Introduction aux contraintes d'intervalles. Application à l'estimation à erreurs bornées.

Laurent Granvilliers
IRIN – Université de Nantes
granvilliers@irin.univ-nantes.fr

Luc Jaulin
ISTIA – Université d'Angers
luc.jaulin@univ-angers.fr

Le fonctionnement de beaucoup de systèmes physiques, chimiques ou biologiques peuvent être décrits par des échanges d'une certaine substance entre différents réservoirs. Par exemple, la pharmacocinétique s'intéresse à l'étude de l'absorption, de la distribution et de l'excrétion d'une drogue dans le corps humain. L'évolution temporelle de la quantité de cette substance dans l'un des réservoirs (en général celui qui est accessible à la mesure) peut généralement s'exprimer comme une somme d'exponentielles. Les coefficients de ces exponentielles sont des informations importantes sur le fonctionnement des autres réservoirs (ceux qui ne sont pas accessibles à la mesure). Or, le problème de l'estimation des coefficients d'une somme d'exponentielles à partir d'un faible nombre de mesures est un problème mal conditionné, avec de nombreux optima locaux, et les méthodes existantes ne peuvent fournir une estimation fiable de ces coefficients, même lorsque le nombre d'exponentielles est faible (deux ou trois).

Dans cet article, nous allons décrire le formalisme des contraintes d'intervalles appliquées au problème de l'estimation d'une somme de trois exponentielles dans un contexte à erreurs bornées. Les outils dédiés aux contraintes constituent une alternative aux méthodes d'optimisation classiques. Ils accélèrent la convergence, et garantissent des solutions fiables.

Définition du problème

Considérons le système décrit par l'équation

$$y(\mathrm{p}, t) = p_1 \exp(-p_2 t) + p_3 \exp(-p_4 t) + p_5 \exp(-p_6 t),$$

où les p_i sont les paramètres à estimer, t est le temps et $y(\mathrm{p}, t)$ est la sortie du système au temps t. On génère par simulation un ensemble de mesures bruitées $\check{y}_i$ au temps t_i. Cela consiste à calculer $y(\mathrm{p}^*, t_i)$ pour tous les i avec

$$\mathrm{p}^* = (10, 1, -5, 0.1, 1, 0.01),$$

puis à arrondir les valeurs à une décimale. Les mesures obtenues sont rangées dans la table ci-dessous :

i	1	2	3	4	5	6	7	8	9	10	11
t_i	0	1	4	9	16	25	36	49	64	81	100
$\check{y}_i$	6	0.1	-2.2	-1.1	-0.2	0.4	0.6	0.6	0.5	0.4	0.4

Supposons aussi que pour tout i, l'erreur de mesure $|\check{y}_i - y_i|$ est inférieure à 0.05, où $\check{y}_i$ est la mesure bruitée et y_i la mesure idéale (inconnue).

L'arithmétique d'intervalles définie par Ramon E. Moore dans les années 1960 [14] apporte une solution fiable à la représentation des données incertaines. Un *intervalle* est un ensemble connexe de nombres réels. Nous considérons ici l'ensemble $\mathbb{IF}$ des intervalles clos dont les bornes sont des nombres flottants IEEE [10]. Ainsi, on peut représenter une borne sûre de l'erreur de mesure sur y_i par l'encadrement

$$y_i \in [\boldsymbol{y}_i] = [\lfloor \check{y}_i - 0.05 \rfloor, \lceil \check{y}_i + 0.05 \rceil],$$

où $\lfloor r \rfloor$ est le nombre flottant obtenu de l'arrondi de r vers $-\infty$, et $\lceil r \rceil$ le nombre flottant résultant de l'arrondi de r vers $+\infty$. Notons que les modes d'arrondis sont spécifiés dans la norme IEEE 754 et qu'ils peuvent être positionnés dynamiquement par instructions. En notation vectorielle, on écrit $\mathrm{y} \in [\mathbf{y}]$, où $[\mathbf{y}]$ est défini par le produit cartésien d'intervalles $[\boldsymbol{y}_1] \times \cdots \times [\boldsymbol{y}_{11}]$, appelé *boîte*.

Dans un contexte à erreurs bornées, estimer p revient à caractériser *l'ensemble de vraisemblance* $\mathbb{S}$ des valeurs du vecteur des paramètres qui sont consistantes avec les mesures et les informations *a priori*. Cet ensemble est défini par

$$\mathbb{S} = \Big\{ \mathrm{p} \in [\mathbf{p}^{(0)}] \mid \forall\, i \in \{1, \ldots, 11\},\ \exists y_i \in [\boldsymbol{y}_i],\ y\,(\mathrm{p}, t_i) = y_i \Big\},$$

où $[\mathbf{p}^{(0)}]$ est une boîte dont on sait *a priori* qu'elle contient p^*. Il est aisé de vérifier que p^* est bien un élément de $\mathbb{S}$.

Notons que si aucune information *a priori* n'est disponible sur les paramètres, c'est-à-dire $[\mathbf{p}^{(0)}] = \mathbb{R}^6$, et si $\mathrm{p} \in \mathbb{S}$, alors les 5 autres vecteurs obtenus de p en permutant les couples (p_1, p_2), (p_3, p_4) et (p_5, p_6) appartiennent aussi à $\mathbb{S}$. Pour cette raison, le modèle est dit *non-identifiable*. En pratique, on doit choisir un domaine $[\mathbf{p}^{(0)}]$ qui élimine ces symétries.

Une caractérisation fine de $\mathbb{S}$ est un problème difficile dès que sa dimension dépasse 3 (ici, elle vaut 6). On veut généralement trouver une boîte $[\mathbf{p}]$ qui enferme $\mathbb{S}$ avec un certain degré de précision. Chacune des composantes $[\boldsymbol{p}_i]$ donne alors un encadrement garanti de la valeur p_i^* recherchée.

Modélisation CSP–Intervalles

Un *problème de satisfaction de contraintes* (CSP) est donné par un ensemble de variables, un domaine initial, et un ensemble de contraintes. Dans notre contexte, le domaine est une boîte représentant le domaine du vecteur des variables, et les contraintes sont des équations ou inéquations non-linéaires sur les nombres réels. L'article d'Ernest Davis [6] a fondé le domaine des CSPs numériques.

Le CSP associé à notre problème d'estimation est décrit par le triplet $(C, [\mathbf{p}], \{p_1, \ldots, p_6\})$ où les p_i sont les inconnues. Le domaine initial $[\mathbf{p}]$ correspond au vecteur connu *a priori* $[\mathbf{p}^{(0)}]$. L'ensemble des contraintes C est donné ci-dessous :

$$\left\{ \begin{array}{rcl} y_1 & = & p_1 \exp(-p_2 t_1) + p_3 \exp(-p_4 t_1) + p_5 \exp(-p_6 t_1) \\ & & \vdots \\ y_{11} & = & p_1 \exp(-p_2 t_{11}) + p_3 \exp(-p_4 t_{11}) + p_5 \exp(-p_6 t_{11}) \end{array} \right.$$

Les t_i sont des réels connus alors que les réels y_i, qui représentent les mesures non bruitées, sont incertaines. On sait juste qu'ils appartiennent aux intervalles $[\boldsymbol{y}_i]$.

Le problème de la résolution d'un CSP est celui de déterminer les valeurs prises dans le domaine initial $[\mathbf{p}]$ qui vérifient toutes les contraintes. On dit qu'une valeur $(p_1, \ldots, p_6)$ *satisfait ou vérifie* la i-ème contrainte de C si on a la relation

$$[\boldsymbol{y}_i] \ni p_1 \exp(-p_2 t_i) + p_3 \exp(-p_4 t_i) + p_5 \exp(-p_6 t_i).$$

Le problème essentiel est de garantir un calcul numérique fiable. En d'autres termes, on veut être sûr de vérifier numériquement la relation si elle est vraie (sur les nombres réels). Pour cela, il suffit de réaliser les calculs sur l'ensemble des intervalles $\mathbb{IF}$. On touche ici un aspect essentiel de l'arithmétique d'intervalles qui est la quantification des erreurs d'arrondis. Considérons, à titre d'exemple, le problème de l'évaluation garantie de $\exp(x)$ où x est un nombre réel non représentable. La première étape consiste à représenter x par l'intervalle

$$x \in [\lfloor x \rfloor, \lceil x \rceil]$$

contenant x. La deuxième étape consiste à calculer l'image par la fonction exponentielle de cet intervalle (voir la figure 1). Comme la fonction exponentielle est monotone croissante sur $\mathbb{R}$, on a la relation

$$\exp(x) \in [\exp(\lfloor x \rfloor), \exp(\lceil x \rceil)].$$

Enfin, il faut encore arrondir « vers l'extérieur » les deux bornes qui ne sont pas nécessairement représentables. On calcule ainsi l'encadrement garanti

$$\exp(x) \in [\lfloor \exp(\lfloor x \rfloor) \rfloor, \lceil \exp(\lceil x \rceil) \rceil].$$

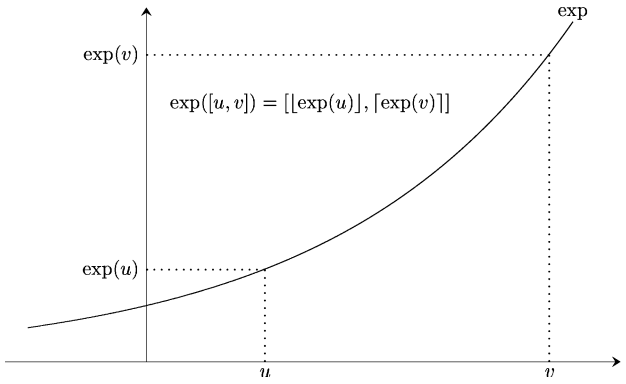

FIG. 1 – Image d'un intervalle par la fonction exponentielle.

Les opérations de l'arithmétique d'intervalles sont des extensions ensemblistes des opérations sur $\mathbb{R}$. Comme pour l'exponentielle, les formules de calcul explicites pour les autres opérations sont définies de façon à garantir que chaque intervalle calculé contient la variation de la fonction réelle.

L'arithmétique d'intervalles donne un moyen de calculer l'image d'une boîte par une fonction réelle. Ce moyen est appelé *évaluation sur les intervalles*. On en déduit un algorithme de réfutation pour la satisfaction de la contrainte c_i. Plus précisément, on évalue l'expression

$$\begin{array}{l}[\boldsymbol{p}_1]\exp(-[\boldsymbol{p}_2]\times[\lfloor t_i\rfloor,\lceil t_i\rceil])\ +\\ [\boldsymbol{p}_3]\exp(-[\boldsymbol{p}_4]\times[\lfloor t_i\rfloor,\lceil t_i\rceil])\ +\\ [\boldsymbol{p}_5]\exp(-[\boldsymbol{p}_6]\times[\lfloor t_i\rfloor,\lceil t_i\rceil]).\end{array}$$

sur $\mathbb{IF}$. Elle correspond au membre droit de c_i où chaque constante (ici, t_i) est remplacée par le plus petit sur-ensemble de $\mathbb{IF}$, chaque variable p_i est remplacée par son domaine $[\boldsymbol{p}_i]$, et chaque opération est l'opération sur $\mathbb{IF}$. Le résultat, notons le $[\boldsymbol{z}]$, est nécessairement un sur-ensemble de la variation de l'expression réelle sur le domaine $[\mathbf{p}]$. On en déduit que si la relation

$$[\boldsymbol{y}_i]\cap[\boldsymbol{z}]=\varnothing$$

est vérifiée, alors $[\mathbf{p}]$ ne contient aucune solution de c_i. Plus généralement, on peut rejeter $[\mathbf{p}]$ s'il existe une contrainte de C qui n'est pas satisfaite.

Par exemple, considérons la contrainte $x^2-1=0$ et le domaine $[2,3]$. On conclut que ce domaine ne contient aucune solution car

$$[2,3]^2-[1,1]\equiv[3,8]\quad\text{ et }\quad[3,8]\cap[0,0]=\varnothing.$$

Néanmoins, cet algorithme de réfutation ne suffit pas. En pratique, on veut à la fois réduire le domaine en coupant des sous-domaines ne contenant pas de solution des contraintes, et séparer les solutions différentes. Ces

techniques de réduction-découpage des domaines (*branch-and-prune*) sont abordées dans la section suivante.

Algorithmes de type réduction-découpage

Les algorithmes de type réduction-découpage maintiennent un ensemble de boîtes à considérer (à l'initialisation, le singleton $\{[\mathbf{p}^{(0)}]\}$). Le résultat est un ensemble de boîtes d'une précision donnée. En pratique, la précision est représentée par un scalaire ε. Un intervalle $[a, b]$ est dit précis à ε si sa largeur vérifie

$$\lceil b - a \rceil \leqslant \varepsilon.$$

Par extension, une boîte est précise à ε si tous ses composants sont précis à ε. Un pas de calcul consiste à choisir une boîte qui n'est pas précise à ε, à réduire la boîte, puis à la découper en plusieurs sous-boîtes si la précision ε n'est pas atteinte après réduction. La propriété essentielle de ces algorithmes est la propriété de complétude, induite par la mise en œuvre de l'arithmétique d'intervalles : toutes les solutions réelles du CSP sont contenues dans au moins l'une des boîtes en sortie. On ne perd pas de solution.

Le découpage d'une boîte consiste généralement à générer deux sous-boîtes en coupant suivant une dimension en deux parties. La dimension choisie est en général celle de l'intervalle le moins précis. C'est la stratégie la meilleure en moyenne. Elle est appelée *max-dom* dans le contexte des CSPs.

Mais l'efficacité de ces algorithmes dépend essentiellement de la puissance des techniques de réductions. À l'origine, dans la communauté « Intervalles », seule la preuve par réfutation décrite précédemment était employée. On avait ainsi des algorithmes de type découpage-évaluation. Considérons, par exemple, le CSP à une contrainte $c : x^2 = 1$ avec le domaine $[0, 3]$ pour x. Une première évaluation permet de déduire que c est satisfaite car

$$[0, 3]^2 \cap [1, 1] \neq \varnothing.$$

On doit donc couper le domaine de x en deux sous-domaines $[0, 1.5]$ et $[1.5, 3]$. Puis on rejette $[1.5, 3]$ car

$$[1.5, 3]^2 \cap [1, 1] = \varnothing.$$

Cependant, on doit itérer le processus pour $[0, 1.5]$. Comme le domaine n'est pas centré sur la racine, la convergence est lente.

Dans l'algorithme développé ci-dessus, on ne profite pas de notre connaissance *a priori* de l'égalité $x^2 = 1$. En effet, on connaît bien la technique d'inversion de c permettant d'exprimer x en fonction d'une expression. Ici, on aurait $x = \pm\sqrt{1}$. On pourrait ainsi réduire le domaine initial de x à l'intervalle $[1, 1]$ en un coup.

Propagation de contraintes

La méthode de réduction suggérée dans le paragraphe précédent a pour nom la consistance d'enveloppe (*hull consistency*). Elle a pour origine les travaux de John G. Cleary [4], de Eero Hyvönen [9], puis de Frédéric Benhamou et Bill Older [3].

Une opération de réduction élémentaire cherche à réduire les domaines de toutes les variables apparaissant dans une contrainte dite primitive. Les contraintes primitives sont des contraintes avec au plus un symbole d'opération, comme « $z = x + y$ », « $z = x \times y$ » et « $y = \exp(x)$ ». Considérons, par exemple, la contrainte $y = \exp(x)$ sur le domaine $[\boldsymbol{x}] \times [\boldsymbol{y}] = [0, 1] \times [-1, 2]$. Cette contrainte représente la relation

$$\{(x, y) \in [\boldsymbol{x}] \times [\boldsymbol{y}] \mid y = \exp(x)\}.$$

La réduction de chaque domaine correspond à la projection de cet ensemble sur chaque variable. La projection par rapport à y est définie par

$$\{y \in [\boldsymbol{y}] \mid \exists x \in [\boldsymbol{x}],\ y = \exp(x)\}.$$

On en déduit l'instruction de réduction de $[\boldsymbol{y}]$, soit

$$[\boldsymbol{y}] := [\boldsymbol{y}] \cap \exp([\boldsymbol{x}]).$$

Ainsi, le domaine de y est réduit et devient $[1, 2] = [-1, 2] \cap [0, \lceil e \rceil]$. La projection par rapport à x est définie par

$$\begin{array}{rl} & \{x \in [\boldsymbol{x}] \mid \exists y \in [\boldsymbol{y}],\ y = \exp(x)\} \\ = & \{x \in [\boldsymbol{x}] \mid \exists y \in [\boldsymbol{y}],\ x = \exp^{-1}(y)\}. \end{array}$$

L'instruction de réduction pour x considère la forme inverse de la contrainte (voir la figure 2). Elle a pour expression

$$[\boldsymbol{x}] := [\boldsymbol{x}] \cap \log([\boldsymbol{y}] \cap]0, +\infty[),$$

où la fonction logarithme, de domaine $]0, +\infty[$, est l'inverse de la fonction exponentielle. Ainsi, le domaine de x devient $[0, \lceil \log(2) \rceil]$. En résumé, on constate que les domaines des variables sont réduits par des opérations simples, alors que l'évaluation par intervalles permet seulement de satisfaire la contrainte (on a $[-1, 2] \cap \exp([0, 1]) \neq \varnothing$). Notons que des instructions de réduction explicites existent pour chaque type de contrainte primitive.

Historiquement, ces opérations ont été rassemblées sous le nom d'arithmétique d'intervalles relationnelle (par opposition à l'arithmétique fonctionnelle de R. E. Moore). Cela vient du traitement au même niveau de toutes les variables d'un contrainte primitive. Le but était d'avoir une arithmétique « logique » (d'après J. G. Cleary) compatible avec les machines Prolog.

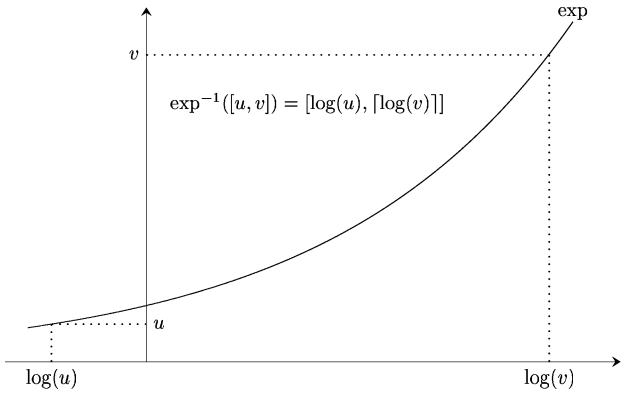

FIG. 2 – Image inverse de la fonction exponentielle.

Pour un ensemble de contraintes non primitives, on décompose chaque contrainte en contraintes primitives en introduisant de nouvelles variables ayant pour domaine $[-\infty, +\infty]$. Par exemple, pour $x + \exp(y + 1) = 0$, on génère l'ensemble $\{u = y + 1, x + u = 0\}$, où u est une nouvelle variable.

Les opérations de réduction pour toutes les contraintes primitives obtenues des contraintes du CSP sont appliquées par un algorithme de point-fixe dit de propagation de contraintes. On itère le processus tant que des réductions sont encore possibles. Pourvu que chaque opération de réduction élémentaire soit correcte, monotone et contractante, l'algorithme de point-fixe est correct, contractant, confluent et termine. En d'autres termes, quelque soit l'ordre d'application des réductions, le résultat est unique, et il est obtenu en temps fini.

Il existe d'autres techniques de réduction plus performantes qu'on ne décrira pas ici. Citons par exemple la consistance de boîte |2| (*box consistency*), la famille des consistances fortes (*kB consistency*) d'Olivier Lhomme [13], et les algorithmes implémentés dans Numerica [15] combinant ces techniques aux algorithmes d'analyse numérique de type Newton-Raphson.

Résultats expérimentaux

Nous avons appliqué le solveur `Inbox` [7] au problème qui fait l'objet de cette étude. La boîte $[\mathbf{p}^{(0)}]$ choisie comme domaine initial vaut

$$[-1000, 1000] \times [0.5, 2] \times [-1000, 1000] \times [0.05, 0.5] \times [-1000, 1000] \times [0, 0.05].$$

Pour la technique de réduction basée sur la 3B consistance, nous obtenons

$$\begin{array}{rlllllll} \mathbb{S} \subset & [9.7174, 10.2186] & \times & [0.9658, 1.0677] & \times & [-5.1577, -4.7736] & \times \\ & [0.0952, 0.1041] & \times & [0.9618, 1.0164] & \times & [0.0099, 0.0101] & \end{array}$$

en 370s sur un PC Pentium III/933MHz. On peut vérifier que $\mathrm{p}^* \in \mathbb{S}$.

Perspectives

Les algorithmes de type réduction-découpage présentés précédemment calculent des approximations extérieures (des sur-ensembles) des solutions des CSPs, où les variables sont des variables libres. Ils sont bien adaptés au calcul de solutions ponctuelles, et en particulier au traitement des systèmes carrés d'équations à coefficients réels. Par contre, ils sont insuffisants pour traiter des hypersurfaces, ou encore des variables quantifiées. C'est le cas de notre problème d'étude. En effet, chaque contrainte contient une variable incertaine y_i et peut donc être remplacée par une double inégalité. L'ensemble solution $\mathbb{S}$ à caractériser est donc un hypervolume de $\mathbb{R}^6$. Le cas d'une somme d'exponentielles est tellement mal conditionné que $\mathbb{S}$ ressemble à un fil très fin de volume très petit, mais non nul.

La caractérisation fine d'une hypersurface peut découler d'une combinaison de calculs d'approximations extérieures et d'approximations intérieures (on approche la frontière de l'extérieur et par l'intérieur). Ces méthodes hybrides ont été peu étudiées dans le domaine des CSPs numériques. Elles sont pourtant essentielles.

Un autre domaine de recherche actif est celui des algorithmes symboliques-numériques [8]. Les tranformations symboliques ont pour but de générer des expressions des contraintes adaptées aux calculs numériques. Dans le cas de l'analyse par intervalles, on peut chercher à éliminer des occurrences de variables pour minimiser l'effet du *problème de dépendance*, ou encore à combiner les contraintes pour parer la localité des raisonnements réalisés par les méthodes de consistances locales.

Les contraintes avec quantificateurs ont été étudiées récemment par Alain Colmerauer et Thi Bich Hanh Dao [5] dans une théorie générale du premier ordre, et plus spécifiquement par F. Benhamou et Frédéric Goualard [1] pour les contraintes d'intervalles avec une variable quantifiée. La problématique de ces derniers prend sa source dans une application de synthèse d'images, où le problème est de garantir des contraintes sur une scène filmée quelque soit le temps de prise de vue.

D'autres travaux cherchent à étendre et appliquer les techniques CSPs à l'optimisation [15], à la résolution d'équations différentielles [11], etc. On conclura en disant que notre soucis est de proposer nos outils à différentes communautés de sciences appliquées, et de se nourrir des applications pour les faire progresser. Le livre de l'un des auteurs [12] est un bon exemple d'une telle avancée en automatique.

Références

[1] F. Benhamou and F. Goualard. Universally Quantified Interval Constraints. In *Proceedings of CP'2000*, volume 1894 of *LNCS*, pages 67–82, 2000.

[2] F. Benhamou, D. McAllester, and P. Van Hentenryck. CLP(Intervals) Revisited. In *Proceedings of ILPS'94*, 1994.

[3] F. Benhamou and W. J. Older. Applying Interval Arithmetic to Real, Integer and Boolean Constraints. *Journal of Logic Programming*, 32(1) :1–24, 1997.

[4] J. G. Cleary. Logical arithmetic. *Future Computing Systems*, 2(2) :125–149, 1987.

[5] A. Colmerauer and T.-B.-H. Dao. Expressiveness of full first order constraints in the algebra of finite or infinite trees. In *Proceedings of CP'2000*, volume 1894 of *LNCS*, pages 172–186, 2000.

[6] E. Davis. Constraint Propagation with Interval Labels. *Artificial Intelligence*, 32 :281–331, 1987.

[7] L. Granvilliers. On the Combination of Interval Constraint Solvers. *Reliable Computing*, 7(6) :467–483, 2001.

[8] L. Granvilliers, E. Monfroy, and F. Benhamou. Symbolic-Interval Cooperation in Constraint Programming. In *Proceedings of ISSAC'2001*, 2001.

[9] E. Hyvönen. Constraint Reasoning based on Interval Arithmetic. The Tolerance Propagation Approach. *Artificial Intelligence*, 58 :71–112, 1992.

[10] Institute of Electrical and Electronics Engineers. IEEE Standard for Binary Floating-Point Arithmetic. Technical Report IEEE Std 754-1985, 1985. Reaffirmed 1990.

[11] M. Janssen, P. Van Hentenryck, and Y. Deville. A Constraint Satisfaction Approach to Parametric Differential Equations. In *Proceedings of IJCAI'2001*, 2001.

[12] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis : With Examples in Parameter and State Estimation, Robust Control and Robotics*. Springer, 2001.

[13] O. Lhomme. Consistency Techniques for Numeric CSPs. In *Proceedings of IJCAI'93*, 1993.

[14] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.

[15] P. Van Hentenryck, L. Michel, and Y. Deville. *Numerica : a Modeling Language for Global Optimization*. MIT Press, 1997.