X}_1, \mathbb{X}_2) = \{x_1 + x_2 - x_1 | x_1 \in \mathbb{X}_1, x_2 \in \mathbb{X}_2\} = \{x_2 | x_2 \in \mathbb{X}_2\} = \mathbb{X}_2. \tag{2.26}$$

Alors que

$$\mathbb{F}(\mathbb{X}_1, \mathbb{X}_2) = \mathbb{X}_1 + \mathbb{X}_2 - \mathbb{X}_1 = \{x_1 + x_2 - x_3 | x_1 \in \mathbb{X}_1, x_2 \in \mathbb{X}_2, x_3 \in \mathbb{X}_1\} \tag{2.27}$$

contient $f(\mathbb{X}_1, \mathbb{X}_2)$*. La dépendance entre* x_1 *et* x_3 *(qui est* $x_1 = x_3$*) n'apparaît pas dans cette expression rajoutant ainsi un degré de liberté dans la fabrication de l'ensemble* $\mathbb{F}(\mathbb{X}_1, \mathbb{X}_2)$*.*

2.4 Représentation des ensembles

2.4.1 Différents types de représentation

La représentation exacte ou approchée de sous-ensembles de $\mathbb{R}^n$ est fondamentale dans notre approche. Elle doit être suffisamment simple pour permettre des manipulations efficaces des sous-ensembles. Elle doit également être suffisamment riche pour représenter tous les sous-ensembles qui nous intéressent et qui sont parfois non connexes et avec tout type de formes. Parmi les approches classiques pour représenter les sous-ensembles de $\mathbb{R}^n$, on trouve :

- *La représentation par équations et inéquations* : Cette représentation permet une manipulation exacte des ensembles. Malheureusement, les algorithmes qui en découlent utilisent le calcul formel et ne s'appliquent qu'à une classe réduite d'équations ou d'inéquations. Ils sont souvent incapables de réaliser certaines opérations élémentaires sur les ensembles (essayez par exemple de mettre sous forme d'inéquations la somme de deux ellipses de $\mathbb{R}^2$). De plus, la complexité de ces algorithmes (en temps de calcul et en place mémoire) les rend souvent inexploitables, même pour des problèmes académiques.

- *La représentation par nuage de points* : Un ensemble $\mathbb{X}$ est représenté par un nuage de points contenus dans $\mathbb{X}$. Ces points peuvent avoir été obtenus par un tirage aléatoire (approche Monte-Carlo), ou par un balayage déterministe. Cette représentation est très approximative et l'interprétation du résultat est très souvent arbitraire. Que penser par exemple lorsque le nuage est vide ? Dans ce cas, la seule propriété de $\mathbb{X}$ que l'on peut garantir est qu'il existe des points qui ne sont pas dans $\mathbb{X}$. En revanche, la manipulation des nuages de points est très facile.

- *La représentation par recouvrement* : Il convient tout d'abord de définir une classe de sous-ensembles de $\mathbb{R}^n$ que l'on sait représenter et manipuler aisément. Les éléments de cette classe sont appelés des *récipients* (car ils sont souvent intéressants de par ce qu'ils sont censés contenir). Cette classe peut être l'ensemble des ellipsoïdes, des pavés ou des polytopes, zonotopes (voir figure 2.1) ... Pour une classe de récipients donnée, un ensemble $\mathbb{X}$ peut généralement être représenté par une union de récipients qui recouvre complètement $\mathbb{X}$. Beaucoup de caractéristiques intéressantes de $\mathbb{X}$ se trouvent contenues dans cette union. Par exemple, nous avons la garantie que si le recouvrement est vide, l'ensemble $\mathbb{X}$ est lui aussi vide. Dans ce document, les récipients considérés seront les pavés de $\mathbb{R}^n$.

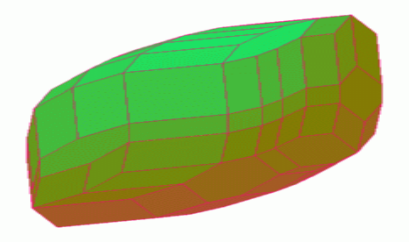

Figure 2.1: Un zonotope de $\mathbb{R}^3$.

- *La représentation par encadrement* : Dans cette approche présentée dans ma thèse [Jau94], on considère une classe de récipients telle que tout compact $\mathbb{X}$ de $\mathbb{R}^n$ peut

être encadré par deux unions de récipients $\mathbb{X}^-$ et $\mathbb{X}^+$ dans le sens où

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+. \tag{2.28}$$

Une connaissance du couple $[\mathbb{X}^-, \mathbb{X}^+]$ nous donne de précieuses informations sur $\mathbb{X}$. Par exemple, $\text{vol}(\mathbb{X}^-) \leq \text{vol}(\mathbb{X}) \leq \text{vol}(\mathbb{X}^+)$, ou encore, si $\mathbb{X}^+$ est vide, alors $\mathbb{X}$ est aussi vide. On peut même en déduire des informations sur les propriétés de connexité de $\mathbb{X}$. Par exemple, si $\mathbb{X}^-$ et $\mathbb{X}^+$ sont constitués chacun de deux composantes connexes disjointes : $\mathbb{X}^- = \mathbb{X}_1^- \cup \mathbb{X}_2^-$ et $\mathbb{X}^+ = \mathbb{X}_1^+ \cup \mathbb{X}_2^+$, et si, de plus, $\mathbb{X}_1^- \subset \mathbb{X}_1^+$, $\mathbb{X}_2^- \subset \mathbb{X}_2^+$ alors, $\mathbb{X}$ est non connexe. Notons que de telles informations sur la connexité ne seraient pas disponibles avec les représentations par nuage de points ou par recouvrement.

Dans ce manuscrit, nous allons principalement utiliser une représentation par recouvrement où les récipients sont des pavés.

2.4.2 Distances entre compacts

Pour évaluer la qualité d'une représentation, il convient de définir différentes distances entre deux compacts de $\mathbb{R}^n$. Munissons $\mathbb{R}^n$ de la distance

$$L_\infty(\mathbf{x}, \mathbf{y}) \triangleq \max_{i \in \{1,\ldots,n\}} |y_i - x_i| \tag{2.29}$$

La *proximité* de $\mathbb{A}$ à $\mathbb{B}$ est définie par

$$h_\infty^0(\mathbb{A}, \mathbb{B}) \triangleq \inf\left\{r \in \mathbb{R}^+ | \mathbb{A} \subset \mathbb{B} + r\mathbb{U}\right\} \tag{2.30}$$

où $\mathbb{U}$ est la sphère unité de $(\mathbb{R}^n, L_\infty)$. La *distance de Hausdorff* (voir [Ber79]) entre deux sous-ensembles $\mathbb{A}$ à $\mathbb{B}$ de $\mathbb{R}^n$ est donnée par

$$h_\infty(\mathbb{A}, \mathbb{B}) \triangleq \max\left\{h_\infty^0(\mathbb{A}, \mathbb{B}), h_\infty^0(\mathbb{B}, \mathbb{A})\right\} \tag{2.31}$$

La distance de Hausdorff est une distance dans l'ensemble des compacts de $\mathbb{R}^n$ puisqu'elle vérifie

$$\begin{array}{lll} \textit{(i)} & \text{séparabilité} & h_\infty(\mathbb{A}, \mathbb{B}) = 0 \Rightarrow \mathbb{A} = \mathbb{B} \\ \textit{(ii)} & \text{symétrie} & h_\infty(\mathbb{A}, \mathbb{B}) = h_\infty(\mathbb{B}, \mathbb{A}) \\ \textit{(iii)} & \text{inégalité triangulaire} & h_\infty(\mathbb{A}, \mathbb{C}) \leq h_\infty(\mathbb{A}, \mathbb{B}) + h_\infty(\mathbb{B}, \mathbb{C}) \end{array} \tag{2.32}$$

2.5 Approximation du calcul ensembliste

Considérons une classe $\mathcal{R}$ de sous-ensembles de $\mathbb{R}^n$ dont les éléments sont appelés récipients. Le moindre calcul sur des récipients, comme nous l'avons défini dans un contexte

ensembliste, génère un sous-ensemble de $\mathbb{R}^n$ qui n'est, en général, plus un récipient. Par exemple, si les récipients considérés sont les ellipses de $\mathbb{R}^2$, la somme de deux ellipses n'est généralement plus une ellipse. Pour rendre les opérations internes dans $\mathcal{R}$, on peut utiliser le *calcul extérieur :*

$$f_{\mathcal{R}}\left(\mathbb{X}_1,\ldots,\mathbb{X}_n\right)=\square\left(f\left(\mathbb{X}_1,\ldots,\mathbb{X}_n\right)\right) \tag{2.33}$$

où $\mathbb{X}_1,\ldots,\mathbb{X}_n$ sont des récipients et $\square(\mathbb{A})$ désigne le plus petit récipient (au sens de l'inclusion) qui contient $\mathbb{A}$. L'ensemble ainsi obtenu donne des informations sur le calcul ensembliste idéal associé. L'opérateur $\square$ peut être mis en oeuvre pour des opérations relativement élémentaires. Lorsqu'on a affaire à la composition de fonctions élémentaires, un pessimisme est introduit :

$$goh\left(\mathbb{X}_1,\ldots,\mathbb{X}_n\right)\subset\square\left(goh\left(\mathbb{X}_1,\ldots,\mathbb{X}_n\right)\right)\subset\square\left(g\left(\square h\left(\mathbb{X}_1,\ldots,\mathbb{X}_n\right)\right)\right) \tag{2.34}$$

De ce fait, comme beaucoup de fonctions intervenant dans des problèmes concrets sont des compositions de fonctions élémentaires, il est souvent aisé de trouver un récipient qui contienne l'ensemble $\mathbb{Y}$ résultant d'un calcul ensembliste. Par contre, il est rare qu'on puisse avoir le plus petit.

2.6 Les sous-pavages

Les sous-pavages forment une classe d'ensembles de $\mathbb{R}^n$ faciles à manipuler à l'aide d'un ordinateur et qui permettent d'approximer avec une précision arbitraire une grande classe de sous-ensembles compacts[3] de $\mathbb{R}^n$ tout en conservant certaines de leurs propriétés comme la forme, la connexité ... Comme un sous-pavage est formé de pavés, il convient, tout d'abord, de fixer les notations et les définitions pour les pavés. Ensuite, nous introduirons mathématiquement les notions de sous-pavage et de pavage et décrirons leur représentation en machine ainsi que quelques algorithmes élémentaires qui serviront à les manipuler.

2.6.1 Pavés

Un *pavé* ou *vecteur intervalle* $[\mathbf{x}]$ de $\mathbb{R}^n$ est le produit cartésien de n intervalles :

$$[\mathbf{x}]=[\underline{x}_1,\bar{x}_1]\times\cdots\times[\underline{x}_n,\bar{x}_n]=[x]_1\times\cdots\times[x]_n\,. \tag{2.35}$$

L'ensemble des pavés est dénoté $\mathbb{IR}^n$. La *longueur* $w([\mathbf{x}]\,)$ (en anglais : *width*) d'un pavé $[\mathbf{x}]$ est la longueur du plus grand de ses côtés. Posons

$$j\triangleq\min\{i|\bar{x}_i-\underline{x}_i=w([\mathbf{x}]\,)\} \tag{2.36}$$

[3] On rappelle qu'un compact de $\mathbb{R}^n$ est un ensemble fermé et borné de $\mathbb{R}^n$.

Définissons les deux pavés $L[\mathbf{x}]$ et $R[\mathbf{x}]$ comme suit :

$$\begin{array}{rcl} L[\mathbf{x}] & \triangleq & [\underline{x}_1, \bar{x}_1] \times \cdots \times \left[\underline{x}_j, \frac{\underline{x}_j + \bar{x}_j}{2}\right] \times \cdots \times [\underline{x}_n, \bar{x}_n] \\ R[\mathbf{x}] & \triangleq & [\underline{x}_1, \bar{x}_1] \times \cdots \times \left[\frac{\underline{x}_j + \bar{x}_j}{2}, \bar{x}_j\right] \times \cdots \times [\underline{x}_n, \bar{x}_n] \end{array} \tag{2.37}$$

$L[\mathbf{x}]$ est appelé *fils gauche* de $[\mathbf{x}]$ et $R[\mathbf{x}]$, *fils droit* de $[\mathbf{x}]$. Ici L et R peuvent être vus comme des opérateurs de $\mathbb{IR}^n$ vers $\mathbb{IR}^n$. L'opération qui consiste à générer ces deux fils à partir de $[\mathbf{x}]$ est appelée *bissection* de $[\mathbf{x}]$. Les deux sous-pavés $L[\mathbf{x}]$ et $R[\mathbf{x}]$ sont dits frères. Par exemple, frère$(RRRL[\mathbf{x}]) = LRRL[\mathbf{x}]$. La *réunification* est l'opération qui consiste à réunir deux pavés frères $L[\mathbf{x}]$ et $R[\mathbf{x}]$ pour n'en former qu'un seul, le père $[\mathbf{x}]$. On notera $[\mathbf{x}] := (L[\mathbf{x}] \,|\, R[\mathbf{x}])$.

Soit $\mathbb{X}$ un ensemble compact de $\mathbb{R}^n$. On appelle *pavé enveloppe* de $\mathbb{X}$ le plus petit pavé qui contient $\mathbb{X}$. On le notera $[\mathbb{X}]$. Soient $\mathbb{X}$ et $\mathbb{Y}$ deux compacts de $\mathbb{R}^n$, le pavé enveloppe de leur union est défini par

$$\mathbb{A} \sqcup \mathbb{B} = [\mathbb{A} \cup \mathbb{B}]\,. \tag{2.38}$$

2.6.2 Pavages et sous-pavages

Un *sous-pavage* d'un pavé $[\mathbf{x}] \subset \mathbb{R}^n$ est une union de sous-pavés de $[\mathbf{x}]$ qui ne se chevauchent pas. Néanmoins, deux pavés d'un même sous-pavage peuvent avoir une intersection non-vide (c'est le cas s'ils ont une frontière commune), mais, leurs intérieurs sont tous deux à deux disjoints. Lorsqu'un sous-pavage $\widehat{\mathcal{P}}$ recouvre tout le pavé $[\mathbf{x}]$, $\widehat{\mathcal{P}}$ est appelé pavage de $[\mathbf{x}]$. Un sous-pavage de $[\mathbf{x}]$ est dit *régulier* si chacun de ses pavés peut être obtenu par bissections successives de $[\mathbf{x}]$. L'ensemble de tous les sous-pavages réguliers de $[\mathbf{x}]$ est dénoté $\mathcal{SPR}([\mathbf{x}])$. Les sous-pavages réguliers permettent d'approximer une très grande classe d'ensembles compacts de $\mathbb{R}^n$ et, comme nous allons le montrer, ils peuvent être manipulés par un ordinateur très facilement. Nous allons aussi manipuler des sous-pavages non réguliers, mais les opérations ensemblistes associées comme l'intersection ou le test d'appartenance d'un pavé seront beaucoup plus lourdes. Un pavage régulier $\widehat{\mathcal{P}} = \{[\mathbf{x}]_1, \ldots, [\mathbf{x}]_9\}$ du pavé $[\mathbf{x}] = [0,8] \times [0,8]$ est représenté sur la figure 2.2.

Les boîtes grises forment un sous-pavage régulier $\widehat{\mathcal{Q}}$ de $[\mathbf{x}] = [0,8] \times [0,8]$. Remarquons que $\widehat{\mathcal{Q}}$ a un double statut : il peut être vu comme une liste finie de pavés

$$\begin{array}{rcl} \widehat{\mathcal{Q}} & = & \{LL[\mathbf{x}], LRR[\mathbf{x}], LLRRR[\mathbf{x}], LLRRRR[\mathbf{x}]\} \\ & = & \{[0,4] \times [0,4]; [4,6] \times [4,8]; [6,7] \times [4,6]; [6,7] \times [6,7]\}, \end{array} \tag{2.39}$$

et dans ce cas, on notera $[0,4]^2 \in \widehat{\mathcal{Q}}$ et $[0,1]^2 \notin \widehat{\mathcal{Q}}$. Mais $\widehat{\mathcal{Q}}$ peut aussi être vu comme un sous-ensemble de $\mathbb{R}^2$. Ainsi, on notera $[0,1]^2 \subset \widehat{\mathcal{Q}} \subset \mathbb{R}^2$. Ce double statut peut parfois engendrer quelques confusions. Par exemple, que penser d'une inclusion du type $\widehat{\mathcal{P}} \subset \widehat{\mathcal{Q}}$,

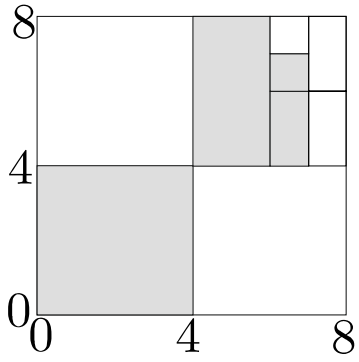

Figure 2.2: Sous-pavage et pavage réguliers d'un pavé

lorsque $\widehat{\mathcal{P}} = \{[0,3]\,;[4,5]\}$ et $\widehat{\mathcal{Q}} = \{[0,9]\}$? Elle est vraie si $\widehat{\mathcal{P}}$ et $\widehat{\mathcal{Q}}$ sont vus comme des sous-ensembles de $\mathbb{R}$, mais fausse s'ils sont vus comme des listes d'intervalles. Dans ce manuscrit, nous ferons en sorte que de telles ambiguïtés ne se présentent pas et puissent être levées en s'aidant du contexte.

A partir d'un sous-pavage régulier $\widehat{\mathcal{Q}}$ de $\mathcal{SPR}([\mathbf{x}])$, on peut engendrer deux sous-pavages

$$L\widehat{\mathcal{Q}} = \{[y] \in \widehat{\mathcal{Q}} |\, [y] \subset L[\mathbf{x}]\}, \tag{2.40}$$

$$R\widehat{\mathcal{Q}} = \{[y] \in \widehat{\mathcal{Q}} |\, [y] \subset R[\mathbf{x}]\}. \tag{2.41}$$

Ces deux sous-pavages sont appelés fils gauche et droit respectivement. Par exemple, dans le sous-pavage de la figure 2.2, on a

$$\begin{array}{rcl} L\widehat{\mathcal{Q}} & = & \{LL[\mathbf{x}]\} = \{[0,4] \times [0,4]\} \\ R\widehat{\mathcal{Q}} & = & \{LRR[\mathbf{x}], LLRRR[\mathbf{x}], LLRRRR[\mathbf{x}]\} \\ & = & = \{[4,6] \times [4,8]; [6,7] \times [4,6]; [6,7] \times [6,7]\}. \end{array} \tag{2.42}$$

Les sous-pavages seront utilisés pour approximer les sous-ensembles de $\mathbb{R}^n$. La figure 2.3 donne une illustration de l'encadrement par sous-pavages de l'anneau

$$\mathbb{S} = \{(x,y) | x^2 + y^2 \in [1,2]\}, \tag{2.43}$$

avec des précisions différentes. Le cadre correspond au pavé $[-2,2] \times [-2,2]$. Le sous-pavage $\Delta\widehat{\mathbb{S}}$ formé des pavés en gris contient la frontière de $\mathbb{S}$ alors que le sous-pavage $\widehat{\mathbb{S}}^-$ formé des pavés blancs dans l'anneau est inclus dans $\mathbb{S}$. On a donc l'encadrement

$$\widehat{\mathbb{S}}^- \subset \mathbb{S} \subset \widehat{\mathbb{S}}^+ \text{ avec } \widehat{\mathbb{S}}^+ \triangleq \widehat{\mathbb{S}}^- \cup \Delta\widehat{\mathbb{S}}. \tag{2.44}$$

Les sous-pavages ne peuvent généralement approximer avec une précision raisonnable que des compacts $\mathbb{X}$ de $\mathbb{R}^n$, où n est faible (typiquement inférieur à 5). Toutefois, lorsque $\mathbb{X}$ est minuscule, quelques pavés peuvent suffire pour son approximation, même lorsque la dimension de $\mathbb{X}$ est grande.

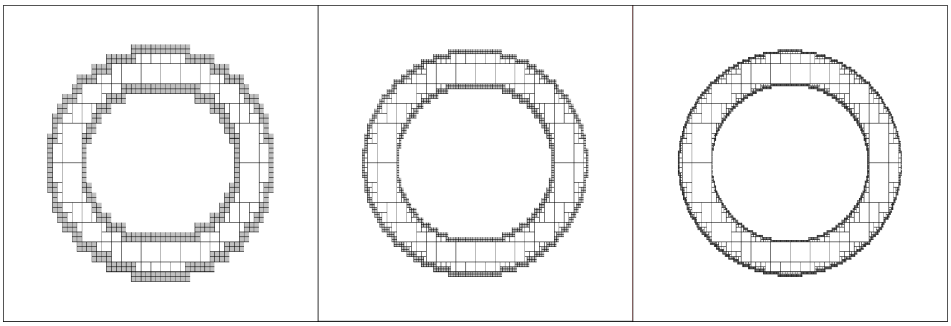

Figure 2.3: Encadrement d'un ensemble entre deux sous-pavages, pour différentes précisions

2.6.3 Représentation binaire d'un sous-pavage régulier

En machine, un sous-pavage régulier peut être représenté par un arbre binaire. La figure 2.4 contient l'arbre associé au sous-pavage de la figure 2.2, où 1, signifie *appartient au sous-pavage*. Un arbre binaire $\mathcal{T}$ peut aussi être codé par une chaîne en notation polonaise inversée[4] $p(\mathcal{T})$, obtenue comme suit : Si $\mathcal{T}$ est une feuille, c'est-à-dire que $\mathcal{T}$ n'a qu'un noeud qui est sa racine, alors $p(\mathcal{T}) = \mathcal{T}$, sinon $p(\mathcal{T}) = p(L\mathcal{T})|p(R\mathcal{T})|\{c\}$, où $|$ est l'opérateur de concaténation et c est un code signifiant *couper*. Par exemple si $\mathcal{T}$ est l'arbre de la figure 2.4, $p(\mathcal{T}) = \{10c0110c10c0ccccc\}$[5]. Cette séquence peut être codée en binaire en remplaçant c par 1, 0 par 00 et 1 par 01. Les c de fin de séquence ne sont bien sûr pas nécessaires pour retrouver l'arbre. $\mathcal{T}$ est alors représenté par la séquence binaire suivante $p_2(\mathcal{T}) = \{0100100010100101001 00\}$. Remarquons que ce codage prend moins de 3 octets. Ce type de codage par séquences binaires a été abondamment étudié dans la littérature. Des manipulations comme les intersections ou les unions de sous-pavages peuvent être faites beaucoup plus rapidement si on travaille directement sur les séquences binaires avec un codage adapté (voir par exemple [Pru90]).

2.6.4 Opérations sur les sous-pavages réguliers

Nous allons maintenant présenter quelques opérations de base sur les sous-pavages réguliers, à savoir la réunification, l'union et l'intersection. Nous allons aussi donner un

[4]Cette notation est classique pour représenter une chaîne de calcul. Par exemple l'expression parenthésée $(1+2)*((3+4)-5)$ peut être représentée par un arbre à 9 noeuds où les feuilles sont les chiffres et les autres noeuds sont les opérateurs ($*$ est la racine). Cette expression en notation polonaise inversée est 1 2 + 3 4 + 5 − $*$. Remarquons que l'ordre d'apparition des chiffres est la même dans les deux expressions.

[5]Pour une meilleure compréhension, nous donnons l'expression parenthésée : $\{1c0\}\, c\, \{0c\, [1c\, ((1c0)c((1c0)c0))]\}$

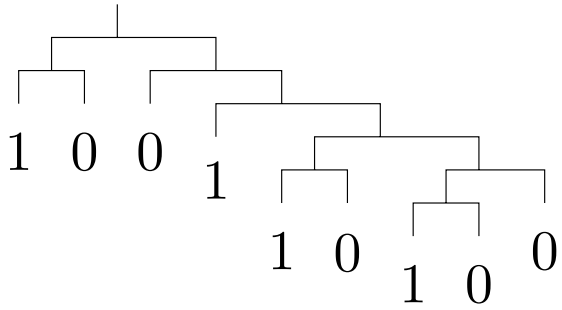

Figure 2.4: Représentation d'un sous-pavage régulier par un arbre binaire.

algorithme capable de tester si un pavé est inclus dans un sous-pavage. Notons que ces opérations sont très rapides grâce à l'utilisation d'arbres binaires. Pour les sous-pavages non réguliers, ces opérations ne peuvent plus être effectuées aussi simplement.

Réunification de deux sous-pavages : Considérons un pavé $[\mathbf{x}]$ et deux sous-pavages réguliers $\widehat{\mathcal{X}} \in \mathcal{SPR}(L[\mathbf{x}])$ et $\widehat{\mathcal{Y}} \in \mathcal{SPR}(R[\mathbf{x}])$. Le sous-pavage *réunifié* $\widehat{\mathcal{Z}} \triangleq (\widehat{\mathcal{X}}|\widehat{\mathcal{Y}})$ élément de $\mathcal{SPR}([\mathbf{x}])$ est défini et calculé comme suit :

$$\begin{array}{l} \text{Si } \widehat{\mathcal{X}} = \{L[\mathbf{x}]\} \text{ et } \widehat{\mathcal{Y}} = \{R[\mathbf{x}]\},\ \widehat{\mathcal{Z}} = [\mathbf{x}]\,; \\ \text{Si } \widehat{\mathcal{X}} = \emptyset \text{ et } \widehat{\mathcal{Y}} = \emptyset, \text{ alors } \widehat{\mathcal{Z}} = \emptyset; \\ \text{Sinon, } L\widehat{\mathcal{Z}} = \widehat{\mathcal{X}} \text{ et } R\widehat{\mathcal{Z}} = \widehat{\mathcal{Y}} \end{array} \tag{2.45}$$

Notons que le nombre de pavés $\#\widehat{\mathcal{Z}}$ du sous-pavage $\widehat{\mathcal{Z}}$ n'est pas forcément égal à $\#\widehat{\mathcal{X}}+\#\widehat{\mathcal{Y}}$. Si par exemple $[\mathbf{x}] = [0,2]^2$, $\widehat{\mathcal{X}} = \{[0,1] \times [0,2]\}$ et $\widehat{\mathcal{Y}} = \{[1,2] \times [0,2]\}$, alors on a $\widehat{\mathcal{X}} = \{L[\mathbf{x}]\}$ et $\widehat{\mathcal{Y}} = \{R[\mathbf{x}]\}$. Et donc $\#\widehat{\mathcal{Z}} = 1$ alors que $\#\widehat{\mathcal{X}} + \#\widehat{\mathcal{Y}} = 2$. Si on travaille avec des arbres binaires, toutes ces opérations sont élémentaires. Par exemple l'instruction $L\widehat{\mathcal{Z}} = \widehat{\mathcal{X}}$ et $R\widehat{\mathcal{Z}} = \widehat{\mathcal{Y}}$ revient à raccrocher les deux arbres $\widehat{\mathcal{X}}$ et $\widehat{\mathcal{Y}}$ à un noeud pour former l'arbre $\widehat{\mathcal{Z}}$. Notons enfin que la réunification peut être vue comme l'opération inverse des opérateurs L et R dans le sens où

$$\widehat{\mathcal{Z}} = (\widehat{\mathcal{X}}|\widehat{\mathcal{Y}}) \Leftrightarrow \widehat{\mathcal{X}} = L\widehat{\mathcal{Z}} \text{ et } \widehat{\mathcal{Y}} = R\widehat{\mathcal{Z}} \tag{2.46}$$

Intersection : L'intersection $\widehat{\mathcal{X}} \cap \widehat{\mathcal{Y}}$ de deux sous-pavages $\widehat{\mathcal{X}} \in \mathcal{SPR}([\mathbf{x}])$ et $\widehat{\mathcal{Y}} \in \mathcal{SPR}([\mathbf{x}])$, est un sous-pavage de $\mathcal{SPR}([\mathbf{x}])$. Il est obtenu par l'algorithme récursif suivant :

$$\textsc{Intersection}\left(\widehat{\mathcal{X}}, \widehat{\mathcal{Y}}, [\mathbf{x}]\right) \tag{2.47}$$

$$\begin{array}{l} \text{Si } \widehat{\mathcal{X}} = \{[\mathbf{x}]\}, \text{ renvoyer } \widehat{\mathcal{Y}}; \\ \text{Si } \widehat{\mathcal{Y}} = \{[\mathbf{x}]\}, \text{ renvoyer } \widehat{\mathcal{X}}; \\ \text{Si } \widehat{\mathcal{X}} = \emptyset \text{ ou si } \widehat{\mathcal{Y}} = \emptyset, \text{ renvoyer } \emptyset; \\ \text{Renvoyer } (\left(\textsc{Intersection}\left(L\widehat{\mathcal{X}}, L\widehat{\mathcal{Y}}, L[\mathbf{x}]\right)\right) | \textsc{Intersection}\left(R\widehat{\mathcal{X}}, R\widehat{\mathcal{Y}}, R[\mathbf{x}]\right)\big)\,; \end{array} \tag{2.48}$$

Union : Si $\widehat{\mathcal{X}}$ et $\widehat{\mathcal{Y}}$ sont deux sous-pavages de $\mathcal{SPR}(\,[\mathbf{x}]\,)$, l'union $\widehat{\mathcal{X}} \cup \widehat{\mathcal{Y}}$ est un sous-pavage régulier de $\mathcal{SPR}(\,[\mathbf{x}]\,)$ qui peut être obtenu par l'algorithme récursif :

$$\textsc{Union}\left(\widehat{\mathcal{X}}, \widehat{\mathcal{Y}}, [\mathbf{x}]\right) \tag{2.49}$$

$$\begin{array}{l} \text{Si } \widehat{\mathcal{X}} = \emptyset \text{ ou si } \widehat{\mathcal{Y}} = \{[\mathbf{x}]\}, \text{ renvoyer } \widehat{\mathcal{Y}}, \\ \text{Si } \widehat{\mathcal{Y}} = \emptyset \text{ ou si } \widehat{\mathcal{X}} = \{[\mathbf{x}]\}, \text{ renvoyer } \widehat{\mathcal{X}}, \\ \text{Renvoyer } (\textsc{Union}\left(L\widehat{\mathcal{X}}, L\widehat{\mathcal{Y}}, L[\mathbf{x}]\right), \textsc{Union}\left(R\widehat{\mathcal{X}}, R\widehat{\mathcal{Y}}, R[\mathbf{x}]\right)). \end{array} \tag{2.50}$$

Tester si un pavé est inclus dans un sous-pavage : Si $[\mathbf{z}]$ est un pavé contenu dans $[\mathbf{x}]$ et $\widehat{\mathcal{X}} \in \mathcal{SPR}([\mathbf{x}])$, l'algorithme récursif qui suit permet de tester si $[\mathbf{z}] \subset \widehat{\mathcal{X}}$ ou si $[\mathbf{z}] \cap \widehat{\mathcal{X}} = \emptyset$. Si l'algorithme renvoie 1, alors, $[\mathbf{z}] \subset \widehat{\mathcal{X}}$. S'il renvoie 0, alors, $[\mathbf{z}] \cap \widehat{\mathcal{X}} = \emptyset$. Sinon, l'algorithme renvoie l'intervalle $[0,1]$ qui signale[6] que $[\mathbf{z}]$ est à cheval sur la frontière de $\widehat{\mathcal{X}}$.

$$\textsc{Inside}\left([\mathbf{z}], \widehat{\mathcal{X}}, [\mathbf{x}]\right)$$

$$\begin{array}{l} \text{Si } [\mathbf{z}] = \emptyset \text{ ou si } \widehat{\mathcal{X}} = \{[\mathbf{x}]\}, \text{ renvoyer 1;} \\ \text{Si } \widehat{\mathcal{X}} = \emptyset, \text{ renvoyer 0;} \\ a = \textsc{Inside}\left([\mathbf{z}] \cap L[\mathbf{x}], L\widehat{\mathcal{X}}, L[\mathbf{x}]\right); \\ \text{Si } a = [0,1], \text{ renvoyer } [0,1]; \\ b = \textsc{Inside}\left([\mathbf{z}] \cap R[\mathbf{x}], R\widehat{\mathcal{X}}, R[\mathbf{x}]\right); \\ \text{Si } a \neq b, \text{ renvoyer } [0,1]; \\ \text{Si } a = b, \text{ renvoyer } a; \end{array}$$

2.7 Algorithmes ensemblistes

Le *calcul ensembliste* est une notion un peu idéale car on ne sait pas vraiment calculer avec les ensembles. Dans nos applications, nous développerons des *algorithmes ensemblistes* qui utiliseront le calcul ensembliste et qui donc, eux aussi, seront idéaux. Voici un exemple d'algorithme ensembliste type.

	AlgEns
Entrées	$\mathbb{A}$, $\mathbb{B}$, sous-ensembles de $\mathbb{R}^n$
Initialisation	$\mathbb{C} = \{\mathbf{0}\}$;
Début	for $i = 1$ to 10, $\mathbb{C} = \mathbb{A} + \mathbb{B} + [\mathbb{C}]$
	$\mathbb{D} = \mathbf{f}(\mathbb{C}) + \mathbf{f}^{-1}(\mathbb{A})$
Sortie	$\mathbb{D}$, sous-ensemble de $\mathbb{R}^n$

[6] L'explication du codage du chevauchement par un intervalle $[0,1]$ résulte de la notion de booléen intervalle. Cette notion sera présentée dans le chapitre suivant et se trouve associée à une logique trivaluée.

où $\{\mathbf{0}\}$ est le singleton contenant le vecteur nul, $[\mathbb{C}]$ est le pavé enveloppe de l'ensemble $\mathbb{C}$ et $\mathbf{f}$ une fonction de $\mathbb{R}^n$ dans $\mathbb{R}^n$. Il est clair que AlgEns n'est pas implémentable en l'état. Toutefois, il peut être approximé par le même algorithme, où les sous-pavages se substituent aux ensembles (de même qu'un algorithme sur les réels est approximé par le même algorithme sur les flottants). Si le calcul sur les sous-pavages est implémenté, ce qui fera l'objet du prochain chapitre, une mise en oeuvre de AlgEns sera :

$$\begin{array}{lll} & \multicolumn{2}{c}{\textsc{AlgExt}} \\ \text{Entrées} & \multicolumn{2}{l}{\widehat{\mathbb{A}},\ \widehat{\mathbb{B}},\ \text{sous-pavages de } \mathbb{R}^n\ ;} \\ \text{Initialisation} & \multicolumn{2}{l}{\widehat{\mathbb{C}} = \{\mathbf{0}\}\ ;} \\ \text{Début} & \multicolumn{2}{l}{\text{for } i = 1 \text{ to } 10,\ \widehat{\mathbb{C}} = \widehat{\mathbb{A}} + \widehat{\mathbb{B}} + [\widehat{\mathbb{C}}]\ ;} \\ & \multicolumn{2}{l}{\widehat{\mathbb{D}} = \mathbf{f}(\widehat{\mathbb{C}}) + \mathbf{f}^{-1}(\widehat{\mathbb{A}})\ ;} \\ \text{Sortie} & \multicolumn{2}{l}{\widehat{\mathbb{D}},\ \text{sous-pavage de } \mathbb{R}^n.} \end{array} \tag{2.51}$$

Cet algorithme sera appelé *algorithme extérieur* car, à tout instant, les ensembles de AlgEns sont contenus dans les sous-pavages correspondant dans AlgExt. Chaque opération sur les sous-pavages de AlgExt doit donc être telle que le sous-pavage résultant contienne le résultat ensembliste associé. Ainsi, $\mathbf{f}(\widehat{\mathbb{C}})$, dans AlgExt, représente un sous-pavage qui contient l'ensemble $\{\mathbf{f}(\mathbf{x}) \,|\mathbf{x} \in \widehat{\mathbb{C}}\}$ qui lui même contient l'ensemble $\mathbf{f}(\mathbb{C})$ de AlgEns. De la même façon, $\widehat{\mathbb{A}} + \widehat{\mathbb{B}}$ de AlgExt est un sous-pavage qui enferme $\{\mathbf{a} + \mathbf{b} | \mathbf{a} \in \widehat{\mathbb{A}}, \mathbf{b} \in \widehat{\mathbb{B}}\}$ qui enferme $\mathbb{A} + \mathbb{B}$ de AlgEns.

Nous dirons que AlgExt est une *mise en oeuvre* de AlgEns. L'expression *mise en oeuvre* peut paraître un peu forte puisqu'il s'agit seulement d'une approximation. Pourtant un algorithme sur les nombres réels est souvent *mis en oeuvre* par le même algorithme mais sur les flottants. Par *mise en oeuvre* d'un algorithme, il faut donc comprendre que les variables de l'algorithme implémenté sont très proches de celles de l'algorithme idéal. Par exemple, si on utilise des ellipsoïdes pour représenter les ensembles de AlgEns, on ne peut pas réellement parler de mise en oeuvre car les ensembles manipulés par AlgEns ne sont pas des ellipsoïdes. Par contre, en représentant les ensembles par des sous-pavages, on peut vraiment parler d'une mise en oeuvre de AlgEns car les sous-pavages de AlgExt sont arbitrairement proches des ensembles de AlgEns.

2.8 Conclusion

Dans ce chapitre, nous avons présenté l'approche ensembliste comme outil de résolution de problèmes non-linéaires et de représentation de l'incertitude. Après quelques brefs rappels sur la théorie des ensembles, nous avons introduit la notion de sous-pavage. Ces sous-pavages nous permettront par la suite d'approximer et de manipuler les ensembles impliqués dans nos problèmes. Nous n'avons pas directement parlé de la mise en oeuvre

du calcul ensembliste, car elle requiert d'une part, l'introduction du calcul par intervalles, qui sera faite au chapitre suivant, et d'autre part, quelques algorithmes spécifiques, dont certains seront décrits au chapitre 4.

Chapitre 3

Analyse par intervalles

3.1 Introduction

L'analyse par intervalles (voir [Moo79]) repose sur une idée très simple qui consiste à représenter les nombres réels ou entiers par des intervalles qui les contiennent. Cette idée a permis d'apporter de la rigueur aux algorithmes numériques classiques. Mais elle a aussi permis le développement d'algorithmes spécifiques aux intervalles, pouvant concurrencer sérieusement l'approche formelle, dans le sens où les résultats obtenus sont garantis. L'analyse par intervalles permet par exemple de trouver toutes les solutions d'un ensemble d'équations non-linéaires, ou d'optimiser de façon garantie un critère non convexe [Han92b], ou même de démontrer des propositions. Sur ces terrains, il est raisonnable d'affirmer que l'analyse par intervalles n'admet, à ce jour, aucun concurrent sérieux.

L'analyse par intervalles apparaît donc de façon inespérée au moment où on commençait à affirmer qu'il n'existerait jamais aucune méthode numérique capable de minimiser de façon garantie une fonction coût non convexe. Et pourtant elle reste confidentielle, principalement parce que l'implémentation des algorithmes qui en découlent est difficile.

On peut légitimement se demander *où est le miracle ? Qu'est ce qui fait que soudainement on arrive à résoudre autant de problèmes réputés impossibles ?* En bref, on peut dire que la vision ensembliste de l'analyse par intervalles permet une analyse par blocs d'un espace de recherche alors que classiquement, nous ne pouvions le parcourir que par un nuage de points. Or comme cet espace est en général constitué d'une infinité de points, il nous était impossible de balayer l'espace tout entier en un temps fini. Une autre question naturelle est " *Qu'est-ce-qui permet à l'analyse par intervalles un travail par blocs que d'autres types d'approches ensemblistes ne permettent pas ?* ". Nous allons maintenant tenter d'apporter une réponse à cette question. Considérons un pavé $[\mathbf{x}]$ de $\mathbb{R}^n$, une fonction $f : \mathbb{R}^n \to \mathbb{R}$ et

un ensemble $\mathbb{S}$ de $\mathbb{R}^n$ défini par un ensemble de contraintes (égalités, inégalités ...) reliées par des opérateurs logiques (et, ou, non ...). L'analyse par intervalles possède trois clefs qui permettent de réaliser les trois opérations suivantes

- *encadrer l'image de* $[\mathbf{x}]$ *par* f, c'est-à-dire trouver un intervalle qui contienne l'ensemble image $\{f(\mathbf{x})|\mathbf{x} \in [\mathbf{x}]\}$. Cette clef est une conséquence directe du calcul par intervalles et utilise la notion de *fonction d'inclusion* qui sera présentée en section 3.3.

- *tester l'appartenance de* $[\mathbf{x}]$ *à un ensemble* $\mathbb{S}$, c'est-à-dire montrer que $[\mathbf{x}] \subset \mathbb{S}$ ou que $[\mathbf{x}] \cap \mathbb{S} = \emptyset$. Pour cela, nous développerons le concept de *test d'inclusion* en section 3.4.

- *réduire* $[\mathbf{x}]$ *par rapport à* $\mathbb{S}$, c'est-à-dire trouver un pavé $[\mathbf{q}]$ contenu dans $[\mathbf{x}]$ tel que $[\mathbf{x}] \cap \mathbb{S} = [\mathbf{q}] \cap \mathbb{S}$. La réduction est principalement fondée sur les trois approches suivantes : (i) la propagation de contraintes sur les intervalles présentée en section 3.6, (ii) la méthode de Newton par intervalles brièvement rappelée en section 3.7 et (iii) la linéarisation extérieure que nous introduirons en section 3.7. Les opérateurs $\mathcal{N}_{\mathbb{S}}$ qui permettent cette opération seront appelés *opérateur de réduction* (en anglais *narrowing operators*). Ainsi, on notera $[\mathbf{q}] = \mathcal{N}_{\mathbb{S}}([\mathbf{x}])$ si $[\mathbf{q}] \subset [\mathbf{x}]$ et si $[\mathbf{x}] \cap \mathbb{S} = [\mathbf{q}] \cap \mathbb{S}$.

Comme nous le verrons, les deux dernières clefs sont des conséquences directes de la première. Ces trois clefs forment la base de tous les algorithmes auxquels nous allons nous intéresser. Toute autre théorie mathématique qui permettrait d'implémenter ces trois opérations pourrait se substituer à l'analyse par intervalles, mais il n'en existe pas à ma connaissance. L'analyse par intervalles permet la réalisation de ces trois opérations

- pour une classe énorme de fonctions f,

- pour une classe énorme de contraintes définissant $\mathbb{S}$,

- de façon garantie par rapport aux erreurs d'arrondi de la machine.

L'analyse par intervalles a été développée dans les années 70, (voir Moore [Moo79]) pour quantifier l'erreur sur le résultat d'un calcul numérique due à la précision finie du calcul flottant de l'ordinateur. De nos jours, elle est aussi très utilisée pour la résolution d'équations ou d'inéquations non-linéaires ainsi que pour la minimisation d'une fonction coût non convexe. Pour cela, elle est généralement appelée par des algorithmes de type *branch-and-bound*, qui découpent l'espace (*branch*) tout en affinant les domaines de certaines variables (*bound*).

3.2 Calcul par intervalles

Etendre aux intervalles les opérations arithmétiques de base $\{+,-,.,/\}$ est une idée naturelle pour évaluer l'ensemble que décrit la somme, le produit, *etc.* de deux réels incertains inclus dans des intervalles. On définit ainsi pour $\circ \in \{+,-,.,/\}$,

$$[x] \circ [y] = \{x \circ y | x \in [x] \text{ et } y \in [y]\}. \tag{3.1}$$

La caractérisation de $[x] \circ [y]$ se fait à l'aide du formulaire

$$\begin{cases} [x] + [y] = \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right], \\ [x] - [y] = \left[\underline{x} - \overline{y}, \overline{x} - \underline{y}\right], \\ [x].[y] = \left[\min\left(\underline{x}.\underline{y}, \overline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\overline{y}\right), \max\left(\underline{x}.\underline{y}, \overline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\overline{y}\right)\right], \\ [x]/[y] = [x] \times \left[1/\overline{y}, 1/\underline{y}\right], \text{ si } 0 \notin [y] \text{ et indéfini sinon.} \end{cases} \tag{3.2}$$

Les propriétés algébriques des opérations sur les intervalles ont été largement étudiées. Parmi les plus souvent rappelées, citons la *sous-distributivité* de la multiplication par rapport à l'addition

$$[a].([b] + [c]) \subseteq [a].[b] + [a].[c]. \tag{3.3}$$

De même, les fonctions élémentaires comme exp, tan, sin, cos ... peuvent être étendues aux intervalles. Si $f : \mathbb{R} \to \mathbb{R}$ est une fonction élementaire, on définit son *extension intervalle* comme suit :

$$[f] : \begin{array}{ccc} \mathbb{IR} & \to & \mathbb{IR} \\ [x] & \to & [f]([x]) = [\{f(x) | x \in [x]\}]. \end{array} \tag{3.4}$$

où $[\mathbb{A}]$ désigne l'*intervalle enveloppe* de l'ensemble $\mathbb{A} \subset \mathbb{R}^n$, c'est-à-dire le plus petit intervalle qui contient $\mathbb{A}$ et, rappelons le, $\mathbb{IR}$ désigne l'ensemble de tous les intervalles. Ainsi, en prenant en compte la monotonie de l'exponentielle, on a $[\exp]([x]) = [\exp(\underline{x}), \exp(\overline{x})] = \exp([x])$. Dans le cas de l'exponentielle, comme c'est le cas pour toutes les fonctions élémentaires continues, l'extension intervalle $[f]([x])$ est égale à l'image par f de $[x]$, ce qui se traduit par $[f]([x]) = f([x])$. Pour les fonctions non monotones comme le sinus, il faut développer un algorithme spécifique. Pour bien assimiler le calcul par intervalles, je vous conseille d'aller jouer avec la calculatrice par intervalles sur le site

http: //www.mscs.mu.edu/~globsol/JavaIntervalCalc/I_ C.html

3.3 Fonction d'inclusion

Soit $\mathbf{f}$ une fonction de $\mathbb{R}^n$ dans $\mathbb{R}^m$. Une fonction ensembliste $[\mathbf{f}]$ définie de $\mathbb{IR}^n$ dans $\mathbb{IR}^m$, est une *fonction d'inclusion* de $\mathbf{f}$ si

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \quad \mathbf{f}([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}]). \tag{3.5}$$

Notons que quelque soit la fonction $\mathbf{f}$, nous savons trouver une fonction d'inclusion. Par exemple la fonction $[\mathbf{f}]([\mathbf{x}]) = \mathbb{R}^n$, pour tout $[\mathbf{x}]$ de $\mathbb{IR}^n$, est une fonction d'inclusion pour toute fonction $\mathbf{f}$ de $\mathbb{R}^n$ dans $\mathbb{R}^m$. L'analyse par intervalles s'évertue à proposer un ensemble d'outils permettant de trouver des fonctions d'inclusion *efficaces* (c'est-à-dire précises et rapide à évaluer) pour tout type de fonctions $\mathbf{f}$ (polynomiales [MMTG92], possédant expression analytique [Moo79], définie par un algorithme [JWLM99], solution d'une équation différentielle [Loh87], [Kuh98], ...). Une fonction d'inclusion $[\mathbf{f}]$ pour $\mathbf{f}$ est *convergente*, si pour toute suite $[\mathbf{x}](k)$,

$$\lim_{k\to\infty} w([\mathbf{x}](k)) = 0 \Rightarrow \lim_{k\to\infty} w([\mathbf{f}]([\mathbf{x}](k)) = 0. \tag{3.6}$$

Rappelons que $w([\mathbf{x}](k))$ dénote la longueur de $[\mathbf{x}](k)$. Cette propriété est demandée pour la convergence de la plupart des algorithmes utilisant l'analyse par intervalles.

Soient $[f_j] : \mathbb{IR}^n \to \mathbb{IR}$, m fonctions d'inclusion associées aux fonctions coordonnées f_j d'une fonction vectorielle $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$. Une fonction d'inclusion pour $\mathbf{f}$ sera donnée par

$$[\mathbf{f}]([\mathbf{x}]) = [f_1]([\mathbf{x}]) \times \ldots \times [f_m]([\mathbf{x}]). \tag{3.7}$$

L'élaboration de fonctions d'inclusion efficaces d'une fonction vectorielle $\mathbf{f}$ se ramène donc à l'élaboration de fonctions d'inclusion efficaces pour ses fonctions coordonnées. Focalisons nous donc sur le calcul de fonctions d'inclusion de fonctions à valeurs dans $\mathbb{R}$ et plus particulièrement sur les *fonctions termes,* qui représentent une vaste classe de fonctions non-linéaires. Nous renvoyons à la littérature [MMTG92], [Moo66], [Loh87], [JWLM99] pour le traitement d'autres classes de fonctions. Brièvement, une fonction terme (voir [Gra98] page 10 pour une définition précise) est une fonction qui peut être construite par compositions des fonctions élémentaires usuelles, sin, cos, exp, sqrt, sqr, abs, max, ... et des opérateurs $+, -, *, /$. Un exemple de fonction terme est donnée par $f(x_1, x_2, x_3) = \sqrt{x_1 + \exp(x_2)} * \log(x_3^3 + \sin(x_1))$. Il existe plusieurs méthodes pour obtenir une fonction d'inclusion d'une fonction terme f. La méthode la plus simple consiste à remplacer les occurrences des variables scalaires $(x, y, \ldots)$ dans l'expression de f par les variables intervalles correspondantes $([x], [y], \ldots)$, et toutes les fonctions élémentaires (exp, sin, ...) par leur extension intervalle. Nous obtenons ainsi la *fonction d'inclusion naturelle* de f. Cependant, bien souvent, cette fonction d'inclusion n'est pas minimale au sens de l'inclusion. Elle ne l'est que lorsque chacune des variables n'intervient qu'une seule fois dans l'expression de f. En outre, la longueur des intervalles obtenus dépend assez fortement de l'expression initiale de la fonction comme l'illustre l'exemple suivant tiré de la thèse de M. Kieffer [Kie99].

Exemple 3.3.1 *Comparons les performances des fonctions d'inclusion établies à partir de quatre formulations d'une même fonction terme :*

$$f_1(x) = x(x+1), \tag{3.8}$$

$$f_2(x) = x.x + x, \tag{3.9}$$
$$f_3(x) = x^2 + x, \tag{3.10}$$
$$f_4(x) = (x + \tfrac{1}{2})^2 - \tfrac{1}{4}. \tag{3.11}$$

Pour $[x] = [-1, 1]$*, on obtient*

$$[f_1]([x]) = [x]([x] + 1) = [-2, 2], \tag{3.12}$$
$$[f_2]([x]) = [x].[x] + [x] = [-2, 2], \tag{3.13}$$
$$[f_3]([x]) = [x]^2 + [x] = [-1, 2], \tag{3.14}$$
$$[f_4]([x]) = \left([x] + \tfrac{1}{2}\right)^2 - \tfrac{1}{4} = \left[-\tfrac{1}{4}, 2\right]. \tag{3.15}$$

Selon l'expression de la fonction, l'encadrement obtenu sera donc plus ou moins grossier (voir la figure 3.1). Les deux expressions $[x].[x]$ *et* $[x]^2$ *ne sont pas équivalentes, car dans le premier cas, chacune des occurrences de x peut varier indépendamment. Précisons que l'ensemble image de la fonction initiale est* $\left[-\frac{1}{4}, 2\right]$*.* $[f_4]$ *est donc minimale.* ♢

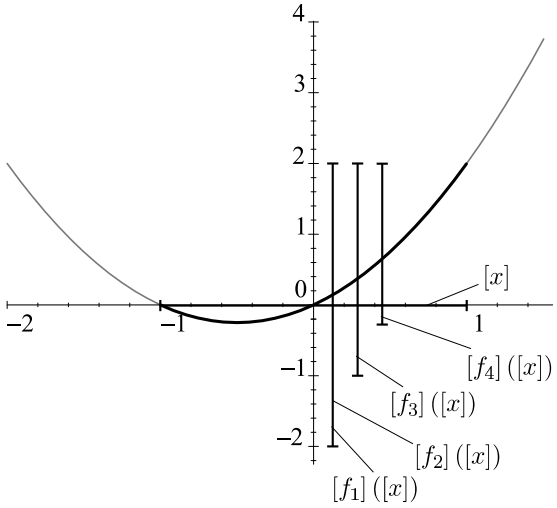

Figure 3.1: Comparaison de fonctions d'inclusion d'une même fonction

Une fonction d'inclusion qui fournit un intervalle image qui n'est pas égal à l'évaluation intervalle de la fonction réelle sur l'intervalle considéré est dite *pessimiste*. Il n'existe pas de méthode systématique permettant, pour une fonction terme donnée, d'obtenir une fonction d'inclusion minimale. Cependant, on peut constater empiriquement que plus une variable apparaît fréquemment dans l'expression de f, plus le pessimisme a des chances de devenir important. Cela peut s'interpréter par le fait que deux occurrences d'une même variable sont considérées comme variant indépendamment l'une de l'autre. Ainsi, les incertitudes se combinent, pour donner un résultat plus mauvais que si l'on avait considéré les occurrences comme variant de façon identique.

Il existe d'autres types de fonctions d'inclusion, faisant intervenir des développements en série de la fonction initiale. Soit $f : \mathbb{R}^n \rightarrow \mathbb{R}$, une fonction dérivable sur un pavé $[\mathbf{x}]$, et $\mathbf{m}$ le centre de $[\mathbf{x}]$. Le théorème de la valeur moyenne donne

$$\forall \mathbf{x} \in [\mathbf{x}], \exists \mathbf{z} \in [\mathbf{x}] \mid f(\mathbf{x}) = f(\mathbf{m}) + f'(\mathbf{z})(\mathbf{x} - \mathbf{m}). \tag{3.16}$$

où $f'(\mathbf{z})$ est une matrice ligne $1 \times n$ correspondant à la dérivée de f en $\mathbf{z}$. Par conséquent

$$f(\mathbf{x}) \in f(\mathbf{m}) + f'([\mathbf{x}])(\mathbf{x} - \mathbf{m}), \tag{3.17}$$

d'où

$$f([\mathbf{x}]) \subseteq f(\mathbf{m}) + [f']([\mathbf{x}])([\mathbf{x}] - \mathbf{m}). \tag{3.18}$$

La fonction intervalle définie par

$$[f_c]([\mathbf{x}]) \triangleq f(\mathbf{m}) + [f']([\mathbf{x}])([\mathbf{x}] - \mathbf{m}), \tag{3.19}$$

où $[f']$ une fonction d'inclusion de la dérivée f' de f, est donc une fonction d'inclusion de f. Cette fonction est appelée *forme centrée standard* de f sur l'intervalle $[x]$. Elle donne des résultats généralement moins pessimistes que la fonction d'inclusion naturelle lorsque $[x]$ est de faible longueur et que le problème de dépendance se pose pour f.

La notion de fonction d'inclusion est la clef de voûte de tous nos algorithmes. On peut raisonnablement dire que le calcul par intervalles ne nous sert qu'à calculer des fonctions d'inclusion. Si on disposait d'un outil plus puissant pour cette tâche, nous pourrions entièrement nous en dispenser.

3.4 Test d'inclusion

Un *booléen* est un élément de $\mathbb{B} \triangleq \{0, 1\}$ où 0 signifie *faux* et 1 signifie *vrai*. Par extension[1], un *intervalle booléen* est un élément de l'ensemble $\mathbb{IB} \triangleq \{[0,0], [1,1], [0,1]\}$. Par abus de notations et par similarité avec les notations classiques sur les intervalles, $[0,0]$, $[1,1]$ et $[0,1]$ seront dénotés $0, 1$ et $[0,1]$, respectivement. Les opérations sur les intervalles booléens se définissent comme pour les intervalles réels, à savoir

$$[a] + [b] = \{a + b | a \in [a], b \in [b]\}, \tag{3.20}$$

$$[a].[b] = \{a.b | a \in [a], b \in [b]\}, \tag{3.21}$$

$$\rceil [a] = \{\rceil a | a \in [a]\}, \tag{3.22}$$

[1]Pour un ensemble muni d'une relation d'ordre partielle $(\mathbb{E}, \leq)$, on peut toujours définir l'ensemble $\mathbb{IE}$, des couples $[a, b]$ tels que $a \in \mathbb{E},\ b \in \mathbb{E}$ et $a \leq b$. Les éléments de $\mathbb{E}$ seront qualifiés d'intervalles. $\mathbb{E}$ peut représenter l'ensemble $\mathbb{R}^n$ ou l'ensemble des booléens $\mathbb{B}$, ou encore l'ensemble des compacts muni de la relation d'ordre partielle $\subset$.

où $\rceil$ est l'opérateur *non,* c'est-à-dire, $\rceil 0 = 1$ et $\rceil 1 = 0$. Par exemple,

$$([0,1]+1).([0,1].1) = 1.[0,1] = [0,1]. \tag{3.23}$$

Si $[a]$ est un intervalle booléen, nous avons

$$0.[a] = 0;\ 1.[a] = [a];\ 0+[a] = [a];\ 1+[a] = 1;\ [a].[a] = [a]+[a] = [a]. \tag{3.24}$$

Soit $\beta : \mathbb{B}^n \to \mathbb{B}$ une fonction. Une telle fonction sera appelée *expression booléenne*. Les notions d'*extension naturelle* et de *fonction d'inclusion*, désormais classiques pour les intervalles se généralisent immédiatement pour les expressions booléennes. Ainsi, l'*extension naturelle* $[\beta_N]()$ de $\beta()$ est obtenue en remplaçant tous les arguments de β et tous les opérateurs la constituant par leur équivalent intervalle. $[\beta] : \mathbb{IB}^n \to \mathbb{IB}$ est une *fonction d'inclusion* pour β si

$$\forall [b_1] \in \mathbb{IB}, \ldots, \forall [b_n] \in \mathbb{IB}, \beta([b_1], \ldots, [b_n]) \subset [\beta]([b_1], \ldots, [b_n]). \tag{3.25}$$

$[\beta]([b_1], \ldots, [b_n])$ est *minimale* si l'inclusion peut être remplacée par une égalité. Notons que le problème de dépendance existe toujours lorsque l'opérateur de complémentation $\rceil$ est utilisé. Par exemple, on aura

$$[a] \subset [a].[b] + [a].\rceil[b] \tag{3.26}$$

(l'égalité n'est pas vérifiée pour $b = [0,1]$) alors que $a = ab + a\rceil b$. Tout comme cela était le cas lorsque nous travaillions avec les intervalles réels, lorsque l'expression manipulée, appelons-la $\beta(b_1, \ldots, b_n)$, est croissante en ses variables, la fonction d'inclusion minimale est donnée[2] par

$$[\beta^*]([b_1], \ldots, [b_n]) = \left[\beta\left(b_1^-, \ldots, b_n^-\right), \beta\left(b_1^+, \ldots, b_n^+\right)\right]. \tag{3.27}$$

Remarquons que cette situation arrive fréquemment en pratique. C'est le cas par exemple lorsque l'opérateur de complémentation n'est pas utilisé, c'est-à-dire lorsque β est un polynôme. Il existe aussi de nombreuses expressions booléennes qui ne sont pas monotones. C'est le cas du *ou-exclusif*

$$\beta(b_1, b_2) = b_1\rceil b_2 + \rceil b_1 b_2 \tag{3.28}$$

Dans le cas général, il est possible d'obtenir la fonction d'inclusion minimale lors de l'évaluation car les variables booléennes ne peuvent prendre qu'un nombre de valeurs discrètes. Par exemple, cherchons à encadrer $\beta([b_1], [b_2])$ avec $\beta(b_1, b_2) = b_1.b_2 + b_1.\rceil b_2$ pour $[b_1] = 1$ et $[b_2] = [0,1]$. L'extension naturelle nous donne $[\beta](1, [0,1]) = 1.[0,1] + 1.\rceil[0,1] = [0,1]$. L'évaluation minimale est obtenue en écrivant $\beta(1, [0,1]) = \beta(1,0) \cup \beta(1,1) = 1$. Cette façon de procéder nous conduit bien sûr à une explosion combinatoire, lorsque

[2] Cette formule peut s'adapter bien évidemment au cas ou β est monotone.

le nombre de variables augmente, que l'on peut éviter en manipulant les expressions booléennes à l'aide des tableaux de Karnaugh ou en utilisant des règles de simplification dans le but de réduire les occurrences. A ma connaissance, le problème de l'évaluation minimale d'une expression booléenne n'a jamais été étudié.

Un *test* est une fonction booléenne $t : \mathbb{R}^n \rightarrow \{0,1\}$. Un *test d'inclusion* pour t est une fonction $[t] : \mathbb{IR}^n \rightarrow \mathbb{IB}$ telle que pour tout $[\mathbf{x}] \in \mathbb{IR}^n$,

$$\begin{array}{rcl} [t]([\mathbf{x}]) = 1 & \Rightarrow & \forall \mathbf{x} \in [\mathbf{x}]\,,\ t(\mathbf{x}) = 1, \\ [t]([\mathbf{x}]) = 0 & \Rightarrow & \forall \mathbf{x} \in [\mathbf{x}]\,,\ t(\mathbf{x}) = 0. \end{array} \tag{3.29}$$

Un test d'inclusion $[t]$ est *fin* si pour tout $\mathbf{x}$, $[t](\mathbf{x}) = t(\mathbf{x})$. $[t]$ est *minimal* si

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, [t]([\mathbf{x}]) = \{t(\mathbf{x}), \mathbf{x} \in [\mathbf{x}]\}. \tag{3.30}$$

Notons qu'un test minimal est forcément fin.

Exemple 3.4.1 *Considérons le test*

$$t : \begin{array}{ccc} \mathbb{R}^2 & \rightarrow & \{0,1\} \\ (x_1, x_2)^T & \rightarrow & (x_1 = 5), \end{array} \tag{3.31}$$

c'est-à-dire, $t(\mathbf{x}) = 1$ *si et seulement si* $x_1 = 5$. *Le test d'inclusion minimal* $[t]$ *est donné par*

$$[t]([\mathbf{x}]) = \begin{cases} 1 & si \quad [x_1] = 5 \\ 0 & si \quad 5 \notin [x_1] \\ [0,1] & sinon \end{cases} \tag{3.32}$$

♢

Grâce à l'analyse par intervalles et à la notion de fonction d'inclusion, on peut facilement construire un test d'inclusion pour les tests du type

$$t(\mathbf{x}) = \beta(t_1(\mathbf{x}), \ldots, t_n(\mathbf{x})) \text{ avec} \tag{3.33}$$

$$t_i(\mathbf{x}) \Leftrightarrow (f_i(\mathbf{x}) \geq 0)\,, i = 1, \ldots, n \tag{3.34}$$

où $\beta : \mathbb{B}^n \rightarrow \mathbb{B}$ est une expression booléenne quelconque. Il est donné par

$$[t]([\mathbf{x}]) = [\beta]([t_1]([\mathbf{x}]), \ldots, [t_n]([\mathbf{x}])) \text{ avec} \tag{3.35}$$

$$[t_i]([\mathbf{x}]) \Leftrightarrow ([f_i]([\mathbf{x}]) \geq 0)\,, i = 1, \ldots, n \tag{3.36}$$

où $[\beta]$ est une fonction d'inclusion pour β. Notons enfin que même si les tests $[t_i]([\mathbf{x}])$ sont tous minimaux et si $[\beta]$ est minimale, le problème de dépendance demeure et du pessimisme peut donc être introduit. Par exemple le test $t(x) \triangleq (x \leq 7)$ ou $(x \geq 6)$ admet

pour test d'inclusion $[t]([x]) \triangleq ([x] \leq 7)$ ou $([x] \geq 6)$. Notons que $[t]([x])$ est composé de deux tests d'inclusion minimaux et d'une expression booléenne croissante (puisque polynomiale) donnée par $\beta(b_1, b_2) = b_1 + b_2$. Pourtant, pour $[x] = [5, 8]$, $[t]([x]) = [0, 1]$ alors que $t([x]) = 1$.

Notation : Un test d'inclusion associé au test $t(x) = (x \leq a)$, sera noté $[t]([x]) = ([x] \ [\leq] \ a)$. Ainsi, on dira que le prédicat $([1, 2] \ [\leq] \ 5)$ a une valeur de vérité égale à 1 alors que $([1, 7] \ [\leq] \ 5)$ a une valeur de vérité égale à $[0, 1]$. De même, un test d'inclusion associé à $t(\mathbf{x}) = (\mathbf{x} \in \mathbb{A})$, où $\mathbb{A}$ est un ensemble, sera noté $[t]([\mathbf{x}]) = ([\mathbf{x}] \ [\in] \ \mathbb{A})$. Ainsi,

$$\begin{aligned} ([\mathbf{x}] \ [\in] \ \mathbb{A}) = 1 \ &\Rightarrow \ [\mathbf{x}] \subset \mathbb{A}, \\ ([\mathbf{x}] \ [\in] \ \mathbb{A}) = 0 \ &\Rightarrow \ [\mathbf{x}] \cap \mathbb{A} = \emptyset, \\ ([\mathbf{x}] \ [\in] \ \mathbb{A}) = [0, 1] \ &\Leftarrow \ \exists \mathbf{x}_1 \in [\mathbf{x}], \exists \mathbf{x}_2 \in [\mathbf{x}], \ \mathbf{x}_1 \notin \mathbb{A}, \mathbf{x}_2 \in \mathbb{A}. \end{aligned} \tag{3.37}$$

$[\in]$ et $[\leq]$ sont appelés *extensions trivaluées* de $\in$ et $\leq$. ♢

3.5 Réduction d'un pavé sous une contrainte

Dans ce paragraphe, on considère une seule contrainte de la forme $f(x_1, \ldots, x_n) \in [y]$, où $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$, $[y]$ est un intervalle et f est une fonction continue de $\mathbb{R}^n$ dans $\mathbb{R}$. Cette contrainte est souvent assimilée à un sous-ensemble $\mathbb{C}_1$ de $\mathbb{R}^n$ (voir [BO97]) avec

$$\mathbb{C}_1 = \{\mathbf{x} \in \mathbb{R}^n | f(\mathbf{x}) \in [y]\} = f^{-1}([y]), \tag{3.38}$$

La *réduction d'un pavé* $[\mathbf{x}]$ *sous la contrainte* $\mathbb{C}_1$ consiste à trouver un pavé $[\mathbf{q}] \subset [\mathbf{x}]$, si possible petit et sans faire de bissections, qui contienne de façon garantie l'ensemble

$$\mathbb{S}_1 \triangleq \mathbb{C}_1 \cap [\mathbf{x}]. \tag{3.39}$$

On aura ainsi la chaîne d'inclusion $(\mathbb{C}_1 \cap [\mathbf{x}]) \subset [\mathbf{q}] \subset [\mathbf{x}]$. Et donc, en intersectant par $\mathbb{C}_1$ chaque terme de la chaîne, $(\mathbb{C}_1 \cap [\mathbf{x}]) \subset (\mathbb{C}_1 \cap [\mathbf{q}]) \subset (\mathbb{C}_1 \cap [\mathbf{x}])$. Ceci implique que,

$$[\mathbf{x}] \cap \mathbb{C}_1 = [\mathbf{q}] \cap \mathbb{C}_1 \tag{3.40}$$

La figure 3.2 illustre le principe de réduction d'un pavé $[\mathbf{x}]$ sous une seule contrainte $\mathbb{C}_1$. Lorsque le pavé réduit $[\mathbf{q}]$ est égal au pavé enveloppe $[\mathbb{S}_1]$ on parlera de *réduction minimale*.

Nous allons tout d'abord donner le principe de la réduction d'un pavé sous une contrainte du type $f(x_1, \ldots, x_n) \in [y]$ dans le cas particulier où f est une fonction terme pour laquelle on sait isoler chaque x_i dans l'expression $f(x_1, \ldots, x_n) = y$. Ce cas est relativement fréquent dans les applications. La procédure d'isolement des variables x_i demande un traitement formel qui peut être rendu automatique et plus général par l'utilisation d'arbres

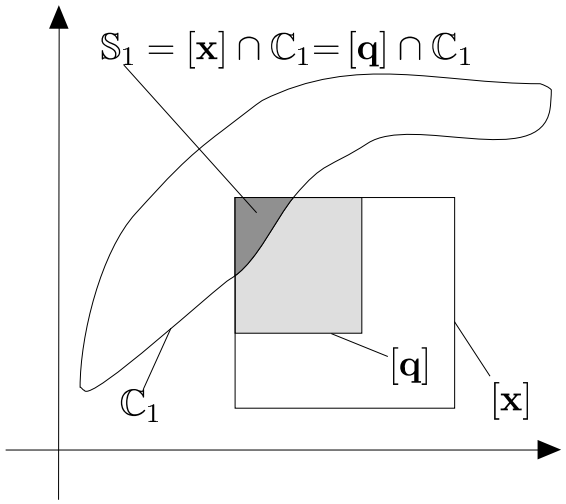

Figure 3.2: Réduction d'un pavé $[\mathbf{x}]$ en un pavé $[\mathbf{q}]$ relativement à une seule contrainte $\mathbb{C}_1$

comme nous le verrons au paragraphe 3.5.2. L'algorithme présenté au paragraphe 3.5.4 utilise ces arbres pour effectuer une réduction minimale des pavés sous une contrainte du type $f(x_1, \ldots, x_n) \in [y]$, lorsque les x_i n'apparaissent qu'une seule fois dans l'expression de f.

3.5.1 Principe de la réduction

Une méthode possible pour effectuer cette réduction utilise sur le théorème suivant :

Théorème 3.5.1 *Supposons qu'il soit possible d'isoler x_i dans l'expression $f(\mathbf{x}) = y$, c'est-à-dire qu'il existe une fonction g_i qui satisfasse*

$$f(\mathbf{x}) = y \Leftrightarrow x_i = g_i({}^i\mathbf{x}, y), \tag{3.41}$$

où ${}^i\mathbf{x} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)^T$. La fonction g_i est appelée fonction solution associée à x_i. *Soit $proj_i$ l'opérateur de projection suivant la $i^{ème}$ direction et soit $[g_i]$ une fonction d'inclusion pour g_i. Nous avons*

$$proj_i(\mathbb{S}_1) \subset [g_i]\left({}^i[\mathbf{x}], [y]\right) \cap [x_i] \tag{3.42}$$

où ${}^i[\mathbf{x}] = \left([x_1], \ldots, [x_{i-1}], [x_{i+1}], \ldots, [x_n]\right)^T$. De plus, si g_i est continue et si $[g_i]$ est minimale, l'inclusion (3.42) devient une égalité. ♢

Preuve: Du fait des équivalences suivantes

$$\begin{array}{rcl} f(x_1, \ldots, x_n) \in [y] & \Leftrightarrow & (\exists y \in [y] \,|\, f(x_1, \ldots, x_n) = y) \\ & \overset{(3.41)}{\Leftrightarrow} & (\exists y \in [y] \,|\, x_i = g_i({}^i\mathbf{x}, y)), \end{array} \tag{3.43}$$

nous avons

$$\begin{array}{rcl} \mathrm{proj}_i(\mathbb{S}_1) & = & \{x_i \in [x_i] | \forall k \neq i, \exists x_k \in [x_k], f(x_1, \ldots, x_n) \in [y]\} \\ & \overset{(3.43)}{=} & \{x_i \in [x_i] | \forall k \neq i, \exists x_k \in [x_k], \exists y \in [y] \text{ avec } x_i = g_i({}^i\mathbf{x}, y)\} \\ & = & \{x_i \in [x_i] | x_i \in g_i({}^i[\mathbf{x}], [y])\}. \end{array} \tag{3.44}$$

L'ensemble projeté $\mathrm{proj}_i(\mathbb{S}_1)$ est donc inclus dans l'intervalle

$$[\mathrm{proj}_i](\mathbb{S}_1) = \left\{x_i \in [x_i] | x_i \in [g_i]({}^i[\mathbf{x}], [y])\right\} = [g_i]\left({}^i[\mathbf{x}], [y]\right) \cap [x_i] \tag{3.45}$$

avec $\mathrm{proj}_i(\mathbb{S}_1) = [\mathrm{proj}_i](\mathbb{S}_1)$ si $[g_i]$ est minimale et si g_i est continue. ♢

Exemple 3.5.1 *Considérons la contrainte* $x_1 x_2 \in [8, 40]$ *et le pavé* $[\mathbf{x}] = [1, 4] \times [1, 4]$. *Cherchons le plus petit pavé contenant l'ensemble* $\mathbb{S}_1$ *défini par (3.39) en réduisant* $[\mathbf{x}]$ *sous cette contrainte. Puisque* $x_1 x_2 = y \Leftrightarrow x_1 = \frac{y}{x_2} \Leftrightarrow x_2 = \frac{y}{x_1}$ *(pour* $x_i \neq 0$*), les fonctions solutions sont* $g_1(x_2, y) = \frac{y}{x_2}$ *et* $g_2(x_1, y) = \frac{y}{x_1}$. *D'après le théorème 3.5.1 et du fait que les fonctions d'inclusion naturelles pour* g_1 *et* g_2 *sont minimales, les projections canoniques de* $\mathbb{S}_1$ *sont*

$$\begin{array}{l} proj_1(\mathbb{S}_1) = [g_1]([x_2], [y]) \cap [x_1] = ([y] / [x_2]) \cap [x_1] = \frac{[8,40]}{[1,4]} \cap [1, 4] = [2, 4], \\ proj_2(\mathbb{S}_1) = [g_2]([x_1], [y]) \cap [x_2] = ([y] / [x_1]) \cap [x_2] = [2, 4], \end{array} \tag{3.46}$$

et donc, le pavé enveloppe de $\mathbb{S}_1$ *est* $[\mathbb{S}_1] = [2, 4] \times [2, 4]$. ♢

3.5.2 Arbre d'une fonction terme acyclique réelle

Les manipulations requises par la technique de réduction, évoquée dans le paragraphe précédent, mélangent calcul numérique et calcul formel. Or, une telle manipulation mixte est difficile à mettre en oeuvre, que ce soit avec des langages formels comme Maple, ou procéduraux comme C++. Un moyen simple de contourner ce problème est d'utiliser des graphes pour représenter les fonctions et les relations. Grâce aux pointeurs, il sera alors relativement aisé d'implémenter les algorithmes sur les graphes en C++. C'est cette approche que nous allons désormais adopter et ce paragraphe présente les notions de base dont nous aurons besoin par la suite pour représenter des fonctions par des graphes.

Une fonction terme $f : \mathbb{R}^n \to \mathbb{R}; \mathbf{x} = (x_1, \ldots, x_n) \to f(\mathbf{x})$ est *acyclique* si elle peut être représentée par un arbre[3] pour lequel

- chaque feuille est soit une composante x_i de $\mathbf{x}$, soit un réel quelconque,

[3] On rappelle qu'un arbre est un graphe connexe sans cycle. Les arbres que nous considérerons sont des arbres enracinés (en anglais : *rooted trees*), c'est-à-dire, qu'un de ses noeuds est désigné pour être la racine.

- chaque noeud z qui n'est pas une feuille est associé soit à un opérateur binaire $\oplus_z \in \{*, +\}$, soit à une fonction élémentaire $f_z \in \{\sin, \cos, \exp, \text{sqr}, \text{sqrt}, \text{inv}, \ldots\}$, avec $\text{sqr}(z) = z^2$, $\text{sqrt}(z) = \sqrt{z}$ et $\text{inv}(z) = 1/z$,
- chaque variable $x_1, \ldots, x_n$ doit apparaître au plus une fois dans l'arbre.

Par exemple, la fonction $f(\mathbf{x}) = x_1 \exp(x_2) + 5 * \sin(x_3)$ est *acyclique*. L'arbre correspondant est représenté sur la figure 3.3. Lorsque qu'une variable apparaît plus d'une fois, un cycle apparaît dans le graphe comme illustré par la figure 3.4 où le cycle est donné par $(z_2, z_5, z_7, z_9, z_8, z_6, z_2)$. Le graphe n'est donc plus un arbre. Notons les opérateurs non-symétriques $-$ et $/$ ne sont pas autorisés et que les expressions du type $x_1/\exp(x_2 - x_3)$ doivent être reformulées par une expression équivalente du type $x_1 * \text{inv}(\exp(x_2 + (-1) * x_3))$. La notion de fonction acyclique est aussi connue sous le nom barbare de *totally decomposable tree structure decomposition function* (TDTSD) dans la littérature internationale (voir [ABK+93], [Bar88], [BAH89]).

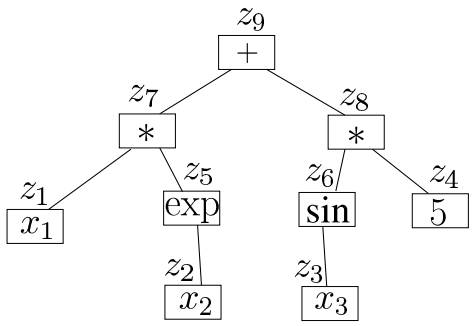

Figure 3.3: Représentation par un arbre de la fonction $f(\mathbf{x}) = x_1 \exp(x_2) + 5 \sin(x_3)$

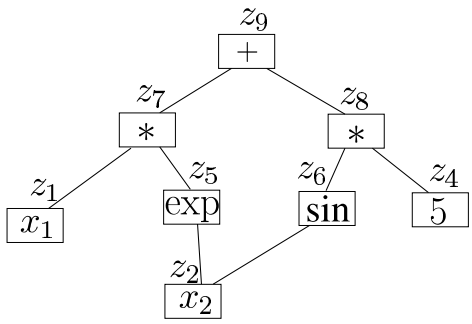

Figure 3.4: La fonction $f(\mathbf{x}) = x_1 \exp(x_2) + 5 \sin(x_2)$ est acyclique.

Une contrainte du type $f(\mathbf{x}) \in [y]$ avec f acyclique sera appelée *contrainte acyclique*. Nous allons nous intéresser particulièrement à ce type de contrainte. Cela se fera sans réelle perte de généralité car les contraintes que nous rencontrerons peuvent généralement

se décomposer en plusieurs contraintes acycliques. Il nous faudra pour cela introduire des variables intermédiaires. Par exemple la contrainte $x_1 - \exp(x_1 x_2) \in [1,2]$ peut se décomposer deux contraintes acycliques données par

$$\begin{cases} x_1 - a_1 & \in & [0,0] \\ a_1 - \exp(x_1 x_2) & \in & [1,2] \end{cases} \tag{3.47}$$

3.5.3 Opérations élémentaires sur les arbres des fonctions acycliques

Considérons un arbre associé à une fonction acyclique $f(\mathbf{x})$. Chaque noeud z de l'arbre est associé à un domaine $\mathbb{D}(z)$, c'est-à-dire, un sous-ensemble de $\mathbb{R}$ et à une variable qui appartient à ce domaine et que nous noterons aussi z, par abus de notation. Rappelons que chaque noeud z de l'arbre qui n'est pas une feuille est associé soit à une fonction élémentaire $f_z \in \{\sin, \cos, \text{sqr}, \text{inv}, \ldots\}$, soit à un opérateur binaire commutatif $\oplus_z \in \{+, *\}$. Dans le premier cas, z a un fils noté $\mathit{fils}(z)$ et dans le deuxième cas, z a deux fils notés $\mathit{fils}_1(z)$ et $\mathit{fils}_2(z)$. L'opérateur inverse de $\oplus_z$ est dénoté $\ominus_z$, c'est-à-dire, si $\oplus_z = +$ alors $\ominus_z = -$, et si $\oplus_z = *$ alors $\ominus_z = /$. Les opérations élémentaires sur les arbres affectent seulement les domaines des noeuds mais pas la structure de l'arbre. En nous inspirant du travail de F. Benhamou, F. Goualard, L. Granvilliers et J.F. Puget [BGGP99a], définissons les opérations élémentaires sur un nœud comme suit :

Evaluation directe d'un noeud unaire :
$\mathbb{D}(z) = f_z(\mathbb{D}(\text{fils}(z)))$.
Evaluation directe d'un noeud binaire :
$\mathbb{D}(\text{z}) = \mathbb{D}(\text{fils}_1(z)) \oplus_z \mathbb{D}(\text{fils}_2(z))$.
Evaluation arrière d'un noeud unaire :
$\mathbb{D}(\text{fils}_1(z)) = f_z^{-1}(\mathbb{D}(z)) \cap \mathbb{D}(\text{fils}(z))$.
Evaluation arrière d'un noeud binaire :
$\mathbb{D}(\text{fils}_1(z)) = (\mathbb{D}(z) \ominus_z \mathbb{D}(\text{fils}_2(z))) \cap \mathbb{D}(\text{fils}_1(z))$.
$\mathbb{D}(\text{fils}_2(z)) = (\mathbb{D}(z) \ominus_z \mathbb{D}(\text{fils}_1(z))) \cap \mathbb{D}(\text{fils}_2(z))$.

Nous supposerons que les domaines des noeuds sont des intervalles ou des unions d'intervalles. L'analyse par intervalles pourra donc être utilisée pour effectuer les évaluations directes et arrières des noeuds.

Exemple 3.5.2 *Considérons un noeud z tel que $\oplus_z = *$. Puisque $*$ est un opérateur binaire, z a deux fils : z_1 et z_2. Ce noeud modélise une relation sur les trois variables z, z_1, z_2 (associées à trois nœuds dans l'arbre) donnée par $z = z_1 * z_2$. Supposons que les*

domaines associés à ces trois nœuds soient : $\mathbb{D}(z) = [1, \infty]$; $\mathbb{D}(z_1) = [-2, 2]$; $\mathbb{D}(z_2) = [-2, 2]$; *Une évaluation arrière nous donne :*

$$\begin{aligned} \mathbb{D}(\mathit{fils}_1(z)) &= ([1,\infty]/[-2,2]) \cap [-2,2] && (3.48)\\ &= ([1,\infty] *]-\infty,-1/2] \cup [1/2,\infty[) \cap [-2,2] && (3.49)\\ &=]-\infty,-1/2] \cup [1/2,\infty[\cap [-2,2] && (3.50)\\ &= [-2,-1/2] \cup [1/2,2] && (3.51) \end{aligned}$$

pour $\mathit{fils}_1(z)$ *et la même chose pour* $\mathit{fils}_2(z)$. *Une évaluation directe nous donne*

$$\begin{aligned} \mathbb{D}(z) &= ([-2,-1/2] \cup [1/2,2]) * ([-2,-1/2] \cup [1/2,2]) \cap [1,\infty] && (3.52)\\ &= ([-4,-1/4] \cup [1/4,4]) \cap [1,\infty] = [1,4]. && (3.53) \end{aligned}$$

On obtient ainsi de nouveaux domaines plus petits pour z, z_1 *et* z_2. ♢

3.5.4 Algorithme de propagation-rétropropagation

Considérons un pavé $[\mathbf{x}] \in \mathbb{IR}^n$, une fonction acyclique f et un intervalle $[y]$. L'algorithme de propagation-rétropropagation [BGGP99a] permet de réduire de façon minimale le pavé $[\mathbf{x}]$ sous la contrainte $f(\mathbf{x}) \in [y]$, c'est-à-dire, de trouver le plus petit pavé qui contient l'ensemble

$$\mathbb{S}_1 = f^{-1}([y]) \cap [\mathbf{x}] \tag{3.54}$$

Cet algorithme utilise deux sous-routines PROPAGATION et RETROPROPAGATION. La propagation fait une évaluation directe pour tous les noeuds de l'arbre en commençant par les feuilles et en finissant à la racine. La rétropropagation commence de la racine et effectue une propagation arrière de tous les noeuds en partant de la racine jusqu'aux feuilles. Dans ce qui suit, z_y est la racine de l'arbre, $z_{x_1}, \ldots, z_{x_n}$ sont les feuilles de l'arbre associées aux composantes $x_1, \ldots, x_n$ de $\mathbf{x}$.

PROPAGATION-RÉTROPROPAGATION$(f, [\mathbf{x}], [y])$
Construire l'arbre associé à f.
PROPAGATION(z_y);
$\mathbb{D}(z_y) := \mathbb{D}(z_y) \cap [y]$;
RETROPROPAGATION(z_y);
Retourner $[\mathbf{q}] := [\mathbb{D}(z_{x_1}) \times \ldots \times \mathbb{D}(z_{x_n})]$;

PROPAGATION(z)

Si z est une feuille,
  si z est associé à un nombre réel c, $\mathbb{D}(z) := c$;
  si z est associé à une composante x_i de $\mathbf{x}$, $\mathbb{D}(z) := [x_i]$;
Si z est associé à un opérateur binaire $\oplus_z$;
  PROPAGATION $(\text{fils}_1(z))$; PROPAGATION $(\text{fils}_2(z))$;
  $\mathbb{D}(z) := \mathbb{D}(\text{fils}_1(z)) \oplus_z \mathbb{D}(\text{fils}_2(z))$;
Si z est associé à une fonction élémentaire f_z,
  PROPAGATION $(\text{fils}(z))$;
  $\mathbb{D}(z) := f_z(\mathbb{D}(\text{fils}(z)))$;

RETROPROPAGATION(z)

Si z est associé à un opérateur binaire $\oplus$,
  $\mathbb{D}(\text{fils}_1(z)) = (\mathbb{D}(z) \ominus_z \mathbb{D}(\text{fils}_2(z))) \cap \mathbb{D}(\text{fils}_1(z))$;
  $\mathbb{D}(\text{fils}_2(z)) = (\mathbb{D}(z) \ominus_z \mathbb{D}(\text{fils}_1(z))) \cap \mathbb{D}(\text{fils}_2(z))$;
  RETROPROPAGATION $(\text{fils}_1(z))$; RETROPROPAGATION $(\text{fils}_2(z))$;
Si z est associé à une fonction élémentaire f_z,
  $\mathbb{D}(\text{fils}_1(z)) = f_z^{-1}(\mathbb{D}(z)) \cap \mathbb{D}(\text{fils}(z))$
  RETROPROPAGATION $(\text{fils}(z))$;

L'optimalité de la réduction est une conséquence directe du fait qu'un arbre n'a pas de cycle [Hyv92]. Lorsque f n'est pas acyclique, l'algorithme peut toujours être utilisé, mais nous n'avons plus la garantie de l'optimalité de la réduction en raison du problème de dépendance.

Afin de bien comprendre cet algorithme, nous allons présenter, sur l'exemple de la figure 3.3 une façon équivalente de raisonner (mais moins automatique). Le lecteur pourra vérifier que le résultat obtenu par l'algorithme de propagation-rétropropagation est identique. Décomposons la relation $y = f(x_1, x_2, x_3)$ en contraintes, dites *primitives*, comme suit

$$y = x_1 \exp(x_2) + 5\sin(x_3) \Leftrightarrow \begin{cases} y = a_1 + a_2 \\ a_1 = x_1 a_3 \\ a_3 = \exp(x_2) \\ a_2 = 5a_4 \\ a_4 = \sin(x_3) \end{cases} \tag{3.55}$$

où les a_i sont des variables intermédiaires. Chaque a_i apparaît exactement une fois sur la gauche et une fois sur la droite. Chaque x_i apparaît exactement une fois sur la droite et y exactement une fois sur la gauche. Cherchons par exemple à calculer le plus petit domaine pour x_3 compatible avec la contrainte $f(x_1, x_2, x_3) \in [y]$. Transformons les relations

primitives de façon à avoir y sur la droite et x_3 sur la gauche. Nous avons

$$\left.\begin{array}{l} a_2 = y - a_1 \\ a_1 = x_1 a_3 \\ a_3 = \exp(x_2) \\ a_4 = a_2/5 \\ x_3 = \sin^{-1}(a_4) \end{array}\right\} \Leftrightarrow x_3 = \sin^{-1}\left(\frac{y - x_1 \exp(x_2)}{5}\right) \tag{3.56}$$

Le domaine réduit pour x_3 est donc

$$\mathbb{D}(x_3) = [x_3] \cap \sin^{-1}\left(\frac{[y] - [x_1] * \exp([x_2])}{5}\right) \tag{3.57}$$

Notons enfin que l'utilisation de domaines (c'est-à-dire des sous-ensembles de $\mathbb{R}$) est nécessaire pour l'optimalité de la réduction. L'utilisation d'intervalles, pour remplacer les domaines, peut introduire un pessimisme comme le montre l'exemple suivant.

Exemple 3.5.3 *Cherchons à réduire* $[x] = [-\pi, \pi]$ *sous la contrainte* $sqr(sin(x)) \in [1/2, 2]\}$. *La décomposition en contraintes primitives est*

$$y = sqr(sin(x)) \Leftrightarrow \left\{\begin{array}{l} y = sqr(a_1) \\ a_1 = \sin(x) \end{array}\right. \Leftrightarrow \left\{\begin{array}{l} a_1 = sqr^{-1}(y) \\ x = \sin^{-1}(a_1) \end{array}\right. \tag{3.58}$$

Soulignons le fait que $sqr^{-1}(y)$ *et* $\sin^{-1}(a_1)$ *sont à comprendre au sens de la théorie des ensembles et ne doivent pas être confondues avec* $\sqrt{y}$ *et* $\arcsin(x_1)$. *Par exemple,* $\sqrt{4} = 2$ *alors que* $sqrt^{-1}(4) = \{-2, 2\}$. *Nous obtenons*

$$\begin{array}{rcl} \mathbb{D}(a_1) & = & sqr^{-1}(\mathcal{Y}) = sqr^{-1}([1/2, 2]) = \left[-\sqrt{2}, -1/\sqrt{2}\right] \cup \left[1/\sqrt{2}, \sqrt{2}\right] \\ \mathbb{D}(x) & = & \sin^{-1}(\mathbb{D}(a_1)) \cap [-\pi, \pi] \\ & = & \sin^{-1}(\left(\left[-\sqrt{2}, -1/\sqrt{2}\right] \cup \left[1/\sqrt{2}, \sqrt{2}\right]\right) \cap [-\pi, \pi] \\ & = & [-3\pi/4, -\pi/4] \cup [\pi/4, 3\pi/4] \end{array} \tag{3.59}$$

La réduction ainsi obtenue est donc $[x] = [-3\pi/4, 3\pi/4]$. *L'utilisation d'intervalles au lieu de domaines introduit du pessimisme :*

$$[a_1] = sqr^{-1}([y]) = sqr^{-1}([1/2, 2]) = \left[-\sqrt{2}, \sqrt{2}\right], \tag{3.60}$$

$$[x] = \sin^{-1}([a_1]) \cap [-\pi, \pi] = [-\pi, \pi]. \tag{3.61}$$

Cet intervalle contient bien celui obtenu par (3.59). ♢

3.6 Propagation de contraintes sur les intervalles

Considérons un ensemble $\mathbb{S}$ et un pavé $[\mathbf{x}]$. Nous allons présenter dans ce paragraphe un outil relativement récent : *la propagation de contraintes sur les intervalles* [Dav87],

[Cle87], qui permet de *réduire* $[\mathbf{x}]$ par rapport à $\mathbb{S}$, c'est-à-dire, de trouver un pavé $[\mathbf{q}]$ contenu dans $[\mathbf{x}]$ tel que $[\mathbf{x}] \cap \mathbb{S} = [\mathbf{q}] \cap \mathbb{S}$. Par simplicité, l'ensemble $\mathbb{S}$ sera ici supposé être formé d'une conjonction de contraintes du type $f_i(x_1, \ldots, x_n) \in [y_i]\,,\ i = 1, \ldots, m$. Lorsque d'autres types de contraintes sont utilisées ou lorsque les opérateurs logiques liant ces contraintes sont formés de *ou* ou de *et,* l'approche se généralise, mais demande des adaptations pas toujours évidentes. L'idée de base repose sur la propagation de contraintes développée initialement par Waltz [Wal75] (sans les intervalles) et qui s'apparente à une méthode de relaxation. L'extension aux intervalles a été proposée indépendamment par Davis [Dav87] et Cleary [Cle87]. Le principe est de prendre une contrainte $\mathbb{C}_i$ (rappelons qu'une contrainte peut être représentée par l'ensemble des $\mathbf{x}$ qui la satisfont), de réduire $[\mathbf{x}]$, d'en prendre une deuxième $\mathbb{C}_j$ et réduire à nouveau $[\mathbf{x}]$, etc. Insistons sur le fait que, dans ce que nous entendons par propagation de contraintes sur les intervalles, aucune idée de bissection n'est admise.

L'algorithme que nous allons présenter génère une séquence décroissante de sous-pavés $[\mathbf{x}](k)$ de $[\mathbf{x}]$, qui contient $\mathbb{S}$. Par *décroissante*, il faut comprendre $[\mathbf{x}](k+1) \subset [\mathbf{x}](k)$. C'est une version simplifiée et moins efficace que l'algorithme de filtrage de Waltz [Wal75]. Son principal avantage est qu'il est rapide à programmer, car il ne demande pas la gestion de structures dynamiques impliquant l'utilisation de pointeurs. Lorsque la réduction $[\mathbf{x}](k)$ est trop faible, l'algorithme s'arrête.

$\boxed{\text{WALTZ}([\mathbf{x}]\,, [\vec{y}])}$
Entrée : $k = 0; [\mathbf{x}](0) = [\mathbf{x}];$
Répéter
  Pour $j = 1$ jusqu'à m,
    $[\mathbf{x}](k+1) = \text{REDUIT}([\mathbf{x}](k), \mathbb{C}_j)\,;$
    $k = k + 1;$
  Fin de la boucle
Tant que la réduction est suffisante;
Sortie : $[\mathbf{x}](k);$

L'algorithme est qualifié de local parce que les contraintes sont prises les unes séparément des autres. Une approche globale de propagation de contraintes a été proposée par Hyvonen [Hyv92]. Ce dernier suggère de supprimer les cycles dans le réseau global de contraintes, mais cela semble n'être possible que pour des situations très particulières. Illustrons l'algorithme de WALTZ sur un exemple où les contraintes ont été choisies linéaires afin de pouvoir faire le calcul à la main.

Exemple 3.6.1 *Soit* $[\mathbf{x}] \in [-10, 10] \times [-10, 10]$ *un pavé que l'on cherche à réduire suivant les deux contraintes* $\mathbb{C}_1 : x_1 + 2x_2 \in [-1, 1]$ *et* $\mathbb{C}_2 : x_1 - x_2 \in [-1, 1]$. *Les fonctions solutions*

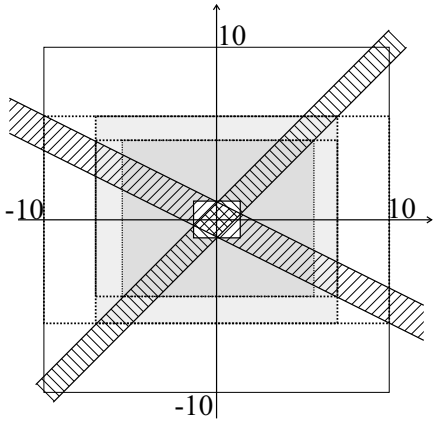

Figure 3.5: Les deux premières itérations de l'algorithme de filtrage de Waltz

sont $g_1^1(x_2, y) = y_1 - 2x_2$, $g_2^1(x_1, y) = \frac{y_1 - x_1}{2}$, $g_1^2(x_2, y) = y_2 + x_2$ *et* $g_2^2(x_1, y) = -y_2 + x_1$. *Une seule itération de la boucle* Répéter-tant que *se trouve décrite ci-dessous*

$$[x_1] = ([-1,1] - 2[-10,10]) \cap [-10,10] = [-10,10] \quad (3.62)$$

$$[x_2] = \tfrac{1}{2}([-1,1] - [-10,10]) \cap [-10,10] = [-\tfrac{11}{2}, \tfrac{11}{2}] \quad (3.63)$$

$$[x_1] = \left([-1,1] + [-\tfrac{11}{2}, \tfrac{11}{2}]\right) \cap [-10,10] = [-\tfrac{13}{2}, \tfrac{13}{2}] \quad (3.64)$$

$$[x_2] = \left([-1,1] + [-\tfrac{13}{2}, \tfrac{13}{2}]\right) \cap [-\tfrac{11}{2}, \tfrac{11}{2}] = [-\tfrac{11}{2}, \tfrac{11}{2}] \quad (3.65)$$

Les deux contraintes sont représentées par la zone hachurée de la figure 3.5. On a $[\mathbf{x}](1) = [-\frac{13}{2}, \frac{13}{2}] \times [-\frac{11}{2}, \frac{11}{2}]$ *en gris clair et* $[\mathbf{x}](2)$ *en gris foncé. Si on continue ainsi jusqu'à l'infini, on aimerait que la séquence* $[\mathbf{x}](k)$ *converge vers le pavé enveloppe* $[\mathbb{S}] = [\mathbb{C}_1 \cap \mathbb{C}_2]$ *représenté en blanc, au centre de la figure 3.5. Pourtant, pour un certain* k, *chaque contrainte contient deux sommets opposés de* $[\mathbf{x}](k)$, *ce qui engendre* $[\mathbf{x}](k+1) = [\mathbf{x}](k)$. *On dit que* $[\mathbf{x}](k)$ *est un pavé fixe. Voici un exemple d'une situation de blocage :*

$$x_1 + x_2 \in [-1,1],\ x_1 - x_2 \in [-1,1] \quad (3.66)$$

$$[\mathbf{x}] = [-10,10] \times [-10,10] \quad (3.67)$$

L'algorithme de Waltz *est alors incapable de réduire* $[\mathbf{x}]$. ♢

La propagation de contrainte sur les intervalles que nous venons de présenter est un outil efficace qui ouvre la porte vers le traitement de problèmes de grandes dimensions. La limitation principale de cette approche est que la réduction peut bloquer sur un pavé $[\mathbf{x}]$ comme cela a été illustré par le système de contraintes (3.66). Pour sortir de ce blocage, plusieurs méthodes peuvent être envisagées.

- *L'ajout de contraintes redondantes* : En rajoutant des contraintes redondantes, il est souvent possible d'améliorer les performances de la propagation de contraintes ou

d'empêcher le blocage (voir [BG97], [MR95]). Ces contraintes redondantes peuvent être obtenues par manipulation algébrique. Par exemple, dans le problème (3.66), on peut obtenir une troisième contrainte en additionnant les deux existantes. Nous obtenons $2x_1 \in [-2, 2]$. L'algorithme de WALTZ est désormais capable de réduire $[\mathbf{x}]$. Si possible, les manipulations algébriques doivent être effectuées dans le but de faire apparaître peu de variables dans les nouvelles contraintes. ♢

- *L'épluchage* : Cette technique repose sur le constat que lorsque le système de contraintes n'admet aucune solution, la propagation de contraintes par intervalles bloque rarement. Notamment dans le cas linéaire (c'est le cas de l'exemple 3.6.1), un tel blocage ne peut avoir lieu car, lorsque la propagation bloque, c'est que le centre du pavé courant $[\mathbf{x}]$ est solution[4]. L'épluchage consiste à prendre une fine tranche de $[\mathbf{x}]$ contenant une de ses faces. Cette tranche est alors réduite par propagation de contrainte dans le but de montrer qu'elle ne contient aucune solution. Si ce but est atteint, on réduit $[\mathbf{x}]$ en lui prélevant cette tranche. On continue la réduction de $[\mathbf{x}]$ en lui prélevant une autre tranche jusqu'à obtenir à nouveau une situation de blocage. Cette technique est celle utilisée par la *box-consistance* (voir [BGGP99b]) et $3B$*-consistance* [5] ([Lho93], [LR97]). L'épluchage est une technique assez coûteuse mais elle permet très souvent de sortir de cette situation de blocage.

- *La linéarisation extérieure* : Cette approche est décrite dans la section suivante. Le principe est de trouver un système de contraintes linéaires dont l'ensemble solution enferme $[\mathbf{x}] \cap \mathbb{S}$. Une technique linéaire par intervalles ou de programmation linéaire permet alors de réduire $[\mathbf{x}]$ sur le système linéarisé et donc de réduite $[\mathbf{x}]$ sur le système de contraintes non-linéaire initial. La méthode de Newton par intervalles [Moo79] est une méthode de linéarisation extérieure.

- *La bissection* : Elle consiste à couper le pavé $[\mathbf{x}]$ en deux pavés $[\mathbf{x}](1)$ et $[\mathbf{x}](2)$, à tenter une réduction sur ces deux pavés puis à en déduire un pavé réduit pour $[\mathbf{x}]$. Lorsque la dimension de $[\mathbf{x}]$ est grande, l'opération de bissection est à effectuer en dernier recours, car une bissection suivant une direction engendre souvent une bissection suivant les autres directions et le temps de calcul explose. Par exemple, en dimension 20, une coupure suivant chaque direction d'un seul pavé génère plus d'un million de pavés. Dans le cas idéal, un découpage ne doit servir que pour séparer des solutions ou casser les singularités.

[4] Ce résultat, qui à ma connaissance est nouveau, est très facile à démontrer.

[5] Dans les techniques de filtrage par 3-B-consistance, on travaille avec 3 bornes. La première est celle qui indique la face de $[\mathbf{x}]$ contenue dans la tranche que l'on traite. Une fois cette borne fixée, on utilise une méthode de propagation de contraintes sur les intervalles pour réduire cette tranche, qui travaille, à un instant donné, à la réduction d'un seul intervalle, c'est-à-dire, 2 bornes.

3.7 Linéarisation extérieure

Cette section présente une approche globale de la réduction de pavés. Dans cette approche, les contraintes ne sont plus prises une par une, mais simultanément (contrairement à ce que fait la propagation de contraintes). Lorsque les contraintes sont du type $f_i(\mathbf{x}) = 0$, et que le nombre de contraintes est égal à la dimension de $\mathbf{x}$ (c'est-à-dire que l'on travaille avec un système de n équations à n inconnues), la méthode de Newton par intervalles (voir [Moo79], [Han92a]) est tout-à-fait adaptée pour la réduction de $[\mathbf{x}]$. Comme nous n'y ferons pas référence et que ma contribution à cette méthode est inexistante, elle ne sera pas présentée dans ce mémoire. Disons simplement que le principe de cette méthode est d'enfermer $f_i(\mathbf{x})$ entre deux hyperplans et de résoudre le système linéaire intervalle associé. Il s'agit bien en quelque sorte d'une linéarisation où les non-linéarités se sont transformées en incertitudes. Nous allons maintenant tenter de garder cette approche pour des contraintes du type $f_i(\mathbf{x}) \in [y_i]$, lorsque le nombre de contraintes et la dimension de $\mathbf{x}$ sont différentes. Pour des raisons évidentes, nous allons qualifier cette approche de *linéarisation extérieure.* Notons que cette approche a été présentée pour la première fois, à ma connaissance, dans [Jau99c].

Cette approche pour la réduction est complémentaire de l'approche par propagation dans le sens où elle réduit mieux les petits pavés, car le comportement des contraintes dans un petit pavé est linéaire. En revanche, lorsque les pavés sont gros, l'approche par propagation fonctionne beaucoup mieux car la linéarisation extérieure est tellement pessimiste qu'elle ne permet pas de les réduire.

Rappelons que nous cherchons à réduire $[\mathbf{x}]$ sous les contraintes $\mathbf{f}(\mathbf{x}) \in [\mathbf{y}]$. Le principe de la linéarisation extérieure est d'encadrer sur $[\mathbf{x}]$ chaque $f_i(\mathbf{x})$ par deux hyperplans parallèles (contrairement à la méthode de Newton par intervalles où les hyperplans ne sont pas parallèles). Nous obtenons ainsi un encadrement de la fonction $\mathbf{f}(\mathbf{x})$ de la forme

$$\mathbf{A}.\mathbf{x} + \mathbf{b}^- \leq \mathbf{f}(\mathbf{x}) \leq \mathbf{A}.\mathbf{x} + \mathbf{b}^+. \tag{3.68}$$

Pour trouver un tel encadrement de $\mathbf{f}$ sur $[\mathbf{x}]$, nous proposons d'utiliser le théorème de la valeur moyenne :

$$\forall \mathbf{x} \in [\mathbf{x}], \mathbf{f}(\mathbf{x}) \in \mathbf{f}(\mathbf{x}_0) + \left[\frac{d\mathbf{f}}{d\mathbf{x}}\right]([\mathbf{x}])(\mathbf{x} - \mathbf{x}_0) \text{ avec } \mathbf{x}_0 = \text{Centre}([\mathbf{x}]) \tag{3.69}$$

où $\left[\frac{d\mathbf{f}}{d\mathbf{x}}\right]([\mathbf{x}])$ est une fonction d'inclusion matricielle pour $\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x})$. Ainsi, nous avons $\mathbf{f}(\mathbf{x}) \in \mathbf{A}.\mathbf{x} + [\mathbf{b}]$ avec

$$\mathbf{A} = \left(\frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}_0)\right), \tag{3.70}$$

$$[\mathbf{b}] = \mathbf{f}(\mathbf{x}_0) - \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}_0).\mathbf{x}_0 + \left(\left[\frac{d\mathbf{f}}{d\mathbf{x}}\right]([\mathbf{x}]) - \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}_0)\right)([\mathbf{x}] - \mathbf{x}_0). \tag{3.71}$$

Notons que

$$\frac{w\left([\mathbf{b}]\right)}{w\left([\mathbf{x}]\right)} = \frac{w\left(\left(\left[\frac{d\mathbf{f}}{d\mathbf{x}}\right]\left([\mathbf{x}]\right) - \frac{d\mathbf{f}}{d\mathbf{x}}(\mathbf{x}_0)\right)\left([\mathbf{x}] - \mathbf{x}_0\right)\right)}{w\left([\mathbf{x}]\right)} \tag{3.72}$$

et donc $w\left([\mathbf{b}]\right)/w\left([\mathbf{x}]\right) \to 0$ lorsque $w\left([\mathbf{x}]\right) \to 0$, ce qui signifie que l'encadrement devient de plus en plus précis, au fur et à mesure que $[\mathbf{x}]$ converge vers un point. La figure 3.6 nous donne un encadrement affine $a.x + [b] = \frac{1}{4}x + [\frac{1}{3}, 1]$ d'une fonction f sur $[x] = [0, 2]$. Or, puisque

$$\begin{aligned}
\mathbf{f}\left(\mathbf{x}\right) \in [\mathbf{y}] &\Leftrightarrow \exists \mathbf{y} \in [\mathbf{y}] | \mathbf{f}\left(\mathbf{x}\right) = \mathbf{y} && (3.73)\\
&\Rightarrow \exists \mathbf{b} \in [\mathbf{b}], \exists \mathbf{y} \in [\mathbf{y}] | \mathbf{A}.\mathbf{x} + \mathbf{b} = \mathbf{y} && (3.74)\\
&\Leftrightarrow \exists \mathbf{b} \in [\mathbf{b}], \exists \mathbf{y} \in [\mathbf{y}] | \mathbf{A}.\mathbf{x} = \mathbf{y} - \mathbf{b} && (3.75)\\
&\Leftrightarrow \mathbf{A}.\mathbf{x} \in [\mathbf{y}] - [\mathbf{b}] = \left[\mathbf{y}^- - \mathbf{b}^+, \mathbf{y}^+ - \mathbf{b}^-\right], && (3.76)
\end{aligned}$$

tout élément $\mathbf{x}$ de $\mathbb{S}$ est nécessairement une solution du système linéaire rectangulaire intervalle

$$\mathbf{A}.\mathbf{x} \in \left[\mathbf{y}^- - \mathbf{b}^+, \mathbf{y}^+ - \mathbf{b}^-\right] \tag{3.77}$$

Le plus petit pavé $[\mathbf{x}_L]$ contenant les solutions de (3.77) peut être calculé en utilisant des techniques de programmation linéaire. Le pavé réduit sera donc $[\mathbf{q}] = [\mathbf{x}] \cap [\mathbf{x}_L]$.

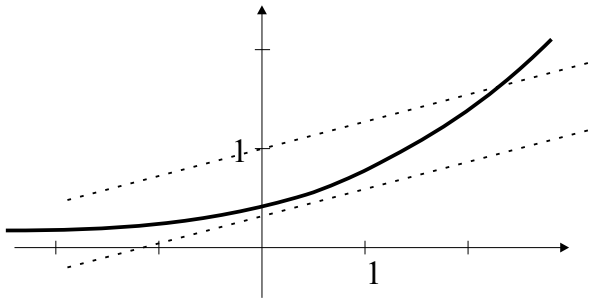

Figure 3.6: Encadrement d'une fonction par deux hyperplans parallèles (ici des droites)

Remarque 3.7.1 *Si nous n'avions pas pris des hyperplans parallèles (comme c'est le cas dans l'approche Newton par intervalles), nous aurions obtenu que tout élément* $\mathbf{x}$ *de* $\mathbb{S}$ *est solution d'un système linéaire intervalle du type*

$$[\mathbf{A}^-, \mathbf{A}^+].\mathbf{x} = \left[\mathbf{b}^-, \mathbf{b}^+\right] \tag{3.78}$$

Ce système est à comprendre en termes de contraintes et de domaines :

$$\mathbf{A}.\mathbf{x} = \mathbf{b},\ \mathbf{A} \in [\mathbf{A}^-, \mathbf{A}^+],\ \mathbf{b} \in \left[\mathbf{b}^-, \mathbf{b}^+\right],\ \mathbf{x} \in [\mathbf{x}] \tag{3.79}$$

Si de plus, $\mathbf{A}^-$ *et* $\mathbf{A}^+$ *avaient été carrées (ce qui n'est pas le cas), nous aurions pu préconditionner le système en multipliant de chaque côté de l'équation (3.78) par l'inverse du*

centre de la matrice intervalle $[\mathbf{A}^-, \mathbf{A}^+]$ *et résoudre le système linéaire intervalle préconditionné par une méthode de Gauss ou de Gauss-Seidel. C'est l'approche classiquement utilisée notamment lorsque l'on utilise la méthode de Newton par intervalles pour résoudre un système du type* $\mathbf{f}(\mathbf{x}) = \mathbf{0}$*. Or, ici* $\mathbf{A}^-$ *et* $\mathbf{A}^+$ *sont rectangulaires, le préconditionnement ne peut donc plus se faire et la résolution efficace de (3.78) est beaucoup plus complexe que dans le cas carré. Une technique de résolution de tels systèmes (surdéterminés, linéaires et intervalles) est donnée dans [Roh96] mais semble être très coûteuse en temps de calcul. Le fait d'avoir imposé* $\mathbf{A}^- = \mathbf{A}^+$ *nous ramène la résolution de* $2n$ *problèmes de programmation linéaire*

$$\begin{cases} \min \pm x_i,\ i = 1, \ldots, n \\ \begin{pmatrix} -\mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{pmatrix} \mathbf{x} \leq \begin{pmatrix} -\mathbf{y}^- + \mathbf{b}^+ \\ \mathbf{y}^+ - \mathbf{b}^- \end{pmatrix} \\ \mathbf{x} \in [\mathbf{x}] \end{cases} \tag{3.80}$$

qui peuvent être résolus avec des méthodes très efficaces. ♢

La figure 3.7 illustre le principe de la linéarisation extérieure sur un exemple type, où la propagation de contraintes bloque. Sur cette figure, $\mathbb{S}$ est l'ensemble des $\mathbf{x}$ qui vérifient les contraintes. Seules les contraintes actives sont représentées (il n'y en a que deux dans notre exemple). Celles que tout $\mathbf{x} \in [\mathbf{x}]$ satisfait n'interviennent plus et ne doivent donc pas être prises en compte. Le pavé $[\mathbf{q}]$ en gris clair est le pavé réduit obtenu par cette technique.

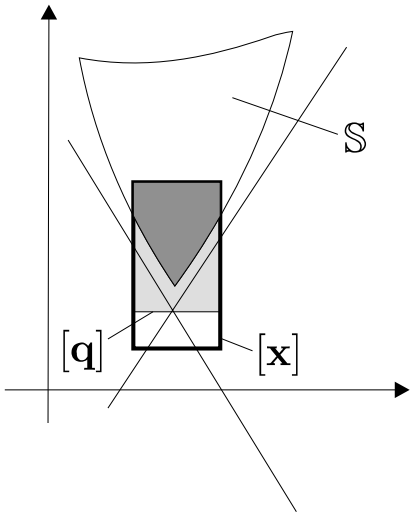

Figure 3.7: Réduction d'un pavé par linéarisation extérieure

Remarque 3.7.2 ***La linéarisation intérieure.*** *A partir de l'encadrement affine de* $\mathbf{f}(\mathbf{x})$ *sur un pavé* $[\mathbf{x}]$*, de la forme*

$$\forall \mathbf{x} \in [\mathbf{x}],\ \mathbf{A}.\mathbf{x} + \mathbf{b}^- \leq \mathbf{f}(\mathbf{x}) \leq \mathbf{A}.\mathbf{x} + \mathbf{b}^+. \tag{3.81}$$

nous pouvons aussi obtenir un polytope contenu dans l'ensemble

$$\mathbb{S}\cap[\mathbf{x}] = \{\mathbf{x} \in [\mathbf{x}] | \mathbf{f}(\mathbf{x}) \in [\mathbf{y}]\}. \tag{3.82}$$

Pour cela, considérons un vecteur $\mathbf{x}$ *de* $[\mathbf{x}]$*, nous avons*

$$\begin{aligned}
\mathbf{f}(\mathbf{x}) \quad &\in \quad [\mathbf{y}] \Leftarrow \left[\mathbf{A}.\mathbf{x} + \mathbf{b}^-, \mathbf{A}.\mathbf{x} + \mathbf{b}^+\right] \subset [\mathbf{y}] && (3.83)\\
&\Leftrightarrow \quad \mathbf{A}.\mathbf{x} + \left[\mathbf{b}^-, \mathbf{b}^+\right] \subset [\mathbf{y}] && (3.84)\\
&\Leftrightarrow \quad \mathbf{A}.\mathbf{x} + \mathbf{b}^- \geq \mathbf{y}^- \textit{ et } \mathbf{A}.\mathbf{x} + \mathbf{b}^+ \leq \mathbf{y}^+ && (3.85)\\
&\Leftrightarrow \quad \mathbf{A}.\mathbf{x} \geq \mathbf{y}^- - \mathbf{b}^- \textit{ et } \mathbf{A}.\mathbf{x} \leq \mathbf{y}^+ - \mathbf{b}^+ && (3.86)\\
&\Leftrightarrow \quad \mathbf{A}.\mathbf{x} \in \left[\mathbf{y}^- - \mathbf{b}^-, \mathbf{y}^+ - \mathbf{b}^+\right]. && (3.87)
\end{aligned}$$

Nous avons donc montré que

$$\mathbf{A}.\mathbf{x} \in \left[\mathbf{y}^- - \mathbf{b}^-, \mathbf{y}^+ - \mathbf{b}^+\right] \Rightarrow \mathbf{f}(\mathbf{x}) \in [\mathbf{y}],$$

c'est-à-dire que tout $\mathbf{x}$ *de* $[\mathbf{x}]$ *qui satisfait* $\mathbf{A}.\mathbf{x} \in [\mathbf{y}^- - \mathbf{b}^-, \mathbf{y}^+ - \mathbf{b}^+]$ *est nécessairement un élément de* $\mathbb{S}$*. On dispose donc d'un polytope*

$$\left\{\mathbf{x} \in [\mathbf{x}] | \mathbf{A}.\mathbf{x} \in \left[\mathbf{y}^- - \mathbf{b}^-, \mathbf{y}^+ - \mathbf{b}^+\right]\right\}$$

contenu dan $\mathbb{S}$*. L'opération qui consiste à obtenir ce polytope sera appelée* linéarisation intérieure *et pourrait avoir des applications pour l'approximation intérieure d'ensembles et pour développer des algorithmes de recherche locale. Mais, cela reste à faire...* ♢

3.8 Conclusion

Dans ce chapitre, nous avons présenté quelques méthodes, principalement fondées sur l'analyse par intervalles, pour calculer des fonctions d'inclusion, tester des pavés ou les réduire lorsqu'ils sont soumis à des contraintes. Ces notions seront utilisées dans les chapitres ultérieurs.

Nous avons particulièrement détaillé différentes techniques de réduction car elles ouvrent les portes vers le traitement de problèmes de plus grandes dimensions. En revanche, elle demandent des contraintes qui possèdent de bonnes propriétés. Lorsque le nombre de variables est très faible (2 ou 3) et si les contraintes entre les variables ne peuvent être exprimées qu'à l'aide d'algorithmes complexes contenant des branchements conditionnels (c'est le cas du problème de localisation présenté au chapitre 5), l'utilisation de techniques de réduction risque alors d'alourdir la mise en oeuvre des algorithmes ensemblistes sans réduire le temps de calcul.

Le chapitre suivant présente des algorithmes pour la mise en oeuvre du calcul ensembliste. Ces derniers utiliseront abondamment les tests d'inclusion, les fonctions d'inclusion ainsi que les opérateurs de réduction, développés dans ce chapitre.

Chapitre 4

Mise en oeuvre du calcul ensembliste

4.1 Introduction

Dans ce chapitre, nous allons montrer comment mettre en oeuvre de façon approchée mais garantie le calcul ensembliste. La mise en oeuvre des algorithmes ensemblistes les plus généraux en découlera comme nous le verrons dans le prochain chapitre.

Pour cela, nous allons utiliser les notions développées au chapitre 3, à savoir les fonctions d'inclusion, les tests d'inclusion et la réduction d'un pavé sous contraintes. Rappelons brièvement que les outils que nous manipulons s'organisent comme indiqué sur la figure 4.1. La flèche est à comprendre dans le sens "*est utilisé par*". Ainsi, si on dispose de l'outil *fonction d'inclusion,* l'outil *calcul par intervalles* n'est pas directement nécessaire à l'élaboration de l'outil *test d'inclusion.*

Dans ce chapitre, où nous cherchons à développer des algorithmes de découpage pour la mise en oeuvre du calcul ensembliste, nous ne parlerons donc pas de calcul par intervalles.

Les sous-pavages seront utilisés pour représenter les ensembles. Parmi les opérations importantes que l'on sera amené à utiliser, mentionnons

- **Le calcul de l'image inverse.** Soit $\widehat{\mathbb{Y}}$ un sous-pavage de $\mathbb{R}^m$ et $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$, il s'agit de trouver un sous-pavage $\widehat{\mathbb{X}}$ qui approxime par l'extérieur et avec une bonne précision l'ensemble
$$\mathbb{X} = f^{-1}\left(\widehat{\mathbb{Y}}\right). \tag{4.1}$$
Ce problème est appelé problème d'*inversion ensembliste.*

- **Le calcul de l'image directe**. Si $\widehat{\mathbb{X}}$ est un sous-pavage de $\mathbb{R}^n$, ce calcul consiste à trouver un sous-pavage qui approxime par l'extérieur et avec une bonne précision

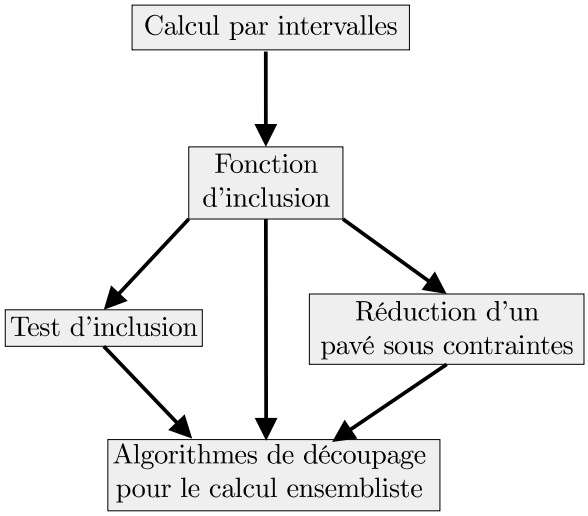

Figure 4.1: Organisation des outils utilisés pour le calcul ensembliste

l'ensemble

$$\mathbb{Y} = \mathbf{f}(\widehat{\mathbb{X}}). \tag{4.2}$$

Ce problème, appelé problème de l'*image ensembliste*, est plus difficile à résoudre que celui de l'inversion ensembliste. Ainsi, lorsque $\mathbf{f}$ est inversible (dans ce cas, $m = n$), il est préférable de transformer ce problème en un problème d'inversion ensembliste.

Nous profiterons aussi de la vue globale du calcul par intervalles pour proposer des algorithmes capables de résoudre des problèmes non-linéaires comme par exemple

- trouver le plus petit pavé qui contient un ensemble défini par des contraintes,
- ou résoudre un problème de minimisation globale.

4.2 Inversion ensembliste

Soit $\mathbf{f}$ une fonction non-linéaire de $\mathbb{R}^n$ dans $\mathbb{R}^m$ et $\widehat{\mathbb{Y}}$ un sous-pavage de $\mathbb{R}^m$. Pour une grande classe de fonctions $\mathbf{f}$, un sous-pavage $\widehat{\mathbb{X}}$ qui enferme avec une précision arbitraire l'ensemble

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) \in \widehat{\mathbb{Y}}\} = \mathbf{f}^{-1}(\widehat{\mathbb{Y}}). \tag{4.3}$$

peut être obtenu par l'algorithme SIVIA1 (Set Inverter Via Interval Analysis) [JW93e] [JW93a]. Notons que les problèmes d'inversion ensembliste apparaissent fréquemment dans les sciences pour l'ingénieur comme par exemple dans l'estimation à erreurs bornées (voir [JW93b]) ou la caractérisation de domaines de stabilité [WJ94].

Nous supposerons que l'on connaît un pavé $[\mathbf{x}]_0$ suffisamment large pour contenir l'ensemble solution $\mathbb{X}$ et que nous disposons d'un test d'inclusion $[t]([\mathbf{x}])$ pour le test $t(\mathbf{x}) = \left(\mathbf{f}(\mathbf{x}) \in \widehat{\mathbb{Y}}\right)$. L'algorithme suivant renvoie un sous-pavage qui contient $\mathbb{X}$.

SIVIA1$([t],[\mathbf{x}],\varepsilon)$
Si $([t]([\mathbf{x}]) = 1)$ ou $(w([\mathbf{x}]) < \varepsilon)$, retourner$([\mathbf{x}])$;
Si $([t]([\mathbf{x}]) = 0)$, retourner $(\emptyset)$;
Retourner SIVIA1$([t], L[\mathbf{x}], \varepsilon) \cup$ SIVIA1$([t], R[\mathbf{x}], \varepsilon)$;

Le réel positif ε est la précision désirée. En machine, l'union des deux sous-pavages est effectuée grâce à l'opérateur de réunification de deux sous-pavages (|) présenté au chapitre 2.

Exemple 4.2.1 *Illustrons cet algorithme à l'aide d'un exemple académique.*

$$\begin{array}{lcl} \exp(x_1) + \exp(x_2) & \in & [10, 11], \\ \exp(2x_1) + \exp(2x_2) & \in & [62, 72]. \end{array} \tag{4.4}$$

Ce problème a été choisi pour son mauvais conditionnement et pour sa symétrie : si (a, b) est une solution, (b, a) est aussi une solution. Il s'agit bien d'un problème d'inversion ensembliste où l'ensemble solution est donné par

$$\mathbb{X} = \mathbf{f}^{-1}([10, 11] \times [62, 72]) \quad \textit{avec } \mathbf{f}(\mathbf{x}) = \begin{pmatrix} \exp(x_1) + \exp(x_2) \\ \exp(2x_1) + \exp(2x_2) \end{pmatrix}. \tag{4.5}$$

Pour $[\mathbf{x}](0) = [0, 4] \times [0, 4]$, et $\varepsilon = 0.001$, SIVIA1 *nous donne en 6 secondes*[1]*, le sous-pavage régulier représenté sur la figure 4.2 (a).* ♢

On peut incorporer à SIVIA1 un opérateur de réduction $[\mathcal{N}]_{\mathbb{X}}$ associé à la contrainte $\mathbf{x} \in \mathbb{X}$. On obtient ainsi l'algorithme suivant

SIVIA2 $([t],[\mathbf{x}],\varepsilon)$
Si $([t]([\mathbf{x}]) = 1)$ ou $(w([\mathbf{x}]) < \varepsilon)$, retourner$([\mathbf{x}])$;
Si $([t]([\mathbf{x}]) = 0)$, retourner $(\emptyset)$;
$[\mathbf{x}] = [N]_{\mathbb{X}}([\mathbf{x}])$;
Retourner SIVIA2 $([t], L[\mathbf{x}], \varepsilon) \cup$ SIVIA2 $([t], R[\mathbf{x}], \varepsilon)$;

Cependant, le sous-pavage généré par SIVIA2 n'est plus régulier et la réunion devient difficile lorsque ce sous-pavage est constitué de beaucoup de pavés. Nous conseillons donc

[1] Tous les temps de calcul référencés dans ce document résultent d'un programme développé en C++ - Builder et exécuté sur un ordinateur personnel équippé d'un Pentium 133MHz.

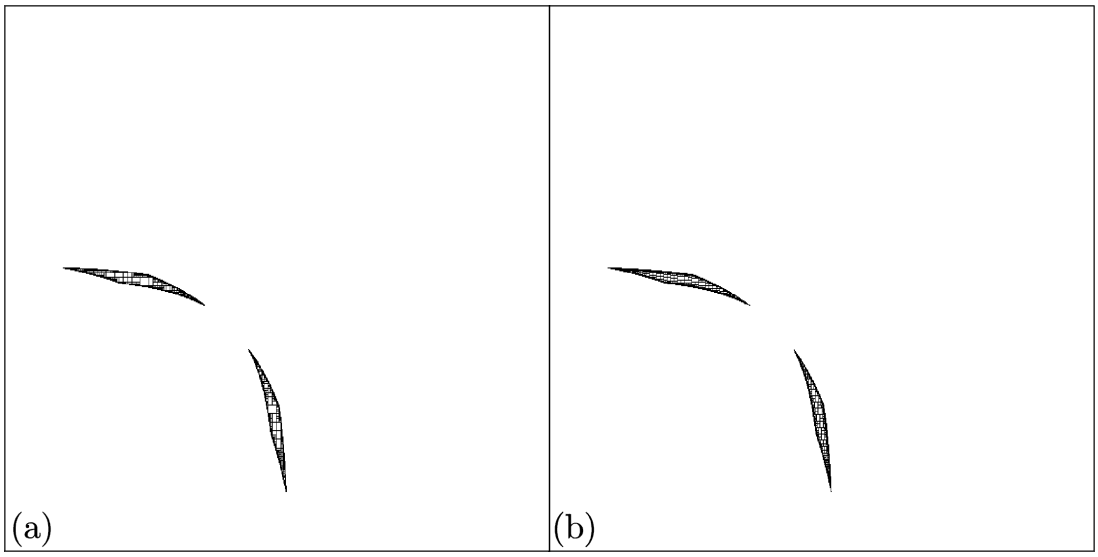

Figure 4.2: Sous-pavages obtenus par SIVIA1 et SIVIA2 enfermant l'ensemble solution.

de n'utiliser la réduction dans SIVIA que lorsque l'ensemble solution est petit (dans ce cas, peu de pavés sont à manipuler et la non régularité n'est plus un problème) et que la dimension de l'espace de recherche est assez élevée (typiquement supérieure à trois). En effet, pour les grandes dimensions, la réduction modère considérablement la dépendance exponentielle du temps de calcul en fonction de la dimension de l'espace de recherche). Dans l'exemple 4.2.1, toujours pour $[\mathbf{x}](0) = [0,4] \times [0,4]$, et $\varepsilon = 0.001$, SIVIA2 obtient le sous-pavage représenté sur la figure 4.2 (b) en 3.8 secondes.

4.3 Image ensembliste

Le calcul de l'image directe d'un sous-pavage est un peu plus compliqué que le calcul de l'image réciproque, car l'analyse par intervalles ne permet pas de construire directement un test d'inclusion pour le test $t(\mathbf{y}) = (\mathbf{y} \in \mathbf{f}(\mathbb{X}))$. Remarquons que, même dans le cas ponctuel, ce test est généralement très difficile à évaluer, contrairement au test d'inclusion associé à l'inversion ensembliste $t(\mathbf{x}) = \left(\mathbf{x} \in \mathbf{f}^{-1}(\mathbb{Y})\right)$. En effet, pour tester si $\mathbf{x} \in \mathbf{f}^{-1}(\mathbb{Y})$, il suffit de calculer $\mathbf{f}(\mathbf{x})$ et de vérifier qu'il est bien dans $\mathbb{Y}$. Par contre, pour tester si $\mathbf{y} \in \mathbf{f}(\mathbb{X})$, il nous faut montrer que le système d'équation $\mathbf{f}(\mathbf{x}) = \mathbf{y}$ sous la contrainte $\mathbf{x} \in \mathbb{X}$ admet au moins une solution, ce qui n'est pas une mince affaire.

Supposons que $\mathbf{f}$ est continue et que une fonction d'inclusion convergente $[\mathbf{f}]$ pour $\mathbf{f}$ est disponible. L'algorithme que nous allons présenter [Kie99] nous génère un sous-pavage régulier $\widehat{\mathbb{Y}}$ qui contient l'image $\mathbb{Y}$ par $\mathbf{f}$ d'un sous-pavage régulier $\widehat{\mathbb{X}}$. On aura donc

$$\mathbb{Y} \triangleq \mathbf{f}\left(\widehat{\mathbb{X}}\right) \subset \widehat{\mathbb{Y}}. \tag{4.6}$$

L'ensemble $\mathbb{Y}$ est inclus dans le pavé $[\mathbf{f}]\left([\widehat{\mathbb{X}}]\right)$, c'est-à-dire, l'image par la fonction d'inclusion du plus petit pavé contenant $\widehat{\mathbb{X}}$. L'algorithme d'approximation extérieure de $\mathbb{Y}$ que nous allons maintenant présenter procède en trois étapes : le *hachage,* l'*évaluation* et la *régularisation.* La précision de l'approximation sera représentée par un réel $\varepsilon > 0$ arbitrairement petit.

1. Le hachage consiste à construire un sous-pavage régulier $\widehat{\mathbb{X}}_\varepsilon$ dont tous les pavés sont de taille inférieure à ε.
2. L'évaluation calcule pour chaque pavé $[\mathbf{x}]$ de $\widehat{\mathbb{X}}_\varepsilon$ le pavé $[\mathbf{f}]([\mathbf{x}])$ et range tous les pavés obtenus dans une liste $\mathcal{Y}$.
3. La régularisation calcule un sous-pavage régulier $\widehat{\mathbb{Y}}^+$ qui contient l'union de tous les pavés de $\mathcal{Y}$. Cette régularisation se fait grâce à un appel à SIVIA où on inverse $\mathcal{Y}$ par la fonction identité. En effet, puisque $\mathbf{f}(\mathbb{X}) \subset \mathcal{Y}$, c'est-à-dire $\mathbf{f}(\mathbb{X}) \subset \mathrm{Id}^{-1}(\mathcal{Y})$, on aura $\mathbf{f}(\mathbb{X}) \subset \textsc{Sivia}([t], [\mathbf{f}]\left([\widehat{\mathbb{X}}]\right), \varepsilon)$, où $[t]$ est un test d'inclusion pour $t(\mathbf{y}) = (\mathbf{y} \in \mathcal{Y})$ et sera noté $[t]([\mathbf{y}]) = ([\mathbf{y}]\,[\in]\,\mathcal{Y})$

L'algorithme est donné ci-dessous

$$\textsc{ImageSP}\ ([\mathbf{f}], \widehat{\mathbb{X}}, \varepsilon)$$
$\mathcal{Y} = \emptyset$;
$\widehat{\mathbb{X}}_\varepsilon =$ Hache $\left(\widehat{\mathbb{X}}, \varepsilon\right)$;
Pour chaque $[\mathbf{x}] \in \widehat{\mathbb{X}}_\varepsilon$, $\mathcal{Y} = \mathcal{Y} \cup \{[\mathbf{f}]([\mathbf{x}])\}$;
$\widehat{\mathbb{Y}}^+ = \textsc{Sivia}([\mathbf{y}]\,[\in]\,\mathcal{Y}, [\mathbf{f}]\left([\widehat{\mathbb{X}}]\right), \varepsilon)$;
Retourner $\widehat{\mathbb{Y}}^+$;

Théorème 4.3.1 *Si* $\mathbf{f}$ *admet une fonction d'inclusion convergente* $[\mathbf{f}]$*, les ensembles* $\widehat{\mathbb{Y}}^+$ *et* $\mathcal{Y}$ *obtenus par* IMAGESP $([\mathbf{f}], \widehat{\mathbb{X}}, \varepsilon)$ *satisfont aux conditions suivantes :*

$$\begin{array}{ll} \text{(i)} & \mathbf{f}\left(\widehat{\mathbb{X}}\right) \subset \widehat{\mathbb{Y}}^+ \\ \text{(ii)} & h_\infty\left(\mathcal{Y}, \mathbf{f}\left(\widehat{\mathbb{X}}\right)\right) \leq \max_{[\mathbf{x}]\in\widehat{\mathbb{X}}} w([\mathbf{f}]([\mathbf{x}])) \\ \text{(iii)} & h_\infty\left(\mathcal{Y}(\varepsilon), \mathbf{f}\left(\widehat{\mathbb{X}}\right)\right) \to 0 \ \textit{lorsque}\ \varepsilon \to 0 \\ \text{(iv)} & h_\infty\left(\widehat{\mathbb{Y}}^+(\varepsilon), \mathbf{f}\left(\widehat{\mathbb{X}}\right)\right) \to 0 \ \textit{lorsque}\ \varepsilon \to 0 \end{array}$$

où $h_\infty(.,.)$ *est la distance de Hausdorff entre deux ensembles compacts.* ♢

Preuve : *(i)* Du fait que

$$\mathcal{Y} = \bigcup_{[\mathbf{x}]\in\widehat{\mathbb{X}}_\varepsilon} [\mathbf{f}]([\mathbf{x}]) \supset \bigcup_{[\mathbf{x}]\in\widehat{\mathbb{X}}_\varepsilon} \mathbf{f}([\mathbf{x}]) = \mathbf{f}\left(\widehat{\mathbb{X}}_\varepsilon\right) = \mathbf{f}\left(\widehat{\mathbb{X}}\right) \tag{4.7}$$

on a $\mathcal{Y} \supset \mathbf{f}\left(\widehat{\mathbb{X}}\right)$. Or, $\widehat{\mathbb{Y}}^{+} = \text{SIVIA}([\mathbf{y}]\,[\in]\,\mathcal{Y}, [\mathbf{f}]\left([\widehat{\mathbb{X}}]\right), \varepsilon) \supset \mathcal{Y}$. Ainsi, $\widehat{\mathbb{Y}}^{+} \supset \mathbf{f}\left(\widehat{\mathbb{X}}\right)$.

(ii) Pour un pavé $[\mathbf{x}]$, on a toujours $h_\infty\left([\mathbf{f}]\left([\mathbf{x}]\right), \mathbf{f}\left([\mathbf{x}]\right)\right) \leq w\left([\mathbf{f}]\left([\mathbf{x}]\right)\right)$. Donc, du fait de la propriété $h_\infty\left(\mathbb{A}_1 \cup \mathbb{A}_2, \mathbb{B}_1 \cup \mathbb{B}_2\right) \leq \max\left(h_\infty\left(\mathbb{A}_1, \mathbb{B}_1\right), h_\infty\left(\mathbb{A}_2, \mathbb{B}_2\right)\right)$

$$h_\infty\left(\bigcup_{[\mathbf{x}]\in\widehat{\mathbb{X}}}[\mathbf{f}]\left([\mathbf{x}]\right), \bigcup_{[\mathbf{x}]\in\widehat{\mathbb{X}}}\mathbf{f}\left([\mathbf{x}]\right)\right) \leq \max_{[\mathbf{x}]\in\widehat{\mathbb{X}}} h_\infty\left([\mathbf{f}]\left([\mathbf{x}]\right), \mathbf{f}\left([\mathbf{x}]\right)\right) \leq \max_{[\mathbf{x}]\in\widehat{\mathbb{X}}} w\left([\mathbf{f}]\left([\mathbf{x}]\right)\right). \quad (4.8)$$

(iii) Soit $\varepsilon(k) > 0$ une suite de réels qui converge vers 0. IMAGESP $([\mathbf{f}], \widehat{\mathbb{X}}, \varepsilon(k))$ génère donc les suites d'ensembles $\widehat{\mathbb{X}}_\varepsilon(k), \mathcal{Y}(k)$ et $\widehat{\mathbb{Y}}^{+}(k)$. Soit $[\mathbf{x}](k)$ une suite de pavés définie comme suit

$$[\mathbf{x}](k) = \arg\max_{[\mathbf{x}]\in\widehat{\mathbb{X}}_\varepsilon(k)} w\left([\mathbf{f}]\left([\mathbf{x}](k)\right)\right). \quad (4.9)$$

$\varepsilon(k) \to 0$ implique que $w([\mathbf{x}](k)) \to 0$ (d'après la définition de $\widehat{\mathbb{X}}_\varepsilon(k)$) et donc $w([\mathbf{f}]([\mathbf{x}](k)))$ $\to 0$ (propriété de convergence de la fonction d'inclusion). Ainsi, d'après (4.9) et *(ii)*, $h_\infty\left(\mathcal{Y}(k), \mathbf{f}\left(\widehat{\mathbb{X}}\right)\right)$ converge vers 0.

(iv) SIVIA inverse la fonction identité qui satisfait les conditions de continuité du théorème de convergence de SIVIA (voir [JW93e]) donc $h_\infty\left(\widehat{\mathbb{Y}}^{+}(\varepsilon), \mathcal{Y}(k)\right) \to 0$. Ainsi, d'après *(iii)*, $h_\infty\left(\widehat{\mathbb{Y}}^{+}(\varepsilon), \mathbf{f}\left(\widehat{\mathbb{X}}\right)\right)$. ♢

Exemple 4.3.1 *Considérons le sous-pavage régulier $\widehat{\mathbb{X}}$ représenté sur la figure 4.3 (a). Tous les cadres des figures de cet exemple correspondent au pavé $[-3,3] \times [-3,3]$. Ce sous-pavage recouvre l'ensemble*

$$\mathbb{X} = \{(x_1, x_2) \in \mathbb{R}^2 | x_1^2 + x_2^2 \in [1,2]\}. \quad (4.10)$$

Appliquons IMAGESP*, pour calculer une approximation de l'image de $\mathbb{X}$ par la fonction*

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x.y \\ x+y \end{pmatrix}. \quad (4.11)$$

Le hachage génère le sous-pavage régulier $\widehat{\mathbb{X}}_\varepsilon$ de la figure 4.3 (b). Notons que l'on a l'égalité ensembliste $\widehat{\mathbb{X}}_\varepsilon = \widehat{\mathbb{X}}$ bien qu'en terme de liste de pavés, l'égalité ne soit pas vérifiée puisque le nombre de pavés de $\widehat{\mathbb{X}}_\varepsilon$ est plus grand que celui de $\widehat{\mathbb{X}}$. Après l'étape d'évaluation, on obtient une liste de pavés $\mathcal{Y}$ qui enferme l'ensemble $\mathbf{f}\left(\widehat{\mathbb{X}}_\varepsilon\right)$ (voir figure 4.3 (c)). L'étape de régularisation donne le sous-pavage régulier $\widehat{\mathbb{Y}}^{+}$ de la figure 4.3 (d). ♢

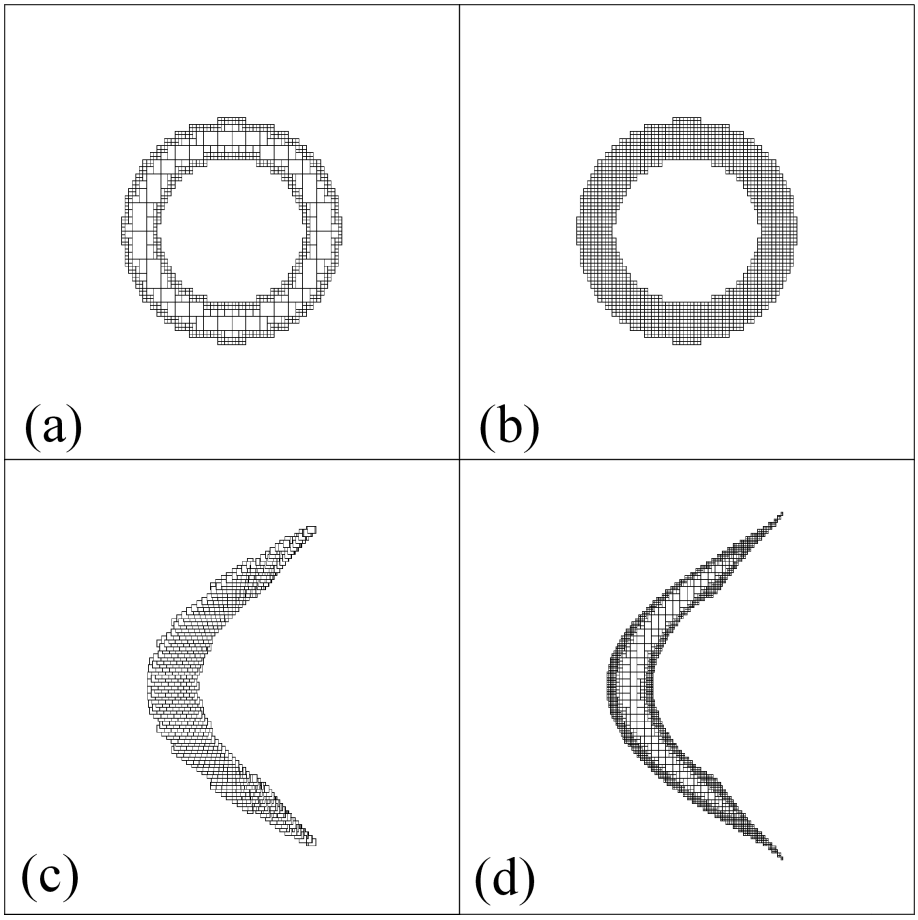

Figure 4.3: Principe de l'algorithme ImageSP. (a) sous-pavage initial, (b) sous-pavage haché, (c) sous-pavage évalué, (d) sous-pavage régularisé

4.4 Calcul du pavé enveloppe d'un ensemble

Soit $\mathbb{X}$ un ensemble défini par une conjonction de contraintes. L'algorithme Sivia nous permet d'obtenir une caractérisation de $\mathbb{X}$ pour une classe très large de problèmes. Mais lorsque la dimension de l'espace de recherche devient trop grand, le sous-pavage engendré s'accumule sur la frontière de $\mathbb{X}$ et la quantité de mémoire et le temps de calcul requis rendent la méthode inutilisable.

Nous allons maintenant nous intéresser à caractériser le *pavé enveloppe* (en anglais : *interval hull*) $[\mathbb{X}]$ de $\mathbb{X}$, c'est-à-dire le plus petit pavé qui contient $\mathbb{X}$. Ainsi, en suivant la remarque de bon sens *si on demande moins ça devrait coûter moins*, nous pouvons espérer obtenir des gains en temps de calcul et en quantité de mémoire. Cette caractérisation simplifiée a souvent un sens parce qu'elle donne les intervalles d'incertitude des variables qui interviennent dans le problème. Nous allons, dans cette section, présenter l'algorithme

HULL [Jau99a] qui encadre $[\mathbb{X}]$ entre deux pavés $[\mathbf{x}_{\text{in}}]$ et $[\mathbf{x}_{\text{out}}]$, au sens où

$$[\mathbf{x}_{\text{in}}] \subset [\mathbb{X}] \subset [\mathbf{x}_{\text{out}}]\,. \tag{4.12}$$

L'algorithme HULL a été utilisé par I. Braems lors de son stage de DEA où elle devait estimer des paramètres dans les systèmes éléctrochimiques [Bra99]. Dans ce qui suit, et sans perte de généralité, nous supposons que $\mathbb{X}$ est contenu dans un pavé de recherche noté $[\mathbf{x}](0)$.

Principe de l'algorithme: HULL construit trois suites: deux suites de pavés $[\mathbf{x}_{\text{in}}](k)$ et $[\mathbf{x}_{\text{out}}](k)$ et une suite de liste de pavés $\mathcal{L}(k)$ qui satisfont:

$$\begin{array}{ll} \textit{(i)} & [\mathbf{x}_{\text{in}}](k) \subset [\mathbb{X}]\,, \\ \textit{(ii)} & \mathbb{X} \subset \mathcal{L}(k) \cup [\mathbf{x}_{\text{in}}](k), \\ \textit{(iii)} & [\mathbf{x}_{\text{in}}](k) \subset [\mathbf{x}_{\text{out}}](k). \end{array} \tag{4.13}$$

La figure 4.4 représente pour une itération k deux pavés $[\mathbf{x}_{\text{in}}](k)$ et $[\mathbf{x}_{\text{out}}](k)$ et une liste $\mathcal{L}(k)$ qui satisfont les conditions *(i), (ii), (iii).* Le principe de HULL est de vider $\mathcal{L}(k)$, en faisant croître $[\mathbf{x}_{\text{in}}](k)$ au maximum et $[\mathbf{x}_{\text{out}}](k)$ au minimum tout en conservant les trois conditions *(i)*, *(ii)* et *(iii).*

Pour cela, plusieurs transformations sont envisagées:

1. **L'inflation intérieure**: Si un point $\mathbf{x} \in \mathbb{X}$ est trouvé (c'est le cas du point représenté par un petit cercle noir sur la figure), on pose $[\mathbf{x}_{\text{in}}](k+1) = [[\mathbf{x}_{\text{in}}](k) \cup \{\mathbf{x}\}]$ et $[\mathbf{x}_{\text{out}}](k+1) = [[\mathbf{x}_{\text{out}}](k) \cup \{\mathbf{x}\}]$.

2. **L'engloutissement**: Si un pavé $[\mathbf{x}]$ de $\mathcal{L}(k)$ est inclus dans $[\mathbf{x}_{\text{in}}](k)$, on l'enlève de $\mathcal{L}(k)$ en posant $\mathcal{L}(k+1) := \mathcal{L}(k) - \{[\mathbf{x}]\}$. On dit que $[\mathbf{x}]$ a été englouti par $[\mathbf{x}_{\text{in}}](k+1)$.

3. **Le rabotage:** Si un pavé $[\mathbf{x}]$ de $\mathcal{L}(k)$ peut être coupé en deux sous-pavés $[\mathbf{x}]_1$ et $[\mathbf{x}]_2$ avec $[\mathbf{x}]_1 \subset [\mathbf{x}_{\text{in}}](k)$ et Intérieur$([\mathbf{x}]_2) \cap [\mathbf{x}_{\text{in}}](k) = \emptyset$, on procède à l'opération $\mathcal{L}(k+1) := \mathcal{L}(k) - \{[\mathbf{x}]\} + \{[\mathbf{x}]_2\}$. Sur la figure 4.4, un seul de tous les pavés gris peut être raboté: c'est celui le plus à gauche du lot de droite.

4. **La réduction**: Si on dispose d'un opérateur de réduction $\mathcal{N}_{\mathbb{X}}$ pour $\mathbb{X}$, on peut remplacer tout pavé $[\mathbf{x}]$ de $\mathcal{L}(k)$ par $\mathcal{N}_{\mathbb{X}}([\mathbf{x}])$. Ainsi $\mathcal{L}(k+1) := \mathcal{L}(k) - \{[\mathbf{x}]\} + \{\mathcal{N}_{\mathbb{X}}([\mathbf{x}])\}$.

5. **L'inflation extérieure**: Si un pavé $[\mathbf{x}]$ de $\mathcal{L}(k)$ est de taille inférieure à ε fixé, on décide de ne pas poursuivre la recherche. On enlève alors $[\mathbf{x}]$ de $\mathcal{L}(k)$ et on gonfle $[\mathbf{x}_{\text{out}}]$ afin qu'il contienne $[\mathbf{x}]$. Ce qui se traduit par: $\mathcal{L}(k+1) = \mathcal{L}(k) - [\mathbf{x}]$; $[\mathbf{x}_{\text{out}}](k+1) = [\mathbf{x}_{\text{out}}](k) \sqcup [\mathbf{x}]$. Dans la situation représentée par la figure 4.4,

si le petit pavé en gris (en haut de la figure) est jugé trop petit, alors, à l'itération suivante, $[\mathbf{x}_{\text{out}}]$ va le contenir, même si ce dernier ne contient aucune solution.

6. **La bissection** : Un pavé $[\mathbf{x}]$ de $\mathcal{L}(k)$ est bissecté en $[\mathbf{x}]_1$ et $[\mathbf{x}]_2$. $\mathcal{L}(k+1) = \mathcal{L}(k) - [\mathbf{x}] + [\mathbf{x}]_1 + [\mathbf{x}]_2$. Cette opération doit être faite seulement lorsque toutes les autres opérations ont échoué à réduire $[\mathbf{x}]$, car c'est cette opération de bissection qui rend la complexité de l'algorithme exponentielle en la dimension de $[\mathbf{x}]$.

Il est aisé de vérifier que les trois propriétés *(i)*, *(ii)* et *(iii)* restent inchangées après chacune de ces transformations entre le rang k et le rang $k+1$.

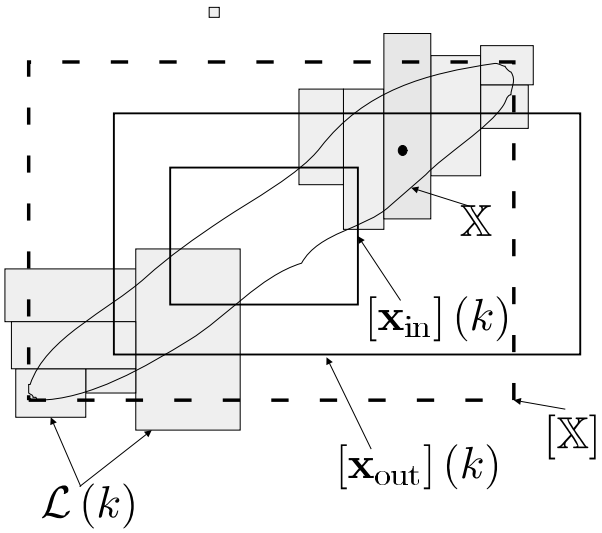

Figure 4.4: Principe de l'algorithme HULL

De plus, nous avons les propriétés suivantes.

$$[\mathbf{x}_{\text{in}}](k) \subset [\mathbf{x}_{\text{in}}](k+1) \text{ (} [\mathbf{x}_{\text{in}}] \text{ est croissante)}, \tag{4.14}$$

$$[\mathbf{x}_{\text{out}}](k) \subset [\mathbf{x}_{\text{out}}](k+1) \text{ (} [\mathbf{x}_{\text{out}}] \text{ est croissante)}, \tag{4.15}$$

$$\mathcal{L}(k) \supset \mathcal{L}(k+1) \text{ } (\mathcal{L} \text{ est décroissante}). \tag{4.16}$$

Donnons maintenant l'algorithme HULL. J'ai choisi, par simplicité, de prendre une file d'attente pour représenter la liste $\mathcal{L}(k)$, mais d'autres structures semblent plus adaptées. On pourrait par exemple décider de représenter $\mathcal{L}(k)$ par une liste triée suivant la distance $h_0([\mathbf{x}], [\mathbf{x}_{\text{out}}])$ (voir équation (2.30) page 24 pour la définition de la proximité). Le pavé maximisant cette distance doit être choisi en premier. Cette idée a été utilisée dans l'algorithme GOVIA (voir [Jau94]).

HULL($[\mathbf{x}]$)

1 $[\mathbf{x}_{\rm in}] := \emptyset$; $[\mathbf{x}_{\rm out}] = \emptyset$; $\mathcal{L} = [\mathbf{x}]$; $k = 0$;
2 Si $\mathcal{L} = \emptyset$, renvoyer $[\mathbf{x}_{\rm out}]$ et $[\mathbf{x}_{\rm in}]$; sinon, prendre un pavé $[\mathbf{x}]$ de $\mathcal{L}$; $k = k + 1$;
3 Chercher par une approche locale, initialisée au centre de $[\mathbf{x}]$, des points de $\mathbb{X} \backslash [\mathbf{x}_{\rm in}]$;
Chaque fois qu'un tel point $\tilde{\mathbf{x}}$ est trouvé, faire $[\mathbf{x}_{\rm in}] = [\mathbf{x}_{\rm in}] \sqcup \{\tilde{\mathbf{x}}\}$; (*inflation intérieure*) ;
4 $[\mathbf{x}_{\rm out}] = [\mathbf{x}_{\rm out}] \sqcup [\mathbf{x}_{\rm in}]$;
5 Si $[\mathbf{x}] \subset [\mathbf{x}_{\rm in}]$, aller à 2 ; (*engloutissement*)
6 Réduire $[\mathbf{x}]$ par *rabotage* de $[\mathbf{x}]$ par $[\mathbf{x}_{\rm in}]$;
7 $[\mathbf{x}] = \mathcal{N}_{\mathbb{X}}([\mathbf{x}])$; (*réduction*)
8 Si $[\mathbf{x}] = \emptyset$, aller à 2 ;
9 Si $w([\mathbf{x}]) < \varepsilon$, $[\mathbf{x}_{\rm out}] = [\mathbf{x}_{\rm out}] \sqcup [\mathbf{x}]$; aller à 2 ; (*inflation extérieure*)
10 Bissecter $[\mathbf{x}]$ et ranger les deux pavés fils dans $\mathcal{L}$;

Rappelons que $\sqcup$, défini par $\mathbb{A} \sqcup \mathbb{B} = [\mathbb{A} \cup \mathbb{B}]$ (voir (2.38), page 26), génère nécessairement un pavé. Parmi les propriétés de HULL, nous avons

$$\exists k_f \;>\; 0 \text{ tel que } \mathcal{L}(k_f) = \emptyset, \tag{4.17}$$

$$[\mathbf{x}_{\rm in}](k_f) \;\subset\; [\mathbb{X}] \subset [\mathbf{x}_{\rm out}](k_f), \tag{4.18}$$

ce qui peut se traduire par le fait que HULL termine et donne un encadrement garanti de $[\mathbb{X}]$. Les conditions de convergence de $[\mathbf{x}_{\rm in}](k_f)$ et $[\mathbf{x}_{\rm out}](k_f)$ vers $[\mathbb{X}]$ lorsque ε tend vers 0 n'ont pas encore été étudiées. Un algorithme de recherche intérieure très efficace CROSS, dont le but est de faire gonfler $[\mathbf{x}_{\rm in}]$ a été proposé dans [Jau99a]. Son originalité est d'utiliser l'analyse par intervalles pour faire une recherche locale très rapide.

Remarque 4.4.1 *Une autre version de l'algorithme existe (voir [Jau99a]) où l'engloutissement et le rabotage se font par rapport à $[\mathbf{x}_{out}]$ au lieu de $[\mathbf{x}_{in}]$ dans les étapes 5 et 6 de* HULL. *Cette version est préférable lorsque on a de grosses difficultés à trouver des points de $\mathbb{X}$ (comme c'est le cas si $\mathbb{X}$ est de volume nul). Dans ce cas, on abandonne l'idée d'avoir une bonne approximation intérieure afin d'avoir un temps de calcul bien meilleur.* ♢

Exemple 4.4.1 *Pour l'exemple du paragraphe 4.2,* HULL *obtient en 0.055 secondes le plus petit cube contenant $\mathbb{X}$, avec une précision de 6 chiffres significatifs. Les deux pavés représentés sur la figure 4.5 résultent de l'unique bissection qui a été nécessaire. L'union de ces deux pavés forme le pavé $[\mathbf{x}_{out}](k_f)$, où k_f est la valeur de k lorsque l'algorithme se termine. Cela signifie que* HULL *a tout d'abord réussi à obtenir, sans le savoir, une approximation extérieure satisfaisante $[\mathbf{x}_{out}]$ de $[\mathbb{X}]$ sans bissection. Il a alors lancé une recherche locale pour gonfler $[\mathbf{x}_{in}]$, mais n'a réussi à parcourir une seule composante connexe, et n'a pas vu la deuxième.* HULL*, voyant une trop grande différence entre $[\mathbf{x}_{in}]$ et $[\mathbf{x}_{out}]$, a donc bissecté le pavé courant, ce qui lui a permis, non pas de réduire plus, mais de relancer à la recherche intérieure associée à l'autre composante connexe de $\mathbb{X}$. Les escaliers de*

la figure correspondent à la recherche locale, qui, rappelons le n'est pas expliquée dans ce document, mais qui peut être trouvée dans [Jau99a]. ♢

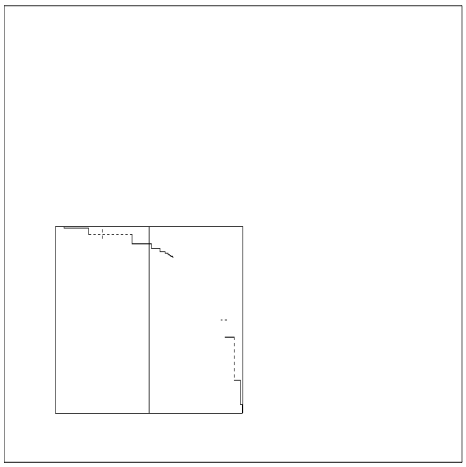

Figure 4.5: HULL appliqué à un exemple

Des exemples beaucoup plus complexes et plus mal conditionnés pris dans la littérature et dans une application en électrochimie ont été résolus [Jau99a] [Bra99] à l'aide de cet algorithme.

4.5 Algorithme de minimisation globale

Nous allons terminer ce chapitre en présentant un algorithme MINIMISE pour le problème de minimisation sans contrainte

$$\min_{\mathbf{x}\in[\mathbf{x}]} f(\mathbf{x}). \tag{4.19}$$

Notre présentation fait une synthèse pédagogique des algorithmes les plus modernes basés sur l'analyse par intervalles. Par simplicité, et bien que l'adaptation ne demande pas beaucoup d'efforts, nous supposons que l'optimum n'est pas sur les bords du pavé $[\mathbf{x}]$ et qu'aucune contrainte, autre que $\mathbf{x} \in [\mathbf{x}]$, n'est présente dans le problème. Dans ce qui suit j^+ est une borne supérieure pour le minimum global j^*, $\mathcal{L}$ est une liste de pavés ayant une structure de file d'attente (le premier entré est le premier sorti). $\mathcal{L}_f$ est une liste de pavés dans laquelle sont rangés tous les pavés jugés trop petits pour être traités. $\mathcal{C}$ est l'ensemble des contraintes à prendre en compte pour la réduction. $\mathcal{L}_f$ contient tous les minimiseurs globaux alors que $[j]$ contient le minimum global.

MINIMISE $([\mathbf{x}], f, \varepsilon)$

1 $j^+ = f\left(\text{centre}\left([\mathbf{x}]\right)\right)$, $\mathcal{L} = \{[\mathbf{x}]\}$, $\mathcal{L}_f = \emptyset$.
2 Si $\mathcal{L} = \emptyset$ aller à 9.
3 Prendre le premier élément de $\mathcal{L}$.
4 Définir l'ensemble des contraintes $\mathcal{C} = \{f\left(\mathbf{x}\right) \leq j^+, \frac{df}{d\mathbf{x}} = \mathbf{0}, \frac{d^2f}{d\mathbf{x}^2} \geq \mathbf{0}, \ldots\}$.
5 Réduire $[\mathbf{x}]$ sous les contraintes $\mathcal{C}$ tout en faisant décroître j^+ autant que possible.
6 Si $[\mathbf{x}] = \emptyset$, aller à 2.
7 Si $w\left([\mathbf{x}]\right) \leq \varepsilon$, ranger $[\mathbf{x}]$ dans $\mathcal{L}_f$ et aller à 2.
8 Bissecter $[\mathbf{x}]$ et ranger les pavés résultants dans $\mathcal{L}$. Aller à 2.
9 Calculer $[j] = \cup_{[\mathbf{x}]\in\mathcal{L}_f}\left[f\right]\left([\mathbf{x}]\right)$.
10 Retourner $\mathcal{L}_f$ et $[j]$.

La réduction au pas 5 utilise toutes les procédures de réduction efficaces disponibles, c'est-à-dire en général, la propagation de contraintes sur les intervalles, le Newton par intervalles ou la linéarisation extérieure. Notons enfin qu'un des algorithmes de minimisation considéré comme un des plus sophistiqués parmi ceux basés sur l'analyse par intervalles est celui d'Hansen [Han92b]. Pourtant, ce dernier n'utilise pas la notion de propagation de contraintes et de ce fait est beaucoup moins performant, pour de nombreux problèmes, que le simple algorithme présenté ici. Enfin, il faut savoir que la propagation de contraintes est très peu répandue dans la communauté des intervalistes. Terminons cette section sur un exemple où la fonction coût résulte d'un problème d'estimation minimax. Cet exemple est tiré de [Jau99c] donné en annexe.

Exemple 4.5.1 *Soit à minimiser la fonction* $f : \mathbb{R}^4 \to \mathbb{R}$ *donnée par*

$$f\left(\mathbf{x}\right) = \max_{k\in\{1,\ldots,10\}} g\left(\mathbf{x}, k\right), \tag{4.20}$$

avec

$$\begin{aligned} g\left(\mathbf{x}, k\right) &= x_1 \exp(\frac{x_2}{4}k^2) + x_3 \exp(\frac{x_4}{4}k^2) && (4.21)\\ &\quad -20\exp(-0.2\ k^2) + 10\exp(-0.05\ k^2) && (4.22)\\ &\quad +0.1\ \sin(\frac{1}{4}k^2). && (4.23) \end{aligned}$$

Notons, que cette fonction n'est pas structurellement globalement minimisable *car une permutation de* x_1 *avec* x_3 *et de* x_2 *avec* x_4 *n'affecte pas la valeur de la fonction coût. Ainsi, toute méthode de minimisation garantie doit normalement générer des solutions symétriques. Ce problème est très mal conditionné et même des méthodes locales classiques ont des difficultés à trouver un minimum local [Jau99c]. De plus, elles ne détectent jamais que le problème a deux solutions. En prenant pour seule contrainte* $(f\left(\mathbf{x}\right) \leq j^+)$ *et aucune contrainte sur les dérivées, l'algorithme* Minimise *trouve en* 1.7 *secondes, pour un pavé de recherche*

$$[\mathbf{x}] = [-60, 60] \times [-1, 0] \times [-60, 60] \times [-1, 0] \tag{4.24}$$

et après 109 *bissections, que l'optimum global est dans l'intervalle* $[0.0653, 0.0657]$*. Le sous-pavage* $\mathcal{L}_f$ *obtenu, censé contenir toutes les solutions, est composé de 44 pavés minuscules et admet deux composantes connexes symétriques. Chaque pavé* $[\mathbf{x}]$ *de* $\mathcal{L}_f$ *satisfait* $f([\mathbf{x}]) \in [0.0653, 0.0657]$*.* ♢

4.6 Conclusion

Dans ce chapitre, nous avons montrer comment mettre en oeuvre le calcul ensembliste de façon approchée mais garantie. Pour cela, un algorithme de calcul de l'image directe et un algorithme de calcul de l'image inverse d'un ensemble ont été présentés. L'approximation choisie pour les ensembles est le sous-pavage extérieur. Ainsi, un algorithme ensembliste peut être approximé par le même algorithme, où les ensembles ont été remplacés par des sous-pavages les enfermant. Les sous-pavages obtenus en sortie de l'algorithme enferment les ensembles qui auraient dû résulter de l'algorithme ensembliste idéal.

Nous avons aussi présenté un algorithme pour le calcul du pavé enveloppe d'un ensemble défini par des contraintes. Cet algorithme utilise une recherche locale, intérieure à l'ensemble solution, afin d'obtenir un critère d'arrêt fiable et accélérer la convergence.

Enfin, un algorithme de minimisation globale a été donné. Cet algorithme est simple, efficace et plus rapide que beaucoup d'algorithmes de la littérature. Il a été appliqué à un problème de minimisation associé un problème d'estimation minimax.

Dans le prochain chapitre, nous allons appliquer la mise en oeuvre du calcul ensembliste à un problème de localisation et de suivi d'un robot mobile.

Chapitre 5

Application à la localisation d'un robot

5.1 Introduction

Ce chapitre a pour but d'illustrer les applications possibles des algorithmes de mise en oeuvre du calcul ensembliste que nous avons proposés au chapitre précédent. Commençons par mentionner quelques applications auxquelles j'ai participé et que nous ne détaillerons pas par souci de brièveté.

- **En estimation à erreurs bornées.** Dans [JGW+97], l'algorithme d'inversion ensembliste a permis d'estimer deux constantes de la physique à partir de mesures réelles. Les particularités de ce problème sont *(i)* que le nombre de mesure est très faible (seulement dix) car chaque mesure obtenue correspond à environ un mois de travail, *(ii)* que les mesures sont de très mauvaise qualité (erreur relative d'environ 50%), et *(iii)* que le modèle néglige certains effets incompris.

- **En planification de chemins** [JG99], [Jau99b]. Nous avons combiné les méthodes ensemblistes avec des algorithmes de théorie des graphes sur les sous-pavages afin de trouver un chemin pour un robot polygonal se déplaçant dans un environnement où les obstacles sont eux aussi polygonaux.

- **En analyse de stabilité** [WJ94],[Did97]. A l'aide de l'inversion ensembliste, nous avons cherché à caractériser des domaines de stabilité pour des modèles paramétriques.

- **En commande robuste** [Did97], [JW96b]. Nous nous sommes intéressé au problème de la recherche d'un régulateur de structure fixée (par exemple PID) capable

de stabiliser à coup sûr un modèle dont les paramètres sont incertains.

- **En commande des systèmes non-linéaires à temps discrets** [JW97]. En formulant le problème de commande en terme d'inversion ensembliste, nous avons proposé une méthode capable de trouver (si elle existe) une commande capable d'amener le système dans une zone donnée de l'espace d'état.

L'application, que j'ai choisi de vous présenter plus en détail dans ce manuscrit, est la localisation et le suivi d'un robot mobile. J'ai travaillé sur ce problème avec D. Meizel, E. Walter et les doctorants O. Lévêque [Lé98] [LJMW97] [JWLM99] et M. Kieffer [KJW98] [KJWM99a]. Le doctorant O. Didrit a aussi participé indirectement à ce travail dans le sens où il a aidé à la résolution du problème d'estimation garantie à erreurs bornées en présence de données aberrantes [JWD96]. Notons que dans le langage des automaticiens, la localisation statique est un problème d'estimation de paramètres (ou identification) alors que le suivi est un problème d'observation de l'état d'un système.

Nous allons tout d'abord définir ce que nous entendons par estimation et observation ensemblistes. Ensuite, nous montrerons comment nos algorithmes peuvent résoudre ces problèmes. Enfin, nous appliquerons les résultats obtenus à la localisation statique puis au suivi du robot.

5.2 Estimateur ensembliste

Dans cette section, nous allons présenter l'estimation ensembliste des paramètres d'un modèle à partir de données expérimentales quand on connaît des bornes sur les erreurs de mesure. Ce type d'estimation, généralement appelée *estimation à erreurs bornées* [JW96a], propose une alternative intéressante à l'estimation par une approche probabiliste. Elle permet en effet souvent de trouver des réponses à des problèmes bien mieux que l'approche probabiliste. C'est le cas, par exemple, lorsqu'interviennent de très fortes non-linéarités ou lorsque le nombre de mesures disponibles est très faible. Nous supposerons ici qu'aucune incertitude n'existe concernant les variables indépendantes (comme les temps de mesure ou les conditions expérimentales par exemple). Toutefois, l'approche s'adapte très bien à de telles incertitudes [JW99].

Considérons un système pour lequel une structure de modèle paramétrique $\mathcal{M}(\mathbf{p})$ est disponible. Cherchons à estimer le vecteur des paramètres $\mathbf{p}$ à partir d'un ensemble de mesures prélevées sur le système. Ces mesures ont été obtenues à la suite d'une expérience ou d'une série d'expériences. Toutes ces mesures sont stockées dans un unique vecteur $\mathbf{y}$. Une simulation de cette série d'expériences à l'aide de notre modèle génère un vecteur

$\mathbf{y}_m(\mathbf{p})$ de même dimension que $\mathbf{y}$. La fonction $\mathbf{y}_m$ est appelée *fonction de simulation* ou *simulateur.* Une estimation ponctuelle consiste à trouver un $\mathbf{p}$ tel que $\mathbf{y}_m(\mathbf{p})$ ressemble le plus possible, au sens d'un critère à définir, au vecteur des mesures $\mathbf{y}$. Dans le cadre de l'estimation ensembliste, on définit l'ensemble des vecteurs des mesures acceptables $\mathbb{Y}$ (en général centré en $\mathbf{y}$), puis on cherche à caractériser l'ensemble $\mathbb{S}$ des $\mathbf{p}$ tels que $\mathbf{y}_m(\mathbf{p}) \in \mathbb{Y}$. L'ensemble $\mathbb{S}$ sera appelé *ensemble de vraisemblance*. Il s'agit ici d'un problème d'inversion ensembliste dans le sens où $\mathbb{S} = \mathbf{y}_m^{-1}(\mathbb{Y})$. L'algorithme SIVIA peut donc être utilisé pour résoudre ce problème. Toutefois, le simulateur est en général une fonction complexe, donnée par un algorithme, avec des boucles et des branchements conditionnels (c'est le cas de notre application sur la localisation du robot traité par la suite). Très souvent, il est impensable d'effectuer la moindre manipulation formelle sur $\mathbf{y}_m(\mathbf{p})$ et ainsi l'opérateur de réduction que l'on aurait pu incorporer dans SIVIA ne sera pas disponible. Par contre, même pour des algorithmes assez complexes, il est généralement possible d'obtenir une fonction d'inclusion efficace $[\mathbf{y}_m]([\mathbf{p}])$ pour $\mathbf{y}_m(\mathbf{p})$ et donc SIVIA peut être utilisé en prenant pour test de vraisemblance $t(\mathbf{p}) = (\mathbf{y}_m(\mathbf{p}) \in \mathbb{Y})$ et pour test d'inclusion $[t]([\mathbf{p}]) = ([\mathbf{y}_m]([\mathbf{p}])\,[\in]\,\mathbb{Y})$. Bien sûr, le nombre de paramètres que l'on peut ainsi espérer identifier est faible (de l'ordre de 4).

Pour des modèles de structure assez simple, comme des sommes d'exponentielles, il est possible d'utiliser des opérateurs de réduction dans SIVIA et on peut ainsi penser résoudre des problèmes impliquant un plus grand nombre de paramètres (typiquement une dizaine).

Dans beaucoup d'applications, les mesures y_i qui forment le vecteur $\mathbf{y}$ sont données avec une barre d'incertitude et l'ensemble des vecteurs des mesures acceptables est un pavé $[\mathbf{y}]$. Il arrive aussi que, du fait de la présence de données aberrantes, l'ensemble $\mathbf{y}_m^{-1}(\mathbb{Y})$ soit vide. Une méthode robuste pour traiter ce genre de problème est d'autoriser qu'un modèle acceptable soit incompatible avec q barres de mesure ([WPL88]). Avec E. Walter et O. Didrit, j'ai montré [JWD96] qu'il s'agissait là d'un problème d'inversion ensembliste où l'ensemble de vraisemblance est défini par

$$\begin{aligned} \mathbb{S}_q &= \mathbf{y}_m^{-1}(\mathbb{Y}_q) \\ \mathbb{Y}_q &= \{\mathbf{y} \in \mathbb{R}^m | \exists j_1, \ldots, j_{m-q} \in \{1, \ldots, m\}, j_1 < \ldots < j_{m-q}, \\ &\qquad y_{j_1} \in [y_{j_1}], \ldots y_{j_{m-q}} \in \left[y_{j_{m-q}}\right] \} \end{aligned} \tag{5.1}$$

et qui peut donc être résolu par SIVIA. Cette technique a été utilisée dans le cadre de la localisation du robot ([LJMW97] et [Kie99]).

5.3 Application à la localisation statique d'un robot

Intéressons nous, dans cette section, à la localisation du robot de l'université de Compiègne dont la photo est donnée sur la figure 5.1. Ce robot dispose d'une carte des différents

obstacles qui l'entourent, et d'une ceinture de 24 capteurs à ultra-sons qui lui fournit des mesures de distances aux obstacles les plus proches.

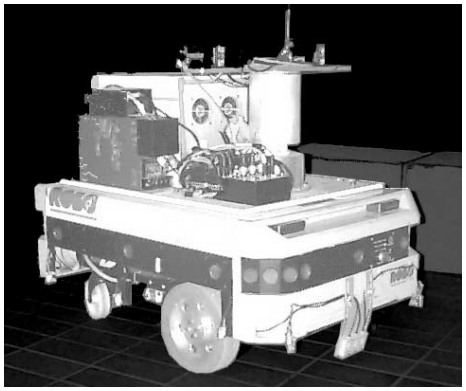

Figure 5.1: Photo du robot *Robuter* de *Robosoft*

Dans ce contexte, les méthodes habituelles de localisation présentent certaines limitations. Dans un premier temps, chaque mesure fournie par un capteur doit être associée à un élément de la carte (voir par exemple [Dru87], [GLP87] ou [LDW91], qui utilisent un arbre pour explorer les différentes possibilités d'association). Cette étape d'association des mesures à la carte se révèle complexe car elle demande un temps de calcul exponentiel en fonction du nombre de segments. Une autre limitation de ces techniques classiques réside en leur faible robustesse par rapport à des données aberrantes, dues à un dysfonctionnement des capteurs ou à une carte partiellement erronée (par exemple [NHTS96] ou [HM97]). Par contre, la méthodologie proposée ici permet d'éviter ces limitations.

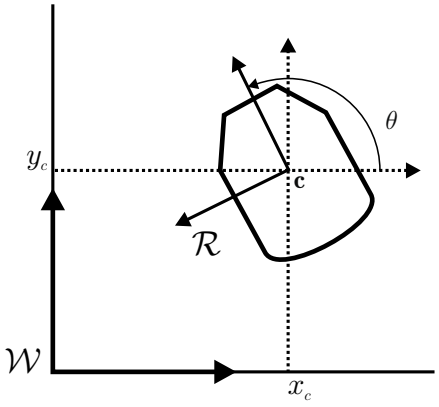

Figure 5.2: Définition de la configuration du robot.

Le robot se déplace dans un espace à deux dimensions et sa configuration est donc donnée par un vecteur $\mathbf{p} = (x_c, y_c, \theta)$, où x_c et y_c représentent les coordonnées de l'origine d'un repère $\mathcal{R}$ attaché au robot dans le repère $\mathcal{W}$ du laboratoire. L'angle θ correspond à

l'orientation de $\mathcal{R}$ par rapport à $\mathcal{W}$ (voir la figure 5.2). Localiser le robot consiste donc à déterminer l'ensemble des valeurs de $\mathbf{p}$ compatibles avec les informations disponibles *a priori*.

Une carte de son environnement est mise à la disposition du robot. Cette carte est constituée d'un ensemble de segments de droite, correspondant aux différents obstacles encombrant l'espace (murs, piliers, *etc.*). Un exemple de carte décrivant trois pièces séparées par des ouvertures est représenté sur la figure 5.3.

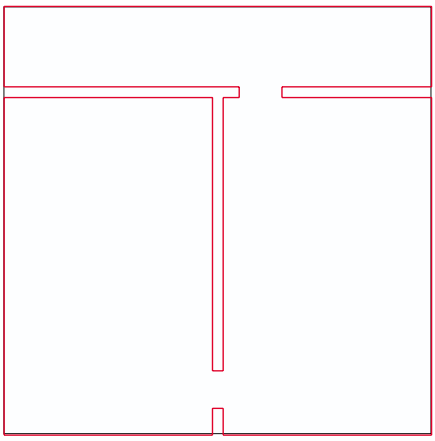

Figure 5.3: Carte de l'environnement du robot.

Chaque capteur i émet une onde ultra-sonore dans une direction donnée qui se propage dans un cône d'émission. Si un obstacle se trouve sur le trajet de l'onde, celle-ci va être réfléchie ; si l'angle d'incidence n'est pas trop élevé, le capteur va percevoir cette onde en retour. Le délai entre émission et réception est converti en une distance d_i composant ainsi la $i^{\text{ème}}$ mesure. Afin de tenir compte des incertitudes de mesures, on forme l'intervalle $[d_i] = [0,98; 1.02]\, d_i$ qui sera supposé contenir la distance réelle du capteur à l'obstacle. L'intervalle $[0,98; 1.02]$ est donné par les caractéristiques des capteurs mesurés en laboratoire. Le problème de localisation est donc un problème d'estimation à erreurs bornées qui peut être résolu par l'algorithme SIVIA (voir la section 5.2). Pour cela, il nous faut disposer d'une fonction d'inclusion pour la fonction de simulation $\mathbf{y}_m(\mathbf{p}) = (d_1^m(\mathbf{p}), \ldots, d_{24}^m(\mathbf{p}))$ qui génère les 24 distances aux obstacles rapportées par les capteurs, connaissant le vecteur de configuration $\mathbf{p}$ et la carte. L'obtention de cette fonction d'inclusion est décrite dans [JWLM99] et [Kie99].

Plaçons notre robot dans la pièce représentée sur la figure 5.3. Une série de mesures est effectuée et le diagramme d'émission obtenu est représenté sur la figure 5.4. La longueur de chaque segment correspond à la distance d_i ; un obstacle ayant produit cette mesure doit se trouver au moins en partie entre les arcs de cercle matérialisant les incertitudes de

mesure.

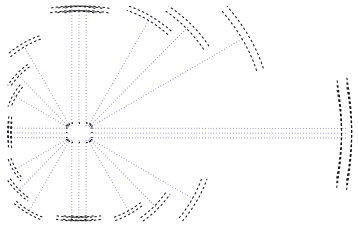

Figure 5.4: Diagramme d'émission des capteurs du robot.

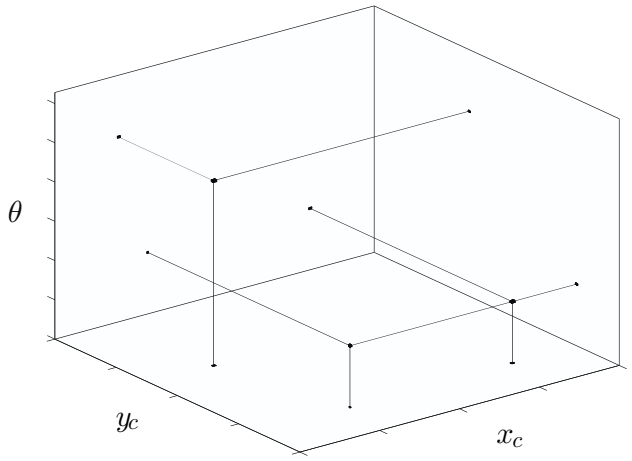

Figure 5.5: Approximation extérieure $\widehat{\mathcal{S}}$ de l'ensemble des configurations admissibles et projections de ces ensembles.

Le pavé de recherche initial est $[\mathbf{p}_0] = [-10\text{ m},\ 10\text{ m}] \times [-10\text{ m},\ 10\text{ m}] \times [0\text{ rad},\ 6.28\text{ rad}]$. SIVIA obtient en 24 s, sur un P233MMX, le sous-pavage $\widehat{\mathcal{S}}$ représenté sur la figure 5.5. Cet ensemble est constitué de trois sous-ensembles disjoints, correspondant chacun à une hypothèse de localisation. La figure 5.6 illustre l'ambiguïté de localisation du robot en indiquant trois configurations possibles. Cette ambiguïté est due à la symétrie locale de la pièce où se trouve le robot. Notons que l'ensemble obtenu contient la vraie configuration $\mathbf{p}^* = (6\text{ m},\ -6.5\text{ m},\ 1.57\text{ rad})$.

L'algorithme d'inversion ensembliste a donc été capable de trouver toutes les configurations du robot compatibles avec les mesures et la carte. Le temps de calcul reste toutefois assez grand, et laisse présager de nombreux problèmes pour la localisation de robot avec un nombre de degrés de liberté plus élevé.

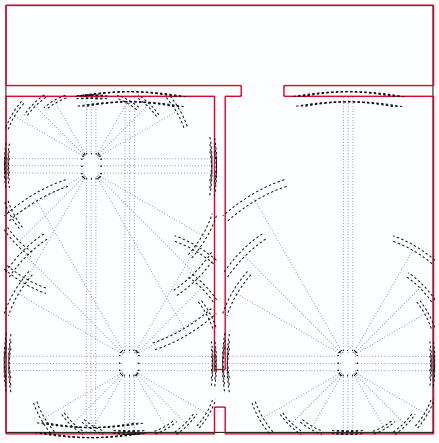

Figure 5.6: Trois des configurations possibles.

5.4 Observateur ensembliste

Dans le but de suivre le robot (c'est-à-dire de le localiser alors que celui-ci bouge, en tenant compte de sa dynamique, des mesures passées et présentes), nous allons définir un estimateur ensembliste idéal que nous mettrons ensuite en oeuvre à l'aide des sous-pavages et des algorithmes de calcul ensembliste présentés dans le chapitre précédent.

Considérons le système autonome discret suivant :

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{f}(\mathbf{x}(k)) \\ \mathbf{y}(k) &= \mathbf{g}(\mathbf{x}(k)) \end{aligned} \tag{5.2}$$

où $\mathbf{x}(k) \in \mathbb{R}^n$ est le vecteur d'état et $\mathbf{y}(k) \in \mathbb{R}^m$ est le vecteur de sortie. Les fonctions vectorielles $\mathbf{f}$ et $\mathbf{g}$ sont non-linéaires. Si k représente le temps, (5.2) est un système à temps discret. Si k est un numéro d'événement et si les composantes de $\mathbf{y}(k)$ sont les temps auxquels certains événements se produisent, (5.2) modélise un système à événements discrets.

Un observateur tente d'estimer les états $\mathbf{x}(k)$ en fonction des mesures bruitées des vecteurs de sortie $\mathbf{y}(0), \ldots, \mathbf{y}(k)$ pour une observation causale et $\mathbf{y}(0)$, $\ldots$, $\mathbf{y}(k)$, $\mathbf{y}(k+1)$, $\ldots, \mathbf{y}\left(k_{\max}\right)$ pour une observation non causale.

Suivant les hypothèses faites sur l'équation d'évolution de l'état et les bruits, l'observation peut être effectuée de plusieurs manières. Dans un contexte linéaire et lorsque les bruits d'état et de sortie sont à distribution gaussienne, le filtre de Kalman est la solution de référence. Dans un contexte à erreurs bornées, le vecteur d'état peut être enfermé dans des récipients tels que des ellipsoïdes ou des parallélotopes (voir par exemple [Sch73], [MN96] ou [DPW96a]). Dans un contexte d'évolution non-linéaire, la dynamique décrivant l'évolution de l'état est souvent linéarisée afin d'utiliser un filtre de Kalman étendu (EKF), mais c'est au prix d'une perte de garantie sur les résultats obtenus.

Rappelons que un cadre ensembliste, les incertitudes sur les grandeurs manipulées sont prises en compte en associant aux vecteurs $\mathbf{x}(k)$ et $\mathbf{y}(k)$ les domaines $\mathbb{X}(k)$ et $\mathbb{Y}(k)$, censés les contenir. Un estimateur ensembliste cherche à réduire les domaines pour $\mathbf{x}(k)$ et $\mathbf{y}(k)$ connaissant les contraintes sur ces variables données par (5.2). Dans le but de simplifier l'énoncé des théorèmes et de faciliter leur démonstration, nous allons supposer que la fonction $\mathbf{f}$ est injective. Cette condition est souvent satisfaite en pratique et elle le sera dans notre application concernant le robot mobile. L'adaptation au cas où $\mathbf{f}$ n'est pas injective est relativement aisée.

5.4.1 Observateur ensembliste causal

Supposons que l'état initial $\mathbf{x}(0)$ appartienne à un ensemble $\mathbb{X}(0)$, qui peut être égal à $\mathbb{R}^n$ dans le cas où aucune information n'est disponible *a priori* sur $\mathbf{x}(0)$. De plus, à chaque instant k, nous est donné un sous-ensemble $\mathbb{Y}(k)$ de $\mathbb{R}^m$, censé contenir la vrai mesure $\mathbf{y}(k)$.

Considérons la suite d'ensembles $\mathbb{X}(k)$ obtenus par l'algorithme ensembliste suivant.

$$\begin{array}{ll} \text{Entrée}: & \mathbb{X}(0), \mathbf{f}, \mathbf{g}, \mathbb{Y}(1), \mathbb{Y}(2), \cdots \\ \text{Initialisation}: & k := 0; \\ \text{Répéter} & k := k+1 \\ & \mathbb{X}(k) := \mathbf{f}(\mathbb{X}(k-1)) \cap \mathbf{g}^{-1}(\mathbb{Y}(k)) \\ \text{Toujours} & \end{array} \tag{5.3}$$

Rappelons que la notion d'algorithme ensembliste est idéale dans le sens où on considère, à tort, savoir calculer avec les ensembles comme on sait calculer avec les réels. L'approximation par sous-pavages des ensembles permet d'en déduire un algorithme approché mais implémentable pour cet algorithme idéal.

Théorème 5.4.1 *Soit $\mathbb{X}(0), \ldots, \mathbb{X}(k)$ la suite d'ensemble générée par notre algorithme ensembliste (5.3) et $\mathbf{x}(0), \ldots, \mathbf{x}(k), \mathbf{y}(1), \ldots, \mathbf{x}(k)$ une suite de vecteurs satisfaisant (5.2). Alors, nous avons l'équivalence :*

$$\mathbf{x}(0) \in \mathbb{X}(0), \mathbf{y}(1) \in \mathbb{Y}(1), \ldots, \mathbf{y}(k) \in \mathbb{Y}(k) \Leftrightarrow \mathbf{x}(k) \in \mathbb{X}(k). \tag{5.4}$$

Preuve: La preuve se fait par récurrence sur k. Supposons ce théorème vrai au rang k, et prouvons qu'il est aussi vrai au rang $k+1$. Du fait que $\mathbb{X}(k+1) = \mathbf{f}(\mathbb{X}(k)) \cap \mathbf{g}^{-1}(\mathbb{Y}(k+1))$, nous avons :

$$\begin{array}{l} \mathbf{x}(k+1) \in \mathbb{X}(k+1) \quad \Leftrightarrow \quad \left\{ \begin{array}{lcl} \mathbf{x}(k+1) & \in & \mathbf{f}(\mathbb{X}(k)) \\ \mathbf{x}(k+1) & \in & \mathbf{g}^{-1}(\mathbb{Y}(k+1)) \end{array} \right. \Leftrightarrow \\ \left\{ \begin{array}{lcl} \mathbf{f}^{-1}(\mathbf{x}(k+1)) & \in & \mathbb{X}(k) \\ \mathbf{g}(\mathbf{x}(k+1)) & \in & \mathbb{Y}(k+1) \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lcl} \mathbf{x}(k) & \in & \mathbb{X}(k) \\ \mathbf{y}(k+1) & \in & \mathbb{Y}(k+1) \end{array} \right. \end{array} \tag{5.5}$$

Pour $k = 0$, la vérification du théorème est immédiate. $\diamondsuit$

Remarque 5.4.1 *Lors de la mise en oeuvre de l'observateur ensembliste causal avec les sous-pavages, à chaque itération, on exécute*

$$\widehat{\mathbb{X}}(k) := \textsc{ImageSP}\left(\mathbf{f},\widehat{\mathbb{X}}(k-1)\right) \cap \textsc{Sivia}\left(\mathbf{g},\widehat{\mathbb{Y}}(k)\right). \tag{5.6}$$

Or, dans de nombreuses applications, il est relativement aisé de trouver un algorithme $\mathbf{f}^{-1}$ *tel que* $\forall \mathbf{x} \in \mathbb{R}^n, \mathbf{f}^{-1}(\mathbf{f}(\mathbf{x})) = \mathbf{x}$. *Ainsi, si on connaît* $\mathbf{x}(k+1)$, *le vecteur d'état à l'instant précédent* $\mathbf{x}(k)$ *peut être calculé à l'aide de l'algorithme* $\mathbf{f}^{-1}$. *Par exemple, si (5.2) résulte d'une discrétisation d'un système physique avec de bonnes propriétés (sans frottement sec, ni hystérésis ...), la fonction* $\mathbf{f}^{-1}$ *peut généralement être obtenue en changeant, dans l'algorithme* $\mathbf{f}$, *les signes de quelques variables comme la vitesse ou les coefficients de frottement. Dans ce cas, l'appel à* IMAGESP *peut être remplacer avantageusement par un appel plus efficace à* SIVIA. *L'itération devient*

$$\widehat{\mathbb{X}}(k) := \textsc{Sivia}\left(\mathbf{f}^{-1},\widehat{\mathbb{X}}(k-1)\right) \cap \textsc{Sivia}\left(\mathbf{g},\widehat{\mathbb{Y}}(k)\right). \tag{5.7}$$

5.4.2 Observateur ensembliste non causal

Supposons que $k_{\max}$ domaines $\mathbb{Y}(k)$ pour les vecteurs de mesures $\mathbf{y}(k)$, $k = 1, \ldots, k_{\max}$ sont disponibles. L'algorithme ensembliste qui suit, calcule les ensembles $\mathbb{X}(k)$ qui contiennent tous les $\mathbf{x}(k)$ compatibles avec les informations disponibles, c'est-à-dire, l'ensemble initial $\mathbb{X}(0)$ et les domaines des mesures $\mathbb{Y}(i)$.

$$\begin{array}{ll}
\text{Entrée :} & \mathbb{X}(0), \mathbf{f}, \mathbf{g}, \mathbb{Y}(1), \mathbb{Y}(2), \cdots, \mathbb{Y}(k_{\max}); \\
\text{For} & k := 1, \text{ to } k_{\max}; \\
 & \mathbb{X}(k) := \mathbf{f}(\mathbb{X}(k-1)) \cap \mathbf{g}^{-1}(\mathbb{Y}(k)) \\
\text{Next} & \\
\text{For} & k := k_{\max} - 1 \text{ to } 0 \\
 & \mathbb{X}(k) := \mathbf{f}^{-1}(\mathbb{X}(k+1)) \\
\text{Next} &
\end{array} \tag{5.8}$$

Théorème 5.4.2

$$\mathbf{x}(0) \in \mathbb{X}(0), \mathbf{y}(1) \in \mathbb{Y}(1), \ldots, \mathbf{y}(k_{\max}) \in \mathbb{Y}(k_{\max}) \Leftrightarrow \mathbf{x}(k) \in \mathbb{X}(k) \tag{5.9}$$

Preuve: La preuve est faite par récurrence descendante sur k. D'après (5.4) le théorème est vrai pour $k = k_{\max}$. Supposons le théorème vrai au rang k et prouvons qu'il est vrai au rang $k-1$:

$$\mathbf{x}(k) \in \mathbb{X}(k) \Leftrightarrow \mathbf{f}^{-1}(\mathbf{x}(k)) \in \mathbf{f}^{-1}(\mathbb{X}(k)) \Leftrightarrow \mathbf{x}(k-1) \in \mathbb{X}(k-1).$$

5.5 Application à la localisation dynamique du robot

Dans cette section nous allons réaliser un suivi du robot en mouvement à l'aide de l'observateur ensembliste causal défini dans la section précédente. La configuration du robot évolue et se trouve donc être maintenant l'état du système. Cette localisation dynamique se fera à partir de la connaissance d'un ensemble *a priori* pour l'état initial et de mesures télémétriques au long du déplacement. Réalisons le suivi du robot à l'aide de l'observateur ensembliste causal décrit précédemment sur un exemple. Notons que l'estimation ensembliste non causale peut présenter de l'intérêt si on s'intéresse à l'historique de celui-ci. Comme par exemple pour savoir sa provenance ou à quelle station il a rechargé ses batteries ...

Considérons pour cela la même pièce que précédemment. Une première localisation statique est mise en œuvre, afin de déterminer un sous-pavage $\widehat{\mathbb{X}_0}$ contenant l'état du robot à l'instant initial. Le sous-pavage obtenu correspond à la solution déjà présentée sur la figure 5.5. Un déplacement consistant en un virage à gauche suivi d'un virage à droite est imposé au robot. La période d'échantillonnage est de 2 s. Le mouvement du robot est obtenue par discrétisation exacte des équations cinématiques.

La figure 5.7 présente la projection sur le plan (x, y) des sous-pavages contenant la véritable configuration à chaque pas d'évaluation. Le sous-pavage du pas 0 correspond à la solution du problème de localisation statique. Après l'étape de correction, au pas 1, un des trois sous-ensembles contenant la configuration admissible a été éliminé, à partir du pas 3, un seul subsiste, qui contient la configuration réelle du robot. L'ensemble de ces 10 pas de simulation est effectué en 17 s. Une application en temps réel est donc envisageable.

La procédure de localisation de l'algorithme de suivi surmonte l'inconvénient de certaines méthodes de suivi disponibles jusqu'alors. Contrairement aux méthode fondées sur le filtrage de Kalman étendu (voir par exemple [Cro89], [LDW91] ou [LDW92]) ou aux techniques à erreur bornées nécessitant une linéarisation, comme celles décrites dans [BR71], [Sch73], [MN96] et [DPW96b], une solution globale et garantie est fournie. Il n'est pas nécessaire d'utiliser un algorithme préalable d'association des mesures avec les éléments de l'environnement, comme dans [Dru87] ou [GLP87]. Contrairement à [HM97], la gestion d'hypothèses multiples de localisation se fait directement. Enfin, aucune procédure préalable d'initialisation n'est nécessaire, contrairement à [NHTS96] ou [LDW91]. En outre, l'estimateur employé peut être rendu très robuste vis-à-vis d'éventuelles données aberrantes (voir [JWD96], [Kie99] et [KJWM99c]).

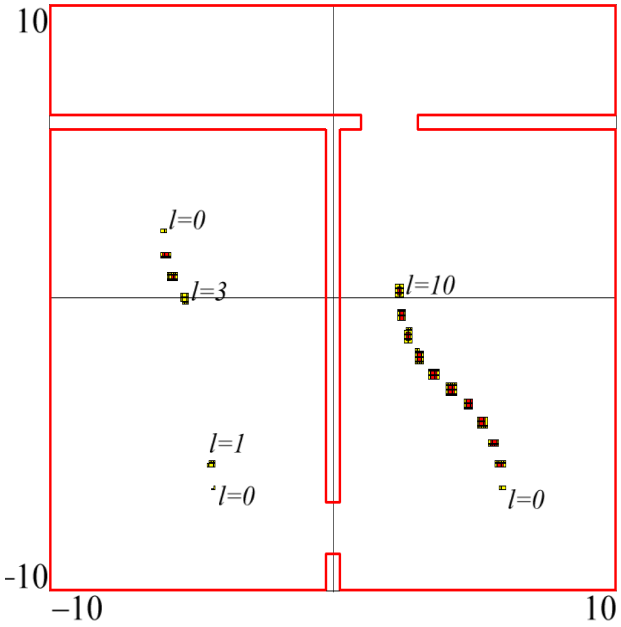

Figure 5.7: Evolution de la projection sur le plan (x, y) des ensembles contenant l'état du robot.

5.6 Conclusion

Dans une approche ensembliste, les variables (réelles ou vectorielles) manipulées sont représentées par des ensembles (ou domaines) censés les contenir. La formulation ensembliste des problèmes d'estimation et d'observation a été faite. Nous avons montré dans ce chapitre que des algorithmes ensemblistes généraux simples sont capables de résoudre ces problèmes. La difficile mise en oeuvre de ces algorithmes ensemblistes est possible, de façon approchée mais garantie, grâce aux sous-pavages et aux algorithmes SIVIA et IMAGESP, proposés au chapitre précédent.

Nous avons ensuite illustré notre approche sur un exemple de localisation et de suivi d'un robot mobile. A chaque instant, l'estimateur a la propriété importante de fournir un ensemble enfermant toutes les valeurs de l'état compatibles avec les informations disponibles.

Chapitre 6

Conclusions et perspectives

Dans ce manuscrit, j'ai tenté de montrer ma contribution à la recherche internationale dans le domaine des sciences pour l'ingénieur. Ce travail de recherche a donné lieu au coencadrement de trois doctorants (O. Didrit, O. Lévêque et M. Kieffer) qui ont soutenu leur thèse. Le travail de I. Braems, qui vient de débuter sa thèse en étudiant les applications de l'analyse par intervalles en électrochimie, a été brièvement présenté.

Résumons à nouveau l'approche que nous avons suivie. Dans beaucoup d'applications des sciences pour l'ingénieur, lorsque les incertitudes sont représentées par des ensembles, il est relativement aisé de trouver un algorithme, dit ensembliste, capable de trouver une, la, ou l'ensemble des solutions recherchées. L'exemple de l'observateur ensembliste causal ou non causal du chapitre 5 en est un exemple probant. Malheureusement, les algorithmes ensemblistes sont idéaux dans le sens où ils font appel à des opérations sur les ensembles que l'on ne sait pas effectuer. En approximant les ensembles par des sous-pavages les enfermant, et en utilisant les opérations élémentaires sur les sous-pavages, il est possible de mettre en oeuvre les algorithmes ensemblistes de façon approximative mais garantie. L'algorithme résultant a été appelé *algorithme extérieur.*

Notons que la formulation ensembliste des problèmes est classique, mais la notion d'algorithme ensembliste et de mise en oeuvre de ces algorithmes ne l'est pas. C'est pourtant cette voie, que je pense prometteuse, que j'ai choisie et qui a été rendue possible grâce au mariage des sous-pavages et de l'analyse par intervalles.

Rappelons comment se structure ce document. Dans le chapitre 2, nous avons montré l'intérêt de l'utilisation d'ensembles pour représenter l'incertitude des quantités manipulées, mais aussi comme outil de parcours de l'espace de recherche. Les opérations et algorithmes ensemblistes ont été définis dans ce même chapitre. Nous avons aussi introduit les sous-pavages qui sont capables d'approximer une grande classe de sous-ensembles de $\mathbb{R}^n$ et qui, grâce au calcul par intervalles, permettent une mise en oeuvre du calcul ensem-

bliste. Le chapitre 3 est consacré au calcul par intervalles qui, rappelons le, est l'outil de base qui a permis l'implémentation de nos algorithmes. Le calcul par intervalles permet de construire des fonctions et des tests d'inclusion efficaces. En le combinant avec des techniques de propagation de contraintes et de linéarisation extérieure, nous avons construit des opérateurs de réduction qui permettent de gagner considérablement en place mémoire et en temps de calcul dans les algorithmes implémentés. Le chapitre 4 a présenté quelques algorithmes utilisant les opérateurs de réduction et les tests d'inclusion dans le but de mettre en oeuvre le calcul ensembliste. Deux de ces algorithmes, SIVIA et IMAGESP ont été utilisés, dans le chapitre 5, pour la localisation d'un robot mobile. Ce problème n'avait jamais obtenu de solutions garanties.

Rappelons les limitations de l'approche ensembliste fondée sur l'analyse par intervalles.

1. *Le problème des grandes dimensions.* Lorsque $\mathbb{X}$ est de grande dimension (typiquement supérieur à 5) et avec un certain volume, une bonne approximation par sous-pavage est impensable. Une piste possible est d'abandonner l'idée d'avoir une bonne approximation pour $\mathbb{X}$ et de se satisfaire d'une approximation grossière (par exemple le nombre de pavés pour représenter $\mathbb{X}$ est limité à 100) et on espère que la redondance des contraintes viendra atténuer l'influence du pessimisme engendré. Lorsque la précision de l'approximation est essentielle et que la dimension de $\mathbb{X}$ est grande, la représentation par sous-pavages devient inapplicable. En bref, on peut dire que l'approche ensembliste par sous-pavages est intéressante lorsque la dimension du problème est faible, que les non-linéarités impliquées sont très fortes et que le temps de résolution n'est pas prépondérant.

2. *Le prix très élevé de la garantie.* Les méthodes ensemblistes proposées sont garanties, mais cette garantie est payée très cher :

   (i) La mise en oeuvre des algorithmes ensemblistes demandent des connaissances en informatique dépassant largement les compétances de l'ingénieur. L'utilisation même des bibliothèques et de logiciels ensemblistes[1], bien que généralement très bien faits, est loin d'être triviale.

   (ii) Le temps de calcul est souvent beaucoup plus long qu'une méthode locale et la classe des problèmes, que l'on peut traiter en un temps raisonnable, est beaucoup plus petite qu'avec des méthodes non garanties.

[1]Pour une présentation des plus courantes, voir *http: //cs.utep.edu/interval-comp/intsoft.html*. Vous trouverez par exemple CLP(BNR) pour la propagation de contraintes sur les intervalles, AWA, FADBAD/TADIFF ou COSY pour les équations différentielles intervalles et la différentiation automatique, INTBLAS pour l'algèbre linéaire sur les intervalles, INTLAB toolbox intervalle pour Matlab 5, PROFIL/BIAS pour Fortran-90 , RVInterval, SvLis, UniCalc, VerGO, ...

(iii) L'ingénieur est souvent très déçu de se retrouver avec plusieurs solutions ou même avec l'ensemble vide, alors qu'il cherche une et une seule solution. L'interprétation d'un résultat ensembliste (non ponctuel) est bien souvent embarrassante, et l'automatisation de cette interprétation est n'est pas une tache facile.

(iv) L'utilisation d'outils ensemblistes va souvent de paire avec une autre formulation des problèmes. En effet, la façon de poser un problème dépend généralement des outils dont on dispose pour le résoudre. Ainsi, dans une approche ensembliste, il peut être préférable de représenter l'incertitude par des bornes plutôt que par des lois de probabilité, de reformuler un problème de minimisation (où on cherche le meilleur) par un problème de satisfaction de contraintes (où on cherche tous ceux qui conviennent), ...

On peut alors se demander si le prix de la garantie n'est un peu lourd. Dans beaucoup d'applications, elle peut être considérée comme du luxe, mais dans d'autres applications, elle peut faire partie du cahier des charges.

La première limitation semble inéluctable, même si on peut la limiter sérieusement par l'utilisation de la propagation de contraintes. Par contre, je pense que la deuxième peut être atténuée à la fois par un effort de pédagogie sur la façon ensembliste de poser les problèmes, par l'élaboration de logiciels conviviaux permettant de résoudre des problèmes concrets et ne demandant aucune connaissance sur l'analyse par intervalles et par le traitement d'applications où toutes les autres méthodes échouent.

Dans un avenir à court et moyen termes, je compte m'intéresser aux quelques idées et problèmes exposés ci-dessous.

- **Traitement des systèmes à temps continu**. De tels systèmes sont généralement régis par des équations différentielles. Bien qu'une formulation ensembliste semble très adaptée dans ce domaine [AF90], les algorithmes de résolution existant semblent peu convinquant. L'approche par sous-pavages, que nous avons proposé, devrait pouvoir s'appliquer, en utilisant des fonctions d'inclusion efficaces pour les solutions d'équation différentielles. De telles fonctions d'inclusion peuvent être obtenues par la méthode de Lohner [Loh87], ou par l'utilisation de zonotopes [Kuh98][2]. Des bibiothèques de calcul de fonction d'inclusion pour les solutions d'équations différentielles sont disponibles[3]. Une telle approche pourrait être utilisée, par exemple, pour prouver la stabilité d'un système de la forme $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$.

[2] Vous pourrez trouver une présentation sur le site *http: //www.zib.de/kuehn/dynamic/index.html.*
[3] voir par exemple la bibliothèque AWA sur *ftp: //iamk4515.mathematik.uni-karlsruhe.de/pub/awa*

- **La linéarisation intérieure** : son principe a été proposé à la remarque 3.7.2 de la page 54. Rappelons que la linéarisation intérieure permet de calculer un polytope contenu dans $[\mathbf{x}] \cap \mathbb{X}$, où $\mathbb{X}$ est défini par une conjonction de contraintes et $[\mathbf{x}]$ est un pavé donné. Cela pourrait nous permettre d'avoir une approximation plus fine de l'intérieur d'un ensemble. De plus, dans beaucoup de problèmes non-linéaires, l'ensemble $\mathbb{X}$ est petit et ressemble à un polytope. On peut alors sans bissection, avoir une approximation fine de l'intérieur de $\mathbb{X}$ en utilisant une telle linéarisation intérieure.
  Une autre application de cette linéarisation intérieure est la recherche intérieure dans un ensemble défini par des contraintes. Ainsi, en combinant la linéarisation intérieure et la programmation linéaire, on peut obtenir un cube contenu dans le pavé enveloppe de $[\mathbf{x}] \cap \mathbb{X}$ et utiliser ce cube pour gonfler $[\mathbf{x}_{\text{in}}](k)$ dans l'algorithme HULL.
  Les algorithmes utilisant l'analyse par intervalles font une analyse extérieure des contraintes, c'est-à-dire qu'ils se limitent généralement à enlever de l'espace de recherche des zones qui ne contiennent pas les solutions du problème. Je pense que l'analyse par intervalles devrait aussi faire ses preuves pour une analyse intérieure des contraintes comme cela est nécessaire dans des algorithmes de recherche locale et de caractérisation intérieure d'ensembles.

- **Etude de la connexité** : Actuellement, aucune méthode numérique ne permet de compter le nombre de composantes connexes d'un ensemble $\mathbb{X}$ défini par des contraintes. On sait seulement avoir un minorant de ce nombre. Cela vient du fait que dans les zones non étudiées, il y a toujours une possibilité d'existence d'un grand nombre de composantes connexes cachées et toutes les méthodes de découpage d'ensembles existantes laissent des zones non étudiées. Une méthode basée sur une linéarisation extérieure devrait nous permettre d'obtenir ce nombre. Notons enfin que l'analyse en connexité d'un ensemble est un problème qui apparaît par exemple en étude d'identifiabilité d'un modèle ou de planification de chemin.
  Une approche du même type devrait être capable de démontrer la convexité d'un ensemble défini par des contraintes.

- **Observation d'état** : A un système $\mathbf{x}(k+1) = \mathbf{f}_k(\mathbf{x}(k)), \mathbf{y}(k) = \mathbf{g}_k(\mathbf{x}(k))$, peut être associé un réseau de contraintes, représenté par un graphe faisant intervenir les variables $x_i(j)$ et $y_i(j)$, $0 \leq j \leq k$. Ce réseau de contraintes évolue avec le temps k. Une technique par propagation de contraintes peut être utilisée pour réduire les domaines associés à chaque variables du réseau lorsque que des mesures de type intervalles nous sont données. Le réseau augmente lorsque de nouvelles variables arrivent et diminue lorsque des variables correctement réduites se trouvent instanciées. Une approche similaire peut aussi être envisagée pour la commande.

- **Contraintes avec quantificateurs** : Beaucoup de problèmes d'automatique peuvent se traduire en termes de contraintes où des quantificateurs $\forall$ interviennent [JW96b]. Or, à ma connaissance, tous les travaux sur la propagation de contraintes sur les intervalles ne concernent que des ensembles avec des contraintes liés au quantificateur $\exists$. Il pourrait donc être intéressant de voir comment les techniques de propagation peuvent s'étendre à des problèmes impliquant des quantificateurs $\forall$.

- **Logiciel d'estimation** : Beaucoup de scientifiques auraient besoin d'outils fiables pour la minimisation et le calcul ensembliste, comme ceux qui ont été proposés dans ce document. Or, les développements informatiques sont complexes et nécessitent une connaissance approfondie de la programmation orientée objet avec surcharges d'opérateurs, qui très souvent rebutent l'utilisateur. Pour que nos méthodes soient utilisées dans le milieu des applications, il ne suffit pas de mettre à disposition des bibliothèques objets en C++ incorporant les outils les plus sophistiqués de l'analyse par intervalles comme la forme centrée, la différentiation automatique ou le Newton par intervalles, il faut créer un logiciel ne réclamant aucune connaissance en C++ et en calcul par intervalles et qui soit directement utilisable par l'ingénieur. Dans cette voie, j'aimerais participer au développement d'un logiciel d'estimation de paramètres où les algorithmes exploiteraient le calcul par intervalles et où les seules connaissances requises pour l'utilisateur du logiciel seraient celles fournies par un cours de base sur l'estimation paramétrique.

Bibliographie

[ABK+93] J. Ackerman, A. Barlett, D. Kaesbauer, W. Sienel, et R. Steinhauser. *Robust Control Systems with Uncertain Physical Parameters*. Springer-Verlag, 1993.

[AF90] J.P. Aubin et H. Frankowska. *Set-Valued Analysis*. Birkhuser, Boston, 1990.

[BAH89] B.R. Barmish, J. E. Ackermann, et H. Z. Hu. The tree structured decomposition: A new approach to robust stability analysis. Dans *Proceedings of the Conference on Information Sciences and Systems*, Baltimore, 1989.

[Bar88] B.R. Barmish. New tools for robustness analysis. Dans *Proceedings of the IEEE Conference on Decision and Control*, pages 399–408, Austin, 1988.

[Bar94] B.R. Barmish. *New tools for robustness of linear systems*. MacMillan, New York, 1994.

[BBKF+99] S. Brahim-Belhouari, M. Kieffer, G. Fleury, L. Jaulin, et E. Walter. Model selection via worst-case criterion for nonlinear bounded-error estimation. A paraître dans Proceedings 16th IEEE Instrumentation and Measurement Tech. Conf., Venise, 24-26 mai 1999.

[Ber79] M. Berger. *Espaces euclidiens, triangles, cercles et sphères*, volume 2 of *Géométrie*. Cedic/Fernand Nathan, Paris, 1979.

[BG97] F. Benhamou et L. Granvilliers. Automatic generation of numerical redundancies for nonlinear constraint solving. *Reliable Computing*, pages 335–344, 1997.

[BGGP99a] F. Benhamou, F. Goualard, L. Granvilliers, et J. F. Puget. Revising hull and box consistency. Dans *International Conference on Logic Programming*, 1999.

[BGGP99b] F. Benhamou, F. Goualard, L. Granvilliers, et J. F. Puget. Revising hull and box consistency. Dans *ICLP'99*, 1999.

[BO97] F. Benhamou et W. Older. Applying interval arithmetic to real, integer and boolean constraints. *Journal of Logic Programming*, pages 1–24, 1997.

[BR71] D. P. Bertsekas et I. B. Rhodes. Recursive state estimation for a set-membership description of uncertainty. *IEEE Trans. on Automatic Control*, 16: 117–128, 1971.

[Bra99] I. Braems. *Estimation Nonlinéaire Robuste et Garantie Grâce À L'analyse Par Intervalles ; Applications En Électrochimie.* D.E.A. Automatique et Traitement du Signal, Paris-Sud., 1999.

[Can96] M. Candev. *Scientific Computation and Validated Numerics*, chapitre On the Application of an Interval Algorithm for Set Inversion, pages 140–146. Akademie Verlag, 1996.

[Cle87] J. C. Cleary. Logical arithmetic. *Future Computing Systems*, pages 125–149, 1987.

[Cro89] J. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. Dans *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 674–680, Scottsdale, Arizona, 1989.

[CWS97] G. Chen, J. Wang, et L. S. Shieh. Interval Kalman filtering. *IEEE Trans. on Aerospace and Electronic Systems*, 33(1): 250–258, 1997.

[Dav87] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, pages 281–331, 1987.

[Did97] O. Didrit. *Analyse par intervalles pour l'automatique; résolution globale et garantie de problèmes non linéaires en robotique et commande robuste.* Thèse de doctorat, Université Paris-Sud, Orsay, juin 1997.

[DJW95] O. Didrit, L. Jaulin, et E. Walter. Guaranteed analysis and optimization of parametric systems, with application to their stability degree. Dans *Proceedings of 3rd European Control Conference*, pages 1412–1417, Rome, 1995.

[DJW97] O. Didrit, L. Jaulin, et E. Walter. Guaranteed analysis and optimization of parametric systems, with application to their stability degree. *European Journal of Control*, 3: 68–80, 1997.

[DPV99] A. L. Dontchev, M. P. Polis, et M. Veliov. A dual approach to deterministic parameter identification. *IEEE Transaction on Automatic Control (accepté)*, 1999.

[DPW96a] C. Durieu, B. Polyak, et E. Walter. Ellipsoidal state outer-bounding for MIMO systems via analytical techniques. Dans *Proc. IMACS—IEEE–SMC CESA'96 Symposium on Modelling and Simulation*, volume 2, pages 843–848, Lille, July 9-12, 1996.

[DPW96b] C. Durieu, B. Polyak, et E. Walter. Trace versus determinant in ellipsoidal outer bounding with application to state estimation. Dans *Proc. 13th IFAC World Congress*, volume I, pages 43–48, San Francisco, June 30-July 5, 1996.

[Dru87] M. Drumheller. Mobile robot localization using sonar. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(2): 325–332, 1987.

[Gar99] J. Garloff. Application of bernstein expension to the solution of control problems. *Reliable Computing (accepté)*, 1999.

[GLP87] W. E. Grimson et T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(4): 469–482, 1987.

[Gra98] L. Granvilliers. *Consistances locales et transformations symboliques de contraintes d'intervalles.* PhD thesis, Université d'Orléans, 1998.

[Han92a] E. R. Hansen. Bounding the solution of interval linear equations. *SIAM Journal of Numerical Analysis*, 29(5): 1493–1503, 1992.

[Han92b] E. R. Hansen. *Global Optimization using Interval Analysis.* Marcel Dekker, New York, 1992.

[HM97] E. Halbwachs et D. Meizel. Multiple hypothesis management for mobile vehicule localization. Dans *CD Rom of the European Control Conference*, Louvain, 1997.

[Hyv92] E. Hyvonen. Constraint reasoning based on interval arithmetic; the tolerance propagation approach. *Artificial Intelligence*, pages 71–112, 1992.

[Jau94] L. Jaulin. *Solution globale et garantie de problèmes ensemblistes ; application à l'estimation non linéaire et à la commande robuste.* PhD dissertation, Université Paris-Sud, Orsay, 1994.

[Jau99a] L. Jaulin. Interval constraint propagation with application to bounded-error estimation. *Automatica (soumis)*, 1999.

[Jau99b] L. Jaulin. Path planning using intervals and graphs. *Reliable Computing (soumis)*, 1999.

[Jau99c] L. Jaulin. Reliable minimax parameter estimation. *Reliable Computing (soumis)*, 1999.

[JB99] L. Jaulin et J. Burger. Proving stability of uncertain parametric models. *Automatica*, pages 627–632, 1999.

[JBH99] L. Jaulin, J. L. Boimond, et L. Hardouin. Estimation via set-inversion: application to discrete-event systems. *Reliable Computing*, pages 165–173, 1999.

[JG99] L. Jaulin et A. Godon. Motion planning using interval analysis. Dans *MISC 99, Application of Interval Analysis to Systems and Control*, pages 335–346, 1999.

[JGW+97] L. Jaulin, J. L. Godet, E. Walter, A. Elliasmine, et Y. Leduff. Light scattering data analysis via set inversion. *Journal of Physics A: Mathematical and General*, pages 7733–7738, 1997.

[JKWM98] L. Jaulin, M. Kieffer, E. Walter, et D. Meizel. Guaranteed robust nonlinear estimation with application to robot localization. *Automatica (soumis)*, 1998.

[JW92] L. Jaulin et E. Walter. Set inversion, with application to guaranteed nonlinear estimation and robust control. Dans *Proc. Of the Workshop on Modeling Techniques for Uncertain Systems (Sopron)*, 1992.

[JW93a] L. Jaulin et E. Walter. Guaranteed nonlinear estimation and robust stability analysis via set inversion. Dans *Proceedings of 2nd European Control Conference*, pages 818–821, 1993.

[JW93b] L. Jaulin et E. Walter. Guaranteed nonlinear parameter estimation from bounded-error data via interval analysis. *Math. and Comput. in Simulation*, 35: 1923–1937, 1993.

[JW93c] L. Jaulin et E. Walter. Guaranteed nonlinear parameter estimation via interval computations. Dans *Conference on Numerical Analysis with Automatic Result Verification, (Lafayette)*, 1993.

[JW93d] L. Jaulin et E. Walter. Guaranteed nonlinear parameter estimation via interval computations. *Interval Computation*, pages 61–75, 1993.

[JW93e] L. Jaulin et E. Walter. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica*, 29(4): 1053–1064, 1993.

[JW94] L. Jaulin et E. Walter. *Modeling Techniques for Uncertain Systems*, chapitre Set Inversion, with Application to Guaranteed Nonlinear Estimation and Robust Control, pages 3–20. Birkhäuser, 1994.

[JW96a] L. Jaulin et E. Walter. *Bounding Approaches to System Identification*, chapitre Guaranteed nonlinear set estimation via interval analysis, pages 363–382. Plenum, 1996.

[JW96b] L. Jaulin et E. Walter. Guaranteed tuning, with application to robust control and motion planning. *Automatica*, 32(8): 1217–12211, 1996.

[JW97] L. Jaulin et E. Walter. Global numerical approach to nonlinear discrete-time control. *IEEE-Trans. on Autom. Control*, 42: 872–875, 1997.

[JW99] L. Jaulin et E. Walter. Guaranteed bounded-error parameter estimation for nonlinear models with uncertain experimental factors. *Automatica*, 35: 849–856, 1999.

[JWD96] L. Jaulin, E. Walter, et O. Didrit. Guaranteed robust nonlinear parameter bounding. *Proc. CESA'96 IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)*, pages 1156–1161, 1996.

[JWLM99] L. Jaulin, E. Walter, O. Lévêque, et D. Meizel. Set inversion for chi-algorithms, with application to guaranteed robot localization. *Automatica (soumis)*, 1999.

[Kie99] M. Kieffer. *Estimation ensembliste par analyse par intervalles, application à la localisation d'un véhicule.* PhD dissertation, Université Paris-Sud, Orsay, 1999.

[KJW98] M. Kieffer, L. Jaulin, et E. Walter. Guaranteed recursive nonlinear state estimation using interval analysis. Dans *Proc. 37th IEEE Conference on Decision and Control*, pages 3966–3971, Tampa, Florida, 16-18 December 1998.

[KJW99] M. Kieffer, L. Jaulin, et E. Walter. Guaranteed recursive nonlinear state estimation using interval analysis. Internal Report Nr LSS-XX, long version of a paper submitted to IEEE Trans. on Aut. Control, 1999.

[KJWM99a] M. Kieffer, L. Jaulin, E. Walter, et D. Meizel. Localisation et suivi garantis d'un robot par analyse par intervalles. Dans *JDA*, 1999.

[KJWM99b] M. Kieffer, L. Jaulin, E. Walter, et D. Meizel. Nonlinear identification based on unreliable priors and data, with application to robot localization. Dans

A. Garulli, A. Tesi, et A. Vicino, editeurs, *Robustness in Identification and Control*, pages 190–203, LNCIS 245, London, 1999. Springer.

[KJWM99c] M. Kieffer, L. Jaulin, E. Walter, et D. Meizel. Robust autonomous robot localization using interval analysis. To appear in *Reliable Computing*, 1999.

[KJWM99d] M. Kieffer, L. Jaulin, E. Walter, et D. Meizel. Guaranteed mobile robot tracking using interval analysis. Dans *Proc. MISC'99 Workshop on Application of Interval Analysis to Systems and Control*, pages 347–359, Girona, February 24-26, 1999.

[Kuh98] W Kuhn. Rigorously computed orbits of dynamical systems without the wrapping effect. *Computing*, 61: 47–67, 1998.

[LÓ98] O. Lévêque. *Méthodes ensemblistes pour la localisation de véhicules*. Thèse de doctorat, Université de Technologie, Compiègne, 1998.

[LDW91] J. J. Leonard et H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Trans. on Robotics and Automation*, 7(3): 376–382, 1991.

[LDW92] J. J. Leonard et H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, Boston, 1992.

[Lho93] Olivier Lhomme. Consistency techniques for numeric CSPs. Dans Wolfgang Wahlster, editeur, *Proceedings of International Joint Conference on Artificial Intelligence*, pages 232–238, Chambéry, France, 1993. Morgan Kaufman.

[LJMW97] O. Lévêque, L. Jaulin, D. Meizel, et E. Walter. Vehicule localization from inaccurate telemetric data: a set inversion approach. Dans *Proc. 5th IFAC Symposium on Robot Control SY.RO.CO.'97*, volume 1, pages 179–186, Nantes, France, 1997.

[Loh87] R. Lohner. *Computer Arithmetic: Scientific Computation and Programming Languages*, chapitre Enclosing the solutions of ordinary initial and boundary value problems, pages 255–286. E. Kaucher and U. Kulisch and Ch. Ullrich, 1987.

[LR97] O. Lhomme et M. Rueher. Application des techniques CSP au raisonnement sur les intervalles. *Revue d'intelligence Artificielle*, 11(3): 283–311, 1997.

[MMTG92] S. A. Malan, M. Milanese, M. Taragna, et J. Garloff. B^3 algorithm for robust performances analysis in presence of mixed parametric and dynamic pertubations. Dans *Proceedings of the 31st IEEE Conference on Decision and Control*, pages 128–133, Tucson, Arizona, 1992.

[MN96] D. Maksarov et J. P. Norton. State bounding with ellipsoidal set description of the uncertainty. *Int. J. of Control*, 65(5): 847–866, 1996.

[MNPLW96] M. Milanese, J. Norton, H. Piet-Lahanier, et E. Walter. *Bounding Approaches to System Identification*. Plenum Press, 1996.

[Moo66] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

[Moo79] R. E. Moore. *Methods and Applications of Interval Analysis*. SIAM Publ., Philadelphia, 1979.

[MR95] P. Marti et M. Rueher. A distributed cooperating constraint solving system. *International Journal of Artificial Intelligence Tools*, pages 93–113, 1995.

[NHTS96] J. Neira, J. Horn, J. D. Tardoz, et G. Schmidt. Multisensor mobile robot localization. Dans *Proc. IEEE International Conference on Robotics and Automation*, pages 673–679, Mineapolis, USA, 1996.

[Nor96] J. P. Norton. Roles for deterministic bounding in environmental modeling. *Ecological Modelling*, 86: 157–161, 1996.

[Oht99] Y. Ohta. Nonconvex polygon interval arithmetic as a tool for the analysis and design of robust control systems. *Reliable Computing (accepté)*, 1999.

[PM96] A. Piazzi et G. Marro. Robust stability using interval analysis. *International Journal of Systems Sciences*, 27: 1381–1390, 1996.

[Pru90] A. Pruski. Multivalue codes: application to autonomous robots. *Robotic and factories of the future*, 1990.

[PV98] A. Piazzi et A. Visioli. Global minimum-time trajectory planning of mechanical manipulators using interval analysis. *International journal of Control*, pages 631–652, 1998.

[PV99] A. Piazzi et A. Visioli. System inversion based control of an overhead crane. Dans *Procedings of CDC99*, 1999.

[RKBM98] F. J. Rivas, S. T. Kolaczkowski, F. J. Beltran, et D. B. Mclurgh. Development of a model for the wet air oxidation of phenol based on a free-radical mechanism. *Chemical Engineering Science*, 53: 2575–2586, 1998.

[Roh96] J. Rohn. Enclosing solutions of overdetermined systems of linear interval equations. *Reliable Computing*, pages 167–171, 1996.

[Sch73] F. C. Schweppe. *Uncertain Dynamic Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[Wal75] D.L. Waltz. Generating semantic descriptions from drawings of scenes with shadows. *The Psychology of Computer Vision*, pages 19–91, 1975.

[Wal90] E. Walter (Ed.). Special issue on parameter identifications with error bounds. *Mathematics and Computers in Simulation*, 32(5&6): 447–607, 1990.

[WJ94] E. Walter et L. Jaulin. Guaranteed characterization of stability domains via set inversion. *IEEE Transactions on Automatic Control*, AC–39(4): 886–889, 1994.

[WPL88] E. Walter et H. Piet-Lahanier. Estimation of the parameter uncertainty resulting from bounded-error data. *Mathematical Biosciences*, 92: 55–74, 1988.

[YW99] J. Yang et W. Wang. Numerical approach to constrained nonlinear generalized predictive control. Dans *IFAC 14th Triennal Word Congress*, pages 13–17, 1999. Beijing, China.

[ZG98] M. Zettler et J. Garloff. Robustness analysis of polynomials with polynomial parameter dependency using bernstein expansion. *IEEE Trans. Automatic Control*, 43: 425–431, 1998.

Notations et conventions

Les vecteurs $\mathbf{v}$ sont imprimés en caractères gras. La $i^{\text{ème}}$ composante du vecteur $\mathbf{v}$ est notée v_i. Les matrices $\mathbf{M}$ sont en majuscules grasses. Les crochets [ ] ne sont jamais utilisés dans le sens de parenthèses, mais sont réservés pour représenter des encadrements. Par exemple, $[x]$ est un intervalle réel et $[\mathbf{x}]$, un intervalle vectoriel ou pavé. Les équations et inégalités vectorielles sont à comprendre composante par composante. Les ensembles $\mathbb{S}$ (autres que les encadrements) sont écrits en caractères doubles. Les approximations ou estimées sont repérées par un chapeau. Par exemple $\widehat{\mathbb{S}}$ est un sous-pavage qui approxime l'ensemble $\mathbb{S}$. Les listes $\mathcal{L}$, piles $\mathcal{P}$, arbres $\mathcal{T}$, graphes $\mathcal{G}$ seront en style calligraphique. Les fonctions sont notées avec les mêmes conventions typographiques que les éléments de leur ensemble d'arrivée. Ainsi $[f](\cdot)$ sera une fonction à valeurs intervalles et $\mathbf{f}(\cdot)$, une fonction à valeurs vectorielles.

$\triangleq$: égal par définition

$:=$: opérateur d'affectation

$\square(\mathbb{A})$: plus petit récipient qui contient $\mathbb{A}$ (voir page 23)

$[\mathbb{A}]$: plus petit pavé qui contient l'ensemble $\mathbb{A}$

$[\in]$: extension trivaluée de $\in$ (voir page 41)

$\#\widehat{\mathbb{X}}$: nombre de pavés appartenant au sous-pavage $\widehat{\mathbb{X}}$

$\backslash$: $\mathbb{A}\backslash\mathbb{B} = \{a \in \mathbb{A} | a \notin \mathbb{B}\}$

$\sqcup$: $[\mathbf{z}] = [\mathbf{x}] \sqcup [\mathbf{y}] \triangleq [[\mathbf{x}] \cup [\mathbf{y}]]$: pavé enveloppe de l'union de deux pavés de $\mathbb{R}^n$

$\rceil a$: opérateur booléen *non*

$\times$: produit cartésien d'ensembles

$\widehat{\mathbb{S}}^-$: sous-pavage qui approxime $\mathbb{S}$ par l'intérieur ($\widehat{\mathbb{S}}^- \subset \mathbb{S}$)

$\widehat{\mathbb{S}}^+$: sous-pavage qui approxime $\mathbb{S}$ par l'extérieur ($\mathbb{S} \subset \widehat{\mathbb{S}}^+$)

$\underline{x}, \bar{x}$: bornes inférieure et supérieure de l'intervalle $[x]$

$\underline{\mathbf{x}}, \bar{\mathbf{x}}$: bornes inférieure et supérieure du pavé $[\mathbf{x}]$

$\mathbb{B}$: ensemble des booléens

$h^0_\infty(\mathbb{A},\mathbb{B})$: proximité de $\mathbb{B}$ à $\mathbb{A}$

$h_\infty(\mathbb{A},\mathbb{B})$: distance de Hausdorff de $\mathbb{A}$ à $\mathbb{B}$

${}^i\mathbf{x} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)^{\mathrm{T}}$

$\mathbb{IR}$: ensemble des intervalles réels

$\mathbb{IB}$: ensemble des intervalles booléens

$\mathbb{IS}$: ensemble des intervalles à bornes dans $\mathbb{S}$

$L[\mathbf{x}]$: fils gauche du pavé $[\mathbf{x}]$

$L\widehat{\mathcal{X}}$: fils gauche du sous-pavage $\widehat{\mathcal{X}}$

$\mathcal{N}_{\mathbb{S}}([\mathbf{x}])$: opérateur de réduction de $[\mathbf{x}]$ relativement à $\mathbb{S}$

$\mathcal{P}(\mathbb{S})$: ensemble des parties de l'ensemble $\mathbb{S}$

$R[\mathbf{x}]$: fils droit du pavé $[\mathbf{x}]$

$R\widehat{\mathcal{X}}$: fils droit du sous-pavage $\widehat{\mathcal{X}}$

$\mathbb{R}$: ensemble des réels

$\mathcal{SPR}([\mathbf{x}])$: ensemble de tous les sous-pavages réguliers de $[\mathbf{x}]$

$w([\mathbf{x}])$: longueur (en anglais *width*) du pavé $[\mathbf{x}]$

Sigles utilisés

HEUDIASYC :	HEUristique et DIAgnostic des SYstèmes Complexes
LAG :	Laboratoire d'Automatique de Grenoble
LISA :	Laboratoire d'Ingénierie et des Systèmes Automatisés
LSS :	Laboratoire des Signaux et Systèmes
POMA :	Propriétés Optiques des Matériaux et Applications

Algorithmes

GOVIA :	Global Optimizer Via Interval Analysis
HULL :	Algorithme de calcul du pavé enveloppe d'un ensemble
IMAGESP :	Calcule l'image d'un sous-pavage par une fonction
INSIDE :	Teste si un pavé est inclu ou est à l'extérieur d'un sous-pavage régulier
INTERSECTION :	Calcule l'intersection entre deux sous-pavages réguliers
MINIMISE :	Calcule le minimum global d'un critère nonlinéaire
SIVIA :	(Set Inverter Via Interval Analysis) calcule l'image réciproque d'un ensemble par une fonction
UNION :	Calcule l'union de deux sous-pavages réguliers

Index