CHAPITRE I

Modélisation de processus et estimation des paramètres d'un modèle

I. INTRODUCTION.

Dans la première partie de ce chapitre, nous rappelons les notions de processus et de modèle, ainsi que divers termes utilisés fréquemment dans le cadre de la modélisation. Dans la seconde partie, nous aborderons le problème de l'estimation des paramètres d'un modèle et nous présenterons les algorithmes qui ont été utilisés dans notre travail.

II. DÉFINITION D'UN PROCESSUS ET D'UN MODÈLE.

II.1 Processus.

Un processus est caractérisé par :

- une ou plusieurs grandeurs de sortie, mesurables, qui constituent le résultat du processus,
- une ou plusieurs grandeurs d'entrée (ou facteurs), qui peuvent être de deux types :
  - des entrées sur lesquelles il est possible d'agir (entrées de commande),
  - des entrées sur lesquelles il n'est pas possible d'agir (perturbations) ; ces dernières peuvent être aléatoires ou déterministes, mesurables ou non mesurables.

Les processus peuvent être de toutes natures : physique, chimique, biologique, écologique, financier, sociologique, etc.

II.2 Modèles.

II.2.1 Qu'est ce qu'un modèle ?

Nous nous intéressons ici aux modèles mathématiques, qui représentent les relations entre les entrées et les sorties du processus par des équations.
Si ces équations sont algébriques, le modèle est dit *statique*. Si ces équations sont des équations différentielles ou des équations aux différences récurrentes, le modèle est dit *dynamique*, respectivement *à temps continu* ou *à temps discret*.
Un modèle est caractérisé par son *domaine de validité*, c'est-à-dire par le domaine de l'espace des entrées dans lequel l'accord entre les valeurs des sorties du processus calculées par le modèle, et leurs valeurs mesurées, est considéré comme satisfaisant compte tenu de l'utilisation que l'on fait du modèle.

II.2.2 Buts d'une modélisation.

Un modèle peut être utilisé soit

- pour simuler un processus : à des fins pédagogiques, de détection d'anomalies de fonctionnement, de diagnostic de pannes, de conception assistée par ordinateur, etc.,
- pour effectuer la synthèse d'une loi de commande, ou pour être incorporé dans un dispositif de commande.

II.2.3 Classification des modèles.

II.2.3.1 Classification selon le mode de conception.

On distingue trois sortes de modèles en fonction des informations mises en jeu pour leur conception :

• **Les modèles de connaissance :** les modèles de connaissance sont construits à partir d'une analyse physique, chimique, biologique (ou autre suivant le type du processus), en appliquant soit les lois générales, fondées sur des principes (lois de la mécanique, de l'électromagnétisme, de la thermodynamique, de la physique quantique, etc.), soit les lois empiriques (finance, économie), qui régissent les phénomènes intervenant au sein des processus étudiés. Ces modèles ne comportent généralement pas de paramètres ajustables, ou des paramètres ajustables en très petit nombre.

Dans la pratique, il est toujours souhaitable d'établir un modèle de connaissance des processus que l'on étudie. Néanmoins, il arrive fréquemment que le processus soit trop complexe, ou que les phénomènes qui le régissent soient trop mal connus, pour qu'il soit possible d'établir un modèle de connaissance suffisamment précis pour l'application considérée. On est alors amené à concevoir des modèles purement empiriques, fondés exclusivement sur les résultats de mesures effectuées sur le processus.

• **Les modèles "boîte noire" :** les modèles "boîte noire" sont construits essentiellement sur la base de mesures effectuées sur les entrées et les sorties du processus à modéliser. La modélisation consiste alors à utiliser, pour représenter les relations entre les entrées et les sorties, des équations (algébriques, différentielles, ou récurrentes) paramétrées, et à estimer les paramètres, à partir des mesures disponibles, de manière à obtenir la meilleure précision possible avec le plus petit nombre possible de paramètres ajustables. Dans ce mémoire, nous désignerons fréquemment l'estimation des paramètres sous le terme *d'apprentissage*.

Le domaine de validité d'un tel modèle ne peut pas s'étendre au-delà du domaine des entrées qui est représenté dans les mesures utilisées pour l'apprentissage.

• **Les modèles "boîte grise" :** lorsque des connaissances, exprimables sous forme d'équations, sont disponibles, mais insuffisantes pour concevoir un modèle de connaissance satisfaisant, on peut avoir recours à une modélisation "boîte grise" (ou modélisation semi-physique) qui prend en considération à la fois les connaissances et les mesures. Une telle démarche peut concilier les avantages de l'intelligibilité d'un modèle de connaissance avec la souplesse d'un modèle comportant des paramètres ajustables.

II.2.3.2 Classification selon l'utilisation.

Indépendamment de la classification précédente, on peut distinguer deux types de modèles en fonction de l'utilisation qui en est faite.

• **Les modèles de simulation (ou simulateurs) :** un modèle de simulation est utilisé de manière indépendante du processus qu'il représente. Il doit donc posséder un comportement aussi semblable que possible à celui du processus. De tels modèles sont utilisés pour valider la conception d'un système avant sa fabrication (conception assistée par ordinateur en mécanique, en micro-électronique, ...), pour la formation de personnels (simulateurs de vols), pour la prévision à long terme, etc.
Du point de vue de la structure du modèle, les sorties passées, *mesurées sur le processus à modéliser*, ne peuvent constituer des entrées du modèle. L'estimation des paramètres et l'utilisation du modèle constituent deux phases successives et distinctes (apprentissage *non adaptatif*).

• **Les modèles de prédiction (ou prédicteurs) :** un modèle de prédiction est utilisé en parallèle avec le processus dont il est le modèle. Il prédit la sortie du processus à une échelle de temps courte devant les constantes de temps du processus. Les prédicteurs sont utilisés pour la synthèse de lois de commande, ou dans le système de commande lui-même (commande avec modèle interne).
Du point de vue de la structure du modèle, les sorties passées, mesurées sur le processus, peuvent constituer des entrées du modèle. L'estimation des paramètres et l'utilisation du modèle peuvent être effectuées simultanément si nécessaire (apprentissage *adaptatif*, utile notamment si les caractéristiques du processus dérivent dans le temps).

Ce mémoire présente la mise en oeuvre de plusieurs types de réseaux de fonctions paramétrées pour la modélisation dynamique de processus, et la comparaison de leurs performances respectives. Il s'agira donc exclusivement de modèles de type "boîte noire" qui peuvent être utilisés indifféremment comme simulateurs ou comme prédicteurs.

III. LES ÉTAPES DE LA CONCEPTION D'UN MODÈLE.

Lors de la conception d'un modèle de connaissance, la relation entre les entrées et la (ou les) sortie(s) du modèle découlent directement de la mise en équation des phénomènes physiques (chimiques, ou autres) qui régissent le fonctionnement du processus. Une fois le modèle obtenu sous forme analytique, des approximations peuvent être faites pour simplifier son expression (par exemple "linéariser" le modèle pour passer d'un modèle non linéaire à un modèle linéaire) si une telle approximation est justifiée.

Dans le cas d'une modélisation de type "boîte noire", la construction du modèle nécessite les trois élements suivants :

- Une hypothèse sur l'existence d'une relation déterministe liant les entrées à la (ou aux) sortie(s). Cette relation est caractérisée par une fonction appelée *fonction de régression* (ou plus simplement *régression)*. L'expression formelle supposée adéquate pour représenter cette relation est appelée *modèle-hypothèse*.
- Une séquence de mesures des entrées et de la sortie du processus.
- Un algorithme d'apprentissage.

Dans la suite de ce paragraphe, nous présentons les différents aspects qui doivent être pris en considération lors du choix d'un modèle-hypothèse.

III.1 Choix d'un modèle-hypothèse.

Les connaissances dont on dispose a priori sur le processus doivent guider le concepteur dans le choix de la modélisation la plus appropriée (statique ou dynamique, linéaire ou non linéaire, ...). L'élaboration du modèle-hypothèse nécessite d'effectuer les choix suivants :

• **Modèle statique ou dynamique** : lorsque l'on cherche à modéliser un processus physico-chimique ou biologique, il est généralement facile de savoir si l'application envisagée nécessite de modéliser la dynamique du processus (c'est-à-dire si l'on doit considérer une échelle de temps petite devant les constantes de temps du processus) ou si une modélisation statique suffit.

• **Modèle linéaire ou non linéaire :** il n'est pas douteux que la plupart des processus que l'on peut rencontrer nécessiteraient des modèles non linéaires s'il fallait les décrire de manière précise dans la totalité de leur domaine de fonctionnement : la plupart des modèles linéaires constituent des approximations valables dans un domaine plus ou moins restreint. Il est donc important de pouvoir élaborer un modèle non linéaire pour rendre compte du comportement d'un processus, non seulement autour de ses points de fonctionnement "habituels", mais également lors des passages d'un point de fonctionnement à un autre.

• **Modèle entrée-sortie ou modèle d'état** : dans le cas où l'on opte pour une modélisation dynamique, deux représentations sont possibles pour le modèle : il s'agit de la représentation d'état ou de la représentation entrée–sortie. L'état d'un processus est définit comme la quantité d'information minimale nécessaire pour prédire son comportement, étant données les entrées présentes et à venir. Il s'agit généralement d'un vecteur de grandeur égale à l'ordre du modèle. La représentation entrée–sortie est un cas particulier de la représentation d'état où le vecteur des états est constitué par la sortie et ses valeurs retardées dans le temps. Si le but de la modélisation est de prédire le comportement entrée–sortie du processus, il existe généralement une infinité de représentations d'état (au sens d'états ayant des trajectoires différentes) solutions du problèmes. En revanche, la représentation entrée–sortie est unique.

• **Présence de perturbations déterministes :** lorsque l'on cherche à réaliser un modèle dynamique, les perturbations déterministes peuvent être modélisées par une entrée supplémentaire (échelon, signal carré, sinusoïde). En particulier, si le modèle est construit pour la synthèse d'une loi de commande, la prise en considération de l'existence d'une perturbation pendant la phase de modélisation peut améliorer les performances de la commande pour le rejet de cette perturbation. Par exemple, il est proposé dans [Mukhopa93] une approche qui consiste à considérer la perturbation comme la sortie d'un processus. La modélisation de ce processus a pour effet d'introduire de nouvelles variables d'état, donc d'augmenter l'ordre du modèle.

• **Présence d'un bruit :** lorsque l'on cherche à réaliser un modèle dynamique, une perturbation de type "bruit" est modélisée par une séquence de variables aléatoires. Un bruit peut agir de différentes manières sur un processus. On distingue notamment le bruit de sortie (bruit additif qui affecte la mesure de la sortie du processus), et le bruit d'état (bruit additif qui affecte l'état du processus). Comme, en général, on ne connaît pas avec précision la nature du bruit qui

affecte le processus, on doit effectuer des hypothèses sur celle-ci ; on déduit de celles-ci la structure du modèle-hypothèse, et l'algorithme utilisé pour l'ajustement des paramètres. Une hypothèse erronée peut dégrader considérablement les performances du modèle. Ces problèmes ont été très largement étudiés dans le cas de la modélisation linéaire [Ljung87]. Dans le cadre de la modélisation non linéaire par réseaux de neurones, ces considérations sont développées dans [Nerrand94].

III.2 Du modèle-hypothèse au prédicteur ou au simulateur.

Un modèle-hypothèse ayant été choisi, l'étape suivante consiste à établir l'expression du prédicteur théorique, c'est-à-dire l'expression de la prédiction de la sortie du processus à l'instant $n+d$ en fonction des données disponibles à l'instant n (entrées et sorties du processus et/ou du prédicteur à l'instant n et aux instants antérieurs). Enfin, la dernière étape consiste à établir l'expression du prédicteur (ou du simulateur) proprement dit : dans le cas d'une modélisation "boîte noire", ce prédicteur utilise une fonction paramétrée, dont on estime les paramètres, à partir de mesures effectuées préalablement sur le processus, de telle manière qu'il constitue la meilleure approximation possible du prédicteur théorique. A l'issue de la procédure d'estimation des paramètres (apprentissage), il faut évaluer la performance du prédicteur (ou du simulateur).

Dans le cadre de ce mémoire nous nous intéressons plus particulièrement à l'étape d'apprentissage et donc aux caractéristiques du prédicteur (complexité, contraintes de mise en oeuvre) et aussi à l'algorithme d'apprentissage (efficacité, robustesse). La plupart des exemples étudiées étant des processus simulés, le problème du choix du modèle-hypothèse ne se pose pas. En revanche, la modélisation d'un processus réel (dans le dernier chapitre) sera l'occasion d'examiner ce problème.

III.3 Présentation de quelques modèles-hypothèses et de leurs prédicteurs associés.

Nous présentons dans ce paragraphe quelques exemples de modèles-hypothèses ainsi que les prédicteurs qui leurs sont associés, pour l'élaboration d'un modèle dynamique entrée-sortie. L'un des principaux paramètres qui interviennent dans le choix d'un modèle-hypothèse est la présence d'un bruit et la manière dont il agit sur le processus. Pour ceci, nous allons considérer deux classes de modèles-hypothèses : le modèle-hypothèse déterministe et des modèles-hypothèses non déterministe (faisant intervenir un bruit dans la modélisation du processus).

III.3.1 Modèle-hypothèse déterministe.

On considère qu'aucun bruit n'agit sur le processus. On propose un modèle-hypothèse déterministe ayant l'expression suivante :

$$y_p(n) = f\left(y_p(n\pm1), \ldots, y_p(n\pm N_s), u(n\pm1), \ldots, u(n\pm N_e)\right) \quad (1)$$

où $y_p(n)$ est la sortie mesurée du processus à l'instant n, N_s est l'ordre du modèle et N_e la mémoire sur l'entrée externe u. f est une fonction non linéaire dont on suppose qu'elle existe, et qu'elle constitue une représentation mathématique du comportement du processus.

La forme prédicteur théorique associée à ce modèle-hypothèse est la suivante :

$$y(n) = f\left(y_p(n\pm1), \ldots, y_p(n\pm N_s), u(n\pm1), \ldots, u(n\pm N_e)\right) \quad (2)$$

où $y(t)$ est la prédiction de la sortie du processus calculée par la forme prédicteur théorique. Étant donné que nous considérons que le processus n'est soumis à aucun bruit, la forme prédicteur théorique doit calculer à tout instant $y(t) = y_p(t)$.

Le prédicteur dont on effectuera l'apprentissage aura pour expression :

$$y(t) = \psi\left(y_p(t\pm1), \ldots, y_p(t\pm N_s), u(t\pm1), \ldots, u(t\pm N_e)\right) \quad (3)$$

où ψ est une fonction paramétrée, dont les paramètres doivent être estimés pour qu'elle approche au mieux la fonction f dans le domaine de fonctionnement considéré. Cette optimisation s'entend au sens de la minimisation de la fonction de coût empirique, que l'on appellera dorénavant fonction de coût et que l'on notera par J. Cette minimisation est réalisée à l'aide d'un algorithme d'apprentissage.

Si l'on est intéressé par la construction d'un modèle de simulation, un autre prédicteur peut être considéré :

$$y(n) = \psi\left(y(n\pm1), \ldots, y(n\pm N_s), u(n\pm1), \ldots, u(n\pm N_e)\right) \quad (4)$$

La seule différence avec la forme prédicteur de la relation (3) réside dans le fait que les entrées d'état du modèle sont les sorties retardées *du modèle*, non celles *du processus*.

III.3.2 Modèles-hypothèses non déterministes.

On désigne par "modèles-hypothèses non déterministes" des modèles-hypothèses qui supposent l'existence d'un bruit agissant sur le processus à modéliser. On peut envisager plusieurs hypothèses concernant la manière dont le bruit agit sur le processus. Nous en présentons deux, que nous considérerons lors de l'étude d'exemples dans ce mémoire.

III.3.2.1 L'hypothèse "Bruit de sortie".

L'hypothèse "Bruit de sortie" (Output Error en anglais) consiste à considérer qu'un bruit agit sur la sortie du processus. L'expression du modèle-hypothèse est :

$$\begin{cases} x(n) = f(x(n\pm1), \dots, x(n\pm N_s), u(n\pm1), \dots, u(n\pm N_e)) \\ y_p(n) = x(n) + w(n) \end{cases} \quad (5)$$

où $\{w(n)\}$ est une séquence de variables aléatoires indépendantes de moyenne nulle et de variance σ^2. La forme prédicteur théorique associée à ce modèle-hypothèse est donnée par l'expression suivante :

$$y(n) = f\left(y(n\pm1), y(n\pm2), \dots, y\left(n\pm N_s\right), u(n\pm1), \dots, u\left(n\pm N_e\right)\right) \quad (6)$$

Le prédicteur réel associé a pour expression :

$$y(n) = \psi\left(y(n\pm1), y(n\pm2), \dots, y\left(n\pm N_s\right), u(n\pm1), \dots, u\left(n\pm N_e\right)\right) \quad (7)$$

où ψ est une fonction réalisée à l'aide d'une fonction paramétrée, par exemple un réseau de neurones. C'est donc un modèle dont les entrées d'état sont ses propres sorties retardées, et non pas les sorties du processus. Si, après apprentissage, la fonction ψ était identique à la fonction f, l'erreur de prédiction commise par ce prédicteur serait une séquence aléatoire de mêmes caractéristiques que w. Lorsque la fonction parmétrée ψ est réalisée par un réseau de neurones, celui-ci est un réseau bouclé, que nous décrirons au paragraphe II.4.2 du chapitre suivant.

III.3.2.2 L'hypothèse "Bruit d'état".

L'hypothèse "Bruit d'état" (Equation Error en anglais) consiste à considérer qu'un bruit agit sur l'état du processus. Ce modèle-hypothèse a la forme suivante :

$$y_p(n) = f\left(y_p(n\pm1), y_p(n\pm2), \dots, y_p\left(n\pm N_s\right), u(n\pm1), \dots, u\left(n\pm N_e\right)\right) + w(n) \quad (8)$$

où $\{w(n)\}$ est une séquence de variables aléatoires indépendantes de moyenne nulle et de variance σ^2. La forme prédicteur théorique associée à ce modèle-hypothèse est donnée par l'expression suivante :

$$y(n) = f\left(y_p(n\pm1), y_p(n\pm2), \dots, y_p\left(n\pm N_s\right), u(n\pm1), \dots, u\left(n\pm N_e\right)\right) \quad (9)$$

Le prédicteur réel associé est de la forme :

$$y(n) = \psi\left(y_p(n\pm1), y_p(n\pm2), \dots, y_p\left(n\pm N_s\right), u(n\pm1), \dots, u\left(n\pm N_e\right)\right) \quad (10)$$

où ψ est une fonction paramétrée. Si ψ était identique à f, l'erreur de prédiction effectuée par ce prédicteur serait une séquence de variables aléatoires de mêmes caractéristiques que le bruit w. Lorsque la fonction parmétrée ψ est réalisée par un réseau de neurones, celui-ci est un réseau non bouclé, que nous décrirons au paragraphe II.4.1 du chapitre suivant.

IV. FONCTIONS PARAMÉTRÉES POUR LA MODÉLISATION "BOÎTE NOIRE".

Comme indiqué ci–dessus, une modélisation de type "boîte noire" est mise en œuvre dans le cas où l'on dispose de peu de connaissance sur le processus étudié, ou si le modèle de connaissance établi est trop compliqué pour être exploité. Dans les deux cas (et particulièrement dans le second) on a besoin d'un outil fournissant un modèle précis, aussi simple que possible en termes de nombre de paramètres ajustables et de nombre de calculs à effectuer, pour prédire la sortie du processus.

En général, un modèle "boîte noire" *statique* est une combinaison paramétrée de fonctions, qui peuvent être elles-mêmes paramétrées. Un modèle "boîte noire" *dynamique* est, comme nous l'avons vu ci-dessus, un ensemble d'équations différentielles (ou d'équations aux différences pour un modèle à temps discret) non linéaires, où la non-linéarité est réalisée, comme dans le cas d'un modèle statique, par une combinaison paramétrées de fonctions éventuellement paramétrées.

Des fonctions paramétrées constituent une famille *d'approximateurs universels* s'il est possible (sous certaines conditions de régularité) d'approcher toute fonction continue, avec la précision voulue, dans un domaine de l'espace des entrées, par une somme pondérée d'un nombre fini de ces fonctions.

Cette condition n'est néanmoins pas suffisante pour qu'une famille de fonctions soit utilisable de manière efficace pour la modélisation "boîte noire" efficace. En effet, parmi tous les modèles possibles, on recherche toujours celui qui possède le plus petit nombre de coefficients ajustables : c'est la propriété de *parcimonie,* dont nous verrons qu'elle n'est pas partagée par tous les types de fonctions paramétrées. A cet égard, il est important de distinguer les modèles *linéaires par rapport aux paramètres* des modèles *non linéaires par rapport aux paramètres.*

IV.1 Les fonctions paramétrées linéaires par rapport aux paramètres.

Une fonction paramétrée est linéaire par rapport aux paramètres si elle est de la forme :

$$\psi(X) = \sum_{i=1}^{N} \theta_i \, \Phi_i(X) \tag{11}$$

où les $\Phi_i(X)$ sont des fonctions non paramétrées d'une ou plusieurs variables groupées dans le vecteur X, et où les θ_i sont des paramètres.

Les fonctions $\Phi_i(X)$ peuvent être quelconques ; traditionnellement on utilise des monômes ; mais on peut également utiliser d'autres types de fonctions : fonctions splines, fonctions gaussiennes dont les centres et les écarts–types sont fixés,

fonctions ondelettes dont les translations et dilatations sont fixées (ces dernières seront présentées au chapitre IV de ce mémoire).

IV.2 Les fonctions paramétrées non linéaires par rapport aux paramètres.

Dans le présent travail, nous utiliserons essentiellement des fonctions non linéaires par rapport aux paramètres, qui sont de la forme

$$\psi(X) = \sum_{i=1}^{N} \theta_i \, \Phi_i(X, \Theta_i) \tag{12}$$

où Θ_i est un vecteur de paramètres de la fonction Φ_i. Ainsi, la fonction réalisée est linéaire par rapport aux θ_i, mais non linéaire par rapport aux paramètres constituant le vecteur Θ_i : c'est une combinaison linéaire de fonctions paramétrées.

Les réseaux de neurones à une couche cachée (présentés au chapitre II), les réseaux de fonctions gaussiennes radiales dont les centres et les écarts-types sont ajustables, les réseaux d'ondelettes (qui sont l'objet essentiel de ce travail) entrent dans cette catégorie de fonctions. Toutes ces fonctions sont des approximateurs universels [Hornik89] mais leur intérêt, par rapport aux fonctions linéaires par rapport aux paramètres, réside dans le caractère *parcimonieux* des modèles qu'ils permettent de réaliser [Hornik94]. Comme nous le verrons au paragraphe V.2, le prix à payer pour cela réside dans le fait que les méthodes habituelles d'estimation de paramètres (méthodes de moindres carrés) sont inutilisables, et que l'on doit avoir recours à des méthodes itératives (méthodes de gradient) dont la mise en œuvre est plus lourde.

Nous présentons brièvement ci-dessous ces trois types de réseaux, dont deux seront repris en détail dans les chapitres suivants.

IV.2.1 Les réseaux de neurones.

Dans ce travail, nous réserverons le terme de réseau de neurones aux réseaux de la forme (12) , où au moins une des fonctions $\Phi_i(X)$ est une fonction croissante bornée, notamment sigmoïde (tangente hyperbolique), d'une combinaison linéaire des entrées ; certaines de ces fonctions peuvent être l'identité. L'expression de ces réseaux est :

$$\psi(X) = \sum_{i=1}^{N} \theta_i \, \Phi_i(\Theta_i^T X) \tag{13}$$

Issus de travaux à connotation biologique dans les années 1940, ces réseaux sont maintenant considérés comme des outils mathématiques, indépendamment de toute référence à la biologie. Ils sont utilisés pour la modélisation et la commande

de processus non linéaires, ainsi que comme outils de classification, notamment pour la reconnaissance de formes.

Les principales étapes dans l'évolution de la théorie et de la pratique des réseaux de neurones ont été la mise au point d'un algorithme, économique en temps de calcul, pour l'évaluation du gradient de la fonction de coût (définie au paragraphe V), appelé *algorithme de rétropropagation* [Rumelhart86], et la preuve de ses propriétés d'approximateur universel [Hornik89] et de parcimonie [Barron93, Hornik94]. L'une des premières applications dans le domaine de la modélisation non linéaire de processus est présentée dans [Narendra90].

IV.2.2 Les réseaux de fonctions radiales (RBF pour Radial Basis Functions).

Les fonctions radiales ont été introduites par [Powell85] dans le cadre de *l'interpolation*, c'est-à-dire de la recherche de fonctions passant *exactement* par un nombre fini de points (dits points de collocation). Dans ce contexte, la fonction recherchée est une combinaison linéaire de fonctions de base, en nombre égal au nombre de points de collocation ; une fonction de base $\Phi_n(x)$, relative au point de collocation x_n, est dite radiale si elle ne dépend que de la distance du point courant x au point de collocation x_n. On peut utiliser diverses fonctions radiales, notamment des fonctions localisées (qui tendent vers zéro dans toutes les directions de l'espace des variables) telles que des gaussiennes centrées aux points de collocation. Bien entendu, la recherche d'une fonction passant exactement par les points n'a de sens que si ces points ne sont pas entachés de bruit.

La référence [Broom88] semble être parmi les premières à proposer l'idée d'utiliser des réseaux de RBF pour *l'approximation de fonctions* non linéaires. La fonction recherchée est toujours une combinaison linéaire de fonctions radiales, mais leur nombre est beaucoup plus petit que le nombre de points, et elles ne sont donc pas forcément centrées en ces points. Son expression est de la forme :

$$\psi(X) = \sum_{i=1}^{N} \theta_i \, \Phi_i\left(\left\| \left(X \pm M_i\right) \right\|, \sigma_i^2\right) \tag{14}$$

où M_ι est le vecteur des centres et σ_i^2 un scalaire (appelé variance dans le cas d'une RBF gaussienne).

La propriété d'approximateurs universels pour ces réseaux n'a été que récemment prouvée pour des gaussiennes radiales [Hartman90] et plus généralement pour des RBF [Park91].

Ces réseaux ont été utilisés comme outil de modélisation "boîte noire" dans le domaine de l'automatique. On les trouve à la base de modèles entrée–sortie [Chen90] et aussi de modèles d'état [Elanayar94]. Certaines spécificités de ces réseaux permettent de les utiliser pour la synthèse de lois de commande adaptatives stables [Behera95, Sanner92, Sanner95]. Le fait que ces réseaux

permettent de garantir la stabilité des correcteurs qu'ils réalisent les rend plus intéressants que les réseaux de neurones pour la résolution des problèmes de commande non linéaire. En revanche, cette propriété se fait au détriment de la parcimonie du réseau.

IV.2.3 Les réseaux d'ondelettes.

Les fonctions ondelettes trouvent leur origine dans des travaux de mathématiciens dès les années 1930. L'idée de départ était de construire une transformation, pour l'étude des signaux, plus commode que la transformation de Fourier, notamment pour des signaux de durée finie.

Les fonctions ondelettes ont subi une évolution au cours des années : celles dont nous disposons aujourd'hui sont plus complexes que leurs aînées, et possèdent des propriétés intéressantes pour l'approximation de fonctions. En particulier, elles possèdent la propriété d'approximateurs universels, ce qui suggère leur utilisation pour la construction de modèles "boîte noire".

La notion de réseaux d'ondelettes existe depuis peu [Pati 93] et l'étude de la propriété de parcimonie n'a pas été abordée. L'un des objectifs de ce mémoire est l'étude de la mise en oeuvre de cette classe de réseaux pour la modélisation entrée–sortie et d'état de processus, ainsi que la comparaison, sur des exemples, de la parcimonie et des performances de cette classe de réseaux par rapport à celle des réseaux de neurones (voir les chapitres III, IV et V).

V. ESTIMATION DES PARAMÈTRES D'UN MODÈLE.

V.1 Position du problème et notations.

Étant données les informations dont on dispose sur le processus (c'est à dire la séquence d'apprentissage) on détermine, dans une famille donnée de fonctions paramétrées $\psi(x, \theta)$ (où x est le vecteur regroupant toutes les entrées du modèle et θ le vecteur des paramètres inconnus de ψ) celle qui minimise une fonction de coût qui, le plus souvent, est la fonction de coût des moindres carrés.

Soit y_p^n la sortie du processus à l'instant n (dans le cas d'une modélisation dynamique), ou la valeur mesurée pour le $n^{\text{ème}}$ exemple de l'ensemble d'apprentissage (dans le cas d'une modélisation statique). De même, y^n est la sortie calculée par le modèle à l'instant n, *ou* pour le $n^{\text{ème}}$ exemple de l'ensemble d'apprentissage. On définit la fonction de coût des moindres carrés $J(\theta)$ par :

$$J(\theta) = \frac{1}{2} \sum_{n=1}^{N} \left(y_p^n \pm y^n \right)^2 \qquad (15)$$

où N est le nombre de mesures (taille de la séquence). $J(\theta)$dépend du vecteur des paramètres, ainsi que de la séquence d'apprentissage. Pour alléger les notations, nous n'indiquerons pas explicitement cette dernière dépendance dans la suite.
On définit l'erreur quadratique moyenne d'apprentissage (EQMA) comme une la moyenne de la fonction de coût calculée sur la séquence d'apprentissage. Elle est donnée par : $\frac{2J(\theta)}{N}$.

Lors de son exploitation, le modèle reçoit des entrées différentes de celles de la séquence d'apprentissage. On peut estimer ses performances en calculant diverses fonctions ; celle que l'on utilise le plus fréquemment est l'erreur quadratique moyenne de performance *EQMP* dont la valeur est calculée sur une séquence différente de celle utilisée pour l'apprentissage.

V.2 Les algorithmes de minimisation de la fonction de coût.

Dans le cas où le modèle est linéaire par rapport aux paramètres à ajuster, la minimisation de la fonction de coût, et donc l'estimation du vecteur des paramètres θ, peut se faire à l'aide la méthode des moindes carrés, qui ramène le problème à la résolution d'un système d'équations linéaires. Nous présentons cette technique dans ce qui suit.

V.2.1 Méthode des moindres carrés ordinaires.

Cette méthode est applicable pour l'apprentissage de modèles statiques ou de prédicteurs non bouclés dont la sortie est linéaire par rapport aux paramètres inconnus. Si cette sortie est linéaire par rapport aux entrées, le prédicteur associé a pour expression :

$$y(n) = \sum_{i=1}^{N_i} \theta_i x_i(n) \tag{16}$$

Ce modèle prédictif peut se être mis sous forme d'une équation matricielle. En effet, on peut l'écrire $Y = X\,\theta$ avec :

$$Y = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix}, X = \begin{bmatrix} x_1(1) & x_2(1) & \cdots & x_{N_i}(1) \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ x_1(N) & \cdots & \cdots & x_N(N) \end{bmatrix}, \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{N_i} \end{bmatrix} \tag{17}$$

L'estimation des paramètres est fondée sur la minimisation de la fonction de coût des moindres carrés (relation (15)). En utilisant la notation matricielle présentée ci–dessus, l'expression de la fonctionde coût $J(\theta)$ devient :

$$J(\theta) = \frac{1}{2}\left(Y_P^T Y_P \pm 2\theta^T X^T Y_p + \theta^T X^T X\theta\right) \tag{18}$$

La fonction de coût étant quadratique (par rapport au vecteur des paramètres à estimer), il atteint son minimum pour la valeur du vecteur des paramètres annulant sa dérivée. Soit θ_{mc} cette valeur du vecteur des paramètres. Elle vérifie :

$$\left.\frac{\partial J}{\partial \theta}\right|_{\theta_{mc}} = 0 \tag{19}$$

Cette dernière équation fournit l'équation normale :

$$\left[X^T X\right]\theta_{mc} = \left[X^T Y_p\right] \tag{20}$$

dont la solution θ_{mc} donnée par :

$$\theta_{mc} = \left[X^T X\right]^{\pm 1}\left[X^T Y_p\right] \tag{21}$$

est l'estimation des moindres carrés du vecteur des paramètres θ_p . Cette solution existe à condition que la matrice $X^T X$ soit inversible. Cette condition est généralement vérifiée lorsque N (le nombres d'exemples) est très grand devant N_i (le nombre d'entrées du modèle).

La méthode des moindres carrés peut être utilisée plus généralement pour l'estimation des paramètres de tout modèle dont la sortie est linéaire par rapport aux paramètres à estimer ; c'est le cas, par exemple, pour l'estimation des paramètres du modèle suivant :

$$y(n) = \sum_{i=1}^{N_i} \theta_i \Phi_i(X) \tag{22}$$

où les Φ_i sont des fonctions non paramétrées du vecteur des entrées X. Plusieurs choix sont possibles pour les fonctions Φ_i (voir paragraphe IV.1).

Les sorties des modèles "boîte noire" que nous utilisons dans ce mémoire ne sont pas linéaires par rapport aux paramètres à ajuster. Une résolution directe du problème comme dans le cas de la solution des moindres carrés n'est donc pas possible : on a donc recours à des algorithmes d'apprentissage qui recherchent une solution suivant une procédure itérative. Ces algorithmes sont généralement applicables sauf dans le cas où des restrictions sur les valeurs possibles pour les paramètres du modèle sont imposées par la nature des fonctions paramétrées utilisées (voir le paragraphe III.1 du chapitre IV).

Dans ce qui suit, nous allons présenter les algorithmes que nous utilisons dans ce mémoire pour la minimisation de la fonction de coût.

V.2.2 Principe des algorithmes de gradient.

Les algorithmes d'apprentissage fondés sur l'évaluation du gradient de la fonction de coût $J(\theta)$ par rapport aux paramètres procèdent à la minimisation de

manière itérative. $J(\theta)$ est une fonction scalaire à variable vectorielle (le vecteur θ des paramètres à ajuster). Son gradient est donc un vecteur défini par :

$$\nabla J = \begin{pmatrix} \dfrac{\partial J}{\partial \theta_1} \\ \vdots \\ \dfrac{\partial J}{\partial \theta_M} \end{pmatrix} \tag{23}$$

où M est le nombre de paramètres inconnus.
Le principe des algorithmes de gradient repose sur le fait qu'un minimum de la fonction de coût est atteint si sa dérivée (son gradient) est nul. Il existe plusieurs types d'algorithmes ; nous présenterons ceux que nous utiliserons dans la suite. Leur déroulement suit le schéma suivant :

A l'itération 0 : Initialiser le vecteur des paramètres à θ^0. Cette initialisation de θ peut avoir une grande influence sur l'issue de l'apprentissage. Nous porterons une attention particulière à cette étape. Nous proposons une technique d'initialisation pour réseaux d'ondelettes au chapitre IV.

A la k^{ème} itération : Calculer la fonction de coût et la norme du gradient avec le vecteur des paramètres courant (obtenu à l'itération précédente).

Si $J(\theta^{k-1}) \le J_{\max}$ ou $\|\nabla J\| \le \varepsilon$ ou $k = k_{max}$ (où J_{max} est une valeur maximale recherchée pour l'EQMA, ou pour l'EQMP si les performances sont évaluées pendant l'apprentissage),

Alors arrêter l'algorithme ; le vecteur $\theta^{k\pm1}$ est une solution,

Sinon calculer θ^k à partir de $\theta^{k\pm1}$ par la formule de mise à jour des paramètres suivante :

$$\theta^k = \theta^{k-1} + \mu_k d_k \tag{24}$$

où μ_k est un scalaire positif appelé pas du gradient et d_k un vecteur calculé à partir du gradient, appelé direction de descente. Les différences entre les méthodes de gradient résident dans le choix de la direction de descente et dans le choix du pas.

V.2.3 La méthode du gradient simple.

V.2.3.1 Présentation de la méthode.

La méthode du gradient simple consiste à la mise en œuvre de la formule de mise à jour des paramètres suivante :

$$\theta^k = \theta^{k-1} \pm \mu_k \nabla J\left(\theta^{k-1}\right) \tag{25}$$

La direction de descente est donc simplement l'opposée de celle du gradient ; c'est en effet la direction suivant laquelle la fonction de coût diminue le plus rapidement.

En pratique, la méthode du gradient simple peut être efficace lorsque l'on est loin du minimum de J . Quand on s'en approche, la norme du gradient diminue et donc l'algorithme progresse plus lentement. A ce moment, on peut utiliser une méthode de gradient plus efficace.

Un "réglage" du pas de gradient μ_k est nécessaire : en effet, une petite valeur de ce paramètre ralentit la progression de l'algorithme ; en revanche une grande valeur aboutit généralement à un phénomène d'oscillation autour de la solution. Diverses heuristiques, plus ou moins efficaces, ont été proposées.

V.2.3.2 Techniques de réglage du pas.

• **Technique du pas constant :** elle consiste à adopter un pas constant $\mu_k = \mu$ tout au long de l'algorithme. Elle est très simple mais peu efficace puisqu'elle ne prend pas en considération la décroissance de la norme du gradient.

• **Technique du pas asservi :** on peut asservir le pas à l'aide de la norme du gradient de sorte que le pas évolue en sens inverse de celle–ci. A chaque étape, le pas peut être calculé par :

$$\mu_k = \frac{\mu}{1 + \| \nabla J \|} \tag{26}$$

où μ est un paramètre constant. Lors de l'utilisation de cette technique, nous avons adopté la valeur $\mu = 10^{-3}$ qui s'est révélée très souvent satisfaisante. Le numérateur est augmenté du nombre 1 afin d'éviter une instabilité numérique au moment de la division dans le cas où la norme du gradient devient très proche du zéro. Cette technique offre un bon compromis du point de vue de la simplicité et de l'efficacité. C'est celle que nous avons utilisée chaque fois que nous avons mis en œuvre la méthode du gradient simple.

V.2.4 Les méthodes de gradient du second ordre.

Les méthodes que nous venons de décrire sont simples mais en général très inefficaces. On a donc systématiquement recours à l'utilisation de méthodes plus

performantes (pour une comparaison numérique entre ces méthodes, voir [Battiti92]). Elles sont dites du second ordre parce qu'elles prennent en considération la dérivée seconde de la fonction de coût. Nous présentons ci-dessous celles que nous avons mises en œuvre dans notre travail, et dont nous comparons les performances lors de l'étude de nos exemples.

V.2.4.1 L'algorithme de BFGS.

L'algorithme de BFGS (du nom de ses inventeurs : Broyden, Fletcher, Goldfarb et Shanno) [Minoux83] fait partie des méthodes d'optimisation dites "quasi–newtoniennes". Ces méthodes sont une généralisation de la méthode de Newton.

La méthode de Newton consiste à l'application de la règle suivante :

$$\theta^k = \theta^{k-1} \pm \left[H\left(\theta^{k-1}\right)\right]^{\pm 1} \nabla J\left(\theta^{k-1}\right) \tag{27}$$

où $H(\theta)$ est le Hessien de la fonction J calculé avec le vecteur des paramètres disponible à l'étape courante. La direction de descente est dans ce cas :

$$d_k = \pm \left[H\left(\theta^{k-1}\right)\right]^{\pm 1} \nabla J\left(\theta^{k-1}\right) \tag{28}$$

Le pas μ_k est constant et égal à 1.

Pour que le déplacement soit en sens contraire du gradient, il est indispensable que la matrice du Hessien soit définie positive. Sous cette condition, et si la fonction de coût est quadratique par rapport aux paramètres, la méthode de Newton converge vers l'unique solution en une seule itération.

En général, et pour les problèmes d'optimisation auxquels nous sommes confrontés dans ce mémoire, la fonction de coût n'est généralement pas quadratique. Elle peut néanmoins l'être localement, à proximité d'un minimum de ses minima. Donc, la méthode de Newton ne peut converger en une seule itération. De plus, cette méthode nécessite l'inversion de la matrice du Hessien à chaque itération (puisqu'il apparaît que plusieurs sont nécessaires), ce qui conduit à des calculs lourds.

L'algorithme de BFGS, ainsi que l'algorithme de Levenberg-Marquardt présenté dans le paragraphe suivant, sont des méthodes "quasi-newtoniennes" qui permettent de pallier ces inconvénients.

L'algorithme de BFGS est une règle d'ajustement des paramètres qui a l'expression suivante :

$$\theta^k = \theta^{k-1} \pm \mu_k M_k \nabla J\left(\theta^{k-1}\right) \tag{29}$$

où M_k est une approximation, calculée itérativement, de l'inverse de la matrice Hessienne. L'approximation de l'inverse du Hessien est modifiée à chaque itération suivant la règle suivante :

$$M_k = M_{k\pm1} + \left[1 + \left(\frac{\gamma_{k-1}^T M_{k-1}\gamma_{k-1}}{\delta_{k-1}^T\gamma_{k-1}}\right)\right]\frac{\delta_{k-1}^T\delta_{k-1}}{\delta_{k-1}^T\gamma_{k-1}} \pm \frac{\delta_{k-1}\gamma_{k-1}^T M_{k-1} + M_{k-1}\gamma_{k-1}\delta_{k-1}^T}{\delta_{k-1}^T\gamma_{k-1}} \qquad (30)$$

avec $\gamma_{k-1} = \nabla J(\theta^k) \pm \nabla J(\theta^{k-1})$ et $\delta_{k-1} = \theta^k \pm \theta^{k-}$. Nous prenons pour valeur initiale de M la matrice identité. Si, à une itération, la matrice calculée n'est pas définie positive, elle est réinitialisée à la matrice identité.

Reste la question du choix du pas μ_k. A cet effet, nous avons opté pour une méthode économique en calculs, la technique de Nash [Nash80]. Cette technique recherche un pas qui vérifie la condition de descente :

$$J(\theta^{k-1} + \mu_k d_k) \le J(\theta^{k-1}) + m_1 \mu_k d_k^T \nabla J(\theta^{k-1}) \qquad (31)$$

où m_1 est un facteur choisi très inférieur à 1 (par exemple $m_1 = 10^{-3}$).

En pratique, la recherche du pas se fait de manière itérative. On initialise μ_k à une valeur positive arbitraire. On teste la condition (31). Si elle est vérifiée, on accepte l'ajustement des paramètres. Sinon, on multiplie le pas par un facteur inférieur à 1 (par exemple 0.2) et on teste à nouveau la condition de descente. On répète cette procédure jusqu'à ce qu'une valeur satisfaisante du pas soit trouvée. Au bout de 22 essais, le pas atteint une valeur de l'ordre de 10^{-16}. On peut considérer alors qu'il n'est pas possible de trouver un pas satisfaisant.

Une méthode "quasi–newtonienne", n'est efficace que si elle est appliquée au voisinage d'un minimum. D'autre part, la règle du gradient simple est efficace lorsqu'on est loin du minimum et sa convergence ralentit considérablement lorsque la norme du gradient diminue (c'est à dire lorsqu'on s'approche du minimum). Ces deux techniques sont donc complémentaires. De ce fait, l'optimisation s'effectue en deux étapes : utilisation de la règle du gradient simple pour approcher un minimum, et de l'algorithme de BFGS pour l'atteindre. Le critère d'arrêt est alors un des critères décrits au paragraphe V.2.2.

V.2.4.2 L'algorithme de Levenberg–Marquardt.

L'algorithme de Levenberg–Marquardt [Levenberg44, Marquardt63] repose sur l'application de la formule de mise à jour des paramètres suivante :

$$\theta^k = \theta^{k-1} \pm \left[H(\theta^{k-1}) + \mu_k I\right]^{\pm1} \nabla J(\theta^{k-1}) \qquad (32)$$

où $H(\theta^{k-1})$ est le Hessien de la fonction de coût et μ_k est le pas. Pour de petites valeurs du pas, la méthode de Levenberg–Marquardt s'approche de celle de Newton. Inversement, pour de grandes valeurs de μ_k, l'algorithme Levenberg–Marquardt est équivalent à l'application de la règle du gradient simple avec un pas de $\frac{1}{\mu_k}$.

La première question relative à cet algorithme est celle de l'inversion de la matrice $\left[H\left(\theta^{k-1}\right)+\mu_k I\right]$. L'expression exacte du Hessien de la fonction J est :

$$H\left(\theta^k\right)=\sum_{n=1}^{N}\left(\frac{\partial e^n}{\partial \theta^k}\right)\left(\frac{\partial e^n}{\partial \theta^k}\right)^T+\sum_{n=1}^{N}\frac{\partial^2 e^n}{\partial \theta^k\,\partial\left(\theta^k\right)^T}e^n \tag{33}$$

avec $e^n=y_p^n\pm y^n$.

Le second terme de l'expression étant proportionnel à l'erreur, il est permis de le négliger en première approximation, ce qui fournit une expression approchée :

$$\tilde{H}\left(\theta^k\right)=\sum_{n=1}^{N}\left(\frac{\partial e^n}{\partial \theta^k}\right)\left(\frac{\partial e^n}{\partial \theta^k}\right)^T=\sum_{n=1}^{N}\left(\frac{\partial y^n}{\partial \theta^k}\right)\left(\frac{\partial y^n}{\partial \theta^k}\right)^T \tag{34}$$

Dans le cas d'un modèle linéaire par rapport aux paramètres, c'est à dire si y est une fonction linéaire de θ, le second terme de l'expression de H est nul est l'approximation devient exacte.

Plusieurs techniques sont envisageables pour l'inversion de la matrice $\left[\tilde{H}+\mu_k I\right]$.

• **Inversion indirecte.**

Un lemme d'inversion permet de calculer la matrice inverse suivant une loi récurrente. En effet, soient *A*, *B*, *C* et *D* quatre matrices. On a la relation suivante : $\left(A+B\,C\,D\right)^{-1}=A^{-1}\pm A^{-1}B\left(C^{-1}+D\,A^{-1}\,B\right)^{-1}D\,A^{-1}$.

D'autre part, en posant $X^n=\dfrac{\partial y^n}{\partial \theta^k}$, l'approximation de la matrice H peut être calculée à partir de la loi de récurrence suivante :

$$\tilde{H}^n=\tilde{H}^{n-1}+X^n\left(X^n\right)^t \quad \text{avec} \quad n=1,\ldots,N$$

De ce fait, on a $\tilde{H}=\tilde{H}^N$.

Si l'on applique le lemme d'inversion à la relation précédente en choisissant $A=\tilde{H}$, $B=X^n$, $C=I$ et $D=\left(X^n\right)^T$, on obtient la relation suivante :

$$\left(\tilde{H}^n\right)^{-1}=\left(\tilde{H}^{n-1}\right)^{-1}\pm\frac{\left(\tilde{H}^{n-1}\right)^{-1}X^n\left(X^n\right)^T\left(\tilde{H}^{n-1}\right)^{-1}}{1+\left(X^n\right)^T\left(\tilde{H}^{n-1}\right)^{-1}X^n} \tag{35}$$

En prenant, à la première étape (n = 1), $\tilde{H}^0=\mu_k I$, on obtient, à l'étape N : $\left(\tilde{H}^N\right)^{-1}=\left[\tilde{H}+\mu_k I\right]^{-1}$.

• **Inversion directe.**

Plusieurs méthodes d'inversion directes existent. Étant donné que l'algorithme est itératif et que la procédure de recherche du pas nécessite souvent plusieurs inversions de matrice, on a intérêt à utiliser une méthode économique en nombre de calculs.

Le fait que l'approximation du Hessien augmentée de μ_k reste une matrice symétrique définie positive nous permet d'utiliser la méthode de Cholesky.

De la même façon que dans le cas de l'algorithme de BFGS, une recherche unidimensionnelle doit être appliquée pour la recherche d'un pas de descente et ceci à chaque itération de l'algorithme. Une stratégie communément utilisée [Bishop95, Walter94] consiste à appliquer la procédure suivante : soit $r > 1$ (généralement égal à 10) un facteur d'échelle pour μ_k. Au début de l'algorithme, on initialise μ_0 à une grande valeur ([Bishop95] propose 0.1). A l'étape k de l'algorithme :

- Calculer $J(\theta^k)$ avec μ_k déterminé à l'étape précédente.
- **Si** $J(\theta^k) < J(\theta^{k-1})$, **alors** accepter le changement de paramètres et diviser μ_k par r.
- **Sinon**, récupérer θ^{k-1} et multiplier μ_k par r. Répéter cette dernière étape jusqu'à ce qu'une valeur de μ_k correspondant à une décroissance de J soit trouvée.

Cet exemple de procédure présente l'avantage de nécessiter peu d'inversions de matrice à chaque itération de l'algorithme. En revanche, le choix du pas initial possède une influence sur la vitesse de convergence de l'algorithme.
Ces observations nous mènent à proposer la procédure suivante :

Au début de l'algorithme, initialiser μ_0 à une valeur positive quelconque. En effet ce choix n'a pas d'influence sur le déroulement de l'algorithme. A l'étape k de l'algorithme :

1. Calculer $J(\theta^k)$ avec le μ_k disponible (le dernier calculé).
2. Si $J(\theta^k) < J(\theta^{k-1})$, **alors** récupérer θ^{k-1}, diviser μ_k par r et aller à l'étape 1.
3. Sinon récupérer θ^{k-1} et multiplier μ_k par r. Répéter cette dernière étape jusqu'à ce qu'une valeur de μ_k correspondant à une décroissance de J soit trouvée.

Cette procédure permet de s'approcher de la méthode de Newton plus rapidement que la méthode précédente. En revanche, étant donné que plusieurs ajustements de paramètres sont testés, elle nécessite unn plus grand nombre d'inversions de matrice.

V.3 Commentaire.

Nous avons présenté dans cette partie les algorithmes du second ordre que nous utilisons dans ce mémoire (c'est à dire l'algorithme de BFGS et celui de Levenberg–Marquardt). La difficulté essentielle lors de l'application de l'algorithme de BFGS réside dans le choix de la condition de passage du gradient simple à la méthode de BFGS. Ce problème ne se pose pas pour l'algorithme de Levenberg–Marquardt, mais le volume de calculs nécessaires à chaque itération de cet algorithme croît rapidement avec le nombre de paramètres.

VI. CONCLUSION

Dans ce chapitre, nous avons présenté les principes de la modélisation "boîte noire", les étapes de la conception d'un tel modèles, ainsi que les fonctions paramétrées utilisables, et les algorithmes qu'il convient de mettre en œuvre pour l'ajustement des paramètres. Les deux chapitres suivants seront consacrés à la présentation et à la mise en œuvre des deux catégories de fonctions paramétrées que nous avons utilisées : les réseaux de neurones et les réseaux d'ondelettes.