Interval analysis and convex optimization to solve a robust constraint feasibility problem

Benoit Clement

▶ **To cite this version:**

Benoit Clement. Interval analysis and convex optimization to solve a robust constraint feasibility problem. Journal Européen des Systèmes Automatisés (JESA), 2012, 46 (4-5), pp.381-395. 10.3166/jesa.46.381-395 . hal-00733844

HAL Id: hal-00733844

https://ensta-bretagne.hal.science/hal-00733844

Submitted on 3 Apr 2013

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.

Analyse par intervalles et optimisation convexe pour résoudre un problème général de faisabilité d'une contrainte robuste

Benoît Clement

ENSTA Bretagne - Lab-STICC UMR CNRS 6285
2 rue F. Verny, 29806 Brest Cedex 9, France
benoit.clement@ensta-bretagne.fr

RÉSUMÉ. *Cet article traite du problème de faisabilité d'une contrainte robuste (robust constraint feasibility en anglais) qui vise à trouver l'ensemble des θ tels qu'il existe un vecteur $\mathbf{x}$ qui satisfasse la contrainte $f(\mathbf{x},\theta) < 0$. Un algorithme à base d'analyse par intervalles répondrait à la question mais avec une complexité rédhibitoire car exponentiellement croissante avec la dimension du vecteur $(\mathbf{x},\theta)$. Si la fonction de contrainte est supposée convexe en $\mathbf{x}$ à θ fixé, nous montrons que la complexité devient polynomiale par rapport à la dimension du vecteur $\mathbf{x}$ et exponentielle par rapport à la dimension du vecteur θ. L'autre contribution de cet article est de proposer un algorithme qui combine l'optimisation convexe et l'analyse par intervalles pour résoudre le problème avec une complexité réduite. Un exemple numérique simple est donné pour illustrer les concepts.*

ABSTRACT. *This paper deals with the Robust Constraint Feasibility (RCF) problem which aims finding all θ such that there exists a vector $\mathbf{x}$ satisfying the constraint $f(\mathbf{x},\theta) < 0$. An interval based algorithm will answer the question but with a crippling computational complexity which is exponential with respect to the dimension of the vector $(\mathbf{x},\theta)$. If the constraint function is assumed to be convex in $\mathbf{x}$ when θ is fixed, the paper shows that the complexity of the problem becomes polynomial with respect to the dimension of vector $\mathbf{x}$ and exponential with respect to the dimension of vector θ. Another contribution of the paper is to provide an algorithm which combines convex optimization and interval analysis to solve the problem with the reduced complexity. As an illustration, a simple numerical example is given.*

MOTS-CLÉS : *analyse par intervalles, optimisation convexe, faisabilité de contrainte robuste.*

KEYWORDS: *interval analysis, convex optimization, robust constraint feasibility.*

DOI:10.3166/JESA.46.381-395 © 2012 Lavoisier

1. Introduction

Le problème de faisabilité d'une contrainte robuste (*robust constraint feasibility* ou RCF) est la caractérisation de l'ensemble Σ défini par :

$$\Sigma = \{\theta \in \mathbb{R}^p \mid \exists \mathbf{x} \in \mathbb{R}^n,\ f(\mathbf{x}, \theta) < 0\} \tag{1}$$

et dans lequel $\mathbf{x} \in \mathbb{R}^n$ est la variable, f est une fonction de $\mathbb{R}^n \times \mathbb{R}^p$ vers $\mathbb{R}$, et $\theta \in \mathbb{R}^p$ est un vecteur paramètre utilisé pour modéliser les incertitudes ou les variations des données du problème nominal. Dans le cas général, la résolution de ce problème (qui est aussi un problème de projection) ne peut pas se faire sans une complexité algorithmique qui croît exponentiellement avec la taille des paramètres. Nous proposons donc une réduction de la complexité en supposant que la fonction f est convexe selon la variable $\mathbf{x}$. La dépendance en θ n'est, quant à elle, sujette à aucune hypothèse restrictive. Le problème RCF devient alors le problème de faisabilité d'une contrainte convexe robuste (*robust convex constraint feasibility* ou RCCF). L'objectif est alors de combiner deux outils très performants en termes d'analyse numérique :

– l'optimisation convexe qui fournit des algorithmes efficaces pour résoudre les problèmes convexes dans un temps polynomial pour une précision donnée (voir (Toh *et al.*, 1999) et (Löfberg, 2001)). Ces outils et ces concepts sont très largement développés dans (Ben-Tal, Nemirovskii, 2001) et (Boyd, Vanbenberghe, 2004) ; ils ne sont donc pas repris ici.

– l'analyse par intervalles qui fournit des outils efficaces pour répondre au problème RCF dans sa globalité mais avec une complexité de calcul rédhibitoire si la taille du problème est élevée. La présentation de ces outils et de leur utilisation est développée dans (Hansen, Walster, 2004) et (Jaulin *et al.*, 2001) ; elle n'est donc pas reprise ici.

Seules les propriétés principales dont nous avons besoin pour mener notre analyse sont rappelées ici. Afin de profiter pleinement des avantages proposés par chacun des domaines, il est proposé de les combiner pour calculer efficacement une partition de $\mathbb{R}^p = \mathcal{S}_{in} \cup \mathcal{S}_{out} \cup \mathcal{S}_u$ telle que :

– les ensembles $\mathcal{S}_{xx}$ sont disjoints et sont représentés par des boîtes (intervalles vectoriels) $[\theta]$ disjointes,

– les inclusions suivantes sont vérifiées,

$$\begin{aligned} &\mathcal{S}_{in} \subset \Sigma \\ &\mathcal{S}_{out} \cap \Sigma = \emptyset \end{aligned} \tag{2}$$

– pour un intervalle $[\theta]$ suffisamment petit par rapport à une précision donnée et dont on ne peut déterminer si il est *inclus* ou *exclus* de l'ensemble des solutions, $[\theta] \in \mathcal{S}_u$.

On génère donc les ensembles des paramètres *faisables*, *infaisables* et *indéterminés*.

L'algorithme qui permet d'aboutir à cette partition calcule la solution du problème non convexe sans oublier de solution ; cette propriété fondamentale est intrinsèque à l'utilisation de l'analyse par intervalle qui détermine une partition de l'ensemble de définition de manière constructive. À notre connaissance, l'analyse par intervalles a été utilisée en optimisation convexe pour obtenir des résultats garantis dans (Jansson, 2004), mais jamais pour résoudre le problème RCF même s'il est mentionné dans (Ghaoui, Niculescu, 2000) qui fait le tour d'horizon sur l'utilisation des inégalités matricielles linéaires (LMI) en automatique ; ce problème est mentionné comme piste de recherche à approfondir.

REMARQUE. — Une première ébauche de cette idée a été présentée lors du workshop SWIM2010 (Clement, 2010). □

1.1. Objectif

L'objectif de cet article est de montrer que la complexité du problème RCCF devient polynomiale par rapport à la dimension du vecteur $\mathbf{x}$ et reste exponentielle par rapport à la dimension du vecteur θ. Cette démonstration est fondée sur un algorithme constructif pour le calcul de l'ensemble des solutions Σ défini dans l'équation (1). La contribution principale est issue de l'utilisation couplée de l'optimisation convexe et de l'analyse par intervalles pour être assuré de trouver une solution au problème RCCF :

– incluant toutes les solutions,

– sans relaxation des contraintes,

– sans conservatisme par rapport aux incertitudes.

Afin de présenter la méthodologie, on se limite ici au cas particulier d'une fonction f qui est deux fois différenciable. Cette classe de problème est certes restrictive mais elle simplifie l'exposé de la méthode et les pistes de généralisation, comme l'utilisation des sous-différentielles, sont abordées à la fin de l'article.

1.2. Etat de l'art

Il ne s'agit pas ici d'une revue complète de la littérature mais plutôt de donner les références principales pour le lecteur qui souhaite approfondir un sujet. Il s'agit aussi de situer les problèmes posés dans leur contexte.

Le problème de l'optimisation robuste est d'abord apparu dans le domaine de l'automatique sous la forme d'un problème de décision robuste. En fait, dans les manuels d'automatique, la robustesse est la raison principale de l'utilisation des systèmes bouclés. La notion de commande robuste est quant à elle liée à une commande qui tiendrait compte de la connaissance a priori des incertitudes de modèle et qui quantifie la distance à l'instabilité ; on peut se référer à l'ouvrage complet (Zhou *et al.*, 1995) sur la commande robuste. Dans d'autres domaines de l'ingénierie, les incertitudes sont généralement modélisées par des aspects stochastiques et ceci ne relève pas de la pro-

blématique exposée ici. A la fin des années 1990, Nemirovskii et Ben Tal ont étudié la complexité des problèmes robustes et une synthèse très complète est donnée dans le livre récent (Ben-Tal *et al.*, 2009).

D'un autre côté, l'analyse par intervalles est bien connue pour être un outil puissant pour l'optimisation globale comme décrit dans (Hansen, Walster, 2004) et les références incluses. Dans le cadre de l'optimisation convexe, la plupart du temps, les incertitudes sont gérées avec une approche pire-cas, une hypothèse polytopique, la théorie des jeux... Mais à notre connaissance aucune des démarches ne mélange l'analyse par intervalles et l'optimisation convexe. C'est donc la première motivation de ce travail.

1.3. Organisation

Après quelques rappels généraux concernant l'optimisation convexe et l'analyse par intervalles dans la section 2, nous définissons la méthode de Newton par intervalles paramétrée et nous l'utilisons pour caractériser l'ensemble de tous les minimiseurs de la fonction f. Il s'agit de l'outil principal utilisé dans ce papier ; son utilisation pour une fonction convexe est une des contributions de notre travail. On présente ensuite l'algorithme de projection qui permet de calculer de manière efficace les ensembles $\mathcal{S}_{in}$, $\mathcal{S}_{out}$ dans la section 3. Chaque étape de l'algorithme est ensuite discutée pour souligner les difficultés et montrer comment elles sont abordées. On termine par un exemple numérique très simple dans la section 4.

2. Outils et méthodes

Cette section est dédiée à la description de l'analyse par intervalles avec toutes les notations et les définitions nécessaires. En raison des aspects très généraux utilisés en optimisation convexe, on se limite au rappel des propriétés fondamentales.

2.1. Optimisation Convexe

Une description très complète du problème d'optimisation robuste est donnée dans l'ouvrage (Ben-Tal *et al.*, 2009) dans lequel on présente toute la difficulté à poser le problème de façon à pouvoir le résoudre de manière informatique. Il est bien connu que les problèmes de programmation conique ou *conic programming* (CP), de programmation quadratique ou *quadratic programming* (QP) et plus particulièrement de pragrammation semi-définie ou *semidefinite programming* (SDP) sont capables de formaliser un grand nombre de problèmes. On peut dire que la formulation SDP englobe pratiquement tous les problèmes convexes que l'on rencontre dans les applications en automatique comme cela est mentionné dans les trois ouvrages de référence (Ben-Tal, Nemirovskii, 2001), (Boyd *et al.*, 1994) et (Boyd, Vanbenberghe, 2004). Même si cet article ne traite que des concepts généraux de l'optimisation convexe, nous gardons à l'esprit que l'approche peut être appliquée concrètement par sous-problème de l'op-

timisation comme l'optimisation SDP. Il a été démontré que les méthodes de points intérieurs pour la programmation linéaire sont généralisables à tous les problèmes d'optimisation convexe (voir (Nesterov, Nemirovskii, 1994) et (Boyd, Vanbenberghe, 2004) pour plus de détails). En même temps, les auteurs proposent un cadre conceptuel simple et une méthodologie pour l'utilisation des méthodes de point intérieur pour de nombreux problèmes d'optimisation convexe. Quant à l'optimisation robuste (RO), même si elle est relativement nouvelle dans le domaine de l'optimisation pour les problèmes contenant des incertitudes, elle a déjà démontré son efficacité dans des applications concrètes difficiles à appréhender dans un cadre classique. Le livre (Ben-Tal *et al.*, 2009) développe de manière très compréhensible tout ce sujet. La notion de robustesse, commune en théorie de la commande, est relativement nouvelle en programmation mathématique et ne doit pas être confondue avec l'analyse de sensibilité. Les modèles d'optimisation robuste utilisés en programmation mathématique ont retenu l'attention de la communauté scientifique ces dernières années comme expliqué dans (Ben-Tal *et al.*, 2009). L'approche présentée ici se limite au problème de faisabilité d'une contrainte robuste (RCF).

2.1.1. Définition

On trouvera ici les principales définitions et propriétés pour les fonctions convexes. Un très complet panorama sur le sujet est donné dans le livre (Boyd, Vanbenberghe, 2004). On note $dom_\theta(f)$ le domaine de définition de θ et $dom_{\mathbf{x}}(f,\theta)$ le domaine de définition de f pour $\mathbf{x}$ pour un $\theta \in dom_\theta(f)$ donné.

DÉFINITION 1. — *La fonction* $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ *est convexe en* $\mathbf{x}$ *si :*

– $\forall \theta \in dom_\theta(f)$*, le domaine* $dom_{\mathbf{x}}(f,\theta)$ *est un ensemble convexe*

– *pour tout* $\mathbf{x}_1, \mathbf{x}_2 \in dom_{\mathbf{x}}(f,\theta)$*, et* $\alpha \in [0,1]$*, on a*

$$f(\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2, \theta) \leq \alpha f(\mathbf{x}_1,\theta) + (1-\alpha) f(\mathbf{x}_2,\theta) \tag{3}$$

2.1.2. Propriétés

L'analyse des fonctions convexes est un domaine relativement ancien qui a été très largement étudié ; nous donnons ici quelques propriétés fondamentales qui montrent l'intérêt de la convexité :

– f est continue sur l'intérieur de son domaine ; il ne peut y avoir discontinuité que sur les bornes ;

– si f est differentiable, alors pour tout $\mathbf{x}_1, \mathbf{x}_2 \in dom_{\mathbf{x}}(f,\theta)$,

$$f(\mathbf{x}_1,\theta) + \nabla_{\mathbf{x}} f(\mathbf{x}_1,\theta)^T(\mathbf{x}_2 - \mathbf{x}_1) \leq f(\mathbf{x}_2,\theta) \tag{4}$$

– si f est deux fois différentiable, le Hessien ou dérivée seconde $\nabla^2_{\mathbf{x}} f$ existe et il est semi-défini positif : pour tout $\mathbf{x} \in dom_{\mathbf{x}}(f,\theta)$, $\nabla^2_{\mathbf{x}} f(\mathbf{x},\theta) \succeq 0$;

– pour le problème $\min_{\mathbf{x}} f(\mathbf{x},\theta)$, une condition (nécessaire et suffisante) pour être optimal en $\mathbf{x}_0$ est $\nabla_{\mathbf{x}} f(\mathbf{x}_0,\theta) = 0$.

La plupart du temps, f n'est pas connue de manière explicite mais $f(\mathbf{x}, \theta)$ peut être évaluée (ainsi que ses dérivées) en chacun des points du domaine. Cette évaluation fait référence à un *oracle* (ou subroutine) et doit être associée à la fonction de coût de calcul. Plus l'oracle est efficace, plus l'algorithme d'optimisation convexe tourne rapidement. Il s'agit d'une raison majeure pour laquelle chaque solveur d'optimisation est dédié à la sous-classe spécifique d'optimisation convexe. Par exemple, la programmation semi-définie possède des oracles simples et efficaces mais n'est pas différentiable dans le cas général. Cette dernière remarque justifie également que l'on se limite ici aux hypothèses suivantes sur la fonction f :

– f est convexe en $\mathbf{x}$

– f est deux fois différentiable selon $\mathbf{x}$.

REMARQUE. — Les sous-problèmes de l'optimisation convexe comme SDP méritent une attention particulière pour améliorer l'algorithme que nous proposons dans cet article. □

2.2. Analyse par intervalles

Parallèlement à l'optimisation convexe, cette section est dédiée à une présentation succincte des outils de l'analyse par intervalles. Pour plus de détails nous renvoyons aux ouvrages de référence (Neumaier, 1990) et (Moore, 1966). Une section décrit l'opérateur de Newton par intervalles paramétré qui est utilisé pour démontrer l'existence et l'unicité du zéro d'une fonction. Nous présentons aussi un outil permettant de décrire l'ensemble des minimiseurs d'une fonction convexe f.

2.2.1. Définitions

DÉFINITION 2. — *Un vecteur intervalle ou une boîte* $[\mathbf{x}]$ *de* $\mathbb{R}^n$ *est définie par*

$$[\mathbf{x}] = [\underline{\mathbf{x}}, \overline{\mathbf{x}}] = \{\mathbf{x} \in \mathbb{R}^n \ : \underline{\mathbf{x}} \leq \mathbf{x} \leq \overline{\mathbf{x}}\} \tag{5}$$

où $\underline{\mathbf{x}}$ *et* $\overline{\mathbf{x}}$ *sont deux éléments de* $\mathbb{R}^n$.

L'ordre partiel doit être compris élément par élément. L'ensemble de toutes les boîtes bornées de $\mathbb{R}^n$ est noté $\mathbb{IR}^n$. La bissection d'une boîte $[\mathbf{x}]$ signifie la couper le long d'un plan de symétrie normal à la face de taille maximale. La taille de cette face est la dimension de $[\mathbf{x}]$ notée $w([\mathbf{x}])$. La bissection de $[\mathbf{x}]$ génère deux boîtes disjointes $[\mathbf{x}_1]$ et $[\mathbf{x}_2]$ telles que $[\mathbf{x}] = [\mathbf{x}_1] \cup [\mathbf{x}_2]$ (le cas scalaire correspond à la dichotomie).

DÉFINITION 3. — *Soit* f *une fonction de* $\mathbb{R}^n$ *vers* $\mathbb{R}^m$. *La fonction intervalle* $[f]$ *de* $\mathbb{IR}^n$ *vers* $\mathbb{IR}^m$, *est une* fonction d'inclusion *de* f *si*

$$\forall [\mathbf{x}] \in \mathbb{IR}^n, \ f([\mathbf{x}]) \subset [f]([\mathbf{x}]). \tag{6}$$

De plus, la fonction d'inclusion $[f]$ *est :*

– monotone, *si* $([\mathbf{x}] \subset [y]) \Rightarrow ([f]([\mathbf{x}]) \subset [f]([y]))$

– minimale, *si* $\forall [\mathbf{x}] \in \mathbb{IR}^n, \ [f]([\mathbf{x}]) = [f([\mathbf{x}])]$

– convergente, *si* $\lim_{k\to\infty} w([\mathbf{x}](k)) = 0 \Rightarrow \lim_{k\to\infty} w([f]([\mathbf{x}](k))) = 0$ *; avec* $[\mathbf{x}](k)$ *une suite d'intervalles.*

Dans (Neumaier, 1990), il est démontré qu'il est toujours possible de trouver une fonction d'inclusion convergente $[f]$ quand la fonction f est continue et définie par une expression arithmétique. Notons qu'en général, $f([\mathbf{x}])$ n'est pas une boîte contrairement à $[f]([\mathbf{x}])$. De plus, puisque $[f([\mathbf{x}])]$ est la boîte enveloppe de $f([\mathbf{x}])$ (la plus petite boîte qui contienne $f([\mathbf{x}])$), les inclusions suivantes sont vérifiées :

$$f([\mathbf{x}]) \subset [f([\mathbf{x}])] \subset [f]([\mathbf{x}]). \tag{7}$$

Une illustration de ces inclusions est donnée sur la figure 1.

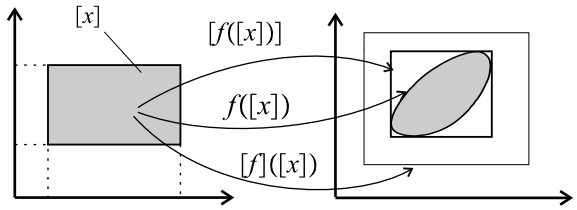

Figure 1. Fonction d'inclusion $[f]$ *pour* f

2.2.2. Opérateur de Newton par intervalles paramétré

Nous rappellons d'abord la définition de l'opérateur de Newton par intervalles noté $\mathcal{N}$ et ensuite sa généralisation au cas paramétré. Cet opérateur permet de garantir qu'une boîte $[\mathbf{x}]$ contient tous les minimiseurs de $f(\mathbf{x}, \theta)$ pour la boîte $[\theta]$. Ce résultat est ensuite utilisé pour créer la partition de Σ l'ensemble des solutions du problème présenté section 1.

REMARQUE. — L'opérateur de Newton est dans un premier temps appliqué à une fonction g quelconque, puis on définit la fonction g comme le gradient de la fonction f pour caractériser les minimiseurs. □

DÉFINITION 4. — *Soit* $g : \mathbb{R}^n \to \mathbb{R}^m$*, une fonction différentiable,* $D_{\mathbf{x}}g(\mathbf{x})$ *sa matrice jacobienne et* $[\mathbf{x}]$ *une boîte de* $\mathbb{IR}^n$*. Si* $[D] = [D_{\mathbf{x}}g([\mathbf{x}])]$ *est l'enveloppe de* $D_{\mathbf{x}}g([\mathbf{x}])$*, et* $\hat{\mathbf{x}} = mid([\mathbf{x}])$ *(le centre de l'intervalle), l'opérateur de Newton par intervalles noté* $\mathcal{N}$ *est défini par :*

$$\mathcal{N}(g, [\mathbf{x}]) = \hat{\mathbf{x}} - Inv([D], g(\mathbf{x})) \tag{8}$$

où $Inv([\mathbf{A}], [\mathbf{b}])$ *est un vecteur intervalle qui contient l'ensemble* $\{\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} : \mathbf{A} \in [\mathbf{A}], \mathbf{b} \in [\mathbf{b}]\}$

THÉORÈME 5. — *Si* $\mathcal{N}(g, [\mathbf{x}]) \subset [\mathbf{x}]$*, alors il existe un unique* $\mathbf{x}_0 \in [\mathbf{x}]$ *tel que* $g(\mathbf{x}_0) = 0$*.*

La démonstration est disponible dans (Hansen, Walster, 2004) et un exemple dans le cas scalaire est illustré par la figure 2. L'opérateur de Newton, $\mathcal{N}(g, [x]) = \hat{x} -$

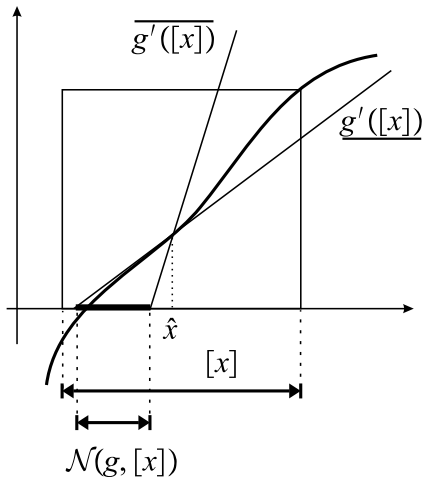

Figure 2. Cas scalaire de l'opérateur de Newton par intervalles

$g(\hat{x})/[g'([x])]$ correspond à l'intersection entre le cone de toutes les dérivées de g sur le boîte $[\mathbf{x}]$ et l'axe $\mathbf{x}$.

Cet opérateur peut être généralisé à une fonction $g(\mathbf{x},\theta)$ de $\mathbb{R}^n \times \mathbb{R}^p$ et son expression devient :

$$\mathcal{N}(g,[\mathbf{x}],[\theta]) = \hat{\mathbf{x}} - Inv([D_{\mathbf{x}}g([\mathbf{x}],[\theta])], g(\hat{\mathbf{x}},[\theta])) \tag{9}$$

Le théorème 5 est étendu au résultat suivant que l'on appelle théorème de l'opérateur de Newton paramétré.

THÉORÈME 6. — *Si* $\mathcal{N}(g,[\mathbf{x}],[\theta]) \subset [\mathbf{x}]$, *alors* $\forall\theta \in [\theta]$, *il existe un unique* $\mathbf{x}_0 \in [\mathbf{x}]$ *tel que* $g(\mathbf{x}_0,\theta) = 0$

Ce théorème peut maintenant être utilisé pour caractériser l'ensemble des minimiseurs d'une fonction. Pour cela on utilise la fonction f et ses propriétés de différentiabilité.

2.2.3. Caractérisation de l'ensemble des minimiseurs

Soit $\hat{X}_\theta$ l'ensemble des minimiseurs de la fonction f pour un vecteur θ donné :

$$\hat{X}_\theta = \arg\min_{\mathbf{x}} f(\mathbf{x},\theta) \tag{10}$$

L'ensemble des minimiseurs de f est $\hat{X}_{[\theta]}$ défini par :

$$\hat{X}_{[\theta]} = \bigcup_{\theta\in[\theta]} \hat{X}_\theta \tag{11}$$

REMARQUE. — Comme la fonction f est par hypothèse convexe selon $\mathbf{x}$, $\hat{X}_\theta$ est un singleton si la convexité est stricte ou un intervalle. □

Le gradient de la fonction f selon $\mathbf{x}$ est noté $g(\mathbf{x},\theta) = \left(\frac{\partial f(\mathbf{x},\theta)}{\partial \mathbf{x}_i}\right)_{i=1..n}$

THÉORÈME 7. — *Si* $\mathcal{N}(g,[\mathbf{x}],[\theta]) \subset [\mathbf{x}]$ *alors* $\hat{X}_{[\theta]} \subset [\mathbf{x}]$.

PREUVE. — La démonstration découle de la propriété d'unicité du résultat de l'opérateur de Newton combinée à la convexité de f. En effet, d'après le théorème 6, $\mathcal{N}(g,[\mathbf{x}],[\theta]) \subset [\mathbf{x}]$ implique : $\forall\theta \in [\theta], \exists \mathbf{x} \in [\mathbf{x}]$, unique, tel que $g(\mathbf{x},\theta) = 0$. Ceci signifie aussi que $\forall\theta \in [\theta], \exists \mathbf{x} \in [\mathbf{x}]$, unique et solution de $\min_{\mathbf{x}} f(\mathbf{x},\theta)$. Comme la fonction f est convexe, alors $\forall\theta \in [\theta]$, les solutions de $\min_{\mathbf{x}} f(\mathbf{x},\theta)$ sont à l'intérieur de $[\mathbf{x}]$. Par définition de $\hat{X}_{[\theta]}$, cela signifie exactement que $\hat{X}_{[\theta]} \subset [\mathbf{x}]$. ■

REMARQUE. — Si la fonction f n'est pas strictement convexe, l'inclusion $\mathcal{N}(g,[\mathbf{x}],[\theta]) \subset [\mathbf{x}]$ ne peut pas être vérifiée pour $[\mathbf{x}]$ borné. On peut ainsi limiter les solutions de $\min_{\mathbf{x}} f(\mathbf{x},\theta)$ à *la* solution. □

Avec ce dernier théorème, nous avons à disposition un outil constructif pour vérifer si une boîte contient l'ensemble $\hat{X}_{[\theta]}$ avec l'opérateur de Newton par intervalles paramétré. Ce résultat est à la base de l'algorithme que nous présentons ci-après.

3. Algorithme de projection

Cette section est dédiée à la présentation de l'algorithme principal qui utilise conjointement l'analyse par intervalles et l'optimisation convexe. Les concepts présentés précédemment sont ensuite utilisés pour montrer comment est réduite la complexité et pour justifier les différentes étapes de la méthode.

3.1. L'algorithme SCONVEX

L'algorithme SCONVEX peut être considéré comme une extension naturelle de l'algorithme SIVIA proposé dans (Jaulin *et al.*, 2001). L'entrée est une boîte initiale $[\theta]_0$ et les sorties sont les ensembles S_{in}, S_{out} et S_u définis par (3).

SCONVEX(in : $[\theta]_0$, out : $\mathcal{S}_{in}, \mathcal{S}_{out}, \mathcal{S}_u$)	
1	$\mathcal{L} := [\theta]_0$
2	while $\mathcal{L} \neq \emptyset$, do
3	$[\theta] = \mathbf{pull}(\mathcal{L}), \hat{\theta} = \mathbf{mid}([\theta])$
4	$(\hat{\mathbf{x}}, \delta) =$CONVEX$(f(\mathbf{x},\hat{\theta}))$
5	if δ, then $in =$TESTIN$(f,\hat{\mathbf{x}},[\theta])$
6	else $out =$TESTOUT$(f,\hat{\mathbf{x}},[\theta])$
7	if in, then $\mathcal{S}_{in} = \mathcal{S}_{in} \cup \{[\theta]\}$
8	elseif out, $\mathcal{S}_{out} = \mathcal{S}_{out} \cup \{[\theta]\}$
9	elseif $w([\theta]) < \varepsilon$, $S_u = S_u \cup \{[\theta]\}$
10	else $\mathbf{push}(\mathcal{L}, \mathbf{split}([\theta]))$
11	Return $\mathcal{S}_{in}, \mathcal{S}_{out}, \mathcal{S}_u$

Les étapes 1, 2, 3 et 10 font la gestion des listes de solutions, tandis que les étapes 4, 5 et 6 s'attachent à l'optimisation convexe et la vérification des contraintes. Les étapes 7, 8 et 9 construisent les trois ensembles $\mathcal{S}_{in}$, $\mathcal{S}_{out}$ et $\mathcal{S}_u$ définis par la relation 3. Les sections suivantes décrivent ces étapes.

3.2. Gestion des listes et complexité

On note $\mathcal{L}$ la liste des boîtes à tester. Chaque fois qu'une boîte $[\theta]$ ne peut pas être classée dans un ensemble et qu'elle est assez "grande", la boîte est divisée en deux boîtes (fonction $\mathbf{split}$). Ceci peut être fait simplement par bissection mais une quelconque infomation a priori permettant d'améliorer le processus peut être utilisée. Les fonctions $\mathbf{pull}$ et $\mathbf{push}$ permettent respectivement de retirer et d'ajouter des éléments de la liste $\mathcal{L}$. La complexité de l'algorithme est liée à :

– La bissection de la boîte $[\theta]$; la complexité de cette étape est donc exponentielle en fonction de la taille de θ ;

– l'optimisation convexe selon la variable $\mathbf{x}$; la compléxité de cette étape est polynomiale en fonction de la taille de $\mathbf{x}$.

Il reste donc à démontrer que la complexité algorithmique des autres fonctions incluses dans l'algorithme est aussi polynomiale. Ceci montrera la complexité globale de la méthode.

3.3. Optimisation convexe et vérification des contraintes

L'étape 4 (CONVEX) résoud le problème convexe non paramétré :

$$\min_{\mathbf{x}} f(\mathbf{x}, \hat{\theta}) \tag{12}$$

où $\hat{\theta}$ est le centre de $[\theta]$ et $\mathbf{x} \in [\mathbf{x}] \subset \mathbb{R}^n$. La façon dont on résoud ce problème n'entre pas dans le périmètre de notre article. La section 2.1 rappelle les principales références pour savoir comment résoudre ce problème. La seule condition nécessaire est qu'il existe un oracle efficace pour f qui procure à la fois $f(\mathbf{x}, \theta)$ et sa jacobienne. Cette étape est considérée comme une boîte noire qui nous donne en sortie $\mathbf{x}$ et $\hat{\lambda}$ solutions de

$$\hat{\lambda} = \min f(\mathbf{x}, \hat{\theta}) = f(\mathbf{x}, \hat{\theta}) \tag{13}$$

La variable δ est un indicateur booléen pour savoir si la contrainte est réalisée ou pas.

En fonction de $\hat{\lambda}$, le problème n'est pas de même nature et il est plus ou moins difficile :

– si $\hat{\lambda} < 0$, i.e. $\delta = TRUE$, alors $\hat{\theta} \in S_{in}$, il est nécessaire de tester si $[\theta]$ est élément de S_{in} par évaluation. La section 3.4 est dédiée à ce test.

– si $\hat{\lambda} \geq 0$, i.e. $\delta = FALSE$, alors $\hat{\theta} \in S_{out}$. Tester si $[\theta] \in S_{out}$ est plus complexe que pour S_{in} comme cela est expliqué dans 3.5.

3.4. Tester si un intervalle est faisable

L'étape 5 (TESTIN) vérifie si l'intervalle $[\theta]$ est dans l'ensemble des solutions au problème S_{in}. Une condition nécessaire pour tester si $[\theta] \in S_{in}$, est qu'une fonction d'inclusion convergente $[f]$ de la fonction f dans la boîte $[\theta]$ existe ($[f] \supset f(\mathbf{x}, [\theta])$).

REMARQUE. — Si la fonction f n'est pas construite avec des opérateurs de base, il est nécessaire de calculer l'image $[f]$ et le problème qui consiste à obtenir une approximation intérieur pour l'ensemble image est un problème ouvert si la fonction n'est pas inversible ; pour plus de détails voir (Jaulin *et al.*, 2001). □

Dans le cas général, la propriété d'évaluation de la fonction en $\mathbf{x}$ et θ pour f est nécessaire ; alors le test devient évident.

Résumé : tester la condition $[\theta] \in S_{in}$ est possible avec une complexité polynomiale et nécessite :

– l'existence d'une fonction d'inclusion convergente $[f] \supset f(\hat{\mathbf{x}}, [\theta])$ qui peut être calculée,

– de tester si $[f] < 0$.

3.5. Tester la non faisabililité d'un intervalle

L'étape 6 (TESTOUT) teste si l'intervalle $[\theta]$ est dans l'ensemble S_{out}. En utilisant les notations présentées dans la section 2.2.3, nous souhaitons vérifier ici si :

$$\forall \mathbf{x} \in \hat{X}_{[\theta]}, \forall \theta \in [\theta] \ : f(\mathbf{x}, \theta) \geq 0. \tag{14}$$

Le point difficile ici est que $\hat{X}_{[\theta]}$ n'est pas calculable exactement. En effet, si c'était le cas, le problème serait déjà résolu. Nous proposons donc de trouver une boîte $[\mathbf{x}]_\varepsilon$ telle que $\hat{X}_{[\theta]} \subset [\mathbf{x}]_\varepsilon$ par ε-inflation pour $\hat{\mathbf{x}}$. Alors pour une fonction d'inclusion $[f]$ pour la fonction f, telle que $[f] \supset f([\mathbf{x}]_\varepsilon, [\theta])$, la relation d'ordre $[f] \geq 0$ est une condition suffisante pour garantir $[\theta] \in S_{out}$.

Si on suppose de plus que la fonction f est deux fois différenciable et on note $g(\mathbf{x}, \theta)$ le gradient associé :

$$g(\mathbf{x}, \theta) = \left(\frac{\partial f(\mathbf{x}, \theta)}{\partial \mathbf{x}_i} \right)_{i=1..n} \tag{15}$$

En utilisant le théorème 7 associé à l'ε-inflation, on obtient un test efficace.

L'ε-inflation crée une boîte $[\mathbf{x}]_\varepsilon$ autour de $\hat{\mathbf{x}}$ qui est supposée inclure $\hat{X}_{[\theta]}$. Le choix pour ε n'est pas évident pour assurer la convergence l'algorithme. Il est nécessaire que $\varepsilon \to 0$ quand $\theta \to 0$. Des choix possibles sont :

$$\varepsilon = \sqrt{w([\theta])} \text{ ou } \varepsilon = w([\theta]) \tag{16}$$

Une fois déterminée la boîte $[\mathbf{x}]_\varepsilon$, L'opérateur de Newton généralisé est calculé pour tester si $[\mathbf{x}]_\varepsilon$ contient l'ensemble des minimiseurs. Si $\mathcal{N}(g, [\mathbf{x}]_\varepsilon, [\theta]) \subset [\mathbf{x}]_\varepsilon$ et $f([\mathbf{x}]_\varepsilon, [\theta]) \geq 0$, alors $[\theta] \in S_{out}$.

Résumé : tester $[\theta] \in S_{out}$ a une complexité polynomiale et nécessite :

– l'existence d'une fonction d'inclusion convergente $[f] \supset f([\mathbf{x}], [\theta])$ qui peut être évaluée,

– que la fonction f soit deux fois différentiable,

– de tester si $\mathcal{N}(g, [\mathbf{x}]_\varepsilon, [\theta]) \subset [\mathbf{x}]_\varepsilon$ avec $\varepsilon = \sqrt{w([\theta])}$,

– de tester si $[f] \geq 0$.

3.6. Discussion

Chaque étape de 3 à 10 de SCONVEX a une complexité polynomiale et peut être calculée numériquement si la fonction f a :

– un oracle efficace pour évaluer f, $\nabla_{\mathbf{x}} f$ et $\nabla^2_{\mathbf{x}} f$;

– une fonction d'inclusion convergente qui peut être calculée.

Même s'il existe évidemment de nombreux problèmes qui n'appartiennent pas à cette classe, chaque étape peut être améliorée (pour ne pas dire optimisée) avec l'utilisation par exemple des sous-gradients ou d'opérateurs plus performant que l'opérateur de Newton. Notre approche ouvre donc les portes à un large champ de recherche en utilisant les propriétés intrinsèques du problème que l'on cherche à résoudre. Par exemple les formulation par LMI ont une fonction d'inclusion naturelle car le problème devient linéaire en $\mathbf{x}$, néanmoins la compléxité vient de la différenciation qui n'est pas possible sur tout le domaine ; des recherches complémentaires doivent être menées concernant les fonctions non différenciables. Un dernier point difficile est le choix de l'ε-inflation qui dépend du prolème à résoudre. La classique inflation quadradique que nous avons utilisée donne des résultats satisfaisants mais une preuve formelle de la convergence doit encore être faite.

4. Exemple numérique

Un exemple numérique très simple est donné ici pour illuster la méthode. Il s'agit plus ici de montrer comment chaque étape de l'algorithme est mise en oeuvre plutôt que de montrer l'efficacité numérique de l'approche. Ainsi l'exemple est très simple, la solution peut être interprétée graphiquement. Considérons la fonction f suivante :

$$f(x, \theta) = x^2 + (\sin(-\theta_1 \theta_2 \pi) + 1) e^{-\theta_2 x} - 0.2 \tag{17}$$

Il est facile de remarquer dans un premier temps qu'il n'est pas possible de résoudre le problème avec l'optimisation convexe seule mais qu'une partie du problème est convexe. Nous avons donc bien une fonction qui répond aux critères de choix de

notre méthode. Comme f est une fonction analytique de $x \in \mathbb{R}$, les matrices jacobienne et hessienne ne sont pas calculées par des oracles mais comme des fonctions analytiques. La contrainte à vérifier est $f(x, \theta_1, \theta_2) < 0$ avec

$$x \in [-1, 1]$$
$$\theta_1 \in [0, 2.5]$$
$$\theta_2 \in [0, 2.5]$$

La matrice jacobienne et la matrice hessienne sont des fonctions scalaires :

$$\nabla_x f(x, \theta) = 2x - (\sin(-\theta_1\theta_2\pi) + 1)\theta_2 e^{-\theta_2 x} \quad (18)$$
$$\nabla_x^2 f(x, \theta) = 2 + (\sin(-\theta_1\theta_2\pi) + 1)\theta_2^2 e^{-\theta_2 x}$$

Notons que $\nabla_x^2 f(x, \theta) > 0$ montre que f est convexe en x et non convexe en (θ_1, θ_2). Une ε-inflation linéaire est choisie. Pour $x = 0$, le figure 3 donne une représentation 3D pour $f(x = 0, \theta)$.

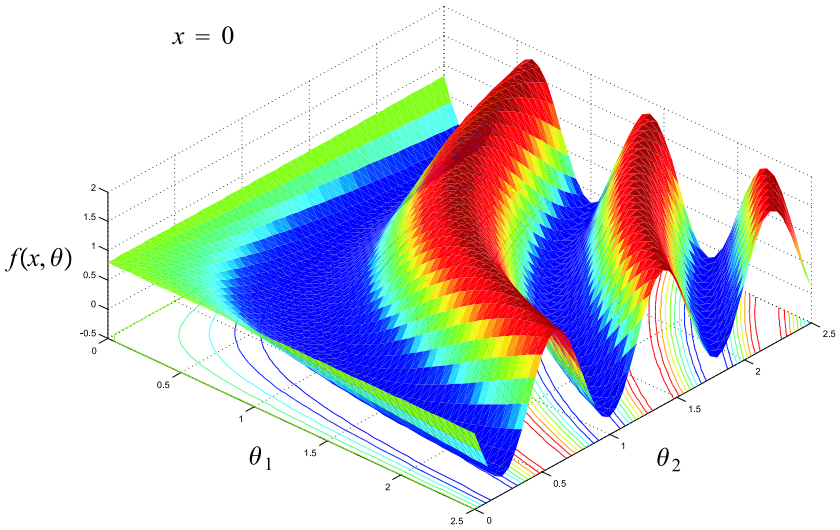

Figure 3. $f(x = 0, \theta)$

L'ensemble faisable avec une précision de 0.01 est représenté par la figure 4.

5. Conclusion et perspectives

Cet article propose un algorithme qui combine l'analyse par intervalles et l'optimisation convexe pour résoudre le problème de faisabilité robuste d'une contrainte convexe (RCCF) dans le cas d'une fonction lisse. Cet algorithme est fondé sur le théorème 5 qui donne une méthode constructive pour caractériser l'ensemble des minimiseurs d'une fonction convexe paramétrée. On profite de la convexité pour réduire la

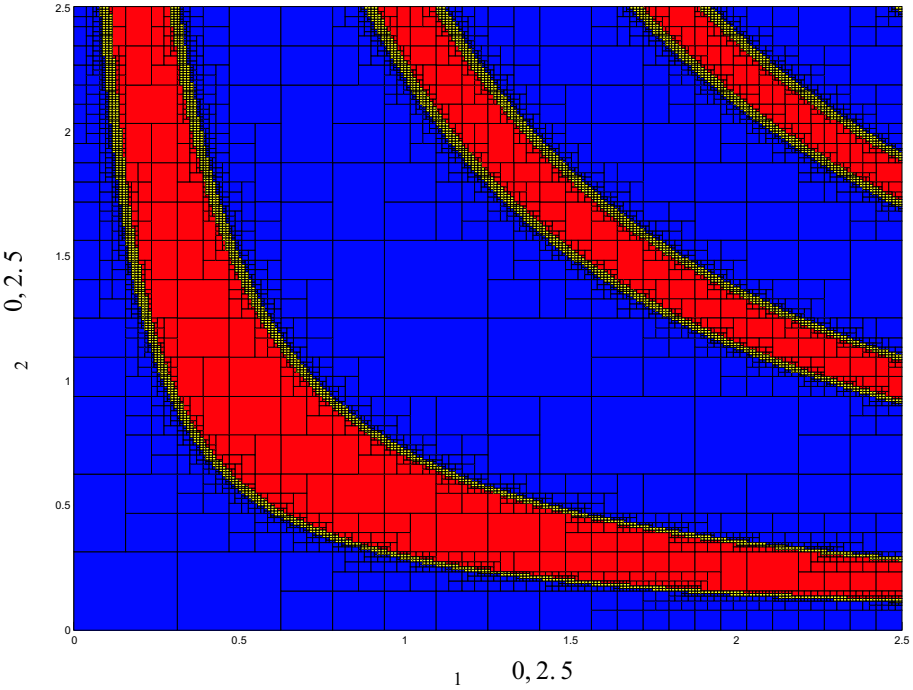

Figure 4. Partition des ensembles

complexité algorithmique du problème RCF, ainsi que des avantages de l'analyse par intervalles pour garantir la résolution des aspects non convexes du problème. L'analyse par intervalle garantie aussi que toutes les solutions sont explorées. Comme mentionné dans la section 2.1, l'efficacité d'un algorithme d'optimisation convexe est liée à l'efficacité de ses oracles, ainsi pour assurer l'efficacité globale de l'agorithme, il est nécessaire de particulariser en sous-classe de problèmes convexes. Comme le problème SDP englobe quasiment tous les problèmes convexes dans les applications de l'automatique ((Ben-Tal, Nemirovskii, 2001; Boyd *et al.*, 1994; Boyd, Vanbenberghe, 2004)), le suite de notre travail est de particulariser l'approche pour l'automatique.

Le champ d'investigation ouvert couvre donc plusieurs aspects (algorithmiques, applicatifs et théoriques) dont les plus intéressants sont : la preuve par l'exemple de la réduction du temps de calcul en optimisant le code, l'application à des problèmes concrets de la commande robuste et la généralisation à des classes de problèmes plus grandes.

Bibliographie

Ben-Tal A., Ghaoui L. E., Nemirovski A. (2009). *Robust optimization*. Princeton and Oxford, Princeton University Press.

Ben-Tal A., Nemirovskii A. (2001). *Lectures on modern convex optimization : analysis, algorithms, and engineering applications*. Philadelphia, PA, SIAM.

Boyd S., Ghaoui L. E., Feron E., Balakrishnan V. (1994). *Linear matrix inequalities in system and control theory* (vol. 15). Philadelphia, PA, SIAM.

Boyd S., Vanbenberghe L. (2004). *Convex optimization*. Cambridge,UK, Cambridge University Press.

Clement B. (2010). A combining lmi and interval tools for robust optimization. In *Swim2010 : 3rd small workshop on interval methods.* Nantes, France.

Ghaoui L. E., Niculescu S. (2000). *Advances in linear matrix inequality methods in control : advances in design and control*. Philadelphia, PA, USA, Society for Industrial and Applied Mathematics.

Hansen E., Walster G. (2004). *Global optimization using interval analysis - 2nd edition*. New York, NY, Marcel Dekker.

Jansson C. (2004). A rigorous lower bound for the optimal value of convex optimization problems. *J. of Global Optimization*, vol. 28, n° 1, p. 121–137.

Jaulin L., Kieffer M., Didrit O., Walter E. (2001). *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*. London, Springer-Verlag.

Löfberg J. (2001). *Yalmip, yet another lmi parser*. Sweden, University of Linköping. (Available at `www.control.isy.liu.se/~johanl`)

Moore R. E. (1966). *Interval analysis*. Englewood Cliffs, NJ, Prentice-Hall.

Nesterov Y., Nemirovskii A. (1994). *Interior-point polynomial methods in convex programming*. Philadelphia, PA, SIAM.

Neumaier A. (1990). *Interval methods for systems of equations*. Cambridge, UK, Cambridge University Press.

Toh K., Todd M., Tutuncu R. (1999). Sdpt3 — a matlab software package for semidefinite programming. *Optimization Methods and Software*, vol. 24, p. 545-581.

Zhou K., Doyle J., Glover K. (1995). *Robust and optimal control*. Englewood Cliffs, NJ, Prentice-Hall.