Sur quelques idées fausses ayant des conséquences en identification

Eric WALTER, Michel KIEFFER

Laboratoire des Signaux et Systèmes
CNRS – Supélec – Université Paris-Sud 11
91192 Gif-sur-Yvette, France

Eric.Walter@lss.supelec.fr, Michel.Kieffer@lss.supelec.fr
http://www.lss.supelec.fr

Résumé— *L'identification est plus un art qu'une science exacte, et fait appel autant à l'intuition qu'à des techniques très variées. La tradition y pèse son poids, et il est important de remettre en question des coutumes dont les justifications sont parfois essentiellement historiques. Un premier exemple est le rôle quasi exclusif joué par la minimisation de critères quadratiques, très liée à une simplification des calculs qui n'a plus guère d'intérêt et à des hypothèses sur le caractère gaussien des erreurs de mesure souvent contredites par les faits. Un deuxième exemple est l'utilisation d'approximation par différences finies pour les calculs de gradient, en ignorant l'existence de techniques exactes et plus économes en calculs pourtant largement inspirées des travaux des automaticiens. Un troisième exemple est le recours quasi exclusif aux techniques locales pour estimer des paramètres dès que l'on sort du cadre important mais très particulier de la minimisation d'une fonction coût quadratique en les paramètres à estimer. Nous donnons quelques pistes pour explorer d'autres voies.*

Mots-clés— *calcul par intervalles, code adjoint, différentiation automatique, données aberrantes, estimation robuste, optimisation globale.*

I. INTRODUCTION

L'identification est plus un art qu'une science exacte, et fait appel autant à l'intuition qu'à des techniques très variées. La tradition y pèse son poids, et il est important de remettre en question des coutumes dont les justifications sont parfois essentiellement historiques. Un premier exemple est le rôle quasi exclusif joué en estimation depuis les travaux de Gauss et Legendre par la minimisation de fonctions coût quadratiques, de type

$$J(\mathbf{p}) = \sum_{i=1}^{n_t} w(t_i)[y_m(t_i, \mathbf{p}) - y(t_i)]^2, \qquad (1)$$

où $\mathbf{p}$ est le vecteur des n_p paramètres à estimer, les $y(t_i)$ sont les données recueillies sur le processus à modéliser, $y_m(t_i, \mathbf{p})$ est la sortie du modèle de paramètres $\mathbf{p}$ et les $w(t_i)$ sont des coefficients de pondération positifs à choisir.

Le choix (1) n'allait pas de soi. Dès 1632, Galilée utilise une fonction coût de type

$$J(\mathbf{p}) = \sum_{i=1}^{n_t} w(t_i) \left| y_m(t_i, \mathbf{p}) - y(t_i) \right|, \qquad (2)$$

qui retient également l'attention de Boscovitch et Laplace [4]. Ce dernier s'intéresse aussi dès 1786 à

$$J(\mathbf{p}) = \max_{i=1}^{n_t} w(t_i) \left| y_m(t_i, \mathbf{p}) - y(t_i) \right|. \qquad (3)$$

Or la fonction coût (1) est aujourd'hui presque toujours utilisée. On peut y voir deux raisons principales. La première est que la valeur optimale $\widehat{\mathbf{p}}$ de $\mathbf{p}$ au sens de $J(\cdot)$ est donnée par une formule explicite très simple, dans le cas particulier où $y_m(t_i, \mathbf{p})$ est linéaire en $\mathbf{p}$. C'était un avantage essentiel à l'époque des calculs à la main, mais cela ne peut plus guère servir de justification aujourd'hui... La seconde raison correspond à des hypothèses sur le caractère gaussien des erreurs de mesure, souvent contredites par les faits. La section II parlera des conséquences de cette violation des hypothèses et des outils que proposent les statistiques robustes pour y remédier.

Quand on ne dispose pas d'une expression explicite pour $\widehat{\mathbf{p}}$, on fait souvent appel à des algorithmes itératifs qui exploitent pour se déplacer dans l'espace des paramètres la valeur prise par le gradient de J au point courant à l'itération k

$$\mathbf{g}(\widehat{\mathbf{p}}^k) = \frac{\partial J}{\partial \mathbf{p}}(\widehat{\mathbf{p}}^k). \qquad (4)$$

Comme $J(\widehat{\mathbf{p}}^k)$ est en général évalué par exécution d'un programme informatique qui peut être relativement complexe, la coutume est d'évaluer $\mathbf{g}(\widehat{\mathbf{p}}^k)$ via une approximation par différences finies. La méthode du code adjoint, largement inspirée des travaux des automaticiens en commande optimale et que nous présenterons en section III, a pourtant deux avantages :

- c'est une *méthode exacte*, qui élimine l'erreur de méthode de l'approximation par différence finie ;
- elle est *remarquablement économe en calcul*, puisqu'il suffit pour évaluer $\mathbf{g}(\widehat{\mathbf{p}}^k)$ d'exécuter une seule fois un code déduit de celui servant à évaluer $J(\widehat{\mathbf{p}}^k)$ et dont la complexité est du même ordre de grandeur, et ceci quelle que soit la dimension de $\mathbf{p}$.

Un troisième exemple de coutume est le recours quasi exclusif aux techniques itératives locales pour estimer des paramètres dès que l'on sort du cadre important mais très particulier de la minimisation d'une fonction coût quadratique en $\mathbf{p}$. La section IV présentera des outils à base d'analyse par intervalles qui permettent d'éviter les limitations bien connues des approches locales et qui, contrairement aux techniques d'exploration aléatoire, sont à même de prouver leurs assertions. La section V montrera que l'analyse par intervalles permet également l'intégration numérique garantie de systèmes d'équations différentielles non-linéaires et incertains.

II. Les erreurs sont gaussiennes

La plupart de ceux qui se préoccupent de la nature statistique du bruit corrompant les données qu'ils vont utiliser pour identifier les paramètres de leur modèle font l'hypothèse d'un bruit blanc additif gaussien. Le mot gaussien est d'ailleurs souvent remplacé par *normal*, ce qui indique bien un état d'esprit que résume la boutade suivante, attribuée au physicien français Gabriel Lippmann (prix Nobel de physique en 1908 pour ses travaux sur la photographie couleur) :

Tout le monde croit que les erreurs sont normales, les mathématiciens parce qu'ils pensent que c'est un fait expérimentalement établi et les expérimentateurs parce qu'ils croient que c'est un théorème.

Le théorème *de la limite centrale* est souvent invoqué pour rassurer ceux qui viendraient à douter de ce caractère gaussien. Il dit que la somme de n variables aléatoires indépendantes et de même distribution (de moyenne μ et de variance σ^2 finie et non nulle) tend à être distribuée suivant une loi gaussienne de moyenne $n\mu$ et de variance $n\sigma^2$.

Une fois admis ce postulat sur la nature gaussienne des erreurs de mesure, la méthode du maximum de vraisemblance permet (sous certaines hypothèses supplémentaires) d'en déduire une procédure d'estimation de paramètres par minimisation d'une fonction coût quadratique qui est quasi universellement utilisée.

Or l'hypothèse gaussienne est au mieux une approximation de la réalité. Des raisons multiples (caractère approximatif du modèle utilisé se traduisant par des erreurs déterministes, pannes momentanées de capteurs produisant des données aberrantes, effet de non linéarités...) font que cette hypothèse est souvent et clairement contredite par les faits. Comme le disait J.W. Tuckey en 1960 :

A tacit hope in ignoring deviations from ideal models was that they would not matter, that statistical procedures which were optimal under the strict model would still be approximately optimal under the approximate model. Unfortunately, it turned out that this hope was often drastically wrong ; even mild deviations often have much larger effects than were anticipated by most statisticians.

Il est ainsi facile de constater que la présence, par exemple, d'une faible proportion de données aberrantes suffit à entraîner des conséquences catastrophiques sur les paramètres estimés. Le caractère quadratique de la fonction coût donne en effet un poids disproportionné à ces données atypiques. La prise de conscience de cette fragilité de l'inférence statistique classique a donné lieu au développement de méthodes statistiques robustes, avec en particulier les travaux de Pearson [23], Box et Draper [3], Tukey [30], Huber [11], Hampel [8], [9], Rousseeuw [25], [26], Schweppe [17], [27], et Mili [19], [18].

Aujourd'hui encore, l'usage des statistiques robustes est loin d'être universellement répandu même chez les statisticiens. Elles ont en effet trois inconvénients principaux.

Le premier est qu'elles conduisent à diminuer l'influence de données particulièrement informatives, de sorte que l'efficacité des estimateurs robustes est moindre que celle des estimateurs non robustes, quand par chance les données vérifient les hypothèses sur lesquelles ces estimateurs non robustes sont fondés. D'où l'idée qui consiste à dire qu'il suffit de s'arranger pour collecter ses données proprement, afin de ne pas avoir à affronter les conséquences de la pollution de sa base de données par des données aberrantes. À ceci on peut répondre que ce sont parfois les données aberrantes qui constituent la partie la plus intéressante de la base, que l'élimination manuelle des données aberrantes est d'autant plus difficile que la base de donnée est grande (or les masses de données à traiter sont de taille sans cesse croissante) et enfin que si les erreurs sont distribuées suivant une loi à queue plus lourde que celle de la loi gaussienne (par exemple suivant une loi de Laplace) il sera parfaitement impossible de détecter a priori celles qu'il convient d'éliminer. Nous avons donc besoin de méthodes qui ne s'écroulent pas dès que les données ne satisfont pas les hypothèses usuelles, quitte à comparer les résultats fournis par ces méthodes à ceux fournis par les estimateurs classiques et à préférer ces derniers quand il n'y a pas contradiction. Notons que les tests statistiques classiques du caractère gaussien fonctionnent mal en présence de données aberrantes, de sorte qu'il est dangereux de s'en remettre à eux pour décider si un estimateur robuste doit être invoqué.

Le second inconvénient est que l'optimisation des fonctions coût liées à ces estimateurs robustes est plus délicate que dans le cas d'un critère quadratique (non différentiabilité, multimodalité, etc.). Il est, par exemple, sans espoir de tenter la minimisation d'une fonction coût du type (2) (pourtant beaucoup plus robuste à des données aberrantes qu'une fonction de type (1)) en invoquant un algorithme fondé sur un développement limité du coût (gradient, Newton, Gauss-Newton, quasi-Newton, gradients conjugués, etc.). Bien que la fonction coût (2) soit dérivable presque partout, ces algorithmes se ruent sur les points où elle ne l'est pas pour y rester bloqués. L'existence d'algorithmes et de logiciels spécifiques rend cette objection beaucoup moins pertinente qu'autrefois.

Le dernier inconvénient, c'est que le développement des méthodes d'estimation robuste n'est pas terminé, de sorte que le corpus des méthodes utilisables évolue et que certaines questions demeurent ouvertes.

Quoi qu'il en soit, les outils actuels permettent de donner des réponses au moins partielles à des questions aussi essentielles que [9]

- Les données sont-elles unanimes dans leur message ?
- Si non, que dit la majorité, et où est la minorité ?
- Quelles sont les données dont l'influence est telle qu'elles doivent bénéficier d'une attention particulière (concept de *point levier*) ?
- Quel est le degré de protection offert contre les données aberrantes (concept de *point de rupture*) ?
- Quelle est la perte d'efficacité subie par rapport à une méthode non robuste ?
- Comment profiter des degrés de liberté dans les expériences réalisables pour augmenter la robustesse de l'estimation ?

Remarquons au passage que ce qui a été dit de la sensibilité des estimateurs classiques à des déviations par rapport à l'hypothèse d'erreurs gaussiennes est également vrai des déviations par rapport à l'hypothèse d'indépendance des erreurs, même si les techniques d'estimation robuste semblent moins développées dans ce domaine.

Exemple 1: Considérons un ensemble de mesures $y(i)$, $i = 0 \dots 20$, que l'on cherche à décrire à l'aide du modèle $y_{\mathrm{m}}(i, \mathbf{p}) = p_1 + p_2 i$. Un premier jeu de données sans bruit est généré par le modèle en prenant $\mathbf{p}^* = (2, 1)^{\mathrm{T}}$ puis ces données sont corrompues par un bruit gaussien additif centré de variance unité. À partir des données bruitées résultantes, l'estimateur au sens des moindres carrés (MC) et celui minimisant la somme des valeurs absolues (VA) des erreurs fournissent dans ce premier cas des résultats proches : $\widehat{\mathbf{p}}_{\mathrm{MC}} = (1.87, 1.03)^{\mathrm{T}}$ et $\widehat{\mathbf{p}}_{\mathrm{VA}} = (2.13, 1.00)^{\mathrm{T}}$, voir la figure 1. Un second jeu de données est obtenu à partir du premier en

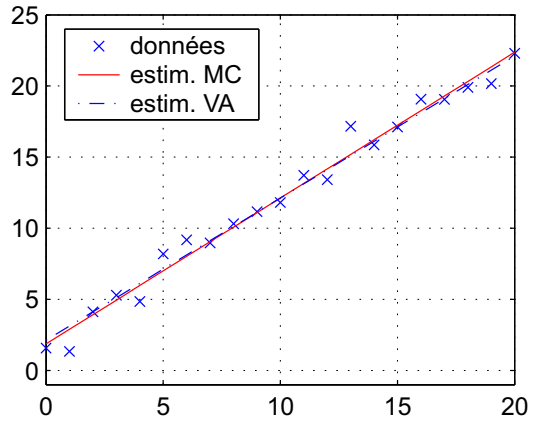

Fig. 1. Estimation de paramètres en l'absence de données aberrantes ; données et sorties des modèles obtenues par minimisation des fonctions coût (1) et (2)

forçant à zéro les mesures $y(14)$ à $y(16)$ pour simuler une panne de capteur. L'estimateur MC se révèle moins robuste à l'égard de ces données aberrantes que l'estimateur VA. En effet, $\widehat{\mathbf{p}}_{\mathrm{MC}} = (2.82, 0.68)^{\mathrm{T}}$ et $\widehat{\mathbf{p}}_{\mathrm{VA}} = (2.15, 0.99)^{\mathrm{T}}$, voir la figure 2. ♢

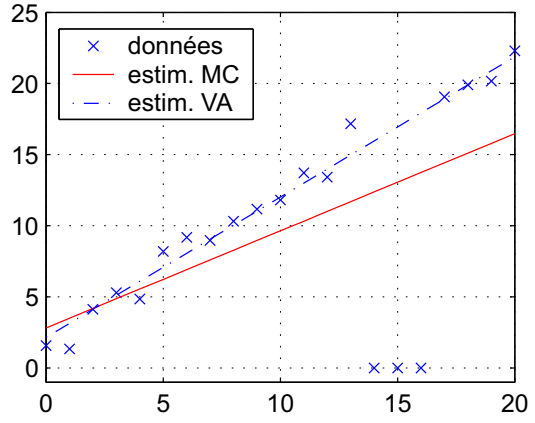

Fig. 2. Estimation de paramètres en présence de données aberrantes ; données et sorties des modèles obtenues par minimisation des fonctions coût (1) et (2)

III. Le calcul de gradient se fait par différences finies

De très nombreux algorithmes itératifs utilisés en identification exploitent la valeur prise par le gradient de la fonction coût au point courant dans l'espace des paramètres. Cette évaluation représente alors une part importante de l'effort de calcul, et doit donc être simplifiée autant qu'il est possible. Sauf cas très particulier, on ne dispose pas d'une expression analytique de la fonction coût, qui est souvent calculée par un programme complexe. La démarche standard consiste alors à utiliser une approximation par différences finies

$$\frac{\partial J(\mathbf{p})}{\partial p_i} = \frac{1}{\Delta p_i}[J(\mathbf{p} + \Delta\mathbf{p}_i) - J(\mathbf{p})], i = 1 \dots n_{\mathrm{p}}, \quad (5)$$

où $\Delta\mathbf{p}_i$ est un vecteur dont toutes les composantes sont nulles sauf la $i$ème, qui est égale à Δp_i. À supposer que Δp_i ait été bien choisi (ce qui n'a rien d'évident), ceci nécessite $n_{\mathrm{p}} + 1$ évaluations du coût et donc autant de simulations du modèle, ce qui peut se révéler très lourd si le modèle est complexe et le nombre de paramètres élevé. S'il faut procéder à un réglage de Δp_i la complexité des calculs augmente encore. Comme il est impossible de faire tendre Δp_i vers zéro à cause des problèmes numériques posés par le calcul de différences entre des nombres flottants très proches, le résultat est toujours approximatif. Une première approche évitant cet écueil consiste, quand c'est possible, à exprimer le gradient du critère en fonction des fonctions de sensibilité du premier ordre de la sortie du modèle, et à évaluer ces dernières. Considérons, par exemple, le modèle constitué par l'équation d'état

$$\mathbf{x}' = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, \mathbf{p}), \quad \mathbf{x}(0) = \mathbf{x}_0(\mathbf{p}). \quad (6)$$

Supposons, aussi pour simplifier, le vecteur des sorties du modèle linéaire en l'état :

$$\mathbf{y}_{\mathrm{m}}(t, \mathbf{p}) = \mathbf{C}(\mathbf{p})\mathbf{x}(t, \mathbf{p}). \quad (7)$$

La sensibilité de $\mathbf{y}_{\mathrm{m}}$ par rapport au paramètre p_i est alors donnée par

$$\frac{\partial \mathbf{y}_{\mathrm{m}}}{\partial p_i} = \frac{\partial \mathbf{C}}{\partial p_i}\mathbf{x} + \mathbf{C}\frac{\partial \mathbf{x}}{\partial p_i}, \quad (8)$$

et est donc facile à calculer à partir de la sensibilité de $\mathbf{x}$ par rapport à p_i, que nous noterons $\mathbf{s}_i$. En différentiant (6) par rapport à p_i, nous obtenons

$$\frac{\mathrm{d}\mathbf{s}_i}{\mathrm{d}t} = \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{x}^{\mathrm{T}}}\mathbf{s}_i + \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{p})}{\partial p_i}, \mathbf{s}_i(0) = \frac{\partial \mathbf{x}_0(\mathbf{p})}{\partial p_i}, i = 1 \dots n_{\mathrm{p}}. \quad (9)$$

Les n_{p} équations ainsi obtenues sont *indépendantes* (elles peuvent donc être simulées l'une après l'autre), *linéaires* (mais variables dans le temps à cause de la dépendance en l'état), et *leur partie homogène est identique*. Ces propriétés peuvent être exploitées pour simuler l'évolution de l'état et de ses n_{p} fonctions de sensibilité au premier ordre beaucoup plus vite qu'avec l'approche par différence finie [31], [2], même si ceci demande (sauf dans le cas particulier où les équations d'état sont linéaires et les conditions initiales nulles) $n_{\mathrm{p}} + 1$

simulations, c'est-à-dire le même nombre qu'avec l'approche par différences finies.

L'*approche par code adjoint* [28], [6] va permettre de dépasser cette limitation, et sera d'autant plus intéressante que la dimension de $\mathbf{p}$ est grande. Elle s'applique à tout coût évalué par un programme, pourvu que ce programme calcule ce coût d'une façon différentiable par rapport aux paramètres. Appelons *code direct* le programme qui évalue la valeur numérique du coût J pour une valeur numérique de ses variables d'entrée (vecteur de paramètres $\mathbf{p}$, vecteur des données expérimentales, *etc.*). Notons $\mathbf{v}$ le vecteur qui contient toutes les variables du code, que celles-ci soient des variables d'entrée (les variables indépendantes) ou des résultats de calculs (les variables dépendantes). Ce vecteur peut être de très grande taille, et constitue seulement un intermédiaire de raisonnement qui ne recevra pas de matérialisation informatique. À un moment quelconque de l'exécution du code direct, la valeur de $\mathbf{v}$ peut être vue comme l'état de ce code, et l'exécution d'une instruction d'assignation (disons la $k$ième) se traduira par la modification d'une des composantes de ce vecteur d'état (disons la $\mu(k)$ième), ce que l'on peut écrire

$$v_{\mu(k)} = \varphi_k(\{v_i \mid i \in \mathbb{I}_k\}), \tag{10}$$

où $\mathbb{I}_k$ est l'ensemble des indices des composantes de $\mathbf{v}$ qui sont des arguments de φ_k. Globalement, on peut donc voir le code direct comme un système décrit par une *équation d'état non linéaire à temps discret*

$$\mathbf{v}(k) = \mathbf{\Phi}_k[\mathbf{v}(k-1)], k = 1 \dots f, \tag{11}$$

où le rôle du temps est joué par le passage d'une instruction d'assignation à celle qui sera exécutée ensuite, et où

$$[\mathbf{\Phi}_k(\mathbf{v})]_i = v_i, \forall i \neq \mu(k), \tag{12}$$

$$[\mathbf{\Phi}_k(\mathbf{v})]_{\mu(k)} = \varphi_k(\{v_i \mid i \in \mathbb{I}_k\}). \tag{13}$$

Notons que l'instruction associée à une valeur donnée de k dépend de l'ordre dans lequel les instructions sont exécutées. Les branchements demanderont donc une attention particulière. De façon arbitraire, et sans influence sur le résultat final, convenons de placer dans les premières composantes de $\mathbf{v}(0)$ les composantes de $\mathbf{p}$ puis celles des autres entrées du code direct. La valeur des autres composantes de $\mathbf{v}(0)$ n'a pas d'importance puis les variables dépendantes seront calculées par le code direct. Convenons également qu'à la fin de l'exécution du code direct la valeur $J(\mathbf{p})$ du coût sera placée dans la dernière composante de l'état, de sorte que

$$J(\mathbf{p}) = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \mathbf{v}(f). \tag{14}$$

La règle des dérivations en chaîne permet d'écrire

$$\frac{\partial J}{\partial \mathbf{p}} = \frac{\partial \mathbf{v}^{\mathrm{T}}(0)}{\partial \mathbf{p}} \frac{\partial \mathbf{\Phi}_1^{\mathrm{T}}}{\partial \mathbf{v}(0)} \cdots \frac{\partial \mathbf{\Phi}_{f-1}^{\mathrm{T}}}{\partial \mathbf{v}(f-2)} \frac{\partial \mathbf{\Phi}_f^{\mathrm{T}}}{\partial \mathbf{v}(f-1)} \frac{\partial J}{\partial \mathbf{v}(f)} \tag{15}$$

On peut envisager d'effectuer les calculs correspondant à (15) de multiples façons, et en particulier de la gauche vers la droite ou de la droite vers la gauche. La façon la plus efficace de procéder en terme de nombre d'opérations consiste à partir de la droite car on ne fait alors que des produits de matrices par des vecteurs. Posons

$$\mathbf{d}(f) = \frac{\partial J}{\partial \mathbf{v}(f)} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \tag{16}$$

et calculons en nous déplaçant vers la gauche suivant la formule

$$\mathbf{d}(k-1) = \frac{\partial \mathbf{\Phi}_k^{\mathrm{T}}}{\partial \mathbf{v}(k-1)} \mathbf{d}(k), k = f \dots 1, \tag{17}$$

ce que l'on peut interpréter comme une récurrence à temps rétrograde. Le vecteur $\mathbf{d}$ ainsi calculé est connu sous le nom d'*état adjoint*. Compte tenu de la façon dont la fonction $\mathbf{\Phi}_k$ a été construite, $\partial \mathbf{\Phi}_k^{\mathrm{T}} / \partial \mathbf{v}(k-1)$ est une matrice identité dont la $\mu(k)$ième colonne a été remplacée par le vecteur $\partial \varphi_k / \partial \mathbf{v}(k-1)$. Ceci entraîne que

$$d_i(k-1) = d_i(k) + \frac{\partial \varphi_k}{\partial v_i(k-1)} d_{\mu(k)}(k) \text{ si } i \neq \mu(k), \tag{18}$$

$$d_{\mu(k)}(k-1) = \frac{\partial \varphi_k}{\partial v_{\mu(k)}(k-1)} d_{\mu(k)}. \tag{19}$$

Partant de l'état adjoint final $\mathbf{d}(f)$, on peut ainsi par récurrence calculer l'état adjoint initial $\mathbf{d}(0)$. D'après (15)

$$\frac{\partial J}{\partial \mathbf{p}} = \frac{\partial \mathbf{v}^{\mathrm{T}}(0)}{\partial \mathbf{p}} \mathbf{d}(0). \tag{20}$$

Or, avec nos conventions,

$$\frac{\partial \mathbf{v}^{\mathrm{T}}(0)}{\partial \mathbf{p}} = \begin{bmatrix} \mathbf{I}_{n_{\mathrm{p}}} & \mathbf{0} \end{bmatrix}. \tag{21}$$

Ceci revient à dire que le gradient $\mathbf{g}(\mathbf{p})$ est contenu dans les n_{p} premières composantes de $\mathbf{d}(0)$. Les composantes suivantes de $\mathbf{d}(0)$ contiennent les dérivées partielles de la fonction coût par rapport à toutes les autres entrées du code direct (les données, par exemple), qui sont ainsi obtenues en prime, sans aucun calcul complémentaire. Notons qu'il n'est pas nécessaire de stocker les valeurs successives de l'état adjoint, car seul $\mathbf{d}(0)$ nous sera utile. C'est pourquoi nous n'indexerons pas temporellement les composantes de $\mathbf{d}$.

La $k$ième instruction d'affectation du code direct

$$v_{\mu(k)} = \varphi_k(\{v_i \mid i \in \mathbb{I}_k\}), \tag{22}$$

se traduit par les instructions adjointes suivantes, dont l'ordre doit être respecté :

$$\textbf{for all } i \in \mathbb{I}_k, i \neq \mu(k), \textbf{ do } \{d_i = d_i + \frac{\partial \varphi_k}{\partial v_i} d_{\mu(k)}\}, \tag{23}$$

$$d_{\mu(k)} = \frac{\partial \varphi_k}{\partial v_{\mu(k)}} d_{\mu(k)}. \tag{24}$$

Supposons, par exemple, que la $k$ième instruction d'affectation du code direct soit

$$\text{cout} = \text{cout} + [y(t) - y_{\mathrm{m}}(t)]^2, \tag{25}$$

de sorte que

$$\varphi_k[\text{cout}, y(t), y_\text{m}(t)] = \text{cout} + [y(t) - y_\text{m}(t)]^2. \quad (26)$$

Notons dcout, $\text{d}y(t)$ et $\text{d}y_\text{m}(t)$ les variables adjointes associées aux variables arguments de φ_k, soit cout, $y(t)$ et $y_\text{m}(t)$. En appliquant (23) et (24) à (26), on obtient les instructions adjointes

$$\text{d}y(t) = \text{d}y(t) + \frac{\partial \varphi_k}{\partial y(t)}\text{dcout}, \quad (27)$$

$$\text{d}y_\text{m}(t) = \text{d}y_\text{m}(t) + \frac{\partial \varphi_k}{\partial y_\text{m}(t)}\text{dcout}, \quad (28)$$

$$\text{dcout} = \text{dcout}. \quad (29)$$

Si on élimine (29), qui est bien sûr superflue, il reste compte tenu de (26)

$$\text{d}y(t) = \text{d}y(t) + 2[y(t) - y_\text{m}(t)]\text{dcout}, \quad (30)$$

$$\text{d}y_\text{m}(t) = \text{d}y_\text{m}(t) - 2[y(t) - y_\text{m}(t)]\text{dcout}. \quad (31)$$

Les instructions de branchement doivent elles aussi être dualisées. Les boucles (do, for, while) du code direct se transforment dans le code adjoint en des boucles où les instructions adjointes sont exécutées dans l'ordre inverse de leurs contreparties du code direct. En présence de branchements conditionnels, il faudra mémoriser le chemin suivi lors de l'exécution du code direct pour être en mesure d'exécuter les instructions du code adjoint dans l'ordre approprié. C'est ainsi que la présence dans le code direct de l'instruction de branchement

if (C) **then** {code A} **else** {code B} (32)

se traduira dans le code adjoint par l'instruction de branchement

if (C) **then** {adjoint de A} **else** {adjoint de B} (33)

La construction du code adjoint peut être résumée ainsi :

- associer à chaque variable du code direct une variable adjointe (choisir une convention permettant facilement de reconnaître qu'une variable est adjointe et à quelle variable du code direct elle correspond),
- initialiser toutes les variables adjointes à zéro, sauf celle associée à la valeur du coût, qui sera initialisée à un,
- dualiser les instructions du code direct dans l'ordre inverse de celui de leur exécution, ce qui suppose d'inverser les boucles et de tenir compte des branchements conditionnels.

Le code direct est exécuté en premier, en prenant soin de mémoriser les branches exécutées et toutes les valeurs numériques dont la connaissance sera nécessaire à l'exécution du code adjoint (c'est-à-dire seulement celles qui interviennent de façon non linéaire comme arguments d'instructions d'affectation du code direct). Le code direct produit la valeur du coût $J(\mathbf{p})$. Le code adjoint est alors exécuté et produit la valeur du gradient $\mathbf{g}(\mathbf{p})$ de ce coût. (Les valeurs des composantes de $\mathbf{g}(\mathbf{p})$ sont données par les valeurs prises en fin d'exécution du code adjoint par les variables adjointes associées aux paramètres.)

Il faut être très soigneux dans la préparation du code adjoint, car des erreurs d'apparence anodine peuvent avoir des résultats désastreux. Une bonne pratique [29] est de

- développer le code adjoint à partir du code direct (et pas à partir des équations que ce dernier met en œuvre),
- donner au code adjoint une structure lisible (chaque variable ou sous-programme du code direct doit avoir sa contrepartie dans le code adjoint),
- ne jamais modifier le code direct sans actualiser le code adjoint.

La propriété suivante peut permettre de détecter des erreurs de dualisation. Soit $\delta\mathbf{v}(k)$ l'état de l'équation (11) linéarisée au voisinage de sa trajectoire nominale. Le produit scalaire de $\delta\mathbf{v}(k)$ avec l'état adjoint $\mathbf{d}(k)$ reste constant le long de cette trajectoire, de sorte que

$$\mathbf{d}^\text{T}(k+1)\delta\mathbf{v}(k+1) = \mathbf{d}^\text{T}(k)\delta\mathbf{v}(k), k = 0 \ldots f-1. \quad (34)$$

Considérons, par exemple [33], [34], le code direct

```
cout = 0;
ym(0) = p2;
% boucle directe
for k = 1 to nt do {
ym(k) = p1*ym(k-1);
cout = cout + [y(k) - ym(k)]^2;
}.
```

Il évalue une fonction coût qui dépend de deux paramètres et de données expérimentales $y(k),\ k = 1 \ldots n_\text{t}$. La technique que nous venons de décrire génère, de façon parfaitement automatisable, le code adjoint suivant

```
% initialisation des variables adjointes
dcout = 1;
for k = 1 to nt do {
dy(k) = 0;
dym(k) = 0;
}
dp1 = 0;
dp2 = 0;
% boucle rétrograde
for k = nt down to 1 do {
% dualisation de la seconde
% instruction de la boucle directe
dy(k) = dy(k) + 2*[y(k) - ym(k)]*dcout;
dym(k) = dym(k) - 2*[y(k) - ym(k)]*dcout;
dcout = dcout;
% dualisation de la première
% instruction de la boucle directe
dym(k-1) = dym(k-1) + p1*dym(k);
dp1 = dp1 + ym(k-1)*dym(k);
dym(k) = 0;
}
% dualisation de l'instruction
% qui précède la boucle directe
dp2 = dp2 + dym(0);
dym(0) = 0.
```

La valeur fournie pour le gradient, contenue dans `dp1` et `dp2`, est exacte (aux erreurs d'arrondi introduites par l'ordinateur

près). Il n'y a donc pas d'erreur de méthode (contrairement à ce qui se passait avec l'approche par différences finies). Le gradient est évalué en deux exécutions (une du code direct et une du code adjoint), et ceci quelque soit le nombre des paramètres (contrairement à ce qui se passait avec l'approche par fonctions de sensibilité). Il faut pour cela disposer du source du code direct, et de suffisamment de mémoire pour stocker les valeurs calculées par le code direct qui sont nécessaires à l'exécution du code adjoint.

IV. L'ESTIMATION NON LINÉAIRE PASSE PAR DES TECHNIQUES LOCALES

L'estimée des paramètres est en général définie par référence à un critère

$$\widehat{\mathbf{p}} = \arg\min_{\mathbf{p}\in\mathbb{P}} J(\mathbf{p}). \tag{35}$$

Contrairement à ce que la notation précédente peut laisser à penser, rien ne garantit que $\widehat{\mathbf{p}}$ soit unique. On cherche donc l'ensemble des arguments du minimum global de $J(\cdot)$ sur $\mathbb{P}$, que nous appellerons les *minimiseurs globaux* de $J(\cdot)$. Quand on ne dispose pas d'une expression analytique pour la valeur de $\widehat{\mathbf{p}}$, il est d'usage d'utiliser des techniques itératives locales, qui partant d'une estimée initiale $\widehat{\mathbf{p}}^0$ calculent une suite de meilleures valeurs, au sens ou

$$J(\widehat{\mathbf{p}}^{k+1}) < J(\widehat{\mathbf{p}}^k). \tag{36}$$

Cette approche pose des problèmes bien connus :

- comment choisir $\widehat{\mathbf{p}}^0$?
- quand arrêter d'itérer ?
- comment ne pas se faire piéger sur des minimiseurs locaux ?
- peut-il y avoir plusieurs minimiseurs globaux (plusieurs valeurs de $\widehat{\mathbf{p}}$ également optimales), et si oui comment les trouver toutes ?

Si les techniques d'exploration aléatoire (algorithmes génétiques ou recuit simulé, par exemple) contribuent à répondre à ces questions, elles ne sauraient garantir les résultats qu'elles produisent en temps fini. Les techniques à base d'analyse par intervalles dont nous allons maintenant parler n'ont pas cet inconvénient et permettent dans certains cas de fournir en temps fini des résultats garantis, y compris pour des problèmes impliquant des modèles décrits par des équations différentielles non linéaires pour lesquelles on ne dispose pas de solution analytique. La présence d'incertitude sur le modèle peut également être prise en compte.

A. Quelques notions de calcul par intervalles

Un intervalle réel $[x] = [\underline{x}, \overline{x}]$ est un ensemble continu de nombres réels, contenant une infinité de nombres non représentables par un ordinateur. L'intervalle $[x]$ peut malgré tout être parfaitement représenté, dès lors que ses bornes $\underline{x}$ et $\overline{x}$ le sont. Il est alors possible d'étendre l'ensemble des opérations arithmétiques $\{+, -, \times, /\}$ aux intervalles de la manière suivante

$$\forall \circ \in \{+, -, \times, /\}, \ [x] \circ [y] = \{x \circ y \mid x \in [x], y \in [y]\}. \tag{37}$$

Ainsi,

$$\begin{aligned}
[x] + [y] &= \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right], && (38)\\
[x] - [y] &= \left[\underline{x} - \overline{y}, \overline{x} - \underline{y}\right], \\
[x] \times [y] &= \left[\min\left(\underline{x}.\underline{y}, \overline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\overline{y}\right), \right. \\
&\quad \left. \max\left(\underline{x}.\underline{y}, \overline{x}.\underline{y}, \underline{x}.\overline{y}, \overline{x}.\overline{y}\right)\right], && (39)\\
[x] / [y] &= [x] \times \left[1/\overline{y}, 1/\underline{y}\right], \text{ lorsque } 0 \notin [y]. && (40)
\end{aligned}$$

Il est tout à fait possible que les bornes de l'intervalle résultat ne soient pas représentables par un ordinateur. Dans ce cas, le nombre représentable immédiatement inférieur ou immédiatement supérieur est utilisé, suivant que l'on cherche à calculer la borne inférieure ou supérieure du résultat. Cette possibilité d'approximation extérieure est mise en œuvre grâce au contrôle de la direction d'arrondi proposé par la norme IEEE 754 [12]. Ainsi, l'intervalle résultat fourni par un ordinateur *contiendra toujours* l'intervalle qui aurait pu être obtenu avec une machine disposant d'une précision infinie. Ceci permet de manipuler des intervalles sur un ordinateur et de *démontrer numériquement* certaines propriétés de fonctions sur un intervalle, comme l'illustre l'exemple suivant.

Exemple 2: Considérons la fonction $f(x) = x^4 - 4x^2$, à minimiser sur l'intervalle $[-10, 10]$. Supposons que f ait été évaluée en $x = 1$ et que $f([2, 4])$, l'ensemble des valeurs prises par f sur l'intervalle $[2, 4]$, soit disponible

$$f(1) = -3 \quad \text{et} \quad f([2, 4]) = [0, 192]. \tag{41}$$

Ces résultats permettent de constater que $f(x) \geqslant 0$ pour tout $x \in [2, 4]$. Comme $f(1) = -3$, le minimum de f sur $[-10, 10]$ ne peut être supérieur à -3. Il est ainsi possible de *démontrer* que l'intervalle $[2, 4]$ ne contient pas de minimiseur global. ♢

C'est le type d'argument présenté dans l'exemple 2 qui est mis en œuvre dans les algorithmes d'optimisation globale utilisant l'analyse par intervalles. La principale difficulté d'une démonstration numérique repose sur l'évaluation de l'ensemble des valeurs prises par une fonction sur un intervalle. Ce problème est résolu grâce aux *fonctions d'inclusion*.

Soit $f(\cdot)$ une fonction réelle définie sur un domaine $\mathcal{D} \subset \mathbb{R}$. L'ensemble des valeurs prises par cette fonction sur un intervalle $[x] \subset \mathcal{D}$ est donné par

$$f([x]) = \{f(x) \mid x \in [x]\}. \tag{42}$$

Pour des fonctions élémentaires monotones, cet ensemble est très simple à caractériser ; ainsi

$$\exp([x]) = [\exp(\underline{x}), \exp(\overline{x})]. \tag{43}$$

Par contre, les fonctions élémentaires non monotones, telles que l'élévation à une puissance positive paire ou le sinus, nécessitent la mise en œuvre d'un algorithme pour obtenir l'ensemble exact des valeurs prises sur un intervalle quelconque. Ainsi, pour tout $k \in \mathbb{N}$,

$$[x]^{2k} = \begin{cases} \left[0, \max\left(\underline{x}^{2k}, \overline{x}^{2k}\right)\right] & \text{si } 0 \in [x] \\ \left[\min\left(\underline{x}^{2k}, \overline{x}^{2k}\right), \max\left(\underline{x}^{2k}, \overline{x}^{2k}\right)\right] & \text{sinon} \end{cases}$$

Pour plus de détails, voir par exemple [7], [14].

Pour une fonction $f(\cdot)$ quelconque, il n'est en général pas possible de décrire exactement l'ensemble $f([x])$ pour un intervalle $[x]$ quelconque. Dans certaines situations, cet ensemble peut même être composé de plusieurs sous-ensembles disjoints. C'est pour contourner cette difficulté que les fonctions d'inclusion ont été introduites. Une fonction d'inclusion $[f](\cdot)$ associée à une fonction $f(\cdot)$ est une fonction à valeurs *intervalles*, qui pour tout $[x] \subset \mathcal{D}$ satisfait

$$f([x]) \subset [f]([x]). \tag{44}$$

La fonction d'inclusion permet ainsi d'approximer un ensemble que l'on ne sait pas forcément calculer par un intervalle que l'on sait calculer et qui le contient. Lorsque dans (44) l'inclusion est une égalité, la fonction d'inclusion est dite *minimale*. Si ce n'est pas le cas, elle est dite *pessimiste*. Il existe une infinité de fonctions d'inclusions pour une fonction donnée. Trouver la fonction d'inclusion minimale n'est pas toujours facile, mais bien souvent, il est possible de se contenter d'une fonction d'inclusion *convergente*, c'est-à-dire telle que $[f]([x])$ converge vers $f([x])$ lorsque la taille de l'intervalle $[x]$ tend vers zéro.

Plusieurs méthodes systématiques existent pour construire une fonction d'inclusion $[f](\cdot)$ d'une fonction $f(\cdot)$ donnée. La plus simple consiste à remplacer toutes les occurences de la variable réelle x par la variable intervalle correspondante $[x]$. La fonction d'inclusion ainsi obtenue est appelée fonction d'inclusion *naturelle*.

Exemple 3: Considérons de nouveau la fonction

$$f(x) = x^4 - 4x^2, \tag{45}$$

que l'on souhaite évaluer sur $[x] = [2, 4]$. Une première fonction d'inclusion naturelle est

$$[f]_1([x]) = [x]^4 - 4[x]^2. \tag{46}$$

On a alors, $[f]_1([2, 4]) = [-48, 240]$, qui contient bien l'ensemble des valeurs prises par $f(\cdot)$ sur $[2, 4]$. Le pessimisme de $[f]_1(\cdot)$ ne permet cependant pas de conclure directement que $[2, 4]$ ne contient pas de minimiseur global. En utilisant une seconde fonction d'inclusion naturelle définie, après factorisation de (45), par

$$[f]_2([x]) = [x]^2\left([x]^2 - 4\right), \tag{47}$$

il est possible d'obtenir $[f]_2([2, 4]) = [0, 192]$, qui correspond cette fois à l'ensemble des valeurs prises par $f(\cdot)$ sur $[2, 4]$. ♢

Bien souvent, une fonction d'inclusion naturelle n'est pas minimale, comme l'illustre l'exemple 3. Ceci est lié au fait que toutes les occurences de la variable intervalle sont traitées comme si elles étaient indépendantes les unes des autres. Ainsi, diminuer le nombre des occurences des variables dans l'expression formelle de $f(\cdot)$ permet de réduire le pessimisme de la fonction d'inclusion obtenue. Une autre solution consiste à utiliser des fonctions d'inclusion plus élaborées, telles que les formes centrées ou les formes de Taylor [7], [14].

L'extension de l'arithmétique d'intervalles et des fonctions d'inclusion aux vecteurs et aux matrices est immédiat. Un vecteur d'intervalles (ou *pavé*) et une matrice d'intervalles seront notés $[\mathbf{x}]$ et $[\mathbf{A}]$ respectivement.

B. Optimisation globale déterministe

Grâce à une fonction d'inclusion d'une fonction donnée, il est possible de prouver que celle-ci ne s'annule pas sur un intervalle ou qu'elle y reste strictement positive. Cette faculté est un élément clé de l'algorithme d'optimisation globale de Hansen [10], dont une version simplifiée est présentée maintenant.

Soit $J : \mathbb{R}^n \to \mathbb{R}$, une fonction coût deux fois continuement dérivable, dont on cherche l'ensemble $\mathbb{P}$ de *tous* les minimiseurs *globaux* appartenant à un pavé de recherche $[\mathbf{p}]_0 \subset \mathbb{R}^n$ donné. L'algorithme de Hansen va fournir une liste de pavés dont l'union contiendra $\mathbb{P}$ de manière garantie, à condition que le minimum global ne soit pas atteint sur la frontière de $[\mathbf{p}]_0$.

Schématiquement, l'algorithme de Hansen va couper $[\mathbf{p}]_0$ en sous-pavés et les placer dans une liste de pavés $\mathcal{L}$. Tout pavé de $\mathcal{L}$ pour lequel on peut démontrer qu'il ne contient pas de minimiseur global est éliminé de $\mathcal{L}$. Trois tests d'élimination sont présentés : le *test du point milieu*, le *test de monotonicité* et le *test de convexité*.

1. Supposons qu'un majorant $\widetilde{J}$ du minimum global de $J(\cdot)$ sur $[\mathbf{p}]_0$ soit disponible. Considérons un pavé $[\mathbf{p}] \in \mathcal{L}$ et posons $\left[\underline{j}, \overline{j}\right] = [J]([\mathbf{p}])$, où $[J](\cdot)$ est une fonction d'inclusion de $J(\cdot)$. Le *test du point milieu* vérifie si

   $$\underline{j} > \widetilde{J}, \tag{48}$$

   auquel cas $[\mathbf{p}]$ ne peut contenir de minimiseur global (voir l'exemple 2). Il peut donc être éliminé de $\mathcal{L}$.

2. Supposons qu'une fonction d'inclusion $[\mathbf{g}](\cdot)$ du gradient de $J(\cdot)$ soit disponible. Si pour un pavé $[\mathbf{p}] \in \mathcal{L}$, on a

   $$\mathbf{0} \notin [\mathbf{g}]([\mathbf{p}]), \tag{49}$$

   alors $J(\cdot)$ est strictement monotone sur $[\mathbf{p}]$. Le *test de monotonicité* ainsi défini permet alors d'éliminer $[\mathbf{p}]$ de $\mathcal{L}$.

3. Si $\mathbf{p}$ est un minimiseur global non contraint de $J(\cdot)$, alors $J(\cdot)$ doit être convexe dans un voisinage de $\mathbf{p}$. Ceci implique que le Hessien $\mathbf{H}(\cdot)$ de $J(\cdot)$ doit être défini non-négatif dans un voisinage de $\mathbf{p}$. Une condition nécessaire en est que les éléments diagonaux de $\mathbf{H}(\cdot)$ soient positifs ou nuls. Supposons donc que des fonctions d'inclusion $[\mathbf{H}_{ii}](\cdot)$, $i = 1 \ldots n$ des éléments diagonaux de $\mathbf{H}(\cdot)$ soient disponibles. Si pour un pavé donné $[\mathbf{p}] \in \mathcal{L}$, on a

   $$\exists i \in \{1 \ldots n\} \text{ tel que } [\mathbf{H}_{ii}]([\mathbf{p}]) \subset \mathbb{R}^*_-, \tag{50}$$

   alors $J(\cdot)$ ne peut être convexe sur $[\mathbf{p}]$ et $[\mathbf{p}]$ peut être éliminé de $\mathcal{L}$ par le *test de convexité* ainsi défini.

Pour mettre en œuvre le test du point milieu, à chaque itération de l'algorithme, une tentative d'amélioration du

majorant $\widetilde{J}$ du minimum global de $J(\cdot)$ sur $[\mathbf{p}]_0$ est effectuée, par exemple par une simple évaluation de $J(\cdot)$ au centre de $[\mathbf{p}]$ ou à l'aide d'une technique d'optimisation locale. Les tests de monotonicité et de convexité nécessitent des fonctions d'inclusion du gradient et des éléments diagonaux du Hessien de $J(\cdot)$. Ces quantités peuvent être obtenues soit grâce à une expression explicite, soit par différentiation automatique, voir [24], par exemple grâce à la technique du code adjoint, voir la section III.

Les pavés de la liste $\mathcal{L}$ sont classés par valeurs croissantes de la borne inférieure de l'évaluation intervalle du critère. Ceci permet d'éliminer rapidement de larges portions de la liste $\mathcal{L}$ dès qu'une amélioration de $\widetilde{J}$ est obtenue. Lorsqu'un pavé $[\mathbf{p}]$ résiste aux trois tests d'élimination, il est coupé en deux, à condition que sa taille soit supérieure à un seuil ε_1 et que la taille de l'évaluation intervalle de la fonction coût sur $[\mathbf{p}]$ soit supérieure à un seuil ε_2. Les deux parties résultantes sont alors replacées dans $\mathcal{L}$. L'algorithme s'arrête lorsqu'aucun pavé de $\mathcal{L}$ ne peut plus être coupé. Les pavés restant dans $\mathcal{L}$ sont tous susceptibles de contenir un minimiseur global de $J(\cdot)$ sur $[\mathbf{p}]_0$.

La version intervalle de l'algorithme de Newton-Gauss-Seidel [7] peut être utilisée pour réduire la taille des pavés $[\mathbf{p}]$ ayant résisté aux tests d'élimination. Cet algorithme peut également découper $[\mathbf{p}]$ en plusieurs parties de manière à isoler plusieurs minimiseurs globaux de $J(\cdot)$ inclus dans $[\mathbf{p}]$. Enfin, il permet, sous certaines conditions, de prouver l'*existence* et l'*unicité* d'un minimiseur global dans $[\mathbf{p}]$. Pour plus de détails, consulter [7]. D'autres techniques d'accélération sont disponibles, voir par exemple [22].

Contrairement aux méthodes d'optimisation classiques, il n'est pas nécessaire de fournir une initialisation $\widehat{\mathbf{p}}^0$. Ici, seul un pavé de recherche $[\mathbf{p}]_0$ doit être fourni. Ce pavé peut être très grand. De plus, quelles que soient les valeurs des paramètres ε_1 et ε_2, *aucun* pavé contenant un minimiseur global n'a pu être éliminé. La taille des pavés obtenus permet d'évaluer la précision avec laquelle les minimiseurs globaux ont été localisés. Toutes ces informations, que ne peuvent pas fournir des techniques d'optimisation classiques, justifient l'accroissement du temps de calcul nécessaire à une optimisation globale garantie.

La section qui suit montrera que l'analyse par intervalles permet également de revenir sur une idée communément admise en ce qui concerne la simulation des modèles décrits par des équations différentielles ordinaires.

V. On ne peut rien dire sur l'erreur globale de simulation d'un système d'EDO non linéaire

L'évolution de très nombreux dispositifs (mécaniques, chimiques...) peut être décrite par des modèles constitués d'un système d'équations différentielles ordinaires du type

$$\begin{cases} \mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{p}, t), \\ \mathbf{x}(0) = \mathbf{x}_0, \end{cases} \tag{51}$$

où $\mathbf{x}(t) \in \mathbb{R}^n$ est l'état du modèle et $\mathbf{p} \in \mathbb{R}^{n_\mathbf{p}}$ est le vecteur de ses paramètres.

L'estimation de $\mathbf{p}$, par exemple à l'aide de techniques de minimisation d'une fonction coût (voir la section I) requiert le calcul des solutions de (51) pour différentes valeurs de $\mathbf{p}$. Lorsque l'on ne dispose pas de la solution explicite de (51), il faut faire appel à des techniques numériques, voir par exemple [5]. La plupart ne permettent pas de caractériser de manière globale l'erreur commise lors de l'évaluation de $\mathbf{x}(t)$, qui peut être considérable. Le problème devient encore bien plus compliqué lorsque $\mathbf{x}(0)$ et $\mathbf{p}$ sont incertains et que l'on souhaite déterminer l'*ensemble* des solutions de (51) lorsque $\mathbf{x}(0) \in [\mathbf{x}_0]$ et $\mathbf{p} \in [\mathbf{p}_0]$. La solution classique consiste à discrétiser $[\mathbf{x}_0]$ et $[\mathbf{p}_0]$ et à calculer un faisceau de solutions, sans garantie que l'enveloppe de ce faisceau contienne toutes les solutions possibles.

A. Intégration numérique garantie

Pour ce problème, le calcul par intervalles est à nouveau capable de fournir, à chaque instant, des pavés contenant de manière garantie l'ensemble des solutions de (51), avec $\mathbf{x}(0) \in [\mathbf{x}_0]$. Il est même possible de prendre en compte un vecteur de paramètres $\mathbf{p}$ appartenant à un pavé $[\mathbf{p}]$. La principale hypothèse est que $\mathbf{f}$ est $k-1$ fois continuement différentiable sur un ouvert $\mathcal{D}$ de $\mathbb{R}^n$. Ceci permet d'obtenir un développement de Taylor d'ordre k de la solution avec évaluation intervalle du reste. La résolution se fait de manière récursive et en deux étapes. Pour en simplifier la présentation, nous partirons de $t = 0$.

1. Il s'agit dans un premier temps de prouver l'*existence* d'une solution entre $t = 0$ et $t = h$ et de calculer une *approximation extérieure* grossière de cette solution. Ainsi, il faut trouver $h > 0$ et $[\tilde{\mathbf{x}}_1] \subset \mathcal{D}$ tels que
$$\forall \mathbf{x}(0) \in [\mathbf{x}_0] \text{ et } \forall t \in [0, h], \mathbf{x}(t) \in [\tilde{\mathbf{x}}_1].$$
2. À partir de $[\tilde{\mathbf{x}}_1]$, il faut ensuite calculer une approximation plus fine de l'ensemble des $\mathbf{x}(h)$.

Plusieurs solutions ont été proposées, et un certain nombre d'outils d'intégration numérique garantie sont disponibles. Le plus ancien est AWA [16], écrit en Pascal. COSY [1], developpé en Fortran, et VNODE [21], écrit en C++, sont les outils les plus utilisés.

Ces techniques d'intégration se révèlent efficaces principalement lorsque $[\mathbf{x}_0]$ et $[\mathbf{p}]$ sont petits. Si ce n'est pas le cas, la taille des pavés obtenus a tendance à exploser, rendant de plus en plus difficile l'obtention d'une approximation extérieure grossière lors de la première étape de l'algorithme. Les théorèmes de Müller [20] permettent de résoudre en partie le problème précédent [32].

B. Théorèmes de Müller

L'idée est d'enfermer les solutions de (51) lorsque $\mathbf{x}_0 \in [\underline{\mathbf{x}}_0, \overline{\mathbf{x}}_0]$ et $\mathbf{p} \in [\underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0]$ entre les solutions de systèmes dynamiques ne faisant plus intervenir de quantités incertaines. Seule une reformulation du théorème de Müller dans le cas général est donnée ici. Pour le cas particulier des systèmes coopératifs, voir par exemple [15].

Théorème 1 (Existence) Soit une fonction $\mathbf{f}(\mathbf{x}, \mathbf{p}, t)$ continue sur un domaine

$$\mathbb{T} : \left\{ \begin{array}{l} \omega(t) \leqslant \mathbf{x} \leqslant \mathbf{\Omega}(t) \\ \underline{\mathbf{p}}_0 \leqslant \mathbf{p} \leqslant \overline{\mathbf{p}}_0 \\ 0 \leqslant t \leqslant h \end{array} \right. \tag{52}$$

où $\omega_i(t)$ et $\Omega_i(t)$, $i = 1 \ldots n$, sont continues sur $[0, h]$ et telles que

1. $\boldsymbol{\omega}(0) = \underline{\mathbf{x}}_0$ et $\mathbf{\Omega}(0) = \overline{\mathbf{x}}_0$,
2. pour $i = 1 \ldots n$,

$$D^{\pm}\omega_i(t) \leqslant \min_{\underline{\mathbb{T}}_i(t)} f_i(\mathbf{x}, \mathbf{p}, t) \tag{53}$$

et

$$D^{\pm}\Omega_i(t) \geqslant \max_{\overline{\mathbb{T}}_i(t)} f_i(\mathbf{x}, \mathbf{p}, t), \tag{54}$$

où $\underline{\mathbb{T}}_i(t)$ et $\overline{\mathbb{T}}_i(t)$ sont des sous-ensembles de $\mathbb{T}$ définis par

$$\underline{\mathbb{T}}_i(\tau) \quad : \quad \left\{ \begin{array}{l} x_i = \omega_i(t), \\ \omega_j(t) \leqslant x_j \leqslant \Omega_j(t), j \neq i, \\ \underline{\mathbf{p}}_0 \leqslant \mathbf{p} \leqslant \overline{\mathbf{p}}_0, \\ t = \tau, \end{array} \right. \tag{55}$$

$$\overline{\mathbb{T}}_i(\tau) \quad : \quad \left\{ \begin{array}{l} x_i = \Omega_i(t), \\ \omega_j(t) \leqslant x_j \leqslant \Omega_j(t), j \neq i, \\ \underline{\mathbf{p}}_0 \leqslant \mathbf{p} \leqslant \overline{\mathbf{p}}_0, \\ t = \tau. \end{array} \right. \tag{56}$$

Alors, pour tout $\mathbf{x}(0) \in [\underline{\mathbf{x}}_0, \overline{\mathbf{x}}_0]$ et $\mathbf{p} \in [\underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0]$, une solution de $\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{p}, t)$ existe dans

$$\mathbb{E} : \left\{ \begin{array}{l} 0 \leqslant t \leqslant h \\ \boldsymbol{\omega}(t) \leqslant \mathbf{x} \leqslant \mathbf{\Omega}(t) \end{array} \right. \tag{57}$$

et est égale à $\mathbf{x}(0)$ à $t = 0$. ♢

Théorème 2 (Unicité) De plus, si pour tout $\mathbf{p} \in [\underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0]$,

$$\mathbf{f}(\mathbf{x}, \mathbf{p}, t) \text{ est Lipschitz par rapport à } \mathbf{x} \text{ sur } \mathbb{D},$$

alors pour tout $\mathbf{x}(0) \in [\underline{\mathbf{x}}_0, \overline{\mathbf{x}}_0]$ et $\mathbf{p} \in [\underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0]$, la solution précédente est unique. ♢

Pour obtenir $\omega(t)$ et $\mathbf{\Omega}(t)$, il faut construire un système couplé

$$\left\{ \begin{array}{l} \underline{\mathbf{x}}' = \underline{\mathbf{g}}(\underline{\mathbf{x}}, \overline{\mathbf{x}}, \underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0, t), \underline{\mathbf{x}}(0) = \underline{\mathbf{x}}_0, \\ \overline{\mathbf{x}}' = \overline{\mathbf{g}}(\underline{\mathbf{x}}, \overline{\mathbf{x}}, \underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0, t), \overline{\mathbf{x}}(0) = \overline{\mathbf{x}}_0, \end{array} \right. \tag{58}$$

dont la solution $\left(\boldsymbol{\omega}^{\mathrm{T}}(t), \mathbf{\Omega}^{\mathrm{T}}(t)\right)^{\mathrm{T}}$ satisfait les conditions du théorème 1.

C. Estimation des paramètres d'un modèle de compartiments

Les propriétés de l'algorithme d'estimation vont être illustrées sur un modèle de compartiments [13]. Ce type de modèles, rencontré fréquemment en pharmacocinétique ou en chimie par exemple, est constitué d'un ensemble de réservoirs homogènes échangeant de la matière entre eux et avec le milieu extérieur.

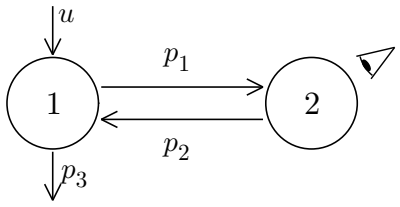

Fig. 3. Modèle à deux compartiments

Le modèle à deux compartiments représenté sur la figure 3 peut être décrit par l'équation d'état

$$\left\{ \begin{array}{l} x_1' = -(p_1 + p_3) x_1 + p_2 x_2, \\ x_2' = p_1 x_1 - p_2 x_2, \end{array} \right. \tag{59}$$

où x_1 et x_2 représentent les quantités de matière contenue dans les compartiments 1 et 2. Le vecteur des paramètres à déterminer est $\mathbf{p} = (p_1, p_2, p_3)^{\mathrm{T}}$. Il correspond aux constantes cinétiques (positives) d'échange de matière entre les compartiments et avec l'extérieur. Une injection unité de matière est effectuée dans le compartiment 1 à $t_0 = 0$, ainsi $\mathbf{x}(0) = (1, 0)^{\mathrm{T}}$. Des mesures $y(t_i)$ de la quantité de matière dans le compartiment 2 sont effectuées aux instants t_i, $i = 1 \ldots N$, ainsi $y(t_i) = x_2(t_i)$. L'estimation au sens des moindres carrés de $\mathbf{p}$ consiste à minimiser

$$J(\mathbf{p}) = \sum_{i=1}^{n_{\mathrm{t}}} \left(x_2(\mathbf{p}, t_i) - y(t_i)\right)^2. \tag{60}$$

Nous avons vu dans la section IV-B, que le gradient et les éléments diagonaux du hessien permettent de prouver que certains pavés ne contiennent pas de minimiseur global. Une solution pour obtenir le gradient voire les éléments diagonaux du hessien de la fonction coût passe par l'évaluation de fonctions de sensibilité, voir la section III. Notons s_{jk} la fonction de sensibilité du premier ordre de x_j par rapport à p_k

$$s_{jk}(\mathbf{p}, t) = \frac{\partial x_j}{\partial p_k}(\mathbf{p}, t). \tag{61}$$

Le gradient de J peut alors être évalué comme

$$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{p}}(\mathbf{p}) = \left(\begin{array}{c} 2 \sum_{i=1}^{n_{\mathrm{t}}} \left(x_2(\mathbf{p}, t_i) - y(t_i)\right) s_{21}(\mathbf{p}, t_i) \\ 2 \sum_{i=1}^{n_{\mathrm{t}}} \left(x_2(\mathbf{p}, t_i) - y(t_i)\right) s_{22}(\mathbf{p}, t_i) \\ 2 \sum_{i=1}^{n_{\mathrm{t}}} \left(x_2(\mathbf{p}, t_i) - y(t_i)\right) s_{23}(\mathbf{p}, t_i) \end{array} \right). \tag{62}$$

Les fonctions de sensibilité par rapport à un paramètre donné p_k satisfont un système d'équations différentielles obtenu en différentiant (59) par rapport à p_k. Il vient

$$\left\{ \begin{array}{l} s_{11}' = -(p_1 + p_3) s_{11} + p_2 s_{21} - x_1, \\ s_{21}' = p_1 s_{11} - p_2 s_{21} + x_1, \\ s_{12}' = -(p_1 + p_3) s_{12} + p_2 s_{22} + x_2, \\ s_{22}' = p_1 s_{12} - p_2 s_{22} - x_2, \\ s_{13}' = -(p_1 + p_3) s_{13} + p_2 s_{23} - x_1, \\ s_{23}' = p_1 s_{13} - p_2 s_{23}. \end{array} \right. \tag{63}$$

Comme $\mathbf{x}(0)$ ne dépend pas de $\mathbf{p}$, les conditions initiales pour chaque fonction de sensibilité sont nulles. Un système

de huit équations différentielles ordinaires permet ainsi de calculer l'évolution de l'état et des fonctions de sensibilité. L'état initial est ici supposé connu. Par contre, le vecteur des paramètres est contenu dans un pavé. Le théorème 1 est donc mis en œuvre pour trouver des systèmes dynamiques déterministes dont les solutions permettent d'enfermer celles du système dynamique incertain.

Comme les p_k et les x_j sont tous positifs, l'obtention de ces systèmes dynamiques bornants est relativement simple. Ainsi, l'obtention de bornes inférieures et supérieures de $\mathbf{x}$, s_{11} et s_{21} peut passer par la résolution des deux systèmes différentiels

$$\left\{\begin{array}{l} \underline{x}_1' = -(\overline{p}_1+\overline{p}_3)\underline{x}_1 + \underline{p}_2\underline{x}_2, \\ \underline{x}_2' = \underline{p}_1\underline{x}_1 - \overline{p}_2\underline{x}_2, \\ \overline{x}_1' = -(\underline{p}_1+\underline{p}_3)\overline{x}_1 + \overline{p}_2\overline{x}_2, \\ \overline{x}_2' = \overline{p}_1\overline{x}_1 - \underline{p}_2\overline{x}_2, \\ \underline{s}_{11}' = -(\overline{p}_1+\overline{p}_3)\underline{s}_{11} + \underline{p}_2\underline{s}_{21} - \overline{x}_1, \\ \underline{s}_{21}' = \underline{p}_1\underline{s}_{11} - \overline{p}_2\underline{s}_{21} + \underline{x}_1 \end{array}\right. \tag{64}$$

et

$$\left\{\begin{array}{l} \underline{x}_1' = -(\overline{p}_1+\overline{p}_3)\underline{x}_1 + \underline{p}_2\underline{x}_2, \\ \underline{x}_2' = \underline{p}_1\underline{x}_1 - \overline{p}_2\underline{x}_2, \\ \overline{x}_1' = -(\underline{p}_1+\underline{p}_3)\overline{x}_1 + \overline{p}_2\overline{x}_2, \\ \overline{x}_2' = \overline{p}_1\overline{x}_1 - \underline{p}_2\overline{x}_2, \\ \overline{s}_{11}' = -(\underline{p}_1+\underline{p}_3)\overline{s}_{11} + \overline{p}_2\overline{s}_{21} - \underline{x}_1, \\ \overline{s}_{21}' = \overline{p}_1\overline{s}_{11} - \underline{p}_2\overline{s}_{21} + \overline{x}_1. \end{array}\right. \tag{65}$$

Une procédure analogue peut être appliquée pour trouver les fonctions de sensibilité par rapport à p_2 et à p_3. Un système différentiel de seize équations serait alors obtenu. Cependant, comme la complexité des outils d'intégration numérique garantie est proportionnelle au cube du nombre des équations différentielles du système à résoudre, il est préférable d'exploiter le fait que ce système de seize équations différentielles peut être remplacé par six systèmes de six équations différentielles.

Une approche similaire pourrait être envisagée pour l'obtention des fonctions de sensibilité du second ordre qui permettent d'évaluer le hessien du critère.

Pour la partie expérimentale, un modèle à deux compartiments décrit par (59) avec $\mathbf{p}^* = (0.6, 0.15, 0.35)^{\mathrm{T}}$ a été simulé et des données ont été collectées en arrondissant $x_2(t_i)$ à deux décimales pour $t_i = i$, avec $i = 1 \ldots 15$.

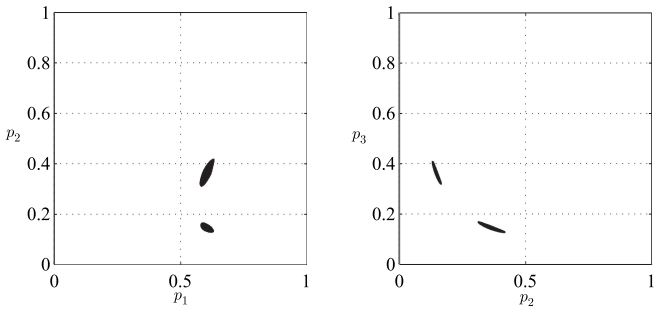

Fig. 4. Projection des ensembles solution obtenus

Le domaine de recherche initial choisi est $[\mathbf{p}]_0 = [0.01, 1]^{\times 3}$. Pour des seuils $\varepsilon_1 = \varepsilon_2 = 0.001$, une version très simplifiée de l'algorithme d'optimisation globale décrit à la section IV-B, ne mettant en œuvre ni le test de convexité ni l'algorithme de Newton-Gauss-Seidel, fournit une liste de pavés dont les projections sur les plans (p_1, p_2) et (p_2, p_3) sont représentées sur la figure 4. Seuls les pavés pour lesquels il n'a pas été possible de prouver qu'ils ne contiennent aucun minimiseur global sont représentés. Ces résultats ont été obtenus en 3 h sur un Pentium IV à 2 GHz. La symétrie de la projection sur le plan (p_2, p_3) de l'ensemble obtenu suggère un problème d'identifiabilité qui peut être confirmé par une étude théorique [34] : les valeurs des paramètres p_2 et p_3 peuvent être échangées sans modification de la sortie du modèle.

VI. Conclusions

L'identification est par nature une activité pluridisciplinaire, et les automaticiens identificateurs sont habitués à discuter avec leurs collègues des domaines d'application auxquels ils s'intéressent. Les quelques exemples mentionnés ici nous semblent illustrer l'intérêt qu'il y a à regarder aussi du côté des outils méthodologiques développés par d'autres communautés scientifiques. C'est ainsi que les statistiques robustes, la différentiation automatique, le calcul par intervalles et l'intégration garantie nous semblent de nature à modifier profondément notre approche de l'identification des systèmes. Au moins dans le cas de la différentiation automatique, il s'agit d'un juste retour des choses puisque le concept de code adjoint est directement inspiré de travaux de commande optimale déjà anciens.

Références

[1] M. Berz and K. Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4(4) :361–369, 1998.

[2] P. Bilardello, X. Joulia, J. M. Le Lann, H. Delmas, and B. Koehret. A general strategy for parameter estimation in differential-algebraic systems. *Computers and Chemical Engineering*, 17 :517–525, 1993.

[3] G. E. P. Box and N. R. Draper. Robust designs. *Biometrika*, 62(2) :347–352, 1975.

[4] R. W. Farebrother. *Statistical Data Analysis Based on the L_1-Norm and Related Methods*, chapter The historical developments of the L_1 and L_∞ estimation procedures 1793-1930, pages 37–63. North Holland, Amsterdam, 1987.

[5] C. W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice Hall, Upper Saddle River, N.J., 1971.

[6] R. Giering and Kaminski T. Recipes for adjoint code construction. *ACM Transactions on Mathematical Software (TOMS) archive*, 24(4) :437–474, 1998.

[7] R. Hammer, M. Hocks, U. Kulisch, and D. Ratz. *C++ Toolbox for Verified Computing*. Springer, Berlin, 1995.

[8] F. R. Hampel. A general qualitative definition of robustness. *Annals of Mathematical Statistics*, 42 :1887–1896, 1971.

[9] F. R. Hampel, E. N. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics. The Approach Based on Influence Functions*. Wiley, New-York, 1986.

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

[11] P. J. Huber. *Robust Statistics*. Wiley, New-York, 1981.

[12] IEEE Computer Society. IEEE standard for binary floating-point arithmetic. Technical Report IEEE Std. 754-1985, American National Standards Institute, 1985. Available at : http ://standards.ieee.org/.

[13] J. A. Jacquez. *Compartmental Analysis in Biology and Medicine*. Elsevier, Amsterdam, 1972.

[14] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer, London, 2001.

[15] M. Kieffer and E. Walter. Guaranteed parameter estimation for cooperative systems. In L. Benvenuti, A. De Santis, and L. Farina, editors, *Positive Systems – LNCIS 294*, pages 103–110, Berlin, 2003. Springer.

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

[17] H. M. Merrill and F. C. Schweppe. Bad data suppression in power system static state estimation. *IEEE Trans. on Power Apparatus and Systems*, 90(6) :2718–2725, 1971.

[18] L. Mili, M. G. Cheniae, N. S. Vichare, and P. J. Rousseeuv. Robust state estimation based on projection statistics. *IEEE Trans. on Power Systems*, 11(2) :1118–1127, 1996.

[19] L. Mili, V. Phaniraj, and P. J. Rousseeuw. Robust estimation theory for bad data diagnostics in electrical power systems. In C. T. Leondes, editor, *Control and Dynamic Systems : Advances in Theory and Applications*, pages 271–325, San Diego, 1990. Academic Press.

[20] M. Müller. Über das Fundamentaltheorem in der Theorie der gewöhnlichen Differentialgleichungen. *Math. Z.*, 26 :619–645, 1926.

[21] N. S. Nedialkov and K. R. Jackson. Methods for initial value problems for ordinary differential equations. In U. Kulisch, R. Lohner, and A. Facius, editors, *Perspectives on Enclosure Methods*, pages 219–264, Vienna, 2001. Springer-Verlag.

[22] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. In A. Iserles, editor, *Acta Numerica*, pages 271–369. Cambridge University Press, 2004.

[23] E. S. Pearson and C. Chandra Sekar. The efficiency of statistical tools and a criterion for the rejection of outlying observations. *Geometrika*, 28 :308–320, 1936.

[24] L. B. Rall. *Automatic Differentiation : Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany, 1981.

[25] P. J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388) :871–880, 1984.

[26] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. Wiley, New York, 1987.

[27] F. C. Schweppe and E. Handschin. Static state estimation in electric power systems. *Proceedings of the IEEE*, 62(7) :972–982, 1974.

[28] B. Speelpenning. *Compiling Fast Partial Derivatives of Functions Given by Algorithms*. PhD thesis, Dpt of Computer Science, University of Illinois at Urbana-Champaign, 1980.

[29] O. Talagran. The use of adjoint equations in numerical modelling of the atmospheric circulation. In A. Griewank and G. Corliss, editors, *Automatic Differentiation of Algorithms : Theory, Implementation, and Application*, pages 169–180, Philadelphia, 1991. SIAM.

[30] J. W. Tuckey. A survey of sampling from contaminated distributions. In I. Olkin, editor, *Contributions to Probability and Statistics*, pages 448–485, Stanford, 1960.

[31] P. Valko and S. Vajda. An extended ODE solver for sensitivity calculations. *Computers & Chemistry*, 8(43) :255–271, 1984.

[32] E. Walter and M. Kieffer. Guaranteed optimisation of the parameters of continuous-time knowledge-based models. In C. Commault and N. Marchand, editors, *Positive Systems – LNCIS 341*, pages 137–144, Heidelberg, 2006. Springer.

[33] E. Walter et L. Pronzato. *Identification des modèles paramétriques à partir de données expérimentales*. Masson, Paris, 1994.

[34] E. Walter and L. Pronzato. *Identification of Parametric Models from Experimental Data*. Springer, London, 1997.