Estimation d'état de systèmes non-linéaires à temps continus par une approche ensembliste

Luc Jaulin

Estimation d'état des systèmes non-linéaires à temps continu par une approche ensembliste

Luc Jaulin
Laboratoire d'Ingénierie des Systèmes Automatisés,
62 avenue Notre Dame du Lac 49 000 Angers
jaulin@univ-angers.fr

http://www.istia.univ-angers.fr/~jaulin/

Résumé: Cet article illustre l'utilisation de l'analyse par intervalles et des méthodes de propagation de contraintes pour l'estimation d'état de

systèmes à temps continu décrit par des équations différéntielles ordinaires. Cette approche est présentée dans un contexte ensembliste, c'est-à-dire que les incertitudes sur les quantités manipulées sont représentées par un ensemble et non plus par une loi de probabilité.

Mots clefs: estimation à erreurs bornées, propagation de contraintes, CSP, identification non-linéaire, analyse par intervalles, estimation d'état, estimation ensembliste.

Abstract: This paper presents a first study on the application of interval analysis and consistency techniques to state estimation of continuous-time systems described by nonlinear ordinary differential equations. The approach is presented in a bounded-error context and the resulting methodology is illustrated on an example.

1 Introduction

Considérons le problème de l'estimation de l'état d'un système non-linéaire décrit par ses équations d'état

$$\left\{ \begin{array}{lcl} \dot{\mathbf{x}}(t) & = & \mathbf{f}(\mathbf{x}(t)), \\ \mathbf{y}(t) & = & \mathbf{g}(\mathbf{x}(t)) \end{array} \right. \tag{1}$$

où

- t est le temps, supposé dans l'intervalle $[\underline{t}, \overline{t}] \subset \mathbb{R}$
- $\mathbf{x}(t) \in \mathbb{R}^n$ est le vecteur d'état à l'instant t,
- et $\mathbf{y}(t) \in \mathbb{R}^m$ est le vecteur de sortie à l'instant t.

Ce problème a déjà été abondamment étudié dans le cas où les fonctions $\mathbf{f}$ et $\mathbf{g}$ sont toutes les deux linéaires [12], [3], ou lorsque le temps t est discret [2], [7] ou encore dans un contexte

probabiliste. Pourtant, à ma connaissance, dans un contexte à erreurs bornées et pour le cas des systèmes non-linéaires à temps continus, l'estimation d'état n'a jamais été traitée, sauf bien sur dans l'article [6] dont cet article reprend les principaux résultats. Les outils que nous allons utiliser sont l'*analyse par intervalles* [9] qui nous permettra d'obtenir des résultats garantis et les *techniques de propagation de contraintes* [4] , [11] qui nous permettront d'améliorer l'efficacité des algorithmes par intervalles.

Nous allons supposer que la fonction $\mathbf{f}$ est différentiable et que pour tout t, des domaines $[x_i](t)$ et $[y_j](t)$ contenant les variables $x_i(t)$ et $y_j(t)$ sont disponibles a priori. Ces domaines peuvent être arbitrairement grands, par exemple $[x_i](t) =]-\infty, \infty[$ si aucune information n'est disponible sur $x_i(t)$. Ils peuvent aussi être arbitrairement petits. Ainsi, si une mesure $\hat{y}_j(t_0)$ de la variable de sortie $y_j(t_0)$ a été recueillie à l'instant $t = t_0$ et si une borne supérieure $\bar{e}$ sur la valeur absolue de l'erreur $\hat{y}_j(t_0) - y_j(t_0)$ est connue, alors, le domaine pour $y_j(t_0)$ sera

$$[y_j](t_0) = [\hat{y}_j(t_0) - \bar{e}, \hat{y}_j(t_0) + \bar{e}]. \tag{2}$$

Le paragraphe 2 rappelle les notions de base sur l'analyse par intervalles qui seront utilisées dans cet article. Le paragraphe 3 introduit une méthode élémentaire permettant de traiter simplement avec des équations différentielles ordinaires. Un algorithme permettant de faire l'estimation d'état du système (1) sera proposé dans le paragraphe 4. Enfin, un exemple sera donné au paragraphe 5.

2 Calcul sur les intervalles

Un *intervalle* $[x] = [\underline{x}, \overline{x}]$ est un ensemble fermé et connexe de $\mathbb{R}$. Les opérations sur les nombres réels peuvent être étendues aux intervalles [9]. En effet, tout opérateur binaire $\diamond$ tel que $+, -, *, /$ et toute fonction élémentaire f telles que exp, sin, sqr, sqrt, sur les nombres réels se généralisent aux intervalles de la façon suivante :

$$\begin{array}{rcl} [x] \diamond [y] & = & [\inf_{x\in[x], y\in[y]} x \diamond y, \\ & & \sup_{x\in[x], y\in[y]} x \diamond y] \\ f([x]) & = & [\inf_{x\in[x]} f(x), \sup_{x\in[x]} f(x)] \end{array} \tag{3}$$

Par exemple,

$$\begin{array}{rcl} [x] + [y] & = & [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \\ [x] * [y] & = & [\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}) \\ & & , \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})] \\ \exp([x]) & = & [\exp(\underline{x}), \exp(\overline{x})], \\ \sqrt{[x]} & = & [\sqrt{\max(\underline{x}, 0)}, \sqrt{\overline{x}}] \text{ si } \overline{x} \geq 0 \\ & & \text{et } \emptyset \text{ si } \overline{x} < 0 \end{array} \tag{4}$$

Un *pavé* $[\mathbf{x}]$ de $\mathbb{R}^n$ est le produit cartésien de n intervalles. Il peut s'écrire sous la forme

$$[\mathbf{x}] = [\underline{x}_1, \bar{x}_1] \times \cdots \times [\underline{x}_n, \bar{x}_n] = [\underline{\mathbf{x}}, \bar{\mathbf{x}}], \tag{5}$$

où $\underline{\mathbf{x}} = (\underline{x}_1, \cdots, \underline{x}_n)^{\mathrm{T}}$ et $\bar{\mathbf{x}} = (\overline{x}_1, \cdots, \bar{x}_n)^{\mathrm{T}}$. L'ensemble de tous les pavés de $\mathbb{R}^n$ sera noté $\mathbb{IR}^n$. Remarquons que l'ensemble

$$\mathbb{R}^n =]-\infty, \infty[\times \cdots \times]-\infty, \infty[\tag{6}$$

est un élément de $\mathbb{IR}^n$. La *longueur* $w([\mathbf{x}])$ du pavé $[\mathbf{x}]$ est la longueur de son plus long côté. Couper $[\mathbf{x}]$ signifie : "couper en deux suivant l'axe de symétrie perpendiculaire au côté le plus long".

Soit $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^p$ une fonction vectorielle. La fonction ensembliste $[\mathbf{f}] : \mathbb{IR}^n \to \mathbb{IR}^p$ est une *fonction d'inclusion* de $\mathbf{f}$ si

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

pour tout $[\mathbf{x}] \in \mathbb{IR}^n$. Une fonction d'inclusion est *convergente* si, pour toute suite de pavés $[\mathbf{x}]$ de $\mathbb{IR}^n$, on a

$$w([\mathbf{x}]) \to 0 \Rightarrow w([\mathbf{f}]([\mathbf{x}])) \to 0, \tag{8}$$

Différentes méthodes existent pour obtenir des fonctions d'inclusion convergentes, [10]. La plus simple d'entre elles et qui permet de traiter une très grande classe de fonction consiste à remplacer toutes les variables d'entrées de la fonction par des intervalles et tous les opérateurs et les fonctions élémentaires définissant la fonction par les opérateurs et fonctions intervalles associés. Illustrons cela par deux exemples.

Exemple 2.1 *Soit la fonction*

$$f(x_1, x_2) = \frac{1}{\sqrt{2\pi x_2}} e^{-\frac{1}{2}\frac{(t-x_1)^2}{x_2}}, \tag{9}$$

où t est un nombre réel donné. Une fonction d'inclusion $[f]$ de f est donnée par

$$[f]([x_1], [x_2]) = \frac{1}{\sqrt{2\pi [x_2]}} e^{-\frac{1}{2}\frac{(t-[x_1])^2}{[x_2]}}, \tag{10}$$

où toutes les opérations utilisées sont celles de l'arithmétique des intervalles.

Exemple 2.2 *Considérons la fonction vectorielle*

$$\mathbf{f}(x_1, x_2) \triangleq \begin{pmatrix} (1 - 0.01x_2)\, x_1 \\ (-1 + 0.02x_1)\, x_2 \end{pmatrix}. \tag{11}$$

Le calcul suivant montre comment se calcule un pavé contenant l'ensemble $\mathbf{f}([\mathbf{x}])$*, où* $[\mathbf{x}] = ([10,20],[40,50])$*:*

$$\begin{aligned}
\mathbf{f}([\mathbf{x}]) &\subset \begin{pmatrix} (1-0.01*[40,50])*[10,20] \\ (-1+0.02*[10,20])*[40,50] \end{pmatrix} \\
&= \begin{pmatrix} (1-[0.4,0.5])*[10,20] \\ (-1+[0.2,0.4])*[40,50] \end{pmatrix} \\
&= \begin{pmatrix} [0.5,0.6]*[10,20] \\ [-0.8,-0.6]*[40,50] \end{pmatrix} \\
&= \begin{pmatrix} [5,12] \\ [-40,-24] \end{pmatrix}.
\end{aligned}$$

Bien sûr, ce calcul peut être appliqué à tout pavé $[\mathbf{x}] = [x_1] \times [x_2]$*. L'algorithme résultant correspond à une fonction d'inclusion pour* $\mathbf{f}$*.* ■

Soient $[\mathbf{x}]$ et $[\mathbf{y}]$ deux pavés de $\mathbb{R}^n$. Nous noterons $[\mathbf{x}] \sqcup [\mathbf{y}]$ le plus petit pavé contenant l'union $[\mathbf{x}] \cup [\mathbf{y}]$ des deux pavés. *Gonfler* un pavé $[\mathbf{x}] = [\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_n, \overline{x}_n]$ de $\varepsilon > 0$ consiste à le remplacer par le pavé

$$\begin{aligned}
\text{gonfle}([\mathbf{x}], \varepsilon) &\triangleq [\underline{x}_1 - \varepsilon, \overline{x}_1 + \varepsilon] \times \cdots \\
&\quad \times [\underline{x}_n - \varepsilon, \overline{x}_n + \varepsilon].
\end{aligned}$$

3 Théorème de Picard

L'utilisation de l'analyse par intervalles pour le traitement d'équations différentielles a été proposé pour la première fois par Moore [9] (voir [1] pour un survol du sujet). Elle permet de développer des méthodes numériques qui fournissent des ensembles contenant à coup sûr les solutions exactes des équations différentielles aux instants d'échantillonnage $t_0, t_1, \ldots$ lorsqu'un pavé $[\mathbf{x}](0)$ contenant l'état initial $\mathbf{x}(t_0)$ est donné. L'approche est fondée sur le théorème de Picard que nous allons maintenant énoncer.

Théorème 3.1 *Soient* t_1 *et* t_2 *deux nombres réels. Supposons que* $\mathbf{x}(t_1)$ *appartienne au pavé* $[\mathbf{x}](t_1)$*. Soit* $[\mathbf{w}]$ *un pavé de* $\mathbb{IR}^n$*. Dans notre contexte, ce pavé sera choisi pas trop gros et de telle façon qu'il ait de fortes chances de contenir la trajectoire* $\mathbf{x}(\tau)$ *pour tout* $\tau \in [t_1, t_2]$*. Si*

$$[\mathbf{x}](t_1) + [0, t_2 - t_1] * [\mathbf{f}]([\mathbf{w}]) \subset [\mathbf{w}],$$

où $[\mathbf{f}](.)$ *est une fonction d'inclusion pour* $\mathbf{f}$ *et où* $[0, t_2 - t_1]$ *est le plus petit intervalle contenant* 0 *et* $t_2 - t_1$*, alors pour tout* $\tau \in [t_1, t_2]$*, on a*

$$\mathbf{x}(\tau) \in [\mathbf{x}](t_1) \ + [0, t_2 - t_1] * [\mathbf{f}]([\mathbf{w}]), \tag{12}$$

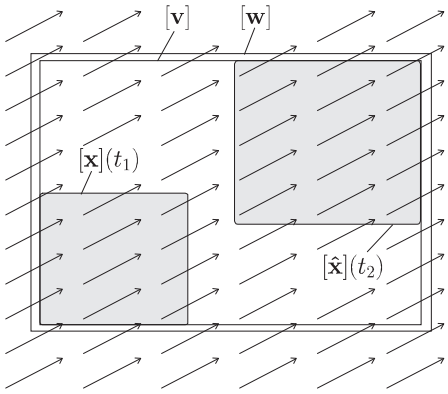

Figure 1: Illustration du principe de l'algorithme $[\boldsymbol{\phi}]$

De plus

$$\mathbf{x}(t_2) \in [\mathbf{x}](t_1) \ + (t_2 - t_1) * [\mathbf{f}]([\mathbf{w}]). \tag{13}$$

Nous pouvons maintenant construire un algorithme $[\boldsymbol{\phi}]$ capable de calculer un pavé $[\mathbf{x}](t_2)$ contenant $\mathbf{x}(t_2)$ à partir d'un pavé $[\mathbf{x}](t_1)$ dont on sait qu'il contient $\mathbf{x}(t_1)$. La figure 1 illustre le principe de cet algorithme.

Fonction $[\phi]$ (in : $t_1, t_2, [\mathbf{x}](t_1)$, out : $[\mathbf{x}](t_2)$)
1 $[\hat{\mathbf{x}}](t_2) := [\mathbf{x}](t_1) + (t_2 - t_1) * [\mathbf{f}]([\mathbf{x}](t_1))$;
2 $[\mathbf{v}] := [\mathbf{x}](t_1) \sqcup [\hat{\mathbf{x}}](t_2)$;
3 $[\mathbf{w}] :=$ gonfle$([\mathbf{v}], \eta)$; //où $\eta > 0$ est petit
4 si $[\mathbf{x}](t_1) + [0, t_2 - t_1] * [\mathbf{f}]([\mathbf{w}])) \not\subseteq [\mathbf{w}]$
5 $\{[\mathbf{x}](t_2) := \mathbb{R}^n$; fin$\}$;
6 $[\mathbf{x}](t_2) := [\mathbf{x}](t_1) + (t_2 - t_1) * [\mathbf{f}]([\mathbf{w}])$.

Le pas 1 calcule un pavé $[\hat{\mathbf{x}}](t_2)$ qui pourrait contenir l'ensemble de tous les $\mathbf{x}(t_2)$ tels que $\mathbf{x}(t_1) \in [\mathbf{x}](t_1)$, mais aucune garantie n'est offerte. Le pas 2 calcule le plus petit pavé $[\mathbf{v}]$ contenant $[\mathbf{x}](t_1)$ et $[\hat{\mathbf{x}}](t_2)$. Le pavé $[\mathbf{v}]$ est alors légèrement gonflé (de $\eta > 0$) au pas 3 pour obtenir le pavé $[\mathbf{w}]$. Si, au pas 4, la condition du théorème 3.1 n'est pas satisfaite, aucun pavé fini contenant $\mathbf{x}(t_2)$ ne peut être calculé et $\mathbb{R}^n$ se trouve alors renvoyé au pas 5. Sinon, le pas 6 calcule un pavé contenant $\mathbf{x}(t_2)$ grâce à la propriété (13).

Remarque 3.1 *Le coefficient de gonflement η de la fonction $[\phi]$ doit être choisi suffisamment faible de façon à avoir $[\mathbf{w}]$ à peine plus gros que $[\mathbf{v}]$, mais pas nul car sinon, la condition du pas 4 a peu de chance d'être satisfaite. Dans l'exemple traité dans la suite de cet article, nous avons choisi $\eta = 0.1.w([\mathbf{v}]) + 0.0001$ mais beaucoup d'autres statégies pour l'obtention de ce coefficient*

donnent des résultats tout aussi satisfaisants. Une recherche dichotomique pourrait aussi être envisagée pour obtenir un η convenable de façon automatique.

L'algorithme suivant utilise l'opérateur $[\boldsymbol{\phi}]$ pour calculer des pavés contenant à coup sûr les quantités $\mathbf{x}(\delta)$, $\mathbf{x}(2\delta)$, ..., $\mathbf{x}(\bar{k}\delta)$, où δ est la période d'échantillonnage et où $\overline{k}$ est le plus grand entier inférieur à $\overline{t}/\delta$. Ce calcul est fait à partir de la connaissance d'un pavé $[\mathbf{x}](0)$ contenant $\mathbf{x}(0)$. Pour une question de concision, la notation $[\mathbf{x}](k)$ sera utilisée à la place $[\mathbf{x}](k\delta)$.

PICARD (in : $[\mathbf{x}](0)$, out : $[\mathbf{x}](k), k \geq 1$)
1 pour $k := 0$ jusqu'à $\bar{k} - 1$
2 $\quad [\mathbf{x}](k+1) := [\boldsymbol{\phi}]\ (k\delta, (k+1)\delta, [\mathbf{x}](k))$

Exemple 3.1 *Considérons le modèle de Lotka-Volterra donné par:*

$$\begin{cases} \dot{x}_1 &= (1 - 0.01x_2)\, x_1 \\ \dot{x}_2 &= (-1 + 0.02x_1)\, x_2 \end{cases} \tag{14}$$

Pour

$$[\mathbf{x}](0) = [49.5; 50.5] \times [49.5; 50.5],$$

$\delta = 0.005$ *et* $\bar{k} = 400$, *l'algorithme* PICARD *génère un ensemble de pavés* $[\mathbf{x}](k)$, *dont la superposition est représentée sur la figure 2. Pour tout* $k \in \{1, \ldots, \bar{k}\}$, *nous avons l'implication*

$$\mathbf{x}(0) \in [\mathbf{x}](0) \Rightarrow \mathbf{x}(k) \in [\mathbf{x}](k). \tag{15}$$

Les $\mathbf{x}(k)$ *en représentés en blanc sur la figure ont été obtenus pour* $\mathbf{x}(0) = (50, 50)$.

L'exemple 3.1 illustre le principal inconvénient des méthodes par intervalles pour le traitement des équations différentielles, à savoir, l'explosion rapide de la taille des pavés $[\mathbf{x}](k)$ lorsque le temps augmente. Cette explosion peut être réduite considérablement en utilisant les méthodes de Lohner [8]. Dans le contexte de l'estimation d'état, cette explosion peut aussi être réduite grâce aux mesures prélevées sur le système à différents instants. Ce dernier point fera l'objet du prochain paragraphe.

4 Algorithme pour l'estimation d'état

Considérons à nouveau le système non-linéaire (1), où des domaines (qui sont en fait des intervalles) $[x_i](t)$ et $[y_j](t)$ pour les variables $x_i(t)$ et $y_j(t)$ sont supposés connus. Rappelons que ces intervalles sont égaux à $]-\infty, \infty[$ si aucune information a priori n'est disponible sur la variable associée. Aux instants d'échantillonnage, $t = k\delta$, les variables $x_i(t)$ et $y_j(t)$ seront notées $x_i(k)$

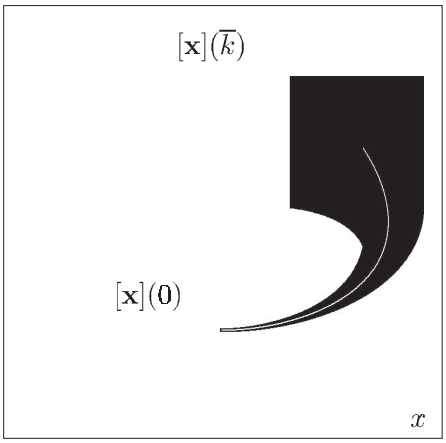

Figure 2: Superposition de pavés contenant toutes les trajectoires possibles du système sachant que le vecteur initial appartient au pavé $[\mathbf{x}](0)$; cette superposition a été générée par l'algorithme PICARD

et $y_j(k)$. Les techniques de propagation permettent de contracter les domaines pour les variables en supprimant des parties des domaines qui sont incompatibles avec (1) et les domaines pour les autres variables. L'utilisation de ces méthodes pour le traitement des équations différentielles a été proposée pour la première fois dans [5]. Les contractions considérées utilisent les implications suivantes.

$$
\begin{array}{ll}
\text{(i)} & \mathbf{x}(k) \in [\mathbf{x}](k) \\
& \Longrightarrow \mathbf{x}(k+1) \in [\boldsymbol{\phi}](k\delta, (k+1)\delta, [\mathbf{x}](k)), \\
\text{(ii)} & \mathbf{x}(k) \in [\mathbf{x}](k) \\
& \Longrightarrow \mathbf{y}(k) \in \mathbf{g}([\mathbf{x}](k)), \\
\text{(iii)} & \mathbf{y}(k) \in [\mathbf{y}](k) \\
& \Longrightarrow \mathbf{x}(k) \in \mathbf{g}^{-1}([\mathbf{y}](k)), \\
\text{(iv)} & \mathbf{x}(k+1) \in [\mathbf{x}](k+1) \\
& \Longrightarrow \mathbf{x}(k) \in [\boldsymbol{\phi}]((k+1)\delta, k\delta, [\mathbf{x}](k+1)).
\end{array}
$$

Chacune de ces implications engendre une procédure de contraction de l'algorithme présenté ci-dessous.

CONTRACT(inout : $[\mathbf{x}](k), [\mathbf{y}](k), k \geq 0$)
1 pour $k := 0$ jusqu'à $\bar{k}$,
2 $[\mathbf{x}](k+1) := [\mathbf{x}](k+1)$
$\cap[\boldsymbol{\phi}](k\delta, (k+1)\delta, [\mathbf{x}](k))$;
3 $[\mathbf{y}](k) := [\mathbf{y}](k) \cap [\mathbf{g}]([\mathbf{x}](k))$;
4 pour $k := \bar{k}$ jusqu'à 0,
5 $[\mathbf{x}](k) := [\mathbf{x}](k) \cap [\mathbf{g}^{-1}]([\mathbf{y}](k))$;
6 $[\mathbf{x}](k) := [\mathbf{x}](k)$
$\cap[\boldsymbol{\phi}]((k+1)\delta, k\delta, [\mathbf{x}](k+1))$;

Au pas 3, $[\mathbf{g}](.)$ représente une fonction d'inclusion pour $\mathbf{g}(.)$. La contraction pour $[\mathbf{x}](k)$ au pas 5 peut se faire grâce à l'algorithme de propagation-rétropropagation (voir par exemple [7]). Cette procédure doit être appelée plusieurs fois, tant que des contractions significatives peuvent être effectuées. Lorsque l'équilibre est atteint, des domaines contenant encore de grosses parties invraisemblables peuvent encore subsister. Un partitionnement doit alors être fait pour les éliminer. La stratégie que nous allons suivre est le partitionnement de $[\mathbf{x}](k_0)$, pour un k_0 donné, en petits pavés. L'algorithme résultant est décrit ci-dessous.

STRANGLE(in : k_0, ε, inout : $[\mathbf{x}](k), [\mathbf{y}](k), k \geq 0$)
1 partitionner $[\mathbf{x}](k_0)$ en ℓ pavés $[\mathbf{x}_1](k_0)$,
$\ldots, [\mathbf{x}_\ell](k_0)$ de longueur plus petite que ε;
2 pour $i := 1$ jusqu'à ℓ,
3 pour tout $k \neq k_0$
4 $\{[\mathbf{x}_i](k) := [\mathbf{x}](k);\ [\mathbf{y}_i](k) := [\mathbf{y}](k)\}$;
5 CONTRACT$([\mathbf{x}_i](k), [\mathbf{y}_i](k), k \geq 0)$;
6 pour $k := 0$ jusqu'à $\bar{k}$,
7 $[\mathbf{x}](k) := [\mathbf{x}_1](k) \sqcup \cdots \sqcup [\mathbf{x}_\ell](k)$;
$[\mathbf{y}](k) := [\mathbf{y}_1](k) \sqcup \cdots \sqcup [\mathbf{y}_\ell](k)$;

Même pour des valeurs infiniment petites de ε, les domaines pour $[\mathbf{x}](k)$ et $[\mathbf{y}](k)$ peuvent encore contenir de grosses parties inconsistantes. Le principe de l'algorithme STATEBOUND ci-dessous est d'appeler STRANGLE pour différents k_0 et de faire décroître ε dans le but de contracter les domaines le plus possible.

StateBound(in : ε, inout : $[\mathbf{x}](k), [\mathbf{y}](k), k \geq 0$)
1 faire
2 faire
3 choisir aléatoirement $k_0 \in \{0, 1, \ldots, \bar{k}\}$;
4 Strangle$(k_0, \varepsilon, [\mathbf{x}](k), [\mathbf{y}](k), k \geq 0)$;
5 tant que la contraction est significative;
6 $\varepsilon := \varepsilon/2$;
7 tant que la contraction est significative.

Remarque 4.1 *Par simplicité, dans l'exemple du paragraphe 5, la boucle externe a été remplacée par la boucle "pour $n_1 := 1$ jusqu'à $\bar{n}_1$", où $\bar{n}_1$ est un entier préalablement défini. La boucle interne a été remplacée par la boucle "pour $n_2 := 1$ jusqu'à 30".* ■

5 Exemple

Considérons à nouveau le modèle de Lotka-Volterra. L'équation d'observation est donnée par $y = x_1$. Les données $\hat{y}(t)$ ont été obtenues en simulant le modèle avec $\mathbf{x}(0) = (50, 50)$ et en y ajoutant un bruit blanc avec une distribution uniforme dont le support est l'intervalle $[-1, -1]$. Les domaines $[y](t)$ pour $y(t)$ ont été pris égaux à

$$[y](t) = [\hat{y}(t) - 1.5, \hat{y}(t) + 1.5], \tag{16}$$

pour prendre en compte le fait que les bornes exactes pour les erreurs de mesures sont rarement disponibles. Les domaines *a priori* disponibles pour $\mathbf{x}(t)$ ont été pris égaux à $[0, 100] \times [0, 200]$. Pour $\delta = 0.03$ et $\overline{k} = 200$, Strangle$(k_0 = 0, \varepsilon = 4)$ génère les $[\mathbf{x}](k)$ représentés sur la figure 3 (a), en 1.33 sec. sur un Pentium 300. Notons que pour k faible, les domaines $[\mathbf{x}](k)$ sont plus précis que pour les grands k. Lançons Strangle$\big(k_0 = \bar{k}, \varepsilon = 4\big)$. Pour des grands k les domaines ont maintenant été considérablement contractés (voir la figure 3 (b)). Pour $\bar{n}_1 = 1$ (c'est-à-dire que la boucle externe n'est parcourue qu'une seule fois), StateBound$(\varepsilon = 4)$ génère des domaines de la Figure 3 (c) en 11.4 sec.

6 Conclusion

Cette article étudie, pour la première fois, les applications de l'analyse par intervalles et des méthodes de propagation pour l'estimation d'état de systèmes non-linéaires décrits par des équations différentielles dans un contexte ensembliste. Un algorithme efficace, capable d'enfermer tous les vecteurs d'état vraisemblables à l'intérieur d'ensembles constitués d'unions de pavés, a été proposé. Son efficacité pourrait être améliorée grâce à l'utilisation de méthodes de consistances plus fortes [5] et les techniques de simulation ensemblistes plus précises [8].

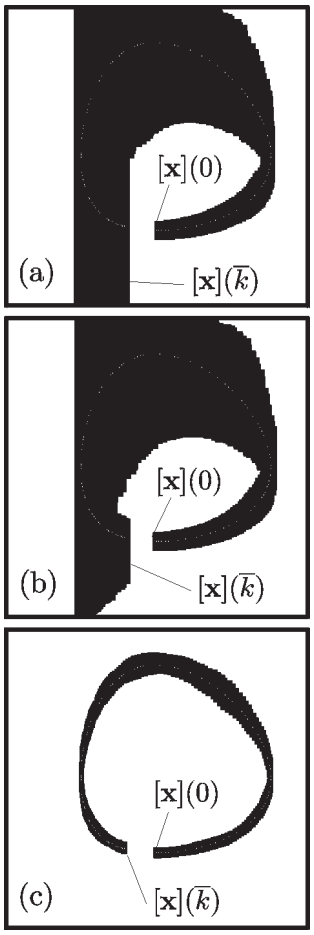

Figure 3: Domaines $[\mathbf{x}](k)$ contractés par StrangLe

Notons que même si la méthode d'enfermement proposée semble efficace pour notre exemple relativement académique, nous n'avons, à ce stade, aucune possibilité d'évaluer la qualité de l'enfermement généré, ni même d'imaginer les performances de l'approche pour des systèmes plus complexes. De plus, l'extension de l'approche pour l'estimation d'état de systèmes incertains n'est a priori immédiate que pour des incertitudes de nature paramétrique. Qu'en est-il pour des perturbations d'une autre nature ? Beaucoup de questions restent donc à résoudre avant de pouvoir espérer traiter des applications réelles.

Les codes C++ (en Builder 3) complets correspondant à l'exemple du paragraphe 5 sont disponibles sur `http:// www.istia.univ-angers.fr /~jaulin/voltera_cpp.zip`.

References

[1] M. Berz, C. Bischof, G. Corliss, and G. A., editors. *Computational Differentiation: Techniques, Applications and Tools*. SIAM, Philadelphia, Penn., 1996.

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

[3] F. L. Chernousko. *State Estimation for Dynamic Systems*. CRC Press, Boca Raton, FL, 1994.

[4] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32(3):281–331, 1987.

[5] Y. Deville, M. Janssen, and P. V. Hentenryck. Consistency techniques in ordinary differential equations. In *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science. Springer Verlag, 1998.

[6] L. Jaulin. Nonlinear bounded-error state estimation of continuous-time systems. to appear in *Automatica*, 2002.

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

[8] R. Lohner. Enclosing the solutions of ordinary initial and boundary value problems. In E. Kaucher, U. Kulisch, and C. Ullrich, editors, *Computer Arithmetic: Scientific Computation and Programming Languages*, pages 255–286. BG Teubner, Stuttgart, Germany, 1987.

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

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

[11] D. Sam-Haroud and B. Faltings. Consistency techniques for continuous constraints. *Constraints*, 1(1-2):85–118, 1996.

[12] F. C. Schweppe. Recursive state estimation: unknown but bounded errors and system inputs. *IEEE Transactions on Automatic Control*, 13(1):22–28, 1968.